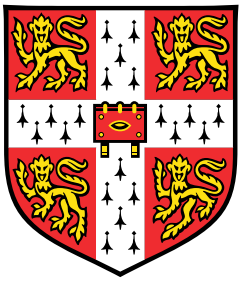
Advances in Probabilistic Meta-Learning and the Neural Process Family



Jonathan Gordon

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Peterhouse College

November 2020

DECLARATION

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words and has less than 150 figures.

Cambridge, November 2020

ABSTRACT

A natural progression in machine learning research is to automate and learn from data increasingly many components of our learning agents. *Meta-learning* is a paradigm that fully embraces this perspective, and can be intuitively described as embodying the idea of *learning to learn*. A goal of meta-learning research is the development of models to assist users in navigating the intricate space of design choices associated with specifying machine learning solutions. This space is particularly formidable when considering deep learning approaches, which involve myriad design choices interacting in complex fashions to affect the performance of the resulting agents. Despite the impressive successes of deep learning in recent years, this challenge remains a significant bottleneck in deploying neural network based solutions in several important application domains. But how can we reason about and design solutions to this daunting task?

This thesis is concerned with a particular perspective for meta-learning in supervised settings. We view supervised learning algorithms as *mappings* that take data sets to predictive models, and consider meta-learning as learning to approximate functions of this form. In particular, we are interested in meta-learners that (i) employ neural networks to approximate these functions in an end-to-end manner, and (ii) provide *predictive distributions* rather than single predictors. The former is motivated by the success of neural networks as function approximators, and the latter by our interest in the *few-shot* learning scenario. The introductory chapters of this thesis formalise this notion, and use it to provide a tutorial introducing the *Neural Process Family* (NPF), a class of models introduced by Garnelo et al. (2018a,b) satisfying the above-mentioned modelling desiderata. We then present our own technical contributions to the NPF.

First, we focus on fundamental properties of the model-class, such as expressivity and limiting behaviours of the associated training procedures. Next, we study the role of *translation equivariance* in the NPF. Considering the intimate relationship between the NPF and the representation of functions operating on *sets*, we extend the underlying theory of *DeepSets* to include translation equivariance. We then develop novel members of the NPF endowed with this important inductive bias. Through extensive empirical evaluation, we demonstrate that, in many settings, they significantly outperform their non-equivariant counterparts.

Finally, we turn our attention to the development of Neural Processes for few-shot *image-classification*. We introduce models that navigate the important tradeoffs associated with this setting, and describe the specification of their central components. We demonstrate that the resulting models—CNAPs—achieve state-of-the-art performance on a challenging benchmark called Meta-Dataset, while adapting faster and with less computational overhead than their best-performing competitors.

Traveller, your footprints are the path and nothing more;
traveller, there is no path, the path is made by walking.
— Anotonio Machado

ACKNOWLEDGEMENTS

I was extremely fortunate to have had several amazing and influential mentors throughout my PhD. I would like to begin by thanking my supervisor, José Miguel Hernández-Lobato. Miguel provided guidance, wisdom, and encouragement that were essential to the successful completion of the process, and helped me develop as a researcher. I would also like to acknowledge and express my gratitude to my advisor, Richard Turner, with whom I worked closely throughout my PhD. The mentorship, support, feedback, and inspiration Rich provided have been of immeasurable value. Perhaps more importantly, he has set a prime example of what it means to be a researcher and scientist that will accompany me well beyond my PhD. Finally, Sebastian Nowozin provided additional guidance and inspiration that I was lucky to stumble upon. Sebastian was an endless source of creativity and inspiration, and he brought an unparalleled energy and excitement to our projects.

It has been one of the greatest pleasures and honours of my life to be a member of the Cambridge Machine Learning Group, not least because of the incredibly talented lab members. I am grateful to them all for having created an inspiring and supportive environment, and making the lab feel like a home. I'd like to thank John Bronskill and Matthias Bauer, who were there from my very first projects, and helped me take my initial (somewhat clumsy) steps towards conducting research. We spent many hours side-by-side in the lab, and I enjoyed every minute of it. I'd also like to thank Eric Nalisnick. I benefited incredibly from an additional, more experienced collaborator who inspired confidence and clarity of thought, and I enjoyed our conversations very much.

I'd like to thank my co-authors Wessel Bruinsma, Andrew Y. K. Foong, Robert Pinsler, James Requeima, Marton Havasi, and Yann Dubois, all of whom are brilliant collaborators. They made conducting research a joy, and I am immensely proud of the work we did together. Additionally, I'd like to thank Will Tebbutt, Siddharth Swaroop, and Adrià Garriga-Alonso for hours upon hours in the lab, providing deep, insightful, inspiring, and often hilarious conversations. It would not have been nearly as fun or meaningful without them.

Finally, thank you Roni. Your support, encouragement, and love have made this thesis possible.

Cambridge, November 2020

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

This thesis is primarily concerned with the learning paradigm known as *meta-learning* (Schmidhuber, 1987; Thrun and Pratt, 2012). Of particular interest is a class of models—the Neural Process Family—which brings together two key ideas: (i) the construction of meta-learners as *function approximators* parametrised by *deep neural networks*, and (ii) incorporating estimates of *uncertainty* around predictions made by the meta-learning models. Throughout this thesis, we will develop these ideas, and present our own extensions to the Neural Process Family, which focus on meta-learning for *few-shot* classification and regression.

1.1 MOTIVATION

A significant challenge of modern machine learning is specifying the *learning pipeline* itself. For example, modern supervised learning tasks often employ deep learning solutions (LeCun et al., 2015; Goodfellow et al., 2016), which have witnessed significant and important successes in recent years. However, invoking such solutions requires human experts to specify myriad design choices in approaching a new problem, such as a neural network architecture, loss function, optimiser and learning rate schedule, regularisation and normalisation strategy, and many more. Moreover, the design of the pipeline can often encode important inductive biases, and have significant effect on the resulting performance. Despite enormous research effort in designing efficient pipelines in recent years, this challenge remains an important bottleneck in making progress with machine learning research. *Meta-learning* provides a framework concerned with automating and learning the machine learning pipeline itself directly from the data. It aims to assist experts in navigating these choices, thus scaling up our ability to discover pipelines that are useful for challenging scenarios.

But how might we formalise and reason about models that approach such a task? In this thesis, we assume the following perspective for meta-learning in supervised settings. We view the supervised learning pipeline itself as a map that takes data sets to predictive functions. Then, we think of meta-learning as parametrising—and learning from data—a function approximator of the appropriate form. In particular, we will be interested in a class of meta-learners that (i) employ deep neural networks in the approximation of such functions, and (ii) model distributions over functions rather than just single predictors. The

former is motivated by the enormous success of deep learning in supervised function approximation in recent years. The latter is motivated by our central application of interest: *few-shot learning*, where uncertainty is of crucial importance.

1.2 OVERVIEW AND MAIN CONTRIBUTIONS

The majority of the results discussed in this thesis were presented in a series of publications. In particular, the thesis mainly draws from Requeima et al. (2019), Gordon et al. (2020a) and Foong et al. (2020). Below, we provide an overview of the structure of the thesis, and the main contributions described in each chapter.

General Contributions to the NPF

Chapter 2 presents our perspective of meta-learning as learning to approximate maps from data sets to predictive stochastic processes, provides a systematic overview of the NPF, and concludes with several technical contributions. Specifically, in Section 2.7.1, we study the implications of maximum-likelihood training for members of the NPF in idealised settings, and provide a framework for reasoning about the guarantees of such procedures. In Section 2.7.3 we leverage these insights and propose a novel training procedure memmbers of the NPF that employ latent variables, which, in later chapters, we demonstrate generally improves performance of these models. The results in these two sections were derived in collaboration with Andrew Y. K. Foong and Wessel Bruinsma under the supervision of Richard E. Turner, and were published by Foong et al. (2020). In Section 2.7.2, we turn our attention to the theory of machine learning on sets, extending the results of Zaheer et al. (2017) to include vector-valued sets of varying sizes. We then leverage this result to provide a representation theorem characterising the expressive power of models in the NPF. The central results here are Theorems 2.3 and 2.4, which were developed and proved (Appendix C) in collaboration with Andrew Y. K. Foong and Wessel Bruinsma. These results are not published elsewhere.

Convolutional DeepSets

In Chapter 3, we introduce the Convolutional DeepSets framework, which further extends the work of Zaheer et al. (2017) to include translation equivariant functions on sets. The central contribution is the introduction of the ConvDeepSets form, paired with a universal representation theorem for functions satisfying the desired properties (Theorem 3.1). The framework and proof were originally co-developed with Wessel Bruinsma and Richard E. Turner, and later improved upon and verified by Andrew Y. K. Foong. Richard E. Turner also supervised and guided

the development of ConvDeepSets throughout. We acknowledge Mark Rowland for further verifying and suggesting improvements to the proof (provided in Appendix D). These results were published by Gordon et al. (2020a).

Translation Equivariant Members of the NPF

In Chapters 4 and 5, we propose models that extend the NPF to include translation equivariance. The main contributions of these chapters are the ConvCNP and ConvNP – novel members of the NPF. The models are empircally evaluated via extensive experimentation to demonstrate their usefulness. These contributions were produced in collaboration with my co-first authors Wessel Bruinsma and Andrew Y. K. Foong, as well as Yann Dubois and James Requeima, who assisted in conceptualising parts of the framework, writing the software and conducting experiments, and writing and editing the papers. The entire development of these models was carried out under the close supervision and guidance of Richard E. Turner. These results were published by Gordon et al. (2020a) and Foong et al. (2020).

Conditional NPF Models for Few-shot Classification

Finally, in Chapter 6, we introduce a novel member of the conditional NPF for few-shot classification. The central contribution of this chapter is the introduction of CNAPs, a member of the NPF that achieves state-of-the-art performance on the challenging few-shot learning benchmark Meta-Dataset. The model was conceptualised and developed with my co-first authors James Requeima and John Bronskill, under the close supervision and guidance of Sebastian Nowozin and Richard E. Turner. The results were published by Requeima et al. (2019).

1.3 LIST OF PUBLICATIONS

The following is a list of publications and software I co-authored while pursuing my PhD, regardless of whether they appear in the chapters of this thesis.

Peer-reviewed Conference proceedings

John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard Turner (2020). 'TaskNorm: Rethinking Batch Normalization for Meta-Learning'. In: *Proceedings of the 37th International Conference on Machine Learning*.

Andrew Y. K. Foong, Wessel P. Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard E. Turner (2020). 'Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes'. In: *Advances in Neural Information Processing Systems 33* (cited on pages 2, 3, 17, 65).

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner (2019). 'Meta-Learning Probabilistic Inference for Prediction'. In: *International Conference on Learning Representations* (cited on pages 17, 80, 84, 88, 89).

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner (2020a). 'Convolutional Conditional Neural Processes'. In: *International Conference on Learning Representations* (cited on pages 2, 3, 17, 25, 39, 46, 51).

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt (2020b). 'Permutation Equivariant Models for Compositional Generalization in Language'. In: *International Conference on Learning Representations*.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato (2019). 'Bayesian Batch Active Learning as Sparse Subset Approximation'. In: *Advances in Neural Information Processing Systems 32*.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner (2019). 'Fast and Flexible Multi-Task Classification using Conditional Neural Adaptive Processes'. In: *Advances in Neural Information Processing Systems 32* (cited on pages 2, 3, 17, 79).

Peer-reviewed Journals

Jonathan Gordon and José Miguel Hernández-Lobato (2019). 'Combining Deep Generative and Discriminative Models for Bayesian Semi-Supervised Learning'. *Pattern Recognition*.

Preprints and unpublished work

Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi (2019). 'Probabilistic Neural Architecture Search'. *arXiv preprint arXiv:1902.05116* (cited on page 37).

Eric Nalisnick, Jonathan Gordon, and José Miguel Hernández-Lobato (2020). 'Predictive Complexity Priors'. *arXiv preprint arXiv:2006.10801*.

Peer-reviewed workshop proceedings

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner (2018a). 'Consolidating the Meta-Learning Zoo: A Unifying Perspective as Posterior Predictive Inference'. In: *MetaLearning Workshop, NeurIPS 2018*.

— (2018b). 'Versa: Versatile and Efficient Few-Shot Learning'. In: *Bayesian Deep Learning Workshop, NeurIPS 2018*.

Jonathan Gordon and José Miguel Hernández-Lobato (2017). 'Bayesian Semi-Supervised Learning with Deep Generative Models'. In: *ICML Workshop on Principled Approaches to Deep Learning*.

Marton Havasi, Jasper Snoek, Dustin Tran, Jonathan Gordon, and José Miguel Hernández-Lobato (2020). 'Refining the Variational Posterior Through Iterative Optimization'. In: *Bayesian Deep Learning Workshop, NeurIPS 2020*.

1.4 CO-AUTHORED SOFTWARE

VERSA: John Bronskill, Jonathan Gordon, and Matthias Bauer (2019). Code for "Meta-learning probabilistic inference for prediction". https://github.com/Gordonjo/versa

CNAPs: John Bronskill, Jonathan Gordon, and James Requeima (2019). Code for "Fast and flexible multi-task classification using Conditional Neural Adaptive Processes" and "TASKNORM: rethinking batch normalisation for meta-learning". https://github.com/cambridge-mlg/cnaps

CONVCNP: Wessel Bruinsma, Jonathan Gordon, and Andrew Y. K. Foong (2020). Code for "Convolutional Conditional Neural Processes". https://github.com/cambridge-mlg/convcnp

NEURALPROCESSES.JL Wessel Bruinsma and Jonathan Gordon (2020). Compositional Neural Processes with Julia. Reproduce experiments from "Meta-learning stationary stochastic processes with Convolutional Neural Processes". https://github.com/wesselb/NeuralProcesses.jl

The NPF: Yann Dubois, Jonathan Gordon, and Andrew Y. K. Foong (2020). A Jupyter-book tutorial on the NPF, accompanied with a unified implementation of prominent models. https://yanndubs.github.io/Neural-Process-Family

2

BACKGROUND

This chapter provides an introduction to the main concepts this thesis builds upon. We begin by introducing the notion of *meta-learning* in Section 2.1, and a probabilistic formulation thereof in Section 2.2. Then, in Section 2.3 we review important results in machine learning on set-structured inputs, which form the backbone for many of the models discussed in the thesis. In Sections 2.4 to 2.6, we introduce the *Neural-Process Family* (NPF). We provide a general overview of the NPF, followed by a more detailed examination of the two main sub-branches: the conditional NPF (Section 2.5) and the latent-variable NPF (Section 2.6). The aim of this chapter is to provide a tutorial and overview of probabilistic meta-learning and the NPF. However, the chapter also presents several novel contributions to the NPF. To introduce these while maintaining a distinction between existing work and novel contributions, we have gathered novel results in the final section of this chapter (Section 2.7).

Notation

We refer to inputs as $\mathbf{x} \in \mathcal{X}$ (typically $\mathcal{X} = \mathbb{R}^d$ for some fixed $d \in \mathbb{N}$), and outputs as $y \in \mathcal{Y}$. Output spaces may be $\mathbb{R}$ in regression tasks, $[0, 1]$ for grey scale images and $[0, 1]^3$ for RGB images, and $\{1, \dots, K\}$ for classification tasks. For ease of notation we typically consider scalar outputs. A *supervised learning* data set is denoted $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ for some $N \in \mathbb{N}$. We use the notation $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$, and similarly $\mathbf{y} = (y_1, \dots, y_N)$.

In the meta-learning setting, we will often be concerned with mappings from a *collection of data sets* to a space of *predictive functions*. To make this notion precise, let $\mathcal{S}_N \coloneqq (\mathcal{X} \times \mathcal{Y})^N$ be the collection of all supervised data sets of size $N \in \mathbb{N}$, and let $\mathcal{S} \coloneqq \bigcup_{N \in \mathbb{N}} \mathcal{S}_N$. In words, $\mathcal{S}$ is the collection of all possible *finite* supervised learning data sets in our domain. We denote $\mathcal{F} = \{f \colon \mathcal{X} \to \mathcal{Y}\}$ as the collection of functions mapping between our input and output spaces. With these collections in place, we can formalise mappings of the above form as $\mathcal{S} \to \mathcal{F}$.

An important object in meta-learning is a *task*, denoted ξ. Tasks consist of a *context set* $\mathcal{D}_c$ and a *target set* $\mathcal{D}_t$, such that $\xi = (\mathcal{D}_c, \mathcal{D}_t)$ with $\mathcal{D}_c, \mathcal{D}_t \in \mathcal{S}$. We use the shorthand $C \coloneqq |\mathcal{D}_c|$ and $T \coloneqq |\mathcal{D}_t|$ to refer to the number of input-output pairs in these sets. We denote $\mathbf{X}_C$ and $\mathbf{y}_C$ as the inputs and corresponding outputs of $\mathcal{D}_c$, and similarly for $\mathbf{X}_T$ and $\mathbf{y}_T$. Finally, we often assume access to a data set of tasks (referred to as a *meta data set*) with which we train meta-learning methods,

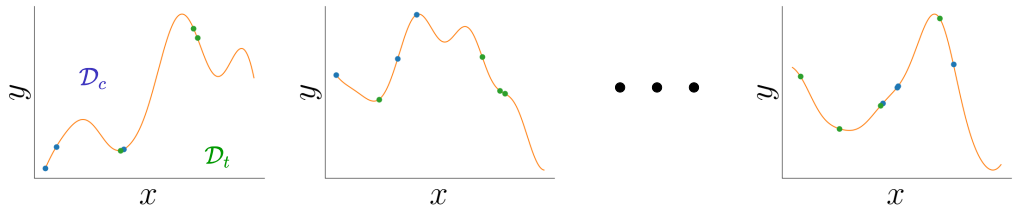


Figure 2.1: Illustrating the notation associated with meta-learning (1d regression setting used for the example). Each subplot represents a single task ξ_i in a meta-dataset Ξ. Each task is comprised of a *context set* $\mathcal{D}_c$ and a *target set* $\mathcal{D}_t$.

denoted $\Xi := \{\xi_i\}_{i=1}^N$ for some $N \in \mathbb{N}$. We assume that tasks in Ξ are identically and independently distributed (i.i.d.) i.e. $\xi_i \sim p(\xi)$. Figure 2.1 illustrates key quantities discussed above.

2.1 META-LEARNING

Informally, meta-learning is a machine learning paradigm that considers the problem of *learning to learn*. To obtain intuition for this idea, let us consider a standard approach to learning in supervised settings. Often, the first step is to have an expert specify and tune a *machine learning pipeline*. In this thesis, we think of machine learning pipelines as mappings $\Phi \colon \mathcal{S} \to\!\!\!\!\triangleright\, \mathcal{F}$. For example, an expert may specify Φ to consist of a single hidden layer neural network, a likelihood function for the observed data (e.g. Gaussian), an optimiser, number of training epochs, learning rate schedule, and ℓ_2 regularisation for the weights of the network. With these choices specified, we can think of Φ as a *mapping*: given any data set $S \in \mathcal{S}$, we can "push" S through Φ, which will output a predictive function over $\mathcal{X}$ to be used in down-stream tasks. Later in this thesis we will consider learners that output predictive distributions over $\mathcal{Y}$ rather than functions in $\mathcal{F}$. The process of producing a predictive function (distribution) by giving Φ a dataset is referred to as *training*. This perspective is illustrated in Figures 2.2 and 2.3.

In the deep learning setting (LeCun et al., 2015; Goodfellow et al., 2016), the set of decisions to be made can often feel daunting: a user must specify an architecture, parameter initialisation scheme, optimiser, learning rate schedule, early stopping patience, normalisation and augmentation schemes, regularisation strategies, and more. Performance of the resulting models is sensitive to several of these decisions, and design choices may interact (across the dimensions) in often unexpected fashions.

For some well-studied domains, best practices are readily available, e.g. the choice of ResNets (He et al., 2016) for image tasks or transformer models (Vaswani et al., 2017) for natural language tasks. However, in less-studied domains, such

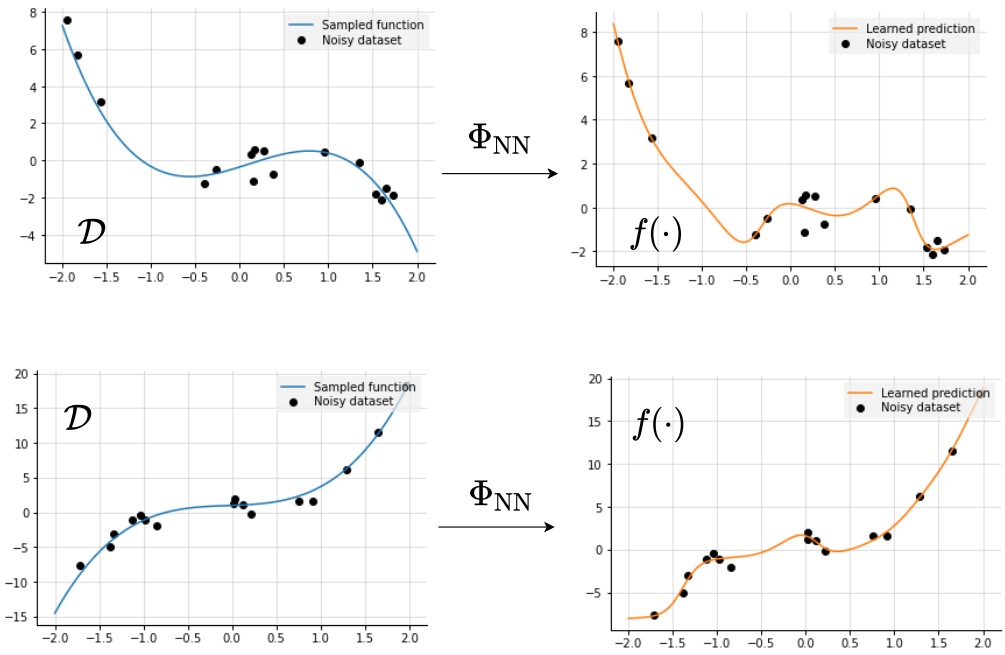


Figure 2.2: Illustration of viewing supervised learning pipelines as *mappings*. The learning pipeline, (Φ_{NN}), is a single hidden-layer neural network with 25 hidden units and ReLU activations, trained to maximise the log-likelihood of the data with Adam (Kingma and Ba, 2015). Two regression data sets are drawn by randomly sampling parameters for a cubic function and evaluating them (with Gaussian noise) at a small number of locations. Each is then used to train Φ_{NN}, and the resulting predictive model is visualised.

as environmental or health-care applications, this problem can be extremely challenging, and inhibit the deployment of machine learning solutions. Intuitively, the goal of meta-learning can be thought of as alleviating this issue by learning large parts of Φ directly from available data. In many senses, this is a natural progression in machine learning research, where the tendency is to automate as many decisions as possible by learning the (approximately) optimal choices directly from data (Clune, 2019; Sutton, 2019).

Meta-Learning Problem Statement

Let us attempt to formalise the meta-learning problem from the perspective of empirical risk minimisation (ERM) (Thrun and Pratt, 2012; Hospedales et al., 2020). Later, we will introduce probabilistic formulations of meta-learning, which will be used throughout this thesis. We assume there exists a distribution of interest over tasks—$p(\xi)$—and our goal is to *learn a learning pipeline* Φ that achieves "good" performance in expectation over $p(\xi)$, as measured by a loss function $\mathcal{L}(f; \xi)$.[1] We

1 In this thesis, we will most often consider the *negative* log-likelihood.

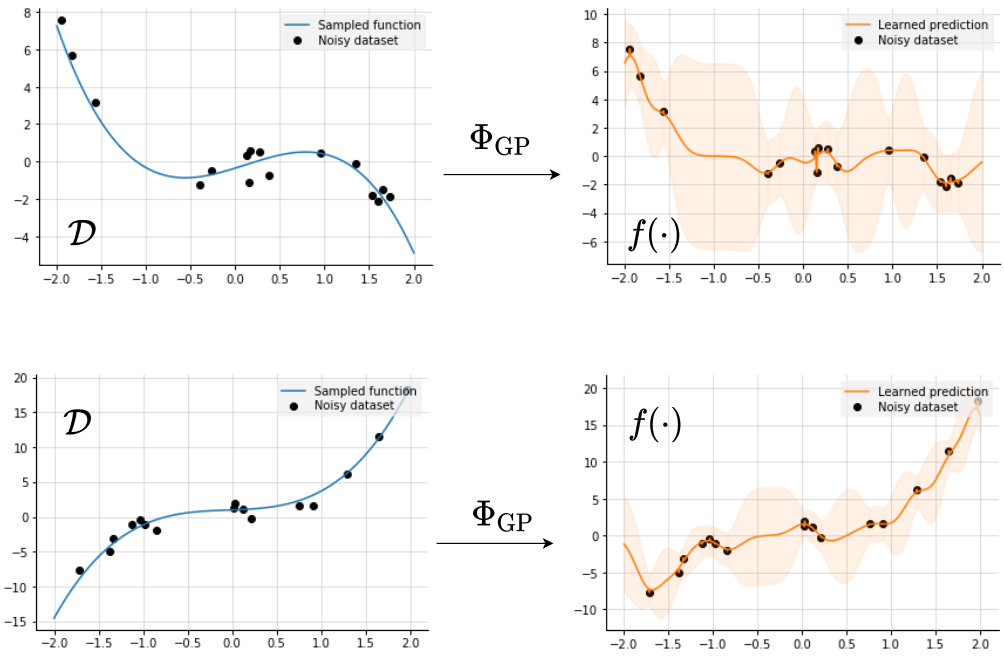


Figure 2.3: Same as Figure 2.2, but with a GP pipeline. The pipeline, (Φ_{GP}), is an EQ-kernel GP that learns the kernel hyper-parameters by optimising the marginal likelihood of the data set with L-BFGS (Liu and Nocedal, 1989) and performs exact posterior inference. The most prominent distinction from Figure 2.2 is that Φ_{GP} produces a *distribution over predictors*, while Φ_{NN} produces a single predictor. Later in this thesis we will formalise the handling of meta-learners that output distributions rather than functions in $\mathcal{F}$.

introduce a family of meta-learners $\{\Psi_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{F}, \boldsymbol{\theta} \in \Theta\}$. Given access to a meta data set $\Xi = \{\xi_i\}_{i=1}^N$ where $\xi_i \sim p(\xi)$, our goal is to find the meta-learner

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\xi \sim p(\xi)} \left[\mathcal{L}\left(\Psi_{\boldsymbol{\theta}} \left(\mathcal{D}_c^{(\xi)} \right); \xi \right) \right]. \tag{2.1}$$

Of course, this optimisation problem is generally intractable, not least because the underlying distribution $p(\xi)$ is typically unknown. Instead we typically assume access to an empirical distribution, Ξ_{train} and Ξ_{test} (referred to as *meta-train* and *meta-test* sets, respectively), and consider

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\xi \sim \Xi_{\text{train}}} \left[\mathcal{L}\left(\Psi_{\boldsymbol{\theta}} \left(\mathcal{D}_c^{(\xi)} \right); \xi \right) \right], \tag{2.2}$$

and measure our held-out performance analogously on Ξ_{test}.

2.2 PROBABILISTIC META-LEARNING

In many applications, rather than providing point predictions, we are interested in expressing the *uncertainty* associated with the predictions (Ghahramani, 2015).

An important example is when the predictions are used in down-stream decision making tasks, e.g. a system making predictions based on a clinical health-care data set to aid doctors in diagnosing patients. Another example, and one with which this thesis will often be concerned, is *few-shot* learning. In this setting, by definition, our context sets $\mathcal{D}_c$ contain only a handful of examples. Hence, we should not expect to obtain a unique predictor from $\mathcal{D}_c$, and may instead desire models that output *distributions* over possible predictions consistent with the available observations. The notion of a distribution over predictors, i.e. a distribution over *functions*, leads us quite naturally to consider *stochastic processes* (SPs; Ross et al., 1996; Lindgren, 2012). We next introduce a probabilistic perspective on meta-learning that frames the problem as recovering *prediction maps* of underlying stochastic processes.

2.2.1 *Stochastic Processes, Kolmogorov Extension, and Prediction Maps*

Distributions over functions are known in mathematics as *stochastic processes* (SPs; Ross et al., 1996; Lindgren, 2012). We first provide a brief overview of stochastic processes, and use the resulting definitions to present a view of *probabilistic* meta-learning.[2]

Overview of Stochastic Processes

For our purposes, a SP on $\mathcal{X}$ will be defined as a probability measure on $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$, i.e., $\mathcal{Y}^{\mathcal{X}}$, equipped with a σ-algebra, denoted Σ.[3] As in practice we only ever observe finite data sets, we consider the measurable sets of Σ as those which can be specified by the values of the function at a countable subset $I \subset \mathcal{X}$ of its input locations. We denote the set of all such measures as $\mathcal{P}(\mathcal{X})$.

We typically assume there is some *ground truth* SP $P \in \mathcal{P}(\mathcal{X})$. In later chapters, we will see examples of such a process, but in practice, we are only ever interested in distributions on collections of finitely many random variables. To make the connection between P and distributions over finite collections, we invoke the *Kolmogorov extension theorem*.

Definition 2.1 (Kolmogorov Consistency)
Let $\mathcal{X}$ be an input space and $n \in \mathbb{N}$. For every $k \in \mathbb{N}$ and finite sequence of distinct variables $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathcal{X}$, let $p_{\mathbf{x}_1,\ldots,\mathbf{x}_k}$ be a density on $(\mathcal{Y})^k$. We call the collection of these densities Kolmogorov consistent *if, for all $y_1, \ldots, y_k$, they are*

2 We note that while a comprehensive treatment of SPs requires concepts from measure theory, this thesis requires only basic notions. We introduce only the concepts necessary for what follows. For a comprehensive introduction to stochastic processes, see e.g. Tao (2011) or Ross et al. (1996).

3 In particular, Σ is the product σ-algebra of the Borel σ-algebra over each index point (Tao, 2011).

1. consistent under permutations, *i.e, for every permutation* π *of* $\{1, \dots, k\}$

$$p_{\mathbf{x}_1,\dots,\mathbf{x}_k}(y_1, \dots, y_k) = p_{\mathbf{x}_{\pi(1)},\dots,\mathbf{x}_{\pi(k)}}\left(y_{\pi(1)}, \dots, y_{\pi(k)}\right); \tag{2.3}$$

2. consistent under marginalisation, *i.e., for any* $m \in \mathbb{N}$

$$p_{\mathbf{x}_1,\dots,\mathbf{x}_k}(y_1, \dots, y_k) = \int p_{\mathbf{x}_1,\dots,\mathbf{x}_{k+m}}(y_1, \dots, y_{k+m})\, \mathrm{d}y_{k+1} \dots \mathrm{d}y_{k+m}. \tag{2.4}$$

The Kolmogorov extension theorem (KET) states that for any collection of measures satisfying Definition 2.1, there exists a unique measure on $(\mathcal{Y}^{\mathcal{X}}, \Sigma)$ that has these densities as its *finite-dimension distributions*. Hence, we can equivalently think of SPs either in terms of (i) their finite-dimensional marginals or (ii) as measures on $\mathcal{F}$.

The Prediction Map

An important notion for this perspective of probabilistic meta-learning is that of the *prediction map* associated with any $P \in \mathcal{P}(\mathcal{X})$. To define the prediction map, we must first clarify what it means to condition on observations from the stochastic process. Let $p(\mathbf{y}|\mathbf{X})$ denote the density with respect to the Lebesgue measure of the finite dimensional marginal of P with index set $\mathbf{X}$. Now, assume we have observed a finite number of points $\mathcal{D}_c = (\mathbf{X}_C, \mathbf{y}_C)$ from P, and let $\mathbf{X}_T$ be another finite index set. Then by definition of conditional distribution, the finite dimensional marginal at $\mathbf{X}_T$ conditioned on $\mathcal{D}_c$ is the distribution with density

$$p(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c) = \frac{p(\mathbf{y}_T, \mathbf{y}_C|\mathbf{X}_T, \mathbf{X}_C)}{p(\mathbf{y}_C|\mathbf{X}_C)}. \tag{2.5}$$

It is straightforward to verify that for a fixed $\mathcal{D}_c$, the conditional marginal distributions for any $\mathbf{X}_T$ from Equation (2.5) satisfy the conditions of Definition 2.1. Hence, the KET implies there is a unique measure $P \in \mathcal{P}(\mathcal{X})$ on $(\mathcal{Y}^{\mathcal{X}}, \Sigma)$ that has Equation (2.5) as its finite marginals. We denote this measure $P_{\mathcal{D}_c}$. With this notation in place, we can state the definition of a prediction map.

Definition 2.2 (Prediction map)
Let P *and* $P_{\mathcal{D}_c}$ *be as defined above. We call* $\pi_P \colon \mathcal{S} \twoheadrightarrow \mathcal{P}(\mathcal{X})$*, the map such that* $\pi_P = \mathcal{D}_c \mapsto P_{\mathcal{D}_c}$*, the* prediction map.

In words, the prediction map takes any observed data set $\mathcal{D}_c$ to the exact predictive SP conditioned on $\mathcal{D}_c$. To gain further intuition, let us consider a well-studied class of SPs in the machine learning literature: GPs. Recall that a GP is completely defined by its mean function $\mu_{\mathrm{GP}} \colon \mathcal{X} \twoheadrightarrow \mathcal{Y}$ and covariance function $k_{\mathrm{GP}} \colon \mathcal{X} \times \mathcal{X} \twoheadrightarrow \mathbb{R}$. For finite index sets $\mathbf{X}$ and $\mathbf{X}'$, we use the notation $\mu_{\mathrm{GP}}(\mathbf{X})_i \coloneqq \mu_{\mathrm{GP}}(\mathbf{X}_i)$

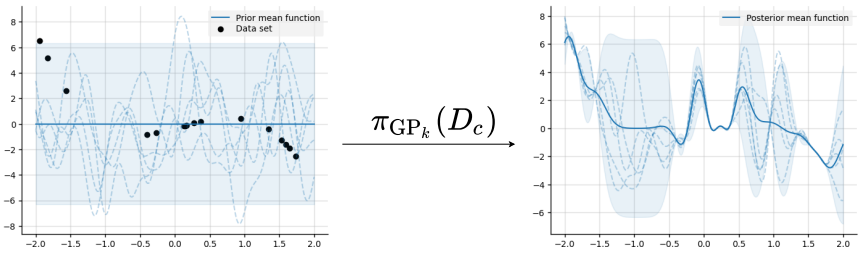


Figure 2.4: Illustration of the prediction map associated with a GP. Left-hand side depicts stochastic process defined by a GP with a zero mean function and EQ covariance function with lengthscale 1 (k). The process is fully-specified by these two functions. An observed data set, produced from a randomly sampled cubic function, is also illustrated on the lefthand side. The prediction map associated with this GP then maps the data set to another GP by outputting *posterior* mean and covariance functions, illustrated on the righthand side.

and $k_{\mathrm{GP}}(\mathbf{X}, \mathbf{X}')_{i,j} := k_{\mathrm{GP}}\left(\mathbf{X}_i, \mathbf{X}'_j\right)$. Then, For any $\mathbf{X}$, a GP specifies $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}; \mu_{\mathrm{GP}}(\mathbf{X}), k_{\mathrm{GP}}(\mathbf{X}, \mathbf{X}))$.

A well-known property of GPs is that, for Gaussian likelihoods, the output of the prediction map is also a GP which can be expressed analytically (Rasmussen, 2003; Williams and Rasmussen, 2006). In particular, for a GP with a zero mean function and covariance function k, we have that

$$\pi_{\mathrm{GP}_k} = \mathcal{D}_c \twoheadrightarrow (\mu_{\mathrm{post}}, k_{\mathrm{post}}) \tag{2.6}$$

$$\mu_{\mathrm{post}}(\mathbf{X}) = k(\mathbf{X}, \mathbf{X}_C)k(\mathbf{X}_C, \mathbf{X}_C)^{-1}\mathbf{y}_C, \tag{2.7}$$

$$k_{\mathrm{post}}(\mathbf{X}, \mathbf{X}) = k(\mathbf{X}, \mathbf{X}) - k(\mathbf{X}, \mathbf{X}_C)k(\mathbf{X}_C, \mathbf{X}_C)^{-1}k(\mathbf{X}_C, \mathbf{X}). \tag{2.8}$$

Here we have specified the mapping as outputting a mean and covariance function, which completely specify the GP output by the map. The prediction map for a particular GP and data set is illustrated in Figure 2.4.

2.2.2 *Probabilistic Meta-Learning as Recovering Stochastic Process Prediction Maps*

We are now ready to formalise our proposed notion of probabilistic meta-learning. Let $\mathcal{S}$ and $\mathcal{P}(\mathcal{X})$ be as defined above, and π_P be the prediction map associated with $P \in \mathcal{P}(\mathcal{X})$. Given access to a meta-data set Ξ, where each $\xi \in \Xi$ is assumed to have arisen from observing P at a finite number of points, we define the goal of probabilistic meta-learning to be recovering π_P.

In this thesis, our strategy for recovering π_P will generally follow two steps:

1. Construct models $\Psi_{\boldsymbol{\theta}} : \mathcal{S} \twoheadrightarrow \mathcal{P}(\mathcal{X})$ parametrised by $\boldsymbol{\theta} \in \Theta$. Ideally, these models should be as expressive as possible, such that at least one member of $\{\Psi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ is "close" to π_P in a meaningful sense.

2. Define a loss function $\mathcal{L}(\Psi_{\boldsymbol{\theta}}; \cdot)$ such that, in suitable data limits, π_P is a global minimiser, and (approximately) solve

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\Psi_{\boldsymbol{\theta}}; \Xi) . \tag{2.9}$$

2.2.3 *Limitations*

Before proceeding to introduce the main components of the NPF, we discuss an important limitation of the formulation of probabilistic meta-learning presented above. In particular, a core assumption in the derivation, which is implicit to the KET assumptions, is that the functions $f \in \mathcal{F}$ *are independent of the inputs* $\mathbf{x}$. While common in the supervised meta-learning literature (see e.g. Finn et al., 2017; Ravi and Larochelle, 2017; Snell et al., 2017; Triantafillou et al., 2020), this assumption may be prohibitive in certain cases.

This assumption is also implicit in the graphical models discussed later in this chapter when describing the NPF subfamilies (Figures 2.7 and 2.10), which treat the inputs as observed (i.e. deterministic), the *labels* as the targets, and the parameters $\boldsymbol{\theta}$ as independent of the inputs. Such models, referred to as *anti-causal models* (Schölkopf et al., 2012), suffer from several limitations. An important example is that anti-causal models inhibit *semi-supervised* learning (Chapelle et al., 2009; Schölkopf et al., 2012).

As a concrete example, in Chapter 6 we focus on an image-recognition application from a challenging dataset dubbed *meta-dataset* (Triantafillou et al., 2020). In this example, where tasks may arise from disparate image-recognition datasets (e.g., MNIST (LeCun et al., 1989), ImageNet (Krizhevsky et al., 2012), etc'), the assumption that the underlying function f is independent of the inputs seems quite restrictive. A more reasonable assumption would be that the underlying functions vary for different datasets in the amalgamation. Moreover, it seems desirable in this setting to have the ability to improve models for this task when given further unlabelled data from the distribution, which is not possible under the current framework. Despite these limitations, the vast majority of models applied to this benchmark make similar assumptions, and we demonstrate in Chapter 6 that such models can achieve excellent performance.

2.3 LEARNING AND FUNCTION APPROXIMATION ON SETS

Before introducing the NPF, we review the intimately related area of machine learning on sets. Representation learning on sets is an important sub-area of machine learning, with applications to point-cloud modelling (Qi et al., 2017a; Qi et al., 2017b; Wu et al., 2019), set retrieval (Zaheer et al., 2017), and image

tagging (Zaheer et al., 2017). It is particularly relevant to this thesis, given the need to parametrise models that operate on $\mathcal{S}$. As $\mathcal{S}$ is a collection of sets, the study of learning and function approximation on sets will form the theoretical underpinning for many of the models considered in the thesis.

2.3.1 *Valid functions on sets*

Our goal is to construct and represent functions operating on sets. However, in practice sets are often represented on computers as *sequences*, or *ordered tuples*. The distinguishing property of sets, as opposed to sequences, is that they are *unordered*. As such, any function f acting on sequences must be invariant to the order in which it processes the elements of its input to be considered a valid function on sets. This leads to the central notion of *permutation invariance*.

Property 2.1 ($\mathbb{S}_n$-invariant and $\mathbb{S}$-invariant functions)
Let $X_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a sequence with n elements $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. Let $\mathcal{X}^*$ be the Kleene closure of $\mathcal{X}$, i.e. the set of all finite sequences made up of elements of $\mathcal{X}$, and denote $\mathcal{X}_n^* \subset \mathcal{X}^*$ be the set of all sequences of n elements. Let $\mathbb{S}_n$ be the group of permutations of n symbols for $n \in \mathbb{N}$. A function f on $\mathcal{X}_n^*$ is called $\mathbb{S}_n$-invariant *if*

$$f(X_n) = f(\pi X_n) \quad \textit{for all} \quad \pi \in \mathbb{S}_n \quad \textit{and} \quad X_n \in \mathcal{X}_n^*, \tag{2.10}$$

where the application of π is given as $\pi \colon X_n \mapsto \left(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(n)}\right)$. A function f on $\mathcal{X}^*$ is called $\mathbb{S}$-invariant *if the restrictions $f|_{\mathcal{X}_n^*}$ are $\mathbb{S}_n$-invariant for all $n \in \mathbb{N}$.*

Definition 2.3 (Functions on sets)
Let f be a function acting on $\mathcal{X}^*$. We say that f is a valid function on sets *if it satisfies Property 2.1, i.e. f is $\mathbb{S}$-invariant.*

2.3.2 *The DeepSets Representation Theorems*

Zaheer et al. (2017) provide a general framework for modelling functions that satisfy Property 2.1 with deep neural networks. They coin the framework *DeepSets*, which is defined by parametrisations of the following form (for functions whose target space is $\mathbb{R}$):

$$f(X) = \rho(E(X)); \qquad E(X) = \sum_{\mathbf{x} \in X} \phi(\mathbf{x}), \tag{2.11}$$

where $\phi \colon \mathcal{X} \to \mathbb{R}^d$, and $\rho \colon \mathbb{R}^d \to \mathbb{R}$, both parametrised by neural networks. Zaheer et al. (2017) further provide *universal representation* statements for DeepSet networks, which we restate below.

Theorem 2.1 (DeepSets: countable case)
Using the notation above, assume $\mathcal{X}$ is countable. *A function $f : \mathcal{X}^* \to \mathbb{R}$ operating on ordered tuples satisfies Property 2.1 if and only if it can be decomposed as*

$$f(X) = \rho\left(\sum_{\mathbf{x}\in X} \phi(\mathbf{x})\right) \tag{2.12}$$

for suitable continuous transformations ρ and ϕ.

The restriction to countable universes is quite severe, as we are seldom interested in such objects in machine learning (Wagstaff et al., 2019). Unfortunately, Zaheer et al. (2017) could only prove the extension to the uncountable case for *fixed-size* sets. The key result of that work can be (re)stated as follows.

Theorem 2.2 (DeepSets: uncountable case)
Let $\mathcal{X} = [0, 1]$ and $f : \mathcal{X}^M \to \mathbb{R}$. Then f is an $\mathbb{S}_M$-invariant and continuous function if and only if it has a representation of the form

$$f(X) = \rho\left(\sum_{\mathbf{x}\in X} \phi(\mathbf{x})\right) \tag{2.13}$$

for some continuous functions $\phi : \mathcal{X} \to \mathbb{R}^{M+1}$ and $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$.

Theorems 2.1 and 2.2 constitute important results in that they demonstrate that *any* function satisfying Property 2.1 must have a representation of such a form. The use of continuous functions for ρ and ϕ motivate the use of learnable neural networks for these. Such parametrisations have been widely adopted and demonstrated to achieve compelling results in several domains (e.g., Edwards and Storkey, 2017; Qi et al., 2017a; Wu et al., 2019).

2.3.3 *Weaknesses of the DeepSets Theorems*

While Theorems 2.1 and 2.2 constitute important milestones in the theory of learning on sets, they have several significant weaknesses. Most importantly, as argued by Wagstaff et al. (2019), Theorem 2.1 is not of particular practical use as it is limited to countable universes. Moreover, Theorem 2.2 holds only for *fixed-sized* sets, and Zaheer et al. (2017) were not able to extend the proof to hold for varying-size sets, which in practice are the objects of interest in machine learning applications. Finally, an often overlooked fact is that Theorem 2.2 only applies to univariate set elements in $[0, 1]$. It turns out that their proof, which relies on sum-of-power-mappings (Zaheer et al., 2017, Lemma 4), is not straightforwardly extended to the multivariate case without modification.

More recently, Wagstaff et al. (2019) provide an extension of Theorem 2.2 to the case of varying-sized sets of elements in $[0, 1]$. Moreover, Bloem-Reddy and Teh

(2020) provide a general (measure-theoretic) extension to Theorem 2.2 that allows for varying-sized sets of elements from arbitrary spaces. In this thesis we will build upon Theorem 2.2 and extend it in two ways:

1. In Section 2.7.2, we provide an additional extension Theorem 2.2 to include vector-valued sets of varying sizes, and relate it directly to the NPF.
2. In Chapter 3 we extend the theory to include functions f that are both permutation invariant and *translation-equivariant*.

2.4 THE NEURAL PROCESS FAMILY

We now turn our attention to the Neural Processes family (NPF): a class of models for probabilistic meta-learning. Neural Processes (NPs) were originally introduced by Garnelo et al. (2018a,b), and subsequently expanded upon in several works (e.g., Gordon et al., 2019; Kim et al., 2019; Louizos et al., 2019; Requeima et al., 2019; Gordon et al., 2020a; Xu et al., 2020; Foong et al., 2020). The central idea in the NPF is to use deep neural networks to parametrise a mapping between a collection of data sets and a space of stochastic processes in an end-to-end fashion. To achieve this, models in the NPF must deal with two major considerations: (i) the design of architectures that accept data sets as inputs (i.e. satisfy Property 2.1), and (ii) ensuring that model outputs respect the properties of SPs, namely, consistency under permutation and marginalisation (Definition 2.1).

Remark 2.1
While the output of a NP is a proper SP for every context set $\mathcal{D}_c$, NPs do not ensure that the resulting predictive processes are consistent with respect to some prior process. *Thus, we should not think of NPs as modelling well-defined SPs, but rather only as parametrising the* mapping *between context sets and predictive processes.*

Below, we discuss the general design principles underlying the NPF, and how these relate to and achieve the modelling desiderata. We then introduce two approaches to enforcing the required consistency constraints on the predictive processes, which define the two major branches of the NPF: factorised models,[4] known as *conditional* NPs (e.g., Garnelo et al., 2018a; Gordon et al., 2020a), and latent variable models (e.g, Garnelo et al., 2018b; Gordon et al., 2019), which we will refer to as latent NPs (LNPs). These are discussed in detail in Sections 2.5 and 2.6, respectively.

4 The term factorised is somewhat ambiguous here, and will be made precise in Section 2.5.

2.4.1 *Design Principles for the Neural Process Model Class*

Let us consider the construction of maps directly from data sets to predictive processes using deep neural networks. The first challenge is that such a map must accept as input a context set, $\mathcal{D}_c$. This poses two major differences from standard neural networks, which typically expect vector- (or tensor-) valued inputs:

- Sets may have varying sizes. We wish to consider architectures that can handle sets of varying size.
- Sets have no intrinsic ordering. Thus, we require our architectures to be invariant to the order with which they process the elements of the set, i.e., satisfy Property 2.1.

To satisfy these requirements, the NPF employs the following approach. First, each observation in the context set is processed separately, using a shared embedding function $\mathbf{r}_c = \phi(\mathbf{x}_c, y_c)$, for each $(\mathbf{x}_c, y_c) \in \mathcal{D}_c$. We call ϕ the *local encoder*, and $\mathbf{r}_c$ a *local encoding* of the c^{th} observation. NPs parametrise ϕ using a neural network module with parameters $\boldsymbol{\theta}_{\boldsymbol{\phi}}$. Next, the local encodings $\{\mathbf{r}_c\}_{c=1}^{C}$ are combined into a single representation $\mathbf{r}$ using an *aggregation* function (see Figure 2.6). Typically, a simple operation such as averaging or summation will be used for aggregation. Importantly, the aggregation is restricted to be a permutation invariant function of the local encodings, and it is straightforward to see that either choice results in a permutation invariant operation due to the commutativity of summation. We can think of $\mathbf{r}$ as a representation of the entire context set, and will sometimes write $\mathbf{r}\left(\mathcal{D}_c\right)$ to make this explicit.

This encoding is then passed to the model decoder, $d_{\boldsymbol{\theta}}$ (also parametrised by a deep neural network), which outputs predictions for the inputs in the target set, $\mathcal{D}_t$. The second major challenge for the NPF is that this predictive distribution must respect the two requirements of the KET. We consider two approaches to achieving this, which define the two major branches of the NPF.

Conditional NPs

Conditional members of the NPF assume that predictions over target inputs factorise conditioned on $\mathbf{r}$, i.e.

$$p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(y_t|\mathbf{x}_t, \mathbf{r}\left(\mathcal{D}_c\right)\right). \tag{2.14}$$

For example, a typical assumption may be

$$p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{r}\left(\mathcal{D}_c\right)\right) = \mathcal{N}\left(y; \boldsymbol{\mu}(\mathbf{x}, \mathbf{r}), \boldsymbol{\sigma}^2(\mathbf{x}, \mathbf{r})\right). \tag{2.15}$$

Here, $\boldsymbol{\mu}(\cdot,\cdot)$ and $\boldsymbol{\sigma}(\cdot,\cdot)$ are functions (parametrised by neural networks) that map inputs $\mathbf{x}$ and representations $\mathbf{r}$ to the parameters of the predictive distribution. In such a case, the decoder $d_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{r}) = (\boldsymbol{\mu}(\mathbf{x},\mathbf{r}),\boldsymbol{\sigma}(\mathbf{x},\mathbf{r})) \in \mathbb{R}\times\mathbb{R}_+$ outputs a mean and variance parameter for any $(\mathbf{x},\mathbf{r})$ pair. We collectively refer to members of the NPF that make this assumption as *conditional* NP models, and to this sub-family as the conditional NPF (CNPF). In Section 2.5 we discuss the CNPF and the implications of the factorisation assumption in further detail.

Latent-Variable NPs

Latent-variable members of the NPF use $\mathbf{r}(\mathcal{D}_c)$ to define a latent variable $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{r}(\mathcal{D}_c))$. For now, assume that $\mathbf{z} \in \mathbb{R}^d$. The predictive distribution is then defined as

$$p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T,\mathcal{D}_c\right) = \int p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{r}(\mathcal{D}_c)) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(y_t|\mathbf{x}_t,\mathbf{z}\right) \mathrm{d}\mathbf{z}. \tag{2.16}$$

Note that if $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{r}(\mathcal{D}_c)) = \delta(\mathbf{z}-\mathbf{r}(\mathcal{D}_c))$, the CNPF predictive distribution is recovered, implying that the LNPF subsumes the CNPF as a special case. Here too, we must specify the form of the conditional distribution. A standard assumption may be of the form

$$p_{\boldsymbol{\theta}}\left(y|\mathbf{x},\mathbf{z}\right) = \mathcal{N}\left(y;\boldsymbol{\mu}(\mathbf{x},\mathbf{z}),\boldsymbol{\sigma}^2(\mathbf{x},\mathbf{z})\right). \tag{2.17}$$

Hence, the decoder has a similar form as in the conditional case, i.e. $d_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) = (\boldsymbol{\mu}(\mathbf{x},\mathbf{z}),\boldsymbol{\sigma}(\mathbf{x},\mathbf{z}))$.

Remark 2.2
Importantly, while the conditional likelihood factorises, the predictive distribution $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T,\mathcal{D}_c\right)$ does not, as $\mathbf{z}$ induces dependencies across target point predictions.

For latent-variable models, we must also specify the form of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{r})$, where again a standard parametrisation is

$$p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{r}\right) = \mathcal{N}\left(\mathbf{z};\boldsymbol{\mu}(\mathbf{r}),\boldsymbol{\sigma}^2(\mathbf{r})\right), \tag{2.18}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are parametrised by additional neural networks. Note that we are overloading the notation of the functions $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, but it should be clear from context to which functions we are referring. We collectively refer to members of the NPF that employ this model structure as latent NPs (LNPs), and refer to this branch of the NPF as the latent NPF (LNPF). In Section 2.6 we discuss the LNPF and the implications of using latent variables in detail.

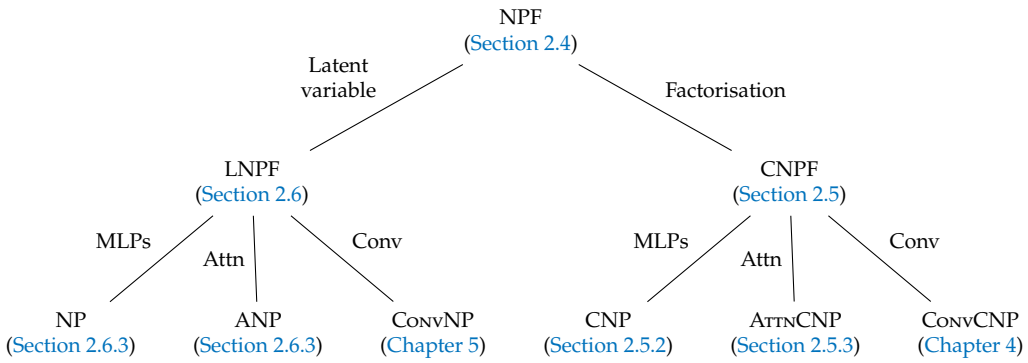


Figure 2.5: Family tree of the NPF members discussed in this thesis. The first level depicts the two approaches to designing predictive distributions satisfying the KET conditions, defining the two major sub-branches of the NPF. The second level relates to the inductive biases introduced when designing the model architectures, which lead to different members of the NPF. Nodes are labeled with the chapters/sections in the thesis in which they are discussed.

SUMMARY To summarise, models in the NPF employ an *encoder-decoder architecture* to parametrise a mapping between data sets and predictive processes with deep neural networks. The encoder maps context sets $\mathcal{D}_c$ to a learned representation space, and the decoder conditions on the representation (or samples from the resulting distribution), and outputs predictive distributions over target locations. A computation graph for a generic member of the NPF is presented in Figure 2.6. Specifying a NP model thus requires specifying:

- a local encoder ϕ with parameters $\boldsymbol{\theta}_\phi$
- an aggregation function $\text{Agg}\colon \{\mathbf{r}_c\}_{c=1}^C \mapsto \mathbf{r}$, where $\mathbf{r}$ may be deterministic (CNPF) or stochastic (LNPF), potentially with additional parameters $\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma$
- a conditional likelihood of the form $p_{\boldsymbol{\theta}}\left(\mathbf{y}|\mathbf{x}, \cdot\right)$, where the final argument may be a deterministic representation or sample of a random variable
- a decoder $d_{\boldsymbol{\theta}}$ that outputs the parameters of the conditional likelihood, with parameters $\boldsymbol{\theta}_d$

We denote the model parameters as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma, \boldsymbol{\theta}_d\}$. In Sections 2.5 and 2.6 we will see that these choices define a rich class of models, and allow us to encode different assumptions into the NP model.

2.4.2 *Training Members of the Neural Process Family*

Similarly to other meta-learners, training members of the NPF requires access to a meta-data set of the form $\Xi = \{\xi_i | i = 1, \ldots, N\}$. Given that NP models define a predictive distribution of the form $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$, a natural idea is

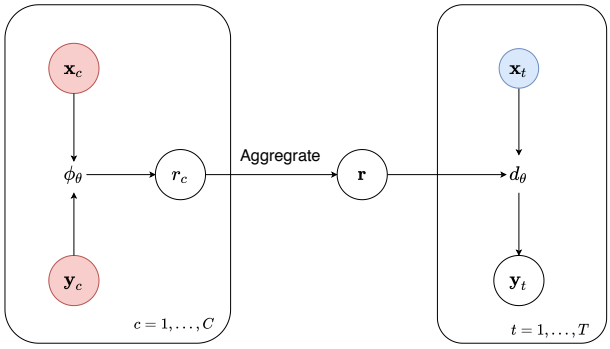


Figure 2.6: Computational graph for NP models, depicting the computation for single task ξ. Computational graphs are directed acyclic graphs (DAGs) representing the order of computation specified by a model. Each circled node in the graph represents a variable or intermediate computation, and arrows into nodes indicate dependency on of a node on other quantities in the graph. We use un-circled intermediate text to label function names.

Algorithm 1: Maximum likelihood training of Neural-Process models.

Input: Model parameters $\boldsymbol{\theta}$, Data $\Xi = \{\xi_i | i = 1, \ldots, N\}$
Parameters: Learning rate α
while *not converged* **do**
  $\mathcal{D}_c, \mathcal{D}_t \leftarrow \xi \sim \Xi$;
  $\mathcal{L} \leftarrow \log p_{\boldsymbol{\theta}}(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}$
end
Output: $\boldsymbol{\theta}$

to train the parameters $\boldsymbol{\theta}$ by maximising this likelihood over Ξ. This is the maximum-likelihood approach to training NP models, and can be expressed as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta} \in \Theta} \mathcal{L}_{\text{ML}}(\boldsymbol{\theta}; \Xi); \qquad \mathcal{L}_{\text{ML}}(\boldsymbol{\theta}; \Xi) \coloneqq \frac{1}{N} \sum_{\xi \in \Xi} \log p_{\boldsymbol{\theta}}(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c), \quad (2.19)$$

where the $\frac{1}{N}$ turns the likelihood to an expectation rather than a sum, but can be simply "absorbed" into the learning rate when applying standard optimisation procedures. A simple maximum-likelihood training procedure, using stochastic gradient descent (Bottou, 2010) with a fixed learning rate α, is detailed in Algorithm 1. In practice, we may use standard neural network software packages (e.g. PyTorch (Paszke et al., 2019) or TensorFlow (Abadi et al., 2016)) that implement backpropagation to compute the gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ (Rumelhart et al., 1986; Baydin et al., 2017) , and employ more sophisticated optimisers (e.g. Polyak, 1964; Tieleman and Hinton, 2012; Sutskever et al., 2013; Kingma and Ba, 2015).

Algorithm 1 assumes the predictive likelihoods $p_{\boldsymbol{\theta}}(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)$ are tractable. As we shall see, this is true for conditional members of the NP family, and is a useful assumption in understanding the goals and tradeoffs associated with training NP

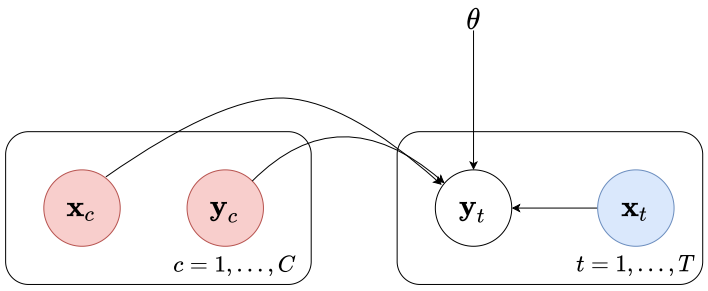


Figure 2.7: Graphical model depicting the factorisation assumption common to Conditional NP models. The graphical representation indicates that, conditioned on the complete context set $\mathcal{D}_c$ and the model parameters $\boldsymbol{\theta}$, the predictive distribution at any given target location $\mathbf{x}_t$ is independent of any other location.

models. In Section 2.7.1 we discuss guarantees offered by maximum-likelihood training of NP members in suitable limits, and in Sections 2.6.2 and 2.7.3 we discuss approximations and alternative training procedures when the tractability assumption does not hold.

2.5 THE CONDITIONAL NEURAL PROCESS SUB-FAMILY

Models in the NPF must output predictive distributions $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$ over target sets of arbitrary size that satisfy the consistency conditions of the KET (Definition 2.1). In this section, we consider what is arguably the simplest way to achieve this: factorised predictive distributions of the form

$$p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right) = \prod_{\mathbf{x},y\in\mathcal{D}_t} p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right), \tag{2.20}$$

where $\mathbf{r}$ is a learned embedding function that maps context sets to a representation space. We can represent this assumption concisely using the graphical model depicted in Figure 2.7. Recall that the general strategy in the NPF is to first *encode* $\mathcal{D}_c$ (via aggregation of the local encodings), and then *decode* this resulting representation to produce predictive distributions. Thus, members of the CNPF can be characterised and organised according to three key design decisions:

1. parametrisation of the *local* encoding function ϕ,
2. design of the aggregation function mapping local encodings to a single representation, and
3. parametrisation of the decoder $d_{\boldsymbol{\theta}}$.

Before introducing several prominent members of the CNPF, we take a closer look at the assumption underlying this class of models.

2.5.1 *On the Factorisation Assumption*

We first verify that the factorisation assumption indeed satisfies the conditions of Definition 2.1, i.e. that the CNPF predictive distribution specifies a consistent SP for any $\mathcal{D}_c, \mathcal{D}_t$. To see that the CNPF prediction is consistent under permutation, note that we can express the predictive density as

$$\begin{aligned} p_{\boldsymbol{\theta}}\left(y_1, \ldots, y_T | \mathbf{x}_1, \ldots, \mathbf{x}_T, \mathcal{D}_c\right) &= \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(y_t | \mathbf{x}_t, \mathbf{r}(\mathcal{D}_c)\right) \\ &= p_{\boldsymbol{\theta}}\left(y_{\pi(1)}, \ldots, y_{\pi(T)} | \mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(T)}, \mathcal{D}_c\right), \end{aligned}$$

where all we have used in the above is the commutativity of multiplication. To verify consistency under marginalisation, consider two target points $\mathbf{x}_1$ and $\mathbf{x}_2$. By marginalising out the second target output, we have

$$\begin{aligned} \int p_{\boldsymbol{\theta}}\left(y_1, y_2 | \mathbf{x}_1, \mathbf{x}_2, \mathcal{D}_c\right) \mathrm{d}y_2 &= \int p_{\boldsymbol{\theta}}\left(y_1 | \mathbf{x}_1, \mathbf{r}(\mathcal{D}_c)\right) p_{\boldsymbol{\theta}}\left(y_2 | \mathbf{x}_2, \mathbf{r}(\mathcal{D}_c)\right) \mathrm{d}y_2 \\ &= p_{\boldsymbol{\theta}}\left(y_1 | \mathbf{x}_1, \mathbf{r}(\mathcal{D}_c)\right) \int p_{\boldsymbol{\theta}}\left(y_2 | \mathbf{x}_2, \mathbf{r}(\mathcal{D}_c)\right) \mathrm{d}y_2 \\ &= p_{\boldsymbol{\theta}}\left(y_1 | \mathbf{x}_1, \mathcal{D}_c\right), \end{aligned}$$

which shows that the predictive distribution obtained by querying the CNPF member at $\mathbf{x}_1$ is the same as that obtained by querying it at $(\mathbf{x}_1, \mathbf{x}_2)$, and marginalising out the second target point. Of course, this simple idea works with collections of any size, and marginalising any subset of the variables.

An important advantage of the factorisation assumption is that the density $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c\right)$ is tractable (for tractable choices of $p_{\boldsymbol{\theta}}\left(y | \mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right)$), implying that we can train members of the CNPF using Algorithm 1 without further modifications. Yet the factorisation assumption has important limitations. First, it is generally not the case that distributions produced by π_P factorise in this way, limiting our ability to recover the true prediction map. Moreover, we can not straightforwardly produce coherent samples from the model without employing complicated auto-regressive sampling procedures (e.g. Larochelle and Murray, 2011; Oord et al., 2016a,b; Parmar et al., 2018; Papamakarios et al., 2019b). Such procedures can be computationally expensive, and require specifying an ordering for $\mathcal{X}$, which conflicts with our notions of permutation invariance, and when used in practical scenarios, will not lead to consistent predictive distributions. Finally, typical parametrisations of CNPF models use Gaussian predictive distributions for $p_{\boldsymbol{\theta}}\left(y | \mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right)$. This choice implies that multi-modal, heavy-tailed, or asymmetric predictive likelihoods cannot be recovered. While this can be avoided by choosing

alternative parametric forms for $p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right)$, this requires additional domain expertise, and places additional modelling burdens on the user.

2.5.2 *The Conditional Neural Process*

We now turn our attention to the simplest member of the CNPF — the *Conditional Neural Process* (CNP; Garnelo et al., 2018a; Eslami et al., 2018). The CNP was the first member of the NPF proposed in the literature, and is defined by the following choices.

ENCODER. The local encoding network is of the form $\phi\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{d_{\text{enc}}}$ for some $d_{\text{enc}} \in \mathbb{N}_+$. The CNP parametrises ϕ with a multi-layer perceptron (MLP) that accepts as input the concatenation $[\mathbf{x}; y]$, and outputs a vector in $\mathbb{R}^{d_{\text{enc}}}$. The aggregation function is specified as a simple averaging, i.e.

$$\text{Agg} = \{\mathbf{r}_c | c = 1, \ldots, C\} \mapsto \frac{1}{C}\sum_{c=1}^{C} \mathbf{r}_c. \tag{2.21}$$

Thus, we can concisely represent the encoder of the CNP as

$$\mathbf{r}(\mathcal{D}_c) = \frac{1}{C}\sum_{c=1}^{C} \phi\left([\mathbf{x}_c; y_c]\right) \in \mathbb{R}^{d_{\text{enc}}}. \tag{2.22}$$

DECODER AND LIKELIHOOD. For simplicity, we assume that the likelihood function is Gaussian, i.e.,

$$p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right) = \mathcal{N}(y; \mu(\mathbf{x}, \mathbf{r}), \sigma^2(\mathbf{x}, \mathbf{r})). \tag{2.23}$$

This assumption is not necessary, but is common in the NP literature.[5] In this case, the decoder is a function of the form $d_{\boldsymbol{\theta}}\colon \mathbb{R}^{d_{\text{enc}}} \times \mathcal{X} \to \mathbb{R} \times \mathbb{R}_+$, parametrised by a MLP that accepts as input the concatenation $[\mathbf{r}(\mathcal{D}_c); \mathbf{x}]$, and outputs (μ, σ), the parameters of the predictive distribution. In practice, we typically treat the output of $d_{\boldsymbol{\theta}}$ as $(\mu, \log\sigma)$, so as to enforce non-negative variances. A schematic of a forward pass through the CNP is illustrated in Figure 2.8.

2.5.3 *The Attentive Conditional Neural Process*

It has been noted by several authors that the representational power of standard CNPs tends to scale poorly with the capacity of the networks ρ and ϕ (Kim et al.,

5 For $\mathbf{y}$ multivariate, we may use a factorised Gaussian distribution of the form $p_{\boldsymbol{\theta}}\left(\mathbf{y}|\mathbf{x}, \mathbf{r}(\mathcal{D}_c)\right) = \mathcal{N}\left(\mathbf{y}; \boldsymbol{\mu}, \text{diag}\left(\boldsymbol{\sigma}^2\right)\right)$, where $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{d_y}$, and $\text{diag}(\cdot)$ generates a diagonal matrix from a vector.

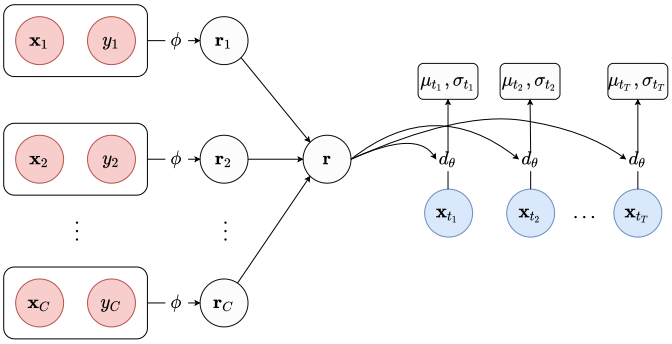


Figure 2.8: Schematic illustration of the CNP. First, each element in the context set is locally encoded using ϕ. The resulting representations are then averaged to produce $\mathbf{r}(\mathcal{D}_c)$. Finally, for each target location of interest, $(\mathbf{r}(\mathcal{D}_c), \mathbf{x}_t)$ are passed through the decoder $\mathrm{d}_{\boldsymbol{\theta}}$ to produce the predictive parameters.

2019; Louizos et al., 2019; Gordon et al., 2020a). As a result, CNPs tend to severely under-fit the modelled SP. Rather than increasing the capacity of ϕ and $d_{\boldsymbol{\theta}}$, a more promising approach is to introduce appropriate inductive biases in the parametrisation.

A powerful idea is to consider *target-specific* representations of the form $\mathbf{r}(\mathcal{D}_c, \mathbf{x}_t)$. This was first proposed by Kim et al. (2019), who introduced the *attentive* CNP (AttnCNP). The AttnCNP replaces the CNP mean aggregation of the local encodings with an *attention mechanism* (Bahdanau et al., 2015; Xu et al., 2015; Vaswani et al., 2017). Intuitively, this enables the model to place larger "weight" on points in the context set that are more relevant to the predictive distribution of a particular target location $\mathbf{x}_t$. More precisely, given the local encodings $\{\mathbf{r}_c\}_{c=1}^C$, we can express the representation for a target input as

$$\mathbf{r}_t := \mathbf{r}(\mathcal{D}_c, \mathbf{x}_t) = \sum_{c=1}^{C} \frac{w_{\boldsymbol{\theta}}\left(\mathbf{x}_c, \mathbf{x}_t\right) \mathbf{r}_c}{\sum_{c'=1}^{C} w_{\boldsymbol{\theta}}\left(\mathbf{x}_{c'}, \mathbf{x}_t\right)}, \tag{2.24}$$

where $w_{\boldsymbol{\theta}} \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a learned attention function. An illustration of the AttnCNP is provided in Figure 2.9. Equation (2.24) highlights two important points regarding the attentive representations:

1. they induce a strict generalisation of the standard CNP, recovering it when $w_{\boldsymbol{\theta}}(\cdot, \cdot)$ is the constant function.

2. they satisfy permutation invariance, as they are produced by a linear combination of the context-set representations.

Though not necessary, the AttnCNP is often used in conjunction with *self-attention* layers (Vaswani et al., 2017; Parmar et al., 2018) for the local encoder ϕ. As demonstrated by Kim et al. (2019) and verified later in this thesis, the AttnCNP achieves significant performance gains over the CNP, and will generally

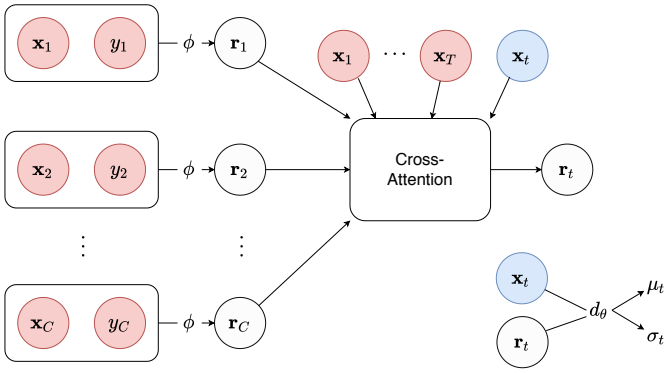


Figure 2.9: Schematic illustration of the AttnCNP. First, each element in the context set is locally encoded using ϕ. The resulting representations are then passed through a *cross-attention* mechanism, together with $\mathbf{X}_C$ and a target location $\mathbf{x}_t$, to produce the target representation $\mathbf{r}_t$. Finally, for each target location of interest, $(\mathbf{r}_t, \mathbf{x}_t)$ are passed through the decoder $\mathrm{d}_{\boldsymbol{\theta}}$ to produce the predictive parameters. Visualisation structure borrowed from Kim et al. (2019).

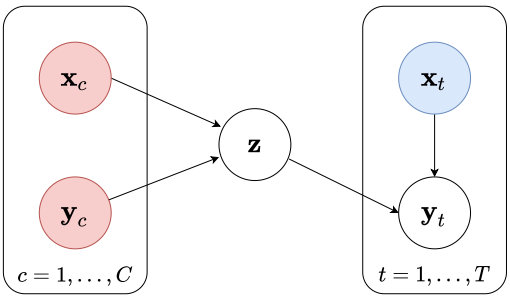


Figure 2.10: Graphical model representation for the latent NP sub-family. The conditional distribution of the latent variable $\mathbf{z}$ depends on the context set $\mathcal{D}_c$. Given $\mathbf{z}$, the predictive distribution over the target set $\mathcal{D}_t$ factorises over the target locations.

be considered the benchmark for comparison in this thesis when proposing new CNPF members for regression.

2.6 THE LATENT-VARIABLE NEURAL-PROCESS SUB-FAMILY

In Section 2.5, we constructed members of the NPF by incorporating a *factorisation* assumption on the predictive distribution. We further noted that the factorisation assumption results in models that have several important drawbacks. In particular, members of the CNPF (i) are unable to produce coherent samples without resorting to auto-regressive sampling schemes, and (ii) require the specification of parametric forms for the likelihood function, which in turn require domain expertise to design appropriate forms.

In this section, we consider an alternative approach to constructing members of the NPF, by introducing a *latent variable* $\mathbf{z}$. The central assumptions for this class of models, depicted in Figure 2.10, are that (i) the distribution of the

latent variable depends only on $\mathcal{D}_c$, and (ii) the predictive distribution over $\mathbf{y}_T$ factorises conditioned on instantiations of $\mathbf{z}$. We can express the resulting predictive distribution as

$$p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right) = \int p_{\boldsymbol{\theta}}\left(\mathbf{y}_T, \mathbf{z}|\mathbf{X}_T, \mathcal{D}_c\right) \mathrm{d}\mathbf{z} \tag{2.25}$$

$$= \int p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) \prod_{(\mathbf{x},y)\in\mathcal{D}_t} p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{z}\right) \mathrm{d}\mathbf{z}. \tag{2.26}$$

We can characterise members of the LNPF according to their two main components:

- the *encoder* $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right)$, which accepts as input context sets $\mathcal{D}_c$, and outputs a *distribution* over the latent variable $\mathbf{z}$.
- the *decoder* $p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{z}\right)$, which outputs predictive distributions conditioned on input locations $\mathbf{x}$ and latent variables $\mathbf{z}$.

The design of the encoder will again follow the principles of the NPF (i.e. local encoding of each $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_c$), followed by an aggregation. However, here we will use these principles to model a conditional *distribution* over the latent variable, rather than a deterministic representation.

2.6.1 *On the Latent Variable Parametrisation*

We first demonstrate that the predictive distributions arising from the latent-variable parametrisation respect the consistency requirements of the KET. To see that these predictive distributions are consistent under permutations, let $\mathbf{X}_T = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ be the input targets, and let π be a permutation of $\{1, \ldots, T\}$. The predictive density can be expressed as

$$\begin{aligned} p_{\boldsymbol{\theta}}\left(y_1, \ldots, y_T|\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathcal{D}_c\right) &= \int p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}\left(y_t|\mathbf{x}_t, \mathbf{z}\right) \mathrm{d}\mathbf{z} \\ &= p_{\boldsymbol{\theta}}\left(y_{\pi(1)}, \ldots, y_{\pi(T)}|\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(T)}, \mathcal{D}_c\right), \end{aligned}$$

where here too all that is required is the commutative property of multiplication. To demonstrate consistency under marginalisation, we again consider two target inputs $\mathbf{x}_1$ and $\mathbf{x}_2$. Marginalising the second target output y_2 we have

$$\begin{aligned}
&\int p_{\boldsymbol{\theta}}\left(y_1, y_2 |\ \mathbf{x}_1, \mathbf{x}_2, \mathcal{D}_c\right) \mathrm{d}y_2 \\
&\qquad = \iint p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) p_{\boldsymbol{\theta}}\left(y_1|\mathbf{x}_1, \mathbf{z}\right) p_{\boldsymbol{\theta}}\left(y_2|\mathbf{x}_2, \mathbf{z}\right) \mathrm{d}\mathbf{z}\mathrm{d}y_2 \\
&\qquad = \int p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) p_{\boldsymbol{\theta}}\left(y_1|\mathbf{x}_1, \mathbf{z}\right) \int p_{\boldsymbol{\theta}}\left(y_2|\mathbf{x}_2, \mathbf{z}\right) \mathrm{d}y_2\mathrm{d}\mathbf{z} \\
&\qquad = \int p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) p_{\boldsymbol{\theta}}\left(y_1|\mathbf{x}_1, \mathbf{z}\right) \mathrm{d}\mathbf{z} \\
&\qquad = p_{\boldsymbol{\theta}}\left(y_1|\mathbf{x}_1, \mathcal{D}_c\right),
\end{aligned}$$

which demonstrates that the predictive distribution obtained by querying an LNPF member at $\mathbf{x}_1$ is the same as that obtained by querying it at $(\mathbf{x}_1, \mathbf{x}_2)$, and then marginalising out the second target point. We can easily generalise this argument to collections of any size, and marginalising any subset of the variables.

Moreover, the latent-variable parametrisation introduces two important advantages for the LNPF. First, $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$ now has dependencies across target locations, induced by the marginalisation of $\mathbf{z}$. This makes the predictive distribution more expressive, and also enables producing coherent samples via simple ancestral sampling. Second, $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$ is no longer restricted to a parametric form. In fact, by noticing that the predictive is an infinite mixture of Gaussians, this form can recover any distribution,[6] relieving users of the need to explicitly designing likelihood functions that suit the application domain. However, these advantages come at a cost. In particular, for non-linear decoders $p_{\boldsymbol{\theta}}(y|\mathbf{x}, \mathbf{z})$, we can no longer evaluate $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$ in closed form. This has important implications for training members of the LNPF, which we discuss next.

2.6.2 *Neural Process Variational Inference (NPVI)*

Due to the intractability of $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$, we must consider alternatives to Algorithm 1 for training LNPs. In this section, we discuss a procedure proposed by Garnelo et al. (2018b). The algorithm is inspired by variational inference (Jordan et al., 1999; Wainwright and Jordan, 2008) employing inference networks (Gershman and Goodman, 2014; Kingma and Ba, 2015; Rezende and Mohamed, 2015; Zhang et al., 2018), and focuses on approximating the posterior distribution of the latent variable $\mathbf{z}$. In Section 2.7.3, we will revisit this procedure, and propose

6 This argument must be qualified somewhat, but is similar to that motivating the parametrisation of variational auto-encoders (Kingma and Welling, 2014; Rezende and Mohamed, 2015).

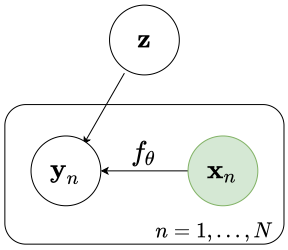


Figure 2.11: Graphical model representation for the "idealised" latent-variable NP. In this model, there is no distinction between context and target sets, and the posterior distribution $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\{(\mathbf{x}_n, y_n)|n = 1, \ldots, N\}\right)$ given observed data is defined directly via Bayes' theorem. We use different colour schemes here to distinguish this model from the ones used in practice in the NP family.

a simplified algorithm that abandons approximate inference of the latent-variable in favour of a more direct approximate maximum-likelihood procedure.

Garnelo et al. (2018b) consider the following generative model for data:

$$\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}); \tag{2.27}$$

$$y(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}; \mathbf{z}); \tag{2.28}$$

$$p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}) = \prod_{(\mathbf{x},y)\in\mathcal{D}_t} \mathcal{N}\left(y; f_{\boldsymbol{\theta}}(\mathbf{x}; \mathbf{z}), \sigma_y^2\right), \tag{2.29}$$

for some $\sigma_y \in \mathbb{R}_+$ and $f_{\boldsymbol{\theta}} \colon \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ (illustrated in Figure 2.11). Note that this model differs from the model used to define the LNPF (e.g. as in Figure 2.10 or Garnelo et al., 2018b, Figure 1a). In particular, this model makes no distinction between context and target sets, and instead treats all observed data equivalently. As a result, the quantity $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$ is not part of the model definition, but rather is implicitly defined via Bayes' rule. For any $S \in \mathcal{S}$, we have

$$p_{\boldsymbol{\theta}}\left(\mathbf{z}|S\right) = \frac{p_{\boldsymbol{\theta}}\left(\mathbf{z}\right) \prod_{(\mathbf{x},y)\in S} p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{z}\right)}{\int p_{\boldsymbol{\theta}}\left(\mathbf{z}\right) \prod_{(\mathbf{x},y)\in S} p_{\boldsymbol{\theta}}\left(y|\mathbf{x}, \mathbf{z}\right) \mathrm{d}\mathbf{z}}. \tag{2.30}$$

For general, non-linear decoders this posterior is intractable.

Using ideas from amortised VI (Kingma and Welling, 2014; Rezende and Mohamed, 2015), we can introduce an *inference network* $q_{\boldsymbol{\phi}} \colon \mathcal{S} \to \mathcal{P}(\mathcal{Z})$ with *variational parameters* $\boldsymbol{\phi} \in \Phi$ to approximate the true posterior distribution under the model (Dayan et al., 1995; Hinton et al., 1995; Gershman and Goodman, 2014). Note that $q_{\boldsymbol{\phi}}$ maps data sets to distributions over the latent variable $\mathbf{z}$, which is a familiar form in the NPF. Keeping in mind the design principles of Section 2.4.1, a typical specification for $q_{\boldsymbol{\phi}}$ employs a set-encoder that outputs the parameters of a factorised, multivariate normal distribution to parametrise $q_{\boldsymbol{\phi}}$. This is equivalent

to using a CNPF member as an inference network. Then, denoting $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_t$, we can derive the following *lower bound*:

$$\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c) = \log \int p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})} p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}) \mathrm{d}\mathbf{z} \tag{2.31}$$

$$\geq \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}) - \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})}\right], \tag{2.32}$$

where in Equation (2.32) we have used Jensen's inequality.

Equation (2.32) is an appealing objective to use for learning, as it would be well motivated both in training the model parameters $\boldsymbol{\theta}$, as well as performing approximate inference for $\mathbf{z}$ given a data set $S \in \mathcal{S}$. Unfortunately, Equation (2.32) requires evaluating the intractable $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right)$, (expressed in Equation (2.30)). Thus, Garnelo et al. (2018b) propose to substitute this term with the tractable inference network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D}_c) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$, yielding the following objective function:

$$\mathcal{L}_{\text{NPVI}}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi\right) \coloneqq \mathop{\mathbb{E}}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}) - \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D}_c)}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})}\right]. \tag{2.33}$$

Equation (2.33) can be estimated without bias and with low variance using Monte-Carlo sampling, i.e. $\mathbf{z}_l \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D})$. While tractable, Equation (2.33) is no longer a valid lower-bound for $\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c)$ in the model described in Equations (2.27) to (2.29), and thus lacks principled justification.

We can however, relate the $\mathcal{L}_{\text{NPVI}}$ directly to the standard LNPF model (Figure 2.10). To so, we identify $q_{\boldsymbol{\phi}}$ with the encoder of a LNP, i.e., with the model component $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$. This leads to a subtly different modelling interpretation: the encoder of the LNP is now both *part of the model* as well as an approximation to the true posterior under the model. With this interpretation, $\mathcal{L}_{\text{NPVI}}$ constitutes a valid ELBO for any task ξ. While admissible from a VI perspective, it is unclear what the implications of this dual role for the encoder are in practice. In Section 2.7.3, we introduce a simplified procedure for training LNPs, which directly targets the intractable likelihood. The relationship between this proposed approach and $\mathcal{L}_{\text{NPVI}}$ will provide further insight into the potential drawbacks of the VI-inspired approach.

2.6.3 *Latent-Variable Neural Process Models*

Armed with a training procedure for LNPF members, we now turn our attention to the models. In the following sections, we discuss the *Neural Process* (Garnelo et al., 2018b) and the *Attentive Neural Process* (ANP; Kim et al., 2019), the latent-variable counterparts of the conditional models introduced in the previous chapter.

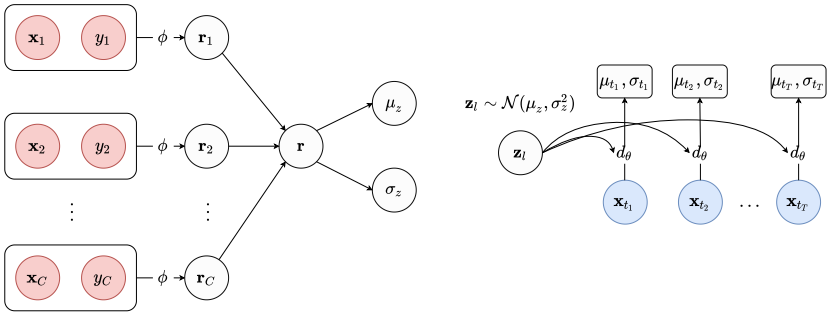


Figure 2.12: Schematic illustration of the NP. As with the CNP, each element in the context set is locally encoded using ϕ. The resulting representations are then averaged to produce $\mathbf{r}(\mathcal{D}_c)$. Then, $\mathbf{r}(\mathcal{D}_c)$ is passed through an additional network to produce the parameters of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$. Given a sample $\mathbf{z}_l \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$ and a target location $\mathbf{x}_t$, the decoder $\mathrm{d}_{\boldsymbol{\theta}}$ produces the predictive parameters.

Neural Processes

The *Neural Process* (NP; Garnelo et al., 2018b; Eslami et al., 2018) is the latent counterpart of the CNP, and is the first member of the LNPF proposed in the literature. The idea is quite simple: given the vector $\mathbf{r}(\mathcal{D}_c)$, which is computed in the same way as for the CNP, we simply pass it through an additional MLP to output the mean and variance of the latent variable $\mathbf{z}$, specifying $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$. We often refer to the computational graph that produces distributions over $\mathbf{z}$ as the *latent path*. Conversely, the computational graph that produces the deterministic representation is often referred to as the *deterministic path*. The decoder then has the same structure as that of the CNP, $d_{\boldsymbol{\theta}} : (\mathbf{x}_t, \mathbf{z}) \mapsto (\boldsymbol{\mu}_t, \log \boldsymbol{\sigma}_t)$, where $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$. A schematic of the NP is provided in Figure 2.12.

Attentive Neural Processes

The *Attentive Neural Process* (ANP; Kim et al., 2019) is the latent counterpart of the AttnCNP. In contrast to the NP, Kim et al. (2019) propose *adding* a latent path to the AttnCNP, rather than *replacing* the deterministic path. The latent path uses the same parametrisation as the NP. Thus, the ANP employs a *target-specific* deterministic path (cross-attention), and a *target-independent* latent path.

2.7 TECHNICAL CONTRIBUTIONS TO THE NEURAL PROCESS FAMILY

We conclude the chapter with several novel contributions to the theory of the NPF. In particular, we present the following contributions:

1. An analysis of the recovered objects when training members of the NPF with maximum-likelihood in suitable limits (Section 2.7.1).

2. An extension of Theorem 2.2 to vector-valued sets of any sizes, which in turn yields a universal representation theorem for CNPs (Section 2.7.2).

3. A simplified (approximate) maximum-likelihood training procedure for members of the LNPF, and an analysis relating it to the NPVI procedure proposed by Garnelo et al. (2018b) (Section 2.7.3).

2.7.1 *Maximum Likelihood – Idealised Case*

To better understand the implications of maximum-likelihood training in the NPF, we consider the following idealised scenario. Rather than access to a meta-data set Ξ, we assume the following generative process for data. First, some finite number of input locations, $\mathbf{X}_C, \mathbf{X}_T$ are sampled. Next, we sample $\mathbf{y}_t, \mathbf{y}_c$ from the finite marginal of the ground truth stochastic process P, which has density $p(\mathbf{y}_t, \mathbf{y}_c | \mathbf{X}_t, \mathbf{X}_c)$. Finally, we set $(D_c, D_t) := ((\mathbf{X}_C, \mathbf{y}_C), (\mathbf{X}_T, \mathbf{y}_T))$. With these definitions in place, we can state the following proposition, which shows that maximum-likelihood training recovers the prediction map π_P of the underlying process P (Definition 2.2) in a suitable limit.

Proposition 2.1
Let $\Psi : \mathcal{S} \to \mathcal{P}(\mathcal{X})$ be any map from data sets to stochastic processes, and let $\mathcal{L}_{\mathrm{ML}}(\Psi) := \mathbb{E}_{p(\xi)}[\log p_\Psi(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)]$, where the density p_Ψ is that of $\Psi(\mathcal{D}_c)$ evaluated at $\mathbf{X}_T$. Then Ψ globally maximises $\mathcal{L}_{\mathrm{ML}}$ if and only if $\Psi = \pi_P$, the prediction map.

Remark 2.3
We assume $\Pr(T = n) > 0$ for all $n > 0$, where $T := |\mathbf{X}_T|$ denotes the number of data points in $\mathbf{X}_T$, and similarly for $C = |\mathbf{X}_C|$. Further, we assume that for each $n > 0$, the distribution of $\mathbf{X}$ given $|\mathbf{X}| = n$ has a continuous density with support over all of $\mathbb{R}^{n \times d_x}$.

Proof.
We have:

$$\mathcal{L}_{\mathrm{ML}}(\Psi) = \mathop{\mathbb{E}}_{p(\mathcal{D}_c, \mathbf{X}_T, \mathbf{y}_T)} [\log p_\Psi(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)] \tag{2.34}$$

$$= \mathop{\mathbb{E}}_{p(\mathcal{D}_c, \mathbf{X}_T)} \left[\mathop{\mathbb{E}}_{p(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)} [\log p_\Psi(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c)] \right] \tag{2.35}$$

$$= - \mathop{\mathbb{E}}_{p(\mathcal{D}_c, \mathbf{X}_T)} [\mathrm{KL}(p(\mathbf{y}_T | \mathbf{X}_T, \mathcal{D}_c) \parallel p_\Psi(\boldsymbol{y} | \mathbf{X}_T, \mathcal{D}_c))] + \text{constant}, \tag{2.36}$$

where the additive constant is constant with respect to Ψ, and $\mathrm{KL}(\cdot \parallel \cdot)$ is the Kullback-Leibler divergence between two distributions (Kullback, 1959). First note that the KL-divergence is non-negative (Kullback, 1959; MacKay, 2003), and that the prediction map sends all the KL-divergences to zero, globally

optimising $\mathcal{L}(\Psi)$. Furthermore, the KL-divergence is equal to zero if and only if the two distributions are equal, and this must hold for all $\mathbf{X}_t, D_c$. For, if this were not the case, the KL-divergence would contribute a non-zero amount to the expectation in Equation (2.36). □

LIMITATIONS Proposition 2.1 tells us that with infinite capacity models, global optimisation procedures, and infinite data, maximum-likelihood training of NPs will recover the prediction map of the underlying process P. This is useful in motivating maximum-likelihood training, and understanding what we are attempting to recover with this training procedure. In particular, Proposition 2.1 demonstrates that maximum-likelihood training is sensible with an expressive model and sufficient data. This is very similar to standard statements regarding maximum-likelihood training, but adapted to the meta-learning setting. However, several assumptions are violated in practice, and may have important implications for the resulting models.

First, in practice we should assume neither infinite capacity nor infinite data. The proof of Proposition 2.1 reveals that in such cases, the resulting model minimises the average KL-divergence between π_P and the model being trained (still assuming global optimisation procedures). Intuitively, we are looking for the model that is "closest" to π_P in the KL sense over our data. Second, Proposition 2.1 assumes the generating process has non-negative support over context sets with arbitrarily many observations, and full support on $\mathcal{X}$. However, in practice we typically have access to data sets of limited size, as well as bounded support on $\mathcal{X}$. Hence, we should only expect the model to learn reasonable predictions within the ranges encountered during training.

2.7.2 *On the Universality of Conditional Neural Processes*

Let us now consider the class of functions that may be represented by members of the CNPF. Our goal will be to use the results of Zaheer et al. (2017) to determine the representation power of the CNP architecture. However, to be compatible with our goals, we must first extend Theorem 2.2 to (i) allow for sets of vector-valued objects, and (ii) handle varying-sized sets.[7]

Theorem 2.3 (Vector-valued Deep Sets)
Let $\mathcal{X} \subset \mathbb{R}^d$ be compact, and let $\mathbf{x}$ denote a generic element of $\mathcal{X}$. Let $\mathcal{X}_m = \underbrace{\mathcal{X} \times \mathcal{X} \ldots \mathcal{X}}_{m \text{ times}}$, $[\mathcal{X}_m]$ be the set of equivalence classes of elements of $\mathcal{X}$ under permutations, and denote X_m an element (set) in $\mathcal{X}_m$. Let $[\mathcal{X}]_{\leq M} = \cup_{m=1}^{M} [\mathcal{X}_m]$, and let T be a topological space.

7 The theorem and proofs rely on basic concepts related to reproducing kernel Hilbert spaces (RKHS; Sejdinovic and Gretton, 2012) and quotient-space topology (Munkres, 1974). In Appendices A and B we provide a brief review of the required concepts.

Then, any map $f\colon [\mathcal{X}]_{\leq M} \to T$*, such that its restrictions* $f|_{[\mathcal{X}_m]}$ *are continuous for all* $1 \leq m \leq M$ *has the form*

$$f\left(X_m\right) = \rho\left(E(X_m)\right); \qquad E(X_m) = \sum_{i=1}^{m} \phi(\mathbf{x}_i), \tag{2.37}$$

for some continuous $\rho\colon \mathbb{R}^{2M} \to T$ *and continuous* $\phi\colon \mathcal{X} \to \mathbb{R}^{2M}$.

A proof of Theorem 2.3 is provided in Appendix C. The central idea is to demonstrate that the mapping $E(X_m)$ homomorphically encodes sets X into a vector space, and that the mapping $f \circ E^{-1}$ has a form that can be represented by a neural network. Thus, by appropriately setting ρ, we have $f = \rho \circ E$. This proof structure is inspired by the work of Zaheer et al. (2017), and is similar in spirit (though less general) to ideas used by Bloem-Reddy and Teh (2020). In Chapter 3 we use a similar proof strategy to extend Theorem 3.1 to include translation-equivariant functions. There, we also provide a high-level sketch of the proof strategy invoked in both cases.

Informally, the theorem states that any function operating on vector valued sets of size less than or equal to M can be represented as a Deep Sets network with a representation space of dimensionality of $2M$. The use of continuous functions for ρ and ϕ motivates the use of neural networks to model them in practice.

One appealing property of Theorem 2.3 is that the dimensionality of the representation space, $2M$, does not depend on the dimensionality of $\mathcal{S}$. We note that Wagstaff et al. (2019) also considered the limitations of Theorem 2.2, and provide a useful and insightful analysis of its strengths and weaknesses. They too provide an extension of Theorem 2.2 to varying-sized sets, and are able to do so using only an M-dimensional representation. However, their statement is restricted to scalar elements, and it is not clear that extending their proof technique to vector-valued sets is straightforward. Next, we explicitly relate the Deep Sets architecture to CNPs, and leverage this relationship to quantify their representational power.

Definition 2.4 (Conditional Neural Processes)
Let $\mathcal{X} \subset \mathbb{R}^d$ *and* $\mathcal{Y} \subset \mathbb{R}$*, both compact, and let* $\mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ *be the space of bounded continuous functions from* $\mathcal{X}$ *to* $\mathcal{Y}$ *(endowed with the infinity norm). Let* $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$*, and let* $\mathcal{S}_m$*,* $[\mathcal{S}]_m$*, and* $[\mathcal{S}]_{\leq M}$ *be defined as in Theorem 2.3. Then, a map* $\Phi\colon [\mathcal{S}]_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ *is said to be a* conditional neural process *(CNP) if for all data sets* $[\mathbf{s}] \in [\mathcal{S}]_{\leq M}$ *and all* $\mathbf{x} \in \mathcal{X}$*,*

$$\Phi([\mathbf{s}])(\mathbf{x}) = \rho\left(\left[\sum_{(\mathbf{x},y)\in\mathbf{s}} \phi\left([\mathbf{x}; y]\right); \mathbf{x}\right]\right). \tag{2.38}$$

Here, $\phi\colon \mathbb{R}^{d+1} \to \mathbb{R}^{d_r}$ *continuous, where* d_r *is the* representation dimension *of the CNP, and* $\rho\colon \mathbb{R}^{d_r+d} \to \mathbb{R}$ *continuous.*

Note that Definition 2.4 employs a special case of the Deep Sets form (Equation (2.37)), where elements of the set are combined via concatenation before being passed to ϕ, and the decoder ρ accepts an additional input $\mathbf{x}$. With this definition in place, we can now state our representation theorem for CNPs.

Theorem 2.4 (CNP representation theorem)
Using the notation from Definition 2.4, let $\Phi\colon [\mathcal{S}]_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$*, with its restrictions* $\Phi|_{[\mathcal{S}]_m}$ *continuous for* $1 \leq m \leq M$*. Then,* Φ *can be represented as a conditional neural process with representation dimension* $d_r = 2M$*. Conversely, any CNP is a map from* $[\mathcal{S}]_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ *with continuous restrictions.*

A proof is provided in Appendix C. Theorem 2.4 tells us that we can use a CNP to represent any continuous map between data-sets and the space of continuous functions. Thus, for example, if we use a CNP to parametrise the mean and variance function of Gaussian distribution, our CNP can recover the marginal distributions of any SP with continuous first and second moments. Of course, Theorem 2.4 only states that there exist *some* continuous ρ and ϕ functions for which this is true. In practice we parametrise these with powerful neural networks, and allow the training procedure to learn approximations to these, relying on Proposition 2.1 for guarantees regarding the training procedure.

2.7.3 *Maximum-Likelihood Training for LNPF Members*

In Section 2.7.1, we saw that maximum-likelihood training of members of the NPF is justified by its properties in suitable limits. Following this line of reasoning, we now propose an alternative, approximate maximum-likelihood training procedure for members of the LNPF. We then relate this to the procedure proposed in Section 2.6.2.

Letting $\Xi = \{\xi_n\}_{n=1}^{N_{\text{tasks}}}$ be a *meta-training* set, we can train a member of the LNPF by stochastic gradient maximization of $\mathcal{L}_{\text{ML}}$ with tasks sampled from Ξ (Algorithm 1). Unfortunately, for non-linear decoders, $\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c)$ is intractable due to the expectation over $\mathbf{z}$ (Equation (2.26)). For a given task ξ, we propose optimizing the following Monte Carlo estimate of $\mathcal{L}_{\text{ML}}(\boldsymbol{\theta}; \xi)$:

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathcal{D}_c) = \log \int p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}\right) p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathcal{D}_c\right) \tag{2.39}$$

$$\approx \log \frac{1}{L} \sum_{l=1}^{L} p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathbf{z}_l\right), \tag{2.40}$$

with $\mathbf{z}_l \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c)$. Importantly, Equation (2.40) is the log of an unbiased estimator, and will thus tend to *underestimate* $\log p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathcal{D}_c)$ (i.e., it is a biased estimator). Plugging in Equation (2.26) and exponentiating log-likelihood terms, we can express Equation (2.40) as a simple log-sum-exp objective for optimization:

$$\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta};\xi) \coloneqq \log\left[\frac{1}{L}\sum_{l=1}^{L}\exp\left(\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T,\mathbf{z}_l)\right)\right]; \quad \mathbf{z}_l \sim p_{\boldsymbol{\theta}}(\mathbf{z}|\mathcal{D}_c), \tag{2.41}$$

where $\log p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T,\mathbf{z}_l)$ can be computed directly from the outputs of the decoder. Equation (2.41) is similar in spirit to importance-weighted likelihood objectives (e.g. Burda et al., 2015), and can be viewed as importance sampling in which the prior is the proposal distribution. Similarly to the IWAE objective proposed by Burda et al. (2015), Equation (2.41) is consistent and monotonically increasing (in expectation) in L.

Prior sampling is typically ineffective as it is unlikely to propose functions that pass near observed data. Here, however, $\mathrm{E}_{\boldsymbol{\theta}}$ depends on context sets $\mathcal{D}_c$, which often is sufficient to constrain prior function samples to be close to $\mathcal{D}_t$. One drawback of this objective is that single sample estimators are not useful, as they drive $\mathbf{z}$ to be deterministic. In Chapter 5, we demonstrate that, perhaps surprisingly, this estimator often significantly outperforms VI-inspired estimators.

Maximum-Likelihood vs NPVI Maximization for Training LNPF Members

We argue that the VI interpretation presented in Section 2.6.2 is unnecessary when focusing on predictive performance, and may be particularly detrimental for members of the LNPF for which $\mathbf{z}$ has many elements. Letting $Z = \int p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T,\mathbf{z})\,\mathrm{d}\mathbf{z}$ and $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_t$, we note the following equivalence

$$\mathcal{L}_{\mathrm{NPVI}}(\boldsymbol{\theta},\boldsymbol{\phi};\xi) = \mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta},\boldsymbol{\phi};\xi) - \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D}) \parallel p_{\boldsymbol{\theta}}(\mathcal{D}_t|\mathbf{z})\, q_{\boldsymbol{\phi}}(\mathbf{z}|\mathcal{D}_c)/Z). \tag{2.42}$$

We are explicitly distinguishing between the parameters of the encoder ($\boldsymbol{\phi}$) and decoder ($\boldsymbol{\theta}$). Thus, we can see that $\mathcal{L}_{\mathrm{NPVI}}$ is equal to $\mathcal{L}_{\mathrm{ML}}$ up to an additional KL term. This KL term encourages *context-set-consistency* among the $q_{\boldsymbol{\phi}}(\mathbf{z}|D)$ in the sense that Bayes' theorem is respected if the target set is subsumed into the context set. In the infinite capacity/data limit, $\mathcal{L}_{\mathrm{NPVI}}$ is globally maximized if the LNP recovers (i) the prediction map π_P for $\mathbf{y}_T$ and (ii) exact inference for $\mathbf{z}$. This follows from (i) Proposition 2.1, since π_P globally optimizes $\mathcal{L}_{\mathrm{ML}}$; and (ii) that exact inference for $\mathbf{z}$ is Bayes-consistent, sending the KL term to zero.

In most applications however, only the predictive distribution over $\mathbf{y}_T$ is of interest. Given only finite capacity/data, it can be advantageous to not expend capacity in enforcing context-set-consistency for $\mathbf{z}$, which suggests it could be beneficial to use $\mathcal{L}_{\mathrm{ML}}$ over $\mathcal{L}_{\mathrm{NPVI}}$. Further, $\mathcal{L}_{\mathrm{ML}}$ has the advantage of being easy

to specify for any map parametrising a predictive process, posing no conceptual issues for complex members of the LNPF that do not employ finite-dimensional latent variables (such models will be introduced in Chapter 5).

2.8 ADDITIONAL FLAVOURS OF META-LEARNING

The focus of this chapter, and indeed this thesis, is the formulation of the NPF for *supervised* meta-learning. In later chapters, we discuss additional approaches to supervised meta-learning, such as gradient-based (Finn and Levine, 2018; Nichol and Schulman, 2018), learned optimisation (Ravi and Larochelle, 2017), and metric-based few-shot learning (Snell et al., 2017). However, a recent surge of interest in meta-learning has lead to many additional flavours and applications which are not discussed in this thesis. Before concluding the chapter, we provide a brief overview of several important applications and approaches to meta-learning not discussed elsewhere in this thesis.

2.8.1 *Meta Reinforcement Learning*

An important example is meta-learning for reinforcement learning (RL; Sutton and Barto, 2018), which has seen rapid progress in recent years. In this setting, the central idea is to train an RL agent over a distribution of related tasks (modelled as Markov Decision Processes; MDPs) such that, at test time, the agent is able to learn to solve a new task rapidly. This idea has been proposed (at least) as early as Hochreiter et al. (2001), and has recently seen important successes using modern deep learning techniques (Wang et al., 2016; Duan et al., 2016). For an accessible and comprehensive review of meta-RL, we refer the reader to Weng (2019).

2.8.2 *Auto ML*

Another important application area of meta-learning not discussed in this thesis is *automated machine learning* (AutoML). AutoML is a subfield of machine learning that is chiefly concerned with the task of automating the design and implementation learning pipelines. Modern AutoML includes important undertakings such as hyper-parameter optimisation (Snoek et al., 2012; Domhan et al., 2015), neural architecture search (Zoph and Le, 2016; Casale et al., 2019), and learned optimisation (Wichrowska et al., 2017; Metz et al., 2018). AutoML research relies heavily on ideas from meta-learning (e.g., inner and outer optimisatoin loops, learning to generalise/optimise etc'), and the two fields are tightly linked. For comprehensive reviews of AutoML and related fields, we refer readers to Hutter et al. (2019) and Elsken et al. (2019).

2.8.3 *Unsupervised Meta Learning*

An additional and important class of meta-learning models and approaches not discussed in this thesis deals with unsupervised meta-learners. In this setting, the lack of labels requires a different formulation of the meta-learning problem. One approach, which is most closely related to the models discussed in this thesis, was proposed by Edwards and Storkey (2017). Here, the authors incorporated latent-variables directly into the graphical models, and focussed the modelling efforts on generative capacities.

More recently, the idea of pretraining increasingly large-scale networks on massive datasets has gained in popularity both in natural language and image modelling. Several authors, such as Brown et al. (2020) and Raffel et al. (2020), have demonstrated that maximum likelihood training of large-scale autoregressive models on massive collections of natural language can lead to the flavour of few-shot generalisation in test-time tasks that is among the central desiderata of meta-learning.

2.9 SUMMARY

In this chapter, we have introduced our perspective of (probabilistic) meta-learning, and leveraged this view to provide a systematic overview of the NPF and its two main sub-branches. Though a comprehensive review of models for few-shot and meta-learning is out of the scope of this thesis, related models are discussed in several sections throughout the chapters of the thesis. Moreover, in Section 7.2 we provide a thorough discussion regarding several models related to the NPF, focussing on the question of when a user may wish to employ one or the other. For a thorough review of meta- and few-shot-learning with neural networks, we refer readers to dedicated review articles such as the works of Hospedales et al. (2020) and Wang et al. (2020).

The remainder of the thesis builds upon and develops the ideas introduced in this chapter. In Chapter 3, we introduce the notion of *translation equivariance*, motivate its importance in the NPF, and develop the necessary machinery for its incorporation. Following this, Chapters 4 and 5 propose two new members of the NPF that incorporate this important inductive bias. Finally, Chapter 6 introduces yet another member of the NPF, which specialises the family to the *few-shot classification* setting. We conclude the thesis in Chapter 7 by reviewing key concepts in the NPF, providing practical advice for potential users, and outlining some ideas for future research.

3

TRANSLATION EQUIVARIANCE IN THE NPF AND CONVOLUTIONAL DEEPSETS

We have discussed how to parametrise and train mappings between data sets and stochastic processes using deep neural networks. We further discussed the idea of introducing inductive biases (e.g. attention) in the model parametrisation to achieve significant performance improvements. In this chapter, we introduce an additional inductive bias that turns out to be quite powerful for NPF members in appropriate situations: *translation equivariance*. The focus of this chapter is to provide the motivation and underlying theory for considering translation equivariance in the NPF. In Chapters 4 and 5 we build on this theory to propose translation equivariant NP models, and demonstrate that they significantly outperform their existing counterparts when the inductive bias is appropriate for the data.

3.1 INTRODUCTION

We begin the chapter by motivating the need for considering translation equivariant members of the NPF by (i) qualitatively studying the predictive distributions produced by existing members and noting several important limitations of the models, and (ii) establishing a concrete relationship between an important subclass of SPs—stationary SPs (Lindgren, 2012)—and translation equivariant mappings. Then, as functions on sets form the backbone of the NPF, we consider the more general question of representing translation equivariant functions on sets. This leads us to propose a novel framework, which we coin *Convolutional DeepSets*, that extends the work of Zaheer et al. (2017) and provides a universal representation theorem for functions of the desired form.

The results discussed in this chapter elaborate on work that was originally published in "Convolutional Conditional Neural Processes" (Gordon et al., 2020a). The research was conducted in collaboration with my co-first author Wessel Bruinsma, as well as Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. I was closely involved in all aspects of the project, including formulation, development of the theory and proof, and writing the paper.

The main contributions of this chapter are:

- motivating the incorporation of translation equivariance via qualitative (Sections 3.2.2 and 3.2.3) and quantitative (Section 3.4.2) analyses

- introduction of the Convolutional DeepSets framework, including a functional form that can be parametrised by neural networks (Equation (3.5))
- providing a universal representation theorem for translation equivariant functions on sets (Theorem 3.1)

3.2 IMPORTANT LIMITATIONS OF NPF MEMBERS

We begin by providing a qualitative analysis of the predictive distributions produced by the CNP and ATTNCNP. In particular, we focus on two important limitations of these members: a tendency to under-fit the data, and an inability to generalise in time or space. We will then argue that the use of translation equivariance as an inductive bias for the NPF will serve to alleviate these issues.

3.2.1 *Visualizing CNPF Predictive Distributions*

We introduce our Gaussian process experiments, which are used throughout the thesis to benchmark the performance of the NPF. These experiments are useful as they enable us to sample as many tasks as we desire, and compare the performance of the NP to the oracle GP. Recall that Proposition 2.1 and Theorem 2.4 imply that with enough capacity and data, a CNP trained with maximum-likelihood should be able to exactly recover the marginal predictions of the underlying GP, allowing us to benchmark performance against a "ground truth" object.

We can generate tasks ξ from a GP by first sampling a function, and evaluating it at a finite number of input locations to form $\mathcal{D}_c$ and $\mathcal{D}_t$. Training then follows Algorithm 1, where instead of a finite dataset Ξ, tasks are sampled (typically in batches) from the generative process. Beyond exponentiated quadratic (EQ) kernel GPs (as proposed in Garnelo et al. (2018a) and Kim et al. (2019)), we will consider more complex data arising from Matérn-$\frac{5}{2}$ and weakly-periodic kernels, as well as a challenging, non-Gaussian sawtooth process with random shift and frequency. Training and testing procedures are fixed across models throughout the thesis. Full details on models, data generation, and training procedures are provided in Appendix E. In this chapter we restrict ourselves to qualitative analyses of the trained models, but in later chapters consider quantitative analyses as well.

Figure 3.1 illustrates the predictive distributions of a trained CNP and ATTNCNP on an EQ kernel for a variety of context sets. The figure demonstrates that both models perform relatively well in this setting. We see that (i) as more data is observed, the predictions tend to become "tighter" (uncertainty decreases) as we might expect, and (ii) the model predictions are generally able to track the predictive functions output by the ground truth GP.

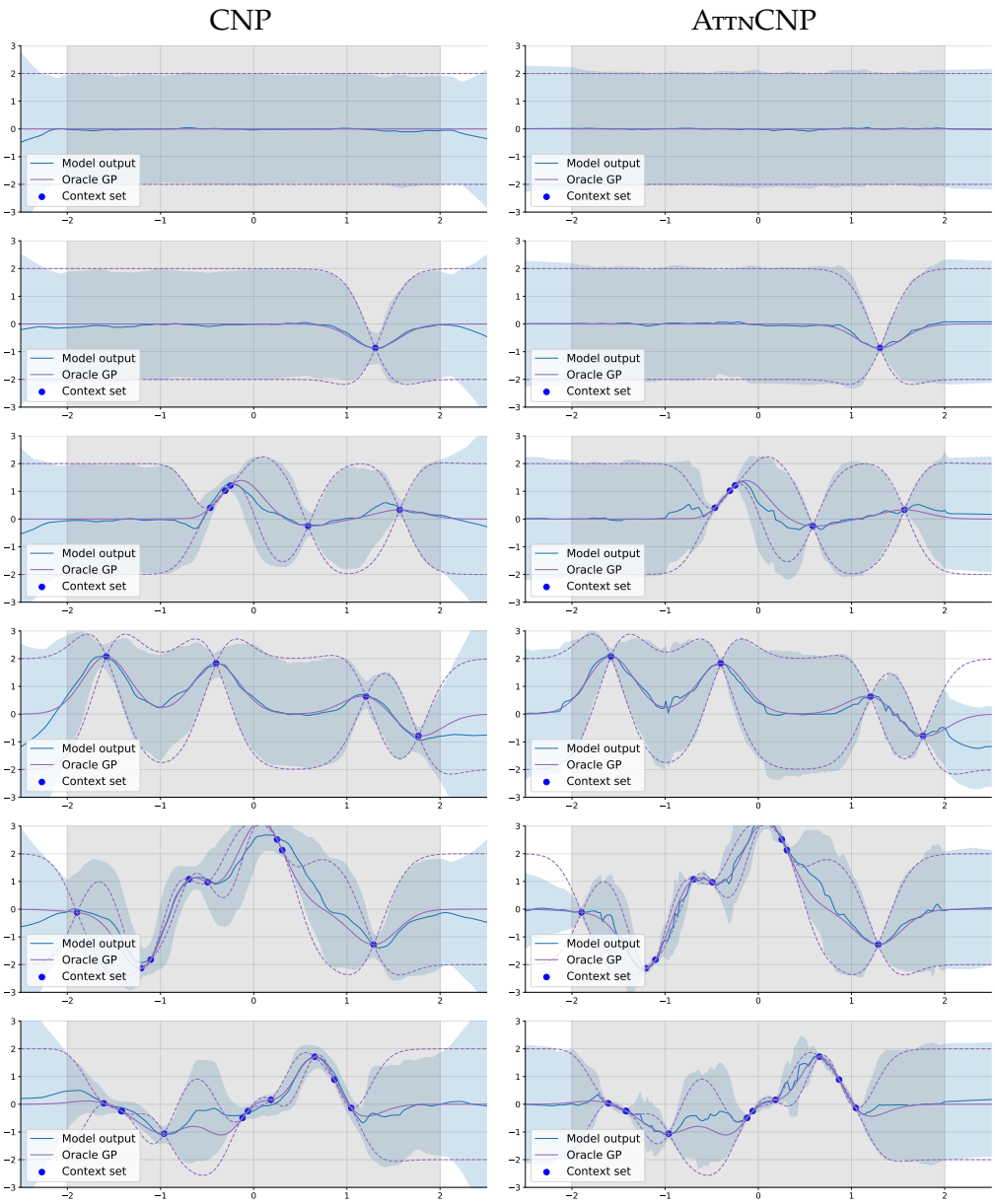


Figure 3.1: Qualitative analysis of the CNP (left column) and AttnCNP (right column) predictive distributions for a GP with an EQ kernel. Solid purple line is the oracle GP mean function, and dashed lines represent two standard deviations around the mean. The blue solid line is the NP model mean function, shaded blue region represents two standard deviations around the mean. The grey shaded region is the *training region*, i.e., where data was observed during training.

However, we can see some weaknesses of the models. For example, we see that the CNP mean function doesn't always pass through the observed data (despite there being no noise in the generating process), and tends to overestimate the uncertainty. The AttnCNP mean function does tend to pass through the observed data, but exhibits "kinks", certain locations where the function is not smooth. We next take a closer look at these issues and additional drawbacks of the models.

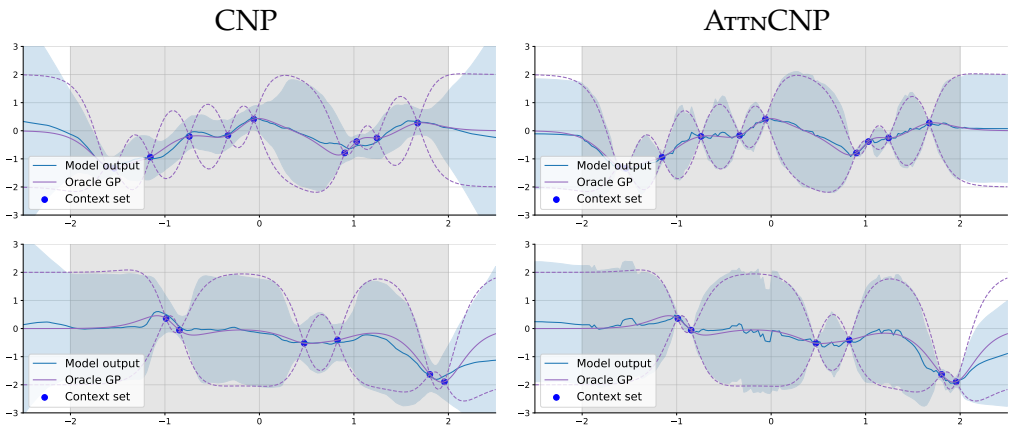


Figure 3.2: Qualitative analysis of the CNP (left) and AttnCNP (right) predictive distributions for a GP with a Matérn-$\frac{5}{2}$ kernel. Coloring is the same as for Figure 3.1. Here we see that the CNP predictive distribution underfits the ground truth predictive distribution. In particular, we see that the CNP struggles to shrink the uncertainty near observed data, and the variance function is far smoother than that of the ground truth distribution.

3.2.2 *Under-fitting the Prediction Map*

The CNP and AttnCNP produce reasonable predictive distributions when trained to mimic a GP with an EQ kernel. However, even in these simple cases, we see that the models tend to underfit the ground truth predictive distribution or produce mean and variance functions with unnatural "kinks". These becomes more apparent when considering more challenging kernels.

Figure 3.2 provides a visualisation of the predictive distributions of a CNP and AttnCNP trained with tasks from a Matérn-$\frac{5}{2}$ kernel. This kernel is more challenging for the models as (i) functions sampled from it are only twice-differentiable, and not as smooth as functions from the EQ kernel, and (ii) we choose a shorter length-scale for the Matérn-$\frac{5}{2}$ kernel, such that functions tend to have more "wiggles" in a given interval. Figure 3.2 demonstrates that here the CNP struggles to fit the true predictive well, producing overly-smooth variance functions and it fails to shrink the uncertainty near observed data. In contrast, the AttnCNP produces functions that closely track the ground truth distribution, but continues to produce unnatural "kinks" in both the mean and variance functions.

Finally, Figure 3.3 illustrates the same comparison, but with a weakly-periodic component in the kernel. Here we see that the CNP completely fails to model the predictive distribution in a meaningful way. The AttnCNP is able to track the mean and variance functions, but does not do so adequately. Moreover, the AttnCNP does not leverage the periodic structure in the data when making its predictions, which is significantly different from the ground truth predictive.

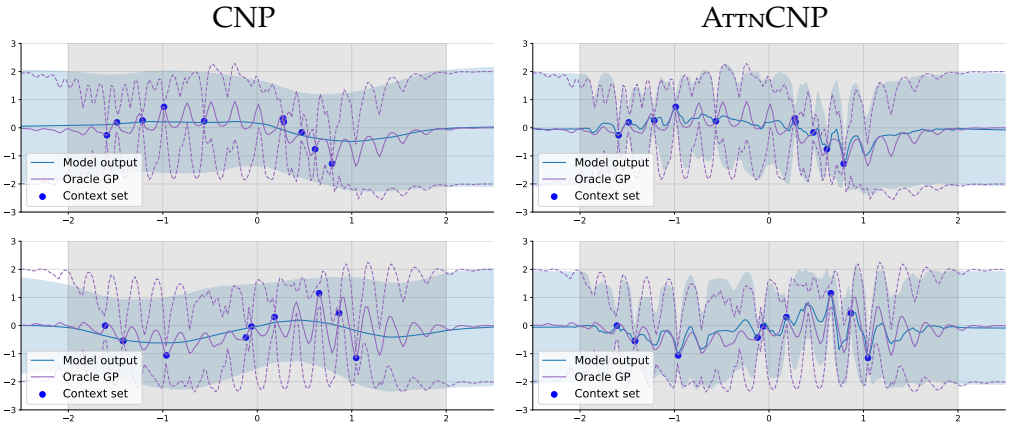


Figure 3.3: Qualitative analysis of the CNP (left) and AttnCNP (right) predictive distributions for a GP with a kernel containing a weakly-periodic component. Colouring is the same as for Figure 3.1. The CNP is unable to model the predictive distributions of this kernel in a meaningful way. The AttnCNP mean function interpolates $\mathcal{D}_c$, and the variance function roughly tracks the variance function of the underlying GP. However, neither captures the periodic structure of the data, and the AttnCNP fails to accurately track the true predictive distribution.

3.2.3 *Generalisation in Time and Space*

We have so far considered the predictive distributions produced by NPF members when data is observed in the same range at meta-training and "meta-test" time. However, in many domains models must be able to make predictions in regions unobserved during training (e.g. astronomical data (Boone, 2019), patient healthcare (Bates et al., 2014), and environmental applications (Delhomme, 1978)). We refer to this type of generalisation in time/space as *extrapolation*.

To study the extrapolation capacity of CNPF members, we consider the scenario where a trained model is conditioned on a context set containing observations outside the training range. An example of such a plot, using the EQ kernel GP, is illustrated in Figure 3.4. We see that when data are observed outside the training range, the predictive distributions produced by both models reduce dramatically in quality, though in subtly different ways. For the CNP, observing data outside the training range has a negative effect on the predictive distribution both inside and outside the training range. In contrast, the quality of the predictive distributions produced by the AttnCNP inside the training range is largely unaffected when observing data from outside the range. However, like the CNP, the AttnCNP fails to produce high quality predictions in the extrapolation range.

This failure mode is dissatisfying for two important reasons. First, it is a significant difference from the underlying process, for which the prediction map can generalise in this way without issue. Second, and importantly from a

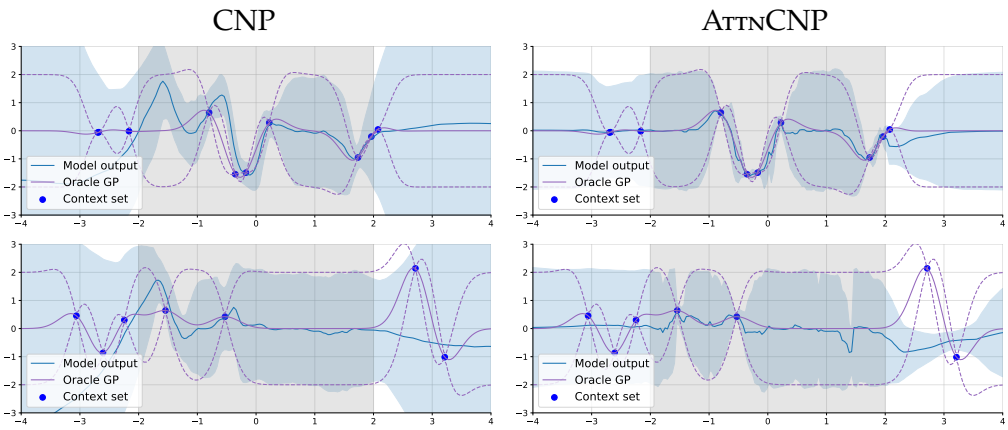


Figure 3.4: Evaluating the predictive distributions of the CNP (left) and AttnCNP (right) when data is observed outside the training range. Models were trained on a GP with an EQ kernel. In both cases and for both models, observations outside the training range cause the predictive distribution to be of poor quality. For the AttnCNP, the predictive distribution inside the training range remains of reasonable quality, while the distribution outside the training range is extremely poor. For the CNP, the predictive distribution both inside and outside the training range is poor when observing data outside the training range.

practical perspective, several important application domains require this form of generalisation (Roberts et al., 2013; Delhomme, 1978; Bates et al., 2014).

We have qualitatively demonstrated two important limitations of the introduced members of the CNPF: a tendency to severely underfit the prediction map, and an inability to generalise in time/space. We hypothesise that by introducing translation equivariance into the NPF, we can alleviate both of these limitations.

3.3 STATIONARITY

Consider the task of predicting rainfall at an unseen test location from rainfall measurements nearby. A powerful inductive bias for this task is *stationarity*. Informally, stationarity corresponds to the assumption that the generative process governing rainfall is spatially homogeneous. Given only observations in a limited part of the space, stationarity allows the model to extrapolate to yet unobserved regions. A more precise definition of stationarity, and *stationary stochastic processes* (Lindgren, 2012), is as follows.

Definition 3.1 (Translating Stochastic Processes)
We define the action of the translation operator $T_{\boldsymbol{\tau}}$ *on stochastic processes, where* $\boldsymbol{\tau} \in \mathcal{X}$ *denotes the shift vector of the translation. For a function* $f \in \mathbb{R}^{\mathcal{X}}$*, define* $T_{\boldsymbol{\tau}} f := f(\mathbf{x} - \boldsymbol{\tau})$ *for all* $\mathbf{x} \in \mathcal{X}$*. Let* $F \in \Sigma$ *be a measurable set of functions, then* $T_{\boldsymbol{\tau}} F := \{T_{\boldsymbol{\tau}} f : f \in F\}$*. For any SP* $P \in \mathcal{P}(\mathcal{X})$*, we define* $T_{\boldsymbol{\tau}} P(F) := P(T_{-\boldsymbol{\tau}} F)$ *for all* $F \in \Sigma$*.*

Definition 3.2 (Stationary Stochastic Processes)
We say a stochastic process is (strictly) stationary *if the densities of its finite marginals satisfy*

$$p(\mathbf{y}_T|\mathbf{X}_T) = p(\mathbf{y}_T|T_{\boldsymbol{\tau}}\mathbf{X}_T) \tag{3.1}$$

for all $\mathbf{y}_T$, $\mathbf{X}_T$, *and* $\boldsymbol{\tau}$, *where* $T_{\boldsymbol{\tau}}\mathbf{X}_T \coloneqq (\mathbf{x}_1 + \boldsymbol{\tau}, \ldots, \mathbf{x}_n + \boldsymbol{\tau})$.

When stationarity is appropriate, incorporating it as an inductive bias for our models yields significant benefits. This has been empirically demonstrated in multiple domains, e.g. in time-series (Roberts et al., 2013), images (LeCun et al., 1998), and spatio-temporal modelling (Delhomme, 1978; Cressie, 1990).

3.4 TRANSLATION EQUIVARIANCE

A notion closely related to stationarity is *translation equivariance*. Informally, translation equivariance is an inductive bias stating that if the data are translated in time or space, then the predictions should be translated correspondingly (Kondor and Trivedi, 2018; Cohen and Welling, 2016) An illustration of a translation equivariant mapping, implemented with a convolutional neural network and MNIST digits (LeCun et al., 1989), is provided in Figure 3.5.

Famously, convolutional neural networks (CNNs) endow MLPs with translation equivariance (LeCun et al., 1998; Cohen and Welling, 2016), accounting in large part for their success in domains such as image (Krizhevsky et al., 2012; He et al., 2016) and time-series modelling (Oord et al., 2016a). Conversely, when translation equivariance is appropriate for the domain, but not "baked into" the model, it must be learned directly from the data. This is sample and parameter inefficient, and has a negative impact on the ability of models to generalise (Kondor, 2008; Kondor and Trivedi, 2018; Ravanbakhsh et al., 2017).

3.4.1 *Translation Equivariant Mappings on Sets*

We now turn to a more formal definition of translation equivariance, and in particular the notion of translation equivariant mappings on sets. Recalling that we denote as $\mathcal{S}$ the space of supervised datasets, we can define translation equivariant mappings as follows.

Property 3.1 (translation equivariant mappings on sets)
Let $\mathcal{H}$ *be an appropriate space of functions on* $\mathcal{X}$, *and define* T *and* T' *as follows:*

$$T : \mathcal{X} \times \mathcal{S} \to \mathcal{S}, \qquad T_{\boldsymbol{\tau}}S = ((\mathbf{x}_1 + \boldsymbol{\tau}, \mathbf{y}_1), \ldots, (\mathbf{x}_m + \boldsymbol{\tau}, \mathbf{y}_m)), \tag{3.2}$$

$$T' : \mathcal{X} \times \mathcal{H} \to \mathcal{H}, \qquad T'_{\boldsymbol{\tau}}h(\mathbf{x}) = h(\mathbf{x} - \boldsymbol{\tau}). \tag{3.3}$$

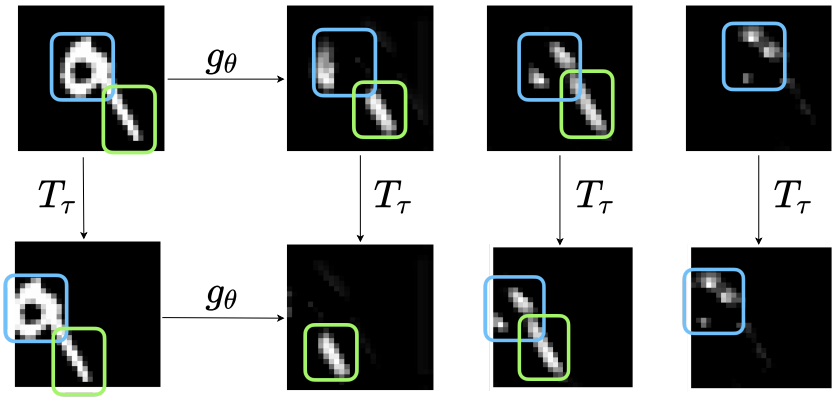


Figure 3.5: Illustration of translation equivariance for feature extractors of images. A digit from the MNIST test set (LeCun et al., 1989) (top left) is passed through a trained convolutional neural network $g_{\boldsymbol{\theta}}$. A subset of the final-layer feature maps are then illustrated (three top right images). A translated version of the digit (bottom left) is also passed through $g_{\boldsymbol{\theta}}$, and the same feature maps are illustrated (three bottom right images). The bounding boxes assist in illustrating that the feature representation of the translated digit are equivalent to the translated feature maps of the original image. The illustration is purposefully constructed analogously to a *commutative diagram* (Munkres, 1974; Kondor, 2008)

.

Then a mapping $\Phi : \mathcal{S} \to \mathcal{H}$ *is called* translation equivariant *if* $\Phi(T_{\boldsymbol{\tau}} S) = T'_{\boldsymbol{\tau}} \Phi(S)$ *for all* $\boldsymbol{\tau} \in \mathcal{X}$ *and* $S \in \mathcal{S}$.

Note that Property 3.1 is defined for mappings between sets and functions. We have seen how members of the CNPF define mappings to stochastic processes in this way. When considering translation equivariant members of the LNPF, we must extend this definition to translation equivariant mappings to stochastic processes, which will simply apply Definition 3.1.

Property 3.2 (translation equivariant mappings to stochastic processes) *Using Definition 3.1, we say that* $\Psi : \mathcal{S} \to \mathcal{P}(\mathcal{X})$ *is* translation equivariant *if* $\Psi(T_{\boldsymbol{\tau}} S) = T_{\boldsymbol{\tau}} \Psi(S)$ *for any data set* $S \in \mathcal{S}$ *and shift* $\boldsymbol{\tau} \in \mathcal{X}$.

3.4.2 *Relating Translation Equivariance and Stationarity*

Intuitively, it seems clear that there is a relationship between the notions of stationarity and translation equivariance. To make this relationship precise, we consider the prediction map (Definition 2.2) of a *stationary* stochastic process P. In this case, the prediction map π_P possesses two important symmetries. First, as discussed in previous chapters, π_P is *invariant* to permutations of $\mathcal{D}_c$ (Zaheer et al., 2017; Gordon et al., 2020a). This is true for any stochastic process, and motivates the use of permutation invariant parametrisations. Second, if the ground truth

process P is *stationary*, then π_P is *translation equivariant*, i.e., satisfies Property 3.2. This relationship is formalised in the following simple proposition.

Proposition 3.1
Let P be a stationary SP. Then the prediction map π_P is translation equivariant.

Proof.
Let $p(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c)$ denote the finite dimensional density of $\pi_P(\mathcal{D}_c)$ at index set $\mathbf{X}_T$. To show that $\pi_P(T_{\boldsymbol{\tau}}\mathcal{D}_c) = T_{\boldsymbol{\tau}}\pi_P(\mathcal{D}_c)$ it suffices to show that $p(\mathbf{y}_T|\mathbf{X}_T, T_{\boldsymbol{\tau}}\mathcal{D}_c) = p\left(\mathbf{y}_T|T_{-\boldsymbol{\tau}}\mathbf{X}_T, \mathcal{D}_c\right)$. We have

$$\begin{aligned} p\left(\mathbf{y}_T|\mathbf{X}_T, T_{\boldsymbol{\tau}}\mathcal{D}_c\right) &= \frac{p\left(\mathbf{y}_T, \mathbf{y}_C|\mathbf{X}_T, T_{\boldsymbol{\tau}}\mathbf{X}_C\right)}{p\left(\mathbf{y}_C|T_{\boldsymbol{\tau}}\mathbf{X}_C\right)} \\ &= \frac{p\left(\mathbf{y}_T, \mathbf{y}_C|T_{-\boldsymbol{\tau}}\mathbf{X}_T, \mathbf{X}_C\right)}{p\left(\mathbf{y}_C|\mathbf{X}_C\right)} \\ &= p\left(\mathbf{y}_T|T_{-\boldsymbol{\tau}}\mathbf{X}_T, \mathcal{D}_c\right), \end{aligned}$$

where we used the stationarity assumption in the second line. □

This simple statement highlights the intimate relationship between stationarity and translation equivariance. Moreover, it suggests that models for the prediction map should also be translation equivariance and permutation invariant. As such models are a small subset of the space of *all* models, building in these properties can greatly improve data efficiency and generalization for stationary SP prediction. In the next section, we introduce the workhorse for extending the NPF to include translation equivariance: convolutional DeepSet networks.

3.5 CONVOLUTIONAL DEEP SETS

We are interested in translation equivariance (Property 3.1) with respect to translations on $\mathcal{X}$. Recall from Section 2.3 that the form of a DeepSet network is

$$f(S) = \rho(E(S)); \qquad E(S) = \sum_{\mathbf{s}\in S} \phi(\mathbf{s}), \tag{3.4}$$

where ϕ and ρ are parametrised by neural networks. The DeepSets encoder maps sets S to an embedding in a vector space $\mathbb{R}^d$ (Zaheer et al., 2017), for which the notion of equivariance with respect to input translations in $\mathcal{X}$ is not well defined. For example, a function f on $\mathcal{X}$ can be translated by $\boldsymbol{\tau} \in \mathcal{X}$: $f(\cdot - \boldsymbol{\tau})$. However, for a vector $\mathbf{x} \in \mathbb{R}^d$, which can be seen as a function $[d] \to \mathbb{R}$, $\mathbf{x}(i) = x_i$, the translation $\mathbf{x}(\cdot - \boldsymbol{\tau})$ is not well-defined.

To overcome this issue, we enrich the encoder $E\colon \mathcal{S} \to \mathcal{H}$ to map into a *function space* $\mathcal{H}$ containing functions on $\mathcal{X}$. Since functions in $\mathcal{H}$ map from $\mathcal{X}$, our notion of translation equivariance (Property 3.1) is now also well defined for the DeepSets.

As we demonstrate below, every translation equivariant function on sets has a representation in terms of a specific functional embedding.

Definition 3.3 (Functional mappings on sets)
Call a map $E\colon \mathcal{S} \to \mathcal{H}$ *a* functional mapping on sets *if it maps from sets* $\mathcal{S}$ *to an appropriate space of functions* $\mathcal{H}$*. Furthermore, call* $E(S)$ *the* functional representation *of the set* S*.*

Considering functional representations of sets leads to the central result of this chapter, which can be summarized as follows. For $\mathcal{S}' \subset \mathcal{S}$ appropriate, a continuous function $\Phi\colon \mathcal{S}' \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ satisfies Properties 2.1 and 3.1 if and only if it has a representation of the form

$$\Phi(S) = \rho\left(E(S)\right), \quad E(S) = \sum_{(\mathbf{x},\mathbf{y})\in S} \phi(\mathbf{y})\psi(\cdot - \mathbf{x}) \in \mathcal{H}, \tag{3.5}$$

for some continuous and translation equivariant $\rho\colon \mathcal{H} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$, and appropriate ϕ and ψ. Note that ρ is a map between function spaces. We also remark that continuity of Φ is not in the usual sense; we return to this below.

In Section 3.5.1, we present our theoretical results in more detail. In particular, Theorem 3.1 establishes equivalence between any function satisfying Properties 2.1 and 3.1 and the representational form in Equation (3.5). In doing so, we provide an extension of the key result of Zaheer et al. (2017) to functional representations on sets, and show that it can naturally be extended to handle varying-size sets. In the next chapter, which deals with the practical implementation of translation equivariant NP models, the design of ρ, ϕ, and ψ is informed by our results in Section 3.5.1.

3.5.1 *Representations of Translation Equivariant Functions on Sets*

In this section we establish the theoretical foundation of ConvDeepSets. We begin by stating a definition that is used in our main result. Throughout this section, we use the notation $[m] := \{1, \dots, m\}$ for an integer m.

Definition 3.4 (Multiplicity)
A collection $\mathcal{S}' \subset \mathcal{S}$ *is said to have* multiplicity K *if, for every set* $S \in \mathcal{S}'$*, every* $\mathbf{x}$ *occurs at most* K *times:*

$$\operatorname{mult} \mathcal{S}' := \sup\left\{\sup\Big\{\underbrace{|\{i \in [m] : \mathbf{x}_i = \hat{\mathbf{x}}\}|}_{\text{number of times every } \mathbf{x} \text{ occurs}} : \hat{\mathbf{x}} = \mathbf{x}_1, \dots, \mathbf{x}_m\Big\} : (\mathbf{x}_i, y_i)_{i=1}^m \in \mathcal{S}'\right\} = K.$$

For example, in the case of real-world data like time series and images, we often observe only one (possibly multi-dimensional) observation per input location,

which corresponds to multiplicity one. We are now ready to state our the central result of this chapter.

Theorem 3.1 (ConvDeepSets Representation Theorem)
Consider an appropriate[1] *collection* $\mathcal{S}'_{\leq M} \subseteq \mathcal{S}_{\leq M}$ *with multiplicity* K*. Then a function* $\Phi\colon \mathcal{S}'_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ *is continuous,*[2] *permutation invariant (Property 2.1), and translation equivariant (Property 3.1) if and only if it has a representation of the form*

$$\Phi(S) = \rho\left(E(S)\right), \qquad E((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) = \sum_{i=1}^{m} \phi(y_i)\psi(\cdot - \mathbf{x}_i) \tag{3.6}$$

for some continuous and translation equivariant $\rho\colon \mathcal{H} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ *and some continuous* $\phi\colon \mathcal{Y} \to \mathbb{R}^{K+1}$ *and* $\psi\colon \mathcal{X} \to \mathbb{R}$*, where* $\mathcal{H}$ *is an appropriate*[3] *space of functions that includes the image of* E*. We call a function* Φ *of the above form a* ConvDeepSet*.*

A constructive proof of Theorem 3.1 is provided in Appendix D. A sketch of the proof is as follows. We begin by demonstrating that our proposed embedding into function space is homeomorphic (Lemmas D.1 and D.2). Then, we show that the embeddings of fixed-sized sets can be extended to varying-sized sets by "pasting" the embeddings together while maintaining their homeomorphic properties (Lemma D.3). Following this, we demonstrate that the resulting embedding may be composed with a continuous mapping to our desired target space, resulting in a continuous mapping between two metric spaces (Lemma D.4). Finally, we combine the above-mentioned results to prove Theorem 3.1.

Here, we discuss several key points from the proof that have practical implications and provide insights for the design of translation equivariant members of the NPF:

- For the construction of E, ψ is set to a flexible positive-definite kernel associated with a Reproducing Kernel Hilbert Space (RKHS; Aronszajn (1950)). This turns out to result in desirable properties for E that are extremely useful in proving Theorem 3.1.
- Following the proof of Zaheer et al. (2017), we set $\phi(y) = (y^0, y^1, \cdots, y^K)$ to be the powers of y up to order K.
- The construction invoked in Appendix D requires ρ to be a powerful function approximator of continuous, translation equivariant maps between functions.

1 For every $m \in [M]$, $\mathcal{S}'_{\leq M} \cap \mathcal{S}_m$ must be topologically closed and closed under permutations and translations.
2 For every $m \in [M]$, the restriction $\Phi|_{\mathcal{S}'_{\leq M} \cap \mathcal{S}_m}$ is continuous.
3 In particular, we construct $\mathcal{H}$ as a Hilbert space of functions for which ψ is an interpolating, continuous positive-definite reproducing kernel. Further details in Appendix D.

In the next chapter, we discuss in more detail how these theoretical results inform our implementations of convolutional NPF members. Moreover, we demonstrate (e.g. in Figure 4.2) that incorporating these results into the NPF leads to significant improvements and alleviates the drawbacks highlighted in this chapter.

3.6 SUMMARY AND CONCLUSIONS

We conclude the chapter by summarising its central result. Theorem 3.1 extends Theorem 2.3 by embedding the set into an infinite-dimensional space—a RKHS—instead of a finite-dimensional space. Beyond allowing the model to exhibit translation equivariance, the RKHS formalism allows us to naturally deal with finite sets of varying sizes, which turns out to be challenging with finite-dimensional embeddings. This fact was also used in proof of Theorem 2.3. Furthermore, our formalism requires $\phi(y) = (y^0, y^1, y^2, \ldots, y^K)$ to expand up to order no more than the *multiplicity* of the sets K; if K is bounded, then our results hold for sets up to any arbitrarily large finite size M, while fixing ϕ to be only $(K+1)$-dimensional.

ConvDeepSets have implications for several domains. A prominent example is point-cloud classification (Qi et al., 2017a; Qi et al., 2017b; Wu et al., 2019; Wang et al., 2018). An important work in this area is PointNet (Qi et al., 2017a; Qi et al., 2017b), which introduces DeepSet-style networks for point-cloud classifiers. The design of PointNet captures the important inductive bias of permutation invariance when working with point-clouds, which are naturally represented as varying-sized sets. However, as with image classification, it is natural to consider translation equivariance as an additional inductive bias for classification tasks in this domain. Indeed, Wu et al. (2019) extend the work of Qi et al. (2017a) by incorporating translation equivariance, using architectures very similar to those proposed in this chapter. Wu et al. (2019) go on to demonstrate that they significantly outperform the state-of-the-art PointNet networks (Qi et al., 2017b) at this task.

However, in this thesis we will mainly be concerned with the application of ConvDeepSets to the NPF. In Chapters 4 and 5, we discuss the construction of translation-equivariant members of the NPF, relying directly on the results of this chapter. Yet ConvDeepSets provide an important stepping stone for architectures incorporating more general forms of equivariance. Since the publication of the results described in this chapter, several works have been published directly extending the work described in this chapter to include NPF members exhibiting equivariance to more general symmetries (e.g. Kawano et al., 2021; Holderrieth et al., 2020).

4

CONVOLUTIONAL CONDITIONAL NEURAL PROCESSES

Leveraging the results of Chapter 3, we now turn our attention to the question of constructing translation equivariant members of the NPF. Our main goal in this chapter is to propose a member of the CNPF that incorporates this inductive bias, and demonstrating that such a model (i) achieves significantly improved performance over existing members of the CNPF, and (ii) is able to generalise in time/space, allowing for its deployment in several important settings. These goals are achieved via the introduction of the *Convolutional Conditional Neural Process* (ConvCNP).

4.1 INTRODUCTION

In this chapter, we introduce the *Convolutional Conditional Neural Process* (ConvCNP), a new member of the CNPF that models translation equivariance in the data. Translation equivariance is an important bias for many problems including time-series modelling, spatial data, and images. In contrast to existing members of the CNPF, the ConvCNP employs ConvDeepSets to embed data sets into infinite-dimensional function space, as opposed to a finite-dimensional vector space. We evaluate ConvCNPs in several settings, demonstrating that they achieve state-of-the-art performance compared to existing members of the CNPF.

The results discussed in this chapter are based on the publication "Convolutional Conditional Neural Processes" (Gordon et al., 2020a). The research was carried out in collaboration with my co-first author Wessel Bruinsma, as well as Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. I was closely involved in all aspects of the project, including formulation, development of the software for the ConvCNP,[1] and writing the paper. The main contributions of this chapter are to

- extend the CNPF to include translation equivariance
- evaluate the ConvCNP and demonstrate that it exhibits excellent performance on several synthetic and real-world benchmarks
- demonstrate that building in translation equivariance enables zero-shot generalisation to challenging, out-of-domain tasks.

1 Software implementing the ConvCNP and to reproduce the experiments in this chapter can be found at https://github.com/cambridge-mlg/convcnp

4.2 MODEL AND PARAMETRISATION

We begin by introducing the parametrisation of the ConvCNP. The parametrisation leans heavily on the development of ConvDeepSets, introduced in Chapter 3. ConvCNPs model the conditional distribution as

$$p(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c) = \prod_{n=1}^{N} p(y_n|\Phi_{\boldsymbol{\theta}}(\mathcal{D}_c)(\mathbf{x}_n)) \tag{4.1}$$

$$= \prod_{n=1}^{N} \mathcal{N}(y_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \text{ with } (\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) = \Phi_{\boldsymbol{\theta}}(\mathcal{D}_c)(\mathbf{x}_n), \tag{4.2}$$

where Φ is a ConvDeepSet network (Equation (3.5)). As mentioned in the Chapter 3, the key considerations for the design of $\Phi_{\boldsymbol{\theta}}$ are then the parametrisation of ρ, ϕ, and ψ. Except for the form of ϕ, we distinguish between implementations of ConvCNPs for data that lie on-the-grid (inputs are evenly spaced) and data that lie off-the-grid (inputs "live" in a continuous space).

FORM OF ϕ The applications considered in this chapter have a single (potentially multi-dimensional) output per input location, so the multiplicity of $\mathcal{S}$ is one (i.e., $K = 1$). Recall that our universality statements for Φ relied on setting $\phi(y) = (y^0, y^1, \cdots, y^K)$ to be the powers of y up to order K. It therefore suffices to let ϕ be a power series of order one, which is equivalent to appending a constant to $\mathbf{y}$ in all data sets, i.e. $\phi(\mathbf{y}) = [1\ \mathbf{y}]^\top$. The first output ϕ_1 thus provides the model with information regarding where data has been observed, which is necessary to distinguish between no observed data point at $\mathbf{x}$ and a data point at $\mathbf{x}$ with $\mathbf{y} = \mathbf{0}$. Denoting the functional representation as $\mathbf{h}$, we can think of the first channel $\mathbf{h}^{(0)}$ as a "density channel". We found it helpful to divide the remaining channels $\mathbf{h}^{(1:)}$ by $\mathbf{h}^{(0)}$ (Algorithm 2, line 5), as this improved performance when there is large variation in the density of input locations. In the image processing literature, this is known as *normalized convolution* (Knutsson and Westin, 1993). The normalization operation can be reversed by ρ and is therefore not restrictive.

4.3 CONVCNPS FOR OFF-THE-GRID DATA

Having specified ϕ, it remains to specify the form of ψ and ρ. Our proof of Theorem 3.1 suggests that ψ should be a stationary, non-negative, positive-definite, interpolating kernel. The EQ kernel with a learnable length scale parameter is thus a natural choice. This kernel is multiplied by ϕ to form the functional representation $E(\mathcal{D}_c)$ (Algorithm 2, line 4; Figure 4.1, arrow 1).

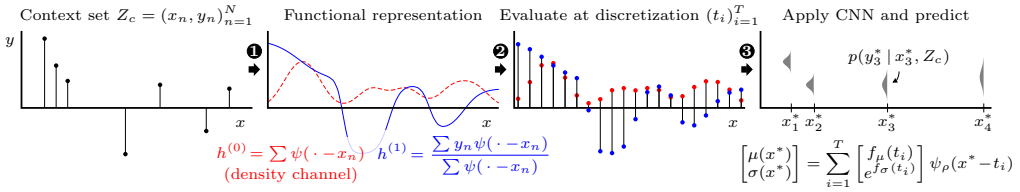


Figure 4.1: Illustration of a forward pass through a ConvCNP. To be read from left to right. The first panel represents the context set, which is input to the model. The context set is then embedded into a functional representation, illustrated in the second panel, for a single "signal" channel and the "density" channel. This representation is then discretised at a fixed grid, as illustrated in the third panel. Finally, this representation is passed through a CNN, and then mapped back to continuous space to produce the predictive distribution at the target locations, as illustrated in the fourth and final panel.

Next, Theorem 3.1 suggests that ρ should be a continuous, translation-equivariant map between function spaces. Kondor and Trivedi (2018) show that any translation-equivariant model has a representation as a CNN. However, CNNs operate on discrete (on-the-grid) input spaces and produce discrete outputs. In order to approximate ρ with a CNN, we discretise the input of ρ, apply the CNN, and finally transform the CNN output back to a continuous function $\mathcal{X} \to \mathcal{Y}$. To do this, for each context and test set, we space points $(\mathbf{t}_i)_{i=1}^n \subset \mathcal{X}$ on a uniform grid (at a pre-specified density) over a hyper-cube that covers both the context and target inputs. We then evaluate $(E(\mathcal{D}_c)(\mathbf{t}_i))_{i=1}^n$ (Algorithm 2, lines 2–3; Figure 4.1, arrow 2). This discretised representation of $E(\mathcal{D}_c)$ is then passed through a CNN (Algorithm 2, line 6; Figure 4.1, arrow 3). We note that a result of the discretisation is that the network is only equivariant to translations larger than $\Delta\mathbf{t} = \mathbf{t}_i - \mathbf{t}_{i-1}$ (which is controlled via the density of the discretisation grid). A further implementation detail relates to a common building block in the design of CNNs—*pooling*—which interacts with translation equivariance (Cohen and Welling, 2016). For example, a single pooling layer with stride s would result in a network that is only equivariant to translations larger than $s\Delta\mathbf{t}$. To avoid this confounding issue, we ommit pooling layers from the design of the CNNs we use in constructing ConvCNPs.

To map the output of the CNN back to a continuous function $\mathcal{X} \to \mathcal{Y}$, we use the CNN outputs as weights for evenly-spaced basis functions (again employing the EQ kernel), which we denote by ψ_ρ (Algorithm 2, lines 7–8; Figure 4.1, arrow 3). The resulting approximation to ρ is not perfectly translation equivariant, but will be approximately so for length scales larger than the spacing of $(E(\mathcal{D}_c)(\mathbf{t}_i))_{i=1}^n$. The resulting continuous functions are then used to generate the (Gaussian) predictive mean and variance at any input. This, in turn, can be used to evaluate the log-likelihood. The complete process is detailed in Algorithm 2

Algorithm 2: Forward pass through ConvCNP for off-the-grid data.

require : $\rho = (\text{CNN}, \psi_\rho)$, ψ, and density γ
require : context $(\mathbf{x}_n, y_n)_{n=1}^N$, target $(\mathbf{x}_m^*)_{m=1}^M$
1 **begin**
2 lower, upper $\leftarrow$ range$\big((\mathbf{x}_n)_{n=1}^N \cup (\mathbf{x}_m^*)_{m=1}^M\big)$
3 $(\mathbf{t}_i)_{i=1}^T \leftarrow$ uniform_grid(lower, upper; γ)
4 $\mathbf{h}_i \leftarrow \sum_{n=1}^N \begin{bmatrix} 1 & y_n \end{bmatrix}^\top \psi(\mathbf{t}_i - \mathbf{x}_n)$
5 $\mathbf{h}_i^{(1)} \leftarrow \mathbf{h}_i^{(1)} / \mathbf{h}_i^{(0)}$
6 $(f_\mu(\mathbf{t}_i), f_\sigma(\mathbf{t}_i))_{i=1}^T \leftarrow \text{CNN}((\mathbf{t}_i, \mathbf{h}_i)_{i=1}^T)$
7 $\boldsymbol{\mu}_m \leftarrow \sum_{i=1}^T f_\mu(\mathbf{t}_i)\psi_\rho(\mathbf{x}_m^* - \mathbf{t}_i)$
8 $\boldsymbol{\sigma}_m \leftarrow \sum_{i=1}^T \text{pos}(f_\sigma(\mathbf{t}_i))\psi_\rho(\mathbf{x}_m^* - \mathbf{t}_i)$
9 **return** $(\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m)_{m=1}^M$
10 **end**

Remark 4.1

The form of ψ as an EQ kernel was used as part of a constructive proof of Theorem 3.1 (see Appendix D for details). However, we note that in some applications, other, less restrictive options may be preferable. For example, it is possible to model ψ as a neural network that accepts as input $\mathbf{x} - \mathbf{x}'$ (see e.g. Wu et al., 2019).

4.4 CONVCNPS FOR ON-THE-GRID DATA

While ConvCNP is readily applicable to many settings where data live on a grid, in this work we focus on the image setting. As such, the following description uses the image completion task as an example, which is often used to benchmark NPs (Garnelo et al., 2018a; Kim et al., 2019). Compared to the off-the-grid case, the implementation becomes simpler as we can choose the discretisation $(\mathbf{t}_i)_{i=1}^n$ to be the pixel locations.

Let $\mathrm{I} \in \mathbb{R}^{H \times W \times C}$ be an image—H, W, C denote the height, width, and number of channels, respectively—and let M_c be the context mask, which is such that $[\mathrm{M}_c]_{i,j} = 1$ if pixel location (i, j) is in the context set, and 0 otherwise. To implement ϕ, we select all context points, $\mathcal{D}_c \coloneqq \mathrm{M}_c \odot \mathrm{I}$, and prepend the context mask: $\phi = [\mathrm{M}_c, \mathcal{D}_c]^\top$ (Algorithm 3, line 4).

Next, we apply a convolution to the context mask to form the density channel: $\mathbf{h}^{(0)} = \text{conv}_{\boldsymbol{\theta}}(\mathrm{M}_c)$ (Algorithm 3, line 4). To all other channels, we apply a normalized convolution: $\mathbf{h}^{(1:C)} = \text{conv}_{\boldsymbol{\theta}}(\mathbf{y})/\mathbf{h}^{(0)}$ (Algorithm 3, line 5), where the division is element-wise. The filter of the convolution is analogous to ψ, which means that $\mathbf{h}$ is the functional representation, with the convolution performing the role of E (the summation in Algorithm 2, line 4). Although the theory suggests using a non-negative, positive-definite kernel, in practice we do not find significant

Algorithm 3: Forward pass through ConvCNP for on-the-grid data.

require : $\rho = \text{CNN}$ and $E = \text{conv}_{\boldsymbol{\theta}}$
require : image I, context M_c, and target mask M_t
1 **begin**
2 // We discretize at the pixel locations.
3 $\mathcal{D}_c \leftarrow \mathrm{M}_c \odot \mathrm{I}$ // Extract context set.
4 $\mathbf{h} \leftarrow \text{conv}_{\boldsymbol{\theta}}([\mathrm{M}_c, \mathcal{D}_c]^\top)$
5 $\mathbf{h}^{(1:C)} \leftarrow \mathbf{h}^{(1:C)} / \mathbf{h}^{(0)}$
6 $f_t \leftarrow \mathrm{M}_t \odot \text{CNN}(\mathbf{h})$
7 $\boldsymbol{\mu} \leftarrow f_t^{(1:C)}$
8 $\boldsymbol{\sigma} \leftarrow \text{pos}(f_t^{(C+1:2C)})$
9 **return** $(\boldsymbol{\mu}, \boldsymbol{\sigma})$
10 **end**

empirical differences between an EQ kernel and using a fully trainable kernel restricted to positive values to enforce non-negativity.

Lastly, we describe the on-the-grid version of $\rho(\cdot)$, which consists of two stages. First, we apply a CNN to $E(\mathcal{D}_c)$ (Algorithm 3, line 6). Second, we apply a shared, pointwise MLP that maps the output of the CNN at each pixel location in the target set to $\mathbb{R}^{2C}$, where we absorb MLP into the CNN (MLP can be viewed as an 1×1 convolution). The first C outputs are the means of a Gaussian predictive distribution and the second C the standard deviations, which then pass through a positivity-enforcing function (Algorithm 3, line 7–8). To summarise, the on-the-grid algorithm is given by

$$(\boldsymbol{\mu}, \text{pos}^{-1}(\boldsymbol{\sigma})) = \underbrace{\text{CNN}}_{\rho}(\overbrace{[\underbrace{\text{conv}(\mathrm{M}_c)}_{\text{density channel}}; \text{conv}(\mathrm{M}_c \odot \mathrm{I}) / \underbrace{\text{conv}(\mathrm{M}_c)}_{\text{multiplies by } \psi \text{ and sums}}]^\top}^{E(\text{context set})}), \tag{4.3}$$

where $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ are the image mean and standard deviation, ρ is implemented with a CNN, and E is implemented with the mask M_c and convolution conv. Pseudo-code for the on-the-grid ConvCNP is given in Algorithm 3.

4.5 EMPIRICAL EVALUATION

We evaluate the performance of ConvCNPs in both on-the-grid and off-the-grid settings focusing on two central questions: (i) Do translation-equivariant models improve performance in appropriate domains? (ii) Can translation equivariance enable ConvCNPs to generalize to settings outside of those encountered during training? We use several off-the-grid data-sets which are irregularly sampled time series ($\mathcal{X} = \mathbb{R}$), comparing to Gaussian processes (GPs; Williams and Rasmussen (2006)) and AttnCNP, the best performing member of the CNPF. We

Table 4.1: Quantitative comparison of CNPF members on held-out tasks from the GP and sawtooth experiments. We can see that both instantiations of the ConvCNP significantly and consistently outperform both the CNP and the AttnCNP on all tasks. Moreover, the second column illustrates that the ConvCNP models are able to achieve this with far fewer parameters than their counterparts. For reference, we also provide (where applicable) two likelihoods achieved by the ground truth GP models, provided beneath the dashed line. GP (full) is the likelihood evaluation for the ground truth GP with a full covariance matrix, and GP (diag) is the likelihood for this GP, but considering independent normal predictions. The latter is the Bayes' optimal value for any model with factorised predictive distributions.

Model	Params	EQ	Weakly-Periodic	Matern	Sawtooth
CNP	66818	-0.86 ± 3e-3	-1.23 ± 2e-3	-0.95 ± 1e-3	-0.16 ± 1e-5
AttnCNP	149250	0.72 ± 4e-3	-1.20 ± 2e-3	0.10 ± 2e-3	-0.16 ± 2e-3
ConvCNP	6537	0.70 ± 5e-3	-0.92 ± 2e-3	0.32 ± 4e-3	1.43 ± 4e-3
ConvCNPXL	50617	**1.06 ± 4e-3**	**-0.65 ± 2e-3**	**0.53 ± 4e-3**	**1.94 ± 1e-3**
GP (diag)	– –	2.87 ± 1e-5	-0.15 ± 7e-5	0.72 ± 1e-5	– –
GP (full)	– –	5.69 ± 1e-6	0.31 ± 8e-5	1.76 ± 1e-5	– –

then evaluate on several on-the-grid image data sets ($\mathcal{X} = \mathbb{Z}^2$). In all settings we observe substantial improvements over existing neural process models. For the CNN component of our model, we propose a small and large architecture for each experiment (in the experimental sections named ConvCNP and ConvCNPXL, respectively). We note that these architectures are different for off-the-grid and on-the-grid experiments, with full details regarding the architectures provided in Appendices E.2 and F.1.

4.5.1 *Synthetic 1D Experiments*

We first consider the synthetic regression problems introduced in the previous chapter. The ConvCNP is compared to the CNP (Garnelo et al., 2018a) and AttnCNP (Kim et al., 2019). In this chapter, we provide both a qualitative analysis of the ConvCNP predictive distributions, and a more rigorous quantitative analysis of model performance based on held-out log-likelihoods. Training and testing procedures are fixed across all models. Full details on models, data generation, and training procedures are provided in Appendix E.

Table 4.1 reports the log-likelihood means and standard errors of the models over 1000 tasks. The context and target points for both training and testing lie within the interval $[-2, 2]$ where training data was observed. Table 4.1 also provides an evaluation of the ground truth GP kernel with a full covariance matrix (GP (full)) and a diagonalised covariance matrix (GP (diag)). The latter provides an upper-bound on the performance of any model with factorised predictive

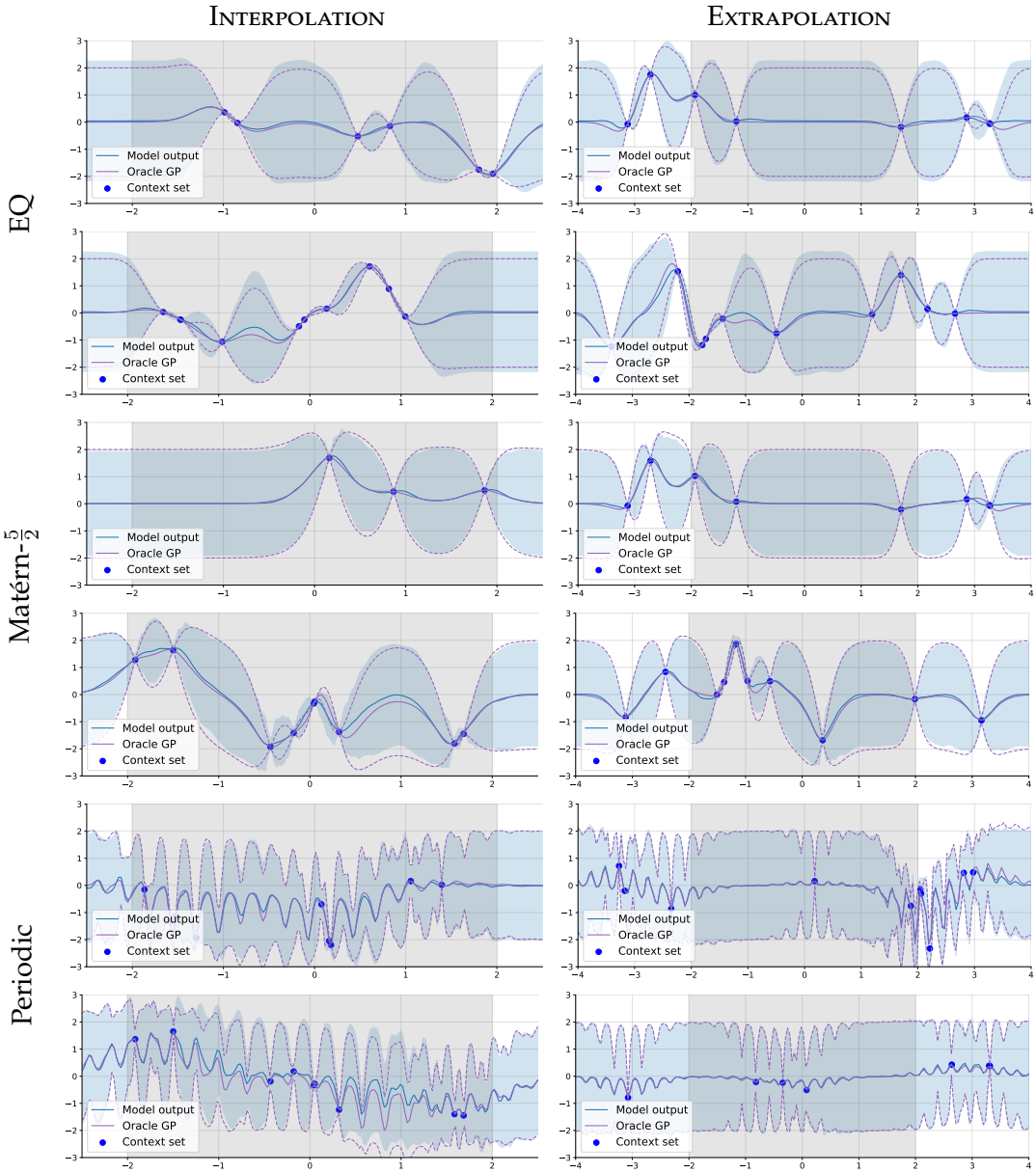


Figure 4.2: Evaluating the predictive distributions of the ConvCNP in interpolation (left) and extrapolation (right) settings. The models depicted here are the ConvCNPXL. Models were trained on tasks generated from GPs with (top) EQ, (center) Matérn-$\frac{5}{2}$, and (bottom) weakly-periodic kernels. For each kernel, two examples of interpolation and extrapolation tasks are provided. From the lefthand column we see that the ConvCNP is able to very closely track the predictive functions of the ground truth GP. Moreover, from the righthand column we see that the ConvCNP is able to seamlessly generalise to settings where data is observed outside the training range. This is an immediate consequence of translation equivariance.

distributions over the target set, and can be thought of as the Bayes' optimal predictive distribution.

Quantitative Analysis

First, we note that the ConvCNP requires far fewer parameters than the CNP or AttnCNP (Table 4.2, second column). Even the "large" version of the ConvCNP has fewer parameters than the CNP. This is one of the important benefits of modelling equivariance, namely, providing a principled manner to share parameters (Kondor and Trivedi, 2018; Ravanbakhsh et al., 2017).

More importantly, Table 4.1 demonstrates that, even when extrapolation is not required, ConvCNP consistently and significantly outperforms other NPF models, despite having fewer parameters. In all cases, ConvCNPXL significantly outperforms the AttnCNP. Moreover, except for the EQ kernel experiment, ConvCNP outperforms the CNP and AttnCNP with an order of magnitude fewer parameters. Finally, we note that despite the significantly improved performance of the ConvCNP, there remains a significant difference in performance with the diagonalised GP, implying that there is room for further improvement.

Qualitative Analysis

Figure 4.2 visualises the predictive distributions of the ConvCNP for each of the GP kernels, in both the interpolation and extrapolation settings. For each setting, two randomly generated context sets are plotted. We see that the predictive distributions produced by the ConvCNP are quite impressive: there is a tight correspondence between the mean and variance functions of the ConvCNP and the ground truth GP. Moreover, the ConvCNP produces smooth functions that appear to have similar inductive biases to those of the underlying GP. For example, the ConvCNP appears able to model the periodic structure in the weakly-periodic kernel GP. Finally, the right-hand column of Figure 4.2 demonstrates that the ConvCNP seamlessly generalises to settings where data is observed outside the training range. Observing data outside the range does not negatively affect the predictive distribution, inside or outside the training range.

Remark 4.2

We note that the ConvCNP cannot exactly recover the underlying process. For example, for the weakly-preiodic kernel, it can only model local *periodicity because it has a bounded receptive field — the size of the input region that can affect a particular output. In fact, this is true for any of the GPs above, all of which have "infinite receptive fields". In principle, an observation at one point affects the predictions along the entire x-axis. This means that no model with a bounded receptive field can exactly recover the GP predictive distribution everywhere. In practice however, most GPs with non-periodic kernels (e.g., with EQ and Matérn-$\frac{5}{2}$ kernels) have a finite length-scale, and points much further apart than the length-scale are, for all practical purposes, independent.*

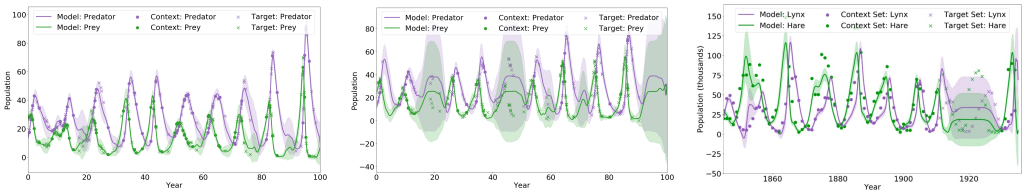


Figure 4.3: Left and centre: two samples from the Lotka–Volterra process (simulated). Right: ConvCNP trained on simulations and applied to the Hudson's Bay lynx-hare dataset (real). Each plot portrays the population levels of the lynx and hare against time. The plots further show the means and two standard deviations of the predictive models.

4.5.2 *Predator-Prey Models: Sim2Real*

The ConvCNP is well suited for applications where simulation data is plentiful, but real world training data is scarce (Sim2Real). The ConvCNP can be trained on a large amount of simulation data, and then be deployed with real-world training data as the context set. To experiment with this setting, we consider the Lotka–Volterra model, which is used to describe the evolution of predator–prey populations (Wilkinson, 2011). This model has been used in the Approximate Bayesian Computation literature where the task is to infer the parameters from samples drawn from the Lotka–Volterra process (Papamakarios and Murray, 2016; Tran et al., 2017). These methods do not straightforwarddly extend to prediction problems such as interpolation or forecasting. In contrast, we train the ConvCNP on synthetic data sampled from the Lotka–Volterra model and can then condition on real-world data from the Hudson's Bay lynx–hare data set (Leigh, 1968) to perform interpolation (see Figure 4.3; full experimental details provided in Appendix E.4).

As demonstrated in Figure 4.3, the ConvCNP is able to provide accurate interpolation predictions for both the simulated (left and center panels) and real-world dataset (right panel). We note that the parameters used to simulate the training data is unlikely to be similar to the parameters that would be learned from the observations in the lynx-hare data set, as these were tuned to produce reasonable time-series from the underlying process (see Appendix E.4 for further details). This demonstrates the ability of the ConvCNP to generalise from simulated data to real-world datasets.

We also attempted to train an AttnCNP for comparison. Due to the nature of the synthetic data generation, many of the training series end before 90 time units, the length of the Hudson's Bay lynx-hare series. Effectively, this means that the AttnCNP was asked to predict outside of its training interval, a task that it struggles with, as demonstrated and discussed in Chapter 3. The plots in Figure 4.4 show that the AttnCNP is able to learn the first part of the time series but is unable to model data outside of the first 20 or so time units.

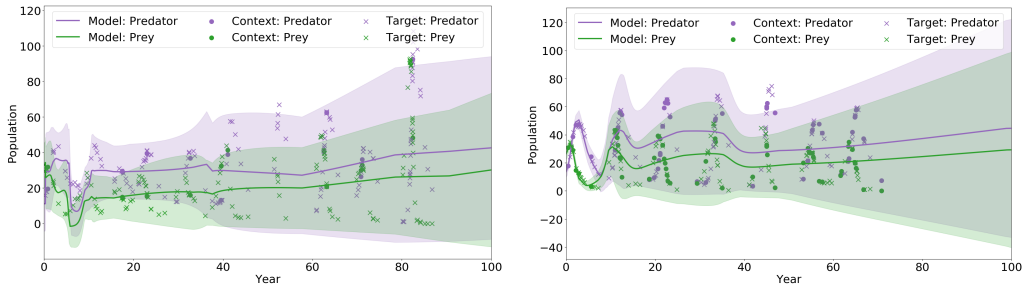


Figure 4.4: AttnCNP performance on two samples from the Lotka–Volterra process. Both correspond to simulated tasks.

4.6 2D IMAGE COMPLETION EXPERIMENTS

To test ConvCNP beyond one-dimensional inputs, we evaluate our model on on-the-grid image completion tasks and compare it to the AttnCNP. Image completion can be cast as a prediction of pixel intensities $\mathbf{y}_t$ ($\in \mathbb{R}^3$ for RGB, $\in \mathbb{R}$ for greyscale) given a target 2D pixel location $\mathbf{x}_t$ conditioned on an observed (context) set of pixel values $\mathcal{D}_c = (\mathbf{x}_n, \mathbf{y}_n)_{n=1}^N$. In the following experiments, the context set can vary but the target set contains all pixels from the image. Further experimental details are provided in Appendix F.

Table 4.2: Mean and two standard errors of log-likelihood from image experiments. For each model, each experiment is run 6 times with different random seeds. All models achieving error bounds overlapping with best performing model are bolded. CelebA64 is not run with AttnCNP due to memory constraints.

Model	Params	MNIST	SVHN	CelebA32	CelebA64	ZSMM
AttnCNP	410k	1.08 ±0.04	3.94 ±0.02	3.18 ±0.02	– –	-0.83 ±0.08
ConvCNP	113k	1.21 ±0.00	3.89 ±0.01	3.22 ±0.02	3.66 ±0.01	**1.18 ±0.04**
ConvCNPXL	400k	**1.27 ±0.01**	**3.97 ±0.02**	**3.39 ±0.02**	**3.73 ±0.01**	0.86 ±0.12

4.6.1 *Standard Image Benchmark Tasks*

We first evaluate the model on four common benchmarks: MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), and 32×32 and 64×64 CelebA (Liu et al., 2018). Importantly, these data sets consist of images containing a single, centred object. As a result, perfect translation-equivariance might hinder the performance of the model when the test data are similarly structured. We therefore also evaluated a larger ConvCNP that can learn such non-stationarity, while still sharing parameters across the input space (ConvCNPXL).

Table 4.2 demonstrates that the ConvCNP significantly outperforms AttnCNP when it has a large receptive field size, while being at least as good with a smaller receptive field size. Qualitative samples for various context sets can be seen in

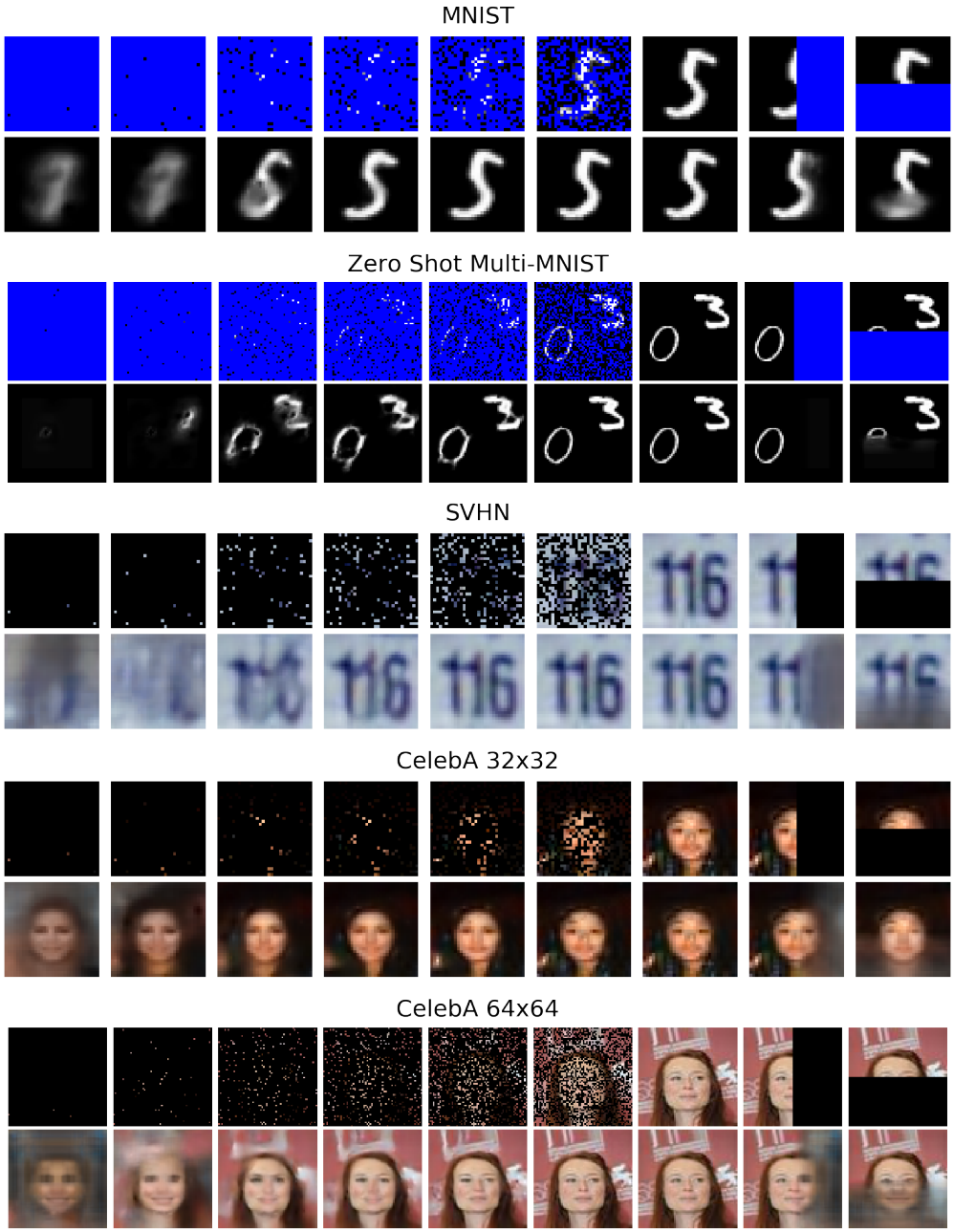


Figure 4.5: Qualitative evaluation of the ConvCNP(XL). For each dataset, an image is randomly sampled, the first row shows the given context points while the second is the mean of the estimated conditional distribution. From left to right the first seven columns correspond to a context set with 3, 1%, 5%, 10%, 20%, 30%, 50%, 100% randomly sampled context points. In the last two columns, the context sets respectively contain all the pixels in the left and top half of the image. ConvCNPXL is shown for all datasets besides ZSMM, for which we show the fully translation equivariant ConvCNP.

Figure 4.5. Further qualitative comparisons and ablation studies can be found in Appendices F.3 and F.4.

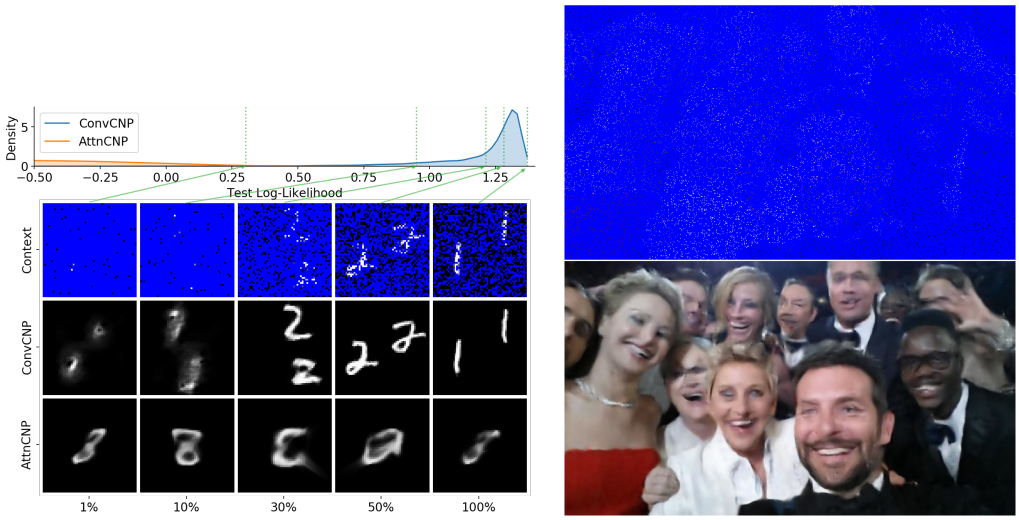


Figure 4.6: (Left) Log-likelihood and qualitative results on ZSMM. The top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), ConvCNP target predictions (middle), and AttnCNP target predictions (bottom). Each column corresponds to a given percentile of the ConvCNP distribution. (Right) Qualitative evaluation of a ConvCNPXL trained on the unscaled CelebA (218×178) and tested on Ellen's Oscar unscaled (337×599) selfie with 5% of the pixels as context (top).

4.6.2 *Generalization to Multiple, Non-Centred Objects*

The data sets above consist of single, centred objects per image. Here we evaluate whether ConvCNPs trained on such data can leverage translation equivariance to generalise to images containing multiple, non-centred objects.

The last column of Table 4.2 evaluates the models in a zero shot multi-MNIST (ZSMM) setting, where images contain multiple digits at test time (see Appendix F.2 for further details). The ConvCNP significantly outperforms the AttnCNP in these settings. Figure 4.6 (left) provides a histogram of the image log-likelihoods for the ConvCNP and AttnCNP, as well as qualitative results at different percentiles of the ConvCNP distribution. Moreover, the ConvCNP is able to extrapolate to this out-of-distribution test set, while AttnCNP appears to model the bias of the training data and predict a centred "mean" digit independently of the context. Interestingly, ConvCNPXL does not perform as well on this task. In particular, we find that, as the receptive field becomes very large, performance on this task decreases. We hypothesize that this has to do with behaviour of the model at the edges of the image. CNNs with larger receptive fields—the region of input pixels that affect a particular output pixel—are able to model non-stationary behaviour by looking at the distance from any pixel to the image boundary. We expand on this discussion and provide further experimental evidence regarding the effects of receptive field on the ZSMM task in Appendix F.6.

Although ZSMM is a contrived task, we note that our field of vision often contains multiple independent objects, thereby requiring generalisation of this form. As a more realistic example, we tested a ConvCNP model trained on CelebA on a natural image of different shape which contains multiple people (Figure 4.6 (right)). Even with 95% of the pixels removed, the ConvCNP is able to produce a qualitatively reasonable reconstruction. A comparison with AttnCNP is given in Appendix F.3.

4.6.3 *Computational Efficiency*

Beyond the performance and generalization improvements, a key advantage of the ConvCNP is its computational efficiency. The memory and time complexity of a single self-attention layer grows quadratically with the number of inputs M (the number of pixels for images) but only linearly for a convolutional layer.

Empirically, we note that with a batch size of 16 on 32×32 MNIST, ConvCNPXL requires 945MB of VRAM, while AttnCNP requires 5839 MB. For the 56×56 ZSMM ConvCNPXL increases its requirements to 1443 MB, while AttnCNP could not fit onto a 32GB GPU. Ultimately, AttnCNP had to be trained with a batch size of 6 (using 19139 MB) and we were not able to fit it for CelebA64. Recently, restricted attention has been proposed to overcome this computational issue (Parmar et al., 2018), but we leave an investigation of this and its relationship to ConvCNPs to future work.

4.7 SUMMARY AND CONCLUSIONS

We have introduced the ConvCNP, a new member of the CNPF endowed with translation equivariance. We demonstrated that this powerful inductive bias enabled the ConvCNP to achieve significant performance improvements over existing members of the CNPF, as well as generalise in interesting and important ways.

ConvCNPs enable the deployment of deep learning tools to applications that require models to (i) handle continuous time/space data, and (ii) generalise in time and space. Example applications include environmental modelling (which is often cast as a spatial modelling problem) and clinical healthcare data (which can be viewed as continuous time-series modelling). In the future, we hope to see applications of ConvCNPs in such settings, where to date deep learning models have struggled to achieve good performance.

An important limitation of the ConvCNP is the inability to capture dependencies in the predictive distribution, and as a result, the inability to produce coherent samples from this distribution. The NPF suggests a path towards alleviating these

issues: extending the ConvCNP to allow for latent variable parametrisations. Due to the nature of the latent representation of the ConvCNP, this is not necessarily a straightforward endeavour. In the next chapter, we consider such a model, which we coin the ConvNP.

5

CONVOLUTIONAL NEURAL PROCESSES

We have seen that incorporating translation equivariance in the CNPF produces models with desirable properties. However, the ConvCNP suffers from several drawbacks associated with membership in the CNPF. In this chapter, we consider the development of translation equivariant members of the LNPF, which overcome several limitations of the ConvCNP, though at the cost of requiring approximate training procedures. This is achieved via the introduction of the *Convolutional Neural Process* (ConvNP).

5.1 INTRODUCTION

In this chapter we introduce the Convolutional Neural Process (ConvNP), which endows Neural Processes (Garnelo et al., 2018b) with translation equivariance. Conversely, we can think of the ConvNP as extending the ConvCNP to allow for dependencies in the predictive distribution. This enables ConvNPs to be deployed in settings which may benefit from coherent samples, such as Bayesian optimisation or conditional image completion. We demonstrate the strong performance and generalisation capabilities of ConvNPs on 1d regression tasks, and show that they are able to produce far more compelling samples than other members of the LNPF. We further demonstrate the strong performance of ConvNPs on image completion, and various tasks with real-world spatio-temporal data.

The work in this chapter is based on the publication "Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes" (Foong et al., 2020). The research was conducted in collaboration with my co-first authors Andrew Y. K. Foong and Wessel Bruinsma, as well as Yann Dubois, James Requeima, and Richard E. Turner. I was closely involved in all aspects of the work, including conceptualisation, development of the theoretical results, development of the software,[1] experimental evaluation, and writing of the paper. The main contributions of this chapter are to

- introduce ConvNPs, extending ConvCNPs to model rich join predictive distributions
- evaluate our proposed training procedure (Section 2.7.3), and demonstrate that it improves performance for members of the LNPF

1 Software to implement the ConvNP and reproduce the experiments in this chapter can be found at https://github.com/wesselb/NeuralProcesses.jl

- demonstrate the usefulness of ConvNPs on synthetic data, image-based sampling and extrapolation, and real-world environmental data sets.

5.2 LIMITATIONS OF THE CONVCNP

Before introducing the parametrisation of the ConvNP, we reiterate the main limitations of the ConvCNP, which are inherited more generally from the CNPF. Recall that the ConvCNP (Equation (4.1)) defines a map from context sets $\mathcal{D}_c$ to predictive SPs. Specifically, let $\mathcal{P}_{\mathcal{N}}(\mathcal{X}) \subset \mathcal{P}(\mathcal{X})$ denote the set of *noise GPs*: GPs on $\mathcal{X}$ whose covariance is given by $\mathrm{Cov}(\mathbf{x}, \mathbf{x}') = \sigma^2(\mathbf{x})\delta[\mathbf{x} - \mathbf{x}']$, where $\sigma^2 \in \mathcal{C}_b(\mathcal{X})$ and $\delta[0] = 1$ with $\delta[\,\cdot\,] = 0$ otherwise. Then the ConvCNP is a map of the form $\mathcal{S} \to\!\!\!\!\triangleright\, \mathcal{P}_{\mathcal{N}}(\mathcal{X})$, with Equation (4.1) defining its finite-dimensional distributions.

Unfortunately, processes in $\mathcal{P}_{\mathcal{N}}(\mathcal{X})$ possess two key limitations. First, it is impossible to obtain coherent function samples as each point of the function is generated independently. Second, Gaussian distributions cannot model multi-modality, heavy-tailedness, or asymmetry.

5.3 PARAMETRISING THE CONVNP

Similarly to other members of the LNPF, the ConvNP extends the ConvCNP by parametrising a map to a richer space of predictive SPs, allowing for coherent sampling and non-Gaussian predictives. This is achieved by passing the output of a ConvCNP through a non-linear, translation equivariant map between function spaces. Specifically, the ConvNP uses an encoder–decoder architecture, where the encoder $\mathrm{E}\colon \mathcal{S} \to\!\!\!\!\triangleright\, \mathcal{P}_{\mathcal{N}}(\mathcal{X})$ is a ConvCNP and the decoder $d\colon \mathbb{R}^{\mathcal{X}} \to\!\!\!\!\triangleright\, \mathbb{R}^{\mathcal{X}}$ is translation equivariant. Conditioned on $\mathcal{D}_c$, ConvNP samples can be obtained by sampling a function $\mathbf{z} \sim \mathrm{ConvCNP}(\mathcal{D}_c)$ and then computing $f = d(\mathbf{z})$. An illustration of this procedure is provided in Figure 5.1.

Importantly, d takes functions to functions and does not necessarily act pointwise: letting $f(\mathbf{x})$ depend on the value of $\mathbf{z}$ at multiple locations is crucial for inducing dependencies in the predictive distribution. This sampling procedure induces a map between SPs, $\mathrm{D}\colon \mathcal{P}_{\mathcal{N}}(\mathcal{X}) \to\!\!\!\!\triangleright\, \mathcal{P}(\mathcal{X})$. Putting these together, with explicit parameter dependence in E and D, the ConvNP is constructed as

$$\mathrm{ConvNP}_{\boldsymbol{\theta}} = \mathrm{D}_{\boldsymbol{\theta}} \circ \mathrm{E}_{\boldsymbol{\phi}}, \quad \mathrm{E}_{\boldsymbol{\phi}} = \mathrm{ConvCNP}_{\boldsymbol{\phi}}, \quad \mathrm{D}_{\boldsymbol{\theta}} = (d_{\boldsymbol{\theta}})_*, \tag{5.1}$$

where $(d_{\boldsymbol{\theta}})_*$ is defined by applying the map $d_{\boldsymbol{\theta}}$ to each sample of E_ϕ. Note that we are explicitly distinguishing between the parameters of the encoder (denoted $\boldsymbol{\phi}$) and those of the decoder (denoted $\boldsymbol{\theta}$).

In practice, we cannot compute samples of noise GPs ($\mathcal{P}_{\mathcal{N}}$) because they comprise uncountably many independent random variables. This further implies that continuous model is not well-defined when the latent variable is defined via noise GPs (though would be if we instead modelled the latent variable with full-covariance GPs). Instead, we consider a discrete version of the model, which enables computation and results in a well-defined model. As in Chapter 4, we discretise the domain of $\mathbf{z}$ on a grid $(\mathbf{x}_i)_{i=1}^K$, with $\mathbf{z} := (z(\mathbf{x}_i))_{i=1}^K$. As a consequence, the model can only be equivariant up to shifts on this discrete grid. With this discretisation, sampling $\mathbf{z} \sim \text{ConvCNP}_{\boldsymbol{\phi}}(\mathcal{D}_c)$ amounts to sampling independent Gaussian random variables, and $d_{\boldsymbol{\theta}}$ is implemented by passing $\mathbf{z}$ through a CNN. Importantly, the discretized version of the model does not have the continuous model as its limit as the discretesation grid becomes finer.

Following Kim et al. (2019), we define the model likelihood by adding heteroskedastic Gaussian observation noise $\sigma_y^2(\mathbf{x}, \mathbf{z})$ to the predictive function draws $f = d_{\boldsymbol{\theta}}(\mathbf{z}) \in \mathbb{R}^{\mathcal{X}}$:

$$p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c) = \mathop{\mathbb{E}}_{\mathbf{z}\sim \mathrm{E}_{\boldsymbol{\phi}}(\mathcal{D}_c)} \left[\prod_{(\mathbf{x},y)\in\mathcal{D}_t} \mathcal{N}\left(y; d_{\boldsymbol{\theta}}(\mathbf{z})(\mathbf{x}), \sigma_y^2(\mathbf{x}, \mathbf{z})\right) \right]. \tag{5.2}$$

Although the product in the expectation factorizes, $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{X}_t, D_c)$ does not: $\mathbf{z}$ induces dependencies in the predictive, in contrast to Equation (4.1).

5.4 TRANSLATION EQUIVARIANCE OF THE CONVNP

It may not be immediately obvious from the above presentation that the ConvNP indeed satisfies Property 3.2. In this section, we provide a simple proof that the discretised version of ConvNPs are indeed translation equivariant, as desired. We prove this by first proving that both the encoder and decoder are (separately) translation equivariant. Thus, as the ConvNP is a composition of the decoder and encoder (Equation (5.1)), it immediately follows that it satisfies the desired property.

Proposition 5.1

Let d be a measurable, translation equivariant map from $(\mathbb{R}^{\mathcal{Z}}, \Sigma)$ to $(\mathbb{R}^{\mathcal{X}}, \Sigma)$, where $\mathcal{Z}$ is the discretised space on which the latent variables is defined. The ConvNP decoder $\mathrm{D} : \mathcal{P}(\mathcal{Z}) \to \mathcal{P}(\mathcal{X})$, defined by $\mathrm{D}(P) = d_*(P)$, where $d_*(P)$ is the pushforward measure under d, is translation equivariant.

Proof.
Let $F \in \Sigma$ be measurable. Then,

$$\begin{aligned}
\mathrm{D}(T_{\boldsymbol{\tau}} P)(F) &\overset{\text{(a)}}{=} T_{\boldsymbol{\tau}} P(d^{-1}(F)) \\
&= P(T_{-\boldsymbol{\tau}} d^{-1}(F)) \\
&\overset{\text{(b)}}{=} P(d^{-1}(T_{-\boldsymbol{\tau}} F)) \\
&= \mathrm{D}(P)(T_{-\boldsymbol{\tau}} F) \\
&= T_{\boldsymbol{\tau}} \mathrm{D}(P)(F).
\end{aligned}$$

Here (a) follows from definition of the pushforward, and (b) follows because

$$\begin{aligned}
T_{-\boldsymbol{\tau}} d^{-1}(F) &= T_{-\boldsymbol{\tau}} \{f : d(f) \in F\} \\
&= \{T_{-\boldsymbol{\tau}} f : d(f) \in F\} \\
&= \{f : d(T_{\boldsymbol{\tau}} f) \in F\} \\
&= \{f : T_{\boldsymbol{\tau}} d(f) \in F\} \\
&= \{f : d(f) \in T_{-\boldsymbol{\tau}} F\} \\
&= d^{-1}(T_{-\boldsymbol{\tau}} F).
\end{aligned}$$

□

Proposition 5.2
The ConvNP *encoder* E *(a* ConvCNP*), is a translation equivariant map from data sets to stochastic processes.*

Proof.
Recall that the mean and variance $\boldsymbol{\mu}(\cdot, S), \boldsymbol{\sigma}^2(\cdot, S)$ (viewed as maps from $\mathcal{S} \to \mathcal{C}_b(\mathcal{X})$) of the encoder E are both given by ConvDeepSets. Due to the translation equivariance of ConvDeepSets Theorem 3.1, $\boldsymbol{\mu}(\cdot, T_{\boldsymbol{\tau}} S) = T_{\boldsymbol{\tau}} \boldsymbol{\mu}(\cdot, S)$ for all $S, \boldsymbol{\tau}$, and similarly for σ^2. Let $F \in \Sigma$. Then since the measure $\mathrm{E}(S) \in \mathcal{P}_{\mathcal{N}}(\mathcal{X})$ is defined entirely by its mean and variance function, $\mathrm{E}(T_{\boldsymbol{\tau}} S)(F) = \mathrm{E}(S)(T_{-\boldsymbol{\tau}} F) = T_{\boldsymbol{\tau}} \mathrm{E}(S)(F)$. □

Finally, noting that a composition of translation equivariant maps is itself translation equivariant (Munkres, 1974; Kondor and Trivedi, 2018), we obtain the following proposition:

Proposition 5.3
Define $\mathrm{ConvNP} = \mathrm{D} \circ \mathrm{E}$*. Then* ConvNP *is a translation equivariant map from data sets to stochastic processes.*

5.5 CONVNPS IN PRACTICE

The ConvNP can be implemented very simply by passing samples from a ConvCNP through an additional CNN decoder, which we denote $d_{\boldsymbol{\theta}}$. In this

Algorithm 4: Forward pass through ConvNP for off-the-grid data.

require : $d_{\boldsymbol{\theta}} = (\text{CNN}, \psi_d)$, $\text{E}_{\boldsymbol{\phi}}$ (off-the-grid ConvCNP), number of samples L
require : context $(\mathbf{x}_n, y_n)_{n=1}^N$, target $(\mathbf{x}_m^*)_{m=1}^M$
1 **begin**
2 $\quad \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}} \leftarrow \text{E}_{\boldsymbol{\phi}}(\mathcal{D}_c)$
3 $\quad$ **for** $l = 1, \ldots, L$ **do**
4 $\quad\quad \mathbf{z}_l \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}\right)$
5 $\quad\quad (f_{\boldsymbol{\mu}}(\mathbf{t}_i), f_{\boldsymbol{\sigma}}(\mathbf{t}_i))_{i=1}^K \leftarrow \text{CNN}(\mathbf{z}_l)$
6 $\quad\quad \boldsymbol{\mu}_{m,l} \leftarrow \sum_{i=1}^T f_{\boldsymbol{\mu}}(\mathbf{t}_i)\psi_d(\mathbf{x}_m^* - \mathbf{t}_i)$
7 $\quad\quad \boldsymbol{\sigma}_{m,l} \leftarrow \text{pos}\left(f_\sigma(\mathbf{t}_i)\right)$
8 $\quad$ **end**
9 $\quad$ **return** $(\boldsymbol{\mu}, \boldsymbol{\sigma})$
10 **end**

Algorithm 5: Forward pass through ConvNP for on-the-grid data.

require : $d_{\boldsymbol{\theta}} = \text{CNN}$, $\text{E}_{\boldsymbol{\phi}}$ (on-the-grid ConvCNP), number of samples L
require : image I, context mask M_c, and target mask M_t
1 **begin**
2 $\quad \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z \leftarrow \text{E}_{\boldsymbol{\theta}}(I, M_c)$
3 $\quad$ **for** $l = 1, \ldots, L$ **do**
4 $\quad\quad \mathbf{z}_l \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$
5 $\quad\quad (f_{\boldsymbol{\mu}}(\mathbf{t}_i), f_{\boldsymbol{\sigma}}(\mathbf{t}_i))_{i=1}^K \leftarrow \text{CNN}(\mathbf{z}_l)$
6 $\quad\quad \boldsymbol{\mu} \leftarrow f_t^{(1:C)}$
7 $\quad\quad \boldsymbol{\sigma} \leftarrow \text{pos}\left(f_t^{(C+1:2C)}\right)$
8 $\quad$ **end**
9 $\quad$ **return** $(\boldsymbol{\mu}, \boldsymbol{\sigma})$
10 **end**

section, we provide further implementation details for the ConvNP, including pseudo-code and illustrations. As in Chapter 4, we distinguish between the "on-the-grid" and "off-the-grid" versions of the model. For an "off-the-grid" ConvNP, similarly to the ConvCNP, we must map the output of a standard CNN back to functions on a continuous domain $\mathcal{X}$. This can be achieved via an RBF mapping, similar to the off-the-grid ConvCNP, e.g. Algorithm 4 lines 6, 7. Pseudo-code for off- and on-the-grid ConvNPs are provided in Algorithms 4 and 5, respectively. Note that for the ConvNP, the discretisation of the latent function $\mathbf{z}$ is typically on a pre-specified grid, and therefore lines 7 and 8 of Algorithm 2 are unnecessary when calling the ConvCNP (Algorithm 4, line 1).

Finally, Figure 5.1 provides an illustration of a forward pass through the ConvNP. The diagram was created using a context set drawn from an EQ kernel, and passed through a trained ConvNP.

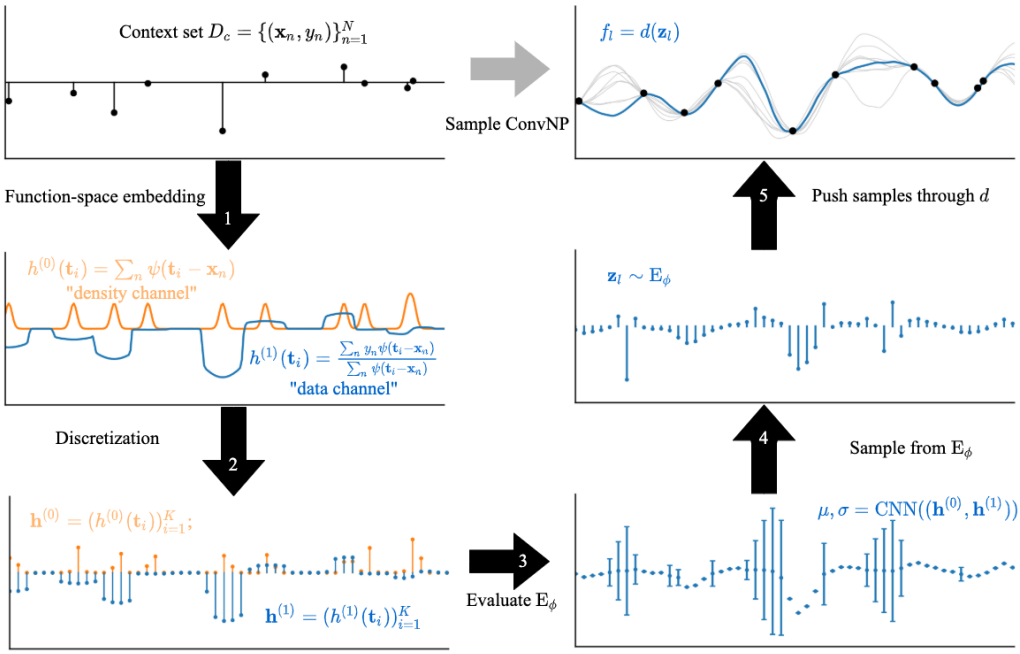


Figure 5.1: Illustration of a forward pass through a ConvNP for off-the-grid data. To be read counter-clockwise, starting at top-left panel. The first panel represents the context set, which is input to the model. The context set is then embedded into a functional representation, illustrated in the second panel, for a single "signal" channel and the "density" channel. This representation is then discretised at a fixed grid, as illustrated in the third panel. Next, the discretised representation is passed through a CNN, which produces the mean and variances of the latent function evaluated at the discretisation locations, illustrated in the fourth panel. A sample from the latent function distribution is sampled at each evaluated point (fifth panel), which is then passed through an additional CNN to produce a sample from the predictive distribution (sixth panel).

5.6 TRAINING CONVNPS

In Chapter 2 we introduced two training procedures for members of the LNPF. The first is inspired by performing approximate VI in a generative model associated with the LNPF, and requires maximising $\mathcal{L}_{\text{NPVI}}$ (Equation (2.33)). The second is inspired by the results in Section 2.7.1, and forgoes approximate inference for the latent variable in favour of a simpler, approximate maximum-likelihood procedure involving maximising $\hat{\mathcal{L}}_{\text{ML}}$ (Equation (2.41)). We briefly discuss implications of these two procedures that are specific to the parametrisation of the ConvNP.

When considering the generative process associated with the LNPF (Equations (2.27) to (2.29)) with ConvNPs, $\mathbf{z}$ is a latent *function*, q_ϕ is a map from data sets to SPs, and d_θ is a map between function spaces. A natural choice is then to use a ConvCNP and CNN for q_ϕ and d_θ, respectively.[2] This results in the same parametrisation as in Section 5.3, but a subtly different modelling interpretation.

2 Recall that $\mathcal{L}_{\text{NPVI}}$ requires the introduction of an *inference network* q_ϕ.

For the non-discretised ConvNP, $\mathcal{L}_{\text{NPVI}}$ involves KL divergences between SPs which cannot be computed directly and must be treated carefully (Matthews et al., 2016; Sun et al., 2019). On the other hand, for the discretised ConvNP, the KL divergences can be computed, but grow in magnitude as the discretisation becomes finer, and it is not clear that the KL divergence between SPs is recovered in the resulting limit. This raises practical issues for the use of $\mathcal{L}_{\text{NPVI}}$ with the ConvNP, as the balance between the two terms depends on the choice of discretisation. In contrast, $\hat{\mathcal{L}}_{\text{ML}}$ has the advantage of being easy to specify for any map parametrising a predictive process, posing no conceptual issues for the ConvNP. In the next sections we demonstrate that ConvNPs trained with $\mathcal{L}_{\text{ML}}$ significantly outperform their counterparts trained with $\mathcal{L}_{\text{NPVI}}$, and moreover, this is often the case for ANPs as well.

5.7 EXPERIMENTS

We evaluate ConvNPs on a broad range of tasks. Our main questions are: (i) Does the ConvNP produce coherent, meaningful predictive samples? (ii) Can it leverage translation equivariance to outperform baseline methods within and beyond the training range (generalisation)? (iii) Does it learn expressive non-Gaussian predictive distributions?

EVALUATION AND BASELINES We use several approaches for evaluating NPs. First, as in (Garnelo et al., 2018b; Kim et al., 2019), we provide qualitative comparisons of samples. These allow us to see if the models display meaningful structure, quantify uncertainty, and are able to generalise spatially. Second, NPs lack closed-form likelihoods, so we evaluate *lower bounds* on their predictive log-likelihoods via importance sampling (Le et al., 2018). As these bounds can be quite loose (see Appendix E.6 for more details and a discussion on this point), they are primarily useful to show when NPs outperform baselines that admit *exact* likelihoods, such as GPs and ConvCNPs. Finally, in Section 5.7.3 we consider Bayesian optimization to evaluate the usefulness of ConvNPs for downstream tasks. In Sections 5.7.1 and 5.7.2, we compare against the ANP (Kim et al., 2019), which in prior work is trained with $\mathcal{L}_{\text{NPVI}}$. The ANP architectures used here are comparable to those in Kim et al. (2019), and have a parameter count comparable to or greater than the ConvNP. Full details are provided in Appendices E to G.

5.7.1 *1D Regression*

As in Chapter 4, we train models on samples from (i) a EQ GP, (ii) a Matérn-$\frac{5}{2}$ GP, (iii) a noisy mixture of EQ kernels GP, (iv) a weakly periodic GP, and

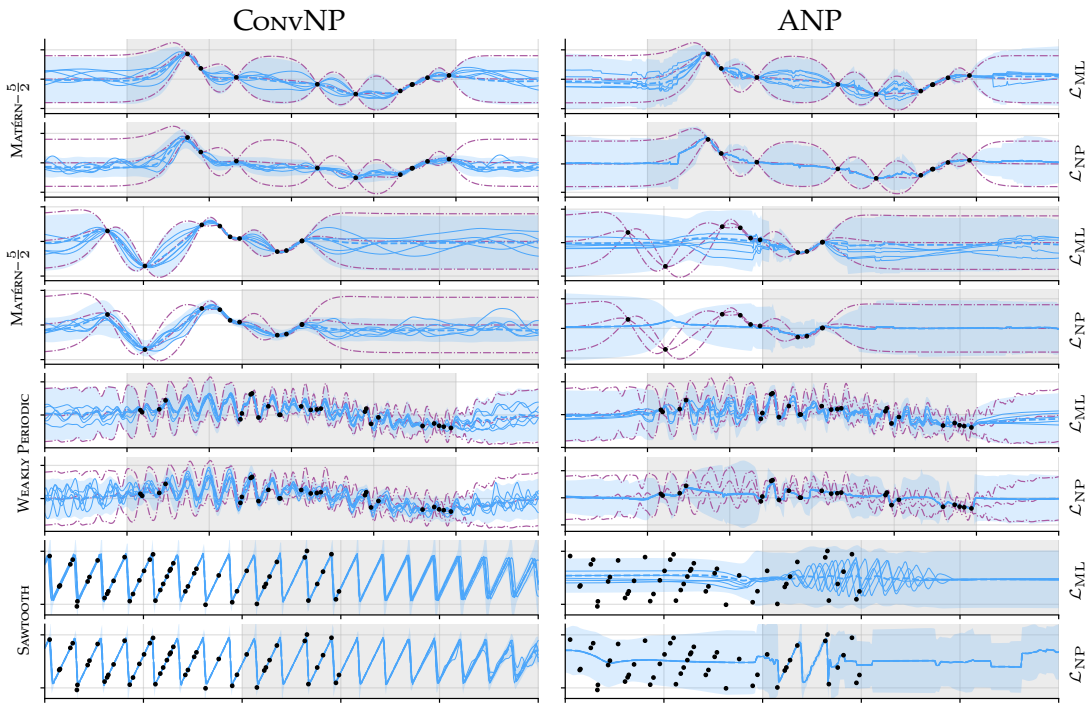


Figure 5.2: Predictions of ConvNPs and ANPs trained with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NPVI}}$, showing interpolation and extrapolation within (grey background) and outside (white background) the training range. Solid blue lines are samples, dashed blue lines are means, and the shaded blue area is $\mu \pm 2\sigma$. Purple dash–dot lines are the ground-truth GP mean and $\mu \pm 2\sigma$. ConvNP handles points outside the training range naturally, whereas this leads to catastrophic failure for the ANP. Note ANP with $\mathcal{L}_{\text{NPVI}}$ tends to collapse to deterministic samples, with all uncertainty explained with the heteroskedastic noise. In contrast, models trained with $\mathcal{L}_{\text{ML}}$ show diverse samples that account for much of the uncertainty.

(v) a non-Gaussian sawtooth process with random shifts and frequency (see Appendix E for complete details).

Qualitative analysis

Figure 5.2 shows predictive samples, where during training the models only observe data within the grey regions (training range). While samples from the ANP exhibit unnatural "kinks" and do not resemble the underlying process, the ConvNP produces smooth samples for Matérn-$\frac{5}{2}$ and samples exhibiting meaningful structure for the weakly periodic and sawtooth processes. The ConvNP also generalises gracefully beyond the training range, whereas ANP fails catastrophically.

Further, we observe that the ANP with $\mathcal{L}_{\text{NPVI}}$ collapses to deterministic samples, with the epistemic uncertainty explained using the heteroskedastic noise $\sigma_y^2(\mathbf{x}, \mathbf{z})$. Similar behaviour was also noted by Le et al. (2018). This behaviour is alleviated when training with $\hat{\mathcal{L}}_{\text{ML}}$, with much of the predictive uncertainty due to variations in the sampled functions, rather than the heteroskedastic noise.

Quantitative analysis

Table 5.1: Log-likelihood for ConvCNP, ConvNP, ANP, and NP. Each of the stochastic models was trained on each data set with $\mathcal{L}_{\mathrm{ML}}$ and $\mathcal{L}_{\mathrm{NP}}$, separately. We evaluate the models in three settings: interpolation within training range (first block), interpolation beyond training range (second block), and extrapolation beyond training range (third block). For each task, we further compare the models' performance to three models with tractable likelihoods: a ConvCNP, the ground truth GP with a full covariance matrix (GP (full)), and a diagonalised (i.e., factorised) version of the ground truth GP (GP (diag)).

		EQ	Matérn–$\frac{5}{2}$	Noisy Mixt.	Weakly Per.	Sawtooth
INTERPOLATION INSIDE TRAINING RANGE						
GP (full)		5.80 ± 0.02	$1.22 \pm 6.3\text{E}{-3}$	$1.00 \pm 4.1\text{E}{-3}$	$-0.06 \pm 4.6\text{E}{-3}$	N/A
GP (diag)		-0.59 ± 0.01	$-0.84 \pm 9.0\text{E}{-3}$	-0.89 ± 0.01	$-1.17 \pm 5.2\text{E}{-3}$	N/A
ConvCNP		-0.70 ± 0.02	-0.88 ± 0.01	-0.92 ± 0.02	$-1.19 \pm 7.0\text{E}{-3}$	1.15 ± 0.04
ConvNP	$\mathcal{L}_{\mathrm{ML}}$	-0.30 ± 0.02	-0.58 ± 0.01	-0.55 ± 0.01	$-1.02 \pm 6.0\text{E}{-3}$	2.30 ± 0.01
ANP	$\mathcal{L}_{\mathrm{ML}}$	-0.52 ± 0.01	-0.73 ± 0.01	-0.69 ± 0.01	$-1.14 \pm 6.0\text{E}{-3}$	$0.09 \pm 3.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{ML}}$	$-0.84 \pm 9.0\text{E}{-3}$	$-0.96 \pm 7.0\text{E}{-3}$	$-0.93 \pm 9.0\text{E}{-3}$	$-1.23 \pm 5.0\text{E}{-3}$	$-0.02 \pm 2.0\text{E}{-3}$
ConvNP	$\mathcal{L}_{\mathrm{NP}}$	-0.50 ± 0.02	-0.77 ± 0.01	-0.48 ± 0.02	$-1.03 \pm 8.0\text{E}{-3}$	$2.47 \pm 8.0\text{E}{-3}$
ANP	$\mathcal{L}_{\mathrm{NP}}$	-0.82 ± 0.01	-0.96 ± 0.01	-1.04 ± 0.01	$-1.37 \pm 6.0\text{E}{-3}$	$0.20 \pm 9.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{NP}}$	$-0.58 \pm 9.0\text{E}{-3}$	$-1.00 \pm 9.0\text{E}{-3}$	-0.72 ± 0.01	$-1.22 \pm 5.0\text{E}{-3}$	$-0.16 \pm 2.0\text{E}{-3}$
INTERPOLATION BEYOND TRAINING RANGE						
GP (full)		5.80 ± 0.02	$1.22 \pm 6.3\text{E}{-3}$	$1.00 \pm 4.1\text{E}{-3}$	$-0.06 \pm 4.6\text{E}{-3}$	N/A
GP (diag)		-0.59 ± 0.01	$-0.84 \pm 9.0\text{E}{-3}$	-0.89 ± 0.01	$-1.17 \pm 5.2\text{E}{-3}$	N/A
ConvCNP		-0.69 ± 0.02	-0.87 ± 0.01	-0.94 ± 0.02	$-1.19 \pm 7.0\text{E}{-3}$	1.11 ± 0.04
ConvNP	$\mathcal{L}_{\mathrm{ML}}$	-0.30 ± 0.02	-0.58 ± 0.01	-0.56 ± 0.01	$-1.03 \pm 6.0\text{E}{-3}$	2.29 ± 0.02
ANP	$\mathcal{L}_{\mathrm{ML}}$	$-1.35 \pm 6.0\text{E}{-3}$	$-1.39 \pm 7.0\text{E}{-3}$	$-1.65 \pm 5.0\text{E}{-3}$	$-1.35 \pm 4.0\text{E}{-3}$	$-0.17 \pm 1.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{ML}}$	$-2.70 \pm 3.0\text{E}{-3}$	$-2.60 \pm 3.0\text{E}{-3}$	$-2.82 \pm 3.0\text{E}{-3}$	-	$-0.03 \pm 2.0\text{E}{-3}$
ConvNP	$\mathcal{L}_{\mathrm{NP}}$	-0.48 ± 0.02	-0.79 ± 0.01	-0.48 ± 0.02	$-1.04 \pm 8.0\text{E}{-3}$	$2.47 \pm 8.0\text{E}{-3}$
ANP	$\mathcal{L}_{\mathrm{NP}}$	-1.91 ± 0.03	$-1.48 \pm 4.0\text{E}{-3}$	$-1.85 \pm 7.0\text{E}{-3}$	-1.66 ± 0.01	$-0.30 \pm 4.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{NP}}$	-13.7 ± 0.82	-3.96 ± 0.04	-3.80 ± 0.02	-	-4.98 ± 0.02
EXTRAPOLATION BEYOND TRAINING RANGE						
GP (full)		$4.29 \pm 6.2\text{E}{-3}$	$0.82 \pm 4.3\text{E}{-3}$	$0.66 \pm 2.2\text{E}{-3}$	$-0.33 \pm 3.4\text{E}{-3}$	N/A
GP (diag)		$-1.40 \pm 5.0\text{E}{-3}$	$-1.41 \pm 4.8\text{E}{-3}$	$-1.72 \pm 6.2\text{E}{-3}$	$-1.40 \pm 4.0\text{E}{-3}$	N/A
ConvCNP		$-1.41 \pm 6.0\text{E}{-3}$	$-1.41 \pm 7.0\text{E}{-3}$	$-1.73 \pm 8.0\text{E}{-3}$	$-1.41 \pm 6.0\text{E}{-3}$	0.27 ± 0.02
ConvNP	$\mathcal{L}_{\mathrm{ML}}$	$-1.09 \pm 5.0\text{E}{-3}$	$-1.11 \pm 5.0\text{E}{-3}$	$-1.30 \pm 4.0\text{E}{-3}$	$-1.24 \pm 4.0\text{E}{-3}$	1.61 ± 0.02
ANP	$\mathcal{L}_{\mathrm{ML}}$	$-1.29 \pm 6.0\text{E}{-3}$	$-1.29 \pm 5.0\text{E}{-3}$	$-1.55 \pm 5.0\text{E}{-3}$	$-1.34 \pm 5.0\text{E}{-3}$	$-0.25 \pm 2.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{ML}}$	$-2.23 \pm 4.0\text{E}{-3}$	$-2.08 \pm 3.0\text{E}{-3}$	$-2.50 \pm 4.0\text{E}{-3}$	$-1.39 \pm 4.0\text{E}{-3}$	$-0.06 \pm 2.0\text{E}{-3}$
ConvNP	$\mathcal{L}_{\mathrm{NP}}$	-1.21 ± 0.01	-1.31 ± 0.01	-1.19 ± 0.01	$-1.51 \pm 8.0\text{E}{-3}$	$2.10 \pm 7.0\text{E}{-3}$
ANP	$\mathcal{L}_{\mathrm{NP}}$	$-1.44 \pm 6.0\text{E}{-3}$	$-1.45 \pm 6.0\text{E}{-3}$	$-1.77 \pm 7.0\text{E}{-3}$	$-1.46 \pm 6.0\text{E}{-3}$	$-0.20 \pm 2.0\text{E}{-3}$
NP	$\mathcal{L}_{\mathrm{NP}}$	-5.85 ± 0.05	$-2.65 \pm 3.0\text{E}{-3}$	-4.06 ± 0.04	$-1.49 \pm 5.0\text{E}{-3}$	$-1.99 \pm 6.0\text{E}{-3}$

Table 5.1 compares lower bounds on the log-likelihood for ConvNP with our proposed $\hat{\mathcal{L}}_{\text{ML}}$ objective, as well as a NP and ANP with both $\hat{\mathcal{L}}_{\text{ML}}$ and the standard $\mathcal{L}_{\text{NPVI}}$ objective. The comparison is carried out in three separate regimes:

- *Interpolation inside training range*: both $\mathcal{D}_c$ and $\mathcal{D}_t$ contain data only within the training range. This is the "vanilla" regime often used to evaluate members of the NPF with GP experiments.
- *Interpolation beyond training range*: both $\mathcal{D}_c$ and $\mathcal{D}_t$ may contain observations from outside the training range. This setting evaluates the models' ability to extrapolate to data observed outside the training range.
- *Extrapolation beyond training range*: $\mathcal{D}_c$ contains data within the training range, $\mathcal{D}_t$ may contain data from outside the training range. This setting evaluates models' ability to make sensible predictions outside the training range, "far" from observed data.

We also show three *exact* log-likelihoods: (i) the ground-truth GP (full) (ii) the ground-truth GP with diagonalised predictions (diag), and (iii) ConvCNP. The ConvCNP performs on par with GP (diag), which is the optimal factorized predictive.[3] Both the ConvNP and ANP trained with $\hat{\mathcal{L}}_{\text{ML}}$ lower-bounds are consistently higher than the GP (diag) in the interpolation setting, demonstrating that correlated predictives significantly improve predictive performance. This is not the case for the NP, or when the models are trained with $\mathcal{L}_{\text{NPVI}}$. The ConvNP consistently and significantly outperforms the other models on all tasks.

Further, the ConvNP performs similarly inside and outside its training range, demonstrating that translation equivariance aids generalisation; this is in contrast to the ANP and NP, which fail catastrophically outside the training range. Finally, we can see that the ANP tends to achieve significantly higher likelihoods when trained with $\hat{\mathcal{L}}_{\text{ML}}$ as opposed to $\mathcal{L}_{\text{NPVI}}$. While this trend appears consistent across our experiments, we believe that further investigation is required to conclusively determine that one objective is strictly superior to the other.

5.7.2 *Image Completion*

We evaluate ConvNPs on image completion tasks focusing on spatial generalisation. To test this, we again consider zero-shot multi MNIST (ZSMM), where we train on single MNIST digits but test on two MNIST digits on a larger canvas. We randomly translate the digits during training, so the generative SP is stationary. The black background on MNIST causes difficulty with heteroskedastic noise,

3 We note that this work was conducted after the work detailed in Chapter 4. Several implementation details that lead to improved performance of the ConvCNP had been introduced at this point, which accounts for the improved quantitative performance of the ConvCNP in Table 5.1.

Table 5.2: Test log-likelihood lower bounds for image completion (5 runs).

	MNIST		CelebA32		SVHN		ZSMM	
	$\mathcal{L}_{\text{ML}}$	$\mathcal{L}_{\text{NP}}$	$\mathcal{L}_{\text{ML}}$	$\mathcal{L}_{\text{NP}}$	$\mathcal{L}_{\text{ML}}$	$\mathcal{L}_{\text{NP}}$	$\mathcal{L}_{\text{ML}}$	$\mathcal{L}_{\text{NP}}$
ConvNP	**2.11** ± 0.01	0.99 ± 0.42	**6.92** ± 0.10	-0.27 ± 0.00	**9.89** ± 0.09	0.17 ± 0.00	**4.58** ± 0.04	0.14 ± 0.00
ANP	1.66 ± 0.03	1.64 ± 0.03	5.98 ± 0.08	6.04 ± 0.10	9.18 ± 0.08	8.91 ± 0.06	-10.8 ± 1.99	-6.45 ± 0.99

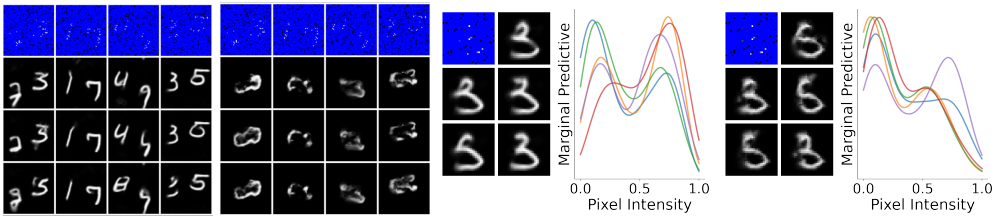


Figure 5.3: Left two plots: predictive samples on zero-shot multi MNIST. Left-most plot is ConvNP, left-centre is ANP. Right two plots: samples and marginal predictives on standard MNIST. We plot the density of the five marginals that maximize Sarle's bimodality coefficient Ellison, 1987. We use $\mathcal{L}_{\text{ML}}$ for training. Blue pixels are not in the context set.

as the models can obtain high likelihood by predicting the background with high confidence whilst ignoring the digits. Hence for MNIST and ZSMM we use homoskedastic noise $\sigma_y^2(\mathbf{z})$. Figure 5.3 (left and centre-left) demonstrate that the ANP fails to generalise spatially, whereas this is naturally handled by the ConvNP.

We also test the ConvNP's ability to learn non-Gaussian predictive distributions. Figure 5.3 (centre-right) demonstrates that the ConvNP learns multi-modal predictives, enabling the generation of diverse yet coherent samples. Finally, a quantitative comparison of models using log-likelihood lower bounds is provided in Table 5.2, where a ConvNP trained with $\hat{\mathcal{L}}_{\text{ML}}$ consistently achieves the highest values. Appendix F provides details regarding the data, architectures, and protocols used in our image experiments. In Appendix F.7, we provide samples and further quantitative comparisons of models trained on SVHN (Netzer et al., 2011), MNIST LeCun et al., 1989, and 32×32 CelebA Netzer et al., 2011 in a range of scenarios, along with full experimental details.

5.7.3 *Environmental Data*

We next consider a real-world spatial data set, ERA5-Land (Balsamo et al., 2015), containing measured environmental variables at a $\sim$9 km spacing across the globe. We consider predicting daily precipitation y at spatial position $\mathbf{x}$. We also provide the model with orography (elevation) and temperature values. We choose a large region of central Europe as our train set, and use regions east, west, and south as held-out test sets. For such tasks, models must be able to make

Table 5.3: Joint predictive log-likelihoods (LL) and RMSEs on ERA5-Land, averaged over 1000 tasks. For each model / task, we plot the mean and standard error of the log-likelihood over the tasks. Bolded values correspond to best performing model, and all models with overlapping error bars.

		Central (train)	West (test)	East (test)	South (test)
LL	ConvNP	**4.47** ± 0.07	**4.55** ± 0.08	**5.07** ± 0.07	**4.65** ± 0.08
	GP	3.33 ± 0.06	3.65 ± 0.06	4.07 ± 0.06	3.34 ± 0.06
RMSE ($\times 10^{-2}$)	ConvNP	**5.72** ± 0.33	**5.77** ± 0.37	**3.23** ± 0.22	**6.92** ± 0.39
	GP	**6.26** ± 0.30	**5.75** ± 0.29	**3.10** ± 0.18	7.94 ± 0.44

predictions at locations spanning a range different from the training set, inhibiting the deployment of NPs not equipped with translation equivariance. To sample a task at train time, we sample a random date between 1981 and 2020, then sample a sub-region within the train region, which is split into context and target sets. In this section, we train using $\hat{\mathcal{L}}_{\mathrm{ML}}$. See Appendix G for experimental details.

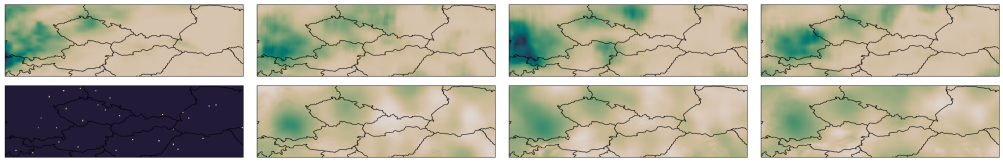


Figure 5.4: Samples from the predictive processes overlaid on central Europe. Darker colours show higher precipitation. Left-most column depicts the (top) ground truth values and (bottom) context set observed by the models. In the context set panel (bottom left), coloured pixels represent observed context points. Remaining plots are samples from model predictive posteriors for (top) ConvNP and (bottom) GP. GP samples often take negative values (lighter than ground truth data, see Appendix G for a discussion), whereas the NP has learned to produce non-negative samples which capture the *sparsity* of precipitation. To produce high-quality samples, the model is trained on subregions roughly the size of the lengthscale of the precipitation process. More samples in Appendix G.2.

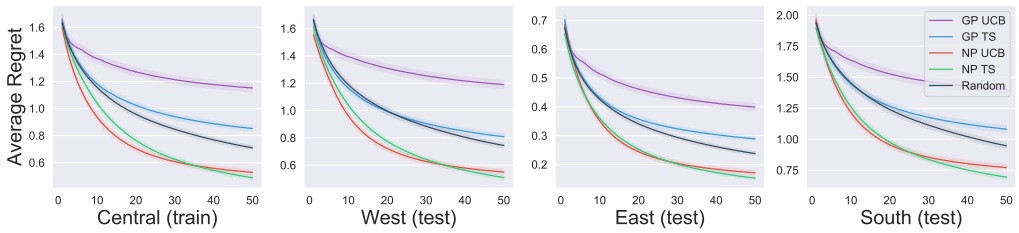


Figure 5.5: Average regret plotted against number of points queried, averaged over 5000 tasks. Plots compare a GP and ConvNP, observing the same context sets, and using UCB and Thompson sampling to perform Bayesian optimisation.

5.7.4 *Prediction*

We first evaluate the ConvNP's predictive performance, comparing to a GP trained individually on each task as a baseline. In about 10% of tasks, the GP obtains a poor likelihood (< 0 nats); we remove these outliers from the evaluation. The results are shown in Table 5.3. The ConvNP and GP have comparable RMSEs except on south, where the ConvNP outperforms the GP. However, the ConvNP consistently outperforms the GP in log-likelihood, which is expected for the following reasons: (i) the GP does not share information between tasks and hence is prone to over-fitting on small context sets, resulting in overconfident predictions; and (ii) the ConvNP can learn non-Gaussian predictive densities (illustrated in Appendix G.2). Figure 5.4 shows samples from the predictive process of a ConvNP and GP, over the whole of the train region. This demonstrates spatial extrapolation, as the ConvNP is trained only on random subregions. Further samples are provided in Appendix G.2.

5.7.5 *Bayesian optimization*

We demonstrate the usefulness of the ConvNP in a downstream task by considering a toy Bayesian optimisation problem (Brochu et al., 2010; Snoek et al., 2012), where the goal is to identify the location with heaviest rainfall on a given day. We also test the ConvNP's spatial generalisation, by optimising over larger regions (for central, west, and south) than the model was trained on. We test both Thompson sampling (TS; Thompson, 1933) and upper confidence bounds (UCB; Auer, 2002) as methods for acquiring points. Note that TS requires coherent samples. The results are shown in Figure 5.5. On all data sets, ConvNP TS and UCB significantly outperform the random baseline by the 50th iteration; the GP does not reliably outperform random. We hypothesize this is due to its overconfidence, in line with the results on prediction.

5.8 SUMMARY AND CONCLUSIONS

We have introduced the ConvNP, a translation equivariant map from data sets to predictive SPs. ConvNPs extend ConvCNPs to model rich predictive distributions with dependencies across the target points, largely overcoming the limitations of ConvCNPs discussed in Section 5.2.

However, as with other members of the LNPF, this comes at the cost of sacrificing the tractability of the likelihood $p_{\boldsymbol{\theta}}(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c)$. This cost has several important implications. First, we must resort to biased estimators of the likelihood for training, rendering guarantees on the recovery of π_P ineffective. Some consequences of

this are apparent when working with members of the LNPF, and, for example, can be seen when observing the predictive distributions produced by the models in Section 5.7.1. More generally, it is difficult to reason about artefacts that arise when training members of the LNPF with either $\hat{\mathcal{L}}_{\text{ML}}$ or $\mathcal{L}_{\text{NPVI}}$.

Second, evaluation of the models is complicated by the lack of a tractable evaluation metric. We must resort to evaluating the models using lower bounds on the quantities of interest, which may be quite loose. This means that we must rely on more qualitative evaluations of the model, limiting scalability of developing the models, or devise more sophisticated evaluation procedures. Finally, both objectives for training members of the LNPF require *sampling based approximations* incurring significant computational overhead in training the models. This is particularly true for $\hat{\mathcal{L}}_{\text{ML}}$, which in our experiments requires on the order of 20-30 samples during training to achieve the desired levels of performance.

Given these limitations, my informal conclusion is that users should prefer working with members of the CNPF if they suit the needs of the application. Members of the LNPF should be invoked only if there is reason to believe the members of the CNPF are not suited for task, e.g. if (i) coherent samples are necessary to perform the task, (ii) the task at hand is dominated by the dependencies across target points, or (iii) if the design of an appropriate likelihood function is overly complex for a human expert. In other cases, my recommendation would be to invoke members of the CNPF, which are easier and faster to train.

We thus conclude our exploration of translation equivariant members of the NPF. In the next section, we turn our attention to the construction and deployment of members of the NPF for *few-shot image classification*.

6

CONDITIONAL NEURAL ADAPTIVE PROCESSES FOR FEW-SHOT CLASSIFICATION

We now turn our attention to the task of *few-shot image classification* (Fei-Fei et al., 2006; Lake et al., 2015). In particular, we consider the case where tasks arise from complicated generative processes inducing a distribution over a broad set of tasks. Extending the NPF to the few-shot classification setting involves several challenges particular to the application. As such, much of the chapter is concerned with modelling developments that address these issues. The central contribution of this chapter is CNAPs, a specialisation of the CNPF to few-shot image classification that achieves state-of-the-art performance on the challenging Meta-Dataset benchmark (Triantafillou et al., 2020).

6.1 INTRODUCTION

In this chapter, we extend the CNPF so as to be well-positioned with respect to important tradeoffs for the few-shot, multi-task classification setting. We name the resulting model Conditional Neural Adaptive Processes (CNAPs). As other members of the CNPF, CNAPs directly model the desired predictive distribution (Geisser, 1983, 2017). Unlike existing members of the CNPF, CNAPs handle varying way classification tasks and introduce a parametrization and training procedure enabling the model to *learn to adapt* the feature representation for classification of diverse tasks at test time.

The work in this chapter is based on the publication "Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes" (Requeima et al., 2019). The research was conducted in collaboration with my co-first authors James Requeima and John Bronskill, as well as Sebastian Nowozin and Richard E. Turner. I contributed equally to all aspects of the work, including conceptualisation of the model, writing the related software,[1] devising the experiments, and writing of the paper. The central contributions of this chapter are to

- introduce the CNAPs modelling framework for multi-task classification,
- demonstrate that CNAPs achieve state-of-the-art performance on the Meta-Dataset benchmark, often by comfortable margins and at a fraction of the time required by competing methods, and

1 The software to implement CNAPs and reproduce the experiments in this section can be found at https://github.com/cambridge-mlg/cnaps

- demonstrate that CNAPs are able to adapt to held-out tasks in a fraction of the time required by competing methods.

6.2 MOTIVATION

Existing work in few-shot image classification typically considers homogeneous task distributions at train and test-time (e.g., *mini*-ImageNet (Vinyals et al., 2016) and Omniglot (Lake et al., 2015)) that therefore require only minimal adaptation. To handle the more challenging case of different task distributions we invoke the NPF and propose several modelling extensions tailored towards few-shot classification and increased capacity.

Current approaches to meta-learning and few-shot learning for classification are characterized by two fundamental tradeoffs. (i) The number of parameters that are adapted to each task. One approach adapts only the top, or head, of the classifier leaving the feature extractor fixed (Snell et al., 2017; Gordon et al., 2019). While useful in simple settings, this approach is prone to under-fitting when the task distribution is heterogeneous (e.g. Triantafillou et al., 2020). Alternatively, we can adapt all parameters in the feature extractor (as in Finn et al., 2017; Nichol and Schulman, 2018) thereby increasing fitting capacity, but incurring a computation cost and opening the door to over-fitting in the low-shot regime. What is needed is a middle ground which strikes a balance between model capacity and reliability of the adaptation. (ii) The adaptation mechanism. Many approaches use gradient-based adaptation (Finn et al., 2017; Nichol and Schulman, 2018). While this approach can incorporate training data in a flexible way, it is computationally inefficient at test-time, may require expertise to tune the optimization procedure, and is again prone to over-fitting. Conversely, function approximators can be used to directly map training data to the desired parameters (e.g. Gordon et al., 2019; Oreshkin et al., 2018). We refer to this as *amortization*. This yields fixed-cost adaptation mechanisms, and enables greater sharing across training tasks. However, it may under-fit if the function approximation is not sufficiently flexible, and high-capacity function approximators require a large number of training tasks to be learned.

We develop a class of models that specialises the CNPF to few-shot classification, and is well-positioned with respect to these important tradeoffs, which we coin *Conditional Neural Adaptive Processes* (CNAPs). CNAPs utilise i) a classification model with shared global parameters and a small number of task-specific parameters. We demonstrate that by identifying a small set of key parameters, the model can balance the trade-off between flexibility and robustness. ii) A rich adaptation neural network with a novel auto-regressive parametrisation that avoids under-fitting while proving easy to train in practice.

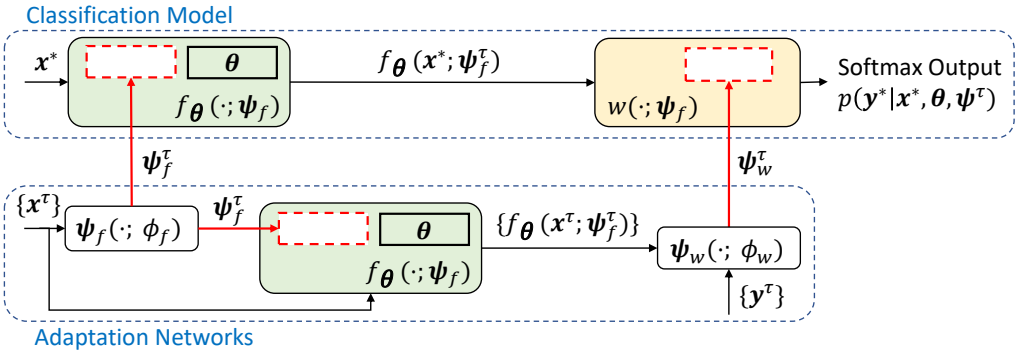


Figure 6.1: Computational diagram depicting the CNAPs model class. Red boxes imply parameters in the model architecture supplied by adaptation networks. Blue shaded boxes depict the feature extractor and the gold box depicts the linear classifier.

6.3 MODEL DESIGN

To facilitate flexible parametrisations of the CNPF, we introduce a hierarchy of parameters in the model. We will use $\boldsymbol{\theta}$ to denote global parameters, that are shared across all tasks. As is standard for the CNPF, there is also set of variables (these will be parameters for CNAPs) that are local at the task level, which we denote ψ_f. Finally, to specialise the CNPF to the classification setting, we also introduce a set of variables (parameters) that are local at the *class level*, which we denote ψ_w. We sometimes collectively refer to all local parameters as $\psi = \{\psi_f, \psi_w\}$.

As usual, we consider a scenario where each task $\xi = (\mathcal{D}_c, \mathcal{D}_t)$ consists of a context and target set. However, we note that in this setting the labels correspond discrete categories, i.e. $y \in \{1, \ldots, C_\tau\}$, where τ indexes tasks in the meta-dataset Ξ. Importantly, the number of categories may vary across tasks. Context sets consist of $\mathcal{D}_c^\tau = (\mathbf{x}_n, y_n)_{n=1}^{N_\tau}$. Target sets similarly consist of $\mathcal{D}_t^{(\tau)} = (\mathbf{x}_m, y_m)_{m=1}^{M_\tau}$, where the labels are observed only during training.

As all members of the CNPF, CNAPs models the predictive distribution for each label independently as

$$p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right) = \prod_{m=1}^{M} p_{\boldsymbol{\theta}}\left(y_m|\mathbf{x}_m, \psi^\tau = \psi_{\boldsymbol{\phi}}\left(\mathcal{D}_c\right)\right). \tag{6.1}$$

The local parameters ψ are produced by a function $\psi_{\boldsymbol{\phi}}(\cdot)$ that acts on $\mathcal{D}_c$. The function $\psi_{\boldsymbol{\phi}}(\cdot)$ has another set of global parameters $\boldsymbol{\phi}$ called *adaptation network parameters*. The learnable parameters of the model are thus $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

CNAPs are characterized by a number of design choices, made specifically for the multi-task image classification setting. The model employs global parameters $\boldsymbol{\theta}$ that are trained off-line to capture high-level features, facilitating transfer

and multi-task learning. Whereas existing members of the CNPF define $\boldsymbol{\psi}^\tau$ to be a fixed dimensional vector used as an input to the model, CNAPs instead let $\boldsymbol{\psi}^\tau$ be specific parameters of the model itself. We discuss our choices (and associated tradeoffs) for these parameters below. Finally, CNAPs employ a novel auto-regressive parametrisation of $\boldsymbol{\psi}_{\boldsymbol{\phi}}(\cdot)$ that significantly improves performance. An schematic of CNAPs is provided in Figure 6.1.

6.3.1 *Specification of the classifier: $\boldsymbol{\theta}$ and $\boldsymbol{\psi}^\tau$*

We begin by specifying the classifier's global parameters $\boldsymbol{\theta}$ followed by how these are adapted by the local parameters $\boldsymbol{\psi}^\tau$.

Global Classifier Parameters

The global classifier parameters will parametrise a feature extractor $f_{\boldsymbol{\theta}}(\mathbf{x})$ whose output is fed into a linear classifier, described below. A natural choice for $f_{\boldsymbol{\theta}}(\cdot)$ in the image setting is a convolutional neural network, e.g., a ResNet (He et al., 2016). In what follows, we assume that the global parameters $\boldsymbol{\theta}$ are fixed and known. In Section 6.4 we discuss the training of $\boldsymbol{\theta}$.

Class-specific classifier parameters: linear classification weights

The final classification layer must be task-specific as each task involves distinguishing a potentially unique set of classes. We use a task specific affine transformation of the feature extractor output, followed by a softmax activation. The class-specific weights are denoted $\boldsymbol{\psi}_w^\tau \in \mathbb{R}^{d_f \times C^\tau}$ (suppressing the biases to simplify notation), where d_f is the dimension of the feature extractor output $f_{\boldsymbol{\theta}}(\mathbf{x})$ and C^τ is the number of classes in task τ.

Task-specific classifier parameters: feature extractor parameters

A sufficiently flexible model must have capacity to adapt its feature representation $f_{\boldsymbol{\theta}}(\cdot)$ as well as the classification layer (e.g. compare the optimal features required for ImageNet versus Omiglot). We therefore introduce a set of local feature extractor parameters $\boldsymbol{\psi}_f^\tau$, and denote $f_{\boldsymbol{\theta}}(\cdot)$ the *unadapted* feature extractor, and $f_{\boldsymbol{\theta}}(\cdot; \boldsymbol{\psi}_f^\tau)$ the feature extractor adapted to task τ.

It is critical in few-shot multi-task learning to adapt the feature extractor in a parameter-efficient manner. Unconstrained adaptation of all the feature extractor parameters (e.g. by fine-tuning Finn et al., 2017) gives flexibility, but it is also slow and prone to over-fitting (Triantafillou et al., 2020). Instead, we employ linear modulation of the convolutional feature maps as proposed by Perez et al. (2018),

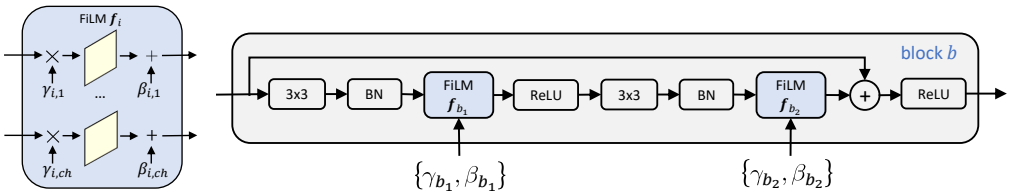


Figure 6.2: (Left) A FiLM layer operating on convolutional feature maps indexed by channel ch. (Right) How a FiLM layer is used within a basic Residual network block (He et al., 2016).

which adapts the feature extractor through a relatively small number of task specific parameters.

A Feature-wise Linear Modulation (FiLM) layer (Perez et al., 2018) scales and shifts the i^{th} unadapted feature map $\mathbf{f}_i$ in the feature extractor $\text{FiLM}(\mathbf{f}_i; \gamma_i^\tau, \beta_i^\tau) = \gamma_i^\tau \mathbf{f}_i + \beta_i^\tau$ using two task specific parameters, γ_i^τ and β_i^τ. Figure 6.2 (left) illustrates a FiLM layer operating on a convolutional layer, and Figure 6.2 (right) illustrates how a FiLM layer can be added to a standard Residual network block (He et al., 2016). A key advantage of FiLM layers is that they enable expressive feature adaptation while adding only a small number of parameters (Perez et al., 2018). For example, in our implementation we use a ResNet18 with FiLM layers after every convolutional layer. The set of task specific FiLM parameters ($\boldsymbol{\psi}_f^\tau = \{\boldsymbol{\gamma}_i^\tau, \boldsymbol{\beta}_i^\tau\}$) constitute fewer than 0.7% of the parameters in the model. Despite this, as we show in Section 5.7, they allow the model to adapt to a broad class of data sets.

6.3.2 *Computing the local parameters via adaptation networks*

The previous sections have specified the form of the classifier $p_{\boldsymbol{\theta}}(\mathbf{y}_T | \mathbf{X}_T, \boldsymbol{\psi}^\tau)$ in terms of the global and task specific parameters, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}^\tau = \{\boldsymbol{\psi}_f^\tau, \boldsymbol{\psi}_w^\tau\}$. Conceptually, we could now learn the local parameters separately for every task τ e.g. via optimization. While in practice this is feasible for small numbers of tasks (see e.g., Rebuffi et al., 2017, 2018)), this approach is computationally demanding, requires expert oversight (e.g. for tuning early stopping), and can over-fit in the low-data regime.

Instead, CNAPs invokes the central ideas of the CNPF, i.e. using a function, such as a neural network, that takes the context set $\mathcal{D}_c$ as an input and returns the task-specific parameters, $\boldsymbol{\psi}^\tau = \boldsymbol{\psi}_{\boldsymbol{\phi}}(\mathcal{D}_c)$. Thus, the function $\boldsymbol{\psi}(\cdot)$ is playing the role of the *encoder* in standard CNPF models. Sacrificing some of the flexibility of the optimisation approach, this method is comparatively cheap computationally (only involving a forward pass through the adaptation network), automatic (with no need for expert oversight), and employs explicit parameter sharing (via $\boldsymbol{\phi}$) across the training tasks.

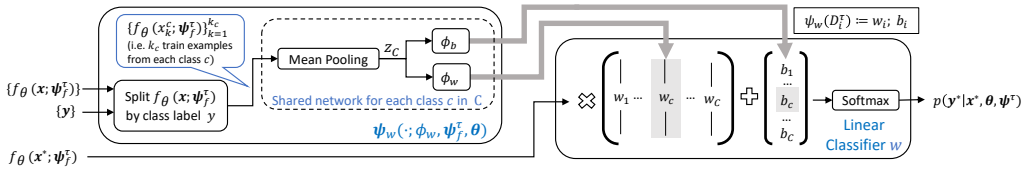


Figure 6.3: Implementation of functional representation of the class-specific parameters ψ_w. In this parametrisation, ψ_w^c are the linear classification parameters for class c, and ϕ_w are the learnable parameters.

Adaptation Network: Linear Classifier Weights

CNAPs represents the linear classifier weights ψ_w^τ as a parametrised function of the form $\psi_w^\tau = \psi_w(\mathcal{D}_c; \phi_w, \psi_f, \theta)$, denoted $\psi_w(\mathcal{D}_c)$ for brevity. There are three challenges with this approach: (i) the dimensionality of the weights depends on the task (ψ_w^τ is a matrix with a column for each class, see Figure 6.3) and thus the network must output parameters of varying dimensionalities; (ii) the number of observations in $\mathcal{D}_c$ will depend on the task. Thus, the network must be able to take inputs of variable cardinality; The latter challenge is standard for members of the CNPF, and is handled with a permutation invariant function, as discussed in previous chapters. To handle the first challenge, we follow Gordon et al. (2019). First, each column of the weight matrix is generated independently, using only the context points from that class, i.e.,

$$\psi_w^\tau = \left[\psi_w\left(D_1^\tau\right), \quad \ldots, \quad \psi_w\left(D_C^\tau\right)\right], \tag{6.2}$$

an approach which scales to arbitrary numbers of classes.

Intuitively, the classifier weights should be determined by the representation of the data points emerging from the adapted feature extractor. We therefore input the adapted feature representation of the data points into the network, rather than the raw data points (hence the dependency of ψ_w on ψ_f and θ). To summarize, $\psi_w(\cdot)$ is a function *on sets* that accepts as input a set of *adapted* feature representations from $\mathcal{D}_c$, and outputs the c^{th} column of the linear classification matrix, i.e.,

$$\psi_w\left(\mathcal{D}_c^\tau; \phi_w, \psi_f, \theta\right) = \psi_w\left(\{f_\theta\left(\mathbf{x}_m; \psi_f\right) | \mathbf{x}_m \in \mathcal{D}_c^\tau, y_m = c\}; \phi_w\right). \tag{6.3}$$

Here ϕ_w are learnable parameters of $\psi_w(\cdot)$. See Figure 6.3 for an illustration.

Adaptation Network: Feature Extractor Parameters

CNAPs represents the task-specific feature extractor parameters ψ_f^τ, comprising the parameters of the FiLM layers γ^τ and β^τ in our implementation, as a parametrised function of the context-set $\mathcal{D}_c$. Thus, $\psi_f(\cdot; \phi_f, \theta)$ is a collection of

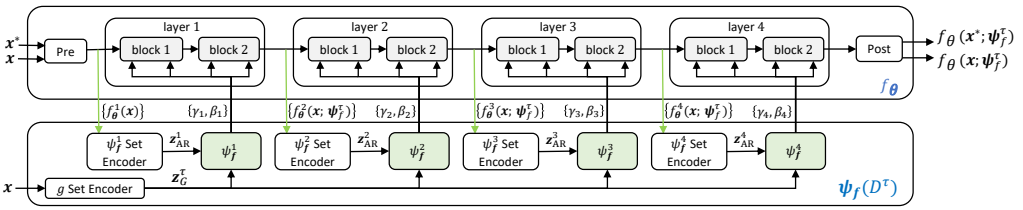


Figure 6.4: Implementation of the feature-extractor: an independently learned set encoder g provides a fixed context that is concatenated to the (processed) activations of $\mathbf{x}$ from the previous ResNet block. The inputs $\mathbf{z}_i = (\mathbf{z}_{\mathrm{G}}^\tau, \mathbf{z}_{\mathrm{AR}}^i)$ are then fed to $\boldsymbol{\psi}_f^i(\cdot)$, which outputs the FiLM parameters for layer i. Green arrows correspond to propagation of auto-regressive representations. Note that the auto-regressive component $\mathbf{z}_{\mathrm{AR}}^i$ is computed by processing the *adapted* activations $\{f_\theta^i(\mathbf{x}; \boldsymbol{\psi}_f^\tau)\}$ of the previous convolutional block.

functions (one for each FiLM layer) with parameters $\boldsymbol{\phi}_f$, many of which are shared across functions. We denote the function generating the parameters for the i^{th} FiLM layer $\boldsymbol{\psi}_f^i(\cdot)$ for brevity.

Our experiments (Section 5.7) show that this mapping requires careful parametrisation. We propose a novel parametrisation that improves performance in complex settings with diverse data sets. Our implementation contains two components: a task-specific representation that provides context about the task to all layers of the feature extractor (denoted $\mathbf{z}_{\mathrm{G}}^\tau$), and an auto-regressive component that provides information to deeper layers in the feature extractor concerning how shallower layers have adapted to the task (denoted $\mathbf{z}_{\mathrm{AR}}^i$). The input to the $\boldsymbol{\psi}_f^i(\cdot)$ network is $\mathbf{z}_i = (\mathbf{z}_{\mathrm{G}}^\tau, \mathbf{z}_{\mathrm{AR}}^i)$. The representation $\mathbf{z}_{\mathrm{G}}^\tau$ is computed for every task τ by passing the inputs $\mathbf{x}_n^\tau$ through a global set encoder g with parameters in $\boldsymbol{\phi}_f$.

To adapt the l^{th} layer in the feature extractor, it is useful for the system to have access to the representation of task-relevant inputs from layer $l-1$. While $\mathbf{z}_G$ could in principle encode how layer $l-1$ has adapted, we opt to provide this information directly to the adaptation network adapting layer l by passing the adapted activations from layer $l-1$. The auto-regressive component $\mathbf{z}_{\mathrm{AR}}^i$ is computed by processing the *adapted* activations of the previous convolutional block with a layer-specific set encoder (except for the first residual block, whose auto-regressive component is given by the *un-adapted* initial pre-processing stage in the ResNet). Both the global and all layer-specific set-encoders are implemented as permutation invariant functions (Zaheer et al., 2017; Qi et al., 2017a) (see Appendix H.2 for details). The full parametrisation is illustrated in Figure 6.4, and the architecture of $\boldsymbol{\psi}_f^i(\cdot)$ networks is illustrated in Figure 6.5.

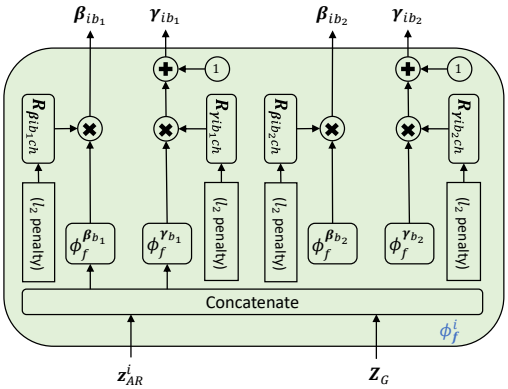


Figure 6.5: Adaptation network ϕ_f. $\mathrm{R}_{\gamma i b_j ch}$ and $\mathrm{R}_{\beta i b_j ch}$ denote a vector of regularization weights that are learned with an l_2 penalty.

6.4 TRAINING PROCEDURE

Having specified the model, we now describe how to train the global classifier parameters $\boldsymbol{\theta}$ and the adaptation network parameters $\boldsymbol{\phi} = \{\boldsymbol{\phi}_f, \boldsymbol{\phi}_w\}$.

6.4.1 *Training the Global Parameters*

A natural approach to training the model, which we have used throughout the thesis, would be to maximize the likelihood of the training data jointly over $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. However, as we demonstrate in Section 6.6.2, in this setting, it is crucially important to adopt a two stage process instead. In the first stage, $\boldsymbol{\theta}$ are trained on a large dataset (e.g. the training set of ImageNet Krizhevsky et al., 2012; Triantafillou et al., 2020) in a full-way classification procedure, mirroring standard pre-training. Second, $\boldsymbol{\theta}$ are fixed and $\boldsymbol{\phi}$ are trained using episodic training over all meta-training data sets in the multi-task setting. We hypothesize that two-stage training is important as during the second stage, $\boldsymbol{\phi}_f$ are trained to adapt $f_{\boldsymbol{\theta}}(\cdot)$ to tasks τ by outputting $\boldsymbol{\psi}_f^\tau$. As $\boldsymbol{\theta}$ has far more capacity than $\boldsymbol{\psi}_f^\tau$, if they are trained in the context of all tasks, there is no need for $\boldsymbol{\psi}_f^\tau$ to adapt the feature extractor, resulting in little-to-no training signal for $\boldsymbol{\phi}_f$ and poor generalisation. Further, as discussed in Section 6.6.2, end-to-end training in a single procedure proved unstable, often diverging. We hypothesize that training stability is related to the implementation of batch normalization (more details in Section 6.6.2), but we leave an in-depth investigation of this matter for future work. Finally, fixing $\boldsymbol{\theta}$ during meta-training is desirable as it results in a dramatic decrease in training time.

Algorithm 6: Stochastic Objective Estimator for Meta-Training CNAPs

Input: Model parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$, Task-data $\xi = \{\mathbf{X}_T, \mathbf{y}_T, \mathcal{D}_c\}$

$\boldsymbol{\psi}_f^\tau \leftarrow \boldsymbol{\psi}_f\left(\{f_{\boldsymbol{\theta}}(\mathbf{x}_n) | \mathbf{x} \in \mathcal{D}_c\}; \boldsymbol{\phi}_f\right)$

$\boldsymbol{\psi}_c^\tau \leftarrow \boldsymbol{\psi}_w\left(\{f_{\boldsymbol{\theta}}(\mathbf{x}_n; \boldsymbol{\psi}_f) | \mathbf{x} \in \mathcal{D}_c, y_n = c\}; \boldsymbol{\phi}_w\right) \quad \forall c \in C^\tau$

for $m = 1, \ldots, M$ **do**

  $\boldsymbol{\pi}_m \leftarrow f_{\boldsymbol{\theta}}(\mathbf{x}_m; \boldsymbol{\psi}_f^\tau)^T \boldsymbol{\psi}_w^\tau$

  $\log p_{\boldsymbol{\theta}}(y_m | \boldsymbol{\pi}_m) \leftarrow \log \mathrm{CAT}(y_m; \boldsymbol{\pi}_m)$

end

$\hat{\mathcal{L}}(\boldsymbol{\phi}; \xi) \leftarrow \frac{1}{M} \sum_M \log p_{\boldsymbol{\theta}}(y_m | \boldsymbol{\pi}_m)$

Output: $\hat{\mathcal{L}}(\boldsymbol{\phi}; \xi)$

6.4.2 *Training the Adaptation Network Parameters*

We train $\boldsymbol{\phi}$ in the standard manner of training CNPF members, i.e. with maximum likelihood. In this setting, we can similarly express $\hat{\mathcal{L}}_{\mathrm{ML}}$ as

$$\hat{\mathcal{L}}_{\mathrm{ML}}\left(\boldsymbol{\phi}; \Xi\right) = \frac{1}{MT} \sum_{\xi_\tau \in \Xi} \sum_{m=1}^{M_\tau} \log p_{\boldsymbol{\theta}}\left(y_m^{(\tau)} | \mathbf{x}_m^{(\tau)}, \boldsymbol{\psi}_{\boldsymbol{\phi}}\left(\mathcal{D}_c^\tau\right)\right). \tag{6.4}$$

Maximum likelihood training therefore naturally uses episodic context / target splits often used in meta-learning for few-shot classification (Vinyals et al., 2016; Ravi and Larochelle, 2017). In our experiments we use the protocol defined by Triantafillou et al. (2020) and META-DATASET for this sampling procedure. Algorithm 6 details computation of the stochastic estimator for a single task.

6.5 RELATED WORK ON FEW-SHOT CLASSIFICATION

This chapter is mainly concerned with models for few-shot classification. There has been much interest in this topic, and other chapters of this thesis have not considered it directly. As such, many of the relevant models and ideas have not been covered elsewhere in this thesis. To this end, we provide a brief review of recent work in this area, focusing on placing the central ideas developed in this chapter in context of the related work.

6.5.1 *Comparison to Standard CNPF Models*

As we have seen throughout this thesis, members of the CNPF directly model the predictive distribution $p_{\boldsymbol{\theta}}(y | \mathbf{x}, \mathcal{D}_c)$ and train the parameters using maximum likelihood. Whereas previous work on the CNPF has focused on homogeneous regression and classification data sets and fairly simple models, here we study multiple heterogeneous classification data sets and use a more complex model

to handle this scenario. In particular, whereas the original CNP approach to classification required pre-specifying the number of classes in advance (Garnelo et al., 2018a), CNAPs handles varying way classification tasks, which is required for e.g. the META-DATASET benchmark. Further, CNAPs employs a parameter-sharing hierarchy that parametrises the feature extractor. This contrasts to the approach discussed in previous chapters, where parameters are all shared across tasks, and latent inputs to the decoder are used to adapt to new tasks. Finally, CNAPs employs a meta-training procedure geared towards *learning to adapt* to diverse tasks. Similarly, our work can be viewed as a deterministic limit of ML-PIP (Gordon et al., 2019) which employs a distributional treatment of the local-parameters $\boldsymbol{\psi}$.

6.5.2 *Design Space for Few-Shot Classification*

Our work specialises the central ideas from the CNPF to multi-task classification by directly modelling the predictive distribution $p(y|\mathbf{x}, \boldsymbol{\psi}(D^\tau))$. Continuing with this perspective, we can organise previous work (e.g. Finn et al., 2017; Gordon et al., 2019; Perez et al., 2018; Ravi and Larochelle, 2017; Rebuffi et al., 2017, 2018; Snell et al., 2017; Triantafillou et al., 2020; Vinyals et al., 2016; Zintgraf et al., 2019; Bauer et al., 2017) in terms of (i) the choice of the parametrisation of the classifier (and in particular the nature of the local parameters), and (ii) the function used to compute the local parameters from the training data. This space is illustrated in Figure 6.6, which is characterised by two important dimensions: the choice of model parameters to adapt, and the mechanism used to adapt the parameters.

The choice of Task-Specific Parameters

Clearly, any approach to multi-task classification must adapt, at the very least, the top-level classifier layer of the model. A number of successful models have proposed doing just this with e.g., neighbourhood-based approaches (Snell et al., 2017), variational inference (Bauer et al., 2017), or inference networks (Gordon et al., 2019). On the other end of the spectrum are models that adapt *all* the parameters of the classifier, e.g., (Finn et al., 2017; Nichol and Schulman, 2018; Yoon et al., 2018). The trade-off here is clear: as more parameters are adapted, the resulting model is more flexible, but also slow and prone to over-fitting. For this reason CNAPs modulates a small portion of the network parameters, following recent work on multi-task learning (Rebuffi et al., 2017, 2018; Perez et al., 2018).

We argue that just adapting the linear classification layer is sufficient when the task distribution is not diverse, as in the standard benchmarks used for few-shot classification (OMNIGLOT (Lake et al., 2011) and *mini*-IMAGENET (Vinyals et al., 2016)). However, when faced with a diverse set of tasks, such as that introduced

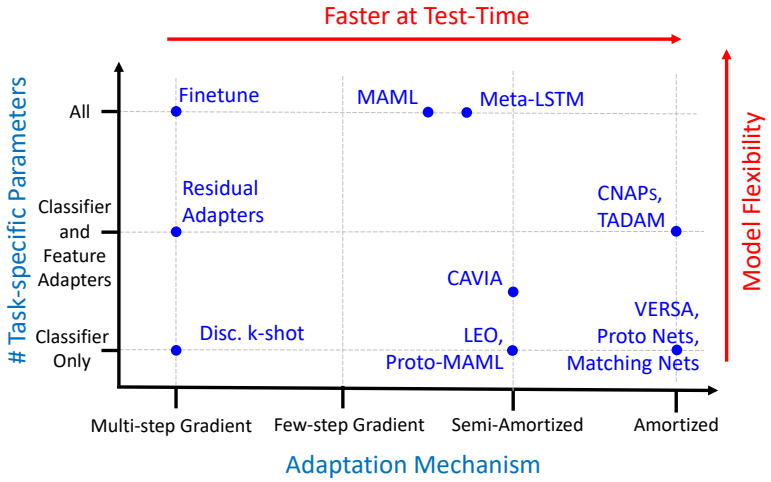


Figure 6.6: Model design space. The y-axis represents the number of task-specific parameters $|\psi^\tau|$. Increasing $|\psi^\tau|$ increases model flexibility, but also the propensity to over-fit. The x-axis represents the complexity of the mechanism used to adapt the task-specific parameters to training data $\psi(D^\tau)$. On the right are *amortized* approaches (i.e. using fixed functions). On the left is gradient-based adaptation. Mixed approaches lie between. Computational efficiency increases to the right. Flexibility increases to the left, but with it over-fitting and need for hand tuning.

recently by Triantafillou et al. (2020), it is important to adapt the feature extractor on a per-task basis as well.

The Adaptation Mechanism

Adaptation varies in the literature from performing full gradient descent learning with $\mathcal{D}_c$ (Yosinski et al., 2014) to relying on simple operations such as taking the mean of class-specific feature representations (Snell et al., 2017; Vinyals et al., 2016). Recent work has focused on reducing the number of required gradient steps by learning a global initialization (Finn et al., 2017; Nichol and Schulman, 2018) or additional parameters of the optimization procedure (Ravi and Larochelle, 2017). Gradient-based procedures have the benefit of being flexible, but are computationally demanding, and prone to over-fitting in the low-data regime. Another line of work has focused on learning neural networks to output the values of ψ, which we denote *amortization* (Gordon et al., 2019). Amortization greatly reduces the cost of adaptation and enables sharing of global parameters, but may suffer from the amortization gap (i.e. underfitting Cremer et al., 2018), particularly in the large data regime. Even more recently, some authors have proposed using semi-amortized inference (Triantafillou et al., 2020), but have done so while only adapting the classification layer parameters.

A model with design choices closely related to CNAPs is TADAM (Oreshkin et al., 2018). TADAM employs a similar set of local parameters, allowing for

adaptation of both the feature extractor and classification layer. However, it uses a far simpler adaptation network (lacking auto-regressive structure) and an expensive and ad-hoc training procedure. Moreover, TADAM was applied to simple few-shot learning benchmarks (e.g. CIFAR100 and mini-ImageNet) and sees little gain from feature extractor adaptation. In contrast, we see a large benefit from adapting the feature extractor. This may in part reflect the differences in the two models, but we observe that feature extractor adaptation has the largest impact when used to adapt to *different data sets* and that two stage training is required to see this.

6.6 EMPIRICAL EVALUATION

Our experiments our focused on two key questions: (i) Can CNAPs improve performance in multi-task few-shot learning? (ii) Does the use of an adaptation network benefit computational-efficiency and data-efficiency? We use the following modelling choices. While CNAPs can utilize any feature extractor, a ResNet18 (He et al., 2016) is used throughout to enable fair comparison with Triantafillou et al. (2020). To ensure that each task is handled independently, batch normalization (Ioffe and Szegedy, 2015) statistics are learned (and fixed) during the pre-training phase for $\boldsymbol{\theta}$. Actual batch statistics of the test data are never used during meta-training or testing. For complete experimental protocols and details, see Appendix H.

6.6.1 *Few Shot Classification with META-DATASET*

Our experiments tackle a demanding few-shot classification challenge called META-DATASET (Triantafillou et al., 2020). META-DATASET is composed of ten (eight train, two test) image classification data sets. The challenge constructs few-shot learning tasks by drawing from the following distribution. First, one of the data sets is sampled uniformly; second, the "way" and "shot" are sampled randomly according to a fixed procedure; third, the classes and context / target instances are sampled. Where a hierarchical structure exists in the data (ILSVRC or OMNIGLOT), task-sampling respects the hierarchy. In the meta-test phase, the identity of the original dataset is not revealed and the tasks must be treated independently (i.e. no information can be transferred between them). Notably, the meta-training set comprises a disjoint and dissimilar set of classes from those used for meta-test. Full details are available in Appendix H.1 and (Triantafillou et al., 2020).

Triantafillou et al. (2020) consider two stage training: an initial stage that trains a feature extractor in a standard classification setting, and a meta-training stage of all parameters in an episodic regime. For the meta-training stage, they consider

Table 6.1: Few-shot classification results on META-DATASET (Triantafillou et al., 2020) using models meta-trained on all training datasets. The feature extractor parameters $\boldsymbol{\theta}$ are trained on ILSVRC, using only observations from the training set according to the META-DATASET split. All figures are percentages and the $\pm$ sign indicates the 95% confidence interval over tasks. Bold text indicates the scores within the confidence interval of the highest score (among methods published concurrently with CNAPs). Tasks from data sets below the dashed line were not used for meta-training. Competing methods' results are as reported by Triantafillou et al. (2020). The final two columns represent the results of Simple CNAPs (Bateni et al., 2020) and Universal Represenation Transoformers (URT Liu et al., 2020), two methods that were published after and improve upon CNAPs, representing state-of-art performance in META-DATASET at the time of writing this thesis.

Dataset	Finetune	MatchingNet	ProtoNet	fo-MAML	Proto-MAML	CNAPs (no ψ_f)	CNAPs (no $\mathbf{z}_{AR}$)	CNAPs	Simple CNAPs	URT
ILSVRC	43.1 ± 1.1	36.1 ± 1.0	44.5 ± 1.1	32.4 ± 1.0	47.9 ± 1.1	43.8 ± 1.0	$\mathbf{51.3 \pm 1.0}$	$\mathbf{52.3 \pm 1.0}$	58.6 ± 1.1	55.7 ± 1.0
Omniglot	71.1 ± 1.4	78.3 ± 1.0	79.6 ± 1.1	71.9 ± 1.2	82.9 ± 0.9	60.1 ± 1.3	$\mathbf{88.0 \pm 0.7}$	$\mathbf{88.4 \pm 0.7}$	91.7 ± 0.6	94.4 ± 0.4
Aircraft	72.0 ± 1.1	69.2 ± 1.0	71.1 ± 0.9	52.8 ± 0.9	74.2 ± 0.8	53.0 ± 0.9	76.8 ± 0.8	$\mathbf{80.5 \pm 0.6}$	82.4 ± 0.7	85.8 ± 0.6
Birds	59.8 ± 1.2	56.4 ± 1.0	67.0 ± 1.0	47.2 ± 1.1	70.0 ± 1.0	55.7 ± 1.0	$\mathbf{71.4 \pm 0.9}$	$\mathbf{72.2 \pm 0.9}$	74.9 ± 0.8	76.3 ± 0.8
Textures	$\mathbf{69.1 \pm 0.9}$	61.8 ± 0.7	65.2 ± 0.8	56.7 ± 0.7	67.9 ± 0.8	60.5 ± 0.8	62.5 ± 0.7	58.3 ± 0.7	67.8 ± 0.8	71.8 ± 0.7
Quick Draw	47.0 ± 1.2	60.8 ± 1.0	64.9 ± 0.9	50.5 ± 1.2	66.6 ± 0.9	58.1 ± 1.0	$\mathbf{71.9 \pm 0.8}$	$\mathbf{72.5 \pm 0.8}$	77.7 ± 0.7	82.5 ± 0.6
Fungi	38.2 ± 1.0	33.7 ± 1.0	40.3 ± 1.1	21.0 ± 1.0	42.0 ± 1.1	28.6 ± 0.9	$\mathbf{46.0 \pm 1.1}$	$\mathbf{47.4 \pm 1.0}$	46.9 ± 1.0	63.5 ± 1.0
VGG Flower	85.3 ± 0.7	81.9 ± 0.7	86.9 ± 0.7	70.9 ± 1.0	$\mathbf{88.5 \pm 0.7}$	75.3 ± 0.7	$\mathbf{89.2 \pm 0.5}$	86.0 ± 0.5	90.7 ± 0.5	88.2 ± 0.6
Traffic Signs	$\mathbf{66.7 \pm 1.2}$	55.6 ± 1.1	46.5 ± 1.0	34.2 ± 1.3	52.3 ± 1.1	55.0 ± 0.9	60.1 ± 0.9	60.2 ± 0.9	73.5 ± 0.7	69.4 ± 0.8
MSCOCO	35.2 ± 1.1	28.8 ± 1.0	39.9 ± 1.1	24.1 ± 1.1	$\mathbf{41.3 \pm 1.0}$	$\mathbf{41.2 \pm 1.0}$	$\mathbf{42.0 \pm 1.0}$	$\mathbf{42.6 \pm 1.1}$	46.2 ± 1.1	52.2 ± 1.1
MNIST						76.0 ± 0.8	88.6 ± 0.5	$\mathbf{92.7 \pm 0.4}$	93.9 ± 0.4	
CIFAR10						$\mathbf{61.5 \pm 0.7}$	$\mathbf{60.0 \pm 0.8}$	$\mathbf{61.5 \pm 0.7}$	74.3 ± 0.7	
CIFAR100						44.8 ± 1.0	$\mathbf{48.1 \pm 1.0}$	$\mathbf{50.1 \pm 1.0}$	60.5 ± 1.0	

two settings: meta-training only on the META-DATASET version of ILSVRC, and on all meta-training data. We focus on the latter as CNAPs rely on training data from a variety of training tasks to learn to adapt. We pre-train $\boldsymbol{\theta}$ on the meta-training set of the META-DATASET version of ILSVRC, and meta-train ϕ in an episodic fashion using all meta-training data. We compare CNAPs to models considered by Triantafillou et al. (2020), including their proposed method (Proto-MAML) in Table 6.1. We meta-test CNAPs on three additional held-out data sets: MNIST, CIFAR10, and CIFAR100. As an ablation study, we compare a version of CNAPs that does not make use of the auto-regressive component $\mathbf{z}_{AR}$, and a version that uses no feature extractor adaptation. In our analysis of Table 6.1, we distinguish between two types of generalization: (i) unseen tasks (classes) in meta-training data sets, and (ii) unseen data sets.

Unseen Tasks

CNAPs achieve significant improvements over existing methods on seven of the eight data sets. The exception is the TEXTURES dataset, which has only seven test classes and accuracy is highly sensitive to the train / validation / test class split. The ablation study demonstrates that removing $\mathbf{z}_{\text{AR}}$ from the feature extractor adaptation degrades accuracy in most cases, and that removing all feature extractor adaptation results in drastic reductions in accuracy.

Unseen data sets

CNAPs-models outperform all competitive models with the exception of FINETUNE on the TRAFFIC SIGNS dataset. Removing $\mathbf{z}_{\text{AR}}$ from the feature extractor decreases accuracy and removing the feature extractor adaptation entirely significantly impairs performance. The degradation is particularly pronounced when the held out dataset differs substantially from the dataset used to pretrain $\boldsymbol{\theta}$, e.g. for MNIST.

Note that the superior results when using the auto-regressive component can not be attributed to increased network capacity alone. In Appendix H.1.1 we demonstrate that CNAPs yields superior classification accuracy when compared to parallel residual adapters (Rebuffi et al., 2018) even though CNAPs requires significantly less network capacity in order to adapt the feature extractor to a given task.

6.6.2 *Joint Training of Global Parameters and Adaptation Networks*

We compare different training procedures to verify the importance of our proposed two-stage procedure for training the model parameters. To study this, we compare

Table 6.2: Few-shot classification results on META-DATASET (Triantafillou et al., 2020) comparing joint training for $\boldsymbol{\theta}$ and ϕ (columns 2 and 3) to two-stage training (column 4). All figures are percentages and the $\pm$ sign indicates the 95% confidence interval. Bold text indicates the highest scores that overlap in their confidence intervals.

Dataset	Joint Training (warmstart BN)	Joint Training (BN train mode)	Two-Stage Training (BN test mode)
ILSVRC	17.3±0.7	41.6±1.0	49.5±1.0
Omniglot	74.9±1.0	80.8±0.9	89.7±0.5
Aircraft	51.4±0.8	70.5±0.7	87.2±0.5
Birds	44.1±1.0	48.3±1.0	76.7±0.9
Textures	49.1±0.7	73.5±0.6	83.0±0.6
Quick Draw	46.6±1.0	71.5±0.8	72.3±0.8
Fungi	20.4±0.9	43.1±1.1	50.5±1.1
VGG Flower	66.6±0.8	71.0±0.7	92.5±0.4
Traffic Signs	21.2±0.8	40.4±1.1	48.4±1.1
MSCOCO	18.8±0.7	37.1±1.0	39.7±0.9

versions of CNAPs where joint training of $\boldsymbol{\theta}$ and ϕ is performed, as is standard for members of the CNPF. Our experiments with joint training of $\boldsymbol{\theta}$ and ϕ demonstrate that the two-stage training procedure proposed in Section 6.4 is crucially important. In particular, we find that joint training diverged in almost all cases we attempted. We were only able to successfully train jointly in two circumstances: (i) Using batch normalization in "train" mode for both context *and* target sets. We stress that this implies computing the batch statistics at test time, and using those to normalize the batches. This is in contrast to the methodology we propose previously in the chapter: only using batch normalization in "eval" mode, which enforces that no information is transferred across tasks or data sets. (ii) "Warm-start" the training procedure with batch normalization in "train" mode, and after a number of epochs (we use 50 for the results shown below), switch to proper usage of batch normalization. All other training procedures we attempted diverged. Table 6.2 demonstrates that these procedures result in consistently and significantly poorer performance than our proposed two-stage procedure, validating its importance.

6.6.3 *FiLM Parameter Learning Performance: Speed-Accuracy Trade-off*

CNAPs generate FiLM layer parameters for each task τ at test time using the adaptation network $\psi_f(\cdot)$. It is also possible to learn the FiLM parameters via gradient descent (Rebuffi et al., 2017, 2018). We compare CNAPs to this approach. Figure 6.7 shows plots of 5-way classification accuracy versus time for four held out data sets as the number of shots was varied. For gradient descent, we used

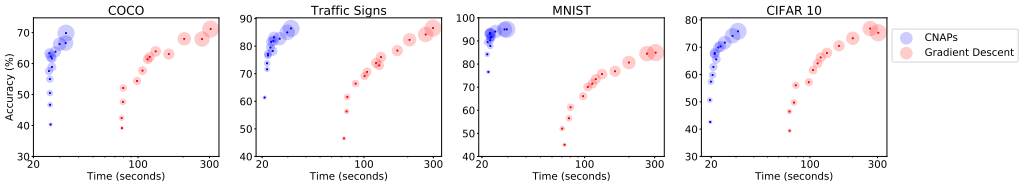


Figure 6.7: Comparing CNAPs to gradient based feature extractor adaptation: accuracy on 5-way classification tasks from withheld data sets as a function of processing time. Dot size reflects shot number (1 to 25 shots).

a fixed learning rate of 0.001 and took 25 steps for each point. The overall time required to produce the plot was 1274 and 7214 seconds for CNAPs and gradient approaches, respectively, on a NVIDIA Tesla P100-PCIE-16GB GPU. CNAPs is at least 5 times faster at test time than gradient-based optimization requiring only a single forward pass through the network while gradient based approaches require multiple forward and backward passes. Further, the accuracy achieved with adaptation networks is significantly higher for fewer shots as it protects against over-fitting. For large numbers of shots, gradient descent catches up, albeit slowly.

6.7 CONCLUSION AND DISCUSSION

This chapter has introduced CNAPs, specialising the CNPF to the few-shot classification setting. CNAPs is an automatic, fast and flexible modelling approach for multi-task classification, which achieves state-of-the-art performance on the challenging META-DATASET benchmark.

Since its introduction, several papers have built directly on the work of CNAPs. For example, Bateni et al. (2020) propose "Simple CNAPs", which replaces the CNAPs linear classifier with a prototypical networks classification scheme (Snell et al., 2017), achieving significant performance improvements over the model proposed in this chapter. In a separate work, Nguyen et al. (2020) propose a new measure—LEEP—to quantify the transferability of representations learned by few-shot classifiers. The authors go on to demonstrate that LEEP can predict the performance of CNAPs, demonstrating a correlation between the LEEP score and test accuracies for held-out tasks. Finally, in a very recent paper, Liu et al. (2020) propose an attention-based architecture for few-shot classification. They pretrain separate feature-extractors for each of the eight training data sets in META-DATASET, and use multi-head attention over the resulting features to produce a single representation for unseen tasks. This leads to state-of-the-art performance for the "unseen tasks" (i.e. in-domain tasks), but still falls short of the CNAPs-based architecture of Bateni et al. (2020) on the "unseen data sets".

7

CONCLUSIONS AND DISCUSSION

We conclude the thesis with a summary of the central contributions, a discussion on the Neural Process Family and their role in meta-learning, and a brief discussion on future research directions in the NPF.

7.1 SUMMARY OF CONTRIBUTIONS

The central focus of this thesis has been the Neural Process Family; a class of models for probabilistic meta-learning. In particular, the thesis is aimed at constructing a framework with which to reason about the NPF, and which can be leveraged to construct new members. Chapter 2 introduces the perspective of meta-learning as parametrising and learning maps from a space of data sets to a space of predictive stochastic processes. With this view in place, the NPF can be thought of as parametrising such maps with deep neural networks while respecting their most important properties (e.g. permutation invariance and the KET conditions). Chapter 2 then provides a general overview of the NPF and its two important sub-families: the Conditional NPF and the Latent-Variable NPF. We conclude Chapter 2 with several novel results, e.g. characterising the expressivity of the CNPF, and the limiting behaviours of NPF training procedures.

Chapter 3 introduces the notion of translation equivariance in the NPF, and motivates its importance. An important step is then taken towards translation equivariant members of the NPF by introducing the ConvDeepSets framework. ConvDeepSets extend the theory of representing and learning functions on sets by providing a universal representation statement for parametrisations that incorporate this translation equivariance. Chapters 4 and 5 then demonstrate how ConvDeepSets can be leveraged to construct translation equivariant members of the NPF, with both conditional and latent variable variants provided. Extensive experimentation is carried out to validate the usefulness of the proposed models.

Finally, in Chapter 6, we consider the development of NPF models for few-shot image classification. We argue that the central challenge in this setting is to balance model capacity and robustness of adaptation to new tasks. To navigate this trade-off, we propose CNAPs, a novel member of the NPF geared towards heterogeneous few-shot image classification tasks. Experiments with META-DATASET (Triantafillou et al., 2020) demonstrate that CNAPs achieves state-of-the-art performance while being faster to deploy on unseen tasks.

7.2 SHOULD YOU USE THE NPF FOR YOUR MACHINE LEARNING APPLICATION?

An important question is whether and when we should use members of the NPF in practice? To address this, we compare the NPF to several of its most natural alternatives.

7.2.1 *Neural Processes or Gradient-Based Meta-Learning?*

Much interest has been devoted to meta-learning, and as a result several approaches have been proposed with modelling capacity similar to that of the NPF. A particularly prominent approach to meta-learning employs gradient-based methods (Finn, 2018). Such models provide similar guarantees regarding the universality of the representation and learning procedures to those presented in Section 2.7 (Finn and Levine, 2018). Yet for both approaches, these guarantees are valid only in infinite data/capacity regimes. How then, should we make practical decisions regarding which approach to invoke? There are several dimensions of a modelling problem which we may consider to help us reason about this difficult question. Below, I discuss what I see as the important factors in answering this question, and my advice to practitioners on the matter.

The Need for Probabilistic Predictions

Augmenting predictions with a measure of uncertainty may or may not be important to the application under consideration. For example, if accuracy is the only important metric for a particular few-shot classifier, there is little evidence that improving the model's ability to provide meaningful uncertainty leads to improved performance. Conversely, if a meta-learning model trained on healthcare records is providing outputs used to augment doctors' decision making processes, it is crucially important that the system provide calibrated uncertainty estimates along with its predictions.

Despite significant research effort, gradient-based meta-learners producing calibrated uncertainty estimates remain elusive. Conversely, when appropriately designed, members of the NPF are able to closely recover the uncertainty associated with the true underlying process. As such, in settings where uncertainty plays an important role, invoking the NPF may be more appropriate than a gradient-based approach.

Flexibility, Over-fitting, and Compute

An important aspect of an application is how much data and compute a user expects to have available at meta-test time, i.e. when encountering unseen tasks.

Considering the capacity of the neural networks typically employed in the NPF encoders, gradient-based learning may be more flexible, and thus may be able to model a larger class of functions, often leading to better test-time performance when large context sets are available. However, this flexibility comes at a cost: when context sets are small, gradient-based learners will be more prone to over-fitting, as demonstrated in Section 6.6. On the other hand, members of the NPF *share* the parameters of the adaptation mechanism across tasks, leading to increased data-efficiency and improved performance when context sets are small.

Similarly, we have seen that invoking amortisation for the adaptation mechanism leads to models that require less compute to adapt, as no gradient-based computations are required. And yet gradient-based learning of neural networks is the de-facto standard procedure for training deep learning models. There are many best practices and "tricks" that enable us to improve performance of models trained this way, and much of our current software is tailored towards this solution-class, often leading to easier-to-deploy solutions.

Thus, if a user expects to have little data/compute to adapt at meta-test time, it may be reasonable to consider amortisation as an adaptation mechanism. Conversely, if a user expects large context sets to be available at test time, or the application allows for the use of more significant compute, it may be beneficial to employ gradient-based meta-learners.

Bridging the Gap: Semi-Amortised Inference

Finally, it should be mentioned that these two approaches are not mutually exclusive. An interesting "compromise" is *semi-amortised* inference (e.g. Kim et al., 2018), which employs amortisation networks to output initial values, and then tune these values with gradient-based procedures. For many applications, such a "middle-ground" may in fact be the most appropriate approach.

7.2.2 *Neural or Gaussian Processes?*

Gaussian processes provide a principled approach to modelling data and admit exact methods for posterior inference. Moreover, they allow us to encode our prior beliefs via the design of the mean and kernel functions. Under what circumstances then, should we use members of the NPF instead of GPs? Below, I provide several instances where we may prefer a Neural process to a Gaussian process.

Sharing Information Across Tasks

An important feature of meta-learning is that it naturally enables sharing across multiple related tasks, which can lead to significant performance gains. In contrast,

doing so with GPs is not as straightforward. A naive approach would be to jointly learn the hyper-parameters of the kernel across multiple tasks, but even this can be difficult to do in a principled way. Thus, if the data at hand can be viewed as a collection of related tasks (as in the experiments in Section 5.7.3 or with patient health-care data), the NPF may be more appropriate.

Non-Gaussian Predictive Distributions

In many applications it may be unreasonable to assume that the predictive distribution has a Gaussian form. This may be due to multi-modality (as in the image experiments in Section 5.7.2) or heavy tailedness and asymmetry (as in the environmental experiments of Section 5.7.3), or even that the task involves varying-way classification, as in Chapter 6. In such cases, employing GPs may be futile, as the resulting predictive distributions are limited to Gaussian forms. Conversely, with little modelling overhead, we can use alternative parametric forms for the predictive distributions of CNPF members, e.g. Laplace for heavy-tailed distributions, finite mixture-of-Gaussians for multi-modal distributions, or categorical for classification tasks. Employing these requires nothing more than defining the likelihood function $p_{\boldsymbol{\theta}}(y|\mathbf{x}, \mathbf{r})$, and modifying $d_{\boldsymbol{\theta}}$ to output the appropriate parameters. Moreover, we can invoke the LNPF to parametrise arbitrarily flexible predictive distributions.

Complicated or High-Dimensional Input Spaces

Finally, we are often interested in applications involving high-dimensional or complex input spaces, for which it can be difficult to design appropriate kernel functions, and costly to compute and store the necessary kernel (Gram) matrices. Here again, the canonical example is modelling of images, where designing appropriate kernel functions is notoriously difficult (see for example Wilk et al., 2017). In contrast, the NPF can be viewed as learning an implicit prior over function space, which is then used for inference when a new context set is observed (Garnelo et al., 2018a,b). This allows us to leverage the power of neural networks for representation learning, which is has proven to be extremely useful for high-dimensional or complicated input spaces.

7.2.3 *Which Member of the NPF?*

Having decided to invoke the NPF for a particular modelling application, which member should the user consider. The two main decisions to be made are whether to consider the CNPF or LNPF, and which inductive biases should be encoded into the architecture.

Conditional or Latent-Variable?

Members of the LNPF have the obvious benefit that they parametrise a richer predictive space of SPs, allowing for dependencies across target points. However, this comes at the significant cost of an intractable likelihood $p_{\boldsymbol{\theta}}\left(\mathbf{y}_T|\mathbf{X}_T, \mathcal{D}_c\right)$, complicating both training and evaluation of the models.

My opinion is that, other than for particular exceptions, working with the CNPF is far easier and more reliable, and should thus be preferred. As mentioned in Chapter 5, the exceptional cases are when (i) sampling from the predictive is a core requirement of the application, (ii) the predictive performance is dominated by dependencies in the distribution, or (iii) designing an appropriate likelihood function is overly complex for a human expert. In any of these cases, invoking a member of the LNPF may be more appropriate than its CNPF counter-part. In all other cases, my recommendation would be to first consider the deployment of the CNPF, which are more robust and less costly to train.

Appropriate Inductive Biases

Choosing the appropriate inductive biases or architecture for the NPF member is highly application-specific. Generally speaking, the best performing architectures in the current literature are most likely attentive or convolutional architectures. One consideration is that the expressive power of convolutional architectures is generally smaller than that of attentive models, since they are (by definition) restricted to translation equivariant mappings, which are a small subset of all possible mappings. In fact, in many cases the assumption of a stationary underlying SP can exclude important desirable characteristics (see e.g. Mishra et al., 2020, for a discussion on the matter). The other is the dimensionality of the input space $\mathcal{X}$: for $\mathcal{X} = \mathbb{R}^d$ with $d > 3$, invoking convolutions will generally be extremely memory intensive, and most likely infeasible. Thus, in such a case, I would recommend employing an attentive member of the NPF before a convolutional one.

Conversely, as we have seen, convolutional architectures enable generalisation in time/space in a way that other architectures cannot. This property may be crucial in several important applications of the NPF. Further, as we have seen, in many cases (even when stationarity is not necessarily "perfectly" appropriate), convolutional NPF members may outperform their non-convolutional counterparts, which may be due to other properties of the inductive bias (e.g. parameter-sharing). Therefore, if stationarity seems even moderately appropriate as an inductive bias on the input space $\mathcal{X}$ (and $\mathcal{X}$ is no more than 3-dimensional), my recommendation would be to first try a convolutional member of the NPF.

7.3 FUTURE WORK

Finally, we conclude with a brief discussion on interesting avenues for future work. Research towards probabilistic meta-learning with the NPF is quite nascent, such that there are many interesting directions to pursue. Below, I focus on several directions which are of particular impact in my own opinion.

7.3.1 *Training Procedures for the LNPF*

In Sections 2.6.2 and 2.7.3 we discussed two procedures for training members of the LNPF. However, both procedures involve biased estimators of the log-marginal likelihood, complicating both training and evaluation of members of the LNPF. Better procedures are required to make the LNPF a more viable class of models for real-world applications.

One line of research would be to consider importance-weighted objectives for training (Burda et al., 2015). In preliminary experiments not discussed in this thesis, we observed that invoking such objectives lead to improved performance for some models, but invoked peculiar artefacts in others. We hypothesise that such behaviours result, at least to a certain degree, from the bias in the training objective. Recently, Luo et al. (2020) proposed an unbiased, importance-weighted objective estimator of the log-marginal likelihood in latent variable models. Similar ideas could be used to derive an unbiased objective for the LNPF, alleviating many of the difficulties associated with training and evaluation of the models.

7.3.2 *Neural Process Flows*

The central drawback of the CNPF is the limitation to parametric and factorised predictive distributions. An alternative to the LNPF could be to invoke normalising flows for the predictive distribution. Normalising flows allow us to express arbitrarily complex distributions that can be evaluated analytically using a change of variables (Papamakarios et al., 2019a).

A straightforward approach would simply be to apply a point-wise flow the density at each target point output by a member of the CNPF. This would solve the problem of the parametric form, but would not allow for dependencies in the predictive distribution. A far more interesting approach would be to develop a flow that operated directly on SPs. Such objects have been considered in the recent literature (e.g. Deng et al., 2020), but a solution not involving a latent variable is yet to be proposed. It is not clear whether such an object is well-defined, but if it could be constructed, it could allow the construction of NPF members that achieve the central advantages of both the CNPF and LNPF.

8

BIBLIOGRAPHY

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard et al. (2016). 'Tensorflow: A System for Large-Scale Machine Learning'. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (cited on page 21).

Nachman Aronszajn (1950). 'Theory of Reproducing Kernels'. *Transactions of the American mathematical society* (cited on pages 49, 117).

Peter Auer (2002). 'Using Confidence Bounds for Exploitation-Exploration Trade-offs'. *Journal of Machine Learning Research* (cited on page 77).

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). 'Layer Normalization'. *arXiv preprint arXiv:1607.06450* (cited on pages 142, 152).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2015). 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (cited on page 25).

Gianpaolo Balsamo, Clement Albergel, Anton Beljaars, Souhail Boussetta, Eric Brun, Hannah Cloke, Dick Dee, Emanuel Dutra, Joaquin Muñoz-Sabater, Florian Pappenberger et al. (2015). 'ERA-Interim/Land: a Global Land Surface Reanalysis Data Set'. *Hydrology and Earth System Sciences* (cited on pages 75, 163).

Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal (2020). 'Improved Few-Shot Visual Classification'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (cited on pages 91, 94).

David W Bates, Suchi Saria, Lucila Ohno-Machado, Anand Shah, and Gabriel Escobar (2014). 'Big Data in Health Care: Using Analytics to Identify and Manage High-Risk and High-Cost Patients'. *Health Affairs* (cited on pages 43, 44).

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartłomiej Świątkowski, Bernhard Schölkopf, and Richard E Turner (2017). 'Discriminative k-Shot Learning Using Probabilistic Models'. *arXiv preprint arXiv:1706.00326* (cited on page 88).

Atilim Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind (2017). 'Automatic Differentiation in Machine Learning: a Survey'. *The Journal of Machine Learning Research* (cited on page 21).

Benjamin Bloem-Reddy and Yee Whye Teh (2020). 'Probabilistic Symmetries and Invariant Neural Networks'. *Journal of Machine Learning Research* (cited on pages 16, 34).

Kyle Boone (2019). 'Avocado: Photometric Classification of Astronomical Transients with Gaussian Process Augmentation'. *arXiv preprint arXiv:1907.04690* (cited on page 43).

Léon Bottou (2010). 'Large-scale machine learning with stochastic gradient descent'. In: *Proceedings of COMPSTAT'2010* (cited on page 21).

Eric Brochu, Vlad M Cora, and Nando De Freitas (2010). 'A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning'. *arXiv preprint arXiv:1012.2599* (cited on page 77).

John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard Turner (2020). 'TaskNorm: Rethinking Batch Normalization for Meta-Learning'. In: *Proceedings of the 37th International Conference on Machine Learning*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. (2020). 'Language models are few-shot learners'. *arXiv preprint arXiv:2005.14165* (cited on page 38).

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov (2015). 'Importance Weighted Autoencoders'. *arXiv preprint arXiv:1509.00519* (cited on pages 36, 100).

Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi (2019). 'Probabilistic Neural Architecture Search'. *arXiv preprint arXiv:1902.05116* (cited on page 37).

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien (2009). 'Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]'. *IEEE Transactions on Neural Networks* (cited on page 14).

Francois Chollet (2017). 'Xception: Deep Learning with Depthwise Separable Convolutions'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cited on pages 140, 150, 152, 165).

Jeff Clune (2019). 'AI-GAs: AI-Generating Algorithms, an Alternate Paradigm for Producing General Artificial Intelligence'. *arXiv preprint arXiv:1905.10985* (cited on page 9).

Taco Cohen and Max Welling (2016). 'Group Equivariant Convolutional Networks'. In: *Proceedings of The 33rd International Conference on Machine Learning* (cited on pages 45, 53).

Chris Cremer, Xuechen Li, and David Duvenaud (2018). 'Inference Suboptimality in Variational Autoencoders'. In: *Proceedings of the 35th International Conference on Machine Learning* (cited on page 89).

Noel Cressie (1990). 'The Origins of Kriging'. *Mathematical geology* (cited on page 45).

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel (1995). 'The Helmholtz Machine'. *Neural computation* (cited on page 29).

JP Delhomme (1978). 'Kriging in the Hydrosciences'. *Advances in water resources* (cited on pages 43–45).

Ruizhi Deng, Bo Chang, Marcus A Brubaker, Greg Mori, and Andreas Lehrmann (2020). 'Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows'. *arXiv preprint arXiv:2002.10516* (cited on page 100).

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter (2015). 'Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves'. In: *Twenty-fourth international joint conference on artificial intelligence* (cited on page 37).

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel (2016). 'RL^2: Fast reinforcement learning via slow reinforcement learning'. *arXiv preprint arXiv:1611.02779* (cited on page 37).

James Dugundji et al. (1951). 'An Extension of Tietze's Theorem.' *Pacific Journal of Mathematics* (cited on page 138).

Harrison Edwards and Amos Storkey (2017). 'Towards a Neural Statistician'. In: *In International Conference on Learning Representations (ICLR)* (cited on pages 16, 38).

Aaron M Ellison (1987). 'Effect of Seed Dimorphism on the Density-Dependent Dynamics of Experimental Populations of Atriplex Triangularis (Chenopodiaceae)'. *American Journal of Botany* (cited on page 75).

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter et al. (2019). 'Neural architecture search: A survey.' *J. Mach. Learn. Res.* (cited on page 37).

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor et al. (2018). 'Neural scene representation and rendering'. *Science* (cited on pages 24, 31).

Li Fei-Fei, Rob Fergus, and Pietro Perona (2006). 'One-Shot Learning of Object Categories'. *IEEE transactions on pattern analysis and machine intelligence* (cited on page 79).

Chelsea Finn (2018). 'Learning to Learn with Gradients'. PhD thesis. UC Berkeley (cited on page 96).

Chelsea Finn, Pieter Abbeel, and Sergey Levine (2017). 'Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks'. In: *Proceedings of the 34th International Conference on Machine Learning* (cited on pages 14, 80, 82, 88, 89).

Chelsea Finn and Sergey Levine (2018). 'Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm'. In: *International Conference on Learning Representations* (cited on pages 37, 96).

Andrew Y. K. Foong, Wessel P. Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard E. Turner (2020). 'Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes'. In: *Advances in Neural Information Processing Systems 33* (cited on pages 2, 3, 17, 65).

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami (2018a). 'Conditional Neural Processes'. In: *Proceedings of the 35th International Conference on Machine Learning* (cited on pages v, 17, 24, 40, 54, 56, 88, 98, 141).

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh (2018b). 'Neural Processes'. *arXiv preprint arXiv:1807.01622* (cited on pages v, 17, 28–32, 65, 71, 98, 142).

Seymour Geisser (1983). *On the Prediction of Observables: a Selective Update*. Technical report. University of Minnesota (cited on page 79).

— (2017). *Predictive Inference*. Routledge (cited on page 79).

Samuel Gershman and Noah Goodman (2014). 'Amortized Inference in Probabilistic Reasoning'. In: *Proceedings of the annual meeting of the cognitive science society* (cited on pages 28, 29).

Zoubin Ghahramani (2015). 'Probabilistic Machine Learning and Artificial Intelligence'. *Nature* (cited on page 10).

Daniel T Gillespie (1977). 'Exact Stochastic Simulation of Coupled Chemical Reactions'. *The journal of physical chemistry* (cited on page 143).

Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press (cited on pages 1, 8).

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner (2019). 'Meta-Learning Probabilistic Inference for Prediction'. In: *International Conference on Learning Representations* (cited on pages 17, 80, 84, 88, 89).

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner (2018a). 'Consolidating the Meta-Learning Zoo: A Unifying Perspective as Posterior Predictive Inference'. In: *MetaLearning Workshop, NeurIPS 2018*.

— (2018b). 'Versa: Versatile and Efficient Few-Shot Learning'. In: *Bayesian Deep Learning Workshop, NeurIPS 2018*.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner (2020a). 'Convolutional Conditional Neural Processes'. In: *International Conference on Learning Representations* (cited on pages 2, 3, 17, 25, 39, 46, 51).

Jonathan Gordon and José Miguel Hernández-Lobato (2017). 'Bayesian Semi-Supervised Learning with Deep Generative Models'. In: *ICML Workshop on Principled Approaches to Deep Learning*.

— (2019). 'Combining Deep Generative and Discriminative Models for Bayesian Semi-Supervised Learning'. *Pattern Recognition*.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt (2020b). 'Permutation Equivariant Models for Compositional Generalization in Language'. In: *International Conference on Learning Representations*.

Marton Havasi, Jasper Snoek, Dustin Tran, Jonathan Gordon, and José Miguel Hernández-Lobato (2020). 'Refining the Variational Posterior Through Iterative Optimization'. In: *Bayesian Deep Learning Workshop, NeurIPS 2020*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). 'Deep Residual Learning for Image Recognition'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (cited on pages 8, 45, 82, 83, 90, 150, 152, 165, 169, 172).

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal (1995). 'The" Wake-Sleep" Algorithm for Unsupervised Neural Networks'. *Science* (cited on page 29).

Sepp Hochreiter, A Steven Younger, and Peter R Conwell (2001). 'Learning to learn using gradient descent'. In: *International Conference on Artificial Neural Networks*. Springer (cited on page 37).

Peter Holderrieth, Michael Hutchinson, and Yee Whye Teh (2020). 'Equivariant Conditional Neural Processes'. *arXiv preprint arXiv:2011.12916* (cited on page 50).

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey (2020). 'Meta-learning in Neural Networks: A Survey'. *arXiv preprint arXiv:2004.05439* (cited on pages 9, 38).

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature (cited on page 37).

Sergey Ioffe and Christian Szegedy (2015). 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. In: *Proceedings of the 32nd International Conference on Machine Learning* (cited on page 90).

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). 'An Introduction to Variational Methods for Graphical Models'. *Machine learning* (cited on page 28).

Makoto Kawano, Wataru Kumagai, Akiyoshi Sannai, Yusuke Iwasawa, and Yutaka Matsuo (2021). 'Group Equivariant Conditional Neural Processes'. In: *International Conference on Learning Representations* (cited on page 50).

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh (2019). 'Attentive Neural Processes'. In: *International Conference on Learning Representations* (cited on pages 17, 24–26, 30, 31, 40, 54, 56, 67, 71, 141, 149, 152).

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush (2018). 'Semi-Amortized Variational Autoencoders'. In: (cited on page 97).

Diederik P Kingma and Jimmy Ba (2015). 'Adam: A Method for Stochastic Optimization'. In: *In International Conference on Learning Representations (ICLR)* (cited on pages 9, 21, 28, 149, 165, 170).

Diederik P. Kingma and Max Welling (2014). 'Auto-Encoding Variational Bayes'. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (cited on pages 28, 29).

Hans Knutsson and C-F Westin (1993). 'Normalized and Differential Convolution'. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE (cited on page 52).

Imre Risi Kondor (2008). *Group theoretical methods in machine learning*. Columbia University (cited on pages 45, 46).

Risi Kondor and Shubhendu Trivedi (2018). 'On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups'. In: *Proceedings of the 35th International Conference on Machine Learning* (cited on pages 45, 53, 58, 68).

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton (2012). 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems 25* (cited on pages 14, 45, 86, 169).

Solomon Kullback (1959). *Information Theory and Statistics*. Courier Corporation (cited on page 32).

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum (2011). 'One Shot Learning of Simple Visual Concepts'. In: *Proceedings of the annual meeting of the cognitive science society* (cited on page 88).

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). 'Human-Level Concept Learning Through Probabilistic Program Induction'. *Science* (cited on pages 79, 80).

Hugo Larochelle and Iain Murray (2011). 'The Neural Autoregressive Distribution Estimator'. In: (cited on page 23).

Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh (2018). 'Empirical Evaluation of Neural Process Objectives'. In: *NeurIPS workshop on Bayesian Deep Learning* (cited on pages 71, 72, 145, 150, 151).

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). 'Deep Learning'. *nature* (cited on pages 1, 8).

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). 'Backpropagation Applied to Handwritten Zip Code Recognition'. *Neural computation* (cited on pages 14, 45, 46, 75).

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner et al. (1998). 'Gradient-Based Learning Applied to Document Recognition'. *Proceedings of the IEEE* (cited on pages 45, 60).

Egbert R Leigh (1968). 'The Ecological Role of Volterra's Equations'. *Some mathematical problems in biology* (cited on pages 59, 143).

Georg Lindgren (2012). *Stationary Stochastic Processes: Theory and Applications*. CRC Press (cited on pages 11, 39, 44).

Dong C Liu and Jorge Nocedal (1989). 'On the Limited Memory BFGS Method for Large Scale Optimization'. *Mathematical programming* (cited on pages 10, 164).

Lu Liu, William Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle (2020). 'A Universal Representation Transformer Layer for Few-Shot Image Classification'. *arXiv preprint arXiv:2006.11702* (cited on pages 91, 94).

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2018). 'Large-Scale Celebfaces Attributes (Celeba) Dataset'. *Retrieved August* (cited on page 60).

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling (2019). 'The Functional Neural Process' (cited on pages 17, 25).

Yucen Luo, Alex Beatson, Mohammad Norouzi, Jun Zhu, David Duvenaud, Ryan P. Adams, and Ricky T. Q. Chen (2020). 'SUMO: Unbiased Estimation of Log Marginal Probability for Latent Variable Models'. In: *International Conference on Learning Representations* (cited on page 100).

David JC MacKay (2003). *Information theory, inference and learning algorithms*. Cambridge university press (cited on page 32).

Alexander G de G Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani (2016). 'On Sparse Variational Methods and the Kullback-Leibler Divergence Between Stochastic Processes'. In: *Artificial Intelligence and Statistics* (cited on page 71).

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein (2018). 'Meta-learning update rules for unsupervised representation learning'. *arXiv preprint arXiv:1804.00222* (cited on page 37).

Swapnil Mishra, Seth Flaxman, and Samir Bhatt (2020). 'πVAE: Encoding stochastic process priors with variational autoencoders'. *arXiv preprint arXiv:2002.06873* (cited on page 99).

J.R. Munkres (1974). *Topology; a First Course*. Prentice-Hall (cited on pages 33, 46, 68, 121, 123, 125, 127, 137).

Eric Nalisnick, Jonathan Gordon, and José Miguel Hernández-Lobato (2020). 'Predictive Complexity Priors'. *arXiv preprint arXiv:2006.10801*.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng (2011). 'Reading Digits in Natural Images with Unsupervised Feature Learning' (cited on pages 60, 75).

Cuong V Nguyen, Tal Hassner, Cedric Archambeau, and Matthias Seeger (2020). 'LEEP: A New Measure to Evaluate Transferability of Learned Representations'. In: *Proceedings of the International Conference on Machine Learning* (cited on page 94).

Alex Nichol and John Schulman (2018). 'Reptile: A Scalable Metalearning Algorithm'. *arXiv preprint arXiv:1803.02999* (cited on pages 37, 80, 88, 89).

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016a). 'WaveNet: A Generative Model for Raw Audio'. In: *Arxiv* (cited on pages 23, 45).

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu koray, Oriol Vinyals, and Alex Graves (2016b). 'Conditional Image Generation with PixelCNN Decoders'. In: *Advances in Neural Information Processing Systems 29* (cited on page 23).

Boris Oreshkin, Pau Rodriguez López, and Alexandre Lacoste (2018). 'TADAM: Task Dependent Adaptive Metric for Improved Few-Shot Learning'. In: *Advances in Neural Information Processing Systems 31* (cited on pages 80, 89).

George Papamakarios and Iain Murray (2016). 'Fast ϵ-Free Inference of Simulation Models with Bayesian Conditional Density Estimation'. In: *Advances in Neural Information Processing Systems 29* (cited on pages 59, 143).

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2019a). 'Normalizing Flows for Probabilistic Modeling and Inference'. *arXiv preprint arXiv:1912.02762* (cited on page 100).

George Papamakarios, David Sterratt, and Iain Murray (2019b). 'Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows'. In: (cited on page 23).

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran (2018). 'Image Transformer'. In: *Proceedings of the 35th International Conference on Machine Learning* (cited on pages 23, 25, 63, 149, 152).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32* (cited on pages 21, 169, 172).

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville (2018). 'FiLM: Visual Reasoning with a General Conditioning Layer'. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (cited on pages 82, 83, 88).

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato (2019). 'Bayesian Batch Active Learning as Sparse Subset Approximation'. In: *Advances in Neural Information Processing Systems 32*.

Boris T Polyak (1964). 'Some Methods of Speeding up the Convergence of Iteration Methods'. *USSR Computational Mathematics and Mathematical Physics* (cited on page 21).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas (2017a). 'Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation'. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (cited on pages 14, 16, 50, 85).

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas (2017b). 'PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space'. In: *Advances in Neural Information Processing Systems 30* (cited on pages 14, 50).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer'. *Journal of Machine Learning Research* (cited on page 38).

Carl Edward Rasmussen (2003). 'Gaussian Processes in Machine Learning'. In: *Summer School on Machine Learning*. Springer (cited on page 13).

Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos (2017). 'Equivariance Through Parameter-Sharing'. In: *Proceedings of the 34th International Conference on Machine Learning* (cited on pages 45, 58).

Sachin Ravi and Hugo Larochelle (2017). 'Optimization as a Model for Few-Shot Learning'. In: *In International Conference on Learning Representations (ICLR)* (cited on pages 14, 37, 87–89).

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi (2017). 'Learning Multiple Visual Domains with Residual Adapters'. In: *Advances in Neural Information Processing Systems 30* (cited on pages 83, 88, 93).

— (2018). 'Efficient Parametrization of Multi-Domain Deep Neural Networks'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (cited on pages 83, 88, 92, 93, 171).

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner (2019). 'Fast and Flexible Multi-Task Classification using Conditional Neural Adaptive Processes'. In: *Advances in Neural Information Processing Systems 32* (cited on pages 2, 3, 17, 79).

Danilo Rezende and Shakir Mohamed (2015). 'Variational Inference with Normalizing Flows'. In: *Proceedings of the 32nd International Conference on Machine Learning* (cited on pages 28, 29).

Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain (2013). 'Gaussian Processes for Time-Series Modelling'. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (cited on pages 44, 45).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015). 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer (cited on page 140).

Sheldon M Ross, John J Kelly, Roger J Sullivan, William James Perry, Donald Mercer, Ruth M Davis, Thomas Dell Washburn, Earl V Sager, Joseph B Boyce, and Vincent L Bristow (1996). *Stochastic Processes*. Wiley New York (cited on page 11).

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams (1986). 'Learning Representations by Backpropagating Errors'. *nature* (cited on page 21).

Jürgen Schmidhuber (1987). 'Evolutionary Principles in Self-Referential Learning'. PhD thesis. Technische Universität München (cited on page 1).

B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij (2012). 'On Causal and Anticausal Learning'. In: *Proceedings of the 29th International Conference on Machine Learning* (cited on page 14).

Dino Sejdinovic and Arthur Gretton (2012). *What is an RKHS?* (Cited on pages 33, 117, 118).

Jake Snell, Kevin Swersky, and Richard Zemel (2017). 'Prototypical Networks for Few-shot Learning'. In: *Advances in Neural Information Processing Systems 30* (cited on pages 14, 37, 80, 88, 89, 94).

Jasper Snoek, Hugo Larochelle, and Ryan P Adams (2012). 'Practical Bayesian Optimization of Machine Learning Algorithms'. In: *Advances in Neural Information Processing Systems 25* (cited on pages 37, 77).

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse (2019). 'Functional Variational Bayesian Neural Networks'. In: *International Conference on Learning Representations* (cited on page 71).

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton (2013). 'On the Importance of Initialization and Momentum in Deep Learning'. In: (cited on page 21).

Richard Sutton (2019). 'The Bitter Lesson'. *Incomplete Ideas (blog), March* (cited on page 9).

Richard S Sutton and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press (cited on page 37).

Terence Tao (2011). *An Introduction to Measure Theory*. American Mathematical Society Providence, RI (cited on page 11).

William R Thompson (1933). 'On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples'. *Biometrika* (cited on page 77).

Sebastian Thrun and Lorien Pratt (2012). *Learning to Learn*. Springer Science & Business Media (cited on pages 1, 9).

Tijmen Tieleman and Geoffrey Hinton (2012). 'Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude'. *COURSERA: Neural networks for machine learning* (cited on page 21).

Dustin Tran, Rajesh Ranganath, and David Blei (2017). 'Hierarchical Implicit Models and Likelihood-Free Variational Inference'. In: *Advances in Neural Information Processing Systems 30* (cited on page 59).

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle (2020). 'Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples'. In: *International Conference on Learning Representations* (cited on pages 14, 79, 80, 82, 86–93, 95, 169–171).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). 'Attention is All you Need'. In: *Advances in Neural Information Processing Systems 30* (cited on pages 8, 25, 142, 152).

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu koray, and Daan Wierstra (2016). 'Matching Networks for One Shot Learning'. In: *Advances in Neural Information Processing Systems 29* (cited on pages 80, 87–89).

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne (2019). 'On the Limitations of Representing Functions on Sets'. In: *Proceedings of the 36th International Conference on Machine Learning* (cited on pages 16, 34).

Martin J Wainwright and Michael Irwin Jordan (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc (cited on page 28).

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick (2016). 'Learning to reinforcement learn'. *arXiv preprint arXiv:1611.05763* (cited on page 37).

Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun (2018). 'Deep Parametric Continuous Convolutional Neural Networks'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (cited on page 50).

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). 'Generalizing from a Few Examples: A Survey on Few-Shot Learning'. *ACM Comput. Surv.* (cited on page 38).

Lillian Weng (2019). *Meta Reinforcement Learning*. URL: https://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html#back-in-2001 (visited on 23/06/2019) (cited on page 37).

Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein (2017). 'Learned Optimizers that Scale and Generalize'. In: (cited on page 37).

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman (2017). 'Convolutional Gaussian Processes'. In: *Advances in Neural Information Processing Systems 30* (cited on page 98).

Darren J Wilkinson (2011). *Stochastic Modelling for Systems Biology*. CRC press (cited on pages 59, 142).

Christopher KI Williams and Carl Edward Rasmussen (2006). *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA (cited on pages 13, 55).

Wenxuan Wu, Zhongang Qi, and Li Fuxin (2019). 'PointConv: Deep Convolutional Networks on 3D Point Clouds'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on pages 14, 16, 50, 54).

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse (2016). 'On the Quantitative Analysis of Decoder-Based Generative Models'. *arXiv preprint arXiv:1611.04273* (cited on page 145).

Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam Kosiorek, and Yee Whye Teh (2020). 'MetaFun: Meta-Learning with Iterative Functional Updates'. In: *Proceedings of the 37th International Conference on Machine Learning* (cited on page 17).

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). 'Show, Attend and Tell: Neural Image Caption Generation with Visual Attention'. In: *Proceedings of the 32nd International Conference on Machine Learning* (cited on page 25).

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn (2018). 'Bayesian Model-Agnostic Meta-Learning'. In: *Advances in Neural Information Processing Systems 31* (cited on page 88).

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson (2014). 'How Transferable are Features in Deep Neural Networks?' In: *Advances in Neural Information Processing Systems 27* (cited on page 89).

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola (2017). 'Deep Sets'. In: *Advances in Neural Information Processing Systems 30* (cited on pages 2, 14–16, 33, 34, 39, 46–49, 85, 132).

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt (2018). 'Advances in Variational Inference'. *IEEE transactions on pattern analysis and machine intelligence* (cited on page 28).

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson (2019). 'Fast Context Adaptation via Meta-Learning'. In: *Proceedings of the 36th International Conference on Machine Learning* (cited on page 88).

Barret Zoph and Quoc V Le (2016). 'Neural architecture search with reinforcement learning'. *arXiv preprint arXiv:1611.01578* (cited on page 37).

APPENDIX

A

REPRODUCING KERNEL HILBERT SPACES

Here we provide a brief review of some concepts related to reproducing kernel Hilbert spaces (RKHS; Sejdinovic and Gretton, 2012) that are used in the thesis. We begin with the definition of a Hilbert space.

Definition A.1 (Hilbert space)
A Hilbert space *is a complete inner product space. In other words, it is a* Banach space *endowed with an inner product.*

Let $\mathcal{X} = \mathbb{R}^d$ and let $\mathcal{Y} \subset \mathbb{R}$ be compact. Let ψ be a symmetric, positive-definite kernel on $\mathcal{X}$. By the Moore–Aronszajn Theorem (Aronszajn, 1950), there is a unique Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ of real-valued functions on $\mathcal{X}$ for which ψ is a reproducing kernel. This means that (i) $\psi(\cdot, \mathbf{x}) \in \mathcal{H}$ for all $\mathbf{x} \in \mathcal{X}$ and (ii) $\langle f, \psi(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = f(\mathbf{x})$ for all $f \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{X}$ (reproducing property). For $\psi \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{X}^n$, and $\mathbf{X}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n) \in \mathcal{X}^n$, we denote

$$\psi(\mathbf{X}, \mathbf{X}') = \begin{bmatrix} \psi(\mathbf{x}_1, \mathbf{x}'_1) & \cdots & \psi(\mathbf{x}_1, \mathbf{x}'_n) \\ \vdots & \ddots & \vdots \\ \psi(\mathbf{x}_n, \mathbf{x}'_1) & \cdots & \psi(\mathbf{x}_n, \mathbf{x}'_n) \end{bmatrix}.$$

Definition A.2 (Interpolating RKHS)
Call $\mathcal{H}$ interpolating *if it interpolates any finite number of points: for every*

$$((\mathbf{x}_i, y_i))_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$$

with $(\mathbf{x}_i)_{i=1}^n$ *all distinct, there is an* $f \in \mathcal{H}$ *such that* $f(\mathbf{x}_1) = y_1, \ldots, f(\mathbf{x}_n) = y_n$.

For example, the RKHS induced by any strictly positive-definite kernel, e.g. the exponentiated quadratic (EQ) kernel $\psi(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(-\frac{1}{2\ell^2} \|\mathbf{x} - \mathbf{x}'\|^2)$, is interpolating: Let $\mathbf{c} = \psi(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}$ and consider $f = \sum_{i=1}^n c_i \psi(\cdot, \mathbf{x}_i) \in \mathcal{H}$. Then $f(\mathbf{X}) = \psi(\mathbf{X}, \mathbf{X})\mathbf{c} = \mathbf{y}$.

The next lemma will be useful in proving several results in Chapters 2 and 3.

Lemma A.1
Let $\mathcal{X} \subseteq \mathbb{R}^d$. *For any* $N \in \mathbb{N}$, *there exists an* N*-dimensional RKHS of functions* $\mathcal{X} \to \mathbb{R}$ *that is* N*-interpolating.*

Remark A.1
Since all N*-dimensional inner product spaces are isomorphic to* $\mathbb{R}^N$ *with the usual inner product, we may take* $\mathcal{H} \cong \mathbb{R}^N$ *by choosing a basis.*

Proof.
Let ψ be the EQ kernel. Fix $\mathbf{Z} = (\mathbf{z}_1, \cdots, \mathbf{z}_N)$ with $\mathbf{z}_i \in \mathcal{X}$ and $\mathbf{z}_i \neq \mathbf{z}_j$ when $i \neq j$. Define $k(\mathbf{x}, \mathbf{x}') \coloneqq \psi(\mathbf{x}, \mathbf{Z})\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \mathbf{x}')$. Note that $\psi(\mathbf{Z}, \mathbf{Z})^{-1}$ always exists since the $\mathbf{z}_i$ are distinct and the EQ kernel is strictly positive definite. We first show that k is a positive semi-definite function. Let $M \geq 1$, $\mathbf{a} \in \mathbb{R}^M$, $\mathbf{X} \in \mathcal{X}^M$. Then

$$\mathbf{a}^\mathsf{T} k(\mathbf{X}, \mathbf{X})\mathbf{a} = \mathbf{a}^\mathsf{T}\psi(\mathbf{X}, \mathbf{Z})\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \mathbf{X})\mathbf{a} \tag{A.1}$$

$$= (\psi(\mathbf{Z}, \mathbf{X})\mathbf{a})^\mathsf{T}\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \mathbf{X})\mathbf{a} \tag{A.2}$$

$$\geq 0, \tag{A.3}$$

which holds since $\psi(\mathbf{Z}, \mathbf{Z})^{-1}$ is positive definite. By the Moore-Aronszajn theorem, there is then a unique RKHS of functions from $\mathcal{X} \to \mathbb{R}$, denoted $\mathcal{H}$, for which k is a reproducing kernel. Consider the pre-RKHS

$$\mathcal{H}_0 = \left\{ \sum_{i=1}^{I} \alpha_i k(\mathbf{x}_i, \cdot) \,\middle|\, I \in \mathbb{N}, \alpha_i \in \mathbb{R}, \mathbf{x}_i \in \mathcal{X} \right\}. \tag{A.4}$$

For $f, g \in \mathcal{H}_0$ with $f = \sum_{i=1}^{I} \alpha_i k(\mathbf{x}_i, \cdot)$ and $g = \sum_{j=1}^{J} \beta_i k(\mathbf{y}_j, \cdot)$, we define the inner product as

$$\langle f, g \rangle_{\mathcal{H}_0} \coloneqq \sum_{i=1}^{I}\sum_{j=1}^{J} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{y}_j). \tag{A.5}$$

It can be easily verified (Sejdinovic and Gretton, 2012)[Theorem 42] that $\mathcal{H}_0$ is a well-defined inner-product space and that k is a reproducing kernel for $\mathcal{H}_0$. To see that $\mathcal{H}_0$ is an RKHS, it remains to be shown that it is complete.

We first show that $\mathcal{H}_0$ is a *finite*-dimensional vector space. Consider the set of $\mathcal{H}_0$-elements $\{k(\mathbf{z}_n, \cdot)\}_{n=1}^N = \{\psi(\mathbf{z}_n, \mathbf{Z})\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \cdot)\}_{n=1}^N$. To show they are linearly independent, choose $\{a_n\}_{n=1}^N$ such that

$$f = \sum_{n=1}^{N} a_n \psi(\mathbf{z}_n, \mathbf{Z})\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \cdot) = 0.$$

Then in particular,

$$[f(\mathbf{z}_1), \ldots, f(\mathbf{z}_N)] = \sum_{n=1}^{N} a_n \psi(\mathbf{z}_n, \mathbf{Z})\psi(\mathbf{Z}, \mathbf{Z})^{-1}\psi(\mathbf{Z}, \mathbf{Z}) \tag{A.6}$$

$$= \sum_{n=1}^{N} a_n \psi(\mathbf{z}_n, \mathbf{Z}) = 0. \tag{A.7}$$

Since $\psi(\mathbf{Z}, \mathbf{Z})$ is invertible, this implies $a_n = 0$ for $1 \leq n \leq N$. Next we show that $\{k(\mathbf{z}_n, \cdot)\}_{n=1}^N$ spans $\mathcal{H}_0$. Let $f \in \mathcal{H}_0$. Then we can write

$$f = \sum_{i=1}^{I} \alpha_i k(\mathbf{x}_i, \cdot) \tag{A.8}$$

$$= \sum_{i=1}^{I} \alpha_i \psi(\mathbf{x}_i, \mathbf{Z}) \psi(\mathbf{Z}, \mathbf{Z})^{-1} \psi(\mathbf{Z}, \cdot). \tag{A.9}$$

But for any $\mathbf{v} \in \mathbb{R}^N$, we have that $\mathbf{v}$ is a linear combination of $\{\psi(\mathbf{z}_n, \mathbf{Z})\}_{n=1}^N$, since $\psi(\mathbf{Z}, \mathbf{Z})$ is invertible. Applying this to $\mathbf{v} = \sum_{i=1}^I \alpha_i \psi(\mathbf{x}_i, \mathbf{Z})$, we see that there exist $c_1, \ldots, c_N \in \mathbb{R}$ such that

$$f = \sum_{n=1}^{N} c_n \psi(\mathbf{z}_n, \mathbf{Z}) \psi(\mathbf{Z}, \mathbf{Z})^{-1} \psi(\mathbf{Z}, \cdot) = \sum_{n=1}^{N} c_n k(\mathbf{z}_n, \cdot).$$

Hence $\{k(\mathbf{z}_n, \cdot)\}_{n=1}^N$ is a basis for $\mathcal{H}_0$, which is N-dimensional. Since all N-dimensional inner product spaces are isomorphic to $\mathbb{R}^N$, which is complete, it follows that $\mathcal{H}_0$ is complete, and hence is the RKHS $\mathcal{H}$ for which k is a reproducing kernel.

Finally, we show that $\mathcal{H}$ is N-interpolating. Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$ with each $\mathbf{x}_i$ distinct. We show that $k(\mathbf{X}, \mathbf{X})$ is invertible. Let $\mathbf{a} \in \mathbb{R}^N$. Then

$$\mathbf{a}^\mathsf{T} k(\mathbf{X}, \mathbf{X}) \mathbf{a} = \mathbf{a}^\mathsf{T} \psi(\mathbf{X}, \mathbf{Z}) \psi(\mathbf{Z}, \mathbf{Z})^{-1} \psi(\mathbf{Z}, \mathbf{X}) \mathbf{a} \tag{A.10}$$

$$= (\psi(\mathbf{Z}, \mathbf{X})\mathbf{a})^\mathsf{T} \psi(\mathbf{Z}, \mathbf{Z})^{-1} \psi(\mathbf{Z}, \mathbf{X}) \mathbf{a} \tag{A.11}$$

$$\geq 0, \tag{A.12}$$

with equality if and only if $\psi(\mathbf{Z}, \mathbf{X})\mathbf{a} = 0$. Suppose $\psi(\mathbf{Z}, \mathbf{X})\mathbf{a} = 0$. Note that $\psi(\mathbf{Z}, \mathbf{X})$ is invertible. Hence $\mathbf{a} = 0$, and $k(\mathbf{X}, \mathbf{X})$ is positive definite, hence invertible. Let $\mathbf{b} = k(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}$ and consider $f = \sum_{n=1}^N b_n k(\mathbf{x}_i, \cdot) \in \mathcal{H}$. Then $f(\mathbf{X}) = k(\mathbf{X}, \mathbf{X})\mathbf{b} = \mathbf{y}$. □

B

THE QUOTIENT SPACE OF PERMUTATIONS

Here we provide a brief review of some concepts quotient space topology (Munkres, 1974) that are used in the thesis. The review is accompanied by several simple statements that will be used in the thesis, along with their proofs.

Let $\mathcal{A}$ be a Banach space. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{A}^n$ and $\mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{A}^n$, let $\mathbf{x} \sim \mathbf{y}$ if $\mathbf{x}$ is a permutation of $\mathbf{y}$; that is, $\mathbf{x} \sim \mathbf{y}$ if and only if $\mathbf{x} = \pi\mathbf{y}$ for some $\pi \in \mathbb{S}_n$ where

$$\pi\mathbf{y} = (y_{\pi(1)}, \ldots, y_{\pi(n)}).$$

Let $\mathcal{A}^n/\mathbb{S}_n$ be the collection of equivalence classes of $\sim$. Denote the equivalence class of $\mathbf{x}$ by $[\mathbf{x}]$; for $A \subset \mathcal{A}^n$, denote $[A] = \{[\mathbf{a}] : \mathbf{a} \in A\}$. Call the map $\mathbf{x} \mapsto [\mathbf{x}] \colon \mathcal{A}^n \twoheadrightarrow \mathcal{A}^n/\mathbb{S}_n$ the canonical map. The natural topology on $\mathcal{A}^n/\mathbb{S}_n$ is the quotient topology, in which a subset of $\mathcal{A}^n/\mathbb{S}_n$ is open if and only if its preimage under the canonical map is open in $\mathcal{A}^n$. In what follows, we show that the quotient topology is metrizable.

On $\mathcal{A}^n$, since all norms on finite-dimensional vector spaces are equivalent, without loss of generality consider

$$\|\mathbf{x}\|_{\mathcal{A}^n}^2 = \sum_{i=1}^{n} \|x_i\|_{\mathcal{A}}^2.$$

Note that $\|\cdot\|_{\mathcal{A}^n}$ is permutation invariant: $\|\pi \cdot\|_{\mathcal{A}^n} = \|\cdot\|_{\mathcal{A}^n}$ for all $\pi \in \mathbb{S}_n$. On $\mathcal{A}^n/\mathbb{S}_n$, define

$$d \colon \mathcal{A}^n/\mathbb{S}_n \times \mathcal{A}^n/\mathbb{S}_n \twoheadrightarrow [0, \infty), \quad d([\mathbf{x}], [\mathbf{y}]) = \min_{\pi \in \mathbb{S}_n} \|\mathbf{x} - \pi\mathbf{y}\|_{\mathcal{A}^n}.$$

Call a set $[A] \subset \mathcal{A}^n/\mathbb{S}_n$ bounded if $\{d([\mathbf{x}], [0]) : [\mathbf{x}] \in [A]\}$ is bounded.

Proposition B.1
The function d is a metric.

Proof.
We first show that d is well defined on $\mathcal{A}^n/\mathbb{S}_n$. Assume $\mathbf{x} \sim \mathbf{x}'$ and $\mathbf{y} \sim \mathbf{y}'$. Then, $\mathbf{x}' = \pi_{\mathbf{x}}\mathbf{x}$ and $\mathbf{y}' = \pi_{\mathbf{y}}\mathbf{y}$. Using the group properties of $\mathbb{S}_n$ and the permutation invariance of $\|\cdot\|_{\mathcal{A}^n}$:

$$\begin{aligned} d([\mathbf{x}'],[\mathbf{y}']) &= \min_{\pi\in\mathbb{S}_n} \|\pi_{\mathbf{x}}\mathbf{x} - \pi\pi_{\mathbf{y}}\mathbf{y}\|_{\mathcal{A}^n} \\ &= \min_{\pi\in\mathbb{S}_n} \|\pi_{\mathbf{x}}\mathbf{x} - \pi\mathbf{y}\|_{\mathcal{A}^n} \\ &= \min_{\pi\in\mathbb{S}_n} \|\mathbf{x} - \pi_{\mathbf{x}}^{-1}\pi\mathbf{y}\|_{\mathcal{A}^n} \\ &= \min_{\pi\in\mathbb{S}_n} \|\mathbf{x} - \pi\mathbf{y}\|_{\mathcal{A}^n} \\ &= d([\mathbf{x}],[\mathbf{y}]). \end{aligned}$$

It is clear that $d([\mathbf{x}],[\mathbf{y}]) = d([\mathbf{y}],[\mathbf{x}])$ and that $d([\mathbf{x}],[\mathbf{y}]) = 0$ if and only if $[\mathbf{x}] = [\mathbf{y}]$. To show the triangle inequality, note that

$$\|\mathbf{x}-\pi_1\pi_2\mathbf{y}\|_{\mathcal{A}^n} \le \|\mathbf{x}-\pi_1\mathbf{z}\|_{\mathcal{A}^n}+\|\pi_1\mathbf{z}-\pi_1\pi_2\mathbf{y}\|_{\mathcal{A}^n} = \|\mathbf{x}-\pi_1\mathbf{z}\|_{\mathcal{A}^n}+\|\mathbf{z}-\pi_2\mathbf{y}\|_{\mathcal{A}^n},$$

using permutation invariance of $\|\cdot\|_{\mathcal{A}^n}$. Hence, taking the minimum over π_1,

$$d([\mathbf{x}],[\mathbf{y}]) \le d([\mathbf{x}],[\mathbf{z}]) + \|\mathbf{z}-\pi_2\mathbf{y}\|_{\mathcal{A}^n},$$

so taking the minimum over π_2 gives the triangle inequality for d. □

Proposition B.2
The canonical map $\mathcal{A}^n \twoheadrightarrow \mathcal{A}^n/\mathbb{S}_n$ is continuous under the metric topology induced by d.

Proof.
Follows directly from $d([\mathbf{x}],[\mathbf{y}]) \le \|\mathbf{x}-\mathbf{y}\|_{\mathcal{A}^n}$. □

Proposition B.3
Let $A \subset \mathcal{A}^n$ be topologically closed and closed under permutations. Then $[A]$ is topologically closed in $\mathcal{A}^n/\mathbb{S}_n$ under the metric topology.

Proof.
Recall that a subset $[A]$ of a metric space is closed iff every limit point of $[A]$ is also in $[A]$. Consider a sequence $([\mathbf{a}_n])_{n=1}^{\infty} \subset [A]$ converging to some $[\mathbf{x}] \in \mathcal{A}^n/\mathbb{S}_n$. Then there are permutations $(\pi_n)_{n=1}^{\infty} \subset \mathbb{S}_n$ such that $\pi_n\mathbf{a}_n \twoheadrightarrow \mathbf{x}$. Here $\pi_n\mathbf{a}_n \in A$, because A is closed under permutations. Thus $\mathbf{x} \in A$, as A is also topologically closed. We conclude that $[\mathbf{x}] \in [A]$. □

Proposition B.4
Let $A \subset \mathcal{A}^n$ be open. Then $[A]$ is open in $\mathcal{A}^n/\mathbb{S}_n$ under the metric topology. In other words, the canonical map is open under the metric topology.

Proof.
Let $[\mathbf{x}] \in [A]$. Because A is open, there is some ball $B_{\mathrm{e}}(\mathbf{y})$ with $\mathrm{e} > 0$ and $\mathbf{y} \in A$ such that $\mathbf{x} \in B_{\mathrm{e}}(\mathbf{y}) \subset A$. Then $[\mathbf{x}] \in B_{\mathrm{e}}([\mathbf{y}])$, since $d([\mathbf{x}], [\mathbf{y}]) \leq \|\mathbf{x}-\mathbf{y}\|_{\mathcal{A}^n} < \mathrm{e}$, and we claim that $B_{\mathrm{e}}([\mathbf{y}]) \subset [A]$. Hence $[\mathbf{x}] \in B_{\mathrm{e}}([\mathbf{y}]) \subset [A]$, so $[A]$ is open.

To show the claim, let $[\mathbf{z}] \in B_{\mathrm{e}}([\mathbf{y}])$. Then $d(\pi\mathbf{z}, \mathbf{y}) < \mathrm{e}$ for some $\pi \in \mathbb{S}_n$. Hence $\pi\mathbf{z} \in B_{\mathrm{e}}(\mathbf{y}) \subset A$, so $\pi\mathbf{z} \in A$. Therefore, $[\mathbf{z}] = [\pi\mathbf{z}] \in [A]$. □

Proposition B.5
The quotient topology on $\mathcal{A}^n/\mathbb{S}_n$ induced by the canonical map is metrizable with the metric d.

Proof.
Since the canonical map is surjective, there exists exactly one topology on $\mathcal{A}^n/\mathbb{S}_n$ relative to which the canonical map is a quotient map: the quotient topology (Munkres, 1974).

Let $p\colon \mathcal{A}^n \twoheadrightarrow \mathcal{A}^n/\mathbb{S}_n$ denote the canonical map. It remains to show that p is a quotient map under the metric topology induced by d; that is, we show that $U \subset \mathcal{A}^n/\mathbb{S}_n$ is open in $\mathcal{A}^n/\mathbb{S}_n$ under the metric topology if and only if $p^{-1}(U)$ is open in $\mathcal{A}^n$.

Let $p^{-1}(U)$ be open in $\mathcal{A}^n$. We have that $U = p(p^{-1}(U))$, so U is open in $\mathcal{A}^n/\mathbb{S}_n$ under the metric topology by Proposition B.4. Conversely, if U is open in $\mathcal{A}^n/\mathbb{S}_n$ under the metric topology, then $p^{-1}(U)$ is open in $\mathcal{A}^n$ by continuity of the canonical map under the metric topology. □

C

PROOFS OF THEOREMS 2.3 AND 2.4

We provide the proofs for Theorems 2.3 and 2.4. Our proof strategy is as follows. We will first show that there exists a homemorphic (i.e., continuous with a continuous inverse) embedding of datasets size to m into a RKHS (Lemma C.1). Then, we will demonstrate that this embedding encodes datasets of different sizes into disjoint subsets of the RKHS, such that these may be "stitched together" to construct a homemorphic embedding for datasets of size less than or equal to M. Finally, we will demonstrate that these subsets are closed in the embedding space, and use the pasting lemma (Munkres, 1974) to show that there exists a continuous map that satisfies our desiderata. We conclude by "putting the results together" to prove our theorems.

Lemma C.1

Let $\mathcal{S} \subset \mathbb{R}^d$ be compact, and let $\mathcal{S}_m = \underbrace{\mathcal{S} \times \ldots \times \mathcal{S}}_{m \text{ times}}$, where $m \leq M$. Let $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be the kernel defined in the proof of Lemma A.1 with $2M$ points in $\mathbf{Z}$. Denote $\mathcal{H}$ as the $2M$-dimensional RKHS associated with k, and for $m \leq M$ let

$$\mathcal{H}_m = \left\{ \sum_{i=1}^{m} k(\cdot, \mathbf{x}_i) \,\middle|\, (\mathbf{x}_i)_{i=1}^m \in \mathcal{S}_m \right\} \subseteq \mathcal{H} \cong \mathbb{R}^{2M}.$$

Then the map

$$E_m : [\mathcal{S}_m] \to \mathcal{H}_m, \quad E_m([(\mathbf{x}_1, ..., \mathbf{x}_m)]) = \sum_{i=1}^{m} k(\cdot, \mathbf{x}_i)$$

is injective, hence invertible, and continuous. Moreover, the inverse E_m^{-1} is continuous, i.e., E_m is a homeomorphism.

Proof.

First note that E_m is clearly permutation invariant and hence well-defined on $[\mathcal{S}_m]$. To show that it is injective, let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ and let $\mathbf{X}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_m)$. Suppose

$$E_m([\mathbf{X}]) = E_m([\mathbf{X}']); \quad \text{and} \quad \sum_{i=1}^{m} k(\cdot, \mathbf{x}_i) = \sum_{i=1}^{m} k(\cdot, \mathbf{x}'_i). \tag{C.1}$$

Taking the inner product with any $f \in \mathcal{H}$ and using the reproducing property,

$$\sum_{i=1}^{m} f(\mathbf{x}_i) = \sum_{i=1}^{m} f(\mathbf{x}'_i), \tag{C.2}$$

for all $f \in \mathcal{H}$. Let $\hat{\mathbf{x}} \in \mathbf{X} \cup \mathbf{X}'$. Since $\mathcal{H}$ is $2M$-interpolating, and the number of distinct points in $\mathbf{X} \cup \mathbf{X}'$ is at most $2M$, we can find a function $f \in \mathcal{H}$ such that $f(\hat{\mathbf{x}}) = 1$ and $f(\mathbf{x}) = 0$ for all other $\mathbf{x}$ in $\mathbf{X} \cup \mathbf{X}'$. Then

$$\sum_{i:\mathbf{x}_i=\hat{\mathbf{x}}} 1 = \sum_{i:\mathbf{x}'_i=\hat{\mathbf{x}}} 1, \tag{C.3}$$

hence the number of occurrences of $\hat{\mathbf{x}}$ in $\mathbf{X}$ is the same as the number of occurrences of $\hat{\mathbf{x}}$ in $\mathbf{X}'$. Since this is true for all $\hat{\mathbf{x}} \in \mathbf{X} \cup \mathbf{X}'$, it follows that $\mathbf{X}$ is a permutation of $\mathbf{X}'$, so that $[\mathbf{X}] = [\mathbf{X}']$, proving injectivity.

To show that E_m is continuous, compute

$$\left\| \sum_{i=1}^{m} k(\cdot, \mathbf{x}_i) - \sum_{i=1}^{m} k(\cdot, \mathbf{x}'_i) \right\|_{\mathcal{H}}^2 \tag{C.4}$$

$$= \left\langle \sum_{i=1}^{m} k(\cdot, \mathbf{x}_i) - k(\cdot, \mathbf{x}'_i) \,,\, \sum_{j=1}^{m} k(\cdot, \mathbf{x}_j) - k(\cdot, \mathbf{x}'_j) \right\rangle_{\mathcal{H}} \tag{C.5}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} \left[k(\mathbf{x}_i, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}'_j) + k(\mathbf{x}'_i, \mathbf{x}'_j) \right], \tag{C.6}$$

where we used the reproducing property of k. This goes to zero as $[\mathbf{X}'] \twoheadrightarrow [\mathbf{X}]$ by continuity of k. Finally, since the continuous image of a compact set is compact, and the canonical map from $\mathcal{S}_m$ to $[\mathcal{S}_m]$ is continuous, $[\mathcal{S}_m]$ is compact. Since any continuous bijection between a compact space $[\mathcal{S}_m]$ and a Hausdorff space $\mathcal{H}_m$ is a homeomorphism, it follows that E_m is a homeomorphism. □

Lemma C.1 demonstrates that there exist homeomorphic map E_m from the space of fixed-sized data sets $[\mathcal{S}]_m$ to finite-dimensional Hilbert spaces. Next, we study the properties of the map E, which has as its restrictions the maps E_m.

Lemma C.2

Let

$$[\mathcal{S}_{\leq M}] = \bigcup_{m=1}^{M} [\mathcal{S}_m]\,; \quad \textit{and} \quad \mathcal{H}_{\leq M} = \bigcup_{m=1}^{M} \mathcal{H}_m \subseteq \mathcal{H} \cong \mathbb{R}^{2M}.$$

Then, the $\{\mathcal{H}_m\}_{m=1}^{M}$ are pairwise disjoint. It follows that the map E

$$E\colon\ [\mathcal{S}_{\leq M}] \twoheadrightarrow \mathcal{H}_{\leq M}; \quad E([S]) = E_m([S]) \quad \textit{if} \quad [S] \in [\mathcal{S}_m]$$

is injective, hence invertible. Denote this inverse as E^{-1}*, where* $E^{-1}(f) = E_m^{-1}(f)$ *if* $f \in \mathcal{H}_m$*.*

Proof.
Assume the $\{\mathcal{H}_m\}_{m=1}^M$ are not pairwise disjoint. Then there exists $[\mathbf{X}] \in [\mathcal{S}_m]$ and $[\mathbf{X}'] \in [\mathcal{S}_{m'}]$ with $m \neq m'$ such that

$$E_m([\mathbf{X}]) = E_{m'}([\mathbf{X}']) \tag{C.7}$$

$$\sum_{i=1}^{m} k(\cdot, \mathbf{x}_i) = \sum_{i=1}^{m'} k(\cdot, \mathbf{x}'_i). \tag{C.8}$$

Again taking the inner product with any $f \in \mathcal{H}$ and using the reproducing property,

$$\sum_{i=1}^{m} f(\mathbf{x}_i) = \sum_{i=1}^{m'} f(\mathbf{x}'_i). \tag{C.9}$$

Since $\mathcal{H}$ is $2M$-interpolating, we set $f(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathbf{X} \cup \mathbf{X}'$, which implies $m = m'$, a contradiction. □

Next, we show that we may use the inverse map E^{-1}, in conjunction with the pasting lemma (Munkres, 1974), to construct a continuous map from the image of the embedding E to a target toplogical space.

Lemma C.3
Let T *be a topological space. Let* $\Phi\colon [\mathcal{S}_{\leq M}] \to T$*, such that the restrictions* $\Phi|_{[\mathcal{S}_m]}$ *are continuous for* $1 \leq m \leq M$*, and let* E *be as defined in Lemma C.2. Then, the map*

$$\Phi \circ E^{-1}\colon \mathcal{H}_{\leq M} \to T \tag{C.10}$$

is continuous.

Proof.
By Lemma C.1, each $E_m^{-1} : \mathcal{H}_m \to [\mathcal{S}_m]$ is continuous. Since

$$\Phi|_{[\mathcal{S}_m]} : [\mathcal{S}_m] \to C_b(\mathcal{X}, \mathcal{Y}) \tag{C.11}$$

is continuous, the composition $\Phi|_{[\mathcal{S}_m]} \circ E_m^{-1} : \mathcal{H}_m \to C_b(\mathcal{X}, \mathcal{Y})$ is also continuous. But $\Phi|_{[\mathcal{S}_m]} \circ E_m^{-1}$ is just $(\Phi \circ E^{-1})|_{\mathcal{H}_m}$. Assume $\mathcal{H}_m$ is closed in $\mathcal{H}_{\leq M}$. Then by the pasting lemma, $\Phi \circ E^{-1}$ is continuous.

It remains to be shown that $\mathcal{H}_m$ is closed in $\mathcal{H}_{\leq M}$. First, as $\mathcal{H}_m$ is the continuous image of a compact set $[\mathcal{S}_m]$, it is compact. Since $\mathcal{H}$ is isomorphic to $\mathbb{R}^{2M}$, it has the Heine-Borel property, implying that $\mathcal{H}_m$ is closed in $\mathcal{H}$. Finally, a set is closed in $\mathcal{H}_{\leq M}$ if and only if it is the intersection of a closed set of $\mathcal{H}$ with $\mathcal{H}_{\leq M}$, which shows that $\mathcal{H}_m$ is closed in $\mathcal{H}_{\leq M}$. □

With these results in place, we are now ready to prove our first theorem regarding the representational power of vector-valued Deep Sets networks of size less than or equal to M.

Theorem C.1 (Deep Sets for vectors)
Let $\mathcal{S} \subset \mathbb{R}^d$ be compact, and let $\mathbf{s}$ denote a generic element of $\mathcal{S}$. Let $\mathcal{S}_m = \underbrace{\mathcal{S} \times \ldots \times \mathcal{S}}_{m \text{ times}}$ and let $[\mathcal{S}_m]$ be the set of equivalence classes of elements of $\mathcal{S}$ under permutation. 5 Let $[\mathcal{S}]_{\leq M} = \bigcup_{m=1}^{M} [\mathcal{S}_m]$. Let T be a topological space. Then any map $g : [\mathcal{S}]_{\leq M} \to T$ such that its restrictions $g|_{[\mathcal{S}_m]}$ are continuous for all $1 \leq m \leq M$ has the form:

$$g([(\mathbf{s}_1, \ldots, \mathbf{s}_m)]) = \rho\left(\sum_{i=1}^{m} \phi(\mathbf{s}_i)\right), \tag{C.12}$$

for some continuous $\rho : \mathbb{R}^{2M} \to T$ and continuous $\phi : \mathcal{S} \to \mathbb{R}^{2M}$. The function ϕ is independent of g.

Proof.
By Lemma C.2 we have that the map $E : [\mathcal{S}]_{\leq M} \to \mathcal{H}_{\leq M}$ is a bijection. However, since $\mathcal{H} \cong \mathbb{R}^{2M}$, we may take $I : \mathcal{H} \to \mathbb{R}^{2M}$ to be an isomorphism of inner product spaces (and hence also a homeomorphism), and define $E' : [\mathcal{S}]_{\leq M} \to I(\mathcal{H}_{\leq M}) \subseteq \mathbb{R}^{2M}$, $E'([S]) = I \circ E([S])$, which is also a bijection. For $[S] \in [\mathcal{S}]_{\leq M}$ we have

$$\begin{aligned} g([S]) &= g([(\mathbf{s}_1, \ldots, \mathbf{s}_m)]) = g(E'^{-1}(E'([S]))) \\ &= g \circ E'^{-1}\left(\sum_{i=1}^{m} I(k(\cdot, \mathbf{s}_i))\right), \end{aligned} \tag{C.13}$$

where we used the linearity of I. Define $\rho : I(\mathcal{H}_{\leq M}) \to T$ by $\rho = g \circ E'^{-1} = g \circ E^{-1} \circ I^{-1}$, which is continuous by Lemma C.3 and continuity of I^{-1} (ρ may be extended continuously in an arbitrary way on $\mathbb{R}^{2M} \setminus I(\mathcal{H}_{\leq M})$ to define a function $\rho : \mathbb{R}^{2M} \to T$). Finally, define $\phi : \mathcal{S} \to \mathbb{R}^{2M}$ as $\phi(\mathbf{s}) = I(k(\cdot, \mathbf{s}))$ and note that it is continuous by continuity of k and I. □

Now, we can use Theorem 2.3 to prove Theorem 2.4, i.e., the representation power of the CNP architecture (Definition 2.4). However, before doing so, we must account for the concatenations inherent to the CNP architecture. This is considered in the following lemma.

Lemma C.4
Let $\rho : \mathbb{R}^{d_r} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ be continuous, where $\mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ is endowed with the topology of uniform convergence. Then there exists $\rho' : \mathbb{R}^{d_r} \times \mathcal{X} \to \mathcal{Y}$ continuous such that for all $\mathbf{r} \in \mathbb{R}^{d_r}$ and $\mathbf{x} \in \mathcal{X}$, $(\rho(\mathbf{r}))(\mathbf{x}) = \rho'((\mathbf{r}, \mathbf{x}))$.

Proof.
We begin by *defining* $\rho'((\mathbf{r}, \mathbf{x})) \coloneqq (\rho(\mathbf{r}))(\mathbf{x})$ for all $\mathbf{r}, \mathbf{x}$. It remains to be shown that ρ' is continuous:

$$\begin{aligned} \|\rho'((\mathbf{r}, \mathbf{x})) - \rho'((\mathbf{r}', \mathbf{x}'))\| &= \|(\rho(\mathbf{r}))(\mathbf{x}) - (\rho(\mathbf{r}'))(\mathbf{x}')\| && \text{(C.14)} \\ &= \|(\rho(\mathbf{r}))(\mathbf{x}) - (\rho(\mathbf{r}'))(\mathbf{x}) + (\rho(\mathbf{r}'))(\mathbf{x}) - (\rho(\mathbf{r}'))(\mathbf{x}')\| && \text{(C.15)} \\ &\leq \|(\rho(\mathbf{r}))(\mathbf{x}) - (\rho(\mathbf{r}'))(\mathbf{x})\| + \|(\rho(\mathbf{r}'))(\mathbf{x}) - (\rho(\mathbf{r}'))(\mathbf{x}')\|. \\ & && \text{(C.16)} \end{aligned}$$

As $(\mathbf{r}', \mathbf{x}') \twoheadrightarrow (\mathbf{r}, \mathbf{x})$, the first term goes to zero by continuity of ρ, and the second term goes to zero by continuity of $\rho(\mathbf{r}')$. □

We are now ready to prove Theorem 2.4.

Theorem C.2 (CNP Representation)
Let $\Phi : [\mathcal{S}]_{\leq M} \twoheadrightarrow \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ with its restrictions $\Phi|_{[\mathcal{S}_m]}$ continuous for $1 \leq m \leq M$. Then Φ can be represented as a conditional neural process with representation dimension $d_r = 2M$. Conversely, any CNP is a map from $[\mathcal{S}]_{\leq M} \twoheadrightarrow C_b(\mathcal{X}, \mathcal{Y})$ with continuous restrictions.

Proof.
Let $[S] = [(\mathbf{s}_1, \ldots, \mathbf{s}_m)] \in [\mathcal{S}_{\leq M}]$. Here each $\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_{\text{in}} + d_{\text{out}}}$. By Theorem 2.3, $\Phi([S]) = \rho\left(\sum_{i=1}^m \phi(\mathbf{s}_i)\right)$ for some continuous $\rho : \mathbb{R}^{2M} \twoheadrightarrow C_b(\mathcal{X}, \mathcal{Y})$ and continuous $\phi : \mathcal{S} \twoheadrightarrow \mathbb{R}^{2M}$. Moreover, by Lemma C.4, there exists a continuous $\rho' : \mathbb{R}^{2M} \times \mathcal{X} \twoheadrightarrow \mathcal{Y}$ such that for all $\mathbf{r} \in \mathbb{R}^{2M}$ and $\mathbf{x} \in \mathcal{X}$, $(\rho(\mathbf{r}))(\mathbf{x}) = \rho'((\mathbf{r}, \mathbf{x}))$. Hence, for all $\mathbf{x} \in \mathcal{X}$,

$$\Phi([S])(\mathbf{x}) = \rho\left(\sum_{i=1}^m \phi(\mathbf{s}_i)\right)(\mathbf{x}) = \rho'\left(\left(\sum_{i=1}^m \phi(\mathbf{s}_i), \mathbf{x}\right)\right) \tag{C.17}$$

Identifying ρ' as the decoder and $E([S]) = \sum_{i=1}^m \phi(\mathbf{s}_i)$ as the encoder with representation dimension $d_r = 2M$ and $\phi = r$, we see that Φ is a CNP. The converse is straightforward. □

D

PROOF OF THEOREM 3.1

We provide the proof of Theorem 3.1. Our proof strategy is very similar to the proof of Theorem 2.3, but specialising the encoder to be translation equivariant, and allowing for embeddings to infinite-dimensional function spaces. We begin by demonstrating that our proposed embedding into function space is homeomorphic (Lemmas D.1 and D.2). Then, we show that the embeddings of fixed-sized sets can be extended to varying-sized sets by "pasting" the embeddings together while maintaining their homeomorphic properties (Lemma D.3). Following this, we demonstrate that the resulting embedding may be composed with a continuous mapping to our desired target space, resulting in a continuous mapping between two metric spaces (Lemma D.4). Finally, we combine the above-mentioned results to prove Theorem 3.1.

EMBEDDINGS OF SETS INTO AN RKHS

Lemma D.3 states that it is possible to homeomorphically embed sets into an RKHS, which is central in proving our main result. Before proving Lemma D.3, we provide several useful intermediate results. We begin by demonstrating that an embedding of sets of a fixed size into a RKHS is continuous and injective.

Lemma D.1
Consider a collection $\mathcal{S}'_M \subset \mathcal{S}_M$ that has multiplicity K. Set

$$\phi : \mathcal{Y} \to \mathbb{R}^{K+1}, \qquad \phi(y) = (y^0, y^1, \cdots, y^K) \tag{D.1}$$

and let ψ be an interpolating, continuous positive-definite kernel. Define

$$\mathcal{H}_M = \left\{ \sum_{i=1}^{M} \phi(y_i)\psi(\cdot, \mathbf{x}_i) : (\mathbf{x}_i, y_i)_{i=1}^{M} \subset \mathcal{S}_M \right\} \subseteq \mathcal{H}^{K+1}, \tag{D.2}$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$-dimensional-vector–valued–function Hilbert space constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel and endowed with the inner product $\langle f, g \rangle_{\mathcal{H}^{K+1}} = \sum_{i=1}^{K+1} \langle f_i, g_i \rangle_{\mathcal{H}}$. Then the embedding

$$E_M : [\mathcal{S}'_M] \to \mathcal{H}_M, \quad E_M([(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)]) = \sum_{i=1}^{M} \phi(y_i)\psi(\cdot, \mathbf{x}_i) \tag{D.3}$$

is injective, hence invertible, and continuous.

Proof.
First, we show that E_M is injective. Suppose that

$$\sum_{i=1}^{M} \phi(y_i)\psi(\cdot, \mathbf{x}_i) = \sum_{i=1}^{M} \phi(y'_i)\psi(\cdot, \mathbf{x}'_i). \tag{D.4}$$

Denote $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_M)$ and $\mathbf{y} = (y_1, \ldots, y_M)$, and denote $\mathbf{X}'$ and $\mathbf{y}'$ similarly. Taking the inner product with any $f \in \mathcal{H}$ on both sides and using the reproducing property of ψ, this implies that

$$\sum_{i=1}^{M} \phi(y_i) f(\mathbf{x}_i) = \sum_{i=1}^{M} \phi(y'_i) f(\mathbf{x}'_i) \tag{D.5}$$

for all $f \in \mathcal{H}$. In particular, since by construction $\phi_1(\cdot) = 1$,

$$\sum_{i=1}^{M} f(\mathbf{x}_i) = \sum_{i=1}^{M} f(\mathbf{x}'_i) \tag{D.6}$$

for all $f \in \mathcal{H}$. Using that $\mathcal{H}$ is interpolating, choose a particular $\hat{\mathbf{x}} \in \mathbf{X} \cup \mathbf{X}'$, and let $f \in \mathcal{H}$ be such that $f(\hat{\mathbf{x}}) = 1$ and $f(\cdot) = 0$ at all other $\mathbf{x}_i$ and $\mathbf{x}'_i$. Then

$$\sum_{i:\mathbf{x}_i = \hat{\mathbf{x}}} 1 = \sum_{i:\mathbf{x}'_i = \hat{\mathbf{x}}} 1, \tag{D.7}$$

so the number of such $\hat{\mathbf{x}}$ in $\mathbf{X}$ and the number of such $\hat{\mathbf{x}}$ in $\mathbf{X}'$ are the same. Since this holds for every $\hat{\mathbf{x}}$, $\mathbf{X}$ is a permutation of $\mathbf{X}'$: $\mathbf{X} = \pi(\mathbf{X}')$ for some permutation $\pi \in \mathbb{S}_M$. Plugging in the permutation, we can write

$$\sum_{i=1}^{M} \phi(y_i) f(\mathbf{x}_i) = \sum_{i=1}^{M} \phi(y'_i) f(\mathbf{x}'_i) \tag{D.8}$$

$$\overset{(\mathbf{X}' = \pi^{-1}(\mathbf{X}))}{=} \sum_{i=1}^{M} \phi(y'_i) f(\mathbf{x}_{\pi^{-1}(i)}) \tag{D.9}$$

$$\overset{(i \leftarrow \pi^{-1}(i))}{=} \sum_{i=1}^{M} \phi(y'_{\pi(i)}) f(\mathbf{x}_i). \tag{D.10}$$

Then, by a similar argument, for any particular $\hat{\mathbf{x}}$,

$$\sum_{i:\mathbf{x}_i = \hat{\mathbf{x}}} \phi(y_i) = \sum_{i:\mathbf{x}_i = \hat{\mathbf{x}}} \phi(y'_{\pi(i)}). \tag{D.11}$$

Let the number of terms in each sum equal S. Since $\mathcal{S}'_M$ has multiplicity K, $S \leq K$. By Lemma 4 from Zaheer et al. (2017), the 'sum-of-power

mapping' from $\{y_i : \mathbf{x}_i = \hat{\mathbf{x}}\}$ to the first $S+1$ elements of $\sum_{i:\mathbf{x}_i=\hat{\mathbf{x}}} \phi(y_i)$, i.e. $\left(\sum_{i:\mathbf{x}_i=\hat{\mathbf{x}}} y_i^0, \ldots, \sum_{i:\mathbf{x}_i=\hat{\mathbf{x}}} y_i^S\right)$, is injective. Therefore,

$$(y_i)_{i:\mathbf{x}_i=\hat{\mathbf{x}}} \quad \text{is a permutation of} \quad (y'_{\pi(i)})_{i:\mathbf{x}_i=\hat{\mathbf{x}}}. \tag{D.12}$$

Note that $\mathbf{x}_i = \hat{\mathbf{x}}$ for all above y_i. Furthermore, note that also $\mathbf{x}'_{\pi(i)} = \mathbf{x}_i = \hat{\mathbf{x}}$ for all above $y'_{\pi(i)}$. We may therefore adjust the permutation π such that $y_i = y'_{\pi(i)}$ for all i such that $\mathbf{x}_i = \hat{\mathbf{x}}$ whilst retaining that $\mathbf{x} = \pi(\mathbf{x}')$. Performing this adjustment for all $\hat{\mathbf{x}}$, we find that $y = \pi(y')$ and $\mathbf{x} = \pi(\mathbf{x}')$.

Second, we show that E_M is continuous. Compute

$$\begin{aligned}
&\left\| \sum_{i=1}^{M} \phi(y_i)\psi(\cdot, \mathbf{x}_i) - \sum_{j=1}^{M} \phi(y'_j)\psi(\cdot, \mathbf{x}'_j) \right\|^2_{\mathcal{H}^{K+1}} \\
&\quad = \sum_{i=1}^{K+1} \Big(\phi_i^\top(\mathbf{y})\psi(\mathbf{X}, \mathbf{X})\phi_i(\mathbf{y}) - 2\phi_i^\top(\mathbf{y})\psi(\mathbf{X}, \mathbf{X}')\phi_i(\mathbf{y}') \\
&\qquad\qquad + \phi_i^\top(\mathbf{y}')\psi(\mathbf{X}', \mathbf{X}')\phi_i(\mathbf{y}') \Big),
\end{aligned}$$

which goes to zero if $[\mathbf{X}', \mathbf{y}'] \twoheadrightarrow [\mathbf{X}, \mathbf{y}]$ by continuity of ψ. □

Having established the injection, we now show that this mapping is a homeomorphism, i.e. that the inverse is continuous. This is formalised in the following lemma.

Lemma D.2

Consider Lemma D.1. Suppose that $\mathcal{S}'_M$ is also topologically closed in $\mathcal{S}_M$ and closed under permutations, and that ψ also satisfies (i) $\psi(\mathbf{x}, \mathbf{x}') \geq 0$, (ii) $\psi(\mathbf{x}, \mathbf{x}) = \sigma^2 > 0$, and (iii) $\psi(\mathbf{x}, \mathbf{x}') \twoheadrightarrow 0$ as $\|\mathbf{x}\| \twoheadrightarrow \infty$. Then $\mathcal{H}_M$ is closed in $\mathcal{H}^{K+1}$ and E_M^{-1} is continuous.

Remark D.1

Before moving on to the proof of Lemma D.2, we remark that Lemma D.2 would directly follow if $\mathcal{S}'_M$ were bounded: then $\mathcal{S}'_M$ is compact, so E_M is a continuous, invertible map between a compact space and a Hausdorff space, which means that E_M^{-1} must be continuous. This argument is the same to the one used in Lemma C.1. The intuition that the result must hold for unbounded $\mathcal{S}'_M$ is as follows. Since $\phi_1(\cdot) = 1$, for every $f \in \mathcal{H}_M$, f_1 is a summation of M "bumps" (imagine the EQ kernel) of the form $\psi(\cdot, \mathbf{x}_i)$ placed throughout $\mathcal{X}$. If one of these bumps goes off to infinity, then the function cannot uniformly converge pointwise, which means that the function cannot converge in $\mathcal{H}$ (if ψ is sufficiently nice). Therefore, if the function does converge in $\mathcal{H}$, $(\mathbf{x}_i)_{i=1}^M$ must be bounded, which brings us to the compact case. What makes this work is the density channel *$\phi_1(\cdot) = 1$, which forces $(\mathbf{x}_i)_{i=1}^M$ to be well behaved. The above argument is formalized in the proof of Lemma D.2.*

Proof.
Define

$$\mathcal{S}_J = ([-J, J]^d \times \mathcal{Y})^M \cap \mathcal{S}'_M, \tag{D.13}$$

which is compact in $\mathcal{A}^M$ as a closed subset of the compact set $([-J, J]^d \times \mathcal{Y})^M$. We aim to show that $\mathcal{H}_M$ is closed in $\mathcal{H}^{K+1}$ and E^{-1} is continuous. To this end, consider a convergent sequence

$$f^{(n)} = \sum_{i=1}^{M} \phi(y_i^{(n)}) \psi(\cdot, \mathbf{x}_i^{(n)}) \twoheadrightarrow f \in \mathcal{H}^{K+1}. \tag{D.14}$$

Denote $\mathbf{X}^{(n)} = (\mathbf{x}_1^{(n)}, \dots, \mathbf{x}_M^{(n)})$ and $\mathbf{y}^{(n)} = (y_1^{(n)}, \dots, y_M^{(n)})$. Claim: $(\mathbf{X}^{(n)})_{n=1}^{\infty}$ is a bounded sequence, so $(\mathbf{X}^{(n)})_{n=1}^{\infty} \subset [-J, J]^{dM}$ for J large enough, which means that $(\mathbf{X}^{(n)}, \mathbf{y}^{(n)})_{n=1}^{\infty} \subset \mathcal{S}_J$ where $\mathcal{S}_J$ is compact. Note that $[\mathcal{S}_J]$ is compact in $\mathcal{S}_M / \mathbb{S}_M$ by continuity of the canonical map.

First, we demonstrate that, assuming the claim, $\mathcal{H}_M$ is closed. Note that by boundedness of $(\mathbf{X}^{(n)}, \mathbf{y}^{(n)})_{n=1}^{\infty}$, $(f^{(n)})_{n=1}^{\infty}$ is in the image of $E_M|_{[\mathcal{S}_J]} \colon [\mathcal{S}_J] \twoheadrightarrow \mathcal{H}_M$. By continuity of $E_M|_{[\mathcal{S}_J]}$ and compactness of $[\mathcal{S}_J]$, the image of $E_M|_{[\mathcal{S}_J]}$ is compact and therefore closed, since every compact subset of a metric space is closed. Therefore, the image of $E_M|_{[\mathcal{S}_J]}$ contains the limit f. Since the image of $E_M|_{[\mathcal{S}_J]}$ is included in $\mathcal{H}_M$, we have that $f \in \mathcal{H}_M$, which shows that $\mathcal{H}_M$ is closed.

Next, we prove that, assuming the claim, E_M^{-1} is continuous. Consider $E_M|_{[\mathcal{S}_J]} \colon [\mathcal{S}_J] \twoheadrightarrow E_M([\mathcal{S}_J])$ restricted to its image. Then $(E_M|_{[\mathcal{S}_J]})^{-1}$ is continuous, because a continuous bijection from a compact space to a metric space is a homeomorphism. Therefore

$$E_M^{-1}(f^{(n)}) = (\mathbf{X}^{(n)}, \mathbf{y}^{(n)}) = (E_M|_{[\mathcal{S}_J]})^{-1}(f^{(n)}) \twoheadrightarrow (E_M|_{[\mathcal{S}_J]})^{-1}(f) = (\mathbf{X}, \mathbf{y}). \tag{D.15}$$

By continuity and invertibility of E_M, then $f^{(n)} \twoheadrightarrow E_M(\mathbf{X}, \mathbf{y})$, so $E_M(\mathbf{X}, \mathbf{y}) = f$ by uniqueness of limits. We conclude that $E_M^{-1}(f^{(n)}) \twoheadrightarrow E_M^{-1}(f)$, which means that E_M^{-1} is continuous.

It now remains to show the claim. Let f_1 denote the first element of f, i.e. the density channel. Using the reproducing property of ψ,

$$|f_1^{(n)}(\mathbf{x}) - f_1(\mathbf{x})| = |\langle \psi(\mathbf{x}, \cdot), f_1^{(n)} - f_1 \rangle| \leq \|\psi(\mathbf{x}, \cdot)\|_{\mathcal{H}} \|f_1^{(n)} - f_1\|_{\mathcal{H}} = \sigma \|f_1^{(n)} - f_1\|_{\mathcal{H}}, \tag{D.16}$$

so $f_1^{(n)} \twoheadrightarrow f_1$ in $\mathcal{H}$ means that it does so uniformly pointwise (over $\mathbf{x}$). Hence, we can let $N \in \mathcal{N}$ be such that $n \geq N$ implies that $|f_1^{(n)}(\mathbf{x}) - f_1(\mathbf{x})| < \frac{1}{3}\sigma^2$ for all $\mathbf{x}$. Let R be such that $|\psi(\mathbf{x}, \mathbf{x}_i^{(N)})| < \frac{1}{3}\sigma^2/M$ for $\|\mathbf{x}\| \geq R$ and all $i \in [M]$. Then, for $\|\mathbf{x}\| \geq R$,

$$|f_1^{(N)}(\mathbf{x})| \leq \sum_{i=1}^{M} |\psi(\mathbf{x}, \mathbf{x}_i^{(N)})| < \tfrac{1}{3}\sigma^2$$
$$\implies |f_1(\mathbf{x})| \leq |f_1^{(N)}(\mathbf{x})| + |f_1^{(N)}(\mathbf{x}) - f_1(\mathbf{x})| < \tfrac{2}{3}\sigma^2.$$

At the same time, by pointwise non-negativity of ψ, we have that

$$f_1^{(n)}(\mathbf{x}_i^{(n)}) = \sum_{j=1}^{M} \psi(\mathbf{x}_j^{(n)}, \mathbf{x}_i^{(n)}) \geq \psi(\mathbf{x}_i^{(n)}, \mathbf{x}_i^{(n)}) = \sigma^2. \tag{D.17}$$

Towards contradiction, suppose that $(\mathbf{X}^{(n)})_{n=1}^{\infty}$ is unbounded. Then $(\mathbf{x}_i^{(n)})_{n=1}^{\infty}$ is unbounded for some $i \in [M]$. Therefore, $\|\mathbf{x}_i^{(n)}\| \geq R$ for some $n \geq N$, so

$$\tfrac{2}{3}\sigma^2 > |f_1(\mathbf{x}_i^{(n)})| \geq |f_1^{(n)}(\mathbf{x}_i^{(n)})| - |f_1^{(n)}(\mathbf{x}_i^{(n)}) - f_1(\mathbf{x}_i^{(n)})| \geq \sigma^2 - \tfrac{1}{3}\sigma^2 = \tfrac{2}{3}\sigma^2, \tag{D.18}$$

which is a contradiction. □

The following lemma states that we may construct an encoding for sets containing no more than M elements into a function space, where the encoding is injective and every restriction to a fixed set size is a homeomorphism.

Lemma D.3
For every $m \in [M]$, consider a collection $\mathcal{S}'_m \subseteq \mathcal{S}_m$ that (i) has multiplicity K, (ii) is topologically closed, and (iii) is closed under permutations. Set

$$\phi : \mathcal{Y} \twoheadrightarrow \mathbb{R}^{K+1}, \quad \phi(y) = (y^0, y^1, \cdots, y^K) \tag{D.19}$$

and let ψ be an interpolating, continuous positive-definite kernel that satisfies (i) $\psi(\mathbf{x}, \mathbf{x}') \geq 0$, (ii) $\psi(\mathbf{x}, \mathbf{x}) = \sigma^2 > 0$, and (iii) $\psi(\mathbf{x}, \mathbf{x}') \twoheadrightarrow 0$ as $\|\mathbf{x}\| \twoheadrightarrow \infty$. Define

$$\mathcal{H}_m = \left\{ \sum_{i=1}^{m} \phi(y_i)\psi(\cdot, \mathbf{x}_i) : (\mathbf{x}_i, y_i)_{i=1}^{m} \subset \mathcal{S}'_m \right\} \subseteq \mathcal{H}^{K+1}, \tag{D.20}$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$-dimensional-vector–valued–function Hilbert space constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel and endowed with the inner product $\langle f, g \rangle_{\mathcal{H}^{K+1}} = \sum_{i=1}^{K+1} \langle f_i, g_i \rangle_{\mathcal{H}}$. Denote

$$[\mathcal{S}'_{\leq M}] = \bigcup_{m=1}^{M} [\mathcal{S}'_m] \quad \text{and} \quad \mathcal{H}_{\leq M} = \bigcup_{m=1}^{M} \mathcal{H}_m. \tag{D.21}$$

Then $(\mathcal{H}_m)_{m=1}^M$ are pairwise disjoint. It follows that the embedding E

$$E \colon [\mathcal{S}'_{\leq M}] \to \mathcal{H}_{\leq M}, \quad E([Z]) = E_m([Z]) \quad \text{if} \quad [Z] \in [\mathcal{S}'_m] \tag{D.22}$$

is injective, hence invertible. Denote this inverse by E^{-1}, where $E^{-1}(f) = E_m^{-1}(f)$ if $f \in \mathcal{H}_m$.

Proof.
Recall that E_m is injective for every $m \in [M]$. Hence, to demonstrate that E is injective it remains to show that $(\mathcal{H}_m)_{m=1}^M$ are pairwise disjoint. To this end, suppose that

$$\sum_{i=1}^{m} \phi(y_i)\psi(\cdot, \mathbf{x}_i) = \sum_{i=1}^{m'} \phi(y'_i)\psi(\cdot, \mathbf{x}'_i) \tag{D.23}$$

for $m \neq m'$. Then, by arguments like in the proof of Lemma D.1,

$$\sum_{i=1}^{m} \phi(y_i) = \sum_{i=1}^{m'} \phi(y'_i). \tag{D.24}$$

Since $\phi_1(\cdot) = 1$, this gives $m = m'$, which is a contradiction. Finally, by repeated application of Lemma D.2, E_m^{-1} is continuous for every $m \in [M]$.□

Lemma D.4
Let $\Phi \colon [\mathcal{S}'_{\leq M}] \to \mathcal{C}_b(\mathcal{Y}, \mathcal{Y})$ be a map from $[\mathcal{S}'_{\leq M}]$ to $\mathcal{C}_b(\mathcal{X}, \mathcal{Y})$, the space of continuous bounded functions from $\mathcal{X}$ to $\mathcal{Y}$, such that every restriction $\Phi|_{[\mathcal{S}'_m]}$ is continuous, and let E be from Lemma D.3. Then

$$\Phi \circ E^{-1} \colon \mathcal{H}_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y}) \tag{D.25}$$

is continuous.

Proof.
Recall that, due to Lemma D.1, for every $m \in [M]$, E_m^{-1} is continuous and has image $[\mathcal{S}'_m]$. By the continuity of $\Phi|_{[\mathcal{S}'_m]}$, then $\Phi|_{[\mathcal{S}'_m]} \circ E_m^{-1}$ is continuous for every $m \in [M]$. Since $\Phi \circ E^{-1}|_{\mathcal{H}_m} = \Phi|_{[\mathcal{S}'_m]} \circ E_m^{-1}$ for all $m \in [M]$, we have that $\Phi \circ E^{-1}|_{\mathcal{H}_m}$ is continuous for all $m \in [M]$. Therefore, as $\mathcal{H}_m$ is closed

in $\mathcal{H}_{\leq M}$ for every $m \in [M]$, the pasting lemma (Munkres, 1974) yields that $\Phi \circ E^{-1}$ is continuous. □

UNIVERSALITY OF CONVOLUTIONAL DEEPSET NETWORKS

From here on, we let ψ be a stationary kernel, which means that it only depends on the difference of its arguments and can be seen as a function $\mathcal{X} \to \mathbb{R}$. With the above results in place, we are finally ready to prove our central result, Theorem 3.1. Here, we provide a slightly more detailed statement of the theorem, and then proceed to prove it.

Theorem D.1
For every $m \in [M]$, consider a collection $\mathcal{S}'_m \subseteq \mathcal{S}_m$ that (i) has multiplicity K, (ii) is topologically closed, (iii) is closed under permutations, and (iv) is closed under translations. Set

$$\phi : \mathcal{Y} \to \mathbb{R}^{K+1}, \quad \phi(y) = (y^0, y^1, \cdots, y^K) \tag{D.26}$$

and let ψ be an interpolating, continuous positive-definite kernel that satisfies (i) $\psi(\mathbf{x}, \mathbf{x}') \geq 0$, (ii) $\psi(\mathbf{x}, \mathbf{x}) = \sigma^2 > 0$, and (iii) $\psi(\mathbf{x}, \mathbf{x}') \to 0$ as $\|\mathbf{x}\| \to \infty$. Define

$$\mathcal{H}_m = \left\{ \sum_{i=1}^{m} \phi(y_i)\psi(\cdot, \mathbf{x}_i) : (\mathbf{x}_i, y_i)_{i=1}^m \subset \mathcal{S}'_m \right\} \subseteq \mathcal{H}^{K+1}, \tag{D.27}$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$ dimensional-vector–valued–function Hilbert space constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel and endowed with the inner product $\langle f, g \rangle_{\mathcal{H}^{K+1}} = \sum_{i=1}^{K+1} \langle f_i, g_i \rangle_{\mathcal{H}}$. Denote

$$\mathcal{S}'_{\leq M} = \bigcup_{m=1}^{M} \mathcal{S}'_m \quad \text{and} \quad \mathcal{H}_{\leq M} = \bigcup_{m=1}^{M} \mathcal{H}_m. \tag{D.28}$$

Then a function $\Phi \colon \mathcal{S}'_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$ satisfies (i) continuity of the restriction $\Phi|_{\mathcal{S}_m}$ for every $m \in [M]$, (ii) permutation invariance (Property 2.1), and (iii) translation equivariance (Property 3.1) if and only if it has a representation of the form

$$\Phi(S) = \rho\left(E(S)\right), \qquad E((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) = \sum_{i=1}^{m} \phi(y_i)\psi(\cdot - \mathbf{x}_i) \tag{D.29}$$

where $\rho \colon \mathcal{H}_{\leq M} \to \mathcal{C}_b(\mathcal{X}, \mathcal{Y})$ is continuous and translation equivariant.

Proof (Proof of sufficiency.).
To begin with, note that permutation invariance (Property 2.1) and translation equivariance (Property 3.1) for Φ are well defined, because $\mathcal{S}'_{\leq M}$ is closed under permutations and translations by assumption. First, Φ is permutation

invariant, because addition is commutative and associative. Second, that Φ is translation equivariant (Property 3.1) follows from a direct verification and that ρ is also translation equivariant:

$$\Phi(T_{\boldsymbol{\tau}}S) = \rho\left(\sum_{i=1}^{M}\phi(y_i)\psi(\cdot-(\mathbf{x}_i+\boldsymbol{\tau}))\right) \tag{D.30}$$

$$= \rho\left(\sum_{i=1}^{M}\phi(y_i)\psi((\cdot-\boldsymbol{\tau})-\mathbf{x}_i)\right) \tag{D.31}$$

$$= \rho\left(\sum_{i=1}^{M}\phi(y_i)\psi(\cdot-\mathbf{x}_i)\right)(\cdot-\boldsymbol{\tau}) \tag{D.32}$$

$$= \Phi(S)(\cdot-\boldsymbol{\tau}) \tag{D.33}$$

$$= T'_{\boldsymbol{\tau}}\Phi(S). \tag{D.34}$$

□

Proof (Proof of necessity.).
To begin with, since Φ is permutation invariant (Property 2.1), we may define

$$\Phi\colon \bigcup_{m=1}^{M}[\mathcal{S}'_m] \to \mathcal{C}_b(\mathcal{X},\mathcal{Y}), \qquad \Phi(S) = \Phi([S]), \tag{D.35}$$

for which we verify that every restriction $\Phi|_{[\mathcal{S}'_m]}$ is continuous. By invertibility of E from Lemma D.3, we have $[S] = E^{-1}(E([S]))$. Therefore,

$$\Phi(S) = \Phi([S]) = \Phi(E^{-1}(E([S]))) = (\Phi\circ E^{-1})\left(\sum_{i=1}^{M}\phi(y_i)\psi(\cdot-\mathbf{x}_i)\right). \tag{D.36}$$

Define $\rho\colon \mathcal{H}_{\leq M} \to \mathcal{C}_b(\mathcal{X},\mathcal{Y})$ by $\rho = \Phi\circ E^{-1}$. First, ρ is continuous by Lemma D.4. Second, E^{-1} is translation equivariant, because ψ is stationary. Also, by assumption Φ is translation equivariant (Property 3.1). Thus, their composition ρ is also translation equivariant. □

Remark D.2
The function $\rho\colon \mathcal{H}_{\leq M} \to \mathcal{C}_b(\mathcal{X},\mathcal{Y})$ may be continuously extended to the entirety of $\mathcal{H}^{K+1}$ using a generalisation of the Tietze Extension Theorem by Dugundji et al. (1951). There are variants of Dugundji's Theorem that also preserve translation equivariance.

E

1D REGRESSION EXPERIMENTAL DETAILS

In this chapter, we provide details regarding the synthetic regression experiments carried out in Chapters 4 and 5.

E.1 DATA DETAILS

We begin by discussing the data generation details. For the 1D regression experiments, we consider the following generative processes:

EQ: samples from a Gaussian process with the following exponentiated-quadratic kernel:

$$k(t,t') = \exp\left(-\frac{1}{8}(t-t')^2\right);$$

Matérn–$\frac{5}{2}$: samples from a Gaussian process with the following Matérn-$\frac{5}{2}$ kernel:

$$k(t,t') = \left(1 + 4\sqrt{5}d + \frac{5}{3}d^2\right)\exp\left(-\sqrt{5}d\right)$$

with $d = 4|x - x'|$;

noisy mixture: samples from a Gaussian process with the following noisy mixture kernel:

$$k(t,t') = \exp\left(-\frac{1}{8}(t-t')^2\right) + \exp\left(-\frac{1}{2}(t-t')^2\right) + 10^{-3}\delta[t-t'];$$

weakly periodic: samples from a Gaussian process with the following weakly-periodic kernel:

$$k(t,t') = \exp\left(-\frac{1}{2}(f_1(t) - f_1(t'))^2 - \frac{1}{2}(f_2(t) - f_2(t'))^2 - \frac{1}{8}(t-t')^2\right)$$

with $f_1(t) = \cos(8\pi t)$ and $f_2(t) = \sin(8\pi t)$; and

sawtooth: samples from the following sawtooth process:

$$f(t) = \frac{A}{2} - \frac{A}{\pi}\sum_{k=1}^{K}(-1)^k \frac{\sin(2\pi k f(t-s))}{k}$$

with $A = 1$, $f \sim \mathcal{U}[3, 5]$, $s \sim \mathcal{U}[-5, 5]$, and $K \in \{10, \ldots, 20\}$ chosen uniformly.

E.2 CNN ARCHITECTURES

For the experiments in Sections 4.5.1 and 4.5.2, we consider two models: ConvCNP (which utilizes a smaller architecture), and ConvCNPXL (with a larger architecture). For all architectures, the input kernel ψ was an EQ (exponentiated quadratic) kernel with a learnable length scale parameter, as detailed in Section 4.3, as was the kernel for the final output layer ψ_ρ. When dividing by the density channel, we add $\varepsilon = 10^{-8}$ to avoid numerical issues. The length scales for the EQ kernels are initialized to twice the spacing $1/\gamma^{1/d}$ between the discretisation points $(\mathbf{t}_i)_{i=1}^T$, where γ is the density of these points and d is the dimensionality of the input space $\mathcal{X}$.

Moreover, we emphasize that the size of the receptive field is a product of the width of the CNN filters and the spacing between the discretisation points. Consequently, for a fixed width kernel of the CNN, as the number of discretisation points increases, the receptive field size decreases. One potential improvement that was not employed in our experiments, is the use of depthwise-separable convolutions (Chollet, 2017). These dramatically reduce the number of parameters in a convolutional layer, and can be used to increase the CNN filter widths, thus allowing one to increase the number of discretisation points without reducing the receptive field. The architectures for ConvCNP and ConvCNPXL are described below.

CONVCNP. For the 1d experiments, we use a simple, 4-layer convolutional architecture, with ReLU non-linearities. The kernel size of the convolutional layers was chosen to be 5, and all employed a stride of length 1 and zero padding of 2 units. The number of channels per layer was set to $[16, 32, 16, 2]$, where the final channels where then processed by the final, EQ-based layer of ρ as mean and standard deviation channels. We employ a SOFTPLUS non-linearity on the standard deviation channel to enforce positivity.

CONVCNPXL. Our large architecture takes inspiration from UNet (Ronneberger et al., 2015). We employ a 12-layer architecture with skip connections. The number of channels is doubled every layer for the first 6 layers, and halved every layer for the final 6 layers. We use concatenation for the skip connections. The following describes which layers are concatenated, where $L_i \leftarrow [L_j, L_k]$ means that the input to layer i is the concatenation of the activations of layers j and k:

- $L_8 \leftarrow [L_5, L_7]$,

- $L_9 \leftarrow [L_4, L_8]$,
- $L_{10} \leftarrow [L_3, L_9]$,
- $L_{11} \leftarrow [L_2, L_{10}]$,
- $L_{12} \leftarrow [L_1, L_{11}]$.

Like for the smaller architecture, we use ReLU non-linearities, kernels of size 5, stride 1, and zero padding for two units on all layers.

E.3 LNPF MODEL DETAILS

We describe the implementation details for the models used in the experiments in Chapters 4 and 5. For the CNPF models (CNP and AttnCNP) in Chapter 4, we follow the implementations of Garnelo et al. (2018a) and Kim et al. (2019), respectively. For the LNPF models and experiments in Chapter 5, we compare the following models, where all activation functions are leaky ReLUs with leak 0.1:

ConvCNP: The first model is the ConvCNP. The architecture of the ConvCNP is equal to that of the encoder in the ConvNP, described next.

ConvNP: The second model is the ConvNP as described in the main body. The functional embedding uses separate length scales for the data channel and density channel (Figure 5.1), which are initialized to twice the inter-point spacing of the discretization and learned during training. The discretization uniformly ranges over $[\min(x) - 1, \max(x) + 1]$ at density $\rho = 64$ points per unit, where $\min(x)$ is the minimum x value occurring in the union of the context and target sets in the current batch and $\max(x)$ is corresponding maximum x value. The discretization is passed through a 10-layer (excluding an initial and final point-wise linear layer) CNN with 64 channels and depthwise-separable convolutions. The width of the filters depends on the data set and is chosen such that the receptive field sizes are as follows:

EQ: 2,
Matérn–$\frac{5}{2}$: 2,
noisy mixture: 4,
weakly periodic: 4,
sawtooth: 16.

The discretized functional representation consists of 16 channels. The smoothing at the end of the encoder also has separate length scales for the mean and variance which are initialized similarly and learned. The encoder parametrizes the standard deviations by passing the output of the CNN through a softplus. The decoder has the same architecture as the encoder.

ANP: The third model is the Attentive NP with latent dimensionality $d = 128$ and 8-head dot-product attention (Vaswani et al., 2017). In the attentive deterministic encoder, the keys (t), queries (t), and values (concatenation of t and y) are transformed by a three-layer MLP of constant width d. The dot products are normalised by $\sqrt{d}$. The output of the attention mechanism is passed through a constant-width linear layer, which is then passed through two layers of layer normalization (Ba et al., 2016) to normalise the latent representation. In the first of these two layers, first the transformed queries are passed through a constant-width linear layer and added to the input. In the second of these two layers, the output of the first layer is first passed through a two-layer constant-width MLP and added to itself, making a residual layer. In the stochastic encoder, the inputs and outputs are concatenated and passed though a three-layer MLP of constant width d. The result is mean-pooled and passed through a two-layer constant-width MLP. The decoder consists of a three-layer MLP of constant width d.

NP: The fourth model is the original NP (Garnelo et al., 2018b). The architecture is similar to that of the ANP, where the architecture of the deterministic encoder is replaced by that of the stochastic encoder.

For all models, positivity of the observation noise is enforced with a softplus function. Parameter counts of the ConvCNP, ConvNP, ANP, and NP are listed in Table E.1.

E.4 SIM2REAL ADDITIONAL DETAILS

We describe the simulation process of training data for the experiment in Section 4.5.2. The description is borrowed from (Wilkinson, 2011).

Let X be the number of predators and Y the number of prey at any point in our simulation. According to the model, one of the following four events can occur:

A: A single predator is born according to rate $\theta_1 XY$, increasing X by one.

	EQ	Matérn–$\frac{5}{2}$	Noisy Mixt.	Weakly Per.	Sawtooth
ConvCNP	42 822	42 822	51 014	51 014	100 166
ConvNP	88 486	88 486	104 870	104 870	203 174
ANP	530 178	530 178	530 178	530 178	530 178
NP	479 874	479 874	479 874	479 874	479 874

Table E.1: Parameter counts for the ConvCNP, ConvNP, ANP, and NP in the 1D regression tasks

B: A single predator dies according to rate $\theta_2 X$, decreasing X by one.

C: A single prey is born according to rate $\theta_3 Y$, increasing Y by one.

D: A single prey dies (is eaten) according to rate $\theta_4 XY$, decreasing Y by one.

The parameter values θ_1, θ_2, θ_3, and θ_4, as well as the initial values of X and Y govern the behavior of the simulation. We choose $\theta_1 = 0.01$, $\theta_2 = 0.5$, $\theta_3 = 1$, and $\theta_4 = 0.01$, which are also used in (Papamakarios and Murray, 2016) and generate reasonable time series. Note that these are likely not the parameter values that would be estimated from the Hudson's Bay lynx–hare data set (Leigh, 1968), but they are used because they yield reasonably oscillating time series. Obtaining oscillating time series from the simulation is sensitive to the choice of parameters and many parametrizations result in populations that simply die out.

Time series are simulated using Gillespie's algorithm (Gillespie, 1977):

1. Draw the time to the next event from an exponential distribution with rate equal to the total rate $\theta_1 XY + \theta_2 X + \theta_3 Y + \theta_4 XY$.
2. Select one of the above events A, B, C, or D at random with probability proportional to its rate.
3. Adjust the appropriate population according to the selected event, and go to 1.

The simulations using these parameter settings can yield a maximum population of approximately 300 while the context set in the lynx–hare data set has an approximate maximum population of about 80 so we scaled our simulation population by a factor of $2/7$. We also remove time series which are longer than 100 units of time, which have more than 10000 events, or where one of the populations is entirely zero. The number of context points n for a training batch are each selected randomly from a uniform distribution between 3 and 80, and the number of target points is $150 - n$. These target and context points are then sampled from the simulated series. The Hudson's Bay lynx–hare data set has time values that range from 1845 to 1935. However, the values supplied to the model range from 0 to 90 to remain consistent with the simulated data.

For evaluation, an interval of 18 points is removed from the the Hudson's Bay lynx–hare data set to act as a target set, while the remaining 72 points act as the context set. This construction highlights the model's interpolation as well as its uncertainty in the presence of missing data.

Models in this setting were trained for 200 epochs with 256 batches per epoch, each batch containing 50 tasks. For this data set, we only used the ConvCNP, as we found the ConvCNPXL to overfit. The learning rate was set to 10^{-3}, and we discretize $E(\mathcal{D}_c)$ by evaluating 100 points per unit.

E.5 TRAINING AND EVALUATION DETAILS FOR GP EXPERIMENTS

For the experiments in Section 5.7.1, all LNPF models are trained with $\mathcal{L}_{\text{ML}}$ ($L = 20$) and $\mathcal{L}_{\text{NPVI}}$ ($L = 5$). For $\mathcal{L}_{\text{NPVI}}$, the context set is appended to the target set when evaluating the objective. The models are optimised using ADAM with learning rate $5 \cdot 10^{-3}$ for 100 epochs. One epoch consists of 2^{14} tasks divided into batches of size 16. For training, the inputs of the context and target sets are sampled uniformly from $[-2, 2]$. The size of the context set is sampled uniformly from $\{0, \ldots, 50\}$ and the size of the target set is fixed to 50. To encourage the LNPF models—but not the CNPF models—to fit and not revert to their conditional variants, the observation noise standard deviation σ is held fixed to 10^{-2} for the first 20 epochs.

For evaluation, the size of the context set is sampled uniformly from $\{0, \ldots, 10\}$, and the losses are evaluated with $L = 5000$ and batch size one. To test interpolation within the training range, the inputs of the context and target sets are, like training, sampled uniformly from $[-2, 2]$. To test interpolation beyond the training range, the inputs of the context and target sets are sampled uniformly from $[2, 6]$. To test extrapolation beyond the training range, the inputs of the context sets are sampled uniformly from $[-2, 2]$ and the inputs of the target sets are sampled uniformly from $[-4, -2] \cup [2, 4]$. As described in Appendix E.6, models trained with $\mathcal{L}_{\text{NPVI}}$ are evaluated using importance weighting to obtain a better estimate of the evaluation loss.

E.6 EFFECT OF NUMBER OF SAMPLES

In this section we empirically examine the effect of L, the number of samples used to estimate likelihood bounds, on the training and evaluation of ConvNPs and ANPs.

Effect of Number of Samples Used for Evaluation

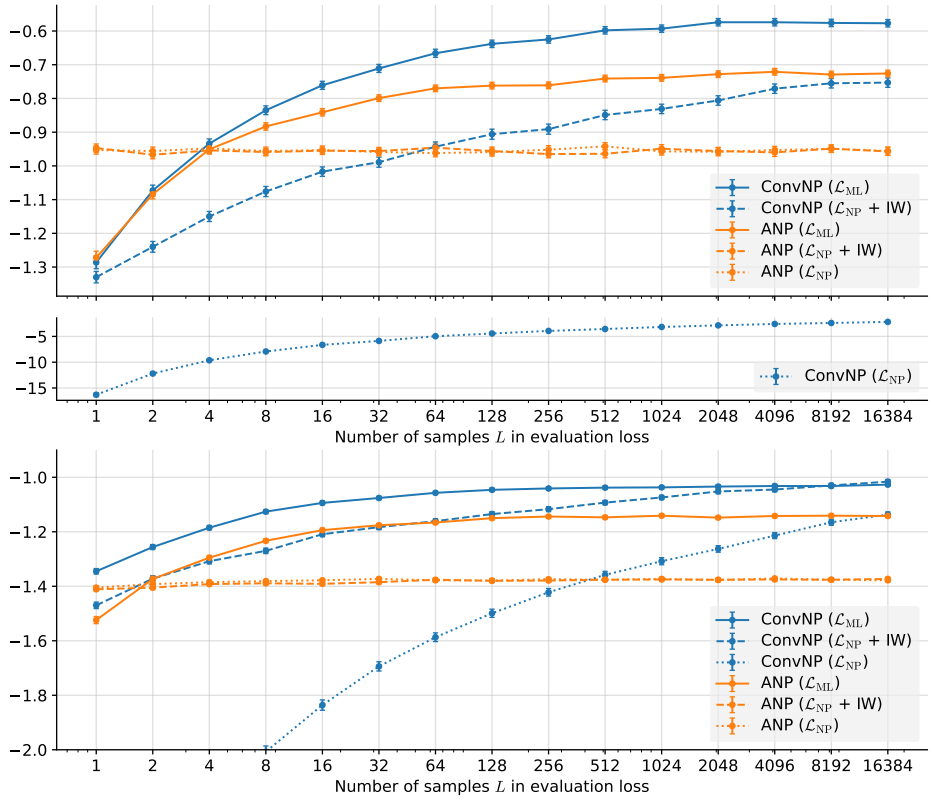


Figure E.1: Log-likelihood bounds achieved by various combination of models and training objectives when evaluated with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{IW}}$ for various numbers of samples L on (top) a Matérn-$\frac{5}{2}$ and (bottom) weakly-periodic kernel GP. Color indicates model. Solid lines correspond to models trained and evaluated with $\mathcal{L}_{\text{ML}}$. Dashed lines correspond to models trained with $\mathcal{L}_{\text{NPVI}}$ and evaluated with $\mathcal{L}_{\text{IW}}$. Dotted lines correspond to models trained with $\mathcal{L}_{\text{ML}}$ and evaluated with $\mathcal{L}_{\text{ML}}$.

As the true log-likelihoods of NP-based models are intractable, quantitative evaluation and comparison of models is challenging. Instead, we compare models by using an estimate of the log-likelihood. A natural candidate is $\mathcal{L}_{\text{ML}}$. However, unless large L is used, $\mathcal{L}_{\text{ML}}$ is conservative and tends to significantly underestimate the log-likelihood. One way to improve the estimate of $\mathcal{L}_{\text{ML}}$ is through importance weighting (IW) (Wu et al., 2016; Le et al., 2018). Denoting $D = \mathcal{D}_c \cup \mathcal{D}_t$, the encoder $\text{E}_{\boldsymbol{\phi}}(D)$ can be used as a proposal distribution:

$$\hat{\mathcal{L}}_{\text{IW}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) \coloneqq \log \left(\frac{1}{L} \sum_{l=1}^{L} \exp \left(\log w(\mathbf{z}_l) + \sum_{(\mathbf{x},y) \in D_t} \log p_{\boldsymbol{\theta}}(y|\mathbf{x}, \mathbf{z}_l) \right) \right), \quad \text{(E.1)}$$

where $\mathbf{z}_l \sim \text{E}_{\boldsymbol{\phi}}(D)$, and the importance weights are given by $\log w(\mathbf{z}_l) \coloneqq \log q_{\boldsymbol{\phi}}(\mathbf{z}|D_c) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|D)$. Here $q_{\boldsymbol{\phi}}(\mathbf{z}|D)$ is the density of the encoder distri-

bution. We find that training models with $\mathcal{L}_{\text{ML}}$ results in encoders that are ill-suited as proposal distributions, so we only use $\mathcal{L}_{\text{IW}}$ to evaluate models trained with $\mathcal{L}_{\text{NP}}$.

Figure E.1 demonstrates the effect of the number of samples L used to estimate the evaluation objective for the ConvNP and ANP trained with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NPVI}}$. The models used to generate Figure E.1 are the same models used in Section 5.7.1, i.e. having heteroskedastic noise. Observe the general trend that the log-likelihood estimates tend to increase with L, as expected. The ANP trained with $\mathcal{L}_{\text{NPVI}}$ collapsed to a conditional ANP, meaning that the encoder became deterministic; in that case, $\mathcal{L}_{\text{ML}}$ is exact, which means that larger L and importance weighting will not increase the estimate. In contrast, the ANP trained with $\mathcal{L}_{\text{ML}}$ did not collapse, and we see that there the estimate increases with L. For the ConvNP trained with $\mathcal{L}_{\text{NPVI}}$, evaluating with $\mathcal{L}_{\text{IW}}$ yields a significant increase, showing that the bound estimated with $\mathcal{L}_{\text{IW}}$ is very loose. The models trained with $\mathcal{L}_{\text{ML}}$ tend to be the best performing, although the ConvNP trained with $\mathcal{L}_{\text{NPVI}}$ is best for weakly periodic kernel and appears to still be increasing with L.

In both the main and the supplement, all log-likelihood lower bounds reported are computed with $\mathcal{L}_{\text{ML}}$ if the model was trained using $\mathcal{L}_{\text{ML}}$ and with $\mathcal{L}_{\text{IW}}$ if the model was trained using $\mathcal{L}_{\text{NPVI}}$.

Effect of Number of Samples Used During Training

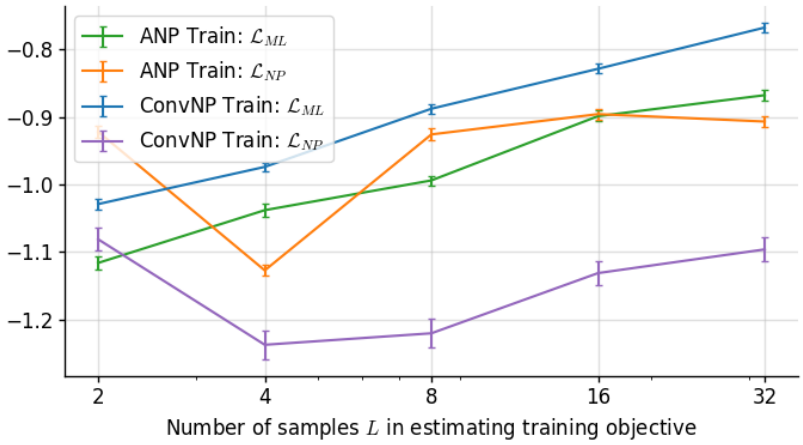


Figure E.2: Interpolation performance (within training range) for context set sizes uniformly sampled from $\{0, \ldots, 50\}$ of the ConvNP and ANP on Matérn-$\frac{5}{2}$ samples. The models are trained with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NPVI}}$ for various number of samples L. Models trained with $\mathcal{L}_{\text{ML}}$ are evaluated with $\mathcal{L}_{\text{ML}}$, while models trained with $\mathcal{L}_{\text{NPVI}}$ are evaluated with $\mathcal{L}_{\text{ML}}$. At evaluation, all bounds are estimated using 2,048 samples.

Figure E.2 shows the effect of the number of samples L in the training objectives on the performance of the ConvNP and ANP. Observe that the performance of $\mathcal{L}_{\text{ML}}$ reliably increases with the number of samples L and that $\mathcal{L}_{\text{ML}}$ outperforms $\mathcal{L}_{\text{NPVI}}$. The performance for $\mathcal{L}_{\text{NP}}$ does not appear to increase with the number of samples L and appears more noisy than $\mathcal{L}_{\text{ML}}$. Note that the models used for Figure E.2 were trained with homoskedastic observation noise. This is achieved by pooling f_σ over the time dimension.

F

IMAGE COMPLETION DATA AND EXPERIMENTAL DETAILS

In this chapter, we provide details regarding the image-completion experiments carried out in Chapters 4 and 5.

F.1 EXPERIMENTAL DETAILS

Training details.

In all experiments, we sample the number of context points uniformly from $\mathcal{U}(\frac{n_{\text{total}}}{100}, \frac{n_{\text{total}}}{2})$, and the number of target points is set to n_{total}. The context and target points are sampled randomly from each of the 16 images per batch. The weights are optimised using Adam (Kingma and Ba, 2015) with learning rate 5×10^{-4}. We use a maximum of 100 epochs, with early stopping of 15 epochs patience. All pixel values are divided by 255 to rescale them to the $[0, 1]$ range. In the following discussion, we assume that images are RGB, but very similar models can be used for greyscale images or other gridded inputs (e.g. 1d time series sampled at uniform intervals).

CNPF architectures

ATTNCNP BASELINE The AttnCNP we use corresponds to the deterministic path of the model described by (Kim et al., 2019) for image experiments. Namely, an encoder first embeds each context point c to a latent representation $(\mathbf{x}_c, \mathbf{y}_c) \mapsto \mathbf{r}_c \in \mathbb{R}^{128}$. This is achieved using a 2-hidden layer MLP of hidden dimensions 128. Every context point then goes through two stacked self-attention layers. Each self-attention layer is implemented with an 8-headed attention, a skip connection, and two layer normalizations (as described in (Parmar et al., 2018), modulo the dropout layer). To predict values at each target point $\mathbf{x}_t$, we embed $\mathbf{x}_t \mapsto \mathbf{r}_t$ and $\mathbf{x}_c \mapsto \mathbf{r}_c$ using the same single hidden layer MLP of dimensions 128. A target representation $\mathbf{r}_c$ is then estimated by applying cross-attention (using an 8-headed attention described above) with keys $\mathrm{K} \coloneqq \{\mathbf{r}_c\}_{c=1}^{C}$, values $\mathrm{V} \coloneqq \{\mathbf{r}_c\}_{c=1}^{C}$, and query $\mathbf{q} \coloneqq \mathbf{r}_t$. Given the target representation $\mathbf{r}_t$, the conditional predictive posterior is given by a Gaussian pdf with diagonal covariance parametrised by $(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_{\text{pre}_t}) = \text{decoder}(\mathbf{r}_t)$ where $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_{\text{pre}_t} \in \mathbb{R}^3$ and decoder is a 4 hidden layer

MLP with 64 hidden units per layer for the images, and the same decoder as the CNP for the 1d experiments.

Following (Le et al., 2018), we enforce we set a minimum standard deviation $\boldsymbol{\sigma}_{\text{min}} = [0.1; 0.1; 0.1]$ to avoid infinite log-likelihoods by using the following post-processed standard deviation:

$$\boldsymbol{\sigma}_t = 0.1\boldsymbol{\sigma}_{\text{min}}^{(t)} + (1 - 0.1)\log(1 + \exp(\boldsymbol{\sigma}_{\text{pre}}^{(t)})). \tag{F.1}$$

CONVCNP ARCHITECTURES Unlike ATTNCNP and off-the-grid CONVCNP, on-the-grid CONVCNP takes advantage of the gridded structure. Namely, the target and context points can be specified in terms of the image, a context mask M_c, and a target mask M_t instead of sets of input–value pairs. Although this is an equivalent formulation, it is more natural, and simpler to implement in standard deep learning libraries. In the following, we dissect the architecture and algorithmic steps succinctly summarized in Section 4.4. Note that all the convolutional layers are actually depthwise separable (Chollet, 2017); this enables a large kernel size (i.e. receptive fields) while being parameter and computationally efficient.

1. Let I denote the image. Select all context points $\text{signal} \coloneqq \mathrm{M}_c \odot \mathrm{I}$ and append a density channel $\text{density} \coloneqq \mathrm{M}_c$, which intuitively says that "there is a point at this position": $[\text{signal}, \text{density}]^\top$. Each pixel value will now have 4 channels: 3 RGB channels and 1 density channel M_c. Note that the mask will set the pixel value to 0 at a location where the density channel is 0, indicating there are no points at this position (a missing value).

2. Apply a convolution to the density channel $\text{density}' = \text{CONV}_{\boldsymbol{\theta}}(\text{density})$ and a normalized convolution to the signal $\text{signal}' \coloneqq \text{CONV}_{\boldsymbol{\theta}}(\text{signal})/\text{density}'$. The normalized convolution makes sure that the output mostly depends on the scale of the signal rather than the number of observed points. The output channel size is 128 dimensional. The kernel size of $\text{CONV}_{\boldsymbol{\theta}}$ depends on the image shape and model used (Table F.1). We also enforce element-wise positivity of the trainable filter by taking the absolute value of the kernel weights $\boldsymbol{\theta}$ before applying the convolution. As discussed in Appendix F.4, the normalization and positivity constraints do not empirically lead to improvements for on-the-grid data. Note that in this setting, $E(\mathcal{D}_c)$ is $[\text{signal}', \text{density}']^\top$.

3. We now describe the on-the-grid version of $\rho(\cdot)$, which we decompose into two stages. In the first stage, we apply a CNN to $[\text{signal}', \text{density}']^\top$. This CNN is composed of residual blocks (He et al., 2016), each consisting of 1 or 2 (Table F.1) convolutional layers with ReLU activations and no batch normalization. The number of output channels in each layer is 128. The

kernel size is the same across the whole network, but depends on the image shape and model used (Table F.1).

4. In the second stage of $\rho(\cdot)$, we apply a shared pointwise MLP : $\mathbb{R}^{128} \to \mathbb{R}^{2C}$ (we use the same architecture as used for the AttnCNP decoder) to the output of the first stage at each pixel location in the target set. Here C denotes the number of channels in the image. The first C outputs of the MLP are treated as the means of a Gaussian predictive distribution, and the last C outputs are treated as the standard deviations. These then pass through a positivity-enforcing function (e.g., softplus).

Table F.1: CNN architectures for image-completion experiments.

Model	Input Shape	$\text{conv}_{\boldsymbol{\theta}}$			
Kernel Size	CNN				
Kernel Size	CNN Num.				
Res. Blocks per Block	Conv. Layers				
ConvCNP	< 50 pixels	9	5	4	1
	> 50 pixels	7	3	4	1
ConvCNP XL	any	9	11	6	2

LNPF ARCHITECTURES AND DETAILS

GENERAL ARCHITECTURE DETAILS For all models, we follow Le et al. (2018) and process the predicted standard deviation of the latent function $\boldsymbol{\sigma}_z$ using a sigmoid and the standard deviation $\boldsymbol{\sigma}$ of the predictive distribution using lower-bounded softplus:

$$\boldsymbol{\sigma}_z = 0.001 + (1 - 0.001)\frac{1}{1 + \exp(f_{\sigma,z})}, \tag{F.2}$$

$$\boldsymbol{\sigma} = 0.001 + (1 - 0.001)\ln(1 + \exp(f_\sigma)). \tag{F.3}$$

As the pixels are rescaled to $[0, 1]$, we also process the mean of the posterior predictive (conditioned on a single sample) to be in $[0, 1]$ using a logistic function

$$\boldsymbol{\mu} = \frac{1}{1 + \exp(-f_\mu)}. \tag{F.4}$$

In the following, we describe the architecture of ANP and ConvNP. Unless stated otherwise, all vectors in the following paragraphs are in $\mathbb{R}^{128}$ and all MLPs have 128 hidden units.

ANP DETAILS As the ANP cannot take advantage of the fact that images are on the grid, we preprocess each pixel so that $\mathbf{x} \in [-1, 1]^2$. The only exception being for the test set of ZSMM, where $\mathbf{x} \in [-\frac{56}{32}, \frac{56}{32}]^2$ as the model is trained on 32×32 but evaluated on 56×56 images. Each context feature is first encoded $\mathbf{x}^{(c)} \mapsto \mathbf{r}_{\mathbf{x}_c}$ by a single hidden layer MLP, while a second single hidden layer MLP encodes values $\mathbf{y}_c \mapsto \mathbf{r}_{\mathbf{y}_c}$. We produce a representation $\mathbf{r}_c$ by summing both representations $\mathbf{r}_{\mathbf{x}_c} + \mathbf{r}_{\mathbf{y}_c}$ and passing them through two self-attention layers (Vaswani et al., 2017). Following Parmar et al. (2018), each self-attention layer is implemented as 8-headed attention, a skip connection, and two layer normalizations (Ba et al., 2016). To predict values at each target point t, we embed $\mathbf{x}_t \mapsto \mathbf{r}_{\mathbf{x}_t}$ using the hidden layer MLP used for $\mathbf{r}_{\mathbf{x}_c}$. A deterministic target representation $\mathbf{r}_t$ is then computed by applying cross-attention (using an 8-headed attention described above) as for the ATTNCNP. For the latent path, we average over context representations $\mathbf{r}_c$, and pass the resulting representation through a single hidden layer MLP that outputs $(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) \in \mathbb{R}^{256}$. $\boldsymbol{\sigma}_z$ is made positive by post-processing it using Equation (F.2). We then sample (with reparametrization) L latent representation $\mathbf{z}_l \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$.

We describe the remainder of the forward pass for a single $\mathbf{z}_l$, though in practice multiple samples may be processed in parallel. The deterministic and latent representations of the context set are concatenated, and the resulting representation is passed through a linear layer $[\mathbf{r}_t; \mathbf{z}_l] \twoheadrightarrow \mathbf{r}'_t \in \mathbb{R}^{128}$. Given the target and context-set representations, the predictive posterior is given by a Gaussian pdf with diagonal covariance parametrised by $(\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}^{(t)}_{\text{pre}}) = \text{decoder}([\mathbf{r}^{(t)}_x; \mathbf{r}^{(t)}_{xyz}])$ where $\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}^{(t)}_{\text{pre}} \in \mathbb{R}^3$ and decoder is a 4 hidden layer MLP. Finally, the $\boldsymbol{\sigma}^{(t)}$ is processed by Equation (F.3) using Equation (F.4). In the case of MNIST and ZSMM, $\boldsymbol{\sigma}^{(t)}$ is also spatially mean pooled, which corresponds to using homoskedastic noise. This improves the qualitative performance by forcing ANP and CONVNP to model the digit instead of focusing on predicting the black background with high confidence. Kim et al. (2019) did not suffer from that issue as they used a much larger lower bound for Equation (F.3).

CONVNP DETAILS The core algorithm of on-the-grid ConvNP is outlined in Algorithm 5 as well as Algorithm 3. Here we discuss the parametrizations used for each step of the algorithm. All convolutional layers are depthwise separable (Chollet, 2017). $\text{CONV}_{\boldsymbol{\theta}}$ is a convolutional layer with kernel size of 11 (no bias). We enforce positivity on the weights in the first convolutional layer by only convolving their absolute value with the signal.

The CNNs are ResNets (He et al., 2016) with 9 blocks, where each convolution has a kernel size of 3. Each residual block consists of two convolutional layers, pre-activation batch normalization layers, and ReLU activations. The output of the pre-latent CNN (CNN in Algorithm 5) goes through a single hidden layer MLP

that outputs $(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) \in \mathbb{R}^{256}$. As with ANP, $f_{\sigma,z}$ is processed by Equation (F.2) and then used to sample (with reparametrization) L latent functions $\mathbf{z}_l$. Importantly, we found that the coherence of samples improves if the model uses a *global representation* in addition to the the pixel dependent representation. We achieve this by mean-pooling half of the functional representation. Namely, we replace $\mathbf{z}_l$ by the channel-wise concatenation of $\mathbf{z}_l^{(1:64)}$ and $\text{MEAN}(\mathbf{z}_l^{(65:128)})$, where the mean is taken over the spatial dimensions. This latent function then goes through the post-latent CNN (CNN in Algorithm 5), as well as a linear layer to output $(f_\mu, f_\sigma) \in \mathbb{R}^{256}$. As for ANP f_μ is processed by Equation (F.4) and f_σ is re-scaled with Equation (F.3) and is spatially pooled in the case of MNIST and ZSMM to obtain homoskedastic noise.

F.2 ZERO SHOT MULTI MNIST (ZSMM) DATA

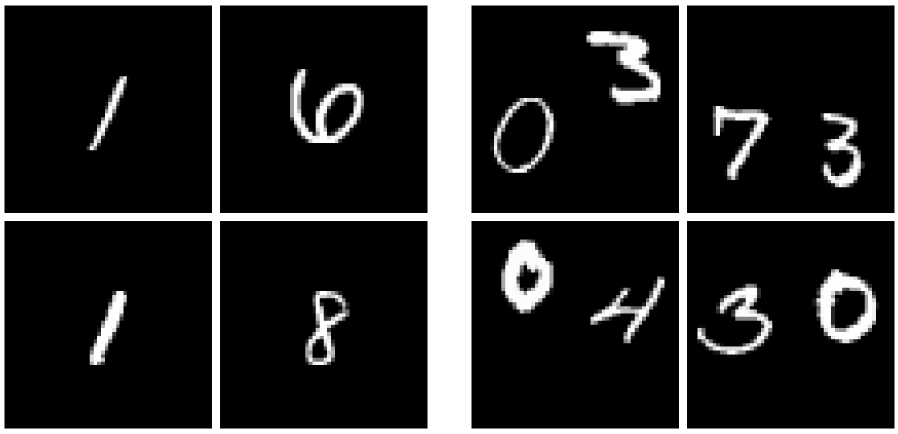


Figure F.1: Samples from our generated Zero Shot Multi MNIST (ZSMM) data set.

In the real world, it is very common to have multiple objects in our field of view which do not interact with each other. Yet, many image data sets in machine learning contain only a single, well-centered object. To evaluate the translation equivariance and generalization capabilities of our model, we introduce the zero-shot multi-MNIST setting.

The training set contains all 60000 28×28 MNIST training digits centered on a black 56×56 background. (Figure F.1 (left)). For the test set, we randomly sample with replacement 10000 pairs of digits from the MNIST test set, place them on a black 56×56 background, and translate the digits in such a way that the digits can be arbitrarily close but cannot overlap (Figure F.1 (right)). Importantly, the scale of the digits and the image size are the same during training and testing.

F.3 ATTNCNP AND CONVCNP QUALITATIVE COMPARISON

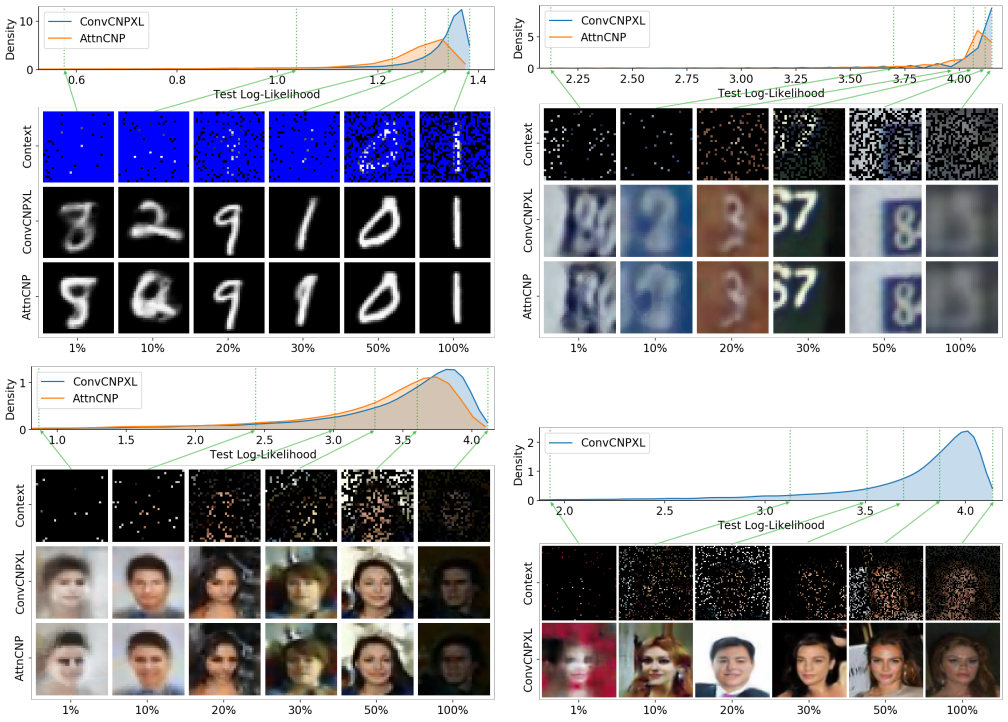


Figure F.2: Log-likelihood and qualitative comparisons between AttnCNP and ConvCNP on four standard benchmarks (from left to right, top to bottom: MNIST, SVHN, CelebA32, CelebA64). The top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), ConvCNP target predictions (middle), and AttnCNP target predictions (bottom). Each column corresponds to a given percentile of the ConvCNP distribution. AttnCNP could not be trained on CelebA64 due to its memory inefficiency.

Figure F.2 shows the test log-likelihood distributions of an AttnCNP and ConvCNP model as well as some qualitative comparisons between the two. Although most mean predictions of both models look relatively similar for SVHN and CelebA32, the real advantage of ConvCNP becomes apparent when testing the generalization capacity of both models. Figure F.3 shows ConvCNP and AttnCNP trained on CelebA32 and tested on a downscaled version of Ellen's famous Oscar selfie. We see that ConvCNP generalizes better in this setting. [1]

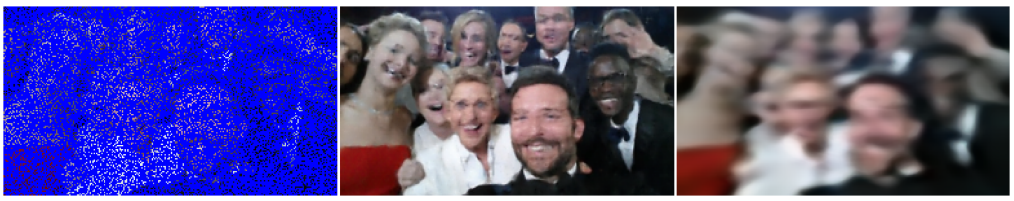


Figure F.3: Qualitative evaluation of a ConvCNP (center) and AttnCNP (right) trained on CelebA32 and tested on a downscaled version (146×259) of Ellen's Oscar selfie with 20% of the pixels as context (left).

Table F.2: Log-likelihood from image ablation experiments (6 runs).

Model	MNIST	SVHN	CelebA32	CelebA64	ZSMM
ConvCNP	1.19 ±0.01	3.89 ±0.01	3.19 ±0.02	3.64 ±0.01	1.21 ±0.00
. . . no density	1.15 ±0.01	3.88 ±0.01	3.15 ±0.02	3.62 ±0.01	1.13 ±0.08
. . . no norm.	1.19 ±0.01	3.86 ±0.03	3.16 ±0.03	3.62 ±0.01	1.20 ±0.01
. . . no abs.	1.15 ±0.02	3.83 ±0.02	3.08 ±0.03	3.56 ±0.01	1.15 ±0.01
. . . no abs. norm.	1.19 ±0.01	3.86 ±0.03	3.16 ±0.03	3.62 ±0.01	1.20 ±0.01
. . . EQ	1.18 ±0.00	3.89 ±0.01	3.18 ±0.02	3.63 ±0.01	1.21 ±0.00

F.4 ABLATION STUDY: FIRST LAYER

To understand the importance of the different components of the first layer, we performed an ablation study by removing the density normalization (ConvCNP no norm.), removing the density channel (ConvCNP no dens.), removing the positivity constraints (ConvCNP no abs.), removing the positivity constraints and the normalization (ConvCNP no abs. norm.), and replacing the fully trainable first layer by an EQ kernel similar to the continuous case (ConvCNP EQ). Table F.2 demonstrates the following: (i) Appending a density channel helps. (ii) Enforcing the positivity constraint is only important when using a normalized convolution. (iii) Using a less expressive EQ filter does not significantly decrease performance, suggesting that the model might be learning similar filters (Appendix F.5).

F.5 QUALITATIVE ANALYSIS OF THE FIRST FILTER

As discussed in Appendix F.4, using a less expressive EQ filter does not significantly decrease performance. Figure F.4 shows that this happens because the fully trainable kernel learns to approximate the EQ filter.

1 The reconstruction looks worse than Figure 4.6 (left) despite the larger context set, because the test image has been downscaled and the models are trained on a low resolution CelebA32. These constraints come from AttnCNP's large memory footprint.

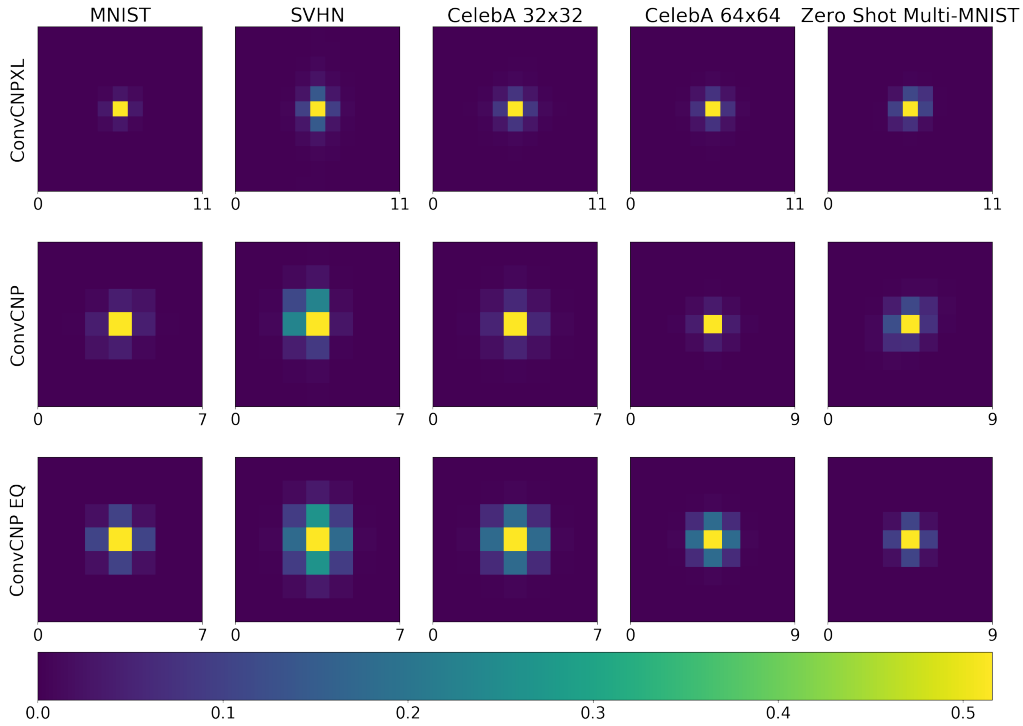


Figure F.4: First filter learned by ConvCNPXL, ConvCNP, and ConvCNP EQ for all our datasets. In the case of RGB images, the plotted filters are for the first channel (red). Note that not all filters are of the same size.

F.6 EFFECT OF RECEPTIVE FIELD ON TRANSLATION EQUIVARIANCE

As seen in Table 4.2, a ConvCNPXL with large receptive field performs significantly worse on the ZSMM task than ConvCNP, which has a smaller receptive field. Figure F.5 shows a more detailed comparison of the models, and suggests that ConvCNPXL learns to model non-stationary behaviour, namely that digits in the training set are centred. We hypothesize that this issue stems from the the treatment of the image boundaries. Indeed, if the receptive field is large enough and the padding values are significantly different than the inputs to each convolutional layer, the model can learn position-dependent behaviour by "looking" at the distance from the padded boundaries.

For ZSMM, Figure F.6 suggests that "circular" padding, where the padding is implied by tiling the image, helps prevent the model from learning non-stationarities, even as the size of the receptive field becomes larger. We hypothesize that this is due to the fact that "circularly" padded values are harder to distinguish from actual values than zeros. We have not tested the effect of padding on other datasets, and note that "circular" padding could result in other issues.

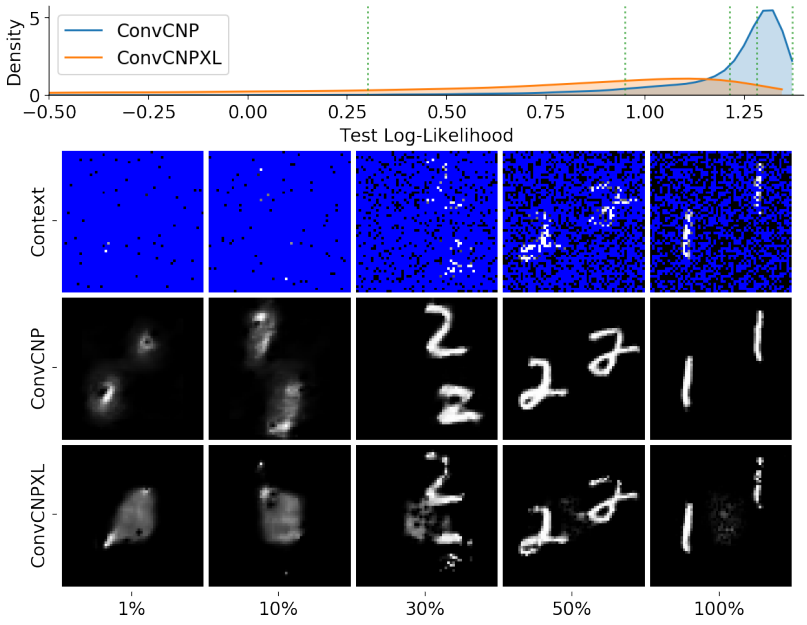


Figure F.5: Log-likelihood and qualitative results on ZSMM. The top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), ConvCNP target predictions (middle), and ConvCNPXL target predictions (bottom). Each column corresponds to a given percentile of the ConvCNP distribution.

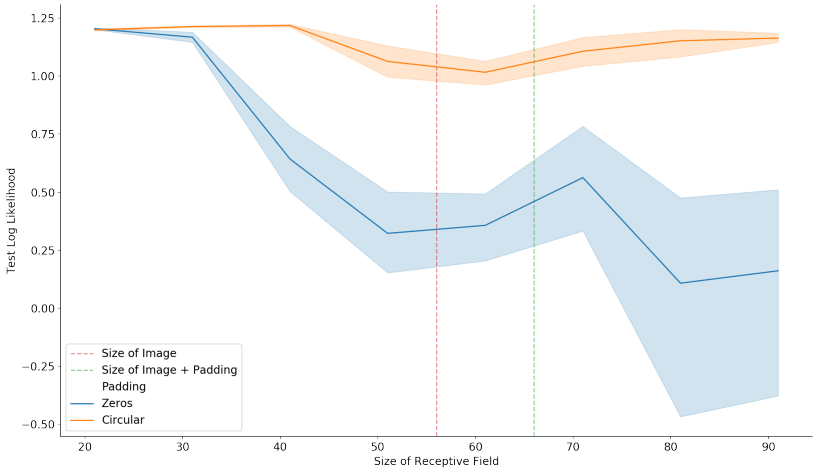


Figure F.6: Effect of the receptive field size on ZSMM's log-likelihood. The line plot shows the mean and standard deviation over 6 runs. The blue curve corresponds to a model with zero padding, while the orange one corresponds to "circular" padding.

F.7 ADDITIONAL RESULTS FOR LNPF IMAGE-COMPLETION

We provide additional qualitative samples and quantitative analyses for the ConvNP and ANP.

Additional ConvNP Samples

Figure F.7 provides further samples from a ConvNP trained with $\mathcal{L}_{\mathrm{ML}}$. We observe that the ConvNP produces reasonably diverse yet coherent samples when evaluated in a regime that resembles the training regime (in the first four sub-columns of MNIST, SVHN, and CelebA). However, Figure F.7 also demonstrates that the ConvNP struggles with context sets that are significantly different from those seen during training.

Further comparisons of ANP and ConvNP

We provide further qualitative comparisons of ConvNPs, ANPs trained with $\mathcal{L}_{\mathrm{ML}}$, and ANPs trained with $\mathcal{L}_{\mathrm{NPVI}}$. We omit ConvNPs trained with $\mathcal{L}_{\mathrm{NP}}$ as these are significantly outperformed by ConvNPs trained with $\mathcal{L}_{\mathrm{ML}}$ (see e.g. Table 5.2).

Figure F.8 demonstrates that all models perform relatively well when context sets are drawn from a similar distribution as employed during training (first four sub-columns of MNIST, SVHN, and CelebA). Furthermore, we observe that samples from the ConvNP prior tend to be closer to samples from the underlying data distribution (e.g. for CelebA).

The qualitative advantage of the ConvNP is most significant in settings that require translation equivariance for generalization. Figure F.8 row 2 (ZSMM) clearly demonstrates that ConvNP generalizes to larger canvas sizes and multiple digits, while ANP attempts to reconstruct a single digit regardless of the context set. Finally, Figure F.9 provides the test log-likelihood distributions of ANP and ConvNP as well as some qualitative comparisons between the two.

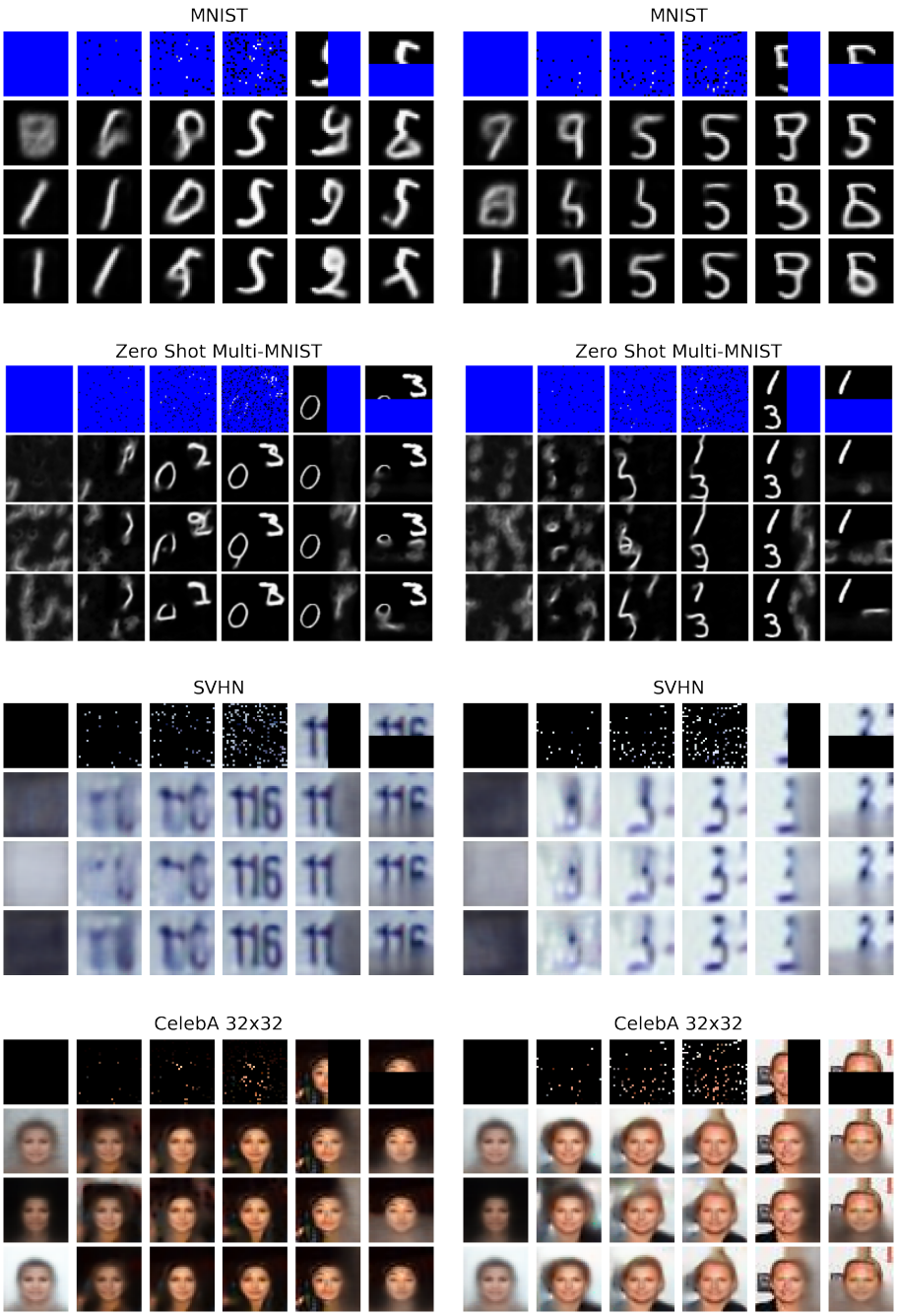


Figure F.7: Qualitative samples for a ConvNP trained with $\mathcal{L}_{\text{ML}}$ in Table 5.2. From top to bottom the four major rows correspond to MNIST, ZSMM, SVHN, CelebA32 datasets. For each dataset and each of the two major columns, a different image is randomly sampled; the first sub-row shows the given context points (missing pixels are in blue for MNIST and ZSMM but in black for SVHN and CelebA), while the next three sub-rows show the mean of the posterior predictive corresponding to different samples of the latent function. To show diverse samples we select three samples that maximize the average Euclidean distance between pixels of the samples. From left to right the first four sub-columns correspond to a context set with 0%, 1%, 3%, 10% randomly sampled context points. In the last two sub-columns, the context sets respectively contain all the pixels in the left and top half of the image.

Figure F.8: Qualitative samples for (left) ConvNP trained with $\mathcal{L}_{\mathrm{ML}}$; (centre) ANP trained with $\mathcal{L}_{\mathrm{ML}}$; and (right) ANP trained with $\mathcal{L}_{\mathrm{NPVI}}$. For each model the figure shows the same format as Figure F.7.

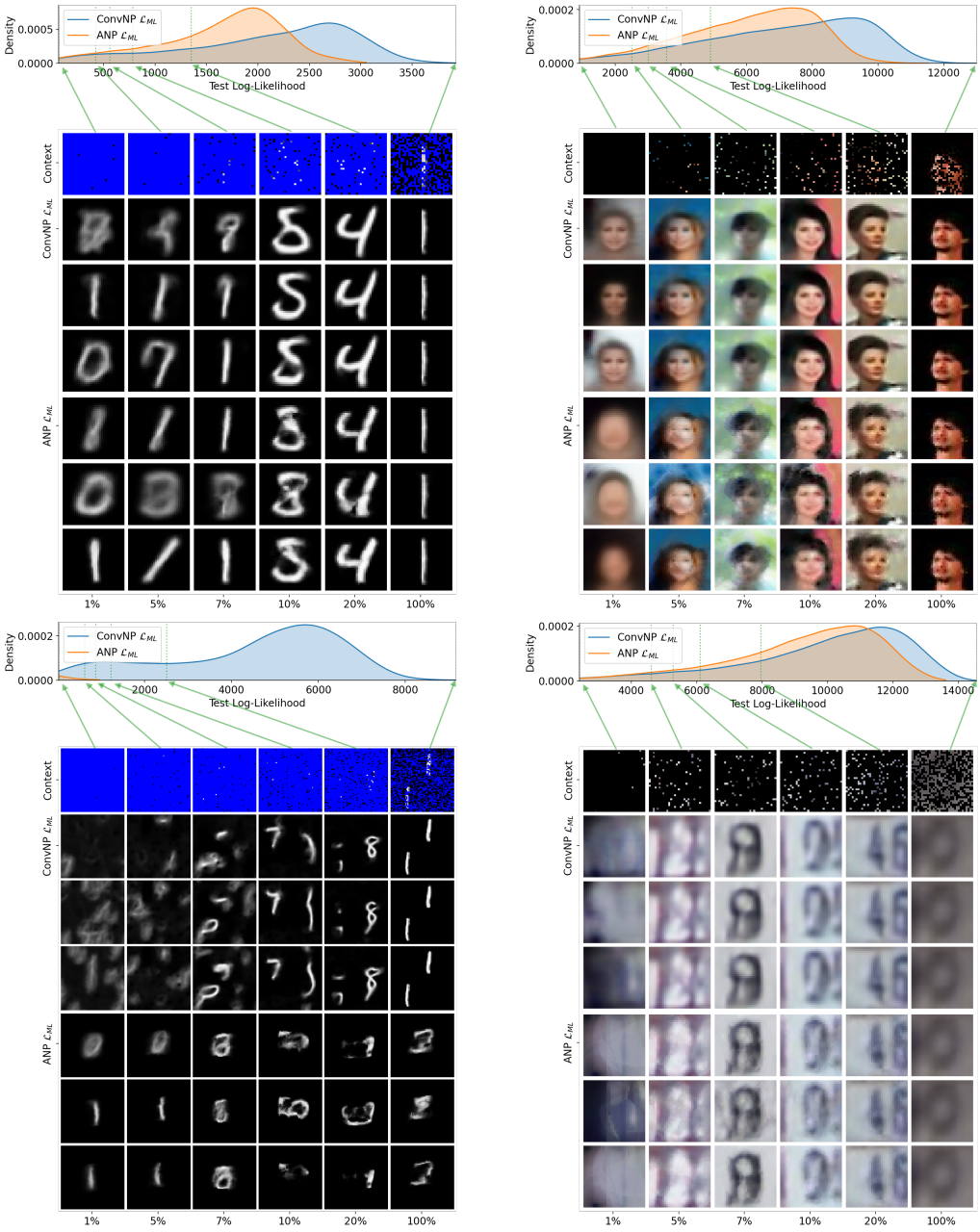


Figure F.9: Log-likelihood and qualitative samples comparing ConvNP and ANP trained with $\mathcal{L}_{\text{ML}}$ on (top-left) MNIST; (top-right) CelebA; (bottom-left) ZSMM; (bottom-right) SVHN. For each sub-figure, the top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), followed by three samples form ConvNP (mean of the posterior predictive corresponding to different samples from the latent function), and three samples from ANP. Each column corresponds to a given percentile of the ConvNP test log likelihood (as shown by green arrows).

G

ENVIRONMENTAL DATA EXPERIMENTAL DETAILS

We provide details regarding the data, models, and experimental protocols used to conduct the experiments with the ERA5-Land data in Section 5.7.3.

G.1 DATA DETAILS

Table G.1: Coordinates for boxes defining the train and test regions. Latitidues are given as (north, south), and longitudes as (west, east).

	Central (train)	Western (test)	Eastern (test)	Southern (test)
Latitudes	$(52, 46)$	$(50, 46)$	$(52, 49)$	$(46, 42)$
Longitudes	$(08, 28)$	$(01, 08)$	$(28, 35)$	$(19, 26)$

ERA5-Land (Balsamo et al., 2015) contains high resolution information on environmental variables at a 9 km spacing across the globe.[1] The data we use contains daily measurements of accumulated precipitation at 11pm and temperature at 11pm at every location, between 1981 and 2020, yielding a total of 14,304 temporal measurements across the spatial grid. In addition, we provide orography (elevation) values for each location. We normalize the data such that the precipitation values in the train set have zero mean and unit standard deviation.

We consider the task of predicting daily precipitation y, with latitude and longitude as $\mathbf{x}$. In addition, at each context and target location, we provide the model with access to side information in the form of orography (elevation) and temperature values. We also normalize the orography and temperature values to have zero mean and unit standard deviation. We choose a large region of central Europe as our train set, and use regions East, West and South of the train set as held out test sets (see Figure G.1 and Table G.1). At train time, to sample a task, we first sample a random date between 1981 and 2020. We then sample a square subregion of grid of values from within the train region (which has size 61×201). We consider two models, one trained on 28×28 subregions, and another trained on 40×40 subregions. During training, each subregion is then split into context

1 URL: https://www.ecmwf.int/en/era5-land. Neither the European Commission nor ECMWF is responsible for any use that may be made of the Copernicus Information or data it contains.

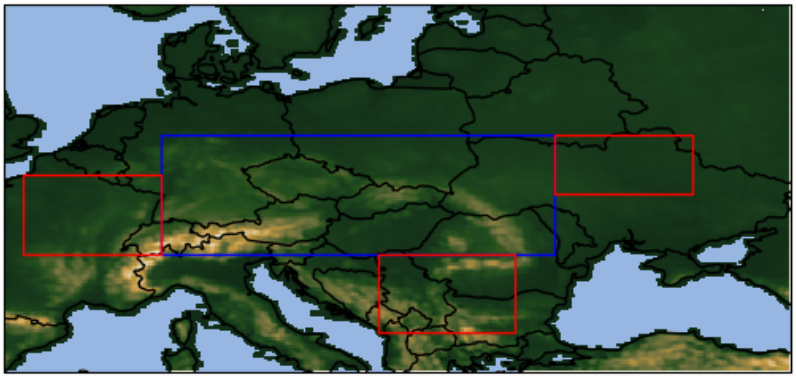


Figure G.1: Training (blue) and test (red) regions in Europe, along with orography data from ERA5Land.

and target sets. Context points are randomly chosen with a keep rate p_{keep} with $p_{\text{keep}} \sim \mathcal{U}[0, 0.3]$. In this section, we train only on the $\mathcal{L}_{\text{ML}}$ objective.

Gaussian Process Baseline

We mean-centre the data for each task for the GP before training, and add the mean offset back for evaluation and sampling. We use an Automatic Relevance Determination (ARD) kernel, with separate factors for latitude/longitude, temperature and orography. In detail, let $\mathbf{x} = (x_{\text{lat}}, x_{\text{lon}})$ denote position, and let ω, t denote orography and precipitation respectively, and let $\mathbf{r} := (\mathbf{x}, \omega, t)$. Then the kernel is given by

$$k(\mathbf{r}, \mathbf{r}') = \sigma_v^2 k_l(\mathbf{x}, \mathbf{x}') k_\omega(\omega, \omega') k_t(t, t') + \sigma_n^2 \delta(\mathbf{r}, \mathbf{r}').$$

Here each of k_l, k_ω and k_t are Matérn-$\frac{5}{2}$ kernels with separate learnable length-scales; $\delta(\mathbf{r}, \mathbf{r}') = 1$ if $\mathbf{r} = \mathbf{r}'$ and 0 otherwise; and σ_v^2, σ_n^2 are learnable signal and noise variances respectively. We learn all hyperparameters by maximising the log-marginal likelihood using Scipy's implementation of L-BFGS (Liu and Nocedal, 1989).

TRANSFORMING THE DATA As the data is non-negative, we considered applying the transform $y \mapsto \log(\epsilon + y)$ for the GP to model. If $\epsilon = 0$, this would guarantee that the GP would only yield positive samples, which would be physically sensible as precipitation is non-negative. However, this cannot be done as precipitation often takes the value $y = 0$, which would lead to the transform being undefined. On the other hand, if $\epsilon > 0$, the GP samples after performing the inverse transform could still predict a precipitation value as low as $-\epsilon$, which is still unphysical. Further, a small value of ϵ leads to large distortion of the y values in transformed space. In

the end, we run all experiments for the GP and NP without log-transforming the data; hence the models have to learn non-negativity.

ConvNP Architecture and Training Details

As the ERA5-Land dataset is regularly spaced, we use the on-the-grid version of the architecture, without the need for an RBF smoothing layer at the input (see Section 5.5). All experiments used a convolutional architecture with 3 residual blocks (He et al., 2016) for the encoder and 3 residual blocks for the decoder. Each residual block is defined with two layers of ReLU activations followed by convolutions, each with kernel size 5. The first convolution in each block is a standard convolution layer, whereas the second is depthwise separable (Chollet, 2017). All intermediate convolutional layers have 128 channels, and the latent function $\mathbf{z}$ has 16 channels. The networks were trained using ADAM (Kingma and Ba, 2015) with a learning rate of 10^{-4}. We used 16 channels for the latent function $\mathbf{z}$, and estimated $\mathcal{L}_{\mathrm{ML}}$ using 16-32 samples at train time, with batches of 8-16 images.

We train the models for between 400 and 500 epochs, where each epoch is defined as a single pass through each day in the training set, where at each day, a random subregion of the full 61×201 central Europe region is cropped. We estimated the predictive density using 2500 samples of $\mathbf{z}$ during test time.

Prediction and Sampling

To create Table 5.3, at test time we sample 28×28 subregions from each of the train and test regions. This is done 1000 times. For the GP, we randomly restart optimisation 5 times per task and use the best hyper-parameters found. In order to remove outliers where the GP has very poor likelihood, we set a log-likelihood threshold for the GP. If the GP has a log-likelihood of less than 0 nats on a particular task, then that task is removed from the evaluation.

We find that to produce high quality samples, we need to train the model on subregions that are roughly as large as the lengthscale of the precipitation process. Hence we sample from the model trained on 40×40 subregions in Figure 5.4 in the main body. We show samples from the model trained on both 28×28 subregions and 40×40 subregions in Appendix G.2. We also compare to samples from GPs trained on each context set (no random restarts were used for sampling).

Bayesian Optimization

We use the same models as described above, trained on random 28×28 subregions of the train region, and compare to the GP baselines described above. For the Bayesian optimization experiments in Figure 5.5 in the main body, we do not perform random restarts as this was too time-consuming. We carry out the Bayesian optimization (BayesOpt) experiments in each of the four regions: Central (train), West (test), East (test), and South (test). Each Bayesian optimization "episode" is defined by randomly sub-sampling a day (uniformly at random between 1981 and 2020), then sampling a sub-region from the tested region. To test the models' spatial generalization capacity (where possible), we sub-sample episodes from each of the four regions with the following sizes: (i) Central: 42x42, (ii) West: 40x40, (iii) East: 28x28, and (iv) South: 36x36.

Episodes begin from empty sets $\mathcal{D}_c^{(0)} =$, and models sequentially query locations for $t = 1, \ldots, 50$. Denoting $(\mathbf{x}^{(t)}, y^{(t)})$ the query location and queried value at iteration t, the context set is then updated as $\mathcal{D}_c^{(t)} = \mathcal{D}_c^{(t-1)} \cup \{(\mathbf{x}^{(t)}, y^{(t)})\}$. Denoting $\mathbf{y}$ as the complete set of rainfall values in the sub-region, and $\mathbf{y}^{(t)}$ as the set of queried values at iteration t, we can define the *instantaneous regret* as $r_t = \max(\mathbf{y}) - \max(\mathbf{y}_c^{(t)})$, and compute the average regret (plotted in Figure 5.5 in the main text) at the t^{th} iteration as $\bar{r}_t = \frac{1}{t} \sum_{i=1}^{t} r_i$.

G.2 ADDITIONAL FIGURES FOR ENVIRONMENTAL DATA

Predictive density

Figure G.2 displays the predictive densities for precipitation at different locations, conditioned on a context set used for testing. The density of the ConvNP is estimated using 2500 samples of $\mathbf{z}$. To examine why the ConvNP outperforms the GP in terms of log-likelihood, we plot cases where the ConvNP likelihood is significantly better than the GP likelihood. We see that this is due to the GP occasionally making very overconfident predictions compared to the ConvNP. We also see that the ConvNP in a small proportion of cases exhibits very non-Gaussian, asymmetric predictive distribtuions.

Additional Samples

In this section we show additional samples from the model trained on 28×28 images (Figures G.3 and G.4) and also on 40×40 images (Figures G.5 and G.6). Training on larger images reduces the occurence of blocky artefacts. Figure 5.4 in

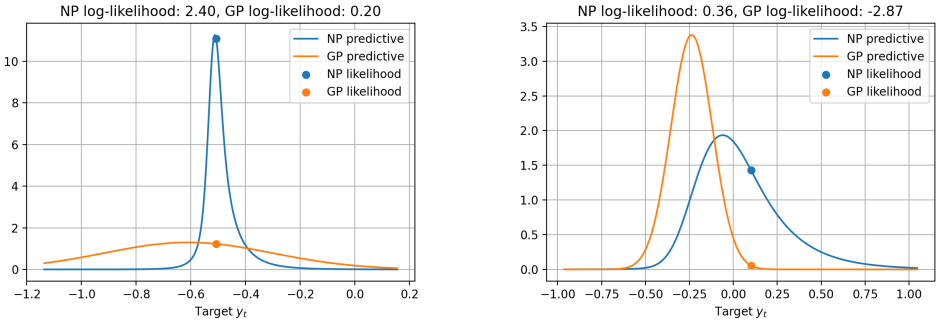


Figure G.2: Predictive density at two target points, in a task where the ConvNP significantly outperforms the GP. The orange and blue circles show the likelihood of the ground truth target value under the GP and ConvNP. Note that as the precipitation values are normalized to zero mean and unit standard deviation, $y_t = -0.53$ corresponds to no rain. The left subplot shows that the ConvNP sometimes produces predictions heavily centered on this value, showing it has learned the sparsity of precipitation values. In the right subplot we see the ConvNP predictive distribution is sometimes asymmetric with a heavier positive tail, reflecting the non-negativity of precipitation.

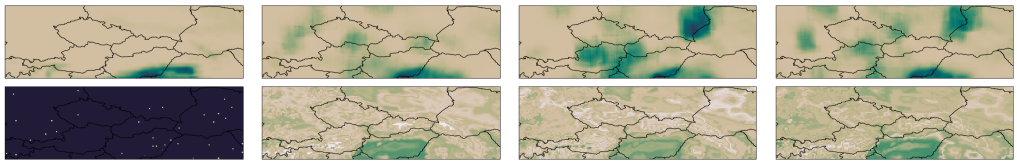


Figure G.3: Samples from the predictive processes overlaid on central Europe, for a model trained on random 28×28 subregions of the full 61×201 central Europe region. Layout is identical to Figure 5.4. Note some blocky artefacts in the ConvNP samples due to training on small subregions. Here the GP has overfit to the orography data, with samples that resemble the orography rather than precipitation.

the main body was trained on 40×40 images. Note that samples shown here are 61×201, i.e. the size of the entire central Europe train region.

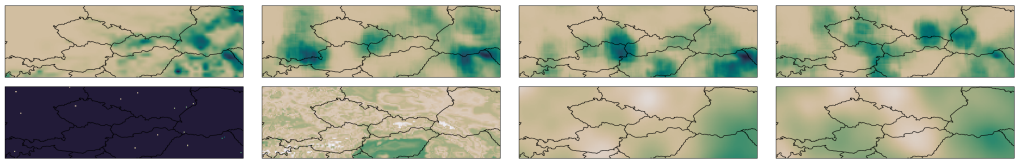


Figure G.4: Samples from the predictive processes overlaid on central Europe, for a model trained on random 28×28 subregions of the full 61×201 central Europe region. Layout is identical to Figure 5.4. Here the GP has learned a lengthscale that is too large.

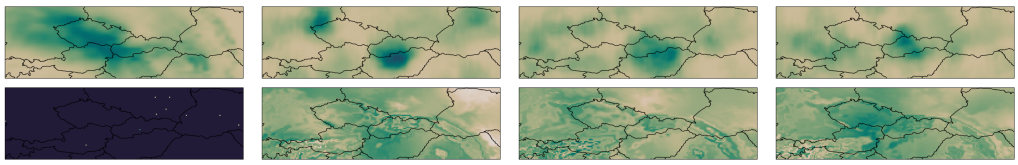


Figure G.5: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region. Layout is identical to Figure 5.4. Here the GP has overfit to the orography data, with samples that resemble the orography rather than precipitation.

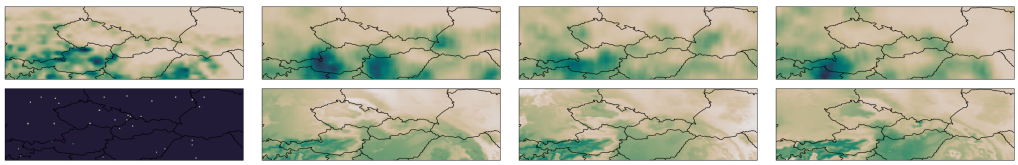


Figure G.6: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region. Layout is identical to Figure 5.4. The GP has again overfit to the orography data.

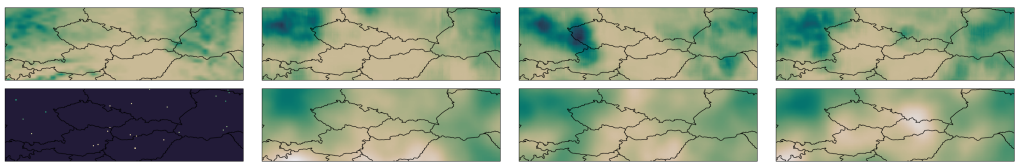


Figure G.7: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region. Layout is identical to Figure 5.4.

H

CNAPS EXPERIMENTAL DETAILS

H.1 EXPERIMENTAL DETAILS AND PROTOCOLS

All experiments were implemented in PyTorch (Paszke et al., 2019) and executed either on NVIDIA Tesla P100-PCIE-16GB or Tesla V100-SXM2-16GB GPUs. The full CNAPs model runs in a distributed fashion across 2 GPUs and takes approximately one and a half days to complete episodic training and testing.

META-DATASET Training and Evaluation Procedure

Feature Extractor Weights Pretraining

We first reduce the size of the images in the ImageNet ILSVRC-2012 dataset (Krizhevsky et al., 2012) to 84×84 pixels. Some images in the ImageNet ILSVRC-2012 dataset are duplicates of images in other datasets included in META-DATASET, so these are removed. We then split the 1000 training classes of the ImageNet ILSVRC-2012 dataset into training, validation, and test sets according to the criteria detailed in (Triantafillou et al., 2020). The test set consists of the 130 leaf-node subclasses of the "device" synset node, the validation set consists of the the 158 leaf-node subclasses of the "carnivore" synset node, and the training set consists of the remaining 712 leaf-node classes. We then pretrain a feature extractor with parameters $\boldsymbol{\theta}$ based on a modified ResNet-18 (He et al., 2016) architecture on the above 712 training classes. The ResNet-18 architecture is detailed in Table H.5. Compared to a standard ResNet-18, we reduced the initial convolution kernel size from 7 to 5 and eliminated the initial max-pool step. These changes were made to accommodate the reduced size of the imagenet training images. We train for 125 epochs using stochastic gradient descent with momentum of 0.9, weight decay equal to 0.0001, a batch size of 256, and an initial learning rate of 0.1 that decreases by a factor of 10 every 25 epochs.

During pretraining, the training dataset was augmented with random crops, random horizontal flips, and random color jitter. The top-1 accuracy after pretraining was 63.9%. For all subsequent training and evaluation steps, the ResNet-18 weights were frozen. The dimensionality of the feature extractor output is $d_f = 512$. The hyper-parameters used were derived from the PyTorch (Paszke et al., 2019) ResNet training tutorial. The only tuning that was performed was on the number of epochs used for training and the interval at which the learning rate was de-

Table H.1: Datasets used to train, validate, and test models. Corresponds to the procedures laid out by Triantafillou et al. (2020) in handling META-DATASET.

ImageNet ILSVRC-2012			All Datasets		
Train	**Validation**	**Test**	**Train**	**Validation**	**Test**
ILSVRC	ILSVRC	ILSVRC	ILSVRC	ILSVRC	ILSVRC
		Omniglot	Omniglot	Omniglot	Omniglot
		Aircraft	Aircraft	Aircraft	Aircraft
		Birds	Birds	Birds	Birds
		Textures	Textures	Textures	Textures
		Quick Draw	Quick Draw	Quick Draw	Quick Draw
		Fungi	Fungi	Fungi	Fungi
		VGG Flower	VGG Flower	VGG Flower	VGG Flower
		MSCOCO		MSCOCO	MSCOCO
		Traffic Signs			Traffic Signs
		MNIST			MNIST
		CIFAR10			CIFAR10
		CIFAR100			CIFAR100

creased. For the number of epochs, we tried both 90 and 125 epochs and selected 125, which resulted in slightly higher accuracy. We also found that dropping the learning rate at an interval of 25 versus 30 epochs resulted in slightly higher accuracy.

Episodic Training of the Adaptation Networks

Next we train the functions that generate the parameters $\boldsymbol{\psi}_f^\tau$, $\boldsymbol{\psi}_w^\tau$ for the feature extractor adapters and the linear classifier, respectively. We train two variants of CNAPs (on ImageNet ILSVRC-2012 only and all datasets – see Table H.1). We generate training and validation episodes using the reader from the repository provided by Triantafillou et al. (2020). We train in an end-to-end fashion for 110,000 episodes with the Adam (Kingma and Ba, 2015) optimizer, using a batch size of 16 episodes, and a fixed learning rate of 0.0005. We validate using 200 episodes per validation dataset. Note that when training on ILSVRC only, we validate on ILSVRC only, however, when training on all datasets, we validate on all datasets that have validation data (see Table H.1) and consider a model to be better if more than half of the datasets have a higher classification accuracy than the current best model.

No data augmentation was employed during the training of $\boldsymbol{\phi}$. Note that while training $\boldsymbol{\phi}$ the feature extractor $f_{\boldsymbol{\theta}}(\cdot)$ is in 'eval' mode (i.e. it will use the fixed batch normalization statistics learned during pretraining the feature extractor weights $\boldsymbol{\theta}$ with a moving average). No batch normalization is used in any of the functions generating the $\boldsymbol{\psi}^\tau$ parameters, with the exception of the set encoder g (that generates the global task representation $\mathbf{z}_G^\tau$). Note that the target points are never passed through the set encoder g. Again, very little hyper-parameter tuning was performed. No grid search or other hyper-parameter search was used.

Table H.2: Few-shot classification results on META-DATASET (Triantafillou et al., 2020) using models trained on all training datasets for Parallel Residual Adapters (Rebuffi et al., 2018) and CNAPs. All figures are percentages and the ± sign indicates the 95% confidence interval over tasks. Bold text indicates the scores within the confidence interval of the highest score. Tasks from datasets below the dashed line were not used for training.

Dataset	Parallel Residual Adapter	CNAPs
ILSVRC	**51.2 ± 1.0**	**52.3 ± 1.0**
Omniglot	**87.3 ± 0.7**	**88.4 ± 0.7**
Aircraft	78.3 ± 0.7	**80.5 ± 0.6**
Birds	67.8 ± 0.9	**72.2 ± 0.9**
Textures	55.5 ± 0.7	**58.3 ± 0.7**
Quick Draw	70.9 ± 0.7	**72.5 ± 0.8**
Fungi	44.6 ± 1.1	**47.4 ± 1.0**
VGG Flower	81.7 ± 0.7	**86.0 ± 0.5**
Traffic Signs	57.2 ± 0.9	**60.2 ± 0.9**
MSCOCO	**43.7 ± 1.0**	**42.6 ± 1.1**
MNIST	91.1 ± 0.4	**92.7 ± 0.4**
CIFAR10	**64.5 ± 0.8**	61.5 ± 0.7
CIFAR100	**50.4 ± 0.9**	**50.1 ± 1.0**

For learning rate we tried both 0.0001 and 0.0005, and selected the latter. We experimented with the number of training episodes in the range of 80,000 to 140,000, with 110,000 episodes generally yielding the best results. We also tried lowering the batch size to 8, but that led to decreased accuracy.

Evaluation

We generate test episodes using the reader provided by Triantafillou et al. (2020). We test all models with 600 episodes each on all test datasets. The classification accuracy is averaged over the episodes and a 95% confidence interval is computed. We compare the best validation and fully trained models in terms of accuracy and use the best of the two. Note that during evaluation, the feature extractor $f_{\boldsymbol{\theta}}(\cdot)$ is also in 'eval' mode.

H.1.1 *Comparison Between CNAPs and Parallel Residual Adapters*

We ablate the parametrisation of our task level parameters $\boldsymbol{\psi}_f$ (using FiLM layers), by comparing to a CNAPs model that instead makes use of *parallel residual adapters* (Rebuffi et al., 2018) to parametrise $\boldsymbol{\psi}_f$. Note that parallel residual adapters add 1×1 convolutions in parallel with each convolution layer. Thus, if the number of feature channels is C, then the number of parameters required for each convolutional layer in the feature extractor is $2C$ for FiLM layers and C^2 for

parallel residual adapters. Hence, parallel residual adapters have $C/2$ times the capacity compared to FiLM layers.

Thus, the ablation study serves two important purposes: (i) It verifies the usefulness of the particular parametrisation employed by CNAPs (i.e. FiLM layers), and (ii) it studies whether the improved performance of the auto-regressive version can be attributed to increased capacity alone: if this is indeed the case, we should see similar improvements by increasing the capacity by alternative means, e.g. increasing the number of adapted parameters.

Despite this advantage, CNAPs achieves superior performance when using FiLM layers, as can be seen in Table H.2. This provides important evidence that one must carefully consider the tradeoff between the number of adapted parameters as well as their *role in the feature extractor* when designing models for few-shot classification.

H.2 NETWORK ARCHITECTURE DETAILS

H.2.1 *ResNet18 Architecture details*

Throughout our experiments in Section 6.6, we use a ResNet18 (He et al., 2016) as our feature extractor, the parameters of which we denote $\boldsymbol{\theta}$. Table H.3 and Table H.4 detail the architectures of the basic block (left) and basic scaling block (right) that are the fundamental components of the ResNet that we employ. Table H.5 details how these blocks are composed to generate the overall feature extractor network. We use the implementation that is provided by the PyTorch (Paszke et al., 2019),[1] though we adapt the code to enable the use of FiLM layers.

Table H.3: ResNet-18 basic block b.

Layers
Input
Conv2d (3×3, stride 1, pad 1)
BatchNorm
FiLM ($\boldsymbol{\gamma}_{b,1}, \boldsymbol{\beta}_{b,1}$)
ReLU
Conv2d (3×3, stride 1, pad 1)
BatchNorm
FiLM ($\boldsymbol{\gamma}_{b,2}, \boldsymbol{\beta}_{b,2}$)
Sum with Input
ReLU

Table H.4: ResNet-18 basic scaling block b.

Layers
Input
Conv2d (3×3, stride 2, pad 1)
BatchNorm
FiLM ($\boldsymbol{\gamma}_{b,1}, \boldsymbol{\beta}_{b,1}$)
ReLU
Conv2d (3×3, stride 1, pad 1)
BatchNorm
FiLM ($\boldsymbol{\gamma}_{b,2}, \boldsymbol{\beta}_{b,2}$)
Downsample Input by factor of 2
Sum with Downsampled Input
ReLU

1 https://pytorch.org/docs/stable/torchvision/models.html

Table H.5: ResNet-18 feature extractor network.

ResNet-18 Feature Extractor (θ) with FiLM Layers: $\mathbf{x} \rightarrow f_{\boldsymbol{\theta}}(\mathbf{x}; \boldsymbol{\psi}_f^\tau)$, $\mathbf{x}^* \rightarrow f_{\boldsymbol{\theta}}(\mathbf{x}^*; \boldsymbol{\psi}_f^\tau)$

Stage	Output size	Layers
Input	$84 \times 84 \times 3$	Input image
Pre-processing	$41 \times 41 \times 64$	Conv2d (5×5, stride 2, pad 1, BatchNorm, ReLU)
Layer 1	$41 \times 41 \times 64$	Basic Block $\times$ 2
Layer 2	$21 \times 21 \times 128$	Basic Block, Basic Scaling Block
Layer 3	$11 \times 11 \times 256$	Basic Block, Basic Scaling Block
Layer 4	$6 \times 6 \times 512$	Basic Block, Basic Scaling Block
Post-Processing	512	AvgPool, Flatten

H.2.2 *Adaptation Network Architecture Details*

We provide the details of the architectures used for our adaptation networks. Table H.6 details the architecture of the set encoder $g : D^\tau \mapsto \mathbf{z}_\text{G}$ that maps context sets to global representations.

Table H.6: Set encoder g.

Set Encoder (g): $\mathbf{x} \rightarrow \mathbf{z}_G^\tau$

Output size	Layers
$84 \times 84 \times 3$	Input image
$42 \times 42 \times 64$	Conv2d (3×3, stride 1, pad 1, ReLU), MaxPool (2×2, stride 2)
$21 \times 21 \times 64$	Conv2d (3×3, stride 1, pad 1, ReLU), MaxPool (2×2, stride 2)
$10 \times 10 \times 64$	Conv2d (3×3, stride 1, pad 1, ReLU), MaxPool (2×2, stride 2)
$5 \times 5 \times 64$	Conv2d (3×3, stride 1, pad 1, ReLU), MaxPool (2×2, stride 2)
$2 \times 2 \times 64$	Conv2d (3×3, stride 1, pad 1, ReLU), MaxPool (2×2, stride 2)
64	AdaptiveAvgPool2d

Table H.7 details the architecture used in the auto-regressive parameterization of $\mathbf{z}_\text{AR}$. In our experiments, there is one such network for every block in the ResNet18 (detailed in Table H.5). These networks accept as input the set of activations from the previous block, and map them (through the permutation invariant structure) to a vector representation of the output of the layer. The representation $\mathbf{z}_i = (\mathbf{z}_\text{G}, \mathbf{z}_\text{AR})$ is then generated by concatenating the global and auto-regressive representations, and fed into the adaptation network that provides the FiLM layer parameters for the next layer. This network is detailed in Table H.8, and illustrated in Figure 6.5. Note that, as depicted in Figure 6.5, each layer has four networks with architectures as detailed in Table H.8, one for each $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, for each convolutional layer in the block.

Table H.7: Network of set encoder ϕ_f (FC stands for fully connected).

Set Encoder (ϕ_f): $\{f_{\boldsymbol{\theta}}^{l_i}(x;\boldsymbol{\psi}_f^\tau)\} \rightarrow \mathbf{z}_{\text{AR}}^i$

Output size	Layers
l_i channels $\times$ l_i channel size	Input $\{f_{\boldsymbol{\theta}}^{l_i}(x;\boldsymbol{\psi}_f^\tau)\}$
l_i channels $\times$ l_i channel size	AvgPool, Flatten
l_i channels	FC, ReLU
l_i channels	2 $\times$ FC with residual connection, ReLU
l_i channels	FC with residual skip connection
l_i channels	mean pooling over instances
l_i channels	Input from mean pooling
l_i channels	FC, ReLU

Table H.8: Network ϕ_f.

Network (ϕ_f): $(\mathbf{z}_{\text{G}}, \mathbf{z}_{\text{AR}}) \rightarrow (\boldsymbol{\gamma}, \boldsymbol{\beta})$

Output size	Layers
$64 + l_i$ channels	Input from Concatenate
l_i channels	FC, ReLU
l_i channels	2 $\times$ FC with residual skip connection, ReLU
l_i channels	FC with residual skip connection

H.2.3 *Linear Classifier Adaptation Network*

Finally, we provide details for the linear classifer $\boldsymbol{\psi}_w^\tau$, and the adaptation network that provides these task-specific parameters $\boldsymbol{\psi}_w(\cdot)$. The adaptation network accepts a class-specific representation that is generated by applying a mean-pooling operation to the adapted feature activations of each instance associated with the class in the context set: $\mathbf{z}_c^\tau = \frac{1}{N_c^\tau} \sum_{\mathbf{x} \in \mathcal{D}_c^\tau} f_{\boldsymbol{\theta}}(\mathbf{x};\boldsymbol{\psi}_f^\tau)$, where N_c^τ denotes the number of context instances associated with class c in task τ. The parametrisation of $\boldsymbol{\psi}_w$ comprises of two separate networks (one for the weights $\boldsymbol{\psi}_w$ and one for the biases $\boldsymbol{\psi}_b$) detailed in Table H.9 and Table H.10, respectively. The resulting weights and biases (for each class in task τ) can then be used as a linear classification layer, as detailed in Table H.11.

Table H.9: Network ϕ_w.

Network (ϕ_w): $\mathbf{z}_c \rightarrow \boldsymbol{\psi}_{w,w}$

Output size	Layers
512	Input from mean pooling
512	2 $\times$ fully connected, ELU
512	fully connected
512	Sum with Input

Table H.10: Network ϕ_b.

Network (ϕ_b): $\mathbf{z}_c \rightarrow \boldsymbol{\psi}_{w,b}$

Output size	Layers
512	Input from mean pooling
512	2 $\times$ fully connected, ELU
1	fully connected

Table H.11: Linear classifier network.

Linear Classifier ($\boldsymbol{\psi}_w$): $f_{\boldsymbol{\theta}}(\mathbf{x}^*; \boldsymbol{\psi}_f^\tau) \rightarrow p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\psi}^\tau(D^\tau), \boldsymbol{\theta})$

Output size	**Layers**
512	Input features $f_{\boldsymbol{\theta}}(\mathbf{x}^*; \boldsymbol{\psi}_f^\tau)$
$512 \times C^\tau$	Input weights w
512×1	Input biases b
C^τ	fully connected
C^τ	softmax