First Year Report

Stratis Markou

Department of Engineering
University of Cambridge

Supervisor: Prof. Carl E. Rasmussen
Advisor: Prof. Richard E. Turner

In partial fulfillment of the probationary requirements for the degree of
Doctor of Philosophy in Engineering

Christ's College August 2021

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this report 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. Chapter 3 of this report, titled *Gaussian Neural Processes*, contains the results of work which was published in Markou et al. (2021), and was work which was conducted in collaboration with James Requeima, Wessel Bruinsma and Prof. Richard Turner. Other than this chapter, this work is entirely my own and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This report contains fewer than 15,000 words including appendices, bibliography, footnotes, tables and equations.

Stratis Markou
August 2021

Acknowledgements

First, I would like to thank my supervisor Prof. Carl Rasmussen and advisor Prof. Richard Turner whose guidance and insights have greatly helped my work during the first year of my PhD. I would also like to thank my collaborators James Requeima and Wessel Bruinsma, who have played a big part in realising the work in this report, especially those presented in the third chapter. Their helpful insights, patience and openness to discussion have made them a pleasure to work with, and I hope we can continue our collaboration in the future.

Abstract

Learning efficiently from multiple datasets is a recently emerged learning setting, presenting exciting opportunities and important challenges. In this report, we discuss meta-algorithms from a probabilistic perspective, using a common viewpoint from which several meta-algorithms can be presented jointly. We place emphasis on the computational and memory efficiency of meta-algorithms. After reviewing some well known meta-algorithms, we focus on the Neural Process Family (NPF) of models.

Conditional Neural Processes (CNP; Garnelo et al., 2018a) are a recently proposed attractive meta-model. CNPs enable fast inference at test time, can be trained via an exact log-likelihood procedure, but are limited by the fact that they do not model statistical output dependencies. This hurts their predictive performance, and makes it impossible to draw coherent function samples, rendering CNPs inapplicable in settings where downstream estimators require function samples. Neural Processes (NPs; Garnelo et al., 2018b) attempt to alleviate this issue by using latent variables, however the latter introduce difficulties stemming from approximate inference. Another alternative (Bruinsma et al., 2021), here referred to as the FullConvGNP, models output dependencies while still being trainable via an exact maximum-likelihood. Unfortunately, the FullConvGNP relies on expensive $2D_x$-dimensional convolutions, which limit its applicability to only one-dimensional data. We present feature-based Gaussian Neural Processes (GNPs), which provide an alternative way to model output dependencies, and are also trainable via the maximum-likelihood, but can be scaled to higher-dimensional inputs. The proposed models exhibit good performance in synthetic experiments. We suggest methods for extending GNPs to non-Gaussian data using marginal transformations. We also identify an inherent limitation of some feature-based models, whereby the model's prior corresponds to constant functions, and propose a method to alleviate this pathology.

Focusing further on CNPs and GNPs, we argue that *equivariance* (Cohen and Welling, 2016a) is a highly useful inductive bias in several practical meta-learning settings.

Unfortunately, existing methods for building equivariant CNPs (Gordon et al., 2020; Holderrieth et al., 2021; Kawano et al., 2021) rely on expensive convolutions, which require memory and compute which increase exponentially with the input dimension. To address this issue, we introduce a method for parameterising translation equivariant (TE), as well as translation and rotoreflection equivariant (TRE) models, based on transformations of their inputs. We present necessary and sufficient conditions for a model to be TE or TRE, providing a method for designing equivariant models which do not rely on convolutions, and are thus considerably cheaper to scale to higher input dimensions. The feasibility of these methods is yet to be empirically assessed, however preliminary experiments have yielded encouraging results.

The report concludes with a summary of the work presented here, a discussion of future directions of research and a tentative timeline for executing these, while progressing through the postgraduate programme.

Table of contents

1 Introduction **1**

1.1 Motivating Meta-Learning 1

1.2 Meta-learning and multi-task learning 4

1.3 Limitations and opportunities of meta-learning 6

1.4 Outline and contributions 8

2 Background **9**

2.1 Introducing the meta-learning problem 9

2.2 Meta-Learning as a probabilistic problem 10

2.2.1 A multi-task probabilistic model 11

2.2.2 Challenges in Bayesian meta-learning 13

2.2.3 Interpreting meta-algorithms as approximation schemes 15

2.3 Conditional and Latent Neural Processes 16

2.3.1 Prediction maps & context and target sets 17

2.3.2 Conditional Neural Processes 18

2.3.3 Convolutional Conditional Neural Processes 19

2.3.4 Neural Processes 21

2.3.5 Convolutional Neural Processes 22

2.3.6 Equivariance 23

2.4 Chapter summary 25

3 Gaussian Neural Processes **27**

3.1 Modelling output dependencies 27

3.2 Feature-based covariance parameterisations 31

3.3 Gaussian Neural Processes on toy data 32

3.4 Modelling non-Gaussian joint distributions 35

3.5 Limitations of feature-based prediction maps 37

3.6 Addressing limitations of feature-based maps 39

3.7 Chapter summary 40

4 Equivariant Neural Processes 41

4.1 Generalisation and Equivariance 41

4.2 Equivariant CNPs without the Convolutions 43

4.3 Chapter summary 46

5 Conclusion, Future Work & Timeline 47

5.1 Conclusion 47

5.2 Future work 48

5.3 Execution timeline 50

References 53

Appendix A Lemmas on equivariant maps 57

Chapter 1

Introduction

1.1 Motivating Meta-Learning

One central endeavour of Machine Learning is designing systems which can efficiently discover patterns in data and use these patterns to inform and improve decision making. With increasing amounts of data becoming available, a new learning regime has recently emerged, presenting new opportunities and challenges. In several applications, we find ourselves with access to not a single, but several related datasets. More specifically, we often have access to a number of similar datasets which, despite representing different modelling tasks, exhibit common patterns. One way to model multiple datasets is to run a supervised algorithm on each one in isolation, as illustrated in fig. 1.1. However, repeating a learning algorithm from scratch on each dataset seems rather wasteful, and does not leverage information which may be mutually useful for modelling different datasets. A natural question is whether we can leverage patterns shared across datasets, to improve the performance and efficiency of our automated systems.

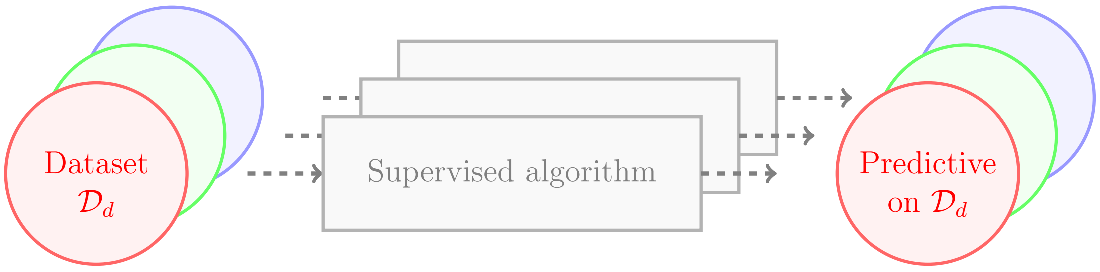

Figure 1.1: When multiple similar datasets are available, we can model them by running a supervised algorithm on each one in isolation. However, this is wasteful from a computational and memory standpoint and also hinders predictive performance.

As a running example for a multi-dataset setting, let us consider environmental modelling, a field where numerous datasets are available (Hersbach et al., 2018a), and new ones are steadily being generated. These datasets often consist of weather observations in different regions of the Earth over different periods of time. Given several similar such datasets, we wish to model them not only effectively, in terms of predictive performance, but also efficiently, in terms of computational and memory costs. Although a collection of datasets could be handled by training a different model on each dataset separately, this would fall short in two important ways, impacting both efficiency and performance.

First, treating datasets independently may incur computational and memory costs which could be avoided if we were to treat them jointly. For example, consider a weather modelling problem where we have access to two datasets containing observations in two countries with similar weather, such as the UK and the Netherlands, for which we wish to make predictions. Suppose that in order to solve this problem, we choose to train one model for each country in isolation. Training and storing two separate models would incur roughly double the computational and memory costs over maintaining a single model for one of the tasks, even though both models may have learnt to recognise roughly similar weather patterns. For instance, the models may need to learn which features are good predictors for the onset of rainfall. Both models would have to learn these features from scratch and in isolation. Similarly, after training, both models would need to be stored, even though the features they have learnt may be highly similar and thus redundant. In this sense, training and storing a separate model for each dataset seems inefficient, not only from a computational, but from a memory perspective also. Furthermore, if a new dataset becomes available, a supervised algorithm would have to be executed anew, in order to make predictions on it. This situation is exacerbated by the fact that modern machine learning systems (Devlin et al., 2018; Krizhevsky et al., 2012; Vaswani et al., 2017) have both large memory footprints and long computation times.

Second, treating each of the datasets separately fails to leverage shared patterns which may exist between them. If we assume that all the datasets in our collection originate from a common underlying generative processes, then patterns that are present in one dataset may be informative about patterns in another dataset. In environmental modelling for example, we know that even though the weather at different regions of the Earth may exhibit significant differences, all weather phenomena originate from a common generative process. This generative process is governed by the same laws everywhere on Earth, that is the physical laws which describe fluid flows, heat transfer and related phenomena. Therefore, even though a dataset pertaining to one region of the Earth, for example Southeast Asia, may be on average substantially different to that at another region, for example Central America, we expect that some patterns within one region may be informative about patterns in the other region. Having observed a storm in Southeast Asia for example, may be informative about the behaviour of a storm in the Gulf of Mexico. Modelling similar datasets in a manner which can leverage shared information may improve performance, while failing to do so would leave this increased modelling capacity untapped.

Meta-learning (Hospedales et al., 2020) is a relatively recent learning framework which takes a step towards addressing both these issues. Whilst a range of different meta-algorithms exist, they are characterised by the following features. Instead of treating the datasets independently, a meta-algorithm either explicitly or implicitly assumes them to originate from a common generative process. By modelling the datasets jointly rather than independently, meta-algorithms can leverage any information that is shared between them. In addition, meta-algorithms are typically designed such that they can rapidly adapt to unseen datasets, via an *adaptation mechanism*. By inferring task-specific parameters in a computationally cheap manner, meta-algorithms can be easily applied to unseen datasets, without requiring the expensive retraining of a new model from scratch. Furthermore, since a single meta-model is sufficient to make predictions over any number of datasets, meta-learning circumvents the need to store a separate model for each task, thereby also reducing the overall memory footprint.

In addition to the aforementioned computational considerations, meta-learning can also be motivated by considering the data efficiency of human learners when generalising across tasks (Bronskill, 2020). Having solved a handful of related tasks, humans can rapidly infer how to solve variants of these tasks, requiring only a modest number of training examples (Lake et al., 2015). Systems which are able to perform this type of learning, referred to as *few-shot learning*, are highly appealing since they can be deployed on tasks where few data are available. Meta-learning provides a general solution to few-shot learning settings, because the joint modelling of multiple tasks allows a meta-learner to draw on the information contained in similar datasets it has observed in the past, when solving a new task.

Having motivated meta-learning from the point of view of efficiency and predictive performance, we take a moment to distinguish it from the similar framework of multi-task learning.

1.2 Meta-learning and multi-task learning

Multi-task learning (Caruana, 1997) is a method that is similar to meta-learning inasmuch as it also relies on joint modelling of datasets. While the terms meta-learning and multi-task learning are often used rather loosely and without much distinction in the literature, here we take them to mean different things, with notable differences.

We will use multi-task learning to describe problems where a single model is trained and used to make predictions on a small and fixed collection of datasets, or a single dataset with multivariate outputs. By contrast, we will use the term meta-learning to refer to algorithms which can not only model different datasets jointly, but can also quickly adapt to unseen ones, with only modest computational and memory overheads. Thus the feature which distinguishes meta-algorithms from multi-task algorithms is that the former involve a *learning-to-learn* component. By this wee mean that executing a meta-algorithm on some data, produces a cheap supervised algorithm which can itself learn. More concretely, a weather modelling problem where we are given a collection of temperature datasets, which we choose to model with a single model, naturally falls in the multi-task learning category. If in addition we choose to model these datasets in a way which supports fast adaptation and predictions on new, unseen temperature datasets, then the term meta-learning is appropriate.

The differences between supervised learning, multi-task learning and meta-learning are diagrammatically illustrated in fig. 1.2 for clarity. A supervised algorithm (top) is trained and makes predictions on, a single dataset. Multi-task algorithms (middle) extend supervised algorithms, by modelling and predicting on several datasets jointly. By contrast, a meta-algorithm (bottom) is trained on a collection of datasets, and returns a supervised algorithm, which can itself accept and make predictions on unseen ones. The efficiency of a meta-learning algorithm depends on the extent to which the supervised algorithm that it produces is cheap to execute, from a computational and memory standpoint.

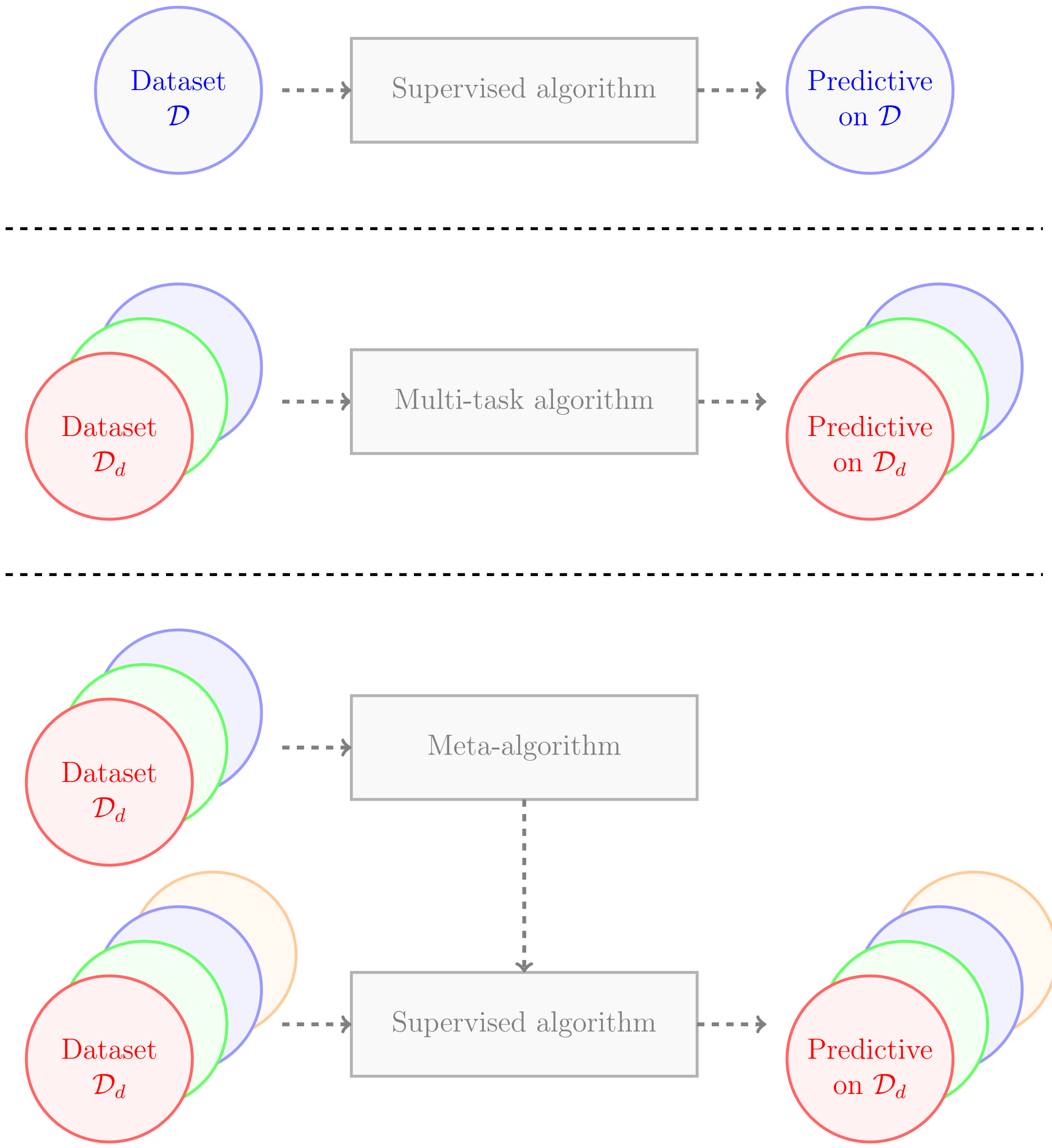

Figure 1.2: Illustration of the inputs and outputs of typical supervised algorithms (top), multi-task algorithms (middle) and meta-algorithms (bottom). Unlike multi-task algorithms, meta-algorithms can be applied to new datasets which have not been used in training (orange).

The defining characteristic which sets meta-algorithms apart from multi-task algorithms is that the former can quickly and cheaply adapt to new data. This feature of meta-algorithms offers a number of exciting opportunities.

1.3 Limitations and opportunities of meta-learning

Unlike supervised or multi-task algorithms, meta-algorithms are designed such that they can quickly adapt to unseen datasets, with minimal computational and memory overheads. On the other hand, the ability to cheaply adapt to new datasets comes at the cost of increased training times and data requirements. Generalising to unseen datasets is an arguably more demanding task than modelling a fixed collection of datasets. and meta-models typically require several datasets, each containing multiple datapoints in order to train. Thus meta-algorithms are typically not applicable in the small data regime. Further, the need to process large amounts of data during training usually implies that longer training times are required. In effect, meta-algorithms can be used to achieve lower costs at inference time, with the drawback of increased training resources, both in terms of data as well as in terms of computation.

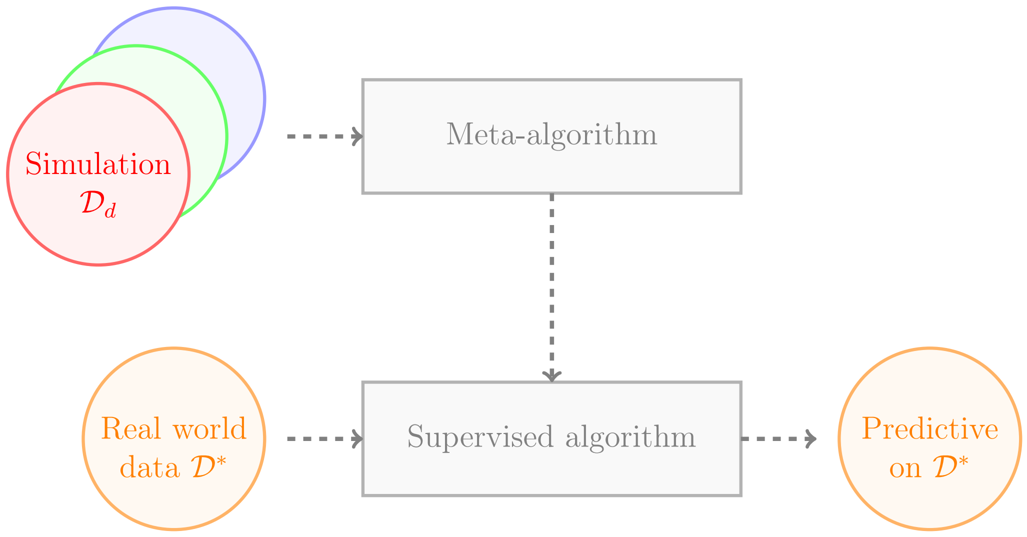
Simulation $\mathcal{D}_d$
Meta-algorithm
Real world data $\mathcal{D}^*$
Supervised algorithm
Predictive on $\mathcal{D}^*$

Figure 1.3: Illustration of a sim-to-real use case of a meta-algorithm. The meta-algorithm is trained on data generated by different simulations which represent different instances of the same system, such as different simulations of a robot with different limb sizes, weights and joint frictions. The meta-algorithm returns a supervised algorithm which can be used to make predictions on a real world dataset.

The trade-off between cost at inference and at training time presents exciting new possibilities. Meta-algorithms are particularly promising for use cases where resources are abundant during training, but are limited during inference. Here we identify three use cases in which trading off training and inference time may be particularly beneficial.

The first setting where meta-algorithms may be advantageous is when computational resources are highly limited at inference time. In mobile applications for example, there are limitations on the processing speed, memory size and permissible power consumption that a programme may use. In such cases, running an expensive supervised algorithm on a dataset collected by a particular user, on the user's device, may be infeasible. This challenge can be addressed by training a meta-algorithm on a central server, where training resources are abundant, and deploying it on resource-limited devices, where it can be used to make predictions cheaply. This is achieved by virtue of the fact that meta-algorithms are designed to be cheap at inference time.

Another setting where meta-algorithms may be useful, and which is also relevant to mobile applications, is that of personalised models and private data. In certain applications, there is need for model which can make predictions on a dataset which is specific to a particular user. For example, a visually impaired mobile phone user may want to identify their personal items using a vision assistive application. As opposed to recognising all items of a given class, such as all backpacks, a personalised recognition system must learn to distinguish specific items owned by the user, such as the user's specific backpack. The recognition model must adapt to the user's data, using as few training examples of these items as possible. Meta-models are ideal for this setting, since they can be trained on a centralised collection of such datasets, and then adapt to an unseen dataset collected by a new user. Further, once trained, these systems can be deployed on new users' devices, and perform inference on-device, without compromising the privacy of a user's data.

Lastly, one particularly exciting opportunity offered by meta-learning, is the ability to learn from synthetic data. In some applications, such as robotics or more generally physical modelling, we may have access to simulators, that models which roughly capture the behaviour of some real world phenomenon which we are interested in. Consider for example a fluid flow simulator which captures the salient behaviour of the weather, or a mechanics simulator which models the dynamics of a moving robot. One can use such a simulator to generate synthetic data which to some extent resemble the behaviour of the real world phenomenon. By using different parameters in each run of the simulator, we can create a variety of synthetic datasets, which we can use to train a single meta-model. Once trained, we can apply this meta-model on real world data, as pictured in fig. 1.3. Sim-to-real can be especially useful in cases where collecting real data is highly expensive or infeasible. In such cases, access to a simulator enables the training of meta-models on substantially larger volumes of data. By drawing on numerous synthetic datasets observed during training, a meta-model may be able to robustly generalise to a real world datasets.

1.4 Outline and contributions

This remainder of this report is structured as follows. Chapter 2 presents the required background. We introduce a definition of the meta-learning problem, and discuss existing meta-algorithms from the perspective of hierarchical probabilistic models. We then review the Neural Process family, a class of meta-models based on neural networks, which are highly flexible and cheap at inference time, both from a computational as well as a memory standpoint. We present existing members of these families, summarising their relative merits and limitations.

Chapter 3 introduces a new family of models, referred to as feature-based Gaussian Neural Processes (GNPs), which are a variant of the Fully Convolutional GNP (FullConvGNP) model of Bruinsma et al. (2021). Feature-based GNPs maintain several of the attractive features of FullConvGNPs, but are significantly easier to scale to higher input dimensions, making them a valuable addition to the NPF. Extending the proposed models, we show how marginal transformations can be applied to the outputs of GNPs to model non-Gaussian data. We also identify key limitations associated with feature-based GNPs and provide practical modifications towards remedying these.

Chapter 4 discusses equivariant CNPs and GNPs. Equivariance is a highly useful inductive bias for numerous machine learning models. However, current approaches for building equivariance into CNPs rely on convolutions, making these models very expensive for data with higher input dimensions. We present alternative methods to certain kinds of equivariances into CNPs, without relying on convolutions, but rather on transformations of the input variables. We include a set of theoretical arguments showing that these transformations are not only sufficient to achieve these equivariances, but also necessarily utilised by any model exhibiting them. Because they do not rely on convolutions, the proposed models promise substantially better computational and memory efficiency for higheer input dimensions.

Chapter 5 concludes with a summary of the work presented in the report. It includes a discussion of directions for future work and a tentative timeline for executing these.

Chapter 2

Background

2.1 Introducing the meta-learning problem

We will adopt a general viewpoint for meta-learning, which covers several problem settings of interest, and encompasses a number of existing algorithms. We assume that a learner has access to a collection of datasets $\mathcal{D}_{1:D}$, each of which consists of input-output tuples, that is

$$\mathcal{D}_{1:D} = \mathcal{D}_1, \dots, \mathcal{D}_D, \quad \text{where} \quad \mathcal{D}_d = ((x_{d,n}, y_{d,n}))_{n=1}^{N_d}, \tag{2.1}$$

where the inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$ belong to the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. The specific form of $\mathcal{X}$ and $\mathcal{Y}$ depend on the task at hand. In a regression setting for instance, $\mathcal{D}_d$ may consist of dates and longitude-latitude coordinates $x_{d,n}$ of certain locations on the Earth, and measurements of average temperatures $y_{d,n}$ at those locations. In a classification setting, $\mathcal{D}_d$ could consist of a set of images $x_{d,n}$ of items and their corresponding custom labels $y_{d,n}$. We leave $\mathcal{X}$ and $\mathcal{Y}$ unspecified for now, to keep the discussion general.

We wish to design a meta-algorithm to jointly model the datasets $\mathcal{D}_{1:D}$. Importantly, this algorithm should include some adaptation mechanism through which it can rapidly adapt to an arbitrary new dataset $\mathcal{D}^*$. Thus, after training, the output of the meta-algorithm should itself be able to accept $\mathcal{D}^*$ as input, and produce an adapted or fine-tuned model which can make predictions on $\mathcal{D}^*$. It is therefore natural to regard the output of a meta-algorithm as a supervised algorithm, as depicted in fig. 1.2. The essential requirement for an efficient meta-algorithm, is that the supervised algorithm it produces can be applied with small computational and memory overheads (Bronskill, 2020). This ability of meta-algorithms to cheaply adapt to new datasets is in contrast to supervised and multi-task learning (Caruana, 1997), where the trained model cannot be used to make predictions on new tasks, without expensive re-training.

In order jointly model multiple datasets, it is necessary to make some assumptions about how the different datasets $\mathcal{D}_{1:D}$ are related. Although there exist a host of meta-algorithms which do not explicitly state the assumptions they make about cross-dataset dependencies, such as (Finn et al., 2017), they invariably involve certain implicit assumptions. Such algorithms can often be re-interpreted by examining their implicit assumptions more carefully (Finn et al., 2019; Kim et al., 2018). We will approach meta-learning from the viewpoint of multi-task probabilistic models, to make such assumptions explicit and view existing algorithms from a unified perspective.

2.2 Meta-Learning as a probabilistic problem

Let us turn to specifying our assumptions about how different datasets might be related to one another in the meta-learning setting. We will initially assume a probabilistic approach for jointly explaining multiple datasets, which will allow us to view several existing algorithms from the literature from a common perspective. Whilst a fully probabilistic approach to inference and learning is highly appealing from a theoretical standpoint, we will subsequently argue in favour of simplifying our approach on the grounds of computational tractability.

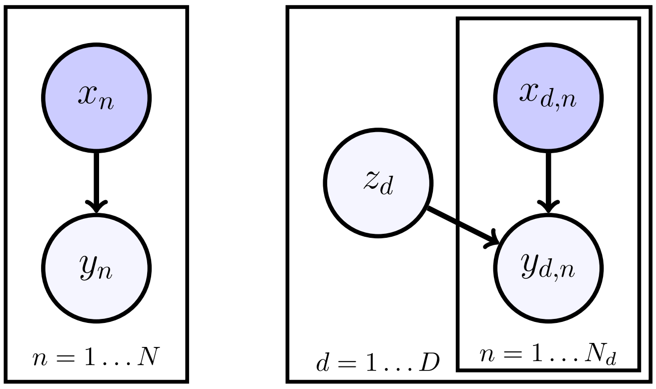

Figure 2.1: Graphical models assumed in supervised (left) and meta-learning (right).

2.2.1 A multi-task probabilistic model

One sensible choice for jointly modelling multiple datasets, is to assume that they are all sampled from a common generative process, such the one illustrated through the graphical model in fig. 2.1. In this generative process, each dataset $\mathcal{D}_d$ is generated by first sampling a random variable z_d from some fixed and common prior $p(z_d)$, independently for each dataset. Then for each input $x_{d,n}$ in $\mathcal{D}_d$, the generative process draws the corresponding output $y_{d,n}$ from a conditional distribution $p(y_{d,n}|x_{d,n}, z_d)$. This conditional distribution may involve a fixed set of parameters ψ, if for example it is a parametric model, which we will omit for brevity. Putting the prior and conditional distributions together gives the joint distribution, conditioned on the inputs, as

$$\log p\left(\mathcal{D}_{1:D}, z_{1:D}\right) = \sum_{d=1}^{D} \left[\log p(z_d) + \sum_{n=1}^{N_d} \log p(y_{d,n}|x_{d,n}, z_d) \right]. \tag{2.2}$$

This multi-task process is in contrast to the generative process assumed by simple supervised learning (fig. 2.1, left), which does not feature neither multiple datasets nor a latent variable z_d. The reason for including this high level latent variable is that a meta-model must be able to not only explain the data within a single dataset, but must also have the capacity to explain multiple different datasets. Consider a regression problem where we wish to model a collection of time-series datasets which exhibit periodic patterns, each with different periods. Using the same regression model on all the datasets, such as a periodic Gaussian Process (GP; Rasmussen, 2003) with a fixed period, would be an ill-suited model because it does not capture crucial ways in which the datasets differ. The generative process in fig. 2.1 can capture statistical dependencies both within a single dataset as well as across different datasets, via the conditional distribution $p(y_{d,n}|x_{d,n}, z_d)$ and the high-level random variable z_d, respectively. For a given dataset, z_d is fixed and the conditional distribution $p(y_{d,n}|x_{d,n}, z_d)$ captures regularities within the dataset. Across datasets, z_d is allowed to vary. As z_d changes, the statistical dependency of the outputs on the inputs changes, since the conditional distribution $p(y_{d,n}|x_{d,n}, z_d)$ changes. Allowing the conditional to change between datasets is crucially important for the model to be able to capture the ways in which these datasets differ.

At this point, we should also note that the multi-task graphical model in fig. 2.1 is general and captures a range of models, making it amenable to numerous settings. In particular, we may make any arbitrary choice for the information contained in z_d and its prior distribution $p(z_d)$ as well as the conditional $p(y_{d,n}|x_{d,n}, z_d)$. In the aforementioned example, we considered a model whose conditional was a periodic GP with a prior placed over its period hyperparameter, however, different conditionals such as neural networks can be used. Prior assumptions which we may have about the data, such as smoothness, periodicity or symmetries, can be directly incorporated into the model. Further, we can define more complicated hierarchies of variables, in order to express more intricate prior assumptions we may have about the data. For example, Lake et al. (2015) introduced a probabilistic model to model images of handwritten characters from different alphabets, produced by different human writers. This model used a latent z_d which involved both discrete and continuous variables conveying high-level information about the characters drawn. Similar multi-task models are used in various other settings, such as the unsupervised meta-algorithm of Edwards and Storkey (2016).

There are several appealing reasons for which we might advocate in favour of the probabilistic approach outlined above. First, a probabilistic approach requires the explicit statement of all relevant assumptions, making the model transparent (Mackay, 1992). Second, given some data $\mathcal{D}_{1:D}$, a probabilistic approach provides principled methods for estimating model parameters, inferring distributions over latent variables, making predictions and comparing alternative candidate models (MacKay, 1992). More specifically, model parameters can be estimated by maximising the marginal likelihood of the data, that is

$$\psi^* = \arg\max_{\psi} \log p(\mathcal{D}_{1:D}), \text{ where } p(\mathcal{D}_d) = \int p(\mathcal{D}_d, z_d) dz_d. \tag{2.3}$$

Given multiple alternative models, we can select the best candidate by choosing the model with greatest marginal likelihood. We can quantify our belief about latent variables given the observed data using the posterior distribution $p(z_d|\mathcal{D}_d)$ computed via Bayes' rule

$$p(z_d|\mathcal{D}_d) = \frac{p(\mathcal{D}_d, z_d)}{p(\mathcal{D}_d)}. \tag{2.4}$$

To make predictions on a new dataset $\mathcal{D}^*$, we can integrate out the latent variable in the conditional, weighted by its posterior given the observed data

$$p(y^*|x^*, \mathcal{D}^*) = \int p(y^*|x^*, z) q(z|\mathcal{D}^*) dz. \tag{2.5}$$

More generally, we can appeal to the well known result that any agent which is reasoning about degrees of belief regarding uncertain quantities, must be reasoning via the sum and product rules of probability (Jaynes, 2003). However, despite the attractive features that Bayesian inference offers, it also presents important challenges.

2.2.2 Challenges in Bayesian meta-learning

Whilst there are numerous reasons for favouring a fully Bayesian approach from a theoretical standpoint, it introduces two central challenges. First, exact Bayesian inference and learning are not analytically tractable for most models of interest. Second, in many cases we may not know what a good prior for the latent variables may be, or how the conditional distribution should depend on the it. Let us discuss these limitations in more detail.

With the exception of models with conjugate distributions (Bishop, 2006), the posterior distribution of most models of interest is not analytically tractable. For example, evaluating the exact marginal likelihood in eq. (2.3) is typically not possible, neither is performing integrals with respect to the posterior for making predictions, as in eq. (2.7). In such cases, two predominant methods, namely Monte Carlo and approximate inference are typically used to address intractabilities.

Unfortunately, while methods such as Markov Chain Monte Carlo (MCMC; Robert and Casella, 2013) can be used to estimate quantities such as integrals, they are often practically unrealistic. For example, the quality of MCMC estimates crucially depends on the mixing time of the Markov Chain, which can be very large in practice. The large computational costs of MCMC are an obstacle in meta-learning, because they go against the requirement that a meta-algorithm should be able to cheaply adapt to new datasets.

On the other hand, approximate inference methods, such as Variational Inference (VI; Jordan et al., 1999), Expectation Propagation (EP; Minka, 2013) or the Laplace method (Mackay, 1992) to name a few, can often offer a cheaper alternative to MCMC. These methods provide approximations to the exact posterior and marginal likelihood, offering various degrees of guarantees. For example, VI using the Evidence Lower Bound (ELBO; Bishop, 2006) approximates the true posterior with a distribution from a tractable family, and maximises a quantity which lower bounds the marginal likelihood, which can be optimised without risk of overfitting. Typically, approximate methods such as VI require shorter training times than Monte Carlo methods and are thus favoured in a range of applications. Approximate inference can also be combined with amortisation, which is the use of a parametric function, such as a neural network, to model the mapping from conditioning data to distribution parameters of latent variables, and has proved enormously successful in practice (Kingma and Welling, 2013). Unfortunately, approximate inference loses several of the theoretical arguments which justified probabilistic methods in the first place. Most approximation schemes are not consistent with desiderata regarding inferential consistency. For example, if we use an approximate algorithm to update our belief about z_d given $\mathcal{D}_d$, we will obtain different answers depending on whether we use all the data in $\mathcal{D}_d$ at once, or if we consider each datapoint sequentially in an online manner (Opper and Winther, 1998). In addition, the quality of a model's predictions can be impacted by the choice of approximate inference scheme (Le et al., 2018).

In light of these difficulties, we can propose a more modest alternative, which satisfies our meta-learning desiderata, namely to model datasets jointly, in a manner that supports rapid adaptation. In many cases, we may not have good priors about how the datasets have been generated, meaning that there may be *mismatches* between the model used and the true generative process. Further, we may not be interested in inferring distributions over latent variables. In such cases, we may solely be interested in the quality of an algorithm's predictions, rather than the faithfulness with which it approximates a posterior over latent variables. This view has been previously argued in the context of supervised learning by Osband et al. (2021), who propose benchmarking models in terms of their predictive performance, as the primary metric of interest.

Therefore, we argue in favour of the following simplified approach to meta-learning. First, in cases where we are not interested in inferring distributions over latent variables, we may instead focus on optimising for predictive performance, ignoring the quality of our posterior approximations. Instead of inferring entire distributions over latent variables, we can make point estimates for them, meaning that we use approximate posteriors of the form

$$q(z|\mathcal{D}^*) = \delta(z - z^*) \tag{2.6}$$

for some deterministic z^* which depends on the data $\mathcal{D}^*$. Then, the predictive integral is tractably approximated as

$$p(y^*|x^*, \mathcal{D}^*) = \int p(y^*|x^*, z)q(z|\mathcal{D}^*)dz \approx p(y^*|x^*, z^*). \tag{2.7}$$

In cases where the conditional has a known conjugate distribution to a subset of the random variables, we can still tractably infer whole distributions over these variables. In addition, while we may not know what a good choice for the exact forms of the prior and the conditional may be, we might still be able to encode useful inductive biases, such as symmetries, into these. Lastly, borrowing from the success of amortisation in VI, we can use an amortisation function which maps the conditioning data $\mathcal{D}^*$ to the latent variable z^*. Amortisation directly addresses the rapid adaptation requirement, since it allows the model to make predictions on unseen datasets, without requiring an expensive inference scheme. Relating this approach to the description in fig. 1.2, this amortisation function can be viewed as the supervised algorithm which infers the task specific latent variable z^* from the conditioning data $\mathcal{D}^*$. Thus, adaptation and inference of latent variables can be regarded as equivalent terms, allowing us to view existing meta-algorithms from the viewpoint of approximate inference.

2.2.3 Interpreting meta-algorithms as approximation schemes

Several meta-algorithms in the literature can be viewed as instances of the description above. Here we briefly discuss some of the more widely known algorithms, for the purposes of illustrating their connection to the above discussion.

Finn et al. (2017) introduced MAML, a novel training scheme which optimises the parameters of a neural network to values such that a few further optimisation steps suffice to tune the network for high performance in a new task. In this case, the adaptation mechanism consists of the application of a number of optimisation steps to the model parameters, using a task-specific loss. Finn et al. (2019) recognised that MAML can be interpreted as a probabilistic model, by regarding the task-specific gradient optimisation as an inference procedure, and proposed a modification to MAML to improve its predictive performance on the basis of this. Other similar gradient-based methods (Kim et al., 2018; Nichol et al., 2018) have built on this work. One attractive feature of these gradient-based methods are model-agnostic, meaning they can be applied to any parametric architecture. However, as noted by Bronskill (2020), allowing all model parameters to adapt may result in overfitting.

Instead of allowing all model parameters to adapt, other lines of work retain only a small set of task-specific parameters. These approaches use parametric functions, such as neural networks, to provide an amortised mapping from the conditioning data to the task-specific parameters. For example, working in the classification setting, Snell et al.

(2017) introduced the prototypical networks. This model uses a neural network to map the inputs of a conditioning set $\mathcal{D}_d$ to embedding vectors. The vectors corresponding to each class in $\mathcal{D}_d$ are averaged, to produce one embedding per class, called a prototype. To classify a new datum, the model embeds its input and uses the distance of this embedding to each of the prototypes as arguments to a softmax function. As Snell et al. (2017) point out, their method can reinterpreted as a mixture model, with the form of the mixture components depending on the distance metric used. The inferred parameters of these mixtures can be viewed as point estimates of latent variables of a probabilistic multi-task model such as the one in fig. 2.1.

More recently, Garnelo et al. (2018a) introduced a promising new kind of amortised meta models, called Conditional Neural Processes. CNPs and their variants (Garnelo et al., 2018b), here collectively referred to as the Neural Process family (NPF), rely on an amortised neural network to map conditioning data to a task-specific set of parameters. These parameters are passed together with a new inputs, to another neural network to make predictions at these input locations. Several works have extended the NPF with models using beneficial inductive biases, or tailored for specific applications, for example by using attention (Kim et al., 2019), convolutional architectures (Foong et al., 2020; Gordon et al., 2018), leveraging symmetries (Holderrieth et al., 2021; Kawano et al., 2021), or using architectures for language modelling Gordon et al. (2019). Members of the NPF have demonstrated promising performance on several meta-learning tasks. Notably, Bronskill (2020) presents an evaluation of several of the aforementioned meta-algorithms, in which members of the NPF achieved state-of-the-art performance on a number of classification problems. Motivated by the competitive empirical performance and low inference costs of the NPF, in the present work we focus on models of this kind.

2.3 Conditional and Latent Neural Processes

In this section, we introduce the NPF models which we will be focusing on in more detail. Before describing the models themselves, we present two pieces of useful terminology, namely *context and target sets* and *prediction maps.*

2.3.1 Prediction maps & context and target sets

Following the work of Foong et al. (2020), we view the NPF through the viewpoint of prediction maps. Suppose $\mathcal{D}$ is a single dataset and consider splitting this into a context set $\mathcal{C}$ and a target set $\mathcal{T}$

$$\mathcal{C} = (x_c, y_c), \text{ where } x_c = (x_{c,1}, \dots, x_{c,N}), y_c = (y_{c,1}, \dots, y_{c,N}) \quad \text{(Context)}$$

$$\mathcal{T} = (x_t, y_t), \text{ where } x_t = (x_{t,1}, \dots, x_{t,M}), y_t = (y_{t,1}, \dots, y_{t,M}) \quad \text{(Target)}$$

such that $\mathcal{D} = \mathcal{C} \cup \mathcal{T}$, where overlapping entries are allowed. The context set will act as the data on which the model is conditioned, and the target set will be the set of points at which predictions are made. A prediction map π is a function which maps (1) a context set (x_c, y_c) and (2) a set of *target inputs* x_t to a distribution over the corresponding *target outputs* y_t:

$$\pi\left(y_t; x_c, y_c, x_t\right) = \int p\left(y_t | x_t, z\right) q(z|\theta) dz, \tag{2.8}$$

where $\theta = \theta(x_c, y_c, x_t)$ is a set of distribution parameters computed using the context set and target input set. A prediction map of the form in eq. (2.8) is consistent to permutations of the datapoints in the target set (xt, yt), as well as to marginalisations of any of the y_t variables. This means that its predictions are consistent, no matter which order the target points are presented to it, or whether some of the target outputs have been marginalised out. The conditional $p\left(y_t | x_t, z\right)$ and distribution parameters $\theta(x_c, y_c, x_t)$ may depend on a set of learnable parameters ψ, which we omit for notational simplicity. Prediction maps include, but are not limited to, Bayesian posteriors. One familiar example of such a Bayesian map is the Gaussian Process (GP; Rasmussen, 2003) posterior

$$\pi\left(y_t; x_c, y_c, x_t\right) = \mathcal{N}\left(y_t; \mathbf{m}, \mathbf{K}\right), \tag{2.9}$$

where $\mathbf{m} = m(x_c, y_c, x_t)$ and $\mathbf{K} = k(x_c, x_t)$ are given by the usual posterior expressions for a GP with mean function m and posterior covariance function k. Bayesian prediction maps start from some prior assumptions and make use of Bayes' rule to express a predictive posterior, which can be written as a prediction map. Unfortunately, as we have already discussed, Bayesian posteriors are most often analytically intractable. Instead, the NPF relies on the idea that neural networks can be used to parametrise prediction maps. As Foong et al. (2020) point out, these prediction maps do not generally correspond to a single consistent probabilistic model. Instead, such a prediction map is equivalent to a family of different probabilistic models, with a different prior

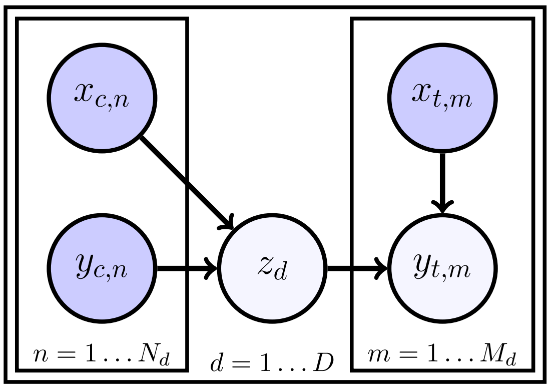

Figure 2.2: Graphical model implicitly assumed by the NPF.

for each context set, resulting in a graphical model such as that depicted in fig. 2.2. Arbitrary prediction maps may not necessarily respect the product rule of probability

$$\begin{aligned}\pi\left(y^*; (x_c, x^\dagger), (y_c, y^\dagger), x^*\right) \pi\left(y^\dagger; x_c, y_c, x^\dagger\right) \neq \\ \neq \ \pi\left(y^\dagger; (x_c, x^*), (y_c, y^*), x^\dagger\right) \pi\left(y^*; x_c, y_c, x^*\right),\end{aligned}$$

and thus are not guaranteed to be equivalent too a single probabilistic model. Nonetheless, they can be optimised for predictive performance on the held-out target set, and make accurate predictions on unseen datasets. We also note that a predictive map such as the one in eq. (2.8) adheres to our meta-learning requirements. First, it represents datasets jointly, by sharing the functional forms of the conditional and amortisation posterior across all datasets. Second, by making the amortisation posterior a parametric function which is cheap to evaluate, adapting the model on a new dataset can be very cheap.

2.3.2 Conditional Neural Processes

The Conditional Neural Process (CNP), introduced by Garnelo et al. (2018a), is a meta-model which has a prediction map of the form

$$\pi\left(y_t; x_c, y_c, x_t\right) = \prod_{m=1}^{M} p(y_{t,m}|x_{t,m}, \theta), \tag{2.10}$$

where $p(y_{t,m}|x_{t,m}, \theta)$ is a Gaussian distribution parameterised by a neural network with a Gaussian output, and $\theta = \theta(x_c, y_c)$ is parameterised by a DeepSet neural network (Zaheer et al., 2017). A DeepSet is a parametric neural network which operates on inputs which are sets, and whose output is invariant to permutations of the elements of this set. This ensures that the output of the prediction map is invariant to permutations of the conditioning data. The prediction map in eq. (2.10) can be obtained by setting $q(z|\theta) = \delta(z - \theta)$ in eq. (2.8). The parameters ψ of a CNP can be trained via maximising the predictive log-likelihood on the target set, conditioned on the context set

$$\psi^* = \arg\max_{\psi} \log \pi\left(y_t; x_c, y_c, x_t\right), \tag{2.11}$$

which can be computed analytically in closed form.[1] The prediction map in eq. (2.10) factorises between different target points and thus does not model any statistical dependencies between different target points, that is $y_{t,m} \perp\!\!\!\perp y_{t,m'}$ whenever $m \neq m'$. We will refer to prediction maps which treat outputs of different target points independently of one another as *mean-field*. This is a significant limitation of the CNP model which has motivated a number of existing models (sections 2.3.4 and 2.3.5) as well as new models introduced in this work (chapter 3).

2.3.3 Convolutional Conditional Neural Processes

Convolutional Conditional Neural Processes (ConvCNPs; Gordon et al., 2020) are a variation of the original CNP, which uses convolutional architectures, to exploit translational symmetries that may be present in the prior generating process. In some

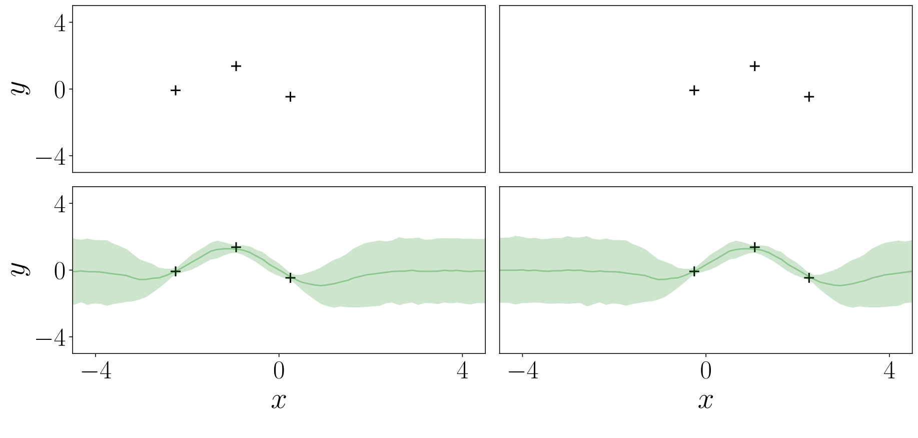

Figure 2.3: Illustration of translation equivariance. Starting from a context set (top left) and applying a translation (top right) followed by the ConvCNP (bottom right) gives the same prediction as if we applied the ConvCNP to the original set (bottom left), followed by a translation of the prediction map (bottom right).

[1]Note that eq. (2.11) involves a single dataset, to keep the notation light, but in practice we consider multiple datasets, summing their contributions to the total predictive log-likelihood.

applications, such as spatial regression or time series, we may know that the data generating process has a stationary prior. For any model with a stationary prior, the exact Bayesian posterior is *translation equivariant* (TE; Cohen and Welling, 2016a; Foong et al., 2020). By this we mean that (1) conditioning on the dataset and translating the posterior predictive, or (2) translating the dataset and conditioning on it, yield in the same posterior predictive, as illustrated in fig. 2.3. When designing a model for stationary data, we can leverage this prior knowledge by designing a prediction map which is also TE.

```
1: **input**: $(x_c, y_c)$, $(x_t, y_t)$, grid density $\rho$
2: $\texttt{min}, \texttt{max} \leftarrow \texttt{range}(x_c \cup x_t)$
3: $g \leftarrow \texttt{uniform\_grid}(\texttt{min}, \texttt{max}, \rho)$
4: $\tilde{y}_{c,n} \leftarrow [1, y_{c,n}]^\top$
5: $h_i \leftarrow \texttt{SetConv}(g, x_c, \tilde{y}_c)_i$
6: $h_i^{(1)} \leftarrow h_i^{(1)} / h_i^{(0)}$
7: $\mu_i, \sigma_i \leftarrow \mathrm{CNN}_\psi(h)_i$
8: $\mu_i \leftarrow \texttt{SetConv}(x_t, g, \mu)_i$
9: $\sigma_i \leftarrow \texttt{SetConv}(x_t, g, \sigma)_i$
10: $\sigma_i \leftarrow \texttt{pos}(f_{\sigma_i})$
11: **return:** $\mu, \sigma$
```

(a)

```
1: **input**: $a, b, c$, $\texttt{size}(b)$ == $\texttt{size}(c)$
2: $s_i \leftarrow \sum_{j=1}^{N} c_j \phi(a_i - b_j)$
3: **return:** $s$
```

(b)

Figure 2.4: Pseudocode for the ConvCNP, modified from Gordon et al. (2020). The forward computation of the ConvCNP is specified in (a) and the operation of the SetConv layer is specified in (b). The symbols ϕ and `pos` respectively denote a Radial Basis Function and a positive-constraining function, such as a Softplus. The notation $h^{(0)}, h^{(1)}$ stresses that the first and second channel of h are accessed.

ConvCNPs achieve TE by applying a series of equivariant maps to the context set and target inputs, illustrated in fig. 2.4. More specifically, the ConvCNP embeds the context set to a continuous real function by applying a SetConv layer, which is a TE function with set inputs. The output of the SetConv is discretised on a grid and passed through a Convolutional Neural Network (CNN; LeCun et al., 1995). Lastly, given a target input, the ConvCNP computes a weighed average of the grid values, where the weights are given by the output of a Radial Basis Function (RBF), applied to the distance between the target input and each grid point. The resulting feature vector is used to parameterise a Gaussian distribution over each target output, independently.

Thus, like the CNP, the ConvCNP also involves a mean-field prediction map of the form

$$\pi\left(y_t; x_c, y_c, x_t\right) = \prod_{m=1}^{M} p(y_{t,m}|x_{t,m}, \theta), \tag{2.12}$$

where $\theta = \theta(x_c, y_c)$ denotes the grid values computed using the context set. Since each layer of the ConvCNP is TE, the whole ConvCNP is also TE. Thus, the ConvCNP can generalise to data which has been translated outside its data distribution, achieving excellent generalisation performance on time series and image modelling tasks (Gordon et al., 2020). Further, due to parameter tying, ConvCNPs can be designed to have a relatively small number of trainable variables, without compromising their performance. Lastly, the ConvCNP can also be trained with the same maximum-likelihood objective as the CNP, given in in eq. (2.11), without requiring approximations.

2.3.4 Neural Processes

As noted by Garnelo et al. (2018a), one important limitation of CNPs model is that they model outputs via a fully factored prediction map. As such they cannot be used to draw function samples with statistical dependencies in the output. In subsequent work, Garnelo et al. (2018b) introduced a variant of the CNP, called the Neural Process (NP) model, with the following prediction map

$$\pi\left(y_t; x_c, y_c, x_t\right) = \int \prod_{m=1}^{M} p(y_{t,m}|x_{t,m}, z) q(z|\theta) dz, \tag{2.13}$$

where q is a tractable distribution, such as a Gaussian, parameterised by the context dependent variable $\theta = \theta(x_c, y_c)$. Since the random variable z is common to all target points, it induces statistical dependencies between the target outputs. Modelling output dependencies is often a crucial element for downstream estimation tasks. We will explore this point in detail in chapter 3.

Whilst Garnelo et al. (2018b) initially proposed training this model by using a approximate version of the Evidence Lower Bound (ELBO), their approximation does not maintain the attractive guarantees of the ELBO. Le et al. (2018) evaluated a number of variations to the objective of Garnelo et al. (2018b), concluding there was no significant performance difference for favouring a particular one of these. However, Foong et al. (2020) showed that, in a variant of this model using attention Kim et al. (2019), Monte

Carlo approximations of the maximum-likelihood style objective

$$\psi^* = \arg\max_{\psi} \log \int p(y_t|x_t, z)q(z|\theta)dz, \tag{2.14}$$

outperformed the objective presented in Garnelo et al. (2018b). However, they also noted that handling the objective in eq. (2.14) presents certain challenges. For example, when training with this objective, we must draw several Monte Carlo samples from $q(z|\theta)$, with post-training performance depending on the number of samples used. This may prove computationally and memory intensive for larger architectures. In addition, Foong et al. (2020) show that $q(z|\theta)$ may need to be heuristically controlled during the initial stages of training to prevent $q(z|\theta)$ collapsing to a delta function.

2.3.5 Convolutional Neural Processes

Motivated by the success of ConvCNPs and the usefulness of modelling output dependencies, Foong et al. (2020) propose the Convolutional Neural Process (ConvNP) model, which is a TE version of the NP model. The ConvNP has a prediction map of the same form as the NP

$$\pi\left(y_t; x_c, y_c, x_t\right) = \int \prod_{m=1}^{M} p(y_{t,m}|x_{t,m}, z)q(z|\theta)dz, \tag{2.15}$$

however, both the conditional and amortised posterior have TE architectures (fig. 2.5). Like the ConvCNP, the ConvNP transforms the context set through a SetConv layer followed by a discretisation step and the application of a CNN. The outputs of this CNN are used to parameterise a distribution for each point on the grid, the values of which are viewed as independent latent variables. The ConvNP then draws a sample from each of these distributions, passes the resulting grid of stochastic values through another CNN and finally through a smoothing step as in the ConvCNP. The second CNN that is applied, correlates the grid values, thus allowing the model to represent statistical dependencies in the output. Similarly to the NP, one sample of the latent grid of the ConvNP corresponds to a different regression model. Foong et al. (2020) proposed using the objective in eq. (2.14) for training ConvNPs. Further, since every layer of the ConvNP is TE, so is the entire ConvNP.

```
1: input: $(x_c, y_c)$, $(x_t, y_t)$, grid density $\rho$
2: min, max $\leftarrow$ range$(x_c \cup x_t)$
3: $g \leftarrow$ uniform_grid(min, max, $\rho$)
4: $\tilde{y}_{c,n} \leftarrow [1, y_{c,n}]^\top$
5: $h_i \leftarrow$ SetConv$(g, x_c, \tilde{y}_c)_i$
6: $h_i^{(1)} \leftarrow h_i^{(1)}/h_i^{(0)}$
7: $\tilde{\mu}_i, \tilde{\sigma}_i \leftarrow \mathrm{CNN}_{\psi_1}(h)_i$
8: $z_i \sim \mathcal{N}(\tilde{\mu}_i, \tilde{\sigma}_i^2)$
9: $\mu_i, \sigma_i \leftarrow \mathrm{CNN}_{\psi_2}(z)$
10: $\mu_i \leftarrow$ SetConv$(x_t, g, \mu)_i$
11: $\sigma_i \leftarrow$ SetConv$(x_t, g, \sigma)_i$
12: $\sigma_i \leftarrow$ pos$(f_{\sigma_i})$
13: return: $\mu, \sigma$
```

Figure 2.5: Pseudocode for the ConvNP. The operation of the SetConv layer is specified in fig. 2.4. Compared to the ConvCNP, the ConvNP has the additional steps in lines 8 and 9, coloured in blue.

2.3.6 Equivariance

In this section we introduce the notion of equivariance in more detail. Whilst we have seen that the ConvCNP and ConvNP models are TE, there exist many other types of equivariances that one may be interested in (Cohen et al., 2018; Cohen and Welling, 2016a). Instead of giving a complete description of equivariances, we focus on some particular ones, namely translation and rotoreflection equivariance, defining them in terms of prediction maps. For a general description of equivariances we refer the reader to Cohen and Welling (2016a). Suppose π is a prediction map, and the input and output spaces are $\mathcal{X} = \mathbb{R}^{D_x}$ and $\mathcal{Y} = \mathbb{R}^{D_y}$ respectively. We say that π is TE if for any translation T

$$\pi\left(y_t; Tx_c, y_c, Tx_t\right) = \pi\left(y_t; x_c, y_c, x_t\right). \tag{2.16}$$

Thus if π is TE and we translate the context set, the predictive posterior also translates accordingly, because

$$\pi\left(y_t; Tx_c, y_c, x_t\right) = \pi\left(y_t; x_c, y_c, T^{-1}x_t\right), \tag{2.17}$$

where T^{-1} is the inverse of T. The ConvCNP and ConvNP satisfy this relation and are TE. This kind of equivariance is useful in applications where there is no preferred location in the input space. For the case of rotoreflection equivariance, we distinguish between the cases when the output is scalar, $D_y = 1$, and when the output is a vector in the same space as the input $D_x = D_y$. If $D_y = 1$, we say that π is rotoreflection equivariant if

$$\pi\left(y_t; Rx_c, y_c, Rx_t\right) = \pi\left(y_t; x_c, y_c, x_t\right). \tag{2.18}$$

If we rotate and reflect the context set a map satisfying eq. (2.19), then the predictive posterior rotates and reflects accordingly, since

$$\pi\left(y_t; Rx_c, y_c, x_t\right) = \pi\left(y_t; x_c, y_c, R^{-1}x_t\right). \tag{2.19}$$

This kind of equivariance is relevant in many applications where we know that the generative process has no preferred orientation, that is the prior is rotoreflection invariant. For the case $D_x = D_y$, we will say that π is rotoreflection equivariant if

$$\pi\left(Ry_t; Rx_c, Ry_c, Rx_t\right) = \pi\left(y_t; x_c, y_c, x_t\right). \tag{2.20}$$

If we rotate and reflect both the inputs and the outputs of the context set, the predictive posterior rotates and reflects accordingly

$$\pi\left(y_t; Rx_c, Ry_c, x_t\right) = \pi\left(R^{-1}y_t; x_c, y_c, R^{-1}x_t\right). \tag{2.21}$$

As opposed to eq. (2.19), in eq. (2.20) we are considering a rotoreflection applied to both the context inputs and outputs. Physical systems such as fluid flows and fields exhibit this kind of equivariance, since the laws of physics which govern such systems often have no preferred direction. Lastly, we can also consider prediction maps which are both translation as well as rotoreflection equivariant.

Kawano et al. (2021) and Holderrieth et al. (2021) introduced Group Equivariant CNPs (EquivCNP) and Steerable CNPs (SteerCNPs) respectively. These models generalise ConvCNPs to a richer set of equivariances, but work according to a similar principle. First, these models map the context set to a continuous function using a generalisation of the SetConv layer, with the appropriate equivariances. They then discretise this function and map it though a group equivariant CNN (Cohen and Welling, 2016a) or steerable CNN (Cohen and Welling, 2016b), also endowed with the desired equivariances, followed by an equivariant smoothing step for making predictions. Since every step of these models is equivariant, so are the models themselves. EquivCNPs and SteerCNPs exhibit favourable performance on tasks such as modelling wind velocity fields.

2.4 Chapter summary

In this chapter we have discussed meta-learning, placing our focus on two important features that every effective meta-algorithm should include. A meta-algorithm should be able to jointly model multiple datasets as well as rapidly adapt to unseen ones. We have seen how meta-learning can be described in terms of probabilistic multi-task models, which enable joint modelling, as well as how rapid adaptation can be achieved by appropriate mechanisms, such as amortisation networks. When sensible priors are not available, or when we have no interest in performing inference over latent variables, performing point estimates of latent variables is a sensible option. We discussed how various meta-algorithms can be viewed from the angle that we have adopted on meta-learning, including the CNP and NP model families. In the subsequent chapters we will address central limitations of these existing models, providing theoretical arguments and practical methods to help remedy these.

Chapter 3

Gaussian Neural Processes

3.1 Modelling output dependencies

One central limitation of the CNP model introduced by Garnelo et al. (2018a) is that it does not model statistical dependencies between outputs at different input locations. In particular, the predictive maps of the CNP and ConvCNP models (Garnelo et al., 2018a; Gordon et al., 2020), factorise between different target outputs as

$$\pi\left(y_t; x_c, y_c, x_t\right) = \prod_{m=1}^{M} p(y_{t,m} | x_{t,m}, \theta), \tag{3.1}$$

which implies that $y_{t,m} \perp\!\!\!\perp y_{t,m'}$ whenever $m \neq m'$. While providing meaningful error bars, mean-field predictives can only be used in settings where solely predictive marginals are required, such as Bayesian Optimisation (BO) with Upper Confidence Bounds (UCB; Srinivas et al., 2009), whilst remaining inapplicable to a wide range of other use cases, such as BO with Thompson sampling (Thompson, 1933). This is because many downstream estimation tasks, such as the minimisation step of Thomposon-based BO, require access to coherent function samples which capture the

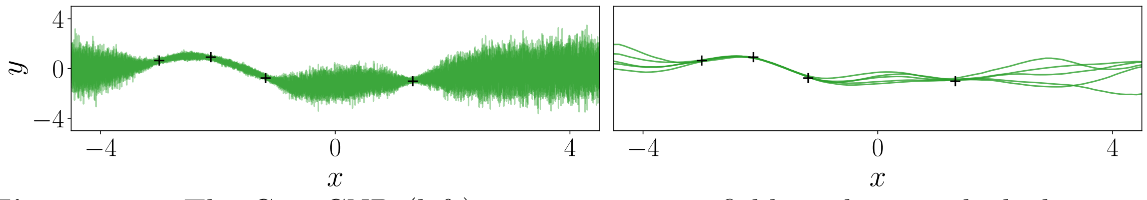

Figure 3.1: The ConvCNP (left) outputs a mean-field predictive which does not model statistical dependencies in the output. The Gaussian Neural Process family discussed in this chapter is able to model output dependencies (right).

dependencies between the output values of a function at different input locations. More specifically, we can distinguish between use cases where a mean-field predictive either (i) severely hurts the performance of the model or (ii) renders the model entirely inapplicable.

As an example for the first case, consider the task of modelling wind speeds over a region of space containing a number of wind farms, in order to predict the total energy these farms will inject to the energy grid. In this use case, it is important to predict this total energy not only accurately, that is to predict its mean correctly, but also precisely, that is to have low uncertainty in the prediction. Treating the wind speeds, and thus the energy produced, at different wind farms as independent will result in an under-confident estimate of the total energy produced. The overly loose uncertainty estimates obtained by summing a set of independent variables, as opposed to a set of correlated variables, would impact the predictive performance of the model.

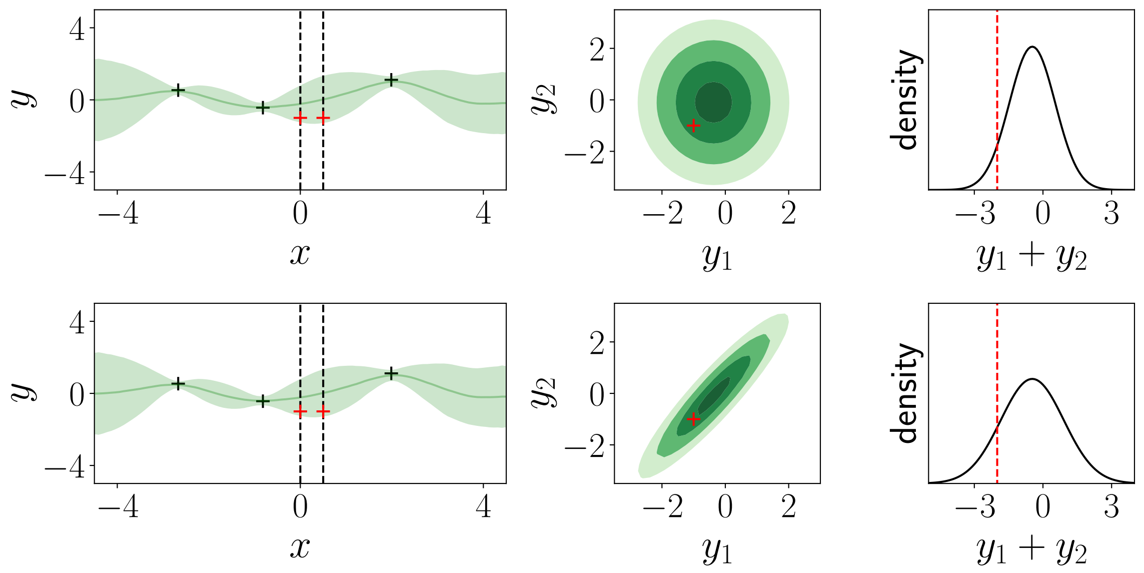

Figure 3.2: Illustration of the effect of output dependencies on a downstream estimation task. The two rows show two models, with (top) and without (bottom) output dependencies, but identical marginals. The left column shows the marginals of the prediction map (green) together with the context (black) and target (red) sets. The middle shows the prediction map on the two target points (green) and the target outputs (red). The right column shows the predicted distribution over the sum of the target outputs (blacK) and the actual sum of the target outputs (red).

As an example for the second case, consider a precipitation modelling problem where we wish to evaluate the probability of the event that the amount of rainfall per day within some region remains above some specified threshold throughout a sustained length of time, which could help assess the likelihood of a flood. Mean-field predictions, which model every input location independently, would assign unreasonably low probabilities to such events, making the model entirely inapplicable to this use case.

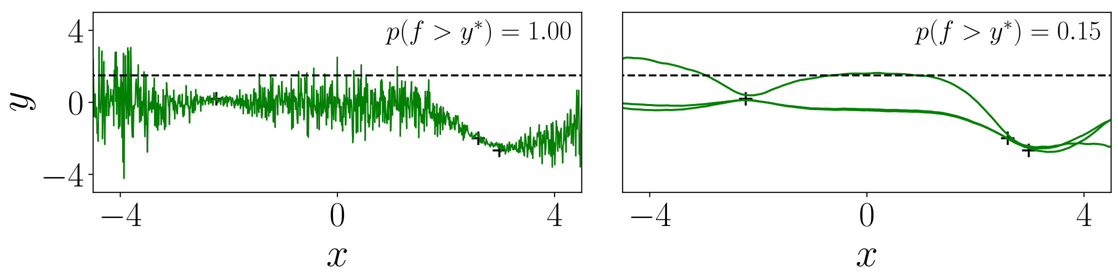

Figure 3.3: Illustration of the effect of output dependencies on a downstream estimation task. As with fig. 3.2, the two models shown here have identical marginals, however one does not model output dependencies (left), whereas the other does (right). The horizontal line shows the threshold $y = y^*$. The probability that the inferred function f exceeds the threshold, $p(f > y^*)$, is shown on the top right of each plot.

Having recognised the importance of modelling correlations for downstream estimation, Garnelo et al. (2018b) introduced the Neural Process (NP) model, which uses a latent variable to model correlations in the output. However, the presence of the latent variable introduces challenges in the training procedure because training via the exact log-likelihood is no longer possible. Instead the model must be trained either via the ELBO, rough approximations of it (Garnelo et al., 2018b; Le et al., 2018), or conservative Monte-Carlo approximations of the log-likelihood (Foong et al., 2020), which themselves introduce new challenges.

More recently Bruinsma et al. (2021) introduced a variant of the CNP called the Gaussian Neural Process, which we here refer to as the FullConvGNP. Unlike the CNP, the FullConvGNP models correlations in the output, whilst it does not utilise a latent variable, unlike the ConvNP. The FullConvGNP achieves this by directly parametrising both the mean and covariance of a predictive Gaussian distribution over the output variables. However, for D_x-dimensional data, the architecture of the FullConvGNP involves $2D_x$-dimensional convolutions, which are costly, and for $D_x > 1$, poorly supported by most Deep Learning libraries. Unlike the original CNP, which parameterises a mean-field prediction map, the FullConvGNP parameterises a correlated prediction map, that is

$$\pi\left(y_t; x_c, y_c, x_t\right) = \mathcal{N}\left(y_t; \mathbf{m}, \mathbf{K}\right), \tag{3.2}$$

where $\mathbf{m}$ is a mean vector and $\mathbf{K}$ is a full-rank positive definite covariance matrix. For the mean $\mathbf{m}$, the FullConvGNP uses a regular ConvCNP Gordon et al. (2020), which relies on a D_x-dimensional grid discretisation over which it applies a sequence of convolutions. The parametrisation of the covariance matrix is similar to that of the mean vector, except that it involves a $2D_x$-dimensional, rather than D_x-dimensional grid. This $2D_x$-dimensional grid can be thought of as the discretisation of a covariance function with a $2D_x$-dimensional input, corresponding to two D_x-dimensional arguments. By mapping the context set to a function, discretising it, and applying a CNN with learnable parameters to it, the FullConvGNP produces a discretised approximation to the predictive covariance. This approximation is then smoothed to obtain the value of the covariance at locations beyond the discretisation locations. Remarkably, the FullConvGNP can approximate a rich variety of GP posteriors very accurately (Bruinsma et al., 2021), including accurate estimates of the ground truth posterior covariances. Further, the FullConvGNP is easily trainable via the same maximum-likelihood procedure which the CNP uses, that is by maximising

$$\psi^* = \arg\max_{\psi} \log \pi\left(y_t; x_c, y_c, x_t\right). \tag{3.3}$$

This training scheme does not require approximations, and thus circumvents the difficulties present in latent variable models. Unfortunately however, the FullConvGNP must keep track of the values of a $2D_x$-dimensional grid, which incurs a large memory cost, whilst it must also apply $2D_x$-dimensional convolutions, which incur a large computational cost.

Despite the fact that the FullConvGNP is not realistically applicable to data with higher dimensional inputs, it points towards an effective method for modelling output dependencies, by specifying a predictive map which directly specifies these dependencies. If we can define a prediction map which explicitly parameterises output dependencies as in eq. (3.2), but does not rely on a grid, it is possible to obtain a model which avoids the scalability issues of the FullConvGNP.

3.2 Feature-based covariance parameterisations

We now turn to the task of specifying computationally and memory efficient ways to specify prediction maps of the form

$$\pi\left(y_t; x_c, y_c, x_t\right) = \mathcal{N}\left(y_t; \mathbf{m}, \mathbf{K}\right). \tag{3.4}$$

We refer to prediction maps of the form of eq. (3.4) as Gaussian Neural Processes (GNPs). The FullConvGNP of Bruinsma et al. (2021) is a member of this class, however it involves expensive convolutions for parameterising $\mathbf{K}$. In order to tackle the costs of the FullConvCNP, we will specify a method for parameterising posterior covariances which does not rely on a discretisation grid (Markou et al., 2021). Instead, our method will rely on *feature maps*, which we will use to efficiently parameterising covariances.

A feature map is a function f mapping a context set and a target input set (x_c, y_c, x_t) to a set of feature vectors in a fixed dimensional space, one for each target input. We further require that f is consistent under permutations of the context set and removals of target inputs. More specifically, f is consistent under permutations if

$$f(x_c, y_c, x_t) = f(\sigma(x_c), \sigma(y_c), x_t) \tag{3.5}$$

for any permutation σ of the context set, and it is consistent under removals of target points if

$$f(x_c, y_c, x_t)_{1:M-1} = f(x_c, y_c, (x_t)_{1:M-1}), \tag{3.6}$$

where $x_t = (x_{t,1}, \dots x_{t,M})$, and the subscript notation refers to selecting the first $M-1$ elements of the object which it subscripts. We can use feature maps to efficiently paramerise the posterior mean and posterior covariance of a GNP according to

$$\mathbf{m}_i = m(x_{t,i}, z), \tag{3.7}$$

$$\mathbf{K}_{ij} = k(f(x_{t,i}, z), f(x_{t,j}, z)) \tag{3.8}$$

where $z = z(x_c, y_c)$, m and f are neural networks with outputs in $\mathbb{R}$ and $\mathbb{R}^{D_f}$ respectively, and k is an appropriately chosen positive-definite function. Note that, since k models a posterior covariance, it cannot be stationary, because if it were, then the marginal variance would not be able to shrink close to the datapoints. Equations (3.7) and (3.8) define a class of GNPs which, unlike the FullConvGNP, do not require costly convolutions, which we refer to as feature-based models. GNPs can be readily trained via the log-likelihood

$$\psi^* = \arg\max_{\psi} \log \pi\left(y_t; x_c, y_c, x_t\right), \tag{3.9}$$

also used in Garnelo et al. (2018a), where ψ collects all the parameters of the neural networks m, f, and z. For example, we may choose these to be feedforward DeepSets, giving rise to Gaussian Neural Processes (GNPs); attentive DeepSets, giving rise to Attentive Gaussian Neural Processes (AGNPs); or convolutional architectures, giving rise to Convolutional Gaussian Neural Processes (ConvGNPs). In this work, we explore these three alternatives, proposing the ConvGNP as a scalable alternative to the FullConvGNP.

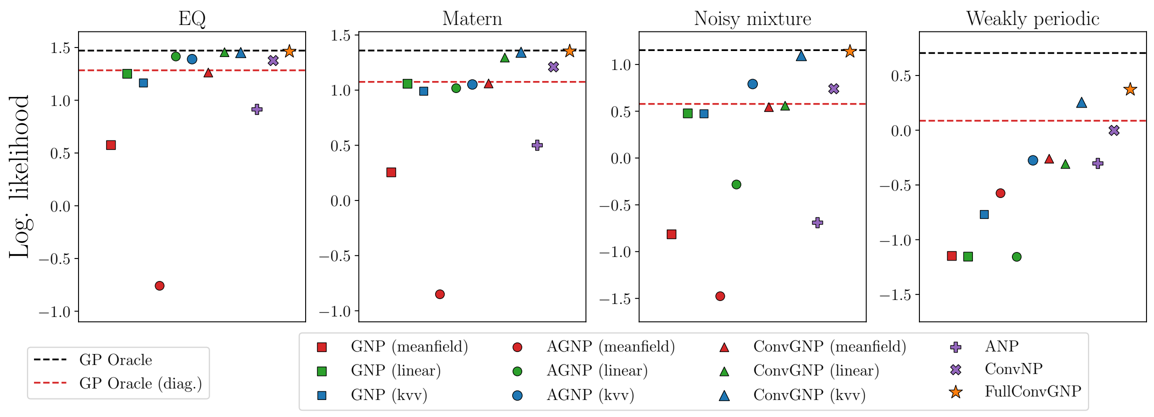

Figure 3.4: Predictive log-likelihood of the GNP family and the latent NP models, on the four tasks. The dashed lines show the predictive log-likelihood of a GP with the oracle covariance using a full covariance (black) and a diagonal covariance (red).

3.3 Gaussian Neural Processes on toy data

We apply the proposed models to synthetic datasets generated from GPs with various covariance functions and known hyperparameters. We sub-sample these datasets into context and target sets, and train via the log-likelihood objective in eq. (3.9). We also train the ANP and ConvNP models using the same objective as Foong et al. (2020), given in eq. (2.14). The NP and ConvNP models represent output dependencies via the shared latent variable z and the amortised posterior q. Figure 3.4 compares the predictive log-likelihood of the models, evaluated on in-distribution data, from which we observe the following trends.

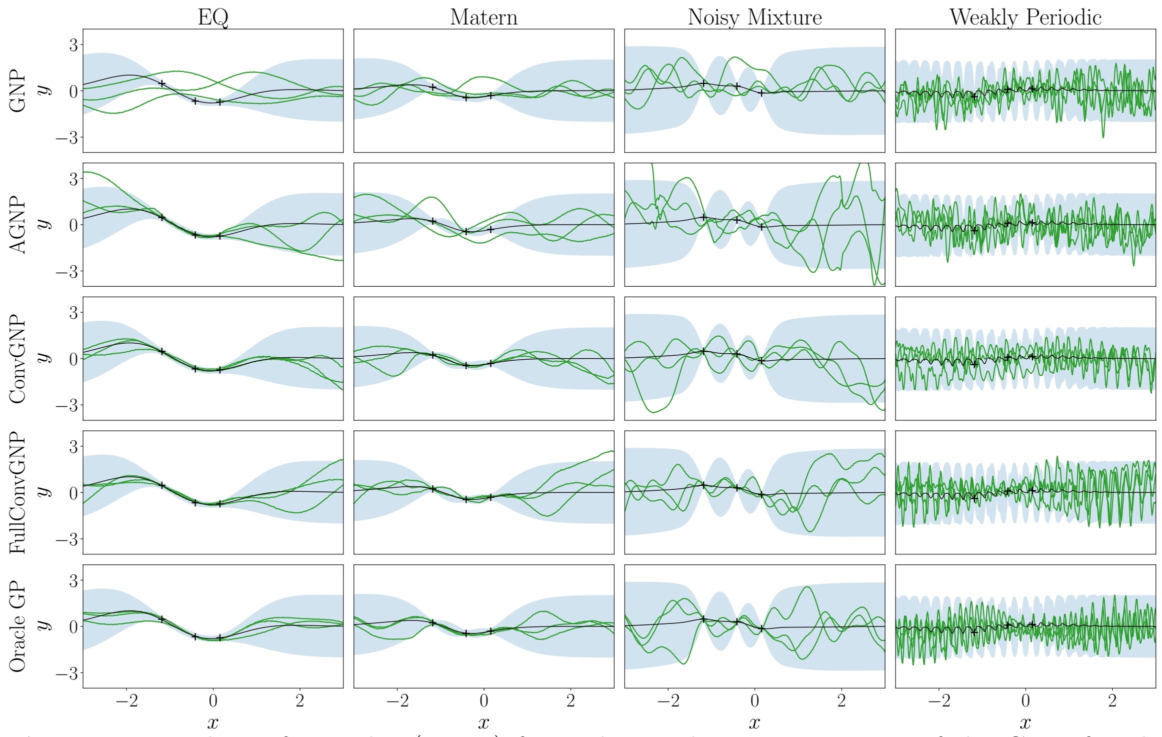

Figure 3.5: Plots of samples (green) from the predictive posteriors of the GNP family and the oracle GP, together with plots of the marginal posterior predictives of the oracle, using two standard deviations (blue).

Modelling dependencies improves performance: We expected that modelling output dependencies would allow CNP models to achieve better log-likelihoods. Indeed, for a fixed architecture, we see that the correlated GNPs (🟩, 🟦, 🟢, 🔵, ▲, ▲) typically outperform their mean-field counterparts (🟥, 🔴, ▲). This result is encouraging and suggests that GNPs can learn meaningful dependencies in practice, in some cases recovering oracle performance.

Comparison with the FullConvGNP: The correlated ConvGNPs (▲, ▲) are often competitive with the FullConvGNP (⬟). The `kvv` ConvGNP (▲) is the only model, from those examined here, which competes with the FullConvGNP in all tasks. Unlike the latter, however, the former is scalable to $D = 2, 3$ dimensions.

Comparison with the ANP and ConvNP: Correlated GNPs typically outperform the latent-variable ANP (✚) and ConvNP (✖) models, which could be explained by the fact that the GNPs have a Gaussian predictive while ANP and ConvNP do not, and all tasks are Gaussian. Despite experimenting with different architectures, and even allowing for many more parameters in the ANP and ConvNP compared to the AGNP

(●, ●) and ConvGNP (▲, ▲), we found it difficult to make the latent variable models competitive with the GNPs. We typically found the GNP family easier to train than these latent variable models.

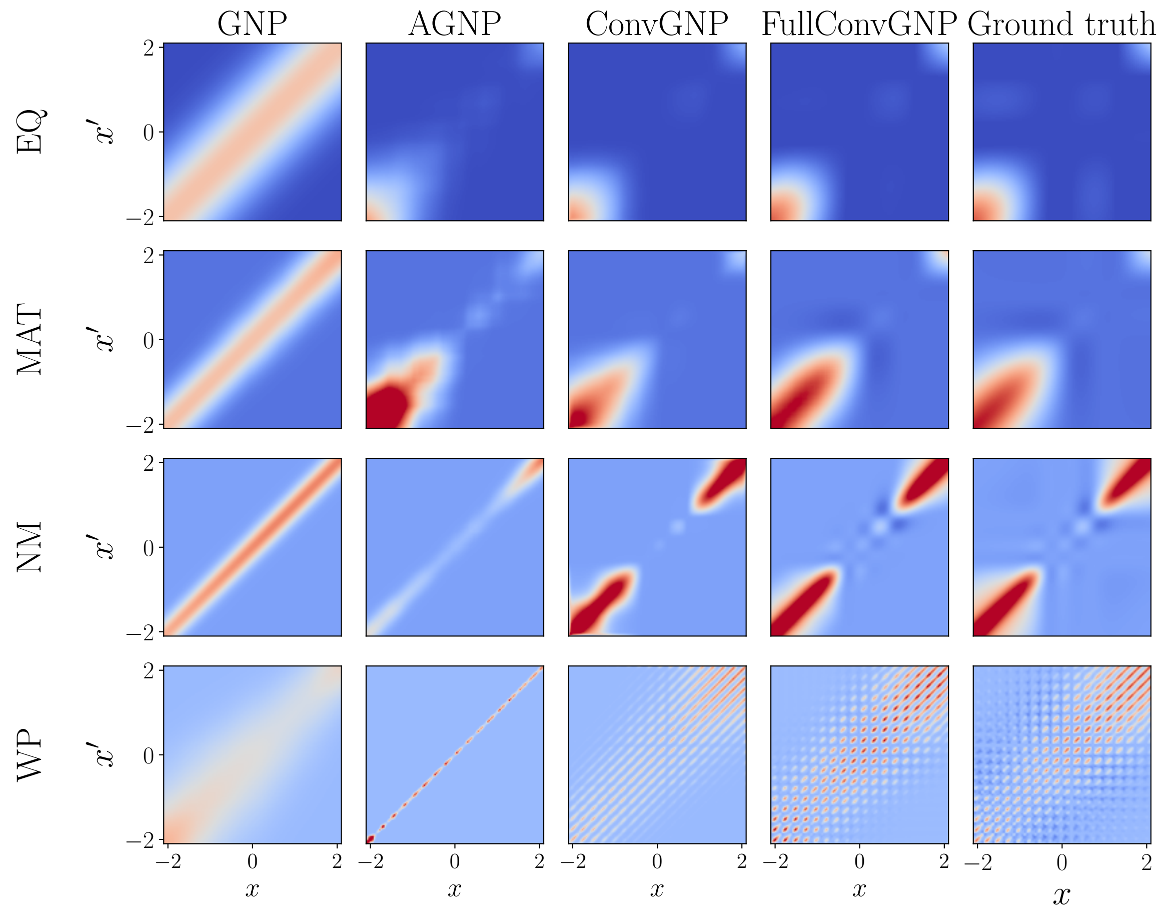

Figure 3.6: Plots of the predictive covariance of the GNP family, together with the covariance of the oracle GP, on the four tasks tested. In each task, a fixed context set is used for all of the models to facilitate comparison.

`Kvv` **outperformed** `linear`**:** We generally observed that the `kvv` models (■, ●, ▲) performed as well, and occasionally better than, their `linear` counterparts (■, ●, ▲). To test whether the `linear` models were limited by the number of basis functions D_g, we experimented with various settings $D_g \in \{16, 128, 512, 2048\}$. We did not observe a performance improvement for large D_g, suggesting that the models are not limited by this factor. This is surprising because, as $D_g \to \infty$ and assuming flexible enough f, g, and z, the `linear` models should, by Mercer's theorem, be able to recover any (sufficiently regular) GP posterior. From preliminary investigations, we leave open the possibility that the `linear` models might be more difficult to optimise and thus struggle to compete with `kvv`. We hope to conduct a more careful study on our training protocol in the future, to determine whether the training method can account for this performance gap, or whether the `kvv` model is fundamentally more powerful than the `linear` model.

Figure 3.5 shows samples drawn from the GNP models, from which we qualitatively observe that, like the FullConvGNP, the ConvGNP produces good quality function samples. These samples are consistent with the observed data, whilst maintaining uncertainty and capturing the behaviour of the underlying process. The ConvGNP is the only conditional model (other than the FullConvGNP) which produces high-quality posterior samples. Figure 3.6 shows plots of the models' covariances. Observe that, like the FullConvGNP, the ConvGNP is able to recover intricate covariance structure.

3.4 Modelling non-Gaussian joint distributions

One central limitation of GNPs, is the fact that they are not appropriate models for non-Gaussian data. In many modelling settings, such as environmental modelling, the ground truth distribution of the outputs may be far from Gaussian. For example, total daily precipitation is a non-negative quantity, whilst a Gaussian has full support on the real line. Thus GNP prediction maps may make poor predictions for non-Gaussian data, or be entirely inapplicable to downstream estimators which make assumptions such as non-negativity of the output.

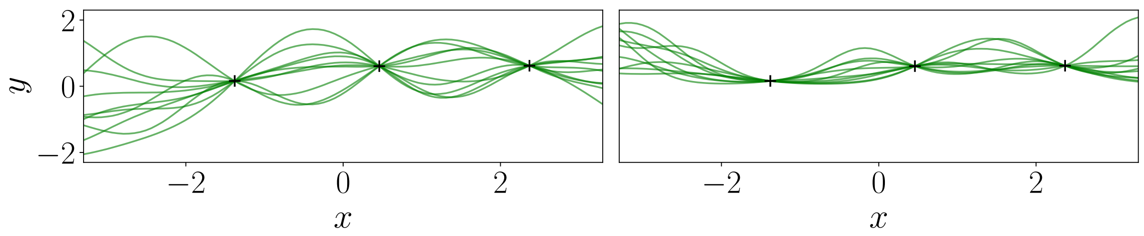

Figure 3.7: Plots of samples from a Gaussian (left) and non-Gaussian (right) posterior, conditioned on some data (black crosses) generated from a non-Gaussian process which is non-negative.

In order to address this limitation, whilst still effectively modelling output dependencies, we can apply marginal transformations to the output of a GNP. More specifically, suppose that $f : \mathbb{R} \to \mathbb{R}$ is a differentiable and invertible function. Then we can form a non-gaussian prediction map, by composing f with a GNP prediction map $\tilde{\pi}$, as

$$y_t = f(u_t), \text{ where } u_t \sim \tilde{\pi}(u_t; x_c, y_c, x_t), \tag{3.10}$$

where $u_t = (u_{t,1}, \dots, u_{t,M})$ denotes an intermediate multivariate Gaussian random variable, and f is applied elementwise to the entries of u_t. By choosing f appropriately, we can represent non-Gaussian prediction maps, whilst still modelling output dependencies, as illustrated in fig. 3.7. In this plot, a context set has been generated from a non-Gaussian and non-negative process, and a Gaussian (left) and non-Gaussian (right) process have been fitted to it, showing posterior samples in each case. By choosing an invertible and non-negative f, we can form a non-Gaussian prediction map which captures the non-negativity of the data, as reflected by the samples. In fig. 3.7, the transformation f was chosen to be the composition of a standard Gaussian CDF followed by a standard Gamma inverse CDF. This f adapts the marginals such that they are Gamma distributed, respecting the non-negativity constraint we know holds of the data. We note that a host of different, and even learnable functions f can be used.

Further, the log-likelihood of a prediction map of the form of eq. (3.10) can be computed in closed form, and can be trained via the exact log-likelihood. Since f is invertible and differentiable, we can use the change of variables formula (Kobyzev et al., 2020; Papamakarios et al., 2021) to compute the log-likelihood as

$$\log \pi(y_t; x_c, y_c, x_t) = \log \tilde{\pi}\left(f^{-1}(y_t); x_c, y_c, x_t\right) + \log \left| \det \frac{\partial u_t}{\partial y_t} \right|. \tag{3.11}$$

Since f is an elementwise transformation, the determinant term can be writteen as

$$\log \left| \det \frac{\partial u_t}{\partial y_t} \right| = \sum_{m=1}^{M} \log \left| \frac{du_{t,m}}{dy_{t,m}} \right|, \tag{3.12}$$

where $du_{t,m}/dy_{t,m}$ is a complete derivative of a univariate function and can thus be cheaply evaluated. Lastly, we can let f depend on a collection of parameters, outputted by the GNP, such as the shape and scale parameters of a Gamma distribution. Letting these parameters vary as a function of the target inputs can give more flexible, input dependent, marginals. We expect that marginal transformations should make GNPs even more competitive with NPs, which are able to represent non-Gaussian prediction maps by construction.

A detailed investigation on the theoretical and empirical properties of marginal transformations is pending. Studying the design choices that could be made in this space and evaluating them on synthetic and real data is a future direction of work.

3.5 Limitations of feature-based prediction maps

As we have demonstrated, feature maps can be used to parameterise equivariant predictive distributions, whilst modelling statistical dependencies in the output. In this way, feature-based ConvGNPs offer lower computational and memory alternative to the FullConvGNP. However, using feature maps introduces an important limitation on the prediction maps produced by the model, which we discuss here. On closer inspection

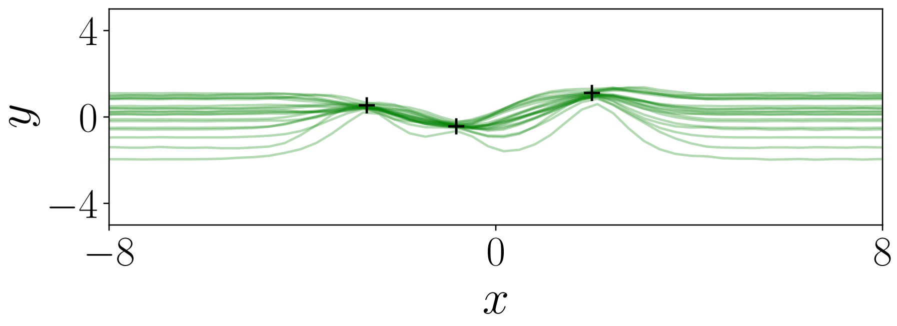

Figure 3.8: Illustration of the failure mode of feature-based equivariant maps. Even though the marginals of this ConvGNP have reasonable widths, samples from the prediction map become constant functions away from the context set, because of the saturation of the SetConv layer.

of samples drawn from ConvGNP models, we observe that the function samples often revert to a constant function far from the context set fig. 3.8. This behaviour is due to the fact that far from the context set, the outputs of the `SetConv` layer decay to zero. For any target input sufficiently far from the context set, the corresponding feature vector outputted by a ConvGNP reverts to a constant feature vector. If two distinct target inputs are both sufficiently far from the context set, their predictive covariance is approximately equal to their marginal variance, regardless of how far apart these points are from one another. Thus for any function sample drawn from such a predictive, the values of the function at any two such points are also approximately equal, explaining the observed behaviour. Clearly, this impacts the applicability of the ConvGNP, as we cannot expect it to produce sensible predictive posteriors far from the context set.

To further elucidate this issue, it is helpful to consider the special case where the context set is empty, that is when $x_c = ()$ and $y_c = ()$.[1] In this case, the output of the SetConv layer is zero everywhere. Thus the feature vector outputted by the ConvGNP is constant with respect to the target input, so samples from the posterior are constant functions. We may further ask whether this behaviour is specific to the ConvGNP. As lemma 3.5.1 suggests, this behaviour is common to all TE GNPs based on feature maps and cannot be avoided, for example by considering alternative architectures. Specifically, when the context set of a TE feature-based prediction map is empty, samples drawn from the prediction map are constant functions plus observation noise.

Lemma 3.5.1: Feature-based equivariant $\pi \implies$ constant prior

Suppose f is a translation-equivariant feature map, and consider a Gaussian prediction map whose mean and covariance are functions of f, that is

$$\pi(y_t; x_c, y_c, x_t) = \mathcal{N}(y_t; \mathbf{m}, \mathbf{K}), \text{ where } \mathbf{m} = m(\mathbf{f}), \text{ and } \mathbf{K} = k(\mathbf{f}) \tag{3.13}$$

for any appropriate m, k. Then whenever the context set is empty, $x_c = (), y_c = ()$, the prediction map satisfies

$$\pi(y_t; x_c, y_c, x_t) = \mathcal{N}\left(y_t; \mathbf{m}, \sigma^2 \mathbf{1}\mathbf{1}^\top + \sigma_n^2 \mathbf{I}\right). \tag{3.14}$$

Lemma 3.5.1 is proved in appendix A. The result follows by generalising the argument made for the ConvGNP above. More specifically, if a feature map f is TE, then whenever the context set is empty, the features that it outputs must be invariant to translations of the target input, and must therefore be constant. Thus the entries in the covariance matrix formed by any set of target inputs must be equal, up to an additive diagonal noise term. This covariance corresponds to a constant function plus observation noise. In this sense, feature-based TE prediction maps cannot learn the covariance structure any useful prior.

Returning to the case where the context is non-empty, we see that lemma 3.5.1 is closely related to the pathology which we observe. Far from the context set, any predictive map is expected to return to the prior produced by an empty context. For feature-based TE prediction maps, this prior corresponds to a constant function. This is all in contrast to the FullConvGNP, which does not depend on feature maps,

[1] Wessel Bruinsma provided a good deal of insight to this setting, and pointed out that lemma 3.5.1 holds.

and is capable of learning meaningful prior covariances. The FullConvGNP can make meaningful predictions even for empty context sets, or at locations far from the context.

Further, although lemma 3.5.1 is specific to TE Gaussian maps, similar results apply to other settings too. First, the discussion can be extended to maps with other kinds of equivariances. Second, it appears that any kind of posterior stochastic process which is parameterised by an equivariant feature map will run into the same issue, so this discussion applies to non-Gaussian prediction maps.

3.6 Addressing limitations of feature-based maps

Having better understood the reason behind this pathology, we may ask how it may be remedied. We have thus far been seeking a model with three appealing properties, namely that (1) the prediction map π is a function of a feature map f; (2) the feature map f is itself TE; (3) the prediction map π can represent meaningful priors, that is to work even with empty context sets. Lemma 3.5.1 tells us these three properties cannot hold simultaneously, so a natural question would be to consider whether we could give any of these up, whilst maintaining a useful model.

It appears that condition (3) can be partly given up to remedy this pathology. In particular, if we lift the requirement that the model should be able to handle an empty context set, $\mathcal{C} = \emptyset$, we may be able to resolve this issue. We can achieve this by modifying the ConvGNP model to work with augmented, context dependent, inputs. Suppose we $\mathcal{C}$ is a non-empty context set. We can first compute the mean of the context inputs

$$m_c = \sum_{n=1}^{N} x_{c,n}, \tag{3.15}$$

and use this to define a collection of K sinusoidal features, and evaluate these at the ConvGNP grid points

$$s_{kn} = \sum_{n=1}^{N} \sin\left(2\pi\omega_k^\top (g_n - m_c)\right), \tag{3.16}$$

where g are the grid point locations, given in fig. 2.4, and $\omega_1, \dots, \omega_K$ are K appropriately chosen frequency vectors. We can then use the features to augment the grid values h according to

$$\tilde{h}_n^{(1)} = [h_n^{(1)}, s_{1n}, \cdots, s_{Kn}]. \tag{3.17}$$

These features are TE, so the composition of this augmentation with the ConvGNP is also TE. In this way, even though the $h^{(1)}$ features decay to 0 far from the context set, the s_{nk} features do not. Therefore, the features outputted by the model will not decay to a constant vector as they do in regular feature-based ConvGNP maps. We intend to explore the feasibility of this method in future work.

3.7 Chapter summary

In this section we introduced feature-based GNPs, which constitute a scalable alternative to the FullConvGNP of Bruinsma et al. (2021). These models rely on features to parameterise the mean and covariance of a prediction map, without requiring expensive $2D_x$ convolutions, and thus significantly lower computational and memory requirements, especially in larger input dimensions. Feature-based GNPs can be readily combined with a host of existing network architectures, giving rise to models such as the GNP, the AGNP or the ConvGNP. A variant of the ConvGNP model competes with the FullConvGNP and the latent ConvNP model on a host of synthetic tasks.

We provided a practical method for modifying GNPs based on marginal transformations, to model non-Gaussian data more effectively. The change of variables formula allows us to train models with non-Gaussian outputs without approximations, via the same maximum-likelihood procedure as GNPs. Lastly, we examined a central limitation of translation equivariant feature-based prediction maps. The prior corresponding to these maps is necessarily a distribution over constant functions, regardless of the particular choice of architecture used to parameterise the GNP. We discussed a practical method for remedying this limitation on the ConvGNP, based on augmenting the grid values using sinusoidal features, however the effectiveness of this approach is yet to be empirically established.

Chapter 4

Equivariant Neural Processes

4.1 Generalisation and Equivariance

Despite their favourable flexibility and scalability, CNPs can fail disastrously when conditioning on data or when making predictions outside the training distribution. Similarly to a host of other probabilistic models, one plausible direction to improve the generalisation performance of CNPs is to choose amortisation functions which introduce inductive biases appropriate for the data at hand. Whilst heuristics such as attention (Kim et al., 2019) have proved useful for improving certain models, more principled approaches based on the notion of equivariance (Cohen and Welling, 2016a), which we have already touched upon, have proved highly fruitful in recent works.

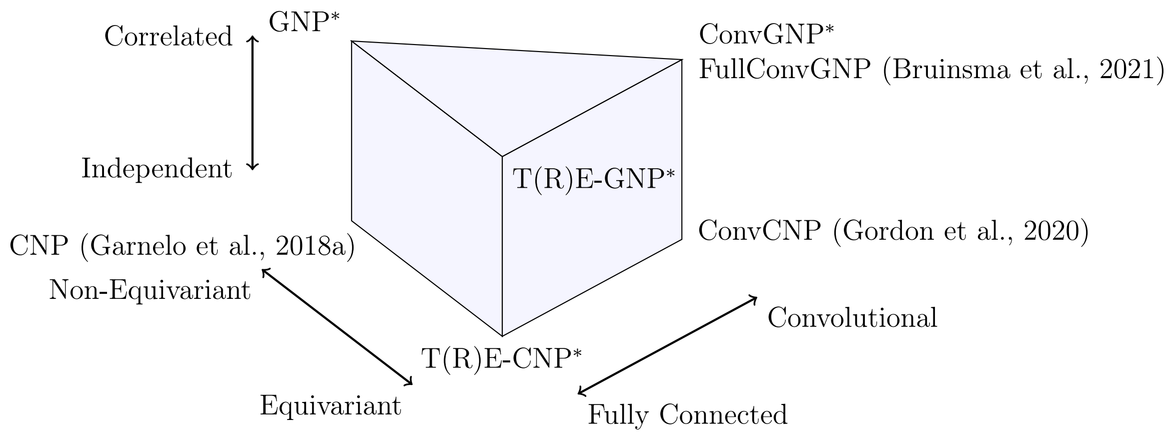

Figure 4.1: An illustration of three orthogonal design choices for CNPs. Starred models are introduced either in Markou et al. (2021) or in this report.

In many problems of interest, we may know that the process which generated the data exhibits certain invariances. When modelling stationary processes for example, the prior is invariant to translations, meaning that a priori, we do not expect the process to behave differently in different regions of the input space. In this case, translation invariance in the prior corresponds to translation equivariance in the posterior (Foong et al., 2020). For point clouds or vector data such as fluid flows and force fields, we may expect the prior process to look the same under not only translations, but also rotations or reflections of the coordinate system, that is we may assume a translation and rotoreflection invariant prior. This invariance in turn implies a translation and rotoreflection equivariant posterior. More generally, arbitrary invariances in the prior imply corresponding equivariances in the posterior. Building appropriate equivariances directly into a CNP via the architectures of its amortisation networks, is likely to improve the generalisation performance of the model. An equivariant model can accurately generalise to data which has been transformed to lie outside its training distribution, via a transformation with respect to which it is equivariant.

Gordon et al. (2020) introduced the ConvCNP, a model using Convolutional Neural Networks (CNNs) to achieve translation equivariance, which was shown to be an extremely useful inductive bias in applications ranging from time series to image completion. Extending this work, Holderrieth et al. (2021) considered a variant of the ConvCNP which used Steerable Convolutional Neural Networks (SteerCNNs) to achieve rotation and reflection (rotoreflection) equivariance, demonstrating performance improvements over non-equivariant models on weather modelling tasks. Along a similar vain, the work of Kawano et al. (2021) generalises equivariance in CNPs to arbitrary group equivariances, again relying on convolutional architectures.

Extending CNPs with appropriate equivariances improves generalisation performance, while reducing the number of parameters in the model via parameter tying. Unfortunately, however, the aforementioned approaches all rely on convolutions, whose computational and memory cost scales exponentially with the input dimension. Whilst convolutions may not present an issue for a single input dimension, both the computational and memory costs become significant for two and three dimensional inputs, whilst the support of higher-dimensional convolutions in deep learning libraries is highly limited. In this chapter, we will consider alternative classes of equivariant CNPs, summarised schematically in fig. 4.1, either translation or both translation and rotoreflection equivariant, which do not rely on convolutions and can be significantly cheaper to scale to higher dimensions. In particular, we will see that convolutions are not necessary in order to achieve translation and rotoreflection equivariance. In this way, as fig. 4.1 suggests, we can regard convolutions and equivariance as orthogonal choices when designing CNPs.

4.2 Equivariant CNPs without the Convolutions

Motivated by the aforementioned benefits of equivariance for generalisation, and aware of the computational and memory costs incurred by convolutional architectures, we now discuss approaches for convolution-free equivariance. In section 2.3.6 we introduced three kinds of equivariances for predictive maps, which we now turn to, starting from translation equivariance. Intuitively, we expect that a TE model should depend only on the differences between context and target inputs rather than their absolute positions. This intuition leads us to state and prove lemma 4.2.1.

Lemma 4.2.1: Translation equivariant prediction maps

A prediction map π is translation equivariant if and only if there exists another function π^* such that

$$\pi(y_t; x_c, y_c, x_t) = \pi^*\Big(y_t; (x_{c,n} - x_{c,n'}, y_n)_{n,n'=1}^{N,N}, (x_{t,m} - x_{c,n})_{m,n=1}^{M,N}\Big).$$

This lemma, proved in appendix A, states that the class of prediction maps which depend on pairwise differences are all TE, whilst any TE map can be expressed as a function which takes pairwise differences, rather than absolute inputs, as inputs. This includes the ConvCNP and ConvGNP models, whose predictions depend entirely on the differences between the target points and the grid points, where the latter in turn depend entirely on the context points.

We expect that parameterising the map from outputs and pairwise differences of inputs to predictive distributions, using a sufficiently flexible neural network such as a DeepSet (Zaheer et al., 2017), should be able to reasonably approximate the ground truth TE posterior. By combining a model with this input parameterisation, together with either a mean-field output (TECNP) or a fully correlated output (TEGNP), we obtain TE models which are entirely convolution-free and thus easier to scale to higher dimensions. Preliminary experiments suggest that a TEGNP with a comparable number of parameters as a ConvGNP, and an identical training procedure on one-dimensional data, exhibit comparable performance. However, TEGNPs remain significantly easier to scale to higher dimensions requiring smaller memory and computational resources at training time, as well as at inference time.

We can extend this discussion to rotoreflection equivariance (RE), and characterise prediction maps which are both translation as well as rotoreflection equivariant (TRE). Extending our intuition about TE, we focus on TRE for scalar prediction maps, that is maps which model a single dimensional output variable $D_y = 1$, and satisfy both eqs. (2.16) and (2.19). We show that a necessary and sufficient condition for a *scalar* CNP to be translation and rotoreflection equivariant is that π should depend on the context and target inputs solely via *pairwise distances* in the Euclidean norm.

Lemma 4.2.2: E(n) equivariant scalar prediction maps

A scalar prediction map π is translation and rotoreflection equivariant if and only if there exists another function π^* such that

$$\pi(y_t; x_c, y_c, x_t) = \pi^*\Big(y_t; ((||x_{c,n} - x_{c,n'}||_2, y_n))_{n,n'=1}^{N,N}, (||x_{t,m} - x_{c,n}||_2)_{m,n=1}^{M,N}\Big).$$

Prediction maps of the form in lemma 4.2.2, which is itself proved in appendix A, can be seen as special cases of TE maps, which in addition are rotoreflection equivariant. Intuitively, Euclidean pairwise distances retain no information about neither the absolute position or orientation of the context and target sets. Again, we can parameterise a map from outputs and pairwise distances of inputs to predictive distributions using a neural network, and relying on the flexibility of this network to learn good prediction maps. As before, we can combine this input transformation with either a mean-field output (TRECNP) or a fully correlated output layer (TREGNP).

The last case of interest is that of TRE prediction maps for vector outputs, where $D_x = D_y$. In this case, we would like the prediction map to rotate and reflect according to eq. (2.20). To achieve this we consider TRE feature maps, which we can use to parameterise prediction maps, as demonstrated for the ConvGNP. By making the feature maps themselves TRE, the prediction maps that these will parameterise also will be TRE. The following lemma gives a necessary and sufficient condition for TRE feature maps which, unlike mappings based on Steerable CNNs (Cohen and Welling, 2016b) or Group Equivariant CNNs (Cohen and Welling, 2016a), do not necessarily rely on convolutions.

Lemma 4.2.3: E(n) equivariant feature maps

A feature map f is translation and rotoreflection equivariant if and only if it can be written as

$$f(x_c, y_c, x_t)_m = \sum_{n=1}^{N} \left[g(x_c, y_c, x_t)_{m,n}\ (x_{t,m} - x_{c,n}) + h(x_c, y_c, x_t)_{m,n}\ y_n\right] \tag{4.1}$$

where g, h are scalar maps, which are invariant to translations and rotoreflections.

This lemma is also proved in appendix A. The features f_m are unaffected by translations of the context and target set, but when a rotoreflection is applied to the context set and target inputs, f_m rotates and reflects accordingly. Prediction maps parameterised using feature maps of this form will also be TRE. We can use translation and rotoreflection invariant neural networks g and h to parameterise equivariant maps of the form in lemma 4.2.3, again relying on the flexibility of these models to learn accurate prediction maps.

We note that the approaches suggested here bear resemblance to other methods which exist in different contexts in the literature. Recently, Pfaff et al. (2020) used a method based on pairwise differences for parameterising TE Graph Neural Networks (GNNs) in the context of mesh-based simulations, demonstrating impressive results. Satorras et al. (2021) used similar reparameterisations of the inputs to a GNN for achieving E(n)-equivariance, using a method which they refer to as equivariant message passing (EMP). With slight modifications, EMP can be modified to parameterise equivariant feature maps of the form of lemma 4.2.3, presenting an interesting alternative to DeepSet based parameterisations of TRECNPs.

We note that a considerable difference between the methods of Pfaff et al. (2020) and Satorras et al. (2021) and those presented here, is that the former are applied to the supervised setting, where an input graph is mapped to equivariant features, whilst the latter are applied to the meta-learning setting, where entire context and target set are mapped to equivariant features. Nevertheless, there appears to be a connection between CNPs and GNNs, whereby CNPs can be viewed as fully connected GNNs, with certain additional constraints to ensure the prediction map is consistent under permutations and marginalisations. This connection warrants further research, and may yield interesting and useful insights.

4.3 Chapter summary

In this chapter we considered equivariant CNPs. We discussed how equivariance can help improve the generalisation performance of CNPs and GNPs, as well as how current methods depend on expensive convolutions. Focusing on TE and TRE models with scalar and vector outputs, we saw how alternative parameterisations of equivariant maps can be used to improve the computational and memory costs of equivariant CNPs and GNPs. Instead of relying on convolutions, the proposed methods use input parameterisations to ensure the output of the GNP is equivariant with its input. These parameterisations are not only sufficient for equivariance but also necessary, meaning that every TE or TRE prediction map can be written in these forms. Augmenting CNP and GNP models with TE or TRE gives rise to a host of corresponding models, such as the T(R)ECNP and T(R)EGNP models. These variants can be combined with the models introduced in the previous chapter, including non-Gaussian output parameterisations. Preliminary experiments using the TEGNP have yielded promising results, encouraging a complete evaluation of these methods.

Chapter 5

Conclusion, Future Work & Timeline

5.1 Conclusion

In this report we have explored how meta-algorithms can be used to jointly model multiple datasets, and rapidly adapt to new ones. Meta-algorithms can be designed such that making predictions on unseen datasets requires only modest amounts of additional computational and memory resources. Bayesian hierarchical models are a sensible starting point for designing such meta-algorithms, however, performing inference and learning in these models introduces additional challenges. This work has argued that if our focus is on making predictions rather than on inferring latent variables, then approaches based on point estimates of latent variables may be a sensible choice. These approaches circumvent the issues stemming from approximate inference, whilst still being able to make accurate predictions.

Following the above reasoning, this work has focused on CNPs, a family of meta-models which are flexible and cheap at inference time. They can be trained via an exact maximum likelihood procedure, thereby avoiding the challenges associated with approximate inference. However, existing CNPs do not model statistical dependencies in the output, which are often necessary for downstream estimation tasks. Whilst latent Neural Processes do model output dependencies, their use of a latent variable complicates training since exact maximum-likelihood training is not possible. On the other hand, the FullConvGNP of Bruinsma et al. (2021) can model output dependencies and is trainable via an exact maximum-likelihood procedure, but is difficult to scale to data of more than one input dimension. To address this issue, we introduce feature-based GNPs, a variant of the CNP model, which can be extended with attentive (AGNP) or convolutional architectures (ConvGNP). The proposed family of models relies on feature maps to compute the covariances of the target outputs, allowing them to be scaled to higher dimensions, whilst maintaining other favourable qualities of the FullConvGNP. The ConvGNP, which is the GNP counterpart of the ConvCNP model of Gordon et al. (2020), has shown performance competitive to that of the FullConvGNP, whilst remaining scalable to higher input dimensions.

We investigated alternative approaches for parameterising covariances using feature maps, referred to as `linear` and `kvv`. We generally found that the former outperformed the latter. Moving beyond Gaussian prediction maps, we suggested how the output of the ConvGNP can be composed with an invertible transformation, to model non-Gaussian data. The resulting models can still be trained via the same exact marginal likelihood procedure, but is likely to be more suitable for non-Gaussian data.

5.2 Future work

From the work presented in previous chapters and the summary above, we identify the following points as promising lines for future work.

Comparing covariance parameterisations: When comparing the `linear` and `kvv` covariances in feature-based GNPs, we observed that the choice of the covariance function affected the predictive performance of the model. We suspect the observed difference is because the covariance matrices produced by `kvv` are full rank, whereas those produced by `linear` are low-rank. However, we have not produced a satisfactory explanation for this behaviour, and a more detailed investigation into this matter would be a useful line of work. Further, there are many choices beyond `linear` and `kvv` for parameterising covariances, such as taking a sum of `kvv` covariances computed with different features. A methodical evaluation of the available choices would also be highly useful towards informing the design of GNPs.

Marginal transformations: We have suggested that GNP models can be composed with invertible transformations, to model non-Gaussian data. The invertible transformation adjusts the marginals of the predictive process, whilst the Gaussian correlations computed prior to applying the transformation account for statistical dependencies in the output. Further development of and investigation into these models is required for them to come to fruition. Fleshing out and evaluating these models on synthetic as well as real data would be another highly useful line of future research.

Structured prediction: One additional direction in which GNPs could be extended, which is in similar vain to marginal transformations, is that of structured prediction. In many cases, we expect that the output marginals of the data distribution may exhibit more intricate structure than a simple Gaussian, or invertible transformation thereof. In precipitation modelling for example, it is often the case that records either show zero precipitation on a given day. More intricate behaviour can be captured via correspondingly more intricate predictives, such as the Bernoulli-Gamma mixture predictive used in Vaughan et al. (2021b). Further research into modifying the models presented here with additional structure, based on expert knowledge of the task at hand, is one possible direction for future investigation, presenting exciting opportunities for designing more effective CNPs.

Equivariance: Whilst this work has presented some theoretical results for and preliminary demonstrations of equivariant CNPs, a detailed investigation of these methods is lacking. In particular, we have argued that the modelling power of equivariant feature-based prediction maps is fundamentally limited. However, this limitation may be addressed by lifting the requirement that the prediction map should be able to handle the empty context set, and introducing auxiliary features, such as sinusoids. Investigating potential limitations of this approach from both a theoretical as well as empirical standpoint would be highly informative about the extent to which equivariant feature-based can model data effectively. Further, recognising the fact that convolutions impose severe limitations on the scalability of CNPs, we have provided a number of necessary and sufficient conditions for equivariant prediction and feature maps, which can be used to design convolution-free equivariant CNPs. A practical implementation and evaluation of these models is a very interesting and useful line of work.

Applications to environmental data: One of the applications which has been used to motivate ConvGNPs throughout this work has been environmental modelling. We believe this is an ideal setting for the application of ConvGNPs, since there is a great deal of available data as well as weather simulators (Hersbach et al., 2018b), which can be used to train meta-models. Initial steps along this line of work by Vaughan et al. (2021a) and Vaughan et al. (2021b), has used ConvCNPs and ConvNPs to model weather data, demonstrating competitive performance when evaluated against strong baselines. These results, together with the availability of data and weather simulators suggest that weather modelling is a promising application for ConvGNPs. We also expect investigations into marginal transformations and structured prediction to complement environmental modelling well, since real world data are expected to be non-Gaussian in general, and possibly involve constraints such as non-negativity which are not respected by Gaussian prediction maps.

Further applications: One especially promising application for meta-algorithms, which has been identified in this work, is sim-to-real. Having access to a simulator which bears rough resemblance to the real world, would allow us to generate synthetic datasets with which we can train meta-models. Generating several such datasets using different settings for the various simulation parameters, Applications such as modelling of robotic

5.3 Execution timeline

Having identified six points for future work, we outline a tentative execution timeline for them. At the time of writing, in collaboration with James Requeima, Wessel Bruinsma and Prof. Richard Turner, we are investigating alternative covariance parameterisations, marginal transformations, applications in weather modelling and equivariant parameterisations. Naturally, the first three fit into a single investigation which we hope to complete and turn into a conference paper, to be submitted in October 2021. We are also currently investigating equivariant CNPs and GNPs, further developing the theory and implementation of the models presented in this work, with the aim of submitting a proof-of-concept workshop paper, also around the autumn of 2021.

This leaves work into structured prediction and sim-to-real for future work. We regard the former as a useful practical extension for GNP models, and we intend to investigate it in the coming months, possibly in conjunction with environmental modelling applications. The latter constitutes a highly interesting and promising application for meta-models, which deserves its own separate treatment. To this end, we aim to identify a set of appropriate sim-to-real tasks which will highlight the merits of meta-models, and apply the methods which we are currently developing on these tasks. Learning from simulated data in higher dimensional input settings is likely to benefit from investigations into convolution-free equivariant models, because these require substantially lower computational and memory resources compared to convolution-based models. We aim to pick up this line of research towards the end of 2021, carrying over into the new year.

References

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Bronskill, J. (2020). *Data and computation efficient meta-learning.* PhD thesis, University of Cambridge.

Bruinsma, W. P., Requeima, J., Foong, A. Y. K., Gordon, J., and Turner, R. E. (2021). The gaussian neural process.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Cohen, T., Geiger, M., and Weiler, M. (2018). A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017.*

Cohen, T. and Welling, M. (2016a). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

Cohen, T. S. and Welling, M. (2016b). Steerable cnns.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Edwards, H. and Storkey, A. (2016). Towards a neural statistician. *arXiv preprint arXiv:1606.02185.*

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks.

Finn, C., Xu, K., and Levine, S. (2019). Probabilistic model-agnostic meta-learning.

Foong, A. Y. K., Bruinsma, W. P., Gordon, J., Dubois, Y., Requeima, J., and Turner, R. E. (2020). Meta-learning stationary stochastic process prediction with convolutional neural processes.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. (2018a). Conditional neural processes. *CoRR*, abs/1807.01613.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. (2018b). Neural processes. *CoRR*, abs/1807.01622.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E. (2018). Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921.*

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. (2020). Convolutional conditional neural processes.

Gordon, J., Lopez-Paz, D., Baroni, M., and Bouchacourt, D. (2019). Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations.*

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J.-N. (2018a). Era5 hourly data on pressure levels from 1979 to present. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS).*

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J.-N. (2018b). Era5 hourly data on single levels from 1979 to present.

Holderrieth, P., Hutchinson, M., and Teh, Y. W. (2021). Equivariant learning of stochastic fields: Gaussian processes and steerable conditional neural processes.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: A survey.

Jaynes, E. T. (2003). *Probability theory: The logic of science.* Cambridge university press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Kawano, M., Kumagai, W., Sannai, A., Iwasawa, Y., and Matsuo, Y. (2021). Group equivariant conditional neural processes. *CoRR*, abs/2102.08759.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, S. M. A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019). Attentive neural processes. *CoRR*, abs/1901.05761.

Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114.*

Kobyzev, I., Prince, S., and Brubaker, M. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Le, T. A., Kim, H., Garnelo, M., Rosenbaum, D., Schwarz, J., and Teh, Y. W. (2018). Empirical evaluation of neural process objectives. In *NeurIPS workshop on Bayesian Deep Learning*.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

MacKay, D. J. (1992). The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736.

Mackay, D. J. C. (1992). *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology.

Markou, S., Requeima, J., Bruinsma, W., and Turner, R. (2021). Efficient gaussian neural processes for regression. *ICML Uncertainty Robustness in Deep Learning*.

Minka, T. P. (2013). Expectation propagation for approximate bayesian inference. *arXiv preprint arXiv:1301.2294*.

Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms.

Opper, M. and Winther, O. (1998). A bayesian approach to on-line learning. *On-line learning in neural networks*, pages 363–378.

Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X., and Roy, B. V. (2021). Epistemic neural networks.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vaughan, A., Lane, N. D., and Herzog, M. (2021a). Multivariate climate downscaling with latent neural processes.

Vaughan, A., Tebbutt, W., Hosking, J. S., and Turner, R. E. (2021b). Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development Discussions.*

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. (2017). Deep sets. *CoRR*, abs/1703.06114.

Appendix A

Lemmas on equivariant maps

Feature-based equivariant $\pi \implies$ constant prior

Here we prove lemma 3.5.1, which states that any feature-based Gaussian prediction map corresponds to a constant prior.

Lemma: Feature-based equivariant $\pi \implies$ constant prior

Suppose f is a translation-equivariant feature map, and consider a Gaussian prediction map whose mean and covariance are functions of f, that is

$$\pi(y_t; x_c, y_c, x_t) = \mathcal{N}(\mathbf{m}, \mathbf{K}), \text{ where } \mathbf{m} = m(\mathbf{f}), \text{ and } \mathbf{k} = K(\mathbf{f}) \tag{A.1}$$

for any appropriate m, k and $\mathbf{f} = f(y_t; x_c, y_c, x_t)$. Then whenever the context set is empty, $x_c = (), y_c = ()$, the prediction map corresponds to a constant function

$$\pi(y_t; x_c, y_c, x_t) = \mathcal{N}\left(\mathbf{m}, \sigma^2 \mathbf{1}\mathbf{1}^\top + \sigma_n^2 \mathbf{I}\right). \tag{A.2}$$

Proof: Suppose f is a translation equivariant feature map. Suppose further that the context set is empty, that is $x_c = ()$ and $y_c = ()$. Then

$$\begin{aligned} f(y_t; x_c, y_c, x_t) &= f(y_t; Tx_c, y_c, Tx_t), \\ &= f(y_t; x_c, y_c, Tx_t), \end{aligned}$$

for any translation T. Since any target input can be expressed as a translation of another target input, the feature map f must be constant whenever the context set is empty. If the feature map is constant and equal for all target inputs, the covariance between any two target points is equal to their marginal variances, up to an additive diagonal noise term. □

As has been noted in the main text, similar results can be shown for predictive maps with equivariances other than translations. Further, these results can be extended for non-Gaussian predictive maps which are parameterised by feature maps.

Necessary and sufficient conditions for equivariance

In this section we prove Lemmas 4.2.1, 4.2.2 and 4.2.3, made in the main text, regarding the equivariance of prediction and feature maps.

Lemma: Translation equivariant prediction maps

A prediction map π is translation equivariant if and only if there exists another prediction map π^* such that

$$\pi(y_t; x_c, y_c, x_t) = \pi^*(y_t; ((x_{c,n} - x_{c,n'}, y_n))_{n,n'=1}^{N,N}, (x_{t,m} - x_{c,n})_{m,n=1}^{M,N}). \tag{A.3}$$

Proof: If there exists π^* such that eq. (A.3) holds, then π is translation invariant because π^* is. Conversely, if π is translation invariant, we can write a function π^* in the form of eq. (A.3), in the following way. Let T_1 denote the map

$$T_1(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{n=1}^M) = (((x_{c,n} - x_{c,n'}, y_n))_{n,n'=1}^{N,N}, (x_{t,m} - x_{c,n})_{m,n=1}^{M,N}),$$

and T_1^{-1} denote the pre-image map of T_1. Then the set

$$\mathcal{S}_1 = T_1^{-1} T_1(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{n=1}^M),$$

consists of all pairs of context sets and target input sets obtained by applying an arbitrary translation to both $((x_{c,n}, y_n))_{n=1}^N$ and $(x_{t,m})_{n=1}^M$. Now, let $\tilde{T}_1^{-1}$ denote the arbitrary choice of an element from this pre-image, rather than the whole pre-image. Since π is translation invariant, the map

$$\pi^* = \pi \circ \tilde{T}_1^{-1}, \tag{A.4}$$

satisfies eq. (A.3), arriving at the result. □

In the argument above, we saw how T_1 maps the context and target sets to new sets in which all translational information has been removed by taking pairwise differences of the inputs. Picking an element from the pre-image of this transformed set reconstructs the original dataset, up to arbitrary translations. If π is assumed to be translation invariant, then it is not affected by such arbitrary translations, providing a way to write down a π^* of the form specified in the lemma. The proof of the next lemma reproduces this argument for translation and rotoreflection invariant prediction maps.

Lemma: E(n) equivariant scalar prediction maps

A scalar ($D_y = 1$) prediction map π is translation and rotoreflection equivariant if and only if there exists another prediction map $\pi^* : \mathcal{Z}_{xy} \times \mathcal{Z}_x \to \mathcal{Q}^1$ such that

$$\begin{aligned}\pi(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{n=1}^M) &= \\ &= \pi^*(((||x_{c,n} - x_{c,n'}||_2, y_n))_{n,n'=1}^{N,N}, (||x_{t,m} - x_{c,n}||_2)_{m,n=1}^{M,N}). \qquad \text{(A.5)}\end{aligned}$$

Proof: If there exists π^* such that eq. (A.5) holds, then π is translation and rotoreflection invariant because π^* is rotoreflection invariant. Conversely, if π is translation and rotoreflection invariant, we can write down a function π^* which satisfies eq. (A.5) in the following way. Define the map T_2 as

$$T_2(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{n=1}^M) = (((||x_{c,n} - x_{c,n'}||, y_n))_{n,n'=1}^{N,N}, (||x_{t,m} - x_{c,n}||)_{m,n=1}^{M,N}). \quad \text{(A.6)}$$

and let T_2^{-1} be the pre-image of T_2. Then the set

$$\mathcal{S}_2 = T_2^{-1} T_2(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{n=1}^M),$$

consists of all pairs of context sets and target inputs obtained by applying an arbitrary rotoreflection followed by an arbitrary translation on $((x_{c,n}, y_n))_{n=1}^N$ and $(x_{t,m})_{n=1}^M$. Now, let $\tilde{T}_2^{-1}$ denote the arbitrary choice of an element from this pre-image, rather than the whole pre-image. Since π is translation equivariant, the map

$$\pi^* = \pi \circ \tilde{T}_2^{-1}, \qquad \text{(A.7)}$$

satisfies eq. (A.5), arriving at the result. □

Lemma: E(n) equivariant feature maps

Let $D = D_1 = D_2$ and $f : \mathcal{Z}_{xy} \times \mathcal{Z}_x \to \mathbb{R}^{\cdot \times D}$ be a feature map. Then f is translation, rotoreflection equivariant if and only if it can be written as

$$f(Z_{xy}, Z_x)_m = \sum_{n=1}^{N} [g(Z_{xy}, Z_x)_{m,n} \ (x_{t,m} - x_{c,n}) + h(Z_{xy}, Z_x)_{m,n} \ y_n] \tag{A.8}$$

where $g, h : \mathcal{Z}_{xy} \times \mathcal{Z}_x \to \mathbb{R}^{\cdot \times 1}$ are scalar feature maps, which are invariant to translations and rotoreflections.

Proof: If f has the above form, then it is translation and rotoreflection equivariant. Conversely, suppose f is translation and rotoreflection equivariant: we will show that f can be expressed in the form of eq. (A.8). First, we show that f must satisfy

$$f_m \in S := \mathrm{span}((x_{t,m} - x_{c,1}), \ldots, (x_{t,m} - x_{c,n}), y_1, ..., y_n), \tag{A.9}$$

via the following argument. If $S = \mathbb{R}^D$ then $f_m \in S$. Now suppose $S \neq \mathbb{R}^D$ and $f_m \notin S$. Then there exists a subspace W with $\dim W = D - 1$, such that $S \subset W$ and $f_m \notin W$. Using W we can define the hyperplane $x_{c,1} + W$, and note that reflections across this hyperplane leave Z_{xy} and Z_x invariant, but do not leave f_m invariant. This leads to a contradiction and therefore we have $f_m \in S$.

Hence f_m can be written in the form given in eq. (A.8) for appropriately chosen $g_{m,n}$ and $h_{m,n}$. We now show that $g_{m,n}$ and $h_{m,n}$ can be chosen such that they are translation and rotoreflection invariant, while satisfying eq. (A.8). Define T_3 as the map

$$T_3(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{m=1}^M) = (((||z_{c,n} - z_{c,n'}||_2)_{n,n'=1}^{2N,2N}, (||x_{t,m} - z_{c,n}||_2)_{m,n=1}^{M,2N}), \tag{A.10}$$

where the vectors $z_{c,n}$ are defined as

$$z_{c,n} = \begin{cases} x_{c,n} & n = 1, \ldots, N \\ x_{c,n} + y_{c,n} & n = N+1, \ldots, 2N, \end{cases} \tag{A.11}$$

and let T_3^{-1} be the pre-image of T_3. Then the set

$$\mathcal{S}_3 = T_3^{-1} T_3(((x_{c,n}, y_n))_{n=1}^N, (x_{t,m})_{m=1}^M),$$

consists of all pairs $(Z_{xy} \in \mathcal{Z}_{xy}, Z_x \in \mathcal{Z}_x)$ obtained by applying an arbitrary rotoreflection followed by an arbitrary translation on $((x_{c,n}, y_n))_{n=1}^N$ and $(x_{t,m})_{n=1}^M$. In a similar way to the earlier proofs, let $\tilde{T}_3^{-1}$ denote the arbitrary choice of an element from the pre-image and consider the maps

$$g^* = g \circ \tilde{T}_3^{-1} \tag{A.12}$$

$$h^* = h \circ \tilde{T}_3^{-1}, \tag{A.13}$$

which are both translation and rotoreflection invariant, because $g_{m,n}$ and $h_{m,n}$ are. Then the map

$$f_m^* = \sum_{n=1}^{N} \left[g_{m,n}^* \, (x_{t,m}^* - x_{c,n}) + h_{m,n}^* \, y_n \right] . \tag{A.14}$$

is equal to f and is of the form stated in the lemma. □