Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes

Andrew Y. K. Foong*
University of Cambridge
ykf21@cam.ac.uk

Wessel P. Bruinsma*
University of Cambridge
Invenia Labs
wpb23@cam.ac.uk

Jonathan Gordon*
University of Cambridge
jg801@cam.ac.uk

Yann Dubois
Facebook AI Research
yannd@fb.com

James Requeima
University of Cambridge
Invenia Labs
jrr41@cam.ac.uk

Richard E. Turner
University of Cambridge
Microsoft Research
ret26@cam.ac.uk

Abstract

Stationary stochastic processes (SPs) are a key component of many probabilistic models, such as those for off-the-grid spatio-temporal data. They enable the statistical symmetry of underlying physical phenomena to be leveraged, thereby aiding generalization. Prediction in such models can be viewed as a *translation equivariant* map from observed data sets to predictive SPs, emphasizing the intimate relationship between stationarity and equivariance. Building on this, we propose the Convolutional Neural Process (ConvNP), which endows Neural Processes (NPs) with translation equivariance and extends convolutional conditional NPs to allow for dependencies in the predictive distribution. The latter enables ConvNPs to be deployed in settings which require coherent samples, such as Thompson sampling or conditional image completion. Moreover, we propose a new maximum-likelihood objective to replace the standard ELBO objective in NPs, which conceptually simplifies the framework and empirically improves performance. We demonstrate the strong performance and generalization capabilities of ConvNPs on 1D regression, image completion, and various tasks with real-world spatio-temporal data.

1 Introduction

Incorporating appropriate inductive biases into machine learning models is key to achieving good generalization performance. Consider, for example, predicting rainfall at an unseen test location from rainfall measurements nearby. A powerful inductive bias for this task is *stationarity*: the assumption that the generative process governing rainfall is spatially homogeneous. Given only observations in a limited part of the space, stationarity allows the model to extrapolate to yet unobserved regions. Closely related to stationarity is *translation equivariance* (TE). TE formalizes the intuitive idea that if observations are shifted in time or space, then the resulting predictions should be shifted by the same amount. When stationarity or TE is appropriate, e.g. in time-series [36], images [27], and spatio-temporal modelling [10, 9], incorporating them into our models yields significant benefits.

A general framework for these tasks is to view them as prediction of a *stochastic process* (SP; [37]). This principled approach has inspired a new set of deep learning architectures that bring the expressivity and fast test-time inference of deep learning to SP modelling. *Conditional Neural Processes* (CNPs; [12]) use neural networks to directly parameterize a map from data sets to predictive

*Authors contributed equally.

Preprint. Under review.

SPs, which is trained via meta-learning [39, 44]. However, CNPs suffer from several drawbacks that inhibit their use in scenarios where other SP models, e.g. Gaussian processes (GPs; [34]), often succeed. First, vanilla CNPs cannot account for TE as an inductive bias. This was recently addressed with the introduction of ConvCNPs [15]. Second, both CNPs and ConvCNPs are limited to factorized, parametric predictive distributions. This makes them unsuitable for producing coherent predictive function samples or modelling complicated likelihoods. *Neural Processes* (NPs; [13]), a latent variable extension of CNPs, were introduced to enable richer joint predictive distributions. However, the NP training procedure uses variational inference (VI) and amortization, which are known to suffer from certain drawbacks [45, 8]. Moreover, existing NPs do not incorporate TE.

This paper builds on ConvCNPs and NPs [13, 15] to develop *Convolutional Neural Processes* (ConvNPs). ConvNPs are a map from data sets to predictive SPs that is both TE *and* capable of expressing complex joint distributions. As training ConvNPs with VI poses technical and practical issues, we instead propose a simplified maximum-likelihood objective, which directly targets the predictive SP. We show that ConvNPs produce compelling samples and generalize effectively, making them suitable for a broad range of spatio-temporal prediction tasks. Our key contributions are:

1. We introduce ConvNPs, extending ConvCNPs to model rich joint predictive distributions.
2. We propose a simplified training procedure, discarding VI in favor of an approximate maximum-likelihood procedure, which improves performance for ConvNPs.
3. We demonstrate the usefulness of ConvNPs on toy time-series experiments, image-based sampling and extrapolation, and real-world environmental data sets.

2 Problem Set-up and Background

Notation. The main paper provides an informal treatment of ConvNPs. We refer the reader to the supplement for precise definitions and statements. Let $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}, \mathcal{Y} = \mathbb{R}$ denote the input and output spaces, and let $(\boldsymbol{x}, y)$ be an input-output pair. Let $\mathcal{S}$ be the collection of all finite data sets, with $D_c, D_t \in \mathcal{S}$ a *context* and *target* set respectively. We will later consider predicting the target set from the context set as in [12, 13]. Let $\boldsymbol{X}_c, \boldsymbol{y}_c$ be the inputs and corresponding outputs of D_c, with $\boldsymbol{X}_t, \boldsymbol{y}_t$ defined analogously. We denote a single *task* as $\xi = (D_c, D_t) = ((\boldsymbol{X}_c, \boldsymbol{y}_c), (\boldsymbol{X}_t, \boldsymbol{y}_t))$. Let $\mathcal{P}(\mathcal{X})$ denote the collection of stochastic processes on $\mathcal{X}$, and let $C_b(\mathcal{X})$ denote the collection of continuous, bounded functions on $\mathcal{X}$.

2.1 Meta-Learning Stochastic Process Prediction

Consider rainfall y as a function of position $\boldsymbol{x}$. To model rainfall, we can view it as a *random* function from $\mathcal{X}$ to $\mathcal{Y}$. Mathematically, this corresponds to a SP on $\mathcal{X}$—a probability distribution over functions from $\mathcal{X}$ to $\mathcal{Y}$—which we denote by P. Given perfect knowledge of P, we could predict rainfall at any location of interest by conditioning P on observations D_c, yielding a *predictive* SP. However, in practice we will only have access to a large collection of sample functions from P. Each function is known only at a finite set of inputs, $D = (\boldsymbol{x}_n, y_n)_{n=1}^N$, which we divide into D_c, D_t for meta-training. Given sufficient data, we can *meta-learn* the map from context sets D_c to the ground-truth predictive distribution: $D_c \mapsto p(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c) = p(\boldsymbol{y}_t, \boldsymbol{y}_c | \boldsymbol{X}_t, \boldsymbol{X}_c) / p(\boldsymbol{y}_c | \boldsymbol{X}_c)$. As long as the predictives for varying $\boldsymbol{X}_t$ are Kolmogorov-consistent [42, Section 2.4], this corresponds to learning a map from data sets directly to *predictive* SPs. We refer to the map that takes a context set D_c to the *exact* ground truth SP conditioned on D_c as the *prediction map* $\pi_P \colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ (details in App A). The general prediction problem may then be viewed as learning to approximate π_P.

2.2 Translation Equivariance and Stationarity

The prediction map π_P possesses two important symmetries. First, π_P is *invariant* to permutations of D_c [50, 15]. Second, if the ground truth process P is *stationary*, then π_P is *translation equivariant*: whenever an input to the map is translated, its output is translated by the same amount (see App B for formal definitions and proofs). This simple statement highlights the intimate relationship between stationarity and TE. Moreover, it suggests that models for the prediction map should also be TE and permutation invariant. As such models are a small subset of the space of *all* models, building in these properties can greatly improve data efficiency and generalization for stationary SP prediction. In Sec 3, we extend the TE maps of Gordon et al. [15] (reviewed next) to construct a rich class of models which incorporate these inductive biases.

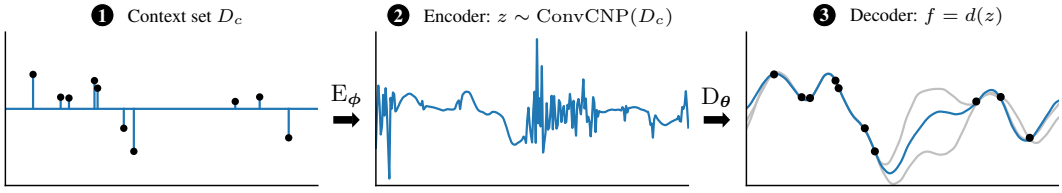


Figure 1: Forward pass through a trained ConvNP. The ConvCNP encoder takes the context set as input (left panel) and outputs a single sample of z (center panel). The decoder takes this as input and outputs a predictive sample (right panel blue; two other samples shown in grey).

2.3 Convolutional Conditional Neural Processes

We review ConvCNPs [15], which are an important building block in our proposed model. ConvCNPs can be viewed from the perspective of SP prediction, revealing their key limitations. Given a context set D_c, the ConvCNP models the predictive distribution over target outputs as:

$$p_{\boldsymbol{\phi}}(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c) = \textstyle\prod_{(\boldsymbol{x},y)\in D_t} \mathcal{N}(y; \mu(\boldsymbol{x}, D_c), \sigma^2(\boldsymbol{x}, D_c)). \tag{1}$$

The mean $\mu(\,\cdot\,, D_c)$ and variance $\sigma^2(\,\cdot\,, D_c)$ are parametrized by *convolutional deep sets* (ConvDeepSets; [15]): a flexible parametrization for TE maps from $\mathcal{S}$ to $C_b(\mathcal{X})$. ConvDeepSets introduce the idea of *functional representations*: whereas the standard DeepSets framework embeds data sets into a finite-dimensional vector space [50], a ConvDeepSet embeds data sets in an infinite-dimensional function space. ConvDeepSets are a composition of two stages. The first stage maps a data set D to its functional representation via $D \mapsto \sum_{(\boldsymbol{x},y)\in D} \phi(y)\psi(\,\cdot - \boldsymbol{x})$. Here $\phi(y) = (1, y) \in \mathbb{R}^2$ and ψ is the Gaussian radial basis function. This functional representation is then passed to the second stage, a TE map between function spaces, implemented by a convolutional neural network (CNN). See App C for a full description of the ConvCNP.

We observe that Eq (1) defines a map from context sets D_c to predictive SPs. Specifically, let $\mathcal{P}_{\mathrm{N}}(\mathcal{X}) \subset \mathcal{P}(\mathcal{X})$ denote the set of *noise GPs*: GPs on $\mathcal{X}$ whose covariance is given by $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2(\boldsymbol{x})\delta[\boldsymbol{x} - \boldsymbol{x}']$, where $\sigma^2 \in C_b(\mathcal{X})$ and $\delta[0] = 1$ with $\delta[\,\cdot\,] = 0$ otherwise. Then the ConvCNP is a map $\mathrm{ConvCNP} : \mathcal{S} \to \mathcal{P}_{\mathrm{N}}(\mathcal{X})$ with Eq (1) defining its finite-dimensional distributions. Since ConvDeepSets are TE, and the means and variances of ConvCNPs are ConvDeepSets, it follows that ConvCNPs are also TE as maps from $\mathcal{S} \to \mathcal{P}_{\mathrm{N}}(\mathcal{X})$ (see App D for a more formal derivation). Unfortunately, processes in $\mathcal{P}_{\mathrm{N}}(\mathcal{X})$ possess two key limitations. First, it is impossible to obtain coherent function samples as each point of the function is generated independently. Second, Gaussian distributions cannot model multi-modality, heavy-tailedness, or asymmetry.

3 The Convolutional Neural Process

We now present the ConvNP, which addresses the weaknesses of ConvCNPs. We introduce their parametrization (Sec 3.1) and a maximum-likelihood meta-training procedure (Sec 3.2).

3.1 Parametrizing Translation Equivariant Maps to Stochastic Processes Using ConvNPs

The ConvNP extends the ConvCNP by parametrizing a map to predictive SPs more expressive than $\mathcal{P}_{\mathrm{N}}(\mathcal{X})$, allowing for coherent sampling and non-Gaussian predictives. It achieves this by passing the output of a ConvCNP through a non-linear, TE map between function spaces. Specifically, the ConvNP uses an encoder–decoder architecture, where the encoder $\mathrm{E}\colon \mathcal{S} \to \mathcal{P}_{\mathrm{N}}(\mathcal{X})$ is a ConvCNP and the decoder $d\colon \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^{\mathcal{X}}$ is TE. Conditioned on D_c, ConvNP samples can be obtained by sampling a function $z \sim \mathrm{ConvCNP}(D_c)$ and then computing $f = d(z)$. This is illustrated in Fig 1. Importantly, d takes functions to functions and does not necessarily act point-wise: letting $f(\boldsymbol{x})$ depend on the value of z at multiple locations is crucial for inducing dependencies in the predictive. This sampling procedure induces a map between SPs, $\mathrm{D}\colon \mathcal{P}_{\mathrm{N}}(\mathcal{X}) \to \mathcal{P}(\mathcal{X})$ (see App D). Putting these together, with explicit parameter dependence in E and D, the ConvNP is constructed as

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

where $(d_{\boldsymbol{\theta}})_*$ is the pushforward under $d_{\boldsymbol{\theta}}$. In App D, we prove that $\mathrm{ConvNP}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ is indeed TE.

In practice, we cannot compute samples of noise GPs ($\mathcal{P}_{\mathrm{N}}$) because they comprise uncountably many independent random variables. Instead, we consider a discrete version of the model, which enables

computation. Following Gordon et al. [15], we discretize the domain of z on a grid $(\boldsymbol{x}_i)_{i=1}^K$, with $\boldsymbol{z} := (z(\boldsymbol{x}_i))_{i=1}^K$. As a consequence, the model can only be equivariant up to shifts on this discrete grid. With this discretization, sampling $\boldsymbol{z} \sim \mathrm{ConvCNP}_{\boldsymbol{\phi}}(D_c)$ amounts to sampling independent Gaussian random variables, and $d_{\boldsymbol{\theta}}$ is implemented by passing $\boldsymbol{z}$ through a CNN. Note that, depending on the amount of padding added to the input, CNNs are not always entirely TE due to the zero padding that occurs at each layer. In practice, we find that this is not an issue.[2] Following Kim et al. [18], we define the model likelihood by adding heteroskedastic Gaussian observation noise $\sigma_y^2(\boldsymbol{x}, \boldsymbol{z})$ to the predictive function draws $f = d_{\boldsymbol{\theta}}(\boldsymbol{z}) \in \mathbb{R}^{\mathcal{X}}$:

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

Although the product in the expectation factorizes, $p_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)$ does not: $\boldsymbol{z}$ induces dependencies in the predictive, in contrast to Eq (1). See App C for full implementation details for the ConvNP.

3.2 Maximum Likelihood Learning of ConvNPs

We now propose a maximum-likelihood training procedure for ConvNPs. Let the ground truth task distribution be $p(\xi) = p(D_c, D_t)$. Let $\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) := \log p_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)$ be the single-task likelihood, and let $\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}) := \mathbb{E}_{p(\xi)}[\log p_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)]$ be the task-averaged likelihood. The following proposition shows that maximizing $\mathcal{L}_{\mathrm{ML}}$ recovers the prediction map π_P in a suitable limit:

Prop 1. Let $\Psi\colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ be a map from data sets to SPs, and let $\mathcal{L}_{\mathrm{ML}}(\Psi) := \mathbb{E}_{p(\xi)}[\log p_\Psi(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)]$ where p_Ψ is the density of $\Psi(D_c)$ at $\boldsymbol{X}_t$. Then Ψ globally maximizes $\mathcal{L}_{\mathrm{ML}}(\Psi)$ if and only if $\Psi = \pi_P$. See App E for more details and conditions.

In practice, we do not have infinite flexibility in our model or infinite data to compute expectations over $p(\xi)$, but Prop 1 shows that maximum-likelihood training is sensible with an expressive model and sufficient data. Letting $\mathcal{D} = \{\xi_n\}_{n=1}^{N_{\mathrm{tasks}}}$ be a *meta-training* set, we can train a ConvNP by stochastic gradient maximization of $\mathcal{L}_{\mathrm{ML}}$ with tasks sampled from $\mathcal{D}$. Unfortunately, for non-linear decoders, $\log p_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)$ is intractable due to the expectation over $\boldsymbol{z}$ (Eq (2)). For a given task ξ, we instead optimize the following Monte Carlo estimate of $\mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi)$, which is conservatively biased, consistent, and monotonically increasing in L (in expectation) [4]:

$$\hat{\mathcal{L}}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) := \log\left[\tfrac{1}{L}\sum_{l=1}^{L} \exp\left(\textstyle\sum_{(\boldsymbol{x},y)\in D_t} \log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{z}_l)\right)\right]; \quad \boldsymbol{z}_l \sim \mathrm{E}_{\boldsymbol{\phi}}(D_c). \tag{3}$$

One drawback of this objective is that single sample estimators are not useful, as they drive $\boldsymbol{z}$ to be deterministic. In our experiments, we set L between 16 and 32. For further discussion of the effect of L see App G. Eq (3) can be viewed as importance sampling in which the prior is the proposal distribution. Prior sampling is typically ineffective as it is unlikely to propose functions that pass near observed data. Here, however, $\mathrm{E}_{\boldsymbol{\phi}}$ depends on context sets D_c, which often is sufficient to constrain prior function samples to be close to D_t. In Sec 5, we demonstrate that, perhaps surprisingly, this estimator often significantly outperforms VI-inspired estimators (discussed next).

4 The Latent Variable Interpretation of ConvNPs

We now describe an alternative approach to training the ConvNP via variational lower bound maximization. This serves the dual purpose of relating ConvNPs to the NP family, and contrasting the existing NP framework with our simplified, maximum-likelihood approach from Sec 3.2.

4.1 A Variational Lower Bound Approach to ConvNPs

Garnelo et al. [13] propose viewing Neural Processes as performing approximate Bayesian inference and learning in the following latent variable model:

$$\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}); \quad y(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{z}); \quad p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z}) = \textstyle\prod_{(\boldsymbol{x},y)\in D_t} \mathcal{N}\left(y; f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{z}), \sigma_y^2\right).$$

To train the model, they propose using *amortized* VI [20, 35]. This involves introducing a variational approximation $q_{\boldsymbol{\phi}}$ which maps data sets $S \in \mathcal{S}$ to distributions over $\boldsymbol{z}$, and maximizing a lower bound on $\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c)$. We can define a similar procedure for ConvNPs. For ConvNPs, $\boldsymbol{z}$ is

[2] See Gordon et al. [15, Appendix D.6] for a discussion.

a latent *function*, $q_{\boldsymbol{\phi}}$ is a map from data sets to SPs, and $f_{\boldsymbol{\theta}}$ is a map between function spaces. A natural choice is to use a ConvCNP and CNN for $q_{\boldsymbol{\phi}}$ and $f_{\boldsymbol{\theta}}$, respectively. This results in the same parameterization as in Sec 3, but a different modelling interpretation and meta-training objective. As with NPs, this bound cannot be evaluated, because it requires computing $p(\boldsymbol{z}|D_c)$, an intractable Bayesian posterior. Garnelo et al. [13] instead propose the following objective:

$$\mathcal{L}_{\mathrm{NP}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) \coloneqq \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c \cup D_t)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z})\right] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c \cup D_t) \| q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)), \tag{4}$$

where we have substituted the intractable term $p(\boldsymbol{z}|D_c)$ with our variational approximation $q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)$. Note that this is no longer an evidence lower bound for the original model. Rather, it may be viewed as separately performing VI in a family of inconsistent models, one for each context set.

For the non-discretized ConvNP, Eq (4) involves KL divergences between SPs which cannot be computed directly and must be treated carefully [29, 41]. On the other hand, for the discretized ConvNP, the KL divergences can be computed, but grow in magnitude as the discretization becomes finer, and it is not clear that the KL divergence between SPs is recovered in the limit. This raises practical issues for the use of Eq (4) with the ConvNP, as the balance between the two terms depends on the choice of discretization.

4.2 Maximum-Likelihood vs Variational Lower Bound Maximization for Training NPs

We argue that the VI interpretation is unnecessary when focusing on predictive performance, and particularly detrimental for ConvNPs, where $\boldsymbol{z}$ has many elements. Noting the equivalence

$$\mathcal{L}_{\mathrm{NP}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) = \mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) - \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c \cup D_t) \| p_{\boldsymbol{\theta}}(D_t|\boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)/Z\right), \tag{5}$$

where Z is a normalizing constant (see App F for a full derivation), we see that $\mathcal{L}_{\mathrm{NP}}$ is equal to $\mathcal{L}_{\mathrm{ML}}$ up to an additional KL term. This KL term encourages *Bayes-consistency* among the $q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)$ in the sense that Bayes' theorem is respected if the target set is subsumed into the context set. In the infinite capacity/data limit, $\mathcal{L}_{\mathrm{NP}}$ is globally maximized if the ConvNP recovers (i) the prediction map π_P for $\boldsymbol{y}_t$ and (ii) exact inference for $\boldsymbol{z}$. This follows from (i) Prop 1, since π_P globally optimizes $\mathcal{L}_{\mathrm{ML}}$; and (ii) that exact inference for $\boldsymbol{z}$ is Bayes-consistent, sending the KL term to zero. In most applications, only the distribution over $\boldsymbol{y}_t$ is of interest. Given only finite capacity/data, it can be advantageous to not expend capacity in enforcing Bayes-consistency for $\boldsymbol{z}$, which suggests it could be beneficial to use $\mathcal{L}_{\mathrm{ML}}$ over $\mathcal{L}_{\mathrm{NP}}$. Further, $\mathcal{L}_{\mathrm{ML}}$ has the advantage of being easy to specify for any map parameterizing a predictive process, posing no conceptual issues for the ConvNP. In Sec 5 we find that $\mathcal{L}_{\mathrm{ML}}$ significantly outperforms $\mathcal{L}_{\mathrm{NP}}$ for ConvNPs, and often also for ANPs.

5 Experiments

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

Evaluation and baselines. We use several approaches for evaluating NPs. First, as in [13, 18], we provide qualitative comparisons of samples. These allow us to see if the models display meaningful structure, quantify uncertainty, and are able to generalize spatially. Second, NPs lack closed-form likelihoods, so we evaluate *lower bounds* on their predictive log-likelihoods via importance sampling [25]. As these bounds can be quite loose (App G.1), they are primarily useful to show when NPs outperform baselines with *exact* likelihoods, such as GPs and ConvCNPs. Finally, in Sec 5.3 we consider Bayesian optimization to evaluate the usefulness of ConvNPs for downstream tasks. In Secs 5.1 and 5.2, we compare against the Attentive NP (ANP; [18]), which in prior work is trained with $\mathcal{L}_{\mathrm{NP}}$. The ANP architectures used here are comparable to those in Kim et al. [18], and have a parameter count comparable to or greater than the ConvNP. Full details provided in the supplement.

5.1 1D Regression

We train on samples from (i) a Matérn-$\frac{5}{2}$ GP, (ii) a weakly periodic GP, and (iii) a non-Gaussian sawtooth process with random shifts and frequency (see App H for details). Fig 2 shows predictive samples, where during training the models only observe data within the grey regions (training range). While samples from the ANP exhibit unnatural "kinks" and do not resemble the underlying process,

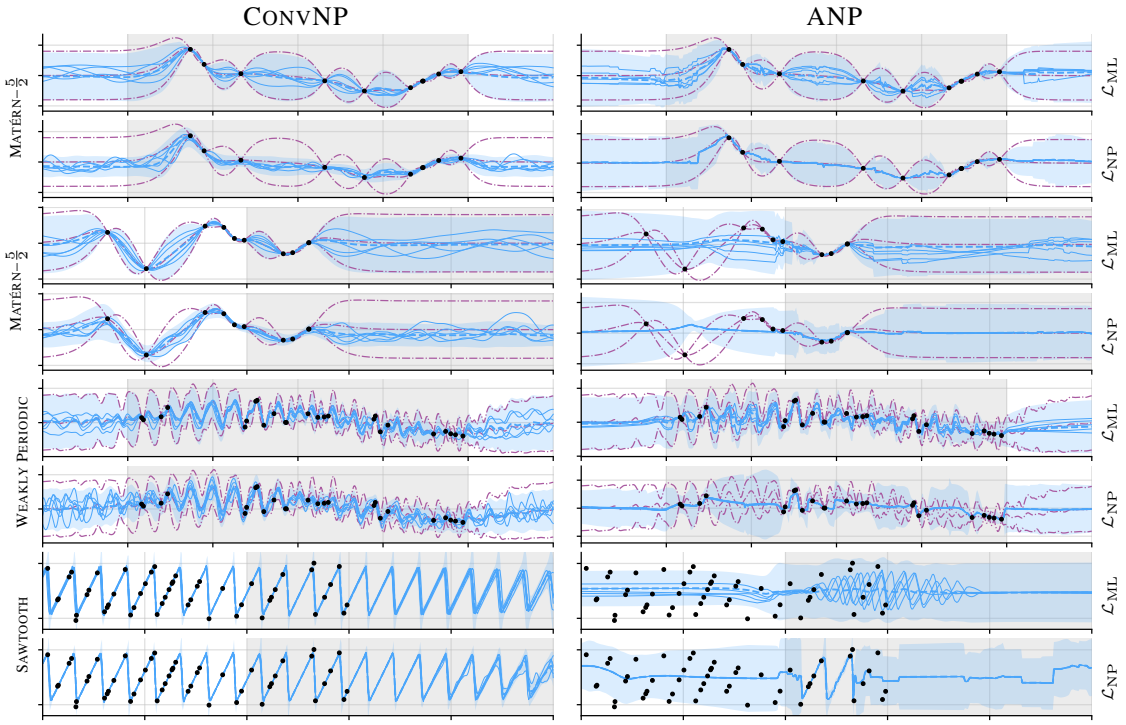


Figure 2: Predictions of ConvNPs and ANPs trained with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NP}}$, showing interpolation and extrapolation within (grey background) and outside (white background) the training range. Solid blue lines are samples, dashed blue lines are means, and the shaded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

Table 1: Log-likelihoods on 1D regression tasks. Lower bounds marked with asterisk. Highest non-GP values in bold.

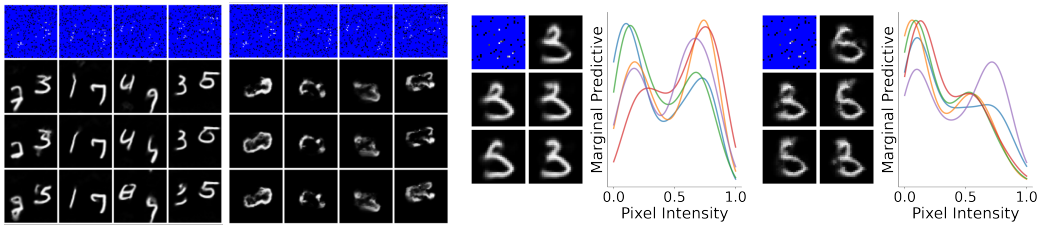
		WITHIN TRAINING RANGE			BEYOND TRAINING RANGE		
		Matérn-$\frac{5}{2}$	Weakly Per.	Sawtooth	Matérn-$\frac{5}{2}$	Weakly Per.	Sawtooth
GP	(full)	$1.22 \pm 6\text{E}{-3}$	$-0.06 \pm 5\text{E}{-3}$	N/A	$1.22 \pm 6\text{E}{-3}$	$-0.06 \pm 5\text{E}{-3}$	N/A
ConvNP*	($\mathcal{L}_{\text{ML}}$)	$\mathbf{-0.58} \pm 0.01$	$\mathbf{-1.02} \pm 6\text{E}{-3}$	$\mathbf{2.30} \pm 0.01$	$\mathbf{-0.58} \pm 0.01$	$\mathbf{-1.03} \pm 6\text{E}{-3}$	$\mathbf{2.29} \pm 0.02$
ANP*	($\mathcal{L}_{\text{ML}}$)	-0.73 ± 0.01	$-1.14 \pm 6\text{E}{-3}$	$0.09 \pm 3\text{E}{-3}$	$-1.39 \pm 7\text{E}{-3}$	$-1.35 \pm 4\text{E}{-3}$	$-0.17 \pm 1\text{E}{-3}$
ANP*	($\mathcal{L}_{\text{NP}}$)	-0.96 ± 0.01	$-1.37 \pm 6\text{E}{-3}$	$0.20 \pm 9\text{E}{-3}$	$-1.48 \pm 4\text{E}{-3}$	-1.66 ± 0.01	$-0.30 \pm 4\text{E}{-3}$
GP	(diag)	$-0.84 \pm 9\text{E}{-3}$	$-1.17 \pm 5\text{E}{-3}$	N/A	$-0.84 \pm 9\text{E}{-3}$	$-1.17 \pm 5\text{E}{-3}$	N/A
ConvCNP		-0.88 ± 0.01	$-1.19 \pm 7\text{E}{-3}$	1.15 ± 0.04	-0.87 ± 0.01	$-1.19 \pm 7\text{E}{-3}$	1.11 ± 0.04

the ConvNP produces smooth samples for Matérn–$\frac{5}{2}$ and samples exhibiting meaningful structure for the weakly periodic and sawtooth processes. The ConvNP also generalizes gracefully beyond the training range, whereas ANP fails catastrophically. The ANP with $\mathcal{L}_{\text{NP}}$ collapses to deterministic samples, with the epistemic uncertainty explained using the heteroskedastic noise $\sigma_y^2(\boldsymbol{x}, \boldsymbol{z})$. This was also noted in Le et al. [25]. This behaviour is alleviated when training with $\mathcal{L}_{\text{ML}}$, with much of the predictive uncertainty due to variations in the sampled functions.

Tab 1 compares lower bounds on the log-likelihood for ConvNP with our proposed $\mathcal{L}_{\text{ML}}$ objective and ANP with both $\mathcal{L}_{\text{ML}}$ and the standard $\mathcal{L}_{\text{NP}}$ objective. We also show three *exact* log-likelihoods: (i) the ground-truth GP (full) (ii) the ground-truth GP with diagonalised predictions (diag), and (iii) ConvCNP. The ConvCNP performs on par with GP (diag), which is the optimal factorized predictive. The ConvNP lower bound is consistently higher than the GP (diag) and ConvCNP log-likelihoods, demonstrating that its correlated predictives improve predictive performance. Further, the ConvNP performs similarly inside and outside its training range, demonstrating that TE helps generalization; this is in contrast to the ANP, which fails catastrophically outside its training range.

Table 2: Test log-likelihood lower bounds for image completion (5 runs).

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



(a) ConvNP (b) ANP (c) ConvNP (d) ANP

Figure 3: Left two plots: predictive samples on zero-shot multi MNIST. Right two plots: samples and marginal predictives on standard MNIST. We plot the density of the five marginals that maximize Sarle's bimodality coefficient [11]. We use $\mathcal{L}_{\text{ML}}$ for training. Blue pixels are not in the context set.

In App I, we provide a thorough comparison of log-likelihood bounds for multiple models, training objectives, and data sets.

5.2 Image Completion

We evaluate ConvNPs on image completion tasks focusing on spatial generalization. To test this, we consider zero-shot multi MNIST (ZSMM), where we train on single MNIST digits but test on two MNIST digits on a larger canvas. We randomly translate the digits during training, so the generative SP is stationary. The black background on MNIST causes difficulty with heteroskedastic noise, as the models can obtain high likelihood by predicting the background with high confidence whilst ignoring the digits. Hence for MNIST and ZSMM we use homoskedastic noise $\sigma_y^2(\boldsymbol{z})$. Figs 3a and 3b show that the ANP fails to generalize spatially, whereas this is naturally handled by the ConvNP.

We also test the ConvNP's ability to learn non-Gaussian predictive distributions. Fig 3c shows that the ConvNP can learn highly multimodal predictives, enabling the generation of diverse yet coherent samples. A quantitative comparison of models using log-likelihood lower bounds is provided in Tab 2, where ConvNP trained with $\mathcal{L}_{\text{ML}}$ consistently achieves the highest values. App J provides details regarding the data, architectures, and protocols used in our image experiments. In App K, we provide samples and further quantitative comparisons of models trained on SVHN [30], MNIST [26], and 32×32 CelebA [30] in a range of scenarios, along with full experimental details.

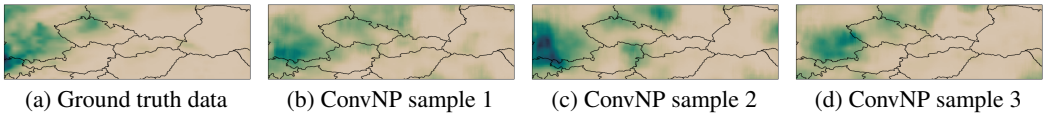
5.3 Environmental Data

We next consider a real-world spatial data set, ERA5-Land [40], containing measured environmental variables at a $\sim$9 km spacing across the globe. We consider predicting daily precipitation y at spatial position $\boldsymbol{x}$. We also provide the model with orography (elevation) and temperature values. We choose a large region of central Europe as our train set, and use regions east, west and south as held-out test sets. For such tasks, models must be able to make predictions at locations spanning a range different from the training set, inhibiting the deployment of NPs not equipped with TE. To sample a task at train time, we sample a random date between 1981 and 2020, then sample a sub-region within the train region, which is split into context and target sets. In this section, we train using $\mathcal{L}_{\text{ML}}$. See App L for experimental details.

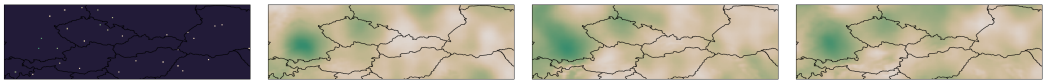
Prediction. We first evaluate the ConvNP's predictive performance, comparing to a GP trained individually on each task as a baseline. In about 10% of tasks, the GP obtains a poor likelihood (< 0 nats); we remove these outliers from the evaluation. The results are shown in Tab 3. The ConvNP and GP have comparable RMSEs except on south, where the ConvNP outperforms the GP. However, the ConvNP consistently outperforms the GP in log-likelihood, which is expected for the following reasons: (i) the GP does not share information between tasks and hence is prone to overfitting on small context sets, resulting in overconfident predictions; and (ii) the ConvNP can learn non-Gaussian

Table 3: Joint predictive log-likelihoods (LL) and RMSEs on ERA5-Land, averaged over 1000 tasks.

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



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3



(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

Figure 4: Samples from the predictive processes overlaid on central Europe. Darker colours show higher precipitation. In (e), coloured pixels represent context points. GP samples often take negative values (lighter than ground truth data, see App L.2 for a discussion), whereas the NP has learned to produce non-negative samples which capture the *sparsity* of precipitation. To produce high-quality samples, the model is trained on subregions roughly the size of the lengthscale of the precipitation process. More samples in App M.

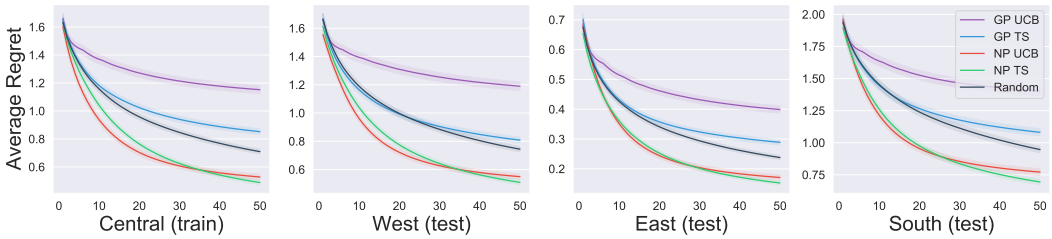


Figure 5: Average regret plotted against number of points queried, averaged over 5000 tasks.

predictive densities (illustrated in App M). Fig 4 shows samples from the predictive process of a ConvNP and GP, over the whole of the train region. This demonstrates spatial extrapolation, as the ConvNP is trained only on random subregions.

Bayesian optimization. We demonstrate the ConvNP in a downstream task by considering a toy Bayesian optimisation problem, where the goal is to identify the location with heaviest rainfall on a given day. We also test the ConvNP's spatial generalization, by optimising over larger regions (for central, west, and south) than the model was trained on. We test both Thompson sampling (TS) [43] and upper confidence bounds (UCB) [1] as methods for acquiring points. Note that TS requires coherent samples. The results are shown in Fig 5. On all data sets, ConvNP TS and UCB significantly outperform the random baseline by the 50th iteration; the GP does not reliably outperform random. We hypothesize this is due to its overconfidence, in line with the results on prediction.

6 Related Work and Discussion

We have introduced the ConvNP, a TE map from observed data sets to predictive SPs. Within the NP framework, ConvNPs bring together three key considerations.

Expressive joint densities. ConvNPs extend ConvCNPs to allow for expressive joint predictive densities. A powerful alternative approach is to combine *autoregressive* (AR) models (such as PixelCNN++ [38] and the Image Transformer [31]) with CNPs. A difficulty in introducing AR sampling to CNPs is the need to specify a sampling ordering, which is in tension with permutation invariance and relates to the discussion on Bayes-consistency (Sec 4.2). Several works have considered *exchangeable* NP models [28, 24, 23], providing an avenue for future investigation.

Translation equivariance. There has been much interest in incorporating equivariance with respect to symmetry groups into neural networks, e.g. [21, 6, 7, 22], with a comprehensive treatment provided by Bloem-Reddy and Teh [3]. ConvNPs leverage a simple relationship between translation equivariance and stationarity to construct a model particularly well suited to stationary SPs. Similar ideas have been explored for 3D point-cloud modelling [32, 33]. For example, the models proposed in [48, 47] perform convolutions over continuous domains, which are both TE and permutation invariant, achieving excellent performance in point-cloud classification. In contrast with ConvNPs, point-cloud models (i) are generally used as classification function approximators, rather than meta or few-shot learners; (ii) are typically tailored towards point clouds, making heavy use of specific properties for function design; and (iii) have not considered latent variable or stochastic generalizations.

Neural Process training procedures. One of the key benefits of CNPs is their simple maximum-likelihood training procedure [12, 15]. In contrast, NPs are usually trained with VI-inspired objectives [13], variants of which are empirically investigated in Le et al. [25]. We propose an alternative training procedure that discards VI in favor of a (biased) maximum-likelihood approach that focuses on directly optimizing predictive performance. In this regard, our work is similar to Gordon et al. [14], albeit in a very different domain. This approach has two benefits: (i) it does not require carefully designed inference procedures, and works "out-of-the-box" for a range of models; and (ii) empirically, we find that it leads to improved performance for ConvNPs and, often, for ANPs.

Acknowledgements

The authors would like to thank Invenia Labs for their support during the project. We thank William Tebbutt for insightful discussions. We thank David R. Burt, Eric Nalisnick, Cozmin Ududec and John Bronskill for helpful comments on the manuscript. Andrew Y. K. Foong gratefully acknowledges funding from a Trinity Hall Research Studentship and the George and Lilian Schiff Foundation. Richard E. Turner is supported by Google, Amazon, ARM, Improbable and EPSRC grants EP/M0269571 and EP/L000776/1.

References

[1] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020. URL `http://jmlr.org/papers/v21/19-322.html`.

[4] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[6] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR.

[7] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1321–1330, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/cohen19d.html`.

[8] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1078–1086, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/cremer18a.html`.

[9] Noel Cressie. The origins of kriging. *Mathematical geology*, 22(3):239–252, 1990.

[10] Jean Pierre Delhomme. Kriging in the hydrosciences. *Advances in water resources*, 1:251–266, 1978.

[11] Aaron M Ellison. Effect of seed dimorphism on the density-dependent dynamics of experimental populations of atriplex triangularis (chenopodiaceae). *American Journal of Botany*, 74(8):1280–1288, 1987.

[12] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/garnelo18a.html`.

[13] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[14] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkxStoC5F7`.

[15] Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Skey4eBYPS`.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SkE6PjC9KX`.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *In International Conference on Learning Representations (ICLR)*, 2015.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] Imre Risi Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.

[22] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2747–2755, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[23] Iryna Korshunova, Yarin Gal, Arthur Gretton, and Joni Dambre. Conditional bruno: A neural process for exchangeable labelled data. *Neurocomputing*, 2020.

[24] Ananya Kumar, SM Eslami, Danilo J Rezende, Marta Garnelo, Fabio Viola, Edward Lockhart, and Murray Shanahan. Consistent generative query networks. *arXiv preprint arXiv:1807.02033*, 2018.

[25] Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical evaluation of neural process objectives. In *NeurIPS workshop on Bayesian Deep Learning*, 2018.

[26] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. 2019. URL `http://arxiv.org/abs/1906.08324`.

[29] Alexander G de G Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Artificial Intelligence and Statistics*, pages 231–239, 2016.

[30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[31] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5099–5108. 2017.

[34] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[35] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[36] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.

[37] Sheldon M Ross, John J Kelly, Roger J Sullivan, William James Perry, Donald Mercer, Ruth M Davis, Thomas Dell Washburn, Earl V Sager, Joseph B Boyce, and Vincent L Bristow. *Stochastic processes*, volume 2. Wiley New York, 1996.

[38] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *In International Conference on Learning Representations (ICLR)*, 2017.

[39] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning*. PhD thesis, Technische Universität München, 1987.

[40] Copernicus Climate Change Service. Copernicus Climate Change Service (C3S) (2019): C3S ERA5-Land reanalysis, 2020. URL `https://cds.climate.copernicus.eu/cdsapp#!/home`. (accessed: 15.05.2020).

[41] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rkxacs0qY7`.

[42] Terence Tao. *An introduction to measure theory*, volume 126. American Mathematical Society Providence, RI, 2011.

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

[44] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[45] Richard E. Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian Time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[47] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[48] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

[50] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Currant Inc., 2017.

A Formal Definitions and Set-up

Notation. We first review the notation introduced in the main body for convenience. Let $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ and $\mathcal{Y} = \mathbb{R}$ denote the input and output spaces respectively, and let $(\boldsymbol{x}, y)$ denote a generic input-output pair (higher-dimensional outputs can be treated easily). Define $\mathcal{S}_N = (\mathcal{X} \times \mathcal{Y})^N$ to be the collection of all data sets of size N, and let $\mathcal{S} \coloneqq \bigcup_{N=1}^{\infty} \mathcal{S}_N$. Let $D_c, D_t \in \mathcal{S}$ denote a *context* and *target* set respectively. Later, as is common in recent meta-learning approaches, we will consider predicting the target set from the context set Garnelo et al. [12, 13]. Let $\boldsymbol{X}_c = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_c})$ denote a matrix of context set inputs, with $\boldsymbol{y}_c = (y_1, \ldots, y_{N_c})$ the corresponding outputs; $\boldsymbol{X}_t, \boldsymbol{y}_t$ are defined analogously. We denote a single *task* as $\xi = (D_c, D_t) = (D_c, (\boldsymbol{X}_t, \boldsymbol{y}_t))$.

Stochastic processes. For our purposes, a stochastic process on $\mathcal{X}$ will be defined[3] as a probability measure on the set of functions from $\mathcal{X} \to \mathbb{R}$, i.e. $\mathbb{R}^{\mathcal{X}}$, equipped with the product σ-algebra of the Borel σ-algebra over each index point [42], denoted Σ. The measurable sets of Σ are those which can be specified by the values of the function at a countable subset $I \subset \mathcal{X}$ of its input locations. Since in practice we only ever observe data at a finite number of points, this is sufficient for our purposes. We denote the set of all such measures as $\mathcal{P}(\mathcal{X})$. We model the world as having a ground truth stochastic process $P \in \mathcal{P}(\mathcal{X})$. Consider a Kolmogorov-consistent (i.e. consistent under marginalization) collection of distributions on finite index sets $I \subset \mathcal{X}$. By the Kolmogorov extension theorem, there exists a unique measure on $(\mathbb{R}^{\mathcal{X}}, \Sigma)$ that has these distributions as its finite marginals. Hence we may think of these stochastic processes as defined by their finite-dimensional marginals.

Conditioning on observations. We now define what it means to condition on observations of the stochastic process P. Let $p(\boldsymbol{y}|\boldsymbol{X})$ denote the density with respect to Lebesgue measure of the finite marginal of P with index set $\boldsymbol{X}$ (we assume these densities always exist). Assume we have observed P at a finite number of points $(\boldsymbol{X}_c, \boldsymbol{y}_c)$, with $p(\boldsymbol{y}_c|\boldsymbol{X}_c) > 0$. Let $\boldsymbol{X}_t$ be another finite index set. Then we define the finite marginal at $\boldsymbol{X}_t$ conditioned on D_c as the distribution with density

$$p(\boldsymbol{y}_t|\boldsymbol{X}_t, D_c) = \frac{p(\boldsymbol{y}_t, \boldsymbol{y}_c|\boldsymbol{X}_t, \boldsymbol{X}_c)}{p(\boldsymbol{y}_c|\boldsymbol{X}_c)}. \tag{6}$$

It can easily be verified that for a fixed D_c, the conditional marginal distributions for different $\boldsymbol{X}_t$ in Eq (6) are Kolmogorov-consistent. Again, the Kolmogorov extension theorem implies there is a unique measure P_{D_c} on $(\mathbb{R}^{\mathcal{X}}, \Sigma)$ that has Eq (6) as its finite marginals. We now define $\pi_P : \mathcal{S} \to \mathcal{P}(\mathcal{X}), \pi_P : D_c \mapsto P_{D_c}$ as the *prediction map*, so called because it maps each observed dataset D_c to the exact predictive stochastic process conditioned on D_c. The meta-learning task may be viewed as learning an approximation to the prediction map.

B Stationary Processes and Translation Equivariance

Def 1 (Translating data sets and SPs)**.** We define the action of the translation operator $T_{\boldsymbol{\tau}}$ on data sets and SPs, where $\boldsymbol{\tau} \in \mathcal{X}$ denotes the shift vector of the translation.[4]

1. Let $(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N = S \in \mathcal{S}$. For the index set $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, the translation by $\boldsymbol{\tau}$ is defined as $T_{\boldsymbol{\tau}}\boldsymbol{X} = (\boldsymbol{x}_1 + \boldsymbol{\tau}, \ldots, \boldsymbol{x}_n + \boldsymbol{\tau})$. Similarly, $T_{\boldsymbol{\tau}} S \coloneqq (\boldsymbol{x}_n + \boldsymbol{\tau}, \boldsymbol{y}_n)_{n=1}^N$.
2. For a function $f \in \mathbb{R}^{\mathcal{X}}$, define $T_{\boldsymbol{\tau}} f(\boldsymbol{x}) \coloneqq f(\boldsymbol{x} - \boldsymbol{\tau})$ for all $\boldsymbol{x} \in \mathcal{X}$. Let $F \in \Sigma$ be a measurable set of functions. Then $T_{\boldsymbol{\tau}} F \coloneqq \{T_{\boldsymbol{\tau}} f : f \in F\}$.
3. For any SP $P \in \mathcal{P}(\mathcal{X})$, we now define $T_{\boldsymbol{\tau}} P$ by setting[5] $T_{\boldsymbol{\tau}} P(F) \coloneqq P(T_{-\boldsymbol{\tau}} F)$ for all $F \in \Sigma$.

Def 2 (Stationary SP)**.** We say a stochastic process is (strictly) *stationary* if the densities of its finite marginals satisfy

$$p(\boldsymbol{y}_t|\boldsymbol{X}_t) = p(\boldsymbol{y}_t|T_{\boldsymbol{\tau}}\boldsymbol{X}_t) \tag{7}$$

for all $\boldsymbol{y}_t$, $\boldsymbol{X}_t$ and $\boldsymbol{\tau}$.

Def 3 (Translation equivariant prediction maps)**.** We say that $\Psi \colon \mathcal{S} \to \mathcal{P}(\mathcal{X})$ is *translation equivariant* if $\Psi(T_{\boldsymbol{\tau}} S) = T_{\boldsymbol{\tau}} \Psi(S)$ for any data set $S \in \mathcal{S}$ and shift $\boldsymbol{\tau} \in \mathcal{X}$.

[3] Strictly speaking, this is non-standard terminology, since P is the *law* of a stochastic process.

[4] To prevent notational clutter, the same symbol, $T_{\boldsymbol{\tau}}$, will denote translations on multiple kinds of objects.

[5] This is well-defined since Σ is closed under translations. Equivalently, we could define $T_{\boldsymbol{\tau}} P$ as the push-forward of P under the the translation map on functions, $T_{\boldsymbol{\tau}} : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^{\mathcal{X}}$.

The following simple statement highlights the link between stationarity and translation equivariance:

Prop 2. Let P be a stationary SP. Then the prediction map π_P is translation equivariant.[6]

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

$$\begin{aligned} p(\boldsymbol{y}_t|\boldsymbol{X}_t, T_{\boldsymbol{\tau}} D_c) &= \frac{p(\boldsymbol{y}_t, \boldsymbol{y}_c|\boldsymbol{X}_t, T_{\boldsymbol{\tau}}\boldsymbol{X}_c)}{p(\boldsymbol{y}_c|T_{\boldsymbol{\tau}}\boldsymbol{X}_c)} \\ &= \frac{p(\boldsymbol{y}_t, \boldsymbol{y}_c|T_{-\boldsymbol{\tau}}\boldsymbol{X}_t, \boldsymbol{X}_c)}{p(\boldsymbol{y}_c|\boldsymbol{X}_c)} \\ &= p(\boldsymbol{y}_t|T_{-\boldsymbol{\tau}}\boldsymbol{X}_t, D_c), \end{aligned}$$

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

C Description and Pseudocode for ConvCNP and ConvNP

We provide additional details and pseudo-code for ConvCNP and ConvNP. Similar to Gordon et al. [15], we distinguish between the "on-the-grid" and "off-the-grid" versions of the model. In our experiments, we use the "off-the-grid" version of the model for the 1d experiments in Sec 5.1, and the "on-the-grid" version for the image and environmental experiments in Secs 5.2 and 5.3.

C.1 ConvCNP Pseudo-Code and Details

Off-the-grid ConvCNP. We begin by providing details for off-the-grid ConvCNP. As detailed in the main text, the encoder $\mathrm{E}_{\boldsymbol{\phi}}$ is defined by a ConvCNP, which provides a distribution over latent functions z. In practice, we consider the discretized version, where we denote the grid of discretization locations as $(\boldsymbol{t}_i)_{i=1}^K$, with $\boldsymbol{t}_i \in \mathcal{X}$. Let $p_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{t}_i, D_c)$ denote the density of the latent function at the ith position, i.e. at $\boldsymbol{z}_i = z(\boldsymbol{t}_i)$. Then in order to sample $\boldsymbol{z} \sim \mathrm{E}_{\boldsymbol{\phi}}$ (as in e.g. Eq (2) in the main body) we specify the density of the entire discretized latent function $\boldsymbol{z}$ as:

$$p_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c) = \prod_{i=1}^K p_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{t}_i, D_c) = \prod_{i=1}^K \mathcal{N}(\boldsymbol{z}_i; \mu(\boldsymbol{t}_i, D_c), \sigma^2(\boldsymbol{t}_i, D_c)), \tag{8}$$

where μ and σ^2 are parametrized by ConvDeepSets [15].

ConvDeepSets can be expressed as the composition of two functions. Let $\Phi = \rho \circ \gamma$ be a ConvDeepSet. γ maps a data set D to its functional representation via

$$\gamma(D) = \sum_{(\boldsymbol{x}, y) \in D} \phi(y)\psi(\cdot - \boldsymbol{x}).$$

Following Gordon et al. [15], we set $\phi(y) = [1, y]^\mathsf{T} \in \mathbb{R}^2$, and ψ to be a radial basis function. $\gamma(D)$ is itself discretized by evaluating it on a grid (which for simplicity we can also take to be $(\boldsymbol{t}_i)_{i=1}^K$).

Next, ρ maps the discretized $\gamma(D)$ to a continuous function, which we denote $f = \rho(\gamma(D))$. γ is itself implemented in two stages. First a deep CNN maps the discretized $\gamma(D)$ to a discretized output. Second, this discrete output is mapped to a continuous function by using the CNN outputs as weights for evenly-spaced basis functions (again employing radial basis functions), which we denote by ψ_ρ.

Whenever models output standard deviations, we enforce positivity via a function (e.g. the soft-plus function), which we denote $\mathrm{pos}(\cdot)$. Pseudo-code for a forward pass through an off-the-grid ConvCNP is provided in Algorithm 1. Note the forward pass involves the computation of a *density channel* $\boldsymbol{h}^{(0)}$, whose role intuitively is to allow the model to know where it has observed datapoints. This is discussed further in Gordon et al. [15].

On-the-grid ConvCNP. Next, we describe the ConvCNP for on-the-grid data, which is used in our image and environmental experiments. This version is simpler to implement in practice, and is

[6] We exclude conditioning on observations that have zero density, so that the prediction map is well defined.

Algorithm 1 Forward pass through ConvCNP (off-the-grid)

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

Algorithm 2 ConvCNP Forward pass (on-the-grid)

Require: $\rho = (\text{CNN}, \psi_\rho)$ and $\text{CONV}_{\boldsymbol{\theta}}$
Require: image I, context M_c, and target mask M_t
1: We discretize at the pixel locations.
2: $\text{I}_c \leftarrow \text{M}_c \odot \text{I}$
3: $\boldsymbol{h} \leftarrow \text{CONV}_{\boldsymbol{\theta}}([\text{M}_c, \text{I}_c]^\top)$
4: $\boldsymbol{h}^{(1:C)} \leftarrow \boldsymbol{h}^{(1:C)} / \boldsymbol{h}^{(0)}$
5: $f_t \leftarrow \text{M}_t \odot \text{CNN}(\boldsymbol{h})$
6: $\boldsymbol{\mu} \leftarrow f_t^{(1:C)}$
7: $\boldsymbol{\sigma} \leftarrow \text{pos}(f_t^{(C+1:2C)})$
8: **return** $(\boldsymbol{\mu}, \boldsymbol{\sigma})$

applicable whenever the input data is confined to a regular grid. As in Gordon et al. [15] we choose the discretization $(\boldsymbol{t}_i)_{i=1}^K$ to be the pixel locations.

Let $\text{I} \in \mathbb{R}^{H \times W \times C}$ be an image of dimensions H, W, C (height, width, and channels, respectively). We define a mask M_c, which is such that $[\text{M}_c]_{i,j} = 1$ if pixel location (i, j) is in the context set, and 0 otherwise. Masking an image is then achieved via element-wise multiplication, denoted $\text{M}_c \odot \text{I}$. This allows us to flexibly define context and target sets for an image (target sets are typically considered as the complete image, so the masks M_c are simply binary-valued tensors with the same dimensions as the image). In this setting, we implement ϕ, by selecting the context points, and prepend the context mask: $\phi = [\text{M}_c, \text{Z}_c]^\top$. We then implement γ by a simple convolutional layer, which we denote $\text{CONV}_{\boldsymbol{\theta}}$ to emphasize that we use a standard 2d convolutional layer. Full pseudo-code for the on-the-grid ConvCNP is provided in Algorithm 2.

C.2 Pseudo-Code for the ConvNP

The ConvNP can be implemented very simply by passing samples from the ConvCNP through an additional CNN decoder, which we denote $d_{\boldsymbol{\theta}}$. For an "off-the-grid" ConvNP, similarly to the ConvCNP, we must map the output of a standard CNN back to functions on a continuous domain $\mathcal{X}$. This can be achieved via an RBF mapping, similar to the off-the-grid ConvCNP, e.g. Algorithm 1 lines 6, 7. Pseudo-code for off- and on-the-grid ConvNPs are provided in Algorithms 3 and 4, respectively. Note that for the ConvNP, the discretization of the latent function $\boldsymbol{z}$ is typically on a pre-specified grid, and therefore lines 6 and 7 of Algorithm 1 are unnecessary when calling the ConvCNP (Algorithm 3, line 1). Finally, Fig 6 provides an illustration of a forward pass through the ConvNP. The diagram was created using a context set drawn from an EQ kernel, and passed through a trained ConvNP.

Algorithm 3 Forward pass through ConvNP (off-the-grid)

Require: $d = (\text{CNN}, \psi_d)$, $\text{E}_{\boldsymbol{\phi}}$ (off-the-grid ConvCNP), and number of samples L
Require: context $(\boldsymbol{x}_n, y_n)_{n=1}^N$, target $(\boldsymbol{x}_m^*)_{m=1}^M$
$\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z \leftarrow \text{E}_{\boldsymbol{\phi}}(D_c)$
for $l = 1, \ldots, L$ **do**
  $\boldsymbol{z}_l \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$
  $(f_\mu(\boldsymbol{t}_i), f_\sigma(\boldsymbol{t}_i))_{i=1}^K \leftarrow \text{CNN}(\boldsymbol{z}_l)$
  $\boldsymbol{\mu}_{m,l} \leftarrow \sum_{i=1}^T f_\mu(\boldsymbol{t}_i)\psi_d(\boldsymbol{x}_m^* - \boldsymbol{t}_i)$
  $\boldsymbol{\sigma}_{m,l} \leftarrow \text{pos}\,(f_\sigma(\boldsymbol{t}_i))$
end for
return $(\boldsymbol{\mu}, \boldsymbol{\sigma})$

Algorithm 4 Forward pass through ConvNP (on-the-grid)

Require: $d = \text{CNN}$, $\text{E}_{\boldsymbol{\phi}}$ (on-the-grid ConvCNP), and number of samples L
Require: image I, context mask M_c, and target mask M_t
$\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z \leftarrow \text{E}_{\boldsymbol{\phi}}(I, M_c)$
for $l = 1, \ldots, L$ **do**
  $\boldsymbol{z}_l \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$
  $(f_\mu(\boldsymbol{t}_i), f_\sigma(\boldsymbol{t}_i))_{i=1}^K \leftarrow \text{CNN}(\boldsymbol{z}_l)$
  $\boldsymbol{\mu} \leftarrow f_t^{(1:C)}$
  $\boldsymbol{\sigma} \leftarrow \text{pos}\left(f_t^{(C+1:2C)}\right)$
end for
return $(\boldsymbol{\mu}, \boldsymbol{\sigma})$

D Translation Equivariance of the ConvNP

We prove that the ConvNP is a translation equivariant map from data sets to stochastic processes, by proving that the decoder and encoder are separately translation equivariant. In this section we suppress the dependence on parameters $(\boldsymbol{\phi}, \boldsymbol{\theta})$.

Lem 1. Let d be a measurable, translation equivariant map from $(\mathbb{R}^{\mathcal{X}}, \Sigma)$ to $(\mathbb{R}^{\mathcal{X}}, \Sigma)$. The ConvNP decoder $\text{D} : \mathcal{P}(\mathcal{X}) \to \mathcal{P}(\mathcal{X})$, defined by $\text{D}(P) = d_*(P)$, where $d_*(P)$ is the pushforward measure under d, is translation equivariant.

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

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

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

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

□

Lem 2. The ConvNP encoder E (a ConvCNP), is a translation equivariant map from data sets to stochastic processes.

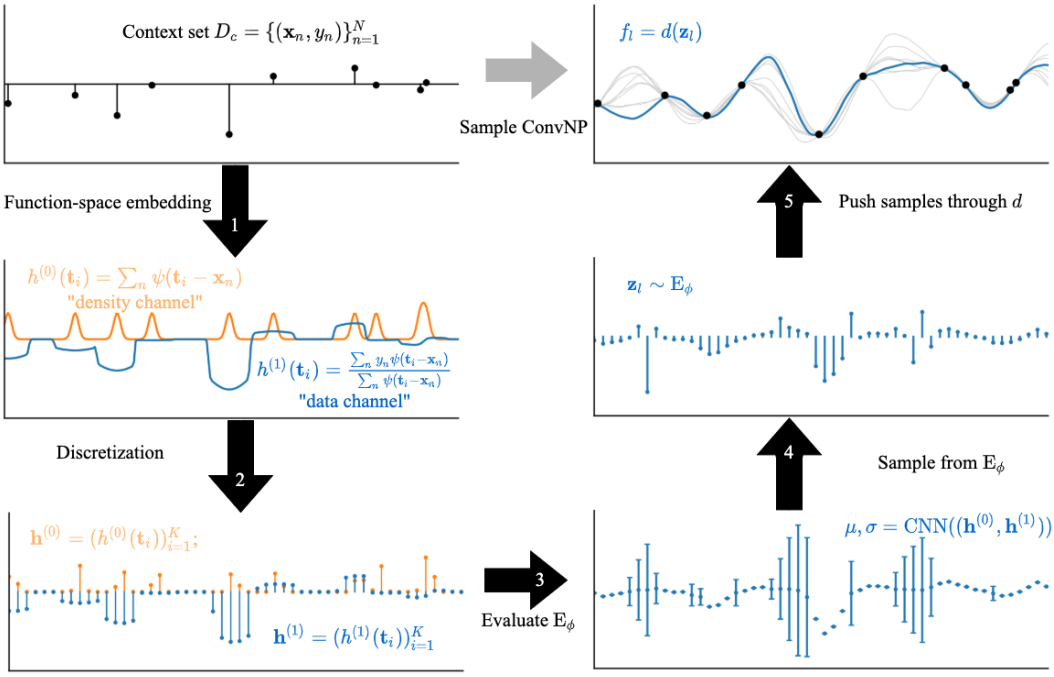


Figure 6: Illustration of a forward pass through a trained ConvNP.

Proof. Recall that the mean and variance $\mu(\cdot, S), \sigma^2(\cdot, S)$ (viewed as maps from $\mathcal{S} \to C_b(\mathcal{X})$) of the encoder E are both given by ConvDeepSets. Due to the translation equivariance of ConvDeepSets [15, Theorem 1], $\mu(\cdot, T_{\boldsymbol{\tau}} S) = T_{\boldsymbol{\tau}} \mu(\cdot, S)$ for all $S, \boldsymbol{\tau}$, and similarly for σ^2. Let $F \in \Sigma$. Then since the measure $\mathrm{E}(S) \in \mathcal{P}_{\mathrm{N}}(\mathcal{X})$ is defined entirely by its mean and variance function, $\mathrm{E}(T_{\boldsymbol{\tau}} S)(F) = \mathrm{E}(S)(T_{-\boldsymbol{\tau}} F) = T_{\boldsymbol{\tau}} \mathrm{E}(S)(F)$. □

Noting that a composition of translation equivariant maps is itself translation equivariant, we obtain the following proposition:

Prop 3. Define $\mathrm{ConvNP} = \mathrm{D} \circ \mathrm{E}$. Then ConvNP is a translation equivariant map from data sets to stochastic processes.

E Recovering the Prediction Map in the Infinite Data / Capacity Limits

Task generation procedure. Assume tasks $\xi = (D_c, D_t)$ are generated as follows: first, some finite number of input locations $\boldsymbol{X}_t, \boldsymbol{X}_c$ are sampled. Assume that $\Pr(|\boldsymbol{X}_t| = n) > 0$ for all $n \in \mathbb{Z}_{\geq 0}$, where $|\boldsymbol{X}_t|$ denotes the number of datapoints in $\boldsymbol{X}_t$, and assume the same is true of $\Pr(|\boldsymbol{X}_c| = n)$. Further assume that for each $n > 0$, the distribution of $\boldsymbol{X}$ given $|\boldsymbol{X}| = n$ has a continuous density with support over all of $\mathbb{R}^{n \times d_{\mathrm{in}}}$. Next, we sample $\boldsymbol{y}_t, \boldsymbol{y}_c$ from the finite marginal of the ground truth stochastic process P, which has density $p(\boldsymbol{y}_t, \boldsymbol{y}_c | \boldsymbol{X}_t, \boldsymbol{X}_c)$. Finally, we set $(D_c, D_t) \coloneqq ((\boldsymbol{X}_t, \boldsymbol{y}_t), (\boldsymbol{X}_c, \boldsymbol{y}_c))$.

Prop 4. Let $\Psi : \mathcal{S} \to \mathcal{P}(\mathcal{X})$ be any map from data sets to stochastic processes, and let $\mathcal{L}_{\mathrm{ML}}(\Psi) \coloneqq \mathbb{E}_{p(\xi)}[\log p_\Psi(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c)]$, where the density p_Ψ is that of $\Psi(D_c)$ evaluated at $\boldsymbol{X}_t$. Then Ψ globally maximises $\mathcal{L}_{\mathrm{ML}}$ if and only if $\Psi = \pi_P$, the prediction map.

Proof. We have:

$$\begin{aligned}
\mathcal{L}_{\mathrm{ML}}(\Psi) &= \mathbb{E}_{p(D_c, \boldsymbol{X}_t, \boldsymbol{y}_t)} \left[\log p_\Psi(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c)\right] && (9)\\
&= \mathbb{E}_{p(D_c, \boldsymbol{X}_t)} \left[\mathbb{E}_{p(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c)} \left[\log p_\Psi(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c)\right]\right] && (10)\\
&= -\mathbb{E}_{p(D_c, \boldsymbol{X}_t)} \left[\mathrm{KL}\left(p(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c) \| p_\Psi(\boldsymbol{y}_t | \boldsymbol{X}_t, D_c)\right)\right] + \text{constant}, && (11)
\end{aligned}$$

where the additive constant is constant with respect to Ψ. First note that the KL-divergence is non-negative, and that the prediction map sends all the KL-divergences to zero, globally optimising $\mathcal{L}(\Psi)$. Furthermore, the KL-divergence is equal to zero if and only if the two distributions are equal, and this must hold for all $\boldsymbol{X}_t, D_c$. For, if this were not the case, the KL-divergence would contribute a non-zero amount to the expectation in Eq (11). □

Strictly speaking, this argument only shows that the finite marginals of the prediction map and Ψ must be equal for almost all $(D_c, \boldsymbol{X}_t)$ with respect to $p(D_c, \boldsymbol{X}_t)$. Since the task generation procedure outlined in this section assumes a finite probability of generating any finite-sized context and target set, this is not very restrictive. However, in practice we often limit the maximum size of the sampled data sets, and also their range in $\mathcal{X}$ space. Hence we can only expect the model to learn reasonable predictions within the ranges seen during train time.

F Relationship Between Neural Process and Maximum-Likelihood Objectives

Let $D \coloneqq D_t \cup D_c$, and let $Z = \int p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)\,\mathrm{d}\boldsymbol{z}$. The NP objective is:

$$\begin{aligned}
\mathcal{L}_{\mathrm{NP}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) &\coloneqq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)}[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z})] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|D) \| q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)) && (12)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)}[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z}) + \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)] && (13)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)}\left[\log Z + \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c)}{Z} - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)\right] && (14)\\
&= \log Z - \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|D) \middle\| \frac{1}{Z} p_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{z}) q(\boldsymbol{z}|D_c)\right). && (15)
\end{aligned}$$

If we identify the approximate posterior $q_{\boldsymbol{\phi}}$ with the encoder of the maximum-likelihood ConvNP, (which in the maximum-likelihood framework does not have an approximate inference interpretation), then $\log Z = \mathcal{L}_{\mathrm{ML}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi)$.

G Effect of Number of Samples Used to Estimate Objective During Training and Evaluation

In this section we empirically examine the effect of L, the number of samples used to estimate likelihood bounds, on the training and evaluation of ConvNPs and ANPs.

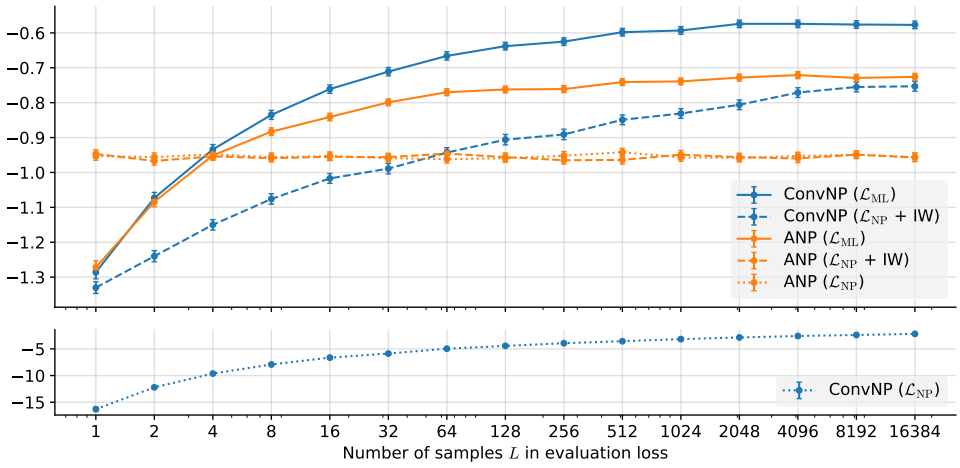
G.1 Effect of Number of Samples Used for Evaluation

As the true log-likelihoods of NP-based models are intractable, quantitative evaluation and comparison of models is challenging. Instead, we compare models by using an estimate of the log-likelihood. A natural candidate is $\mathcal{L}_{\mathrm{ML}}$. However, unless large L is used, $\mathcal{L}_{\mathrm{ML}}$ is conservative and tends to significantly underestimate the log-likelihood. One way to improve the estimate of $\mathcal{L}_{\mathrm{ML}}$ is through importance weighting (IW) [49, 25]. Denoting $D = D_c \cup D_t$, the encoder $\mathrm{E}_{\boldsymbol{\phi}}(D)$ can be used as a proposal distribution:

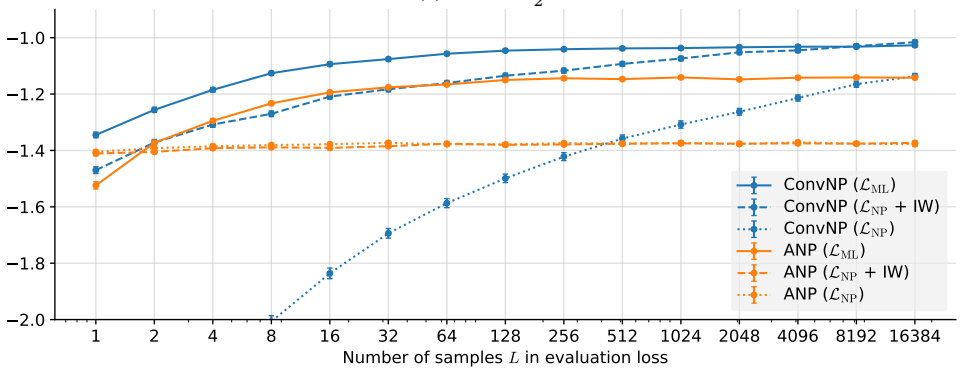
$$\hat{\mathcal{L}}_{\mathrm{IW}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \xi) \coloneqq \log\left(\frac{1}{L}\sum_{l=1}^{L} \exp\left(\log w(\boldsymbol{z}_l) + \sum_{(\boldsymbol{x}, y) \in D_t} \log p_{\boldsymbol{\theta}}(y|\boldsymbol{x}, \boldsymbol{z}_l)\right)\right), \quad \boldsymbol{z}_l \sim \mathrm{E}_{\boldsymbol{\phi}}(D), \tag{16}$$

where the importance weights are given by $\log w(\boldsymbol{z}_l) \coloneqq \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|D_c) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)$. Here $q_{\boldsymbol{\phi}}(\boldsymbol{z}|D)$ is the density of the encoder distribution. We find that training models with $\mathcal{L}_{\mathrm{ML}}$ results in encoders that are ill-suited as proposal distributions, so we only use $\mathcal{L}_{\mathrm{IW}}$ to evaluate models trained with $\mathcal{L}_{\mathrm{NP}}$.

Fig 7 demonstrates the effect of the number of samples L used to estimate the evaluation objective for the ConvNP and ANP trained with $\mathcal{L}_{\mathrm{ML}}$ and $\mathcal{L}_{\mathrm{NP}}$. The models used to generate Fig 7 are the same models used in Sec 5.1, i.e. having heteroskedastic noise. Observe the general trend that the log-likelihood estimates tend to increase with L, as expected. The ANP trained with $\mathcal{L}_{\mathrm{NP}}$ collapsed to a conditional ANP, meaning that the encoder became deterministic; in that case, $\mathcal{L}_{\mathrm{ML}}$ is exact, which means that larger L and importance weighting will not increase the estimate. In contrast, the



(a) Matérn–$\frac{5}{2}$



(b) Weakly periodic kernel

Figure 7: Log-likelihood bounds achieved by various combination of models and training objectives when evaluated with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{IW}}$ for various numbers of samples L. Color indicates model. Solid lines correspond to models trained and evaluated with $\mathcal{L}_{\text{ML}}$. Dashed lines correspond to models trained with $\mathcal{L}_{\text{NP}}$ and evaluated with $\mathcal{L}_{\text{IW}}$. Dotted lines correspond to models trained with $\mathcal{L}_{\text{ML}}$ and evaluated with $\mathcal{L}_{\text{ML}}$.

ANP trained with $\mathcal{L}_{\text{ML}}$ did not collapse, and we see that there the estimate increases with L. For the ConvNP trained with $\mathcal{L}_{\text{NP}}$, evaluating with $\mathcal{L}_{\text{IW}}$ yields a significant increase, showing that the bound estimated with $\mathcal{L}_{\text{IW}}$ is very loose. The models trained with $\mathcal{L}_{\text{ML}}$ tend to be the best performing, although the ConvNP trained with $\mathcal{L}_{\text{NP}}$ is best for weakly periodic kernel and appears to still be increasing with L.

In both the main and the supplement, all log-likelihood lower bounds reported are computed with $\mathcal{L}_{\text{ML}}$ if the model was trained using $\mathcal{L}_{\text{ML}}$ and with $\mathcal{L}_{\text{IW}}$ if the model was trained using $\mathcal{L}_{\text{NP}}$.

G.2 Effect of Number of Samples Used During Training

Fig 8 shows the effect of the number of samples L in the training objectives on the performance of the ConvNP and ANP. Observe that the performance of $\mathcal{L}_{\text{ML}}$ reliably increases with the number of samples L and that $\mathcal{L}_{\text{ML}}$ outperforms $\mathcal{L}_{\text{NP}}$. The performance for $\mathcal{L}_{\text{NP}}$ does not appear to increase with the number of samples L and appears more noisy than $\mathcal{L}_{\text{ML}}$. Note that the models used for Fig 8 were trained with homoskedastic observation noise. This is achieved by pooling f_σ over the time dimension.

H Experimental Details on 1D Regression

For the full results of the 1D regression tasks, see App I.

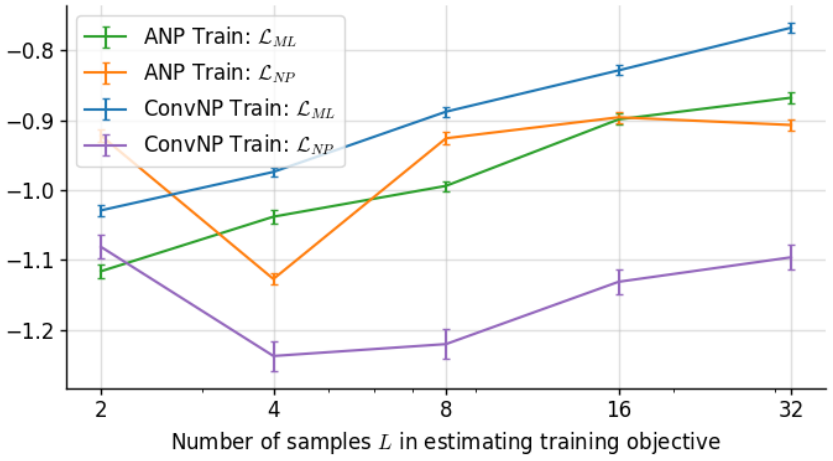


Figure 8: Interpolation performance (within training range) for context set sizes uniformly sampled from $\{0, \ldots, 50\}$ of the ConvNP and ANP on Matérn–$\frac{5}{2}$ samples. The models are trained with $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NP}}$ for various number of samples L. Models trained with $\mathcal{L}_{\text{ML}}$ are evaluated with $\mathcal{L}_{\text{ML}}$, while models trained with $\mathcal{L}_{\text{NP}}$ are evaluated with $\mathcal{L}_{\text{ML}}$. At evaluation, all bounds are estimated using 2,048 samples.

In the 1D regression experiments, we consider the following generative processes:

EQ: samples from a Gaussian process with the following exponentiated-quadratic kernel:

$$k(t, t') = \exp\left(-\frac{1}{8}(t - t')^2\right);$$

Matérn–$\frac{5}{2}$: samples from a Gaussian process with the following Matérn–$\frac{5}{2}$ kernel:

$$k(t, t') = \left(1 + 4\sqrt{5}d + \frac{5}{3}d^2\right)\exp\left(-\sqrt{5}d\right)$$

with $d = 4|x - x'|$;

noisy mixture: samples from a Gaussian process with the following noisy mixture kernel:

$$k(t, t') = \exp\left(-\frac{1}{8}(t - t')^2\right) + \exp\left(-\frac{1}{2}(t - t')^2\right) + 10^{-3}\delta[t - t'];$$

weakly periodic: samples from a Gaussian process with the following weakly-periodic kernel:

$$k(t, t') = \exp\left(-\frac{1}{2}(f_1(t) - f_1(t'))^2 - \frac{1}{2}(f_2(t) - f_2(t'))^2 - \frac{1}{8}(t - t')^2\right)$$

with $f_1(t) = \cos(8\pi t)$ and $f_2(t) = \sin(8\pi t)$; and

sawtooth: samples from the following sawtooth process:

$$f(t) = \frac{A}{2} - \frac{A}{\pi}\sum_{k=1}^{K}(-1)^k\frac{\sin(2\pi k f(t - s))}{k}$$

with $A = 1$, $f \sim \mathcal{U}[3, 5]$, $s \sim \mathcal{U}[-5, 5]$, and $K \in \{10, \ldots, 20\}$ chosen uniformly.

We compare the following models, where all activation functions are leaky ReLUs with leak 0.1:

ConvCNP: The first model is the ConvCNP. The architecture of the ConvCNP is equal to that of the encoder in the ConvNP, described next.

	EQ	Matérn–$\frac{5}{2}$	Noisy Mixt.	Weakly Per.	Sawtooth
ConvCNP	42 822	42 822	51 014	51 014	100 166
ConvNP	88 486	88 486	104 870	104 870	203 174
ANP	530 178	530 178	530 178	530 178	530 178
NP	479 874	479 874	479 874	479 874	479 874

Table 4: Parameter counts for the ConvCNP, ConvNP, ANP, and NP in the 1D regression tasks

ConvNP: The second model is the ConvNP as described in the main body. The functional embedding uses separate length scales for the data channel and density channel (Fig 6), which are initialized to twice the inter-point spacing of the discretization and learned during training. The discretization uniformly ranges over $[\min(x) - 1, \max(x) + 1]$ at density $\rho = 64$ points per unit, where $\min(x)$ is the minimum x value occurring in the union of the context and target sets in the current batch and $\max(x)$ is corresponding maximum x value. The discretization is passed through a 10-layer (excluding an initial and final point-wise linear layer) CNN with 64 channels and depthwise-separable convolutions. The width of the filters depends on the data set and is chosen such that the receptive field sizes are as follows:

EQ: 2,
Matérn–$\frac{5}{2}$: 2,
noisy mixture: 4,
weakly periodic: 4,
sawtooth: 16.

The discretized functional representation consists of 16 channels. The smoothing at the end of the encoder also has separate length scales for the mean and variance which are initialized similarly and learned. The encoder parametrizes the standard deviations by passing the output of the CNN through a softplus. The decoder has the same architecture as the encoder.

ANP: The third model is the Attentive NP with latent dimensionality $d = 128$ and 8-head dot-product attention [46]. In the attentive deterministic encoder, the keys (t), queries (t), and values (concatenation of t and y) are transformed by a three-layer MLP of constant width d. The dot products are normalised by $\sqrt{d}$. The output of the attention mechanism is passed through a constant-width linear layer, which is then passed through two layers of layer normalization [2] to normalise the latent representation. In the first of these two layers, first the transformed queries are passed through a constant-width linear layer and added to the input. In the second of these two layers, the output of the first layer is first passed through a two-layer constant-width MLP and added to itself, making a residual layer. In the stochastic encoder, the inputs and outputs are concatenated and passed though a three-layer MLP of constant width d. The result is mean-pooled and passed through a two-layer constant-width MLP. The decoder consists of a three-layer MLP of constant width d.

NP: The fourth model is the original NP [13]. The architecture is similar to that of the ANP, where the architecture of the deterministic encoder is replaced by that of the stochastic encoder.

For all models, positivity of the observation noise is enforced with a softplus function. Parameter counts of the ConvCNP, ConvNP, ANP, and NP are listed in Tab 4.

The models are trained with $\mathcal{L}_{\mathrm{ML}}$ ($L = 20$) and $\mathcal{L}_{\mathrm{NP}}$ ($L = 5$). For $\mathcal{L}_{\mathrm{NP}}$, the context set is appended to the target set when evaluating the objective. The models are optimised using ADAM with learning rate $5 \cdot 10^{-3}$ for 100 epochs. One epoch consists of 2^{14} tasks divided into batches of size 16. For training, the inputs of the context and target sets are sampled uniformly from $[-2, 2]$. The size of the context set is sampled uniformly from $\{0, \ldots, 50\}$ and the size of the target set is fixed to

Table 5: Log-likelihood for ConvCNP, ConvNP, ANP, and NP. Each of the stochastic models was trained on each data set with $\mathcal{L}_{\mathrm{ML}}$ and $\mathcal{L}_{\mathrm{NP}}$, separately.

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

50. To encourage the NP-based models—not the CNP-based models—to fit and not revert to their conditional variants, the observation noise standard deviation σ is held fixed to 10^{-2} for the first 20 epochs.

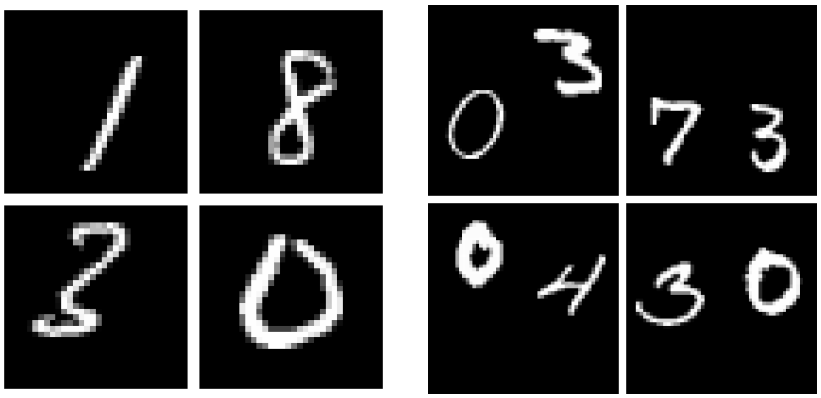
For evaluation, the size of the context set is sampled uniformly from $\{0, \ldots, 10\}$, and the losses are evaluated with $L = 5000$ and batch size one. To test interpolation within the training range, the inputs of the context and target sets are, like training, sampled uniformly from $[-2, 2]$. To test interpolation beyond the training range, the inputs of the context and target sets are sampled uniformly from $[2, 6]$. To test extrapolation beyond the training range, the inputs of the context sets are sampled uniformly from $[-2, 2]$ and the inputs of the target sets are sampled uniformly from $[-4, -2] \cup [2, 4]$. As described in App G.1, models trained with $\mathcal{L}_{\mathrm{NP}}$ are evaluated using importance weighting to obtain a better estimate of the evaluation loss.

I Additional Results on 1D Regression

Tab 5 presents results for all models with all losses on all data sets described in App H according to the evaluation protocol described in Apps G.1 and H.

J Experimental Details on Image Completion

J.1 Data Details



(a) Train (32×32) (b) Test (56×56)

Figure 9: Samples from our generated Zero Shot Multi MNIST (ZSMM) data set.

We use three standard data sets throughout our image experiments: SVHN [30], MNIST [26], and 32×32 CelebA [30]. The aforementioned standard data sets all contain only a single, well-centered object. To evaluate the translation equivariance and generalization capabilities of our model we evaluate on a Zero Shot Multi-MNIST (ZSMM) task, which is similar to ZSMM described in Appendix D.2 of [15]. Namely, we generate a test set by randomly sampling with replacement 10000 pairs of digits from the MNIST test set, place them on a black 56×56 background, and translate the digits in such a way that the digits can be arbitrarily close but cannot overlap (Fig 9b). The difference with the dataset from Gordon et al. [15], is that the training set consists of the standard MNIST digits (instead of a single digit placed in the center of 56×56 canvas), augmented by up to 4 pixel shifts (Fig 9a). The model thus has to generalize both to a larger canvas size as well as to seeing multiple digits.

For all data sets, pixel values are divided by 255 to rescale them to the $[0, 1]$ range. We evaluate on predefined test splits when available (MNIST, SVHN, ZSMM) and make our own test set for CelebA by randomly selecting 10% of the data. For each dataset we also set aside 10% of the training set as validation.

J.2 Training Details

In all experiments, we sample the number of context pixels uniformly from $\mathcal{U}(0, \frac{n_{\text{total}}}{2})$, and the number of target points is set to n_{total}. The weights are optimized using Adam [19] with learning rate 5×10^{-4}. We use a maximum of 100 epochs, with early stopping — based on log likelihood on the validation set — of 10 epochs patience. Unless stated otherwise, we use $L = 16$ samples from the latent function during training, and $L = 128$ at test time. We clip the $L2$ norm of all gradients to 1, which was particularly important for ConvNP. We use a batch size of 32 for all models besides ANP trained on ZSMM which used a batch size of 8 due to memory constraints.

J.3 Architecture Details

General architecture details. For all models, we follow Le et al. [25] and process the predicted standard deviation of the latent function $\boldsymbol{\sigma}_z$ using a sigmoid and the standard deviation $\boldsymbol{\sigma}$ of the predictive distribution using lower-bounded softplus:

$$\boldsymbol{\sigma}_z = 0.001 + (1 - 0.001)\frac{1}{1 + \exp(f_{\sigma,z})} \tag{17}$$

$$\boldsymbol{\sigma} = 0.001 + (1 - 0.001)\ln(1 + \exp(f_{\sigma})) \tag{18}$$

As the pixels are rescaled to $[0, 1]$, we also process the mean of the posterior predictive (conditioned on a single sample) to be in $[0, 1]$ using a logistic function

$$\boldsymbol{\mu} = \frac{1}{1 + \exp(-f_\mu)} \tag{19}$$

In the following, we describe the architecture of ANP and ConvNP. Unless stated otherwise, all vectors in the following paragraphs are in $\mathbb{R}^{128}$ and all MLPs have 128 hidden units.

ANP details. We provide details for the ANP trained with $\mathcal{L}_{\mathrm{ML}}$. As the ANP cannot take advantage of the fact that images are on the grid, we preprocess each pixel so that $\mathbf{x} \in [-1, 1]^2$. The only exception being for the test set of ZSMM, where $\mathbf{x} \in [-\frac{56}{32}, \frac{56}{32}]^2$ as the model is trained on 32×32 but evaluated on 56×56 images. Each context feature is first encoded $\mathbf{x}^{(c)} \mapsto \mathbf{r}_x^{(c)}$ by a single hidden layer MLP, while a second single hidden layer MLP encodes values $\mathbf{y}^{(c)} \mapsto \mathbf{r}_y^{(c)}$. We produce a representation $\mathbf{r}_{xy}^{(c)}$ by summing both representations $\mathbf{r}_x^{(c)} + \mathbf{r}_y^{(c)}$ and passing them through two self-attention layers [46]. Following Parmar et al. [31], each self-attention layer is implemented as 8-headed attention, a skip connection, and two layer normalizations [2]. To predict values at each target point t, we embed $\mathbf{x}^{(t)} \mapsto \mathbf{r}_x^{(t)}$ using the hidden layer MLP used for $\mathbf{r}_x^{(c)}$. A deterministic target representation $\mathbf{r}_{xy}^{(t)}$ is then computed by applying cross-attention (using an 8-headed attention described above) with keys $\mathrm{K} \coloneqq \{\mathbf{r}_x^{(c)}\}_{c=1}^C$, values $\mathrm{V} \coloneqq \{\mathbf{r}_{xy}^{(c)}\}_{c=1}^C$, and query $\mathbf{q} \coloneqq \mathbf{r}_x^{(t)}$. For the latent path, we average over context representations $\mathbf{r}_{xy}^{(c)}$, and pass the resulting representation through a single hidden layer MLP that outputs $(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) \in \mathbb{R}^{256}$. $\boldsymbol{\sigma}_z$ is made positive by post-processing it using Eq (17). We then sample (with reparametrization [20]) L latent representation $\boldsymbol{z}_l \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$.

We describe the remainder of the forward pass for a single $\boldsymbol{z}_l$, though in practice multiple samples may be processed in parallel. The deterministic and latent representations of the context set are concatenated, and the resulting representation is passed through a linear layer $[\mathbf{r}_{xy}^{(t)}; \boldsymbol{z}_l] \to \mathbf{r}_{xyz}^{(t)} \in \mathbb{R}^{128}$. Given the target and context-set representations, the predictive posterior is given by a Gaussian pdf with diagonal covariance parametrised by $(\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}_{\text{pre}}^{(t)}) = \mathrm{decoder}([\mathbf{r}_x^{(t)}; \mathbf{r}_{xyz}^{(t)}])$ where $\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}_{\text{pre}}^{(t)} \in \mathbb{R}^3$ and $\mathrm{decoder}$ is a 4 hidden layer MLP. Finally, the $\boldsymbol{\sigma}^{(t)}$ is processed by Eq (18) using Eq (19). In the case of MNIST and ZSMM, $\boldsymbol{\sigma}^{(t)}$ is also spatially mean pooled, which corresponds to using homoskedastic noise. This improves the qualitative performance by forcing ANP and ConvNP to model the digit instead of focusing on predicting the black background with high confidence. Kim et al. [18] did not suffer from that issue as they used a much larger lower bound for Eq (18).

ConvNP details. The core algorithm of on-the-grid ConvNP is outlined in Algorithm 4 as well as Algorithm 2. Here we discuss the parametrizations used for each step of the algorithm. All convolutional layers are depthwise separable [5]. $\textsc{conv}_{\boldsymbol{\theta}}$ is a convolutional layer with kernel size of 11 (no bias). Following Gordon et al. [15], we enforce positivity on the weights in the first convolutional layer by only convolving their absolute value with the signal.

The CNNs are ResNets [16] with 9 blocks, where each convolution has a kernel size of 3. Each residual block consists of two convolutional layers, pre-activation batch normalization layers [17], and ReLU activations. The output of the pre-latent CNN (CNN in Algorithm 2) goes through a single hidden layer MLP that outputs $(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) \in \mathbb{R}^{256}$. As with ANP, $f_{\sigma,z}$ is processed by Eq (17) and then used to sample (with reparametrization [20]) L latent functions $\mathbf{Z}_l$. Importantly, we found that the coherence of samples improves if the model uses a *global representation* in addition to the the pixel dependent representation. We achieve this by mean-pooling half of the functional representation. Namely, we replace $\boldsymbol{z}_l$ by the channel-wise concatenation of $\boldsymbol{z}_l^{(1:64)}$ and $\textsc{mean}(\boldsymbol{z}_l^{(65:128)})$, where the mean is taken over the spatial dimensions. This latent function then goes through the post-latent CNN (CNN in Algorithm 4), as well as a linear layer to output $(f_\mu, f_\sigma) \in \mathbb{R}^{256}$. As for ANP f_μ is processed by Eq (19) and f_σ is re-scaled with Eq (18) and is spatially pooled in the case of MNIST and ZSMM to obtain homoskedastic noise.

K Additional results on image completion.

We provide additional qualitative samples and quantitative analyses for the ConvNP and ANP.

Additional ConvNP samples. Fig 10 provides further samples from a ConvNP trained with $\mathcal{L}_{\mathrm{ML}}$. We observe that the ConvNP produces reasonably diverse yet coherent samples when evaluated in

a regime that resembles the training regime (in the first four sub-columns of MNIST, SVHN, and CelebA). However, Fig 10 also demonstrates that the ConvNP struggles with context sets that are significantly different from those seen during training.

Further comparisons of ANP and ConvNP. We provide further qualitative comparisons of ConvNPs, ANPs trained with $\mathcal{L}_{\mathrm{ML}}$, and ANPs trained with $\mathcal{L}_{\mathrm{NP}}$. We omit ConvNPs trained with $\mathcal{L}_{\mathrm{NP}}$ as these are significantly outperformed by ConvNPs trained with $\mathcal{L}_{\mathrm{ML}}$ (see e.g. Tab 2).

Fig 11 shows that all models perform relatively well when context sets are drawn from a similar distribution as employed during training (first four sub-columns of MNIST, SVHN, and CelebA). Furthermore, we observe that samples from the ConvNP prior tend to be closer to samples from the underlying data distribution (e.g. for CelebA).

The qualitative advantage of ConvNP is most significant in settings that require translation equivariance for generalization. Fig 11 row 2 (ZSMM) clearly demonstrates that ConvNP generalizes to larger canvas sizes and multiple digits, while ANP attempts to reconstruct a single digit regardless of the context set. Finally, Fig 12 provides the test log-likelihood distributions of ANP and ConvNP as well as some qualitative comparisons between the two.

L Experimental Details on Environmental Data

L.1 Data Details

Table 6: Coordinates for boxes defining the train and test regions. Latitidues are given as (north, south), and longitudes as (west, east).

	Central (train)	Western (test)	Eastern (test)	Southern (test)
Latitudes	$(52, 46)$	$(50, 46)$	$(52, 49)$	$(46, 42)$
Longitudes	$(08, 28)$	$(01, 08)$	$(28, 35)$	$(19, 26)$

ERA5-Land [40] contains high resolution information on environmental variables at a 9 km spacing across the globe.[7] The data we use contains daily measurements of accumulated precipitation at 11pm and temperature at 11pm at every location, between 1981 and 2020, yielding a total of 14,304 temporal measurements across the spatial grid. In addition, we provide orography (elevation) values for each location. We normalize the data such that the precipitation values in the train set have zero mean and unit standard deviation.

We consider the task of predicting daily precipitation y, with latitude and longitude as $\boldsymbol{x}$. In addition, at each context and target location, we provide the model with access to side information in the form of orography (elevation) and temperature values. We also normalize the orography and temperature values to have zero mean and unit standard deviation. We choose a large region of central Europe as our train set, and use regions East, West and South of the train set as held out test sets (see Fig 13 and Tab 6). At train time, to sample a task, we first sample a random date between 1981 and 2020. We then sample a square subregion of grid of values from within the train region (which has size 61×201). We consider two models, one trained on 28×28 subregions, and another trained on 40×40 subregions. During training, each subregion is then split into context and target sets. Context points are randomly chosen with a keep rate p_{keep} with $p_{\mathrm{keep}} \sim \mathcal{U}[0, 0.3]$. In this section, we train only on the $\mathcal{L}_{\mathrm{ML}}$ objective.

L.2 Gaussian Process Baseline

We mean-centre the data for each task for the GP before training, and add the mean offset back for evaluation and sampling. We use an Automatic Relevance Determination (ARD) kernel, with separate factors for latitude/longitude, temperature and orography. In detail, let $\boldsymbol{x} = (x_{\mathrm{lat}}, x_{\mathrm{lon}})$ denote position, and let ω, t denote orography and precipitation respectively, and let $\boldsymbol{r} \coloneqq (\boldsymbol{x}, \omega, t)$.

[7] URL: https://www.ecmwf.int/en/era5-land. Neither the European Commission nor ECMWF is responsible for any use that may be made of the Copernicus Information or data it contains.

Then the kernel is given by

$$k(\boldsymbol{r}, \boldsymbol{r}') = \sigma_v^2 k_l(\boldsymbol{x}, \boldsymbol{x}') k_\omega(\omega, \omega') k_t(t, t') + \sigma_n^2 \delta(\boldsymbol{r}, \boldsymbol{r}').$$

Here each of k_l, k_ω and k_t are Matérn–$\frac{5}{2}$ kernels with separate learnable lengthscales; $\delta(\boldsymbol{r}, \boldsymbol{r}') = 1$ if $\boldsymbol{r} = \boldsymbol{r}'$ and 0 otherwise; and σ_v^2, σ_n^2 are learnable signal and noise variances respectively. We learn all hyperparameters by maximising the log-marginal likelihood using Scipy's implementation of L-BFGS.

Transforming the data. As the data is non-negative, we considered applying the transform $y \mapsto \log(\epsilon + y)$ for the GP to model. If $\epsilon = 0$, this would guarantee that the GP would only yield positive samples, which would be physically sensible as precipitation is non-negative. However, this cannot be done as precipitation often takes the value $y = 0$, which would lead to the transform being undefined. On the other hand, if $\epsilon > 0$, the GP samples after performing the inverse transform could still predict a precipitation value as low as $-\epsilon$, which is still unphysical. Further, a small value of ϵ leads to large distortion of the y values in transformed space. In the end, we run all experiments for the GP and NP without log-transforming the data; hence the models have to learn non-negativity.

L.3 ConvNP Architecture and Training Details

As the ERA5-Land dataset is regularly spaced, we use the on-the-grid version of the architecture, without the need for an RBF smoothing layer at the input (see App C). All experiments used a convolutional architecture with 3 residual blocks [16] for the encoder and 3 residual blocks for the decoder. Each residual block is defined with two layers of ReLU activations followed by convolutions, each with kernel size 5. The first convolution in each block is a standard convolution layer, whereas the second is depthwise separable [5]. All intermediate convolutional layers have 128 channels, and the latent function $\boldsymbol{z}$ has 16 channels. The networks were trained using ADAM with a learning rate of 10^{-4}. We used 16 channels for the latent function $\boldsymbol{z}$, and estimated $\mathcal{L}_{\mathrm{ML}}$ using 16-32 samples at train time, with batches of 8-16 images.

We train the models for between 400 and 500 epochs, where each epoch is defined as a single pass through each day in the training set, where at each day, a random subregion of the full 61×201 central Europe region is cropped. We estimated the predictive density using 2500 samples of $\boldsymbol{z}$ during test time.

L.4 Prediction and Sampling

To create Tab 3, at test time we sample 28×28 subregions from each of the train and test regions. This is done 1000 times. For the GP, we randomly restart optimisation 5 times per task and use the best hyper-parameters found. In order to remove outliers where the GP has very poor likelihood, we set a log-likelihood threshold for the GP. If the GP has a log-likelihood of less than 0 nats on a particular task, then that task is removed from the evaluation.

We find that to produce high quality samples, we need to train the model on subregions that are roughly as large as the lengthscale of the precipitation process. Hence we sample from the model trained on 40×40 subregions in Fig 4 in the main body. We show samples from the model trained on both 28×28 subregions and 40×40 subregions in App M. We also compare to samples from GPs trained on each context set (no random restarts were used for sampling).

L.5 Bayesian Optimization

We use the models described in App L.3, trained on random 28×28 subregions of the train region, and compare to the GP baselines described in App L.2. For the Bayesian optimization experiments in Fig 5 in the main body, we do not perform random restarts as this was too time-consuming. We carry out the Bayesian optimization (BayesOpt) experiments in each of the four regions: Central (train), West (test), East (test), and South (test). Each Bayesian optimization "episode" is defined by randomly sub-sampling a day (uniformly at random between 1981 and 2020), then sampling a sub-region from the tested region. To test the models' spatial generalization capacity (where possible), we sub-sample episodes from each of the four regions with the following sizes: (i) Central: 42x42, (ii) West: 40x40, (iii) East: 28x28, and (iv) South: 36x36.

Episodes begin from empty sets $D_c^{(0)} =$, and models sequentially query locations for $t = 1, \ldots, 50$. Denoting $(\boldsymbol{x}^{(t)}, y^{(t)})$ the query location and queried value at iteration t, the context set is then

updated as $D_c^{(t)} = D_c^{(t-1)} \cup \{(\boldsymbol{x}^{(t)}, y^{(t)})\}$. Denoting $\boldsymbol{y}$ as the complete set of rainfall values in the sub-region, and $\boldsymbol{y}^{(t)}$ as the set of queried values at iteration t, we can define the *instantaneous regret* as $r_t = \max(\boldsymbol{y}) - \max(\boldsymbol{y}_c^{(t)})$, and compute the average regret (plotted in Fig 5 in the main text) at the t^{th} iteration as $\bar{r}_t = \frac{1}{t}\sum_{i=1}^{t} r_i$.

M Additional Figures for Environmental Data

M.1 Predictive density

Fig 14 displays the predictive densities for precipitation at different locations, conditioned on a context set used for testing. The density of the ConvNP is estimated using 2500 samples of $\boldsymbol{z}$. To examine why the ConvNP outperforms the GP in terms of log-likelihood, we plot cases where the ConvNP likelihood is significantly better than the GP likelihood. We see that this is due to the GP occasionally making very overconfident predictions compared to the ConvNP. We also see that the ConvNP in a small proportion of cases exhibits very non-Gaussian, asymmetric predictive distribtuions.

M.2 Additional Samples

In this section we show additional samples from the model trained on 28×28 images (Figs 15 and 16) and also on 40×40 images (Figs 17 and 18). Training on larger images reduces the occurence of blocky artefacts. Fig 4 in the main body was trained on 40×40 images. Note that samples shown here are 61×201, i.e. the size of the entire central Europe train region.

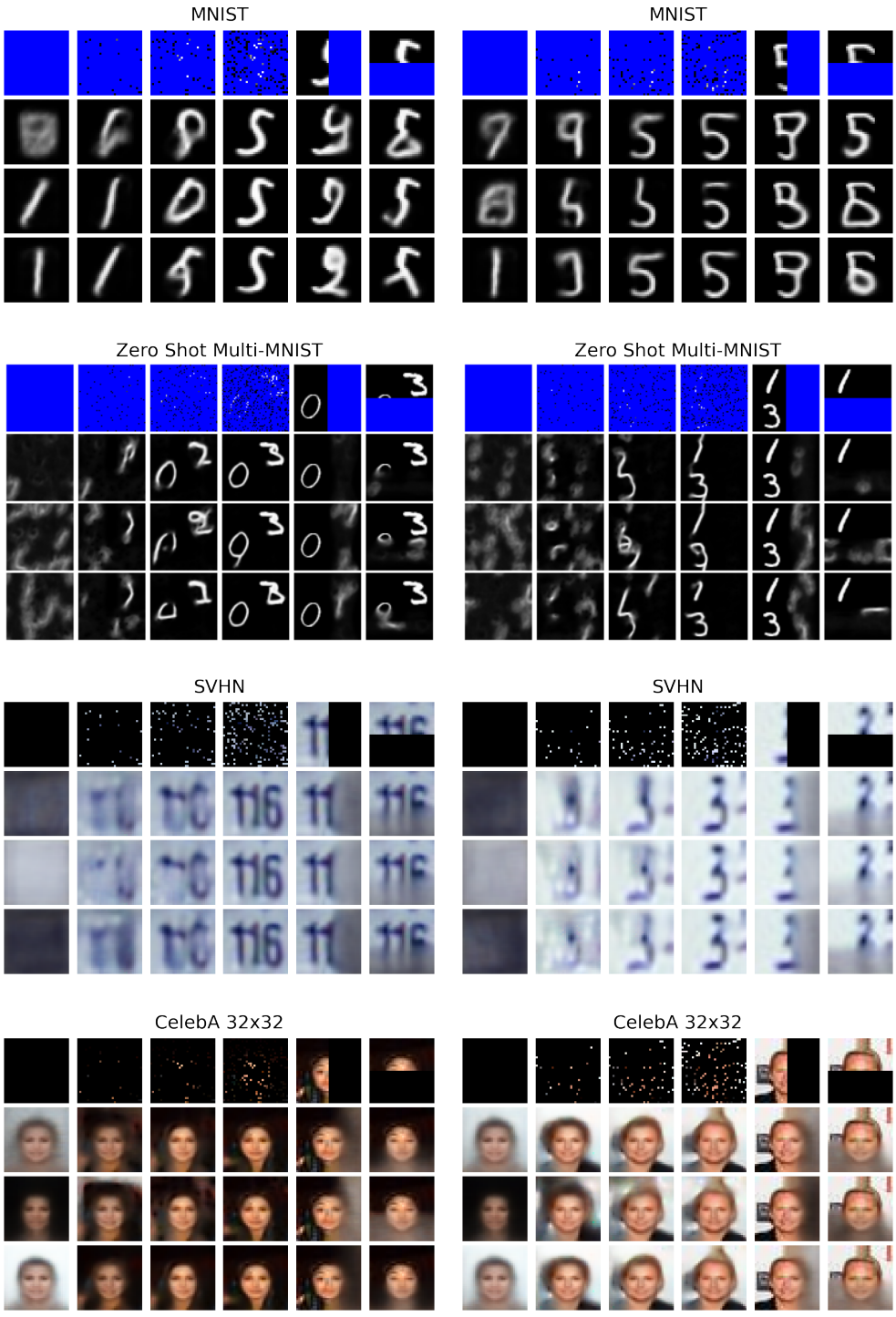
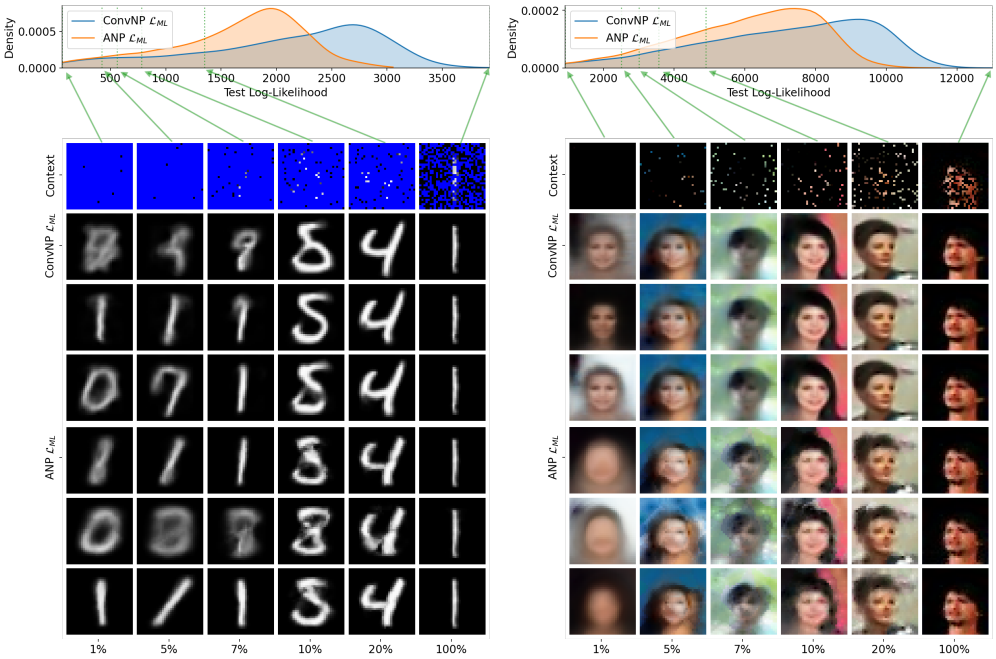


Figure 10: Qualitative samples for one of the ConvNP trained with $\mathcal{L}_{\text{ML}}$ in Tab 2. From top to bottom the four major rows correspond to MNIST, ZSMM, SVHN, CelebA32 datasets. For each dataset and each of the two major columns, a different image is randomly sampled; the first sub-row shows the given context points (missing pixels are in blue for MNIST and ZSMM but in black for SVHN and CelebA), while the next three sub-rows show the mean of the posterior predictive corresponding to different samples of the latent function. To show diverse samples we select three samples that maximize the average Euclidean distance between pixels of the samples. From left to right the first four sub-columns correspond to a context set with 0%, 1%, 3%, 10% randomly sampled context points. In the last two sub-columns, the context sets respectively contain all the pixels in the left and top half of the image.

(a) ConvNP $\mathcal{L}_{\mathrm{ML}}$ (b) ANP $\mathcal{L}_{\mathrm{ML}}$ (c) ANP $\mathcal{L}_{\mathrm{NP}}$

Figure 11: Qualitative samples between (a) ConvNP trained with $\mathcal{L}_{\mathrm{ML}}$; (b) ANP trained with $\mathcal{L}_{\mathrm{ML}}$; (c) ANP trained with $\mathcal{L}_{\mathrm{NP}}$. For each model the figure shows the same as Fig 10.



(a) MNIST

(b) CelebA32

(c) Zero Shot Multi-MNIST

(d) SVHN

Figure 12: Log-likelihood and qualitative samples comparing ConvNP and ANP trained with $\mathcal{L}_{\mathrm{ML}}$ on (a) MNIST; (b) CelebA; (c) ZSMM; (d) SVHN. For each sub-figure, the top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), followed by three samples form ConvNP (mean of the posterior predictive corresponding to different samples from the latent function), and three samples from ANP. Each column corresponds to a given percentile of the ConvNP test log likelihood (as shown by green arrows).

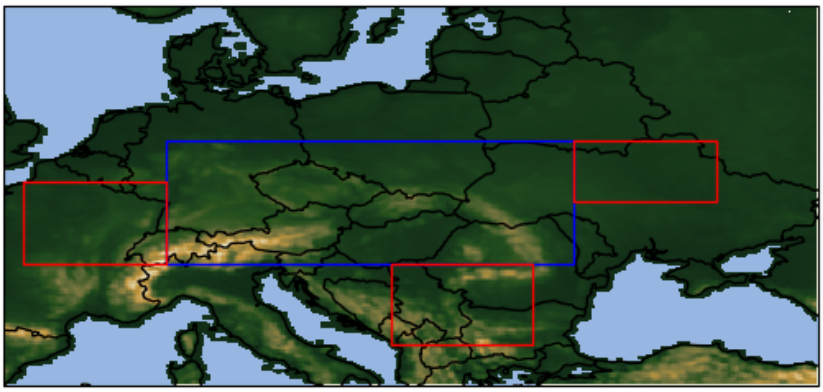
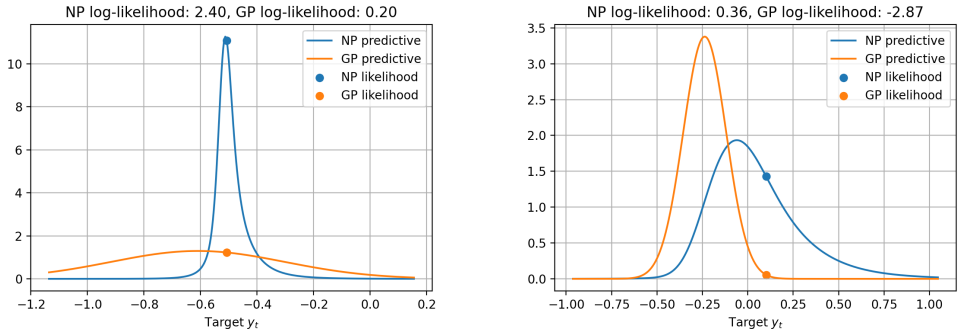
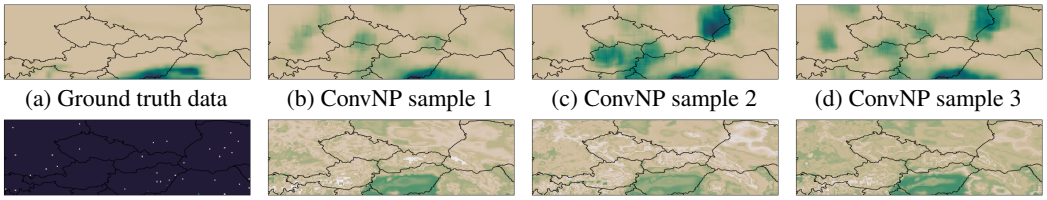


Figure 13: Training (blue) and test (red) regions in Europe, along with orography data from ERA5Land.



(a) (b)

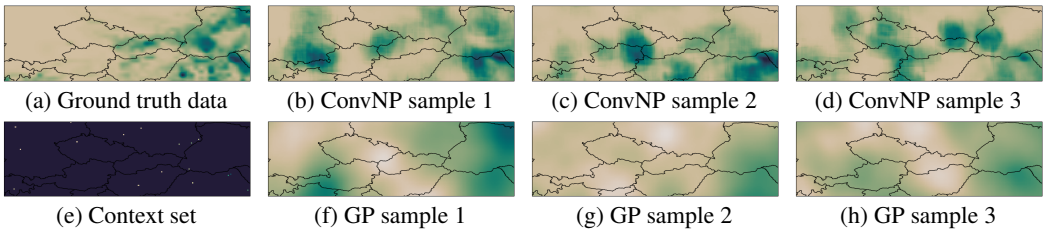
Figure 14: Predictive density at two target points, where the ConvNP significantly outperforms the GP. The orange and blue circles show the likelihood of the ground truth target value under the GP and ConvNP. Note that as the precipitation values are normalized to zero mean and unit standard deviation, $y_t = -0.53$ corresponds to no rain. In Fig 14a, we see the ConvNP sometimes produces predictions heavily centered on this value, showing it has learned the sparsity of precipitation values. In Fig 14b we see the ConvNP predictive distribution is sometimes asymmetric with a heavier positive tail, reflecting the non-negativity of precipitation.



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3

(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

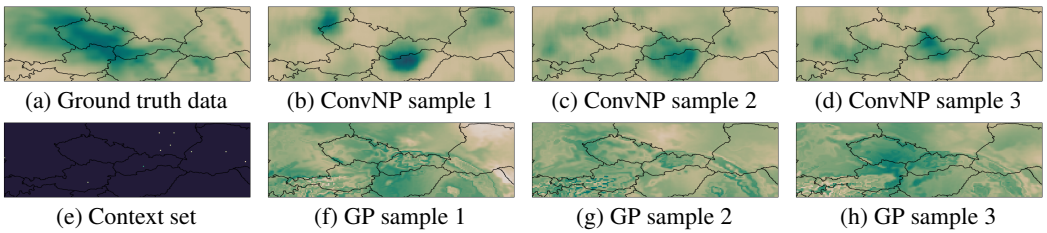
Figure 15: Samples from the predictive processes overlaid on central Europe, for a model trained on random 28×28 subregions of the full 61×201 central Europe region. Note some blocky artefacts in the ConvNP samples due to training on small subregions. Here the GP has overfit to the orography data, with samples that resemble the orography rather than precipitation.



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3

(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

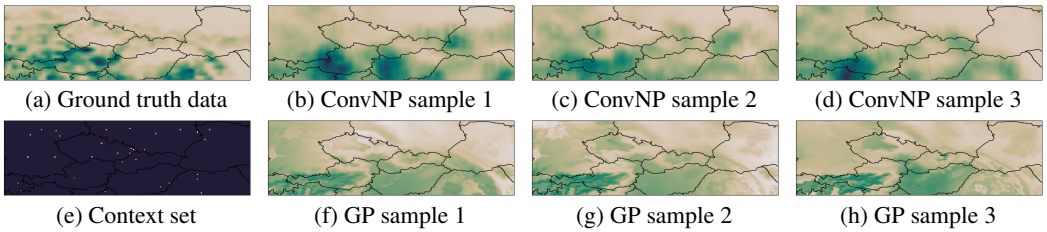
Figure 16: Samples from the predictive processes overlaid on central Europe, for a model trained on random 28×28 subregions of the full 61×201 central Europe region. Here the GP has learned a lengthscale that is too large.



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3

(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

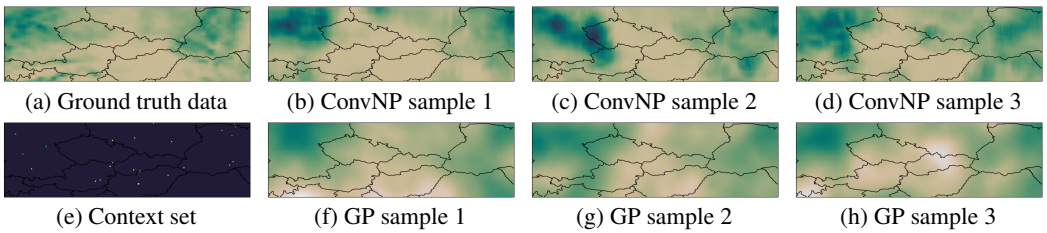
Figure 17: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region. Here the GP has overfit to the orography data, with samples that resemble the orography rather than precipitation.



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3

(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

Figure 18: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region. The GP has again overfit to the orography data.



(a) Ground truth data (b) ConvNP sample 1 (c) ConvNP sample 2 (d) ConvNP sample 3

(e) Context set (f) GP sample 1 (g) GP sample 2 (h) GP sample 3

Figure 19: Samples from the predictive processes overlaid on central Europe, for a model trained on random 40×40 subregions of the full 61×201 central Europe region.