CONVOLUTIONAL CONDITIONAL NEURAL PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce the Convolutional Conditional Neural Process (CONVCNP), a new member of the Neural Process family that models *translation equivariance* in the data. Translation equivariance is an important inductive bias for many learning problems including time series modelling, spatial data, and images. The model embeds data sets into an infinite-dimensional function space, as opposed to finite-dimensional vector spaces. To formalize this notion, we extend the theory of neural representations of sets to include *functional representations*, and demonstrate that any translation-equivariant embedding can be represented using a *convolutional deep-set*. We evaluate CONVCNPs in several settings, demonstrating that they achieve state-of-the-art performance compared to existing NPs. We demonstrate that building in translation equivariance enables zero-shot generalization to challenging, out-of-domain tasks.

1 INTRODUCTION

Neural Processes (NPs; Garnelo et al. (2018b;a)) are a rich class of models that define a conditional distribution over output variables $\boldsymbol{y}$ given input variables $\boldsymbol{x}$, parameters $\boldsymbol{\theta}$, and a set of previously observed data points in a *context set* $Z = \{\boldsymbol{x}_m, \boldsymbol{y}_m\}_{m=1}^M$, that is $p(\boldsymbol{y}|\boldsymbol{x}, Z, \boldsymbol{\theta})$. A key component of NPs is the embedding of context sets into a representation space with encoder $E(Z)$, which is achieved using a deep set function approximator (Zaheer et al., 2017). This simple model specification allows NPs to be used for (i) meta-learning (Thrun & Pratt, 2012; Schmidhuber, 1987) since predictions can be generated on the fly from new context-sets at test-time, and (ii) multi-task or transfer learning (Requeima et al., 2019) since they provide a natural way of sharing information between data sets. Moreover, conditional NPs (CNPs; (Garnelo et al., 2018a)) can be trained with maximum likelihood learning of the parameters $\boldsymbol{\theta}$, which is simple and mimics how the system is used at test time, resulting in strong performance (Gordon et al., 2019).

Natural application areas of NPs include time series, spatial data, and images with missing values. Consequently, such domains have been used extensively to benchmark current NPs (Garnelo et al., 2018a;b; Kim et al., 2019). Often, ideal solutions to prediction problems in such domains should be translation equivariant: if the data are translated in time or space, the predictions should be translated correspondingly (Kondor & Trivedi, 2018; Cohen & Welling, 2016). This relates to the notion of stationarity. As such NPs would ideally have translation equivariance built directly into the modelling assumptions as an inductive bias. However, current NP models must learn this structure from the data set instead, which is sample and parameter inefficient as well as impacting the ability of the models to generalize.

The goal of this paper is to build translation equivariance into NPs. Famously, convolutional neural networks (CNNs) added translation equivariance to standard multilayer perceptrons (LeCun et al., 1998; Cohen & Welling, 2016). However, it is not straightforward to generalize NPs in an analogous way: (i) CNNs require data to live "on the grid" (e.g. image pixels live on a regularly spaced grid), while many of the above domains have data that live "off the grid" (e.g. time series data may be observed irregularly at any time $t \in \mathbb{R}$). (ii) NPs operate on partially observed context sets whereas CNNs typically do not. (iii) NPs rely on embedding sets into a finite-dimensional vector space for which the notion of equivariance with respect to input translations is not well defined, as we detail in Section 3. In this work, we introduce the CONVCNP, a new member of the NP family that accounts for translation equivariance.[1] This is achieved by extending the theory of learning on sets to include

[1] Code will be made available upon publication.

functional representations, which in turn can be used to express any translation-equivariant NP model. Our key contributions can be summarized as follows.

(i) We provide a representation theorem for translation-equivariant functions on sets, extending a key result of Zaheer et al. (2017) to *functional embeddings*, including sets of varying size.

(ii) We extend the NP family of models to include translation equivariance.

(iii) We evaluate the CONVCNP and demonstrate that it exhibits excellent performance on several synthetic and real-world benchmarks.

2 BACKGROUND AND FORMAL PROBLEM STATEMENT

Notation. In the following, let $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^d$, $\mathcal{Y}$ compact, be the spaces of inputs and outputs (though to ease notation, we often assume $\mathcal{Y} \subseteq \mathbb{R}$). Define $\mathcal{Z}_M = (\mathcal{X} \times \mathcal{Y})^M$ as the collection of M input–output pairs, $\mathcal{Z}_{\leq M} = \bigcup_{m=1}^{M} \mathcal{Z}_M$ as the collection of at most M pairs, and $\mathcal{Z} = \bigcup_{m=1}^{\infty} \mathcal{Z}_M$ as the collection of finitely many pairs. Since we will consider *permutation-invariant* (defined later in Property 1) functions on $\mathcal{Z}$, we may refer to elements of $\mathcal{Z}$ as sets or data sets. Furthermore, we will use the notation $[n] = \{1, \ldots, n\}$.

Conditional Neural Processes (CNPs). CNPs model predictive distributions as $p(\boldsymbol{y}|\boldsymbol{x}, Z) = p(\boldsymbol{y}|\Phi(\boldsymbol{x}, Z), \boldsymbol{\theta})$. Φ is defined as a composition $\rho \circ E$ of an encoder $E\colon \mathcal{Z} \to \mathbb{R}^d$ mapping into the *embedding space* $\mathbb{R}^d$ and a decoder $\rho\colon \mathbb{R}^d \to C_b(\mathcal{X}, \mathcal{Y})$. Here $E(Z) \in \mathbb{R}^d$ is a *vector representation* of the set Z, and $C_b(\mathcal{X}, \mathcal{Y})$ is the space of continuous, bounded functions $\mathcal{X} \to \mathcal{Y}$ endowed with the supremum norm. While NPs (Garnelo et al., 2018b) employ latent variables to indirectly specify predictive distributions, in this work we focus on conditional models.

As noted by Lee et al. (2019); Bloem-Reddy & Teh (2019), the CNP form provides a tight relationship to the growing literature on learning and representing functions on sets (Zaheer et al., 2017; Qi et al., 2017; Wagstaff et al., 2019), as E is a function operating on sets. Central to this body of work is the idea that any representation or function on a set must satisfy Property 1 to be considered valid.

Property 1 ($\mathbb{S}_n$-invariant and $\mathbb{S}$-invariant functions)**.** Let $\mathbb{S}_n$ be the group of permutations of n symbols for $n \in \mathbb{N}$. A function Φ on $\mathcal{Z}_n$ is called $\mathbb{S}_n$-*invariant* if

$$\Phi(Z_n) = \Phi(\pi Z_n) \quad \text{for all } \pi \in \mathbb{S}_n \text{ and } Z_n \in \mathcal{Z}_n,$$

where the application of π to Z_n is defined as $\pi Z_n = ((\boldsymbol{x}_{\pi(1)}, \boldsymbol{y}_{\pi(1)}), \ldots, (\boldsymbol{x}_{\pi(n)}, \boldsymbol{y}_{\pi(n)}))$. A function Φ on $\mathcal{Z}$ is called $\mathbb{S}$-*invariant* if $\Phi|_{\mathcal{Z}_n}$ is $\mathbb{S}_n$-*invariant* for all n.

Zaheer et al. (2017) demonstrate that any continuous $\mathbb{S}_M$-invariant function $f\colon \mathcal{Z}_M \to \mathbb{R}$ has a *sum-decomposition* (Wagstaff et al., 2019), i.e. a representation of the form $f(Z) = \rho(\sum_{\boldsymbol{z} \in Z} \phi(\boldsymbol{z}))$ for appropriate ρ and ϕ (though this could only be shown for fixed-sized sets). This is indeed the form employed by the NP family for the encoder that embeds sets into a latent representation.

Translation equivariance. The focus of this work is on learners that are *translation equivariant*: if the input locations of the data are translated by an amount $\boldsymbol{\tau}$, the predictions should be translated correspondingly. Translation equivariance for functions operating on sets is formalized in Property 2.

Property 2 (Translation equivariant mappings on sets)**.** Let $\mathcal{H}$ be an appropriate space of functions on $\mathcal{X}$, and define T and T' as follows:

$$\begin{aligned} T &: \mathcal{X} \times \mathcal{Z} \to \mathcal{Z}, & T_{\boldsymbol{\tau}} Z &= ((\boldsymbol{x}_1 + \boldsymbol{\tau}, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_m + \boldsymbol{\tau}, \boldsymbol{y}_m)), \\ T' &: \mathcal{X} \times \mathcal{H} \to \mathcal{H}, & T'_{\boldsymbol{\tau}} h(\boldsymbol{x}) &= h(\boldsymbol{x} - \boldsymbol{\tau}). \end{aligned}$$

Then a mapping $\Phi\colon \mathcal{Z} \to \mathcal{H}$ is called *translation equivariant* if

$$\Phi(T_{\boldsymbol{\tau}} Z) = T'_{\boldsymbol{\tau}} \Phi(Z) \quad \text{for all } \boldsymbol{\tau} \in \mathcal{X} \text{ and } Z \in \mathcal{Z}.$$

3 CONVOLUTIONAL DEEP SETS

We are interested in translation equivariance (Property 2) with respect to translations on $\mathcal{X}$. The NP family encoder maps sets Z to an embedding in a vector space $\mathbb{R}^d$, for which the notion of

equivariance with respect to input translations in $\mathcal{X}$ is not well defined. For example, a function f on $\mathcal{X}$ can be translated by $\boldsymbol{\tau} \in \mathcal{X}$: $f(\cdot - \boldsymbol{\tau})$. However, for a vector $\boldsymbol{x} \in \mathbb{R}^d$, which can be seen as a function $[d] \to \mathbb{R}^d$, $\boldsymbol{x}(i) = \boldsymbol{x}_i$, the translation $\boldsymbol{x}(\cdot - \boldsymbol{\tau})$ does not make sense. To overcome this, we enrich the encoder $E\colon \mathcal{Z} \to \mathcal{H}$ to map into a *function space* $\mathcal{H}$ containing functions on $\mathcal{X}$. Since functions in $\mathcal{H}$ act on $\mathcal{X}$, our notion of translation equivariance (Property 2) does now also make sense for E. As we demonstrate below, every translation-equivariant function on sets has a representation in terms of a specific functional embedding.

Definition 1 (Functional mappings on sets and functional representations of sets)**.** Call a map $E\colon \mathcal{Z} \to \mathcal{H}$ a *functional mapping on sets* if it maps from sets $\mathcal{Z}$ to an appropriate space of functions $\mathcal{H}$. Furthermore, call $E(Z)$ the *functional representation* of the set Z.

Considering functional representations of sets leads to the key result of this work, which can be summarized as follows. For $\mathcal{Z}' \subseteq \mathcal{Z}$ appropriate, a continuous function $\Phi\colon \mathcal{Z}' \to C_b(\mathcal{X}, \mathcal{Y})$ satisfies Properties 1 and 2 if and only if it has a representation of the form

$$\Phi(Z) = \rho\left(E(Z)\right), \quad E(Z) = \textstyle\sum_{(\boldsymbol{x},\boldsymbol{y})\in Z} \phi(\boldsymbol{y})\psi(\cdot - \boldsymbol{x}) \in \mathcal{H}, \tag{1}$$

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

Equation (1) defines the encoder used by our proposed model, the ConvCNP. In Section 3.1, we present our theoretical results in more detail. In particular, Theorem 1 establishes equivalence between any function satisfying Properties 1 and 2 and the representational form in Equation (1). In doing so, we provide an extension of the key result of Zaheer et al. (2017) to functional representations on sets, and show that it can naturally be extended to handle varying-size sets. The practical implementation of ConvCNPs – the design of ρ, ϕ, and ψ – is informed by our results in Section 3.1 (as well as the proofs, provided in Appendix A), and is discussed for domains of interest in Section 4.

3.1 Representations of Translation Equivariant Functions on Sets

In this section we establish the theoretical foundation of the ConvCNP. We begin by stating a definition that is used in our main result.

Definition 2 (Multiplicity)**.** A collection $\mathcal{Z}' \subseteq \mathcal{Z}$ is said to have *multiplicity* K if, for every set $Z \in \mathcal{Z}'$, every $\boldsymbol{x}$ occurs at most K times:

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

where $[m] = \{1, \ldots, m\}$.

For example, in the case of real-world data like time series and images, we often observe only one (possibly multi-dimensional) observation per input location, which corresponds to multiplicity one. We are now ready to state our key theorem.

Theorem 1. *Consider an appropriate*[2] *collection $\mathcal{Z}'_{\leq M} \subseteq \mathcal{Z}_{\leq M}$ with multiplicity K. Then a function $\Phi\colon \mathcal{Z}'_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$ is continuous*[3]*, permutation invariant (Property 1), and translation equivariant (Property 2) if and only if it has a representation of the form*

$$\Phi(Z) = \rho\left(E(Z)\right), \quad E((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)) = \textstyle\sum_{i=1}^m \phi(y_i)\psi(\cdot - \boldsymbol{x}_i)$$

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

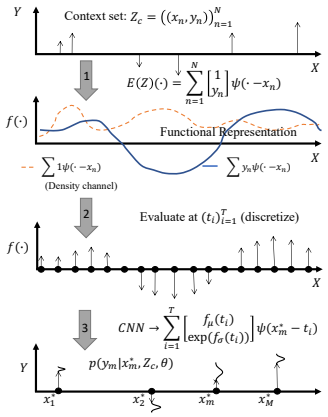
The proof of Theorem 1 is provided in Appendix A. We here discuss several key points from the proof that have practical implications and provide insights for the design of ConvCNPs: (i) For

[2] For every $m \in [M]$, $\mathcal{Z}'_{\leq M} \cap \mathcal{Z}_m$ must be closed and closed under permutations and translations.

[3] For every $m \in [M]$, the restriction $\Phi|_{\mathcal{Z}'_{\leq M} \cap \mathcal{Z}_m}$ is continuous.

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

(a)



(b)

Figure 1: (a) Pseudo-code for and (b) illustration of CONVCNP forward pass.

the construction of ρ and E, ψ is set to a flexible positive-definite kernel associated with a Reproducing Kernel Hilbert Space (RKHS; Aronszajn (1950)), which results in desirable properties for E. (ii) Using the work by Zaheer et al. (2017), we set $\phi(y) = (y^0, y^1, \cdots, y^K)$ to be the powers of y up to order K. (iii) Theorem 1 requires ρ to be a powerful function approximator of continuous, translation-equivariant maps between functions. In Section 4, we discuss how these theoretical results inform our implementations of CONVCNPs.

Theorem 1 extends the result of Zaheer et al. (2017) discussed in Section 2 by embedding the set into an infinite-dimensional space—the RKHS—instead of a finite-dimensional space. Beyond allowing the model to exhibit translation equivariance, the RKHS formalism allows us to naturally deal with finite sets of varying sizes, which turns out to be challenging with finite-dimensional embeddings. Furthermore, our formalism requires $\phi(y) = (y^0, y^1, y^2, \ldots, y^K)$ to expand up to order no more than the *multiplicity* of the sets K; if K is bounded, then our results hold for sets up to any arbitrarily large finite size M, while fixing ϕ to be only $(K+1)$-dimensional.

4 CONVOLUTIONAL CONDITIONAL NEURAL PROCESSES

In this section we discuss the architectures and implementation details for CONVCNPs. Similar to NPs, CONVCNPs model the conditional distribution as

$$p(\boldsymbol{y}|\boldsymbol{x}, Z) = p(\boldsymbol{y}|\Phi_{\boldsymbol{\theta}}(Z)(\boldsymbol{x})) = \mathcal{N}(\boldsymbol{y}; \Phi_\mu, \Phi_\sigma) \quad \text{with} \quad (\Phi_\mu, \Phi_\sigma) = \Phi_{\boldsymbol{\theta}}(Z)(\boldsymbol{x}),$$

where Z is the observed data and Φ a CONVDEEPSET. The key considerations are the design of ρ, ϕ, and ψ for Φ. We provide separate models for data that lie on- and off-the-grid.

Specifying ϕ. The applications considered in this work have a single (potentially multi-dimensional) output per input location, so the multiplicity of $\mathcal{Z}$ is one (i.e., $K = 1$). It then suffices to let ϕ be a power series of order one, which is equivalent to appending a constant to $\boldsymbol{y}$ in all data sets, i.e. $\phi(\boldsymbol{y}) = [1, \boldsymbol{y}]^\top$. In this design, the first output ϕ_1 can be thought of as a "density channel" providing the model with information regarding where data has been observed. Without a density channel, the model would be unable to distinguish between no observed datapoint at $\boldsymbol{x}$ and a datapoint at $\boldsymbol{x}$ with $\boldsymbol{y} = \boldsymbol{0}$. Additionally, we found it helpful to divide the signal channels $\boldsymbol{y}$ by the density channel (Figure 1a, line 5) as this improved performance when there is large variation in the density of input locations. In the image processing literature, this is known as *normalized convolutions* (Knutsson & Westin, 1993). The normalization operation can be reversed by ρ and is therefore not restrictive.

CONVCNPs for off-the-grid data. Having specified ϕ, it remains to specify the form of ψ and ρ. Our proof of Theorem 1 suggests that ψ should be a stationary, positive-definite kernel. The exponentiated-quadratic (EQ) kernel with a learnable length scale parameter is a natural choice. This kernel is multiplied by ϕ to form the functional representation $E(Z)$ (Figure 1a, line 4; and Figure 1b, arrow 1).

Next, Theorem 1 suggests that ρ should be a continuous, translation-equivariant map between function spaces. Kondor & Trivedi (2018) show that, in deep learning, any translation-equivariant model has a representation as a CNN. However, CNNs operate on discrete (on-the-grid) input spaces and produce discrete outputs. In order to approximate ρ with a CNN, we discretize the input of ρ, then apply the CNN, and finally transform the CNN output back to a continuous function space. To do this, for each context and test set, we space points $(\boldsymbol{t}_i)_{i=1}^n \subseteq \mathcal{X}$ on a uniform grid (at a pre-specified density) over a hyper-cube that covers both the context and target inputs. We then evaluate $(E(Z)(\boldsymbol{t}_i))_{i=1}^n$ (Figure 1a, lines 2–3; Figure 1b, arrow 2). This discretized representation of $E(Z)$ is then passed through a CNN (Figure 1a, line 6; Figure 1b, arrow 3).

The output of the CNN must finally be mapped back to a continuous function space. We do this by using the CNN outputs as weights for evenly-spaced basis functions, again using the EQ kernel, which we denote by ψ_ρ (Figure 1a, lines 7–8; Figure 1b, arrow 3). The resulting approximation to ρ is not perfectly equivariant, but will be approximately equivariant for length scales larger than the spacing of $(E(Z)(\boldsymbol{t}_i))_{i=1}^n$. The resulting continuous functions are then used to generate the (Gaussian) predictive mean and variance at any input. This, in turn, can be used to evaluate the log-likelihood.

ConvCNP for on-the-grid data. While ConvCNP is readily applicable to many settings where data live on a grid, in this work we focus on the image setting. As such, the following description uses the image completion task as an example, which is often used to benchmark NPs (Garnelo et al., 2018a; Kim et al., 2019). Compared to the continuous case, the implementation becomes simpler as we can choose the discretization $(\boldsymbol{t}_i)_{i=1}^n$ to be the pixel locations. Let I be the image, and let $\mathrm{M}_c, \mathrm{M}_t$ be context and target masks respectively: $[\mathrm{M}_c]_{i,j} = 1$ if pixel location (i, j) is in the context set, else it is 0. Hence the context set can be represented as $\mathrm{M}_c \odot \mathrm{I}$. Then the on-the-grid algorithm can be succinctly written as $(\boldsymbol{\mu}, \log(\boldsymbol{\sigma})) = \mathrm{CNN}([\textsc{conv}(\mathrm{M}_c); \textsc{conv}(\mathrm{M}_c \odot \mathrm{I})/\textsc{conv}(\mathrm{M}_c)]^\top)$ where CONV is a convolutional layer (discrete version of Figure 1a, line 4). Although the theory suggests using a positive-definite kernel for CONV, we have not found significant empirical differences between an EQ kernel and using a fully trainable kernel restricted to positive values (see Appendices D.4 and D.5 for details). Here $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ are the image mean and standard deviation, and is only evaluated at the target locations defined by M_t. $[\mathrm{M}_c; \mathrm{M}_c \odot \mathrm{I}]^\top$ are the discretized version of the density channel and the signal channel, which is then normalized element-wise by the density channel.

Training. Denoting the data set $D = \{Z_n\}_{n=1}^N \subseteq \mathcal{Z}$ and the parameters by $\boldsymbol{\theta}$, maximum-likelihood training involves (Garnelo et al., 2018a;b)

$$\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta} \in \Theta} \sum_{n=1}^N \sum_{(\boldsymbol{x}, y) \in Z_{n,t}} \log p\left(y \mid \Phi_{\boldsymbol{\theta}}(Z_{n,c})(\boldsymbol{x})\right), \tag{2}$$

where we have split Z_n into context ($Z_{n,c}$) and target ($Z_{n,t}$) sets. This is standard practice in both the NP (Garnelo et al., 2018a;b) and meta-learning settings (Finn et al., 2017; Gordon et al., 2019) and relates to neural auto-regressive models (Requeima et al., 2019). Practically, stochastic gradient descent methods (Bottou, 2010) can be used to perform the optimization.

5 Experiments and Results

We evaluate the performance of ConvCNPs in both on-the-grid and off-the-grid settings focusing on two central questions: (i) Do translation-equivariant models improve performance in appropriate domains? (ii) Can translation equivariance enable ConvCNPs to generalize to settings outside of those encountered during training? We use several off-the-grid data-sets which are irregularly sampled time series ($\mathcal{X} = \mathbb{R}$), comparing to Gaussian processes (GP; Williams & Rasmussen (2006)), and AttnCNP, the best performing member of the CNP family. We then evaluate on several on-the-grid image data sets ($\mathcal{X} = \mathbb{Z}^2$). In all settings we demonstrate substantial improvements over existing neural process models. For the CNN component of our model, we propose a small and large architecture for each experiment (in the experimental sections named ConvCNP and ConvCNPXL, respectively). We note that these architectures are different for off-the-grid and on-the-grid experiments, with full details regarding the architectures given in the appendices.

5.1 Synthetic 1D Experiments

First we consider synthetic regression problems. At each iteration, a function is sampled, followed by context and target sets. Beyond EQ-kernel GPs (as proposed in Garnelo et al. (2018a); Kim et al.

(2019)), we consider more complex data arising from Matern–$\frac{5}{2}$ and weakly-periodic kernels, as well as a challenging, non-Gaussian sawtooth process with random shift and frequency (see Figure 2, for example). CONVCNP is compared to CNP (Garnelo et al., 2018a) and ATTNCNP. Training and testing procedures are fixed across all models. Full details on models, data generation, and training procedures are provided in Appendix C.2.

Table 1: Log-likelihood from synthetic 1-dimensional experiments.

Model	Params	EQ	Weak Periodic	Matern	Sawtooth
CNP	66818	0.88 ± 3e-3	-1.10 ± 2e-3	-0.78 ± 1e-3	-0.16 ± 1e-5
ATTNCNP	149250	2.58 ± 4e-3	-1.10 ± 2e-3	-0.42 ± 2e-3	0.33 ± 2e-3
CONVCNP	6537	2.06 ± 5e-3	-1.14 ± 2e-3	0.37 ± 4e-3	2.21 ± 4e-3
CONVCNPXL	50617	**2.93 ± 4e-3**	**-0.41 ± 2e-3**	**0.50 ± 4e-3**	**2.66 ± 1e-3**

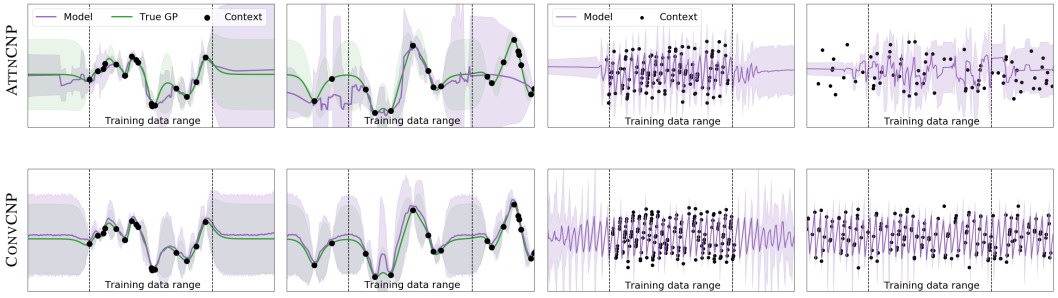


Figure 2: Example functions learned by the ATTNCNP (top row), and CONVCNP (bottom row), when trained on a Matern–$\frac{5}{2}$ kernel with length scale 0.25 (first and second column) and sawtooth function (third and fourth column). Columns one and three show the predictive posterior of the models when data is presented in same range as training, with predictive posteriors continuing beyond that range on either side. Columns two and four show model predictive posteriors when presented with data outside the training data range. Plots show means and two standard deviations.

Table 1 reports the log-likelihood means and standard errors of the models over 1000 tasks. The context and target points for both training and testing lie within the interval $[-2, 2]$ where training data was observed (marked "training data range" in Figure 2). Table 1 demonstrates that, even when extrapolation is not required, CONVCNP significantly outperforms other models in all cases, despite having fewer parameters.

Figure 2 demonstrates that CONVCNP generates excellent fits, even for challenging functions such as Matern–$\frac{5}{2}$ kernels and sawtooth. Moreover, Figure 2 compares the performance of CONVCNP and ATTNCNP when data is observed outside the range where the models where trained: translation equivariance enables CONVCNP to elegantly generalize to this setting, whereas ATTNCNP is unable to generate reasonable predictions.

5.2 PLASTICC EXPERIMENTS

The PLAsTiCC data set (Allam Jr et al., 2018) is a simulation of transients observed by the LSST telescope under realistic observational conditions. The data set contains 3,500,734 "light curves", where each measurement is of an object's brightness as a function of time— taken by measuring the photon flux in six different astronomical filters. The data can be treated as a six-dimensional time-series. The data set was introduced in a Kaggle competition,[4] where the task was to use these light curves to classify the variable sources. The winning entry– Avocado (Boone, 2019) –modeled the light curves with GPs and used these models to generate features for a gradient boosted decision tree classifier. We compare a multi input / output CONVCNP with the GP models used in Avocado.[5] CONVCNP accepts six channels as inputs, one for each astronomical filter, and returns 12

[4]https://www.kaggle.com/c/PLAsTiCC-2018

[5]Full code for Avocado, including GP models, is available at https://github.com/kboone/avocado.

outputs - the means and standard deviations of six Gaussians. Full experimental details are given in Appendix C.3. The mean square error of both approaches is similar, but the held-out log-likelihood from the CONVCNP is far higher (see table 2).

Table 2: Mean and standard errors of log-likelihood and root mean square error over 1000 test objects from the PLastiCC dataset.

Model	Log-likelihood	MSE
Kaggle GP (Boone, 2019)	-0.335 $\pm$ 0.09	0.037 $\pm$ 4e-3
ConvCP (ours)	**1.31** $\pm$ 0.30	0.040 $\pm$ 5e-3

5.3 PREDATOR-PREY MODELS: SIM2REAL

The CONVCNP model is well suited for applications where simulation data is plentiful, but real world training data is scarce (Sim2Real). The CONVCNP can be trained on a large amount of simulation data and then be deployed with real-world training data as a context-set. We consider the Lotka–Volterra model (Wilkinson, 2011) which is used to describe the evolution of predator-prey populations. This model has been used in the Approximate Bayesian Computation literature where the task is to infer the parameters from samples drawn from the Lotka–Volterra process (Papamakarios & Murray, 2016). These methods do not simply extend to prediction problems such as interpolation or forecasting. In contrast, we train CONVCNP on synthetic data sampled from the Lotka–Volterra model, and condition on real-world data from the Hudson's Bay lynx-hare data set (Leigh, 1968) to perform interpolation (see Figure 3; full experimental details are given in Appendix C.4). The CONVCNP performs accurate interpolation as shown in Figure 3. We were unable to successfully train the ATTNCNP for this task. We suspect this is because the simulation data are variable length-time series, which requires models to leverage translation equivariance at training time. As shown in Section 5.1, the ATTNCNP struggles with this setting (see Appendix C.4 for complete details).

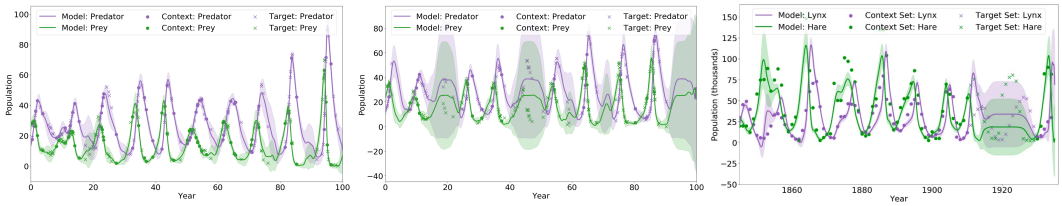


Figure 3: Left and centre: two samples from the Lotka–Volterra process (sim). Right: CONVCNP trained on simulations and applied to the Hudson's Bay lynx-hare dataset (real). Plots show means and two standard deviations.

5.4 2D IMAGE COMPLETION EXPERIMENTS

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

Table 3: Log-likelihood from image experiments (6 runs).

Model	Params	MNIST	SVHN	CelebA32	CelebA64	ZSMM
ATTNCNP	410k	1.08 $\pm$0.04	3.94 $\pm$0.02	3.18 $\pm$0.02		-0.83 $\pm$0.08
CONVCNP	181k	1.19 $\pm$0.01	3.89 $\pm$0.01	3.19 $\pm$0.02	3.64 $\pm$0.01	**1.21 $\pm$0.00**
CONVCNPXL	408k	**1.26 $\pm$0.01**	**3.97 $\pm$0.03**	**3.35 $\pm$0.02**	**3.70 $\pm$0.01**	0.30 $\pm$0.75

Standard Benchmarks. We first evaluate the model on four common benchmarks: MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), as well as 32×32 and 64×64 CelebA (Liu et al., 2018). Importantly, these data sets are biased towards images containing a single well centered object. As a result, perfect translation-equivariance might hinder the performance of the model when the test data are similarly structured. We therefore also evaluated a larger CONVCNP that can learn such non-stationarity, while still sharing parameters across the input space (CONVCNPXL). To understand the importance of the model size for translation-equivariance, note that a $k \times k$ kernel applied to an image of the same size is equivalent to a feed forward layer on the flattened input. More generally, if the receptive field of the CNN – the region of the input space that contributes to all activations of the last CNN layer – is larger than the input image, then the model can learn position specific features.

Table 3 shows that CONVCNP significantly outperforms ATTNCNP when it has a large receptive field size, while being at least as good with a small receptive field size. Qualitative samples for various context sets can be seen in Figure 5. Further qualitative comparisons and ablation studies can be found in Appendix D.3 and Appendix D.4 respectively.

Beyond the performance improvements, a key advantage of the CONVCNP is its computational efficiency. With a batch size of 16 on 32×32 MNIST, CONVCNPXL requires 945MB of VRAM, while ATTNCNP requires 5839 MB. For the 56×56 ZSMM CONVCNPXL increases its requirements to 1443 MB, while ATTNCNP could not fit onto a 32GB GPU. Ultimately, ATTNCNP had to be trained with a batch size of 6 (using 19139 MB) and we were not able to fit it for CelebA64. Recently, restricted attention has been proposed to overcome this computational issue (Parmar et al., 2018), but we leave an investigation of this and its relationship to CONVCNPs to future work.

Generalizing to Natural Images. The data sets considered in the previous section were centered and contained single objects. Here we test whether CONVCNPs trained on such data can generalise to images containing multiple, non-centered objects.

The last column of Table 3 evaluates the models in a zero shot multi-MNIST (ZSMM) setting, where images contain multiple digits at test time (Appendix D.2). CONVCNP significantly outperforms ATTNCNP on such tasks. Figure 4a shows a histogram of the image log-likelihoods for CONVCNP and ATTNCNP, as well as qualitative results at different percentiles of the CONVCNP distribution. CONVCNP seems able to extrapolate to this out-of-distribution test set, while ATTNCNP appears to model the bias of the training data and predict a centered "mean" digit independently of the context.

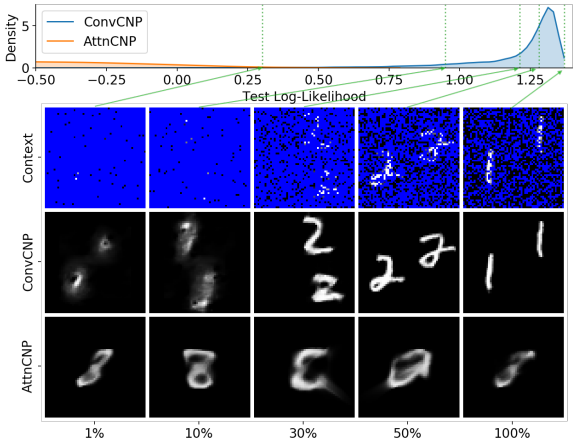
Although ZSMM is a contrived task, note that our field of view usually contains multiple independent objects, thereby requiring translation equivariance. As a more realistic example, we took a CONVCNP model trained on CelebA and tested it on a natural image containing multiple people (Figure 4b). Even with 95% of the pixels removed, the CONVCNP was able to produce a qualitatively reasonable reconstruction. A comparison with ATTNCNP can be seen in Appendix D.3.

6 DISCUSSION AND RELATED WORK

We have introduced CONVCNP, a new member of the CNP family that leverages embedding sets into function space to achieve translation equivariance. The relationship to (i) the NP family, and (ii) representing functions on sets each imply extensions and avenues for future work.

Two key issues in the existing theory on learning with sets (Zaheer et al., 2017; Qi et al., 2017) are (i) the restriction to fixed-size sets, and (ii) the dimensionality of the embedding space must be no less than the cardinality of the embedded sets. Our work implies that by considering embeddings into a function space, both issues (under certain assumptions), are somewhat alleviated. In future work, we aim to further this analysis and formalize it in a more general context.

In this work, we only considered Gaussian conditional distributions. However, there are several lines of research, e.g. normalizing flows (Rezende & Mohamed, 2015), that allow models to go beyond Gaussian distributions. Further, the link to NPs (Garnelo et al., 2018b) implies that consideration of latent-variable extensions (which enforce consistency and add non-Gaussianity) may be useful. Consistency is particularly important for drawing auto-regressive (AR) samples, which is currently under-used in NP models. AR sampling provides a link between the NP family and neural AR models (Salimans et al., 2017; Parmar et al., 2018), but loses consistency. Neural approaches based on the ideas of exchangeability might provide a solution (Korshunova et al., 2018).



(a) Log likelihood and qualitative results on ZSMM. The top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), ConvCNP target predictions (middle), and AttnCNP target predictions (bottom). Each column corresponds to a given percentile of the ConvCNP distribution.

(b) Qualitative evaluation of a ConvCNPXL trained on the unscaled CelebA (218×178) and tested on Ellen's Oscar unscaled (337×599) selfie (DeGeneres, 2014) with 5% of the pixels as context (top).

Figure 4: Zero shot generalization to tasks that require translation equivariance.

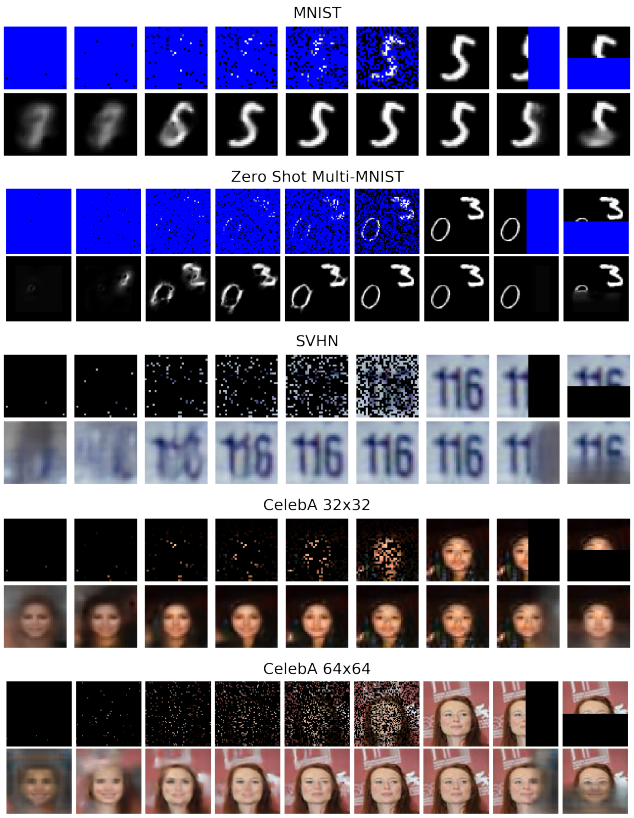


Figure 5: Qualitative evaluation of the ConvCNP(XL). For each dataset, an image is randomly sampled, the first row shows the given context points while the second is the mean of the estimated conditional distribution. From left to right the first seven columns correspond to a context set with 3, 1%, 5%, 10%, 20%, 30%, 50%, 100% randomly sampled context points. In the last 2 columns, the context sets respectively contain all the pixels in the left and top half of the image. ConvCNPXL is shown for all datasets besides ZSMM, for which we show the fully translation equivariant ConvCNP.

REFERENCES

Tarek Allam Jr, Anita Bahmanyar, Rahul Biswas, Mi Dai, Lluís Galbany, Renée Hložek, Emille EO Ishida, Saurabh W Jha, David O Jones, Richard Kessler, et al. The photometric lsst astronomical time-series classification challenge (plasticc): Data set. *arXiv preprint arXiv:1810.00001*, 2018.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*, 2019.

Kyle Boone. Avocado: Photometric classification of astronomical transients with gaussian process augmentation. *arXiv preprint arXiv:1907.04690*, 2019.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

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

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

Ellen DeGeneres. If only Bradley's arm was longer. Best photo ever. Oscars pic.twitter.com/c9u5notgap, Mar 2014.

James Dugundji et al. An extension of tietze's theorem. *Pacific Journal of Mathematics*, 1(3): 353–367, 1951.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

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 (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018a. PMLR. URL http://proceedings.mlr.press/v80/garnelo18a.html.

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

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

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.

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*, pp. 770–778, 2016.

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.

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

Hans Knutsson and C-F Westin. Normalized and differential convolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 515–523. IEEE, 1993.

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 (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2747–2755, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Iryna Korshunova, Yarin Gal, Joni Dambre, and Arthur Gretton. Conditional bruno: A deep recurrent process for exchangeable labelled data. In *Bayesian Deep Learning NeurIPS Workshop*, 2018.

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.

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.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Egbert R Leigh. The ecological role of volterra's equations. *Some mathematical problems in biology*, 1968.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

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

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 1028–1036. 2016.

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

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.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. *arXiv preprint arXiv:1906.07697*, 2019.

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

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

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.

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

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

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6487–6494, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2011.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

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 (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3391–3401. 2017.

A THEORETICAL RESULTS AND PROOFS

In this section, we provide the proof of Theorem 1. We begin with definitions that we will use throughout the section and then present our results.

Let $\mathcal{X} = \mathbb{R}^N$ and let $\mathcal{Y} \subseteq \mathbb{R}$ be compact. Let ψ be a symmetric, positive-definite kernel on $\mathcal{X}$. By the Moore–Aronszajn Theorem, there is a unique Hilbert space $(\mathcal{H}, \langle \cdot\,, \cdot \rangle_{\mathcal{H}})$ of real-valued functions on $\mathcal{X}$ for which ψ is a reproducing kernel. This means that (i) $\psi(\cdot\,, \boldsymbol{x}) \in \mathcal{H}$ for all $\boldsymbol{x} \in \mathcal{X}$ and (ii) $\langle f, \psi(\cdot\,, \boldsymbol{x}) \rangle_{\mathcal{H}} = f(\boldsymbol{x})$ for all $f \in \mathcal{H}$ and $\boldsymbol{x} \in \mathcal{X}$ (reproducing property).

Definition 3 (Interpolating RKHS)**.** Call $\mathcal{H}$ *interpolating* if it interpolates any finite number of points: for every $(\boldsymbol{x}_i, y_i)_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$ with $(\boldsymbol{x}_i)_{i=1}^n$ all distinct, there is an $f \in \mathcal{H}$ such that $f(\boldsymbol{x}_1) = y_1, \ldots, f(\boldsymbol{x}_n) = y_n$.

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

A.1 THE QUOTIENT SPACE $\mathbb{R}^n / \mathbb{S}_n$

For $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$, let $\boldsymbol{x} \sim \boldsymbol{y}$ if $\boldsymbol{x}$ is a permutation of $\boldsymbol{y}$. Let $\mathbb{R}^n / \mathbb{S}_n$ be the collection of equivalence classes, where we denote the equivalence class of $\boldsymbol{x}$ by $[\boldsymbol{x}]$. For $A \subseteq \mathbb{R}^n$, denote $[A] = \{[\boldsymbol{a}] : \boldsymbol{a} \in A\}$. Call the map $\boldsymbol{x} \mapsto [\boldsymbol{x}] \colon \mathbb{R}^n \to \mathbb{R}^n / \mathbb{S}_n$ the canonical map.

On $\mathbb{R}^n$, since all norms on finite-dimensional vector space are equivalent, without loss of generality consider the 2-norm, which we note is permutation invariant: $\|\pi \cdot\|_2 = \|\cdot\|_2$ for all $\pi \in \mathbb{S}_n$. On $\mathbb{R}^n / \mathbb{S}_n$, define

$$d \colon \mathbb{R}^n / \mathbb{S}_n \times \mathbb{R}^n / \mathbb{S}_n \to [0, \infty), \quad d([\boldsymbol{x}], [\boldsymbol{y}]) = \min_{\pi \in \mathbb{S}_n} \|\boldsymbol{x} - \pi \boldsymbol{y}\|_2.$$

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

Proposition 1. *The function d is a metric.*

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

$$\|\boldsymbol{x} - \pi_1\pi_2\boldsymbol{y}\|_2 \leq \|\boldsymbol{x} - \pi_1\boldsymbol{z}\|_2 + \|\pi_1\boldsymbol{z} - \pi_1\pi_2\boldsymbol{y}\|_2 = \|\boldsymbol{x} - \pi_1\boldsymbol{z}\|_2 + \|\boldsymbol{z} - \pi_2\boldsymbol{y}\|_2,$$

using permutation invariance of d. Hence, taking the minimum over π_1,

$$d([\boldsymbol{x}], [\boldsymbol{y}]) \leq d([\boldsymbol{x}], [\boldsymbol{z}]) + \|\boldsymbol{z} - \pi_2\boldsymbol{y}\|_2,$$

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

Proposition 2. *The canonical map $\mathbb{R}^n \to \mathbb{R}^n / \mathbb{S}_n$ is continuous.*

Proof. Follows directly from $d([\boldsymbol{x}], [\boldsymbol{y}]) \leq \|\boldsymbol{x} - \boldsymbol{y}\|_2$. □

Proposition 3. *Let $A \subseteq \mathbb{R}^n$ be topologically closed and closed under permutations. Then $[A]$ is topologically closed.*

Proof. Consider a sequence $([\boldsymbol{a}_n])_{n=1}^\infty \subseteq [A]$ converging to some $[\boldsymbol{x}]$. Then there are permutations $(\pi_n)_{n=1}^\infty \subseteq \mathbb{S}_n$ such that $\pi_n \boldsymbol{a}_n \to x$. Here $\pi_n \boldsymbol{a}_n \in A$, because A is closed under permutations. Thus $\boldsymbol{x} \in A$, as A is also topologically closed. We conclude that $[\boldsymbol{x}] \in [A]$. □

Proposition 4. *Let $A' \subseteq \mathbb{R}^n / \mathbb{S}_n$ be closed and bounded. Then A' is compact. In other words, $\mathbb{R}^n / \mathbb{S}_n$ has the Heine–Borel property.*

Proof. Let $A \subseteq \mathbb{R}^n$ be the preimage of A' under the canonical map. By continuity of the canonical map, A is closed. Furthermore, note that

$$\|\pi \boldsymbol{x} - 0\|_2 = \|\boldsymbol{x}\|_2 \implies d([\boldsymbol{x}], 0) = \|\boldsymbol{x}\|_2$$

so boundedness of A' implies boundedness of A. Since A is closed and bounded, by the Heine–Borel property of $\mathbb{R}^n$, A is compact, which means that $A' = [A]$ is compact by continuity of the canonical map. □

A.2 Embeddings of Sets Into an RKHS

Lemma 3 states that it is possible to homeomorphically embed sets into an RKHS. This result is key to proving our main result.

Lemma 1. *Consider a collection $\mathcal{Z}'_M \subseteq \mathcal{Z}_M$ that has multiplicity K. Set*

$$\phi : \mathcal{Y} \to \mathbb{R}^{K+1}, \quad \phi(y) = (y^0, y^1, \cdots, y^K)$$

and let ψ be an interpolating, continuous positive-definite kernel. Define

$$\mathcal{H}_M = \left\{ \sum_{i=1}^{M} \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i) : (\boldsymbol{x}_i, y_i)_{i=1}^{M} \subseteq \mathcal{Z}'_M \right\} \subseteq \mathcal{H}^{K+1}, \tag{3}$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$-dimensional-vector–valued RKHS constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel. Then the embedding

$$E\colon [\mathcal{Z}'_M] \to \mathcal{H}_M, \quad E([(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_M, y_M)]) = \sum_{i=1}^{M} \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i)$$

is continuous and injective, hence invertible.

Proof. First, we show that E is injective. Suppose that

$$\sum_{i=1}^{M} \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i) = \sum_{i=1}^{M} \phi(y'_i)\psi(\,\cdot\,, \boldsymbol{x}'_i).$$

Denote $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M)$ and $y = (y_1, \ldots, y_M)$, and denote $\boldsymbol{x}'$ and y' similarly. Taking the inner product with any $f \in \mathcal{H}$ on both sides and using the reproducing property of ψ, this implies that

$$\sum_{i=1}^{M} \phi(y_i) f(\boldsymbol{x}_i) = \sum_{i=1}^{M} \phi(y'_i) f(\boldsymbol{x}'_i)$$

for all $f \in \mathcal{H}$. In particular, since by construction $\phi_1(\,\cdot\,) = 1$,

$$\sum_{i=1}^{M} f(\boldsymbol{x}_i) = \sum_{i=1}^{M} f(\boldsymbol{x}'_i)$$

for all $f \in \mathcal{H}$. Using that $\mathcal{H}$ is interpolating, choose a particular $\hat{\boldsymbol{x}} \in \boldsymbol{x} \cup \boldsymbol{x}'$, and let $f \in \mathcal{H}$ be such that $f(\hat{\boldsymbol{x}}) = 1$ and $f(\,\cdot\,) = 0$ at all other $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$. Then

$$\sum_{i : \boldsymbol{x}_i = \hat{\boldsymbol{x}}} 1 = \sum_{i : \boldsymbol{x}'_i = \hat{\boldsymbol{x}}} 1,$$

so the number of such $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}$ and the number of such $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}'$ are the same. Since this holds for every $\hat{\boldsymbol{x}}$, $\boldsymbol{x}$ is a permutation of $\boldsymbol{x}'$: $\boldsymbol{x} = \pi(\boldsymbol{x}')$ for some permutation $\pi \in \mathbb{S}_M$. Plugging in the permutation, we can write

$$\sum_{i=1}^{M} \phi(y_i) f(\boldsymbol{x}_i) = \sum_{i=1}^{M} \phi(y'_i)\psi(\,\cdot\,, \boldsymbol{x}'_i) \overset{(\boldsymbol{x}' = \pi^{-1}(\boldsymbol{x}))}{=} \sum_{i=1}^{M} \phi(y'_i) f(\boldsymbol{x}_{\pi^{-1}(i)}) \overset{(i \leftarrow \pi^{-1}(i))}{=} \sum_{i=1}^{M} \phi(y'_{\pi(i)}) f(\boldsymbol{x}_i).$$

Then, by a similar argument, for any particular $\hat{\boldsymbol{x}}$,

$$\sum_{i:\boldsymbol{x}_i=\hat{\boldsymbol{x}}} \phi(y_i) = \sum_{i:\boldsymbol{x}_i=\hat{\boldsymbol{x}}} \phi(y'_{\pi(i')}).$$

Since $\mathcal{Z}'_M$ has multiplicity K, the number of terms is less than or equal to K. Therefore, by Lemma 4 from Zaheer et al. (2017),

$$(y_i)_{i:\boldsymbol{x}_i=\hat{\boldsymbol{x}}} \quad \text{is a permutation of} \quad (y'_{\pi(i)})_{i:\boldsymbol{x}_i=\hat{\boldsymbol{x}}}.$$

Note that $\boldsymbol{x}_i = \hat{\boldsymbol{x}}$ for all above y_i. Furthermore, note that also $\boldsymbol{x}'_{\pi(i)} = \boldsymbol{x}_i = \hat{\boldsymbol{x}}$ for all above $y'_{\pi(i)}$. We may therefore adjust the permutation π such that $y_i = y'_{\pi(i)}$ for all i such that $\boldsymbol{x}_i = \hat{\boldsymbol{x}}$ whilst retaining that $\boldsymbol{x} = \pi(\boldsymbol{x}')$. Performing this adjustment for all $\hat{\boldsymbol{x}}$, we find that $y = \pi(y')$ and $\boldsymbol{x} = \pi(\boldsymbol{x}')$.

Second, we show that E is continuous. Compute

$$\begin{aligned}
&\left\| \sum_{i=1}^{M} \phi(y_i)\psi(\,\cdot\,,\boldsymbol{x}_i) - \sum_{j=1}^{M} \phi(y'_j)\psi(\,\cdot\,,\boldsymbol{x}'_j) \right\|_{\mathcal{H}}^2 \\
&\quad = \sum_{i=1}^{K+1} \left(\phi_i^\top(y)\psi(\boldsymbol{x},\boldsymbol{x})\phi_i(y) - 2\phi_i^\top(y)\psi(\boldsymbol{x},\boldsymbol{x}')\phi_i(y') + \phi_i^\top(y')\psi(\boldsymbol{x}',\boldsymbol{x}')\phi_i(y') \right),
\end{aligned}$$

which goes to zero if $[\boldsymbol{x}'] \to [\boldsymbol{x}]$ and $[y'] \to [y]$ by continuity of ψ. □

Lemma 2. *Consider Lemma 1. Suppose that $\mathcal{Z}'_M$ is also topologically closed and closed under permutations, and that ψ also satisfies (i) $\psi(\boldsymbol{x},\boldsymbol{x}) = \sigma^2 > 0$ and (ii) $\psi(\boldsymbol{x},\boldsymbol{y}) \to 0$ as $\|\boldsymbol{x}\| \to \infty$ with $\boldsymbol{y}$ fixed or $\|\boldsymbol{y}\| \to \infty$ with $\boldsymbol{x}$ fixed. Then E^{-1} is continuous.*

Remark 1. Before moving on to the proof of Lemma 2, we remark that Lemma 2 would directly follow if $\mathcal{Z}'_M$ were bounded: then $\mathcal{Z}'_M$ is compact, so E is a continuous, invertible map between compact spaces, which means that E^{-1} must be continuous. The intuition that the result must hold for unbounded $\mathcal{Z}'_M$ is as follows. Since $\phi_1(\,\cdot\,) = 1$, for every $f \in \mathcal{H}_M$, f_1 is a summation of M "bumps" (imagine the EQ kernel) of the form $\psi(\,\cdot\,,\boldsymbol{x}_i)$ placed throughout $\mathcal{X}$. If these bumps go off to infinity, then the function cannot uniformly converge pointwise, which means that the function cannot converge in $\mathcal{H}$ if ψ is sufficiently nice. Therefore, if the function does converge in $\mathcal{H}$, $(\boldsymbol{x}_i)_{i=1}^M$ must be bounded, which brings us to the compact case. What makes this work is the *density channel* $\phi_1(\,\cdot\,) = 1$, which forces $(\boldsymbol{x}_i)_{i=1}^M$ to be well behaved. The above argument is formalized in the proof of Lemma 2.

Proof. Define

$$\mathcal{Z}_J = ([-J,J]^N \times \mathcal{Y})^M \cap \mathcal{Z}'_M,$$

which is compact as a closed subset of the compact set $([-J,J]^N \times \mathcal{Y})^M$. We aim to show that E^{-1} is continuous. To this end, consider a convergent sequence

$$f^{(n)} = \sum_{i=1}^{M} \phi(y_i^{(n)})\psi(\,\cdot\,,\boldsymbol{x}_i^{(n)}) \to f \in \mathcal{H}.$$

Denote $\boldsymbol{x}^{(n)} = (\boldsymbol{x}_1^{(n)},\ldots,\boldsymbol{x}_M^{(n)})$ and $y^{(n)} = (y_1^{(n)},\ldots,y_M^{(n)})$. Claim: $(\boldsymbol{x}^{(n)})_{n=1}^\infty$ is a bounded sequence, so $(\boldsymbol{x}^{(n)})_{n=1}^\infty \subseteq [-J,J]^{NM}$ for J large enough, which means that $(\boldsymbol{x}^{(n)},y^{(n)})_{n=1}^\infty \subseteq \mathcal{Z}_J$ where $\mathcal{Z}_J$ is compact.

Assume the claim, and note that $(f^{(n)})_{n=1}^\infty$ is in the range of $E|_{[\mathcal{Z}_J]}$. By continuity of $E|_{[\mathcal{Z}_J]}$ and compactness of $[\mathcal{Z}_J]$, the range of $E|_{[\mathcal{Z}_J]}$ is compact, which means that it contains the limit f. Since $E|_{[\mathcal{Z}_J]}$ is also invertible, its inverse $(E|_{[\mathcal{Z}_J]})^{-1}$ is also continuous. Therefore,

$$E^{-1}(f^{(n)}) = (E|_{[\mathcal{Z}_J]})^{-1}(f^{(n)}) \to (E|_{[\mathcal{Z}_J]})^{-1}(f) = E^{-1}(f),$$

so E^{-1} is continuous.

It remains to show the claim. Consider

$$\|f_1^{(n)} - f_1^{(m)}\|_{\mathcal{H}}^2 = \mathbf{1}^\top (\underbrace{\psi(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}) + \psi(\boldsymbol{x}^{(m)}, \boldsymbol{x}^{(m)}) - 2\psi(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})}_{\boldsymbol{K}^{(n,m)}})\mathbf{1},$$

where $\mathbf{1} \in \mathbb{R}^M$ is the vector of all ones. Since $(f_1^{(n)})_{n=1}^\infty$ is convergent, it is Cauchy, so $\|f_1^{(n)} - f_1^{(m)}\|_{\mathcal{H}}^2 \to 0$ as $n, m \to \infty$. For any $\boldsymbol{x}$ and $\boldsymbol{y}$, by Cauchy–Schwarz[6] and the AM–GM Inequality, we have

$$2\psi(\boldsymbol{x}, \boldsymbol{y}) \leq 2\sqrt{\psi(\boldsymbol{x}, \boldsymbol{x})\psi(\boldsymbol{y}, \boldsymbol{y})} \leq \psi(\boldsymbol{x}, \boldsymbol{x}) + \psi(\boldsymbol{y}, \boldsymbol{y}).$$

Therefore, $\boldsymbol{K}_{ij}^{(n,m)} \geq 0$, which means that $\|f_1^{(n)} - f_1^{(m)}\|^2 \to 0$ implies that all $\boldsymbol{K}_{ij}^{(n,m)} \to 0$.

Suppose that $(\boldsymbol{x}^{(n)})_{n=1}^\infty$ is unbounded. Then $(\boldsymbol{x}_i^{(n)})_{n=1}^\infty$ is unbounded for some $i \in [M]$. Since, by assumption, $\psi(\boldsymbol{x}, \boldsymbol{x}) = \sigma^2 > 0$ and $\psi(\boldsymbol{x}, \boldsymbol{y}) \to 0$ as $\|\boldsymbol{x}\| \to \infty$ with $\boldsymbol{y}$ fixed or $\|\boldsymbol{y}\| \to \infty$ with $\boldsymbol{x}$ fixed, then

$$\boldsymbol{K}_{ii}^{(n,m)} = \psi(\boldsymbol{x}_i^{(n)}, \boldsymbol{x}_i^{(n)}) + \psi(\boldsymbol{x}_i^{(m)}, \boldsymbol{x}_i^{(m)}) - 2\psi(\boldsymbol{x}_i^{(n)}, \boldsymbol{x}_i^{(m)}) = 2\sigma^2 - 2\psi(\boldsymbol{x}_i^{(n)}, \boldsymbol{x}_i^{(m)}) \to 2\sigma^2 > 0$$

as $n \to \infty$. Since this holds for every m, it cannot be that $\boldsymbol{K}_{ii}^{(n,m)} \to 0$ as $m, n \to \infty$, which is a contradiction. □

Remark 2. To define $\mathcal{Z}_2'$ with multiplicity one, one might be tempted to define

$$\mathcal{Z}_2' = \{((\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2)) \in \mathcal{Z}_2 : \boldsymbol{x}_1 \neq \boldsymbol{x}_2\},$$

which indeed has multiplicity one. Unfortunately, $\mathcal{Z}_2'$ is not closed: if $[0, 1] \subseteq \mathcal{X}$ and $[0, 2] \subseteq \mathcal{Y}$, then $((0, 1), (1/n, 2))_{n=1}^\infty \subseteq \mathcal{Z}_2'$, but $((0, 1), (1/n, 2)) \to ((0, 1), (0, 2)) \notin \mathcal{Z}_2'$, because 0 then has two observations 1 and 2. To get around this issue, one can require an arbitrarily small, but non-zero spacing $\epsilon > 0$ between input locations:

$$\mathcal{Z}_{2,\epsilon}' = \{((\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2)) \in \mathcal{Z}_2 : \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq \epsilon\}.$$

This construction can be generalized to higher numbers of observations and multiplicities as follows:

$$\mathcal{Z}_{M,K,\epsilon}' = \{(\boldsymbol{x}_{\pi(i)}, y_{\pi(i)})_{i=1}^M \in \mathcal{Z}_M : \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \epsilon \text{ for } i, j \in [K],\, \pi \in \mathbb{S}_M\}.$$

Lemma 3. *For every $m \in [M]$, consider a collection $\mathcal{Z}_m' \subseteq \mathcal{Z}_m$ that (i) has multiplicity K, (ii) is topologically closed, and (iii) is closed under permutations. Set*

$$\phi : \mathcal{Y} \to \mathbb{R}^{K+1}, \quad \phi(y) = (y^0, y^1, \cdots, y^K)$$

and let ψ be an interpolating, continuous positive-definite kernel that satisfies (i) $\psi(\boldsymbol{x}, \boldsymbol{x}) = \sigma^2 > 0$ and (ii) $\psi(\boldsymbol{x}, \boldsymbol{y}) \to 0$ as $\|\boldsymbol{x}\| \to \infty$ with $\boldsymbol{y}$ fixed or $\|\boldsymbol{y}\| \to \infty$ with $\boldsymbol{x}$ fixed. Define

$$\mathcal{H}_m = \left\{ \sum_{i=1}^m \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i) : (\boldsymbol{x}_i, y_i)_{i=1}^m \subseteq \mathcal{Z}_m' \right\} \subseteq \mathcal{H}^{K+1}, \tag{4}$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$-dimensional-vector–valued RKHS constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel. Denote

$$[\mathcal{Z}_{\leq M}'] = \bigcup_{m=1}^M [\mathcal{Z}_m'] \quad \textit{and} \quad \mathcal{H}_{\leq M} = \bigcup_{m=1}^M \mathcal{H}_m,$$

where $(\mathcal{H}_m)_{m=1}^M$ are closed and pairwise disjoint. Then the embedding E

$$E\colon [\mathcal{Z}_{\leq M}'] \to \mathcal{H}_{\leq M}, \quad E([(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)]) = \sum_{i=1}^m \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i)$$

is injective, hence invertible. Denote this inverse by E^{-1}. Then E and E^{-1} are continuous in the sense that the restrictions $E|_{[\mathcal{Z}_m']}$ and $E^{-1}|_{\mathcal{H}_m}$ are continuous for all $m \in [M]$.

[6] By the reproducing property of ψ,

$$\psi(\boldsymbol{x}, \boldsymbol{y}) = \langle \psi(\boldsymbol{x}, \cdot), \psi(\boldsymbol{y}, \cdot) \rangle_{\mathcal{H}} \leq \|\psi(\boldsymbol{x}, \cdot)\|_{\mathcal{H}} \|\psi(\boldsymbol{y}, \cdot)\|_{\mathcal{H}} = \sqrt{\psi(\boldsymbol{x}, \boldsymbol{x})\psi(\boldsymbol{y}, \boldsymbol{y})}.$$

Proof. From repeated application of Lemmas 1 and 2, we find that every restriction

$$E|_{[\mathcal{Z}'_m]} \colon [\mathcal{Z}'_m] \to \mathcal{H}_m$$

is invertible, continuous, and has a continuous inverse. The result then follows by showing that $(\mathcal{H}_m)_{m=1}^M$ are closed and pairwise disjoint. First, $[\mathcal{Z}'_m]$ is closed and $(E|_{[\mathcal{Z}'_m]})^{-1}$ is continuous, so

$$\mathcal{H}_m = (E|_{[\mathcal{Z}'_m]})^{-1}([\mathcal{Z}'_m])$$

is closed. Second, suppose that

$$\sum_{i=1}^{m} \phi(y_i)\psi(\,\cdot\,, \boldsymbol{x}_i) = \sum_{i=1}^{m'} \phi(y'_i)\psi(\,\cdot\,, \boldsymbol{x}'_i)$$

for $m \neq m'$. Then, by arguments like in the proof of Lemma 1,

$$\sum_{i=1}^{m} \phi(y_i) = \sum_{i=1}^{m'} \phi(y'_i).$$

Since $\phi_1(\,\cdot\,) = 1$, this gives $m = m'$, which is a contradiction. □

Lemma 4. *Let $\Phi \colon [\mathcal{Z}'_{\leq M}] \to C_b(\mathcal{X}, \mathcal{Y})$ be continuous in the sense that every restriction $\Phi|_{[\mathcal{Z}'_m]}$ is continuous, and let E be from Lemma 3. Then*

$$\Phi \circ E^{-1} \colon \mathcal{H}_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$$

is continuous.

Proof. By continuity of Φ and E^{-1}, every restriction $\Phi \circ E^{-1}|_{\mathcal{H}_m}$ is continuous. We show that $\Phi \circ E^{-1}$ is continuous by showing that every convergent sequence in $\mathcal{H}_{\leq M}$ is eventually contained in some $\mathcal{H}_m$ and appealing to continuity of $\Phi \circ E^{-1}|_{\mathcal{H}_m}$. For suppose that $(f^{(n)})_{n=1}^{\infty} \subseteq \mathcal{H}_{\leq M}$ converging to some $f \in \mathcal{H}_{\leq M}$ visits $\mathcal{H}_m$ and $\mathcal{H}_{m'}$ with $m \neq m'$ infinitely often. Then we can extract subsequences $(f^{(n_k)})_{k=1}^{\infty} \subseteq \mathcal{H}_m$ and $(f^{(n_{k'})})_{k'=1}^{\infty} \subseteq \mathcal{H}_{m'}$, which must both be convergent to f. Since $\mathcal{H}_m$ and $\mathcal{H}_{m'}$ are closed and disjoint, this is a contradiction. □

From here on we let ψ be a stationary kernel, which means that it only depends on the difference of its arguments and can be seen as a function $\mathcal{X} \to \mathbb{R}$.

A.3 Proof of Theorem 1

With the above results in place, we are finally ready to prove our central result, Theorem 1.

Theorem 1. *Consider an appropriate[7] collection $\mathcal{Z}'_{\leq M} \subseteq \mathcal{Z}_{\leq M}$ with multiplicity K. Then a function $\Phi \colon \mathcal{Z}'_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$ is continuous[8], permutation invariant (Property 1), and translation equivariant (Property 2) if and only if it has a representation of the form*

$$\Phi(Z) = \rho\left(E(Z)\right), \quad E((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)) = \textstyle\sum_{i=1}^{m} \phi(y_i)\psi(\,\cdot\, - \boldsymbol{x}_i)$$

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

Proof of sufficiency. To begin with, note that permutation invariance (Property 1) and translation equivariance (Property 2) for Φ are well defined, because $\mathcal{Z}'_{\leq M}$ is closed under permutations and translations by assumption. First, Φ is permutation invariant, because addition is commutative and

[7] For every $m \in [M]$, $\mathcal{Z}'_{\leq M} \cap \mathcal{Z}_m$ must be closed and closed under permutations and translations.

[8] For every $m \in [M]$, the restriction $\Phi|_{\mathcal{Z}'_{\leq M} \cap \mathcal{Z}_m}$ is continuous.

associative. Second, that Φ is translation equivariant (Property 2) follows from a direct verification and that ρ is also translation equivariant:

$$\begin{aligned}
\Phi(T_{\boldsymbol{\tau}} Z) &= \rho\left(\sum_{i=1}^{M} \phi(y_i)\psi(\cdot - (\boldsymbol{x}_i + \boldsymbol{\tau}))\right) \\
&= \rho\left(\sum_{i=1}^{M} \phi(y_i)\psi((\cdot - \boldsymbol{\tau}) - \boldsymbol{x}_i)\right) \\
&= \rho\left(\sum_{i=1}^{M} \phi(y_i)\psi(\cdot - \boldsymbol{x}_i)\right)(\cdot - \boldsymbol{\tau}) \\
&= \Phi(Z)(\cdot - \boldsymbol{\tau}) \\
&= T'_{\boldsymbol{\tau}} \Phi(Z).
\end{aligned}$$

□

Proof of necessity. Our proof follows the strategy used by Zaheer et al. (2017); Wagstaff et al. (2019). To begin with, since Φ is permutation invariant (Property 1), we may define $\Phi\colon [\mathcal{Z}'_{\leq M}] \to C_b(\mathcal{X}, \mathcal{Y})$ by $\Phi([Z]) = \Phi(Z)$, which we verify is continuous in the sense that every restriction $\Phi|_{[\mathcal{Z}'_m]}$ is continuous. Furthermore, by invertibility of E from Lemma 3, $[Z] = E^{-1}(E([Z]))$. Therefore,

$$\Phi(Z) = \Phi([Z]) = \Phi(E^{-1}(E([Z]))) = (\Phi \circ E^{-1})\left(\sum_{i=1}^{M} \phi(y_i)\psi(\cdot - \boldsymbol{x}_i)\right).$$

Define $\rho\colon \mathcal{H}_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$ by $\rho = \Phi \circ E^{-1}$. First, ρ is continuous by Lemma 4. Second, E^{-1} is translation equivariant, because ψ is stationary. Also, Φ is translation equivariant (Property 2) by assumption. Thus their composition ρ is also translation equivariant. □

Remark 3. The function $\rho\colon \mathcal{H}_{\leq M} \to C_b(\mathcal{X}, \mathcal{Y})$ may be continuously extended to the entirety of $\mathcal{H}$ using a generalisation of the Tietze Extension Theorem by Dugundji et al. (1951). There are variants of Dugundji's Theorem that also preserve translation equivariance.

B BASELINE NEURAL PROCESS MODELS

In both our 1d and image experiments, our main comparison is to conditional neural process models. In particular, we compare to a vanilla CNP (1d only; Garnelo et al. (2018a)) and a ATTNCNP (Kim et al., 2019). Our architectures largely follow the details given in the relevant publications.

CNP Baseline Our baseline CNP follows the implementation provided by the authors.[9] The encoder is a 3-layer MLP with 128 hidden units in each, and RELU non-linearities. The encoder embeds every context point into a representation, and the representations are then averaged across each context set. Target inputs are then concatenated to the latent representations, and passed to the decoder. The decoder follows the same architecture, outputting mean and standard deviation channels for each input.

Attentive CNP baseline The ATTNCNP we use corresponds to the deterministic path of the model described by Kim et al. (2019) for image experiments. Namely, an encoder first embeds each context points c to a latent representation $(\mathbf{x}^{(c)}, \mathbf{y}^{(c)}) \mapsto \mathbf{r}_{xy}^{(c)} \in \mathbb{R}^{128}$. For the image experiments, this is achieved using a 2 hidden layer MLP of hidden dimensions 128. For the 1d experiments, we use the same encoder as the CNP above. Every context points then goes through 2 stacked self-attention layers. Each self-attention layer is implemented with a 8-headed attention, a skip connection, and two layer normalizations (as described in Parmar et al. (2018) modulo the dropout layer). To predict values at each target points t, we embed $\mathbf{x}^{(t)} \mapsto \mathbf{r}_x^{(t)}$ and $\mathbf{x}^{(c)} \mapsto \mathbf{r}_x^{(c)}$ using the same single hidden layer MLP of dimensions 128. A target representation $\mathbf{r}_{xy}^{(t)}$ is then estimated 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 $\mathbf{q} \coloneqq \mathbf{r}_x^{(t)}$. Given the estimated target representation $\hat{\mathbf{r}}_{xy}^{(t)}$, the conditional predictive posterior is given by a Gaussian PDF with diagonal covariance parametrised by $(\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}_{pre}^{(t)}) = \mathrm{decoder}(\mathbf{r}_{xy}^{(t)})$ where $\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}_4 p(t) \in \mathbb{R}^3$ and $\mathrm{decoder}$ is a 4 hidden layer MLP with each of 64 dimensions for the images, and the same decoder as the CNP for the 1d experiments.

Following Le et al. (2018) we use minimum standard deviation $\boldsymbol{\sigma}_{min}^{(t)} = [0.1; 0.1; 0.1]$ to avoid infinite log-likelihood, by using the following post-processed standard deviation :

$$\boldsymbol{\sigma}_{post}^{(t)} = 0.1\boldsymbol{\sigma}_{min}^{(t)} + (1 - 0.1)\log\left(1 + \exp(\boldsymbol{\sigma}_{pre}^{(t)})\right)$$

[9] https://github.com/deepmind/neural-processes

C 1-DIMENSIONAL EXPERIMENTS

In this section, we give details regarding our experiments for the 1d data. We begin with detailing model architectures, and then provide details for data generating processes and training procedures. The density at which we evaluate the grid differs from experiment to experiment, and so the values are given in the relevant subsections. In all experiments, the weights are optimized using Adam Kingma & Ba (2015) and weight decay of $1e-5$ is applied to all model parameters. The learning rates are specified in the following subsections.

C.1 CNN ARCHITECTURES

In this appendix, we describe the architectures throughout the 1d experiments (Sections 5.1 to 5.3). Throughout the experiments, we consider two architectures: CONVCNP (which utilizes a smaller architecture), and CONVCNPXL (with a larger architecture). For all architectures, E used a EQ kernel with a learnable length-scale parameter, as detailed in Section 4, as did the final output layer for ρ. The architectures for the 1d experiments are described below.

CONVCNP For the 1d experiments, we use a simple, 4-layer convolutional architecture, with RELU nonlinearities. The kernel size of the convolutional layers was chosen to be 5, and all employed a stride of length 1 and zero-padding of 2 units. The number of channels per layer was set to $[16, 32, 16, 2]$, where the final channels where then processed by the final, EQ-based layer of ρ as mean and standard deviation channels. We employ a SOFTPLUS nonlinearity on the standard deviation channel to enforce positivity. This model has 6,537 parameters.

CONVCNPXL Our large architecture takes inspiration from UNets (Ronneberger et al., 2015). We employ a 12-layer architecture with skip connections. The first 6 layers multiply the number of channels by two every layer, and the final 6 layers reduce the number of channels by a factor of two each. We use concatenation for the skip connections. The following describe which layers are concatenated such that $L_i \leftarrow [L_j, L_k]$ implies that the input to layer i is the concatenation of the activations of layers j and k:

- $L_8 \leftarrow [L_5, L_7]$
- $L_9 \leftarrow [L_4, L_8]$
- $L_{10} \leftarrow [L_3, L_9]$
- $L_{11} \leftarrow [L_2, L_{10}]$
- $L_{12} \leftarrow [L_1, L_{11}]$

As for the smaller architecture, we use RELU nonlinearities, kernels of size 5, stride 1, and zero-padding for two units on all layers.

C.2 SYNTHETIC 1D EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

The specific kernels used for the Gaussian Processes which generate the data in this experiment are defined as follows:

- EQ: $k(x, x') = e^{-\frac{1}{2}(x-x')^2}$;
- weakly periodic: $k(x, x') = e^{-\frac{1}{2}(f_1(x)-f_1(x'))^2-\frac{1}{2}(f_2(x)-f_2(x'))^2} \cdot e^{-\frac{1}{8}(x-x')^2}$ with $f_1(x) = \cos(8\pi x)$ and $f_2(x) = \sin(8\pi x)$; and
- Matern–$\frac{5}{2}$: $k(x, x') = (1 + 4\sqrt{5}d + \frac{5}{3}d^2)e^{-\sqrt{5}d}$ with $d = 4|x - x'|$.

During the training procedure, the number of context points and target points for a training batch are each selected randomly from a uniform distribution over the integers between 3 and 50. This number of context and target points are randomly sampled via a function sampled from the process (a Gaussian process with one of the above kernels or the sawtooth process). The points are sampled uniformly at x-values on the interval $-2 \leq x \leq 2$. All models in this experiment were trained on 200 epochs using 256 batches per epoch of batch size 16. We discretize $E(Z)$ by evaluating 64 points per

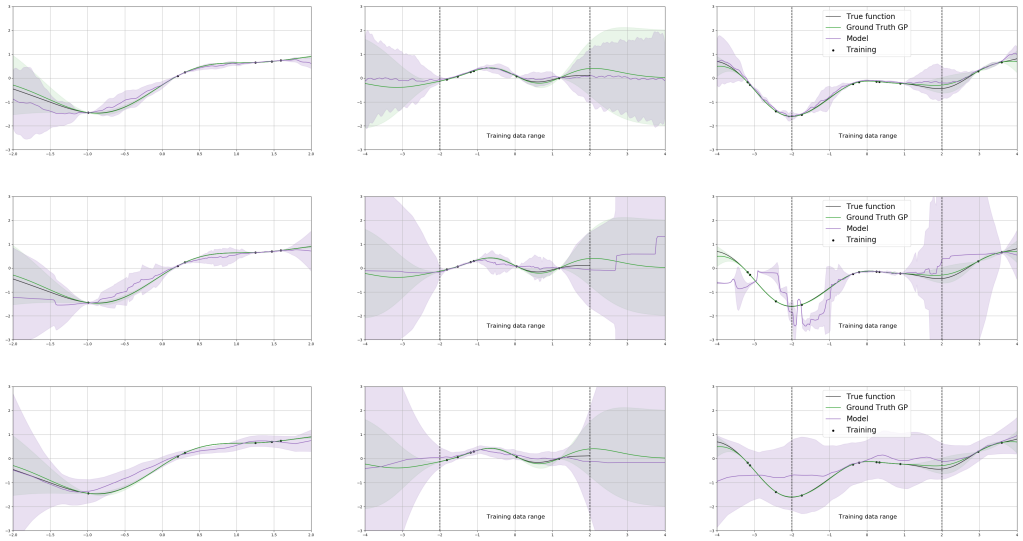


Figure 6: Example functions learned by the (top) CCP, (center) ANP, and (bottom) CNP when trained on an EQ kernel (with lengthscale parameter 1). Left column shows the predictive posterior of the models when data is presented in same range as training. Center column shows model predicting outside training data range when no data is observed there, and right-most column shows model predictive posteriors when presented with data outside the training data range.

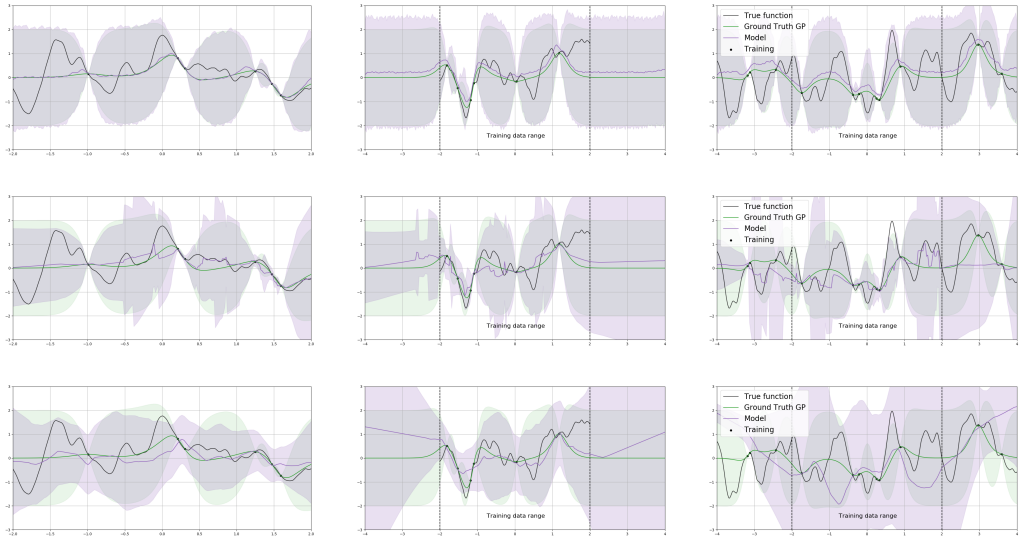


Figure 7: Example functions learned by the (top) CCP, (center) ANP, and (bottom) CNP when trained on a Matern $5/2$ kernel (with length-scale parameter 0.25). Left column shows the predictive posterior of the models when data is presented in same range as training. Center column shows model predicting outside training data range when no data is observed there, and right-most column shows model predictive posteriors when presented with data outside the training data range.

unit in this setting. We use a learning rate of $1e-3$. For the CNP and ATTNCNP, we use a latent dimension of 128, and learning rate of $1e-4$.

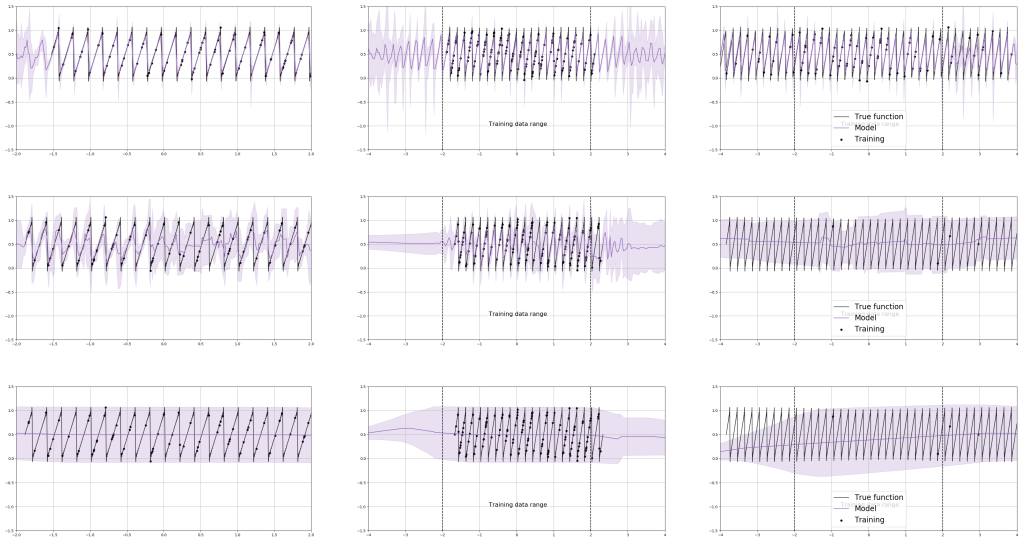


Figure 8: Example functions learned by the (top) CCP, (center) ANP, and (bottom) CNP when trained on a random “saw-tooth function". Left column shows the predictive posterior of the models when data is presented in same range as training. Center column shows model predicting outside training data range when no data is observed there, and right-most column shows model predictive posteriors when presented with data outside the training data range.

The random sawtooth process is generated from the following function:

$$y_{\text{sawtooth}}(t) = \frac{A}{2} - \frac{A}{\pi}\sum_{k=1}^{\infty}(-1)^k \frac{\sin(2\pi k f t)}{k}, \tag{5}$$

where A is the amplitude, f is the frequency, and t is “time”. Throughout training, we fix the amplitude to be one. We truncate the series at an integer K. At every iteration, we sample a frequency uniformly between $(3, 5)$, and K between $(10, 20)$, and a random shift $(-5, 5)$. As the task is much harder, we sample context and target sets with $(3, 100)$. Here the CNP and ATTNCNP employ learning rates of $1e-3$. All other hyper-parameters remain unchanged.

C.3 PLASTICC EXPERIMENTAL DETAILS

CONVCNP was trained on 200 epochs using 1024 batches per epoch of batch size 4. For training and testing, the number of context points for a batch are each selected randomly from a uniform distribution over the integers between 1 and the number of points available in the series (usually between 10-30 per bandwidth). The remaining points in the series are used as the target set. For testing, a batch size of one was used and statistics were computed over 1000 evaluations. We compare CONVCNP to the GP models used in (Boone, 2019) using the implementation in https://github.com/kboone/avocado. The data used for training and testing is normalized according to $t(v) = \frac{v-m}{s}$ with values:

Variable	m	s
time	5.94 $\times 10^4$	8.74 $\times 10^2$
lsstu	1.26	1.63 $\times 10^2$
lsstg	-0.13	3.84 $\times 10^2$
lsstr	3.73	3.41 $\times 10^2$
lssti	5.53	2.85 $\times 10^2$
lsstz	6.43	2.69 $\times 10^2$
lssty	6.27	2.93 $\times 10^2$

These values are approximately estimated from a batch sampled from the training data. To remove outliers in the GP results, log-likelihood values less than -10 are removed from the evaluation. These same datapoints were removed from the CONVCNP results as well.

For this dataset, we only used the CONVCNPXL, as we found the CONVCNP to underfit. Learning rate was set to $1e-3$, and we discretize $E(Z)$ by evaluating 256 points per unit.

C.4 PREDATOR-PREY EXPERIMENTAL DETAILS

The Lotka–Volterra model (Wilkinson, 2011) which is used to describe the evolution of predator-prey populations. Simulated training data for the experiment in Section 5.3 was generated from the Lotka-Volterra model in the following way.

The following description is borrowed from (Wilkinson, 2011). Let X be the number of predators and Y the number of prey at any point in our simulation and according to the model one of the following four events can occur:

1. A single predator is born according to rate $\theta_1 XY$ increasing X by one.
2. A single predator dies according to rate $\theta_2 X$ decreasing X by one.
3. A single prey is born according to rate $\theta_3 Y$ increasing Y by one.
4. A single prey dies (is eaten) according to rate $\theta_4 XY$ decreasing Y by one.

The parameter values $\theta_1, \theta_2, \theta_3$ and θ_4, as well as the initial values of X and Y govern the behavior of the simulation. The constants $\theta_1 = 0.01, \theta_2 = 0.5, \theta_3 = 1$ and $\theta_4 = 0.01$ used are used in (Papamakarios & Murray, 2016) which generate reasonable time series. We also use these values in our experiments. Note that, may not the parameter values that would be estimated from the Hudson's Bay lynx-hare dataset (Leigh, 1968), but obtaining oscillating time series from the Lotka-simulation is sensitive to the choice of parameters and many parametrizations result in populations that simply die out.

Time series are simulated using Gillespie's algorithm (Gillespie, 1977):

1. Draw the time to the next event from an exponential distribution with rate equal to the total rate $\theta_1 XY + \theta_2 X + \theta_3 Y + \theta_4 XY$.
2. Select one of the above events (i, ii, iii, iv) at random, with probability proportional to its rate.
3. Adjust the appropriate population according to the selected event.

The simulations would reach an approximate maximum population of 300 while the context set in the lynx-hare dataset has an approximate maximum population of about 80 so we scaled our simulation population by a factor of $4/14$. We also remove time series longer than 100 units of time, with more than 10000 events or where one of the populations is entirely zero. The number of context n points for a training batch are each selected randomly from a uniform distribution between 3 and 80 and the number of target points is $150 - n$. These target and context points are then sampled from the simulated series. The Hudson's Bay lynx-hare dataset has time values that range from 1845 to 1935 however, the values supplied to the model range from 0 to 90 to remain consistent with the simulated data.

For evaluation, and interval of 18 points is removed from the the Hudson's Bay lynx-hare dataset to act as a target set, while the remaining 72 points for the context set. This construction highlights the model's interpolation as well as its uncertainty in the presence of missing data.

Models in this setting were trained for 200 epochs, with 256 batches per epoch, with each batch containing 50 tasks. For this dataset, we only used the CONVCNP, as we found the CONVCNPXL to overfit. Learning rate was set to $1e-3$, and we discretize $E(Z)$ by evaluating 100 points per unit.

We attempted to train an ATTNCNP for comparison, but due to the nature of the synthetic data generation, many of the training series end before 90 time units, the length of the Hudson's Bay lynx-hare series. Effectively, this means that the ATTNCNP was asked to predict outside of it's training interval, a task that it struggles with as shown in Section 5.1. The plots in Figure 9 show that

the ATTNCNP is able to learn the first part of the time series but are unable to model data outside of the first 20 or so time units. Perhaps with more capacity and training epochs the ATTNCNP training would be more successful. Note from Figure 3 that our model does better on the synthetic data than on the real. This could be due to the parameters of the Lotka–Volterra model used being a poor estimate for the real data.

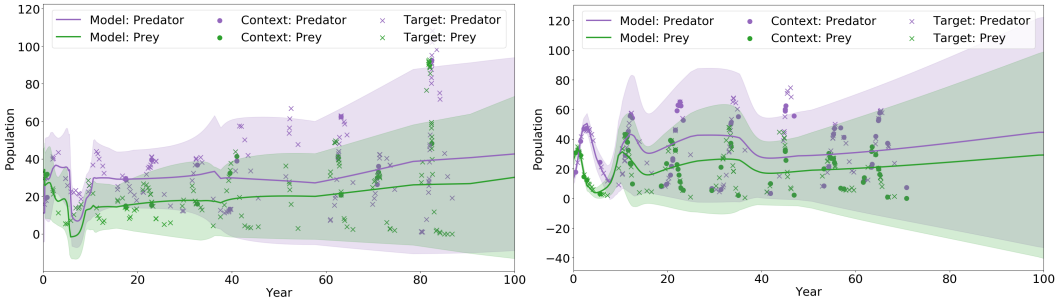


Figure 9: ATTNCNP performance on two samples from the Lotka–Volterra process (sim).

D IMAGE EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

D.1 EXPERIMENTAL DETAILS

Training Details In all our experiments we sample $n_{cntxt} \sim \mathcal{U}(\frac{n_{total}}{100}, \frac{n_{total}}{2})$ context points and $n_{trgt} = n_{total}$ target points from each of the 16 images per batch. The weights are optimised using Adam (Kingma & Ba, 2015) with $5e-4$ learning rate. We use a maximum of 100 epochs, with early stopping of 15 epochs patience. All pixel values are divided by 255 to rescale them in $[0, 1]$ range. In the following discussion, we assume that images are in RGB but very similar models can be used for greyscale images or other grided inputs (e.g. 1D time series sampled at uniform intervals).

Proposed Convolutional CNP Unlike ATTNCNP and off-the-grid CONVCNP, on-the-grid CONVCNP takes advantage of the grided structure. Namely, target and context points can be specified by the whole image, a context mask M_C, and a target mask M_T instead of sets of feature-value pairs. Although this is an equivalent formulation, it makes it more natural and simpler to implement in standard deep learning libraries. In the following, we dissect the architecture and algorithmic steps succinctly summarized in Section 4. Note that all the convolutional layers are actually depthwise separabe (Chollet, 2017), this enables large kernel size (i.e. receptive fields) while being parameter and computationally efficient.

1. Select all context points and append a density channel to them, which intuitively says that "there is 1 point at this position": $\mathrm{M}_c \odot [1; \mathrm{I}]$. Each pixel value will now have 4 channels. Note that the mask will automatically set the density channel to 0, effectively setting the pixel value to 0, while saying "there are 0 point (a missing value) at this position"

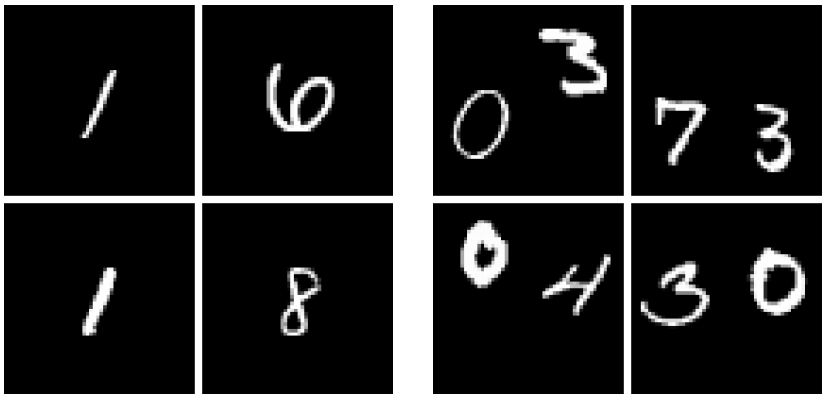
2. Take a normalized convolution of the previously masked signal with a learned kernel: $\frac{\text{CONV}_{\boldsymbol{\theta}}(\mathrm{M}_c \odot [1;\mathrm{I}]^T)}{\text{CONV}_{\boldsymbol{\theta}}(\mathrm{M}_c)}$. The output channel size is 128 dimensional. The kernel size of $\text{CONV}_{\boldsymbol{\theta}}$ depends on the image shape and model used (Table 4). Finally, we enforce element-wise positivity of the trainable filter by only considering its absolute values. As discussed in Appendix D.4 the normalization and positivity constraints are not empirically better for on-the-grid data.

3. Apply a CNN. We stack residual blocks (He et al., 2016) each consisting of 2 convolutional layer, ReLU activations, and no batch normalization. The output channels of each layers is 128. The kernel size is the same across the whole network, but depends on the image shape and model used (Table 4).

4. To get a representation for every target points $\mathbf{r}_{xy}^{(t)} \in \mathbb{R}^{128}$, we mask-select the previous outputs $\mathbf{r}_{xy}^{(t)} = \text{CNN}(\cdot)[\mathrm{M}_t]$ at target positions. All subsequent steps are the same as for ATTNCNP (4 hidden layer MLP decoder and preprocessing of the predicted standard deviation). [10]

Table 4: CNN architecture for the image experiments.

Model	Input Shape	$\text{CONV}_{\boldsymbol{\theta}}$ Kernel Size	CNN Kernel Size	CNN Num. Res. Blocks
ConvCP	< 50 pixels	9	5	4
	> 50 pixels	7	3	4
ConvCP XL	any	11	11	6



(a) Train (b) Test

Figure 10: Samples from our generated Zero Shot Multi MNIST (ZSMM) dataset.

D.2 ZERO SHOT MULTI MNIST (ZSMM) DATA

In the real world, it is very common to have multiple objects in our field of view which do not interact together. Yet, many image data sets in machine learning contain a single well centered object. To evaluate the translation equivariance and generalization capabilities of our model, we use a zero shot multi MNIST setting.

The training set contains all 60000 training 28×28 digits centered in a black 56×56 background. (Figure 10a). For the test set, we randomly sample with replacement 10000 pairs of digits from the MNIST test set, put them in a black 56×56 background, and translate the digits in such a way that the digits can be arbitrarily close but never overlap (Figure 10b). Importantly, the scale of the digits and the image size are the same during training and testing.

D.3 ATTNCNP AND CONVCNP QUALITATIVE COMPARISON

Figure 11 shows the test log-likelihood distributions of an ATTNCNP and CONVCNP model as well as some qualitative comparisons between the two.

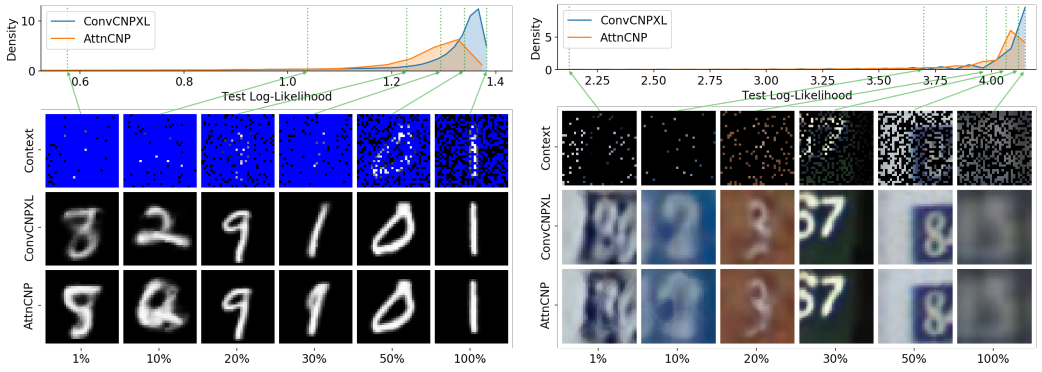
Although most samples of both models look relatively similar for SVHN and Celeba32, the real advantage of CONVCNP becomes apparent when testing the generalization capacity of both models. Figure 12 shows CONVCNP and ATTNCNP trained on CelebA32 and tested on a downscaled version of Ellen's famous Oscar selfie. We see that CONVCNP generalizes better in such setting. [11]

D.4 ABLATION STUDY - FIRST LAYER

To understand the importance of the different components of the first layer, we performed an ablation study by: removing the density normalization (CONVCNP no norm.), removing the density channel (CONVCNP no dens.), removing the positivity constraints (CONVCNP no abs.), removing the positivity constraints and the normalization (CONVCNP no abs. norm.), replacing the fully trainable first layer by an EQ kernel similarly to the continuous case (CONVCNP EQ).

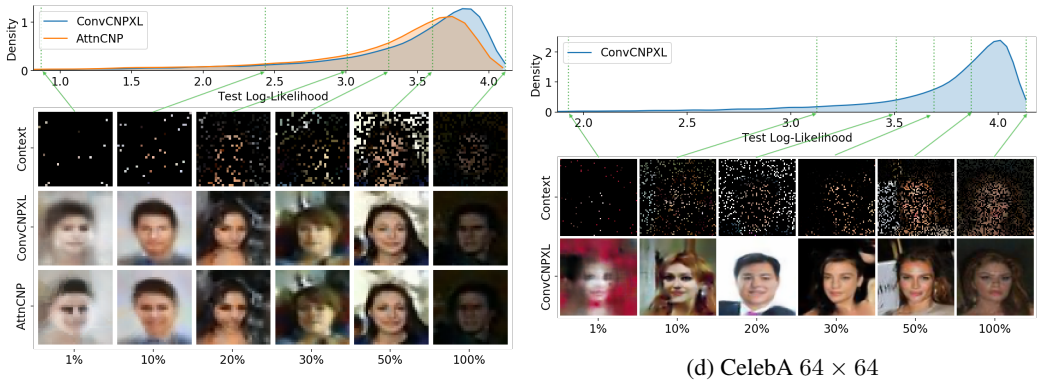
[10] Here we describe the last step as (i) Mask selecting target representations. (ii) Parametrizing the predictive posterior by using the previous representations and an additional decoder. This shows how similar the compared ATTNCNP and CONVCNP models are architecturally. An other equivalent view, which was taken in the main text for clarity is that (i) The CNN encodes a functional representation of the predictive posterior by outputting a parametrisation of the predictive posterior at each position. (ii) Mask selecting the targets directly gives the target predictive posterior. These 2 views are equivalent, because the decoder in the first view is a MLP which can be implemented by applying stacked point wise (kernel size 1) convolutional layers and non linear activations.

[11] The reconstruction looks worst than Figure 4b despite the larger context set, because the test image has been downscaled and the models are trained on a low resolution Celeba32. These constraints come from ATTNCNP's large memory footprint.



(a) MNIST

(b) SVHN



(c) CelebA 32×32

(d) CelebA 64×64

Figure 11: Log Likelihood and qualitative comparisons between ATTNCNP and CONVCNP on four standard benchmarks. The top row shows the log-likelihood distribution for both models. The images below correspond to the context points (top), CONVCNP target predictions (middle), and ATTNCNP target predictions (bottom). Each column corresponds to a given percentile of the CONVCNP distribution. ATTNCNP could not be trained on Celeba64 due to its memory inefficiency.

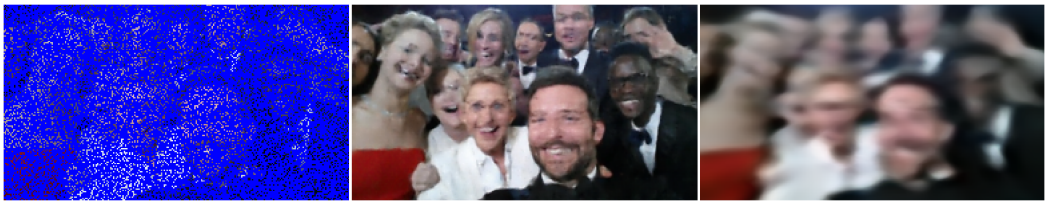


Figure 12: Qualitative evaluation of a CONVCNP (center) and ATTNCNP (right) trained on the CelebA32 and tested on Ellen's Oscar downscaled (146×259) selfie (DeGeneres, 2014) with 20% of the pixels as context (left).

Table 5 shows that (i) Appending a density channel is usefull. (ii) Enforcing the positivity constraint is only important when using a normalized convolution. (iii) Using a less expressive EQ filter does not significantly decrease performance, suggesting that the model might be learning similar filters (Appendix D.5).

D.5 QUALITATIVE ANALYSIS OF THE FIRST FILTER

As discussed in Appendix D.4, using a less expressive EQ filter does not significantly decrease performance. Figure 13 shows that this happens because the fully trainable kernel learns to approximate the EQ filter.

Table 5: Log-likelihood from image ablation experiments (6 runs).

Model	MNIST	SVHN	CelebA32	CelebA64	ZSMM
ConvCNP	**1.19 ±0.01**	**3.89 ±0.01**	**3.19 ±0.02**	**3.64 ±0.01**	1.21 ±0.00
... no density	1.15 ±0.01	3.88 ±0.01	3.15 ±0.02	3.62 ±0.01	1.13 ±0.08
... no norm.	1.19 ±0.01	3.86 ±0.03	3.16 ±0.03	3.62 ±0.01	1.20 ±0.01
... no abs.	1.15 ±0.02	3.83 ±0.02	3.08 ±0.03	3.56 ±0.01	1.15 ±0.01
... no abs. norm.	1.19 ±0.01	3.86 ±0.03	3.16 ±0.03	3.62 ±0.01	1.20 ±0.01
... EQ	1.18 ±0.00	3.89 ±0.01	3.18 ±0.02	3.63 ±0.01	**1.21 ±0.00**

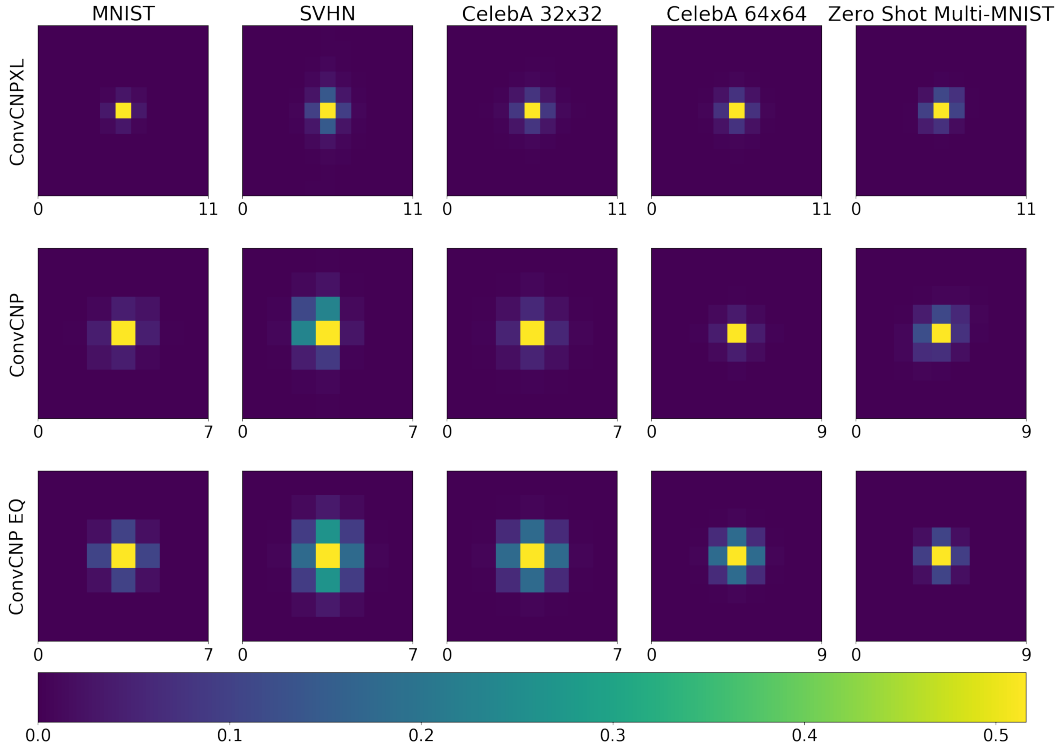


Figure 13: First filter learned by ConvCNPXL, ConvCNP, and ConvCNP EQ for all our datasets. In the case of RGB images, the plotted filters are for the first channel (red). Note that not all filters are of the same size.