GP-ConvCNP: Better Generalization for Convolutional Conditional Neural Processes on Time Series Data

Jens Petersen[1] **Gregor Köhler**[1] **David Zimmerer**[1] **Fabian Isensee**[2] **Paul F. Jäger**[3] **Klaus H. Maier-Hein**[1]

[1]Division of Medical Image Computing, German Cancer Research Center, Heidelberg, Germany
[2]HIP Applied Computer Vision Lab, Division of Medical Image Computing, German Cancer Research Center
[3]Interactive Machine Learning Group, German Cancer Research Center

Abstract

Neural Processes (NPs) are a family of conditional generative models that are able to model a distribution over functions, in a way that allows them to perform predictions at test time conditioned on a number of context points. A recent addition to this family, *Convolutional Conditional Neural Processes* (ConvCNP), have shown remarkable improvement in performance over prior art, but we find that they sometimes struggle to generalize when applied to time series data. In particular, they are not robust to distribution shifts and fail to extrapolate observed patterns into the future. By incorporating a Gaussian Process into the model, we are able to remedy this and at the same time improve performance within distribution. As an added benefit, the Gaussian Process reintroduces the possibility to sample from the model, a key feature of other members in the NP family.

1 INTRODUCTION

Neural Processes [Garnelo et al., 2018a,b] have been proposed as a way to leverage the expressiveness of neural networks to learn a distribution over functions (often referred to as a *stochastic process*), so that they can condition their predictions on observations given at test time, a so-called *context*. But what does it mean to successfully learn such a distribution? We believe that it should be characterized by the following: 1) accurate predictions, meaning predictions should be as close as possible to the true underlying function, 2) good reconstruction of the given observations, 3) generalization, because we assume that there will be some underlying generative process from which the distribution originates and which is valid beyond the finite data we observe. The latter is especially important when only few context observations are given that could be explained by several different functions. Follow-up work to Neural Processes has mostly emphasized the first two aspects, the most prominent of which are *Attentive Neural Processes* (ANP) [Kim et al., 2019] and *Convolutional Conditional Neural Process* (ConvCNP) [Gordon et al., 2020], each improving upon its predecessor in terms of both prediction accuracy and reconstruction ability.

We propose a model that addresses all of the above, with a particular focus on the ability to generalize. By combining ConvCNP with a Gaussian Process, we achieve a significant improvement in generalization: the model, which we call GP-ConvCNP, can better extrapolate far from the provided context observations—meaning into future given past and present observations—and is more robust to a distribution shift at test time. It further reintroduces the ability to sample from the model, something that ConvCNP is incapable of, showing a better sample distribution than both NP and ANP. Finally, we find that our proposed model often yields a significant improvement in predictive performance on in-distribution data as well. We focus our evaluation on time series data, where we see the greatest potential for applications of our model. In this context, we consider several synthetic datasets as well as real time series, specifically weather data and predator-prey population dynamics. We provide a complete implementation[1], including data for convenience, to reproduce all experiments in this work.

2 PROBLEM STATEMENT & METHODS

In the framework of Neural Processes [Garnelo et al., 2018a,b] we assume that we are given a set of N observations $C = \{(x_c, y_c)\}_{c=1}^{N} =: (\mathbf{x_c}, \mathbf{y_c})$, often called the *context*, where $x_c \in X$ are samples from the input space X and $y_c \in Y$ are samples from the output space Y (commonly $X = \mathbb{R}^{d_X}$ and $Y = \mathbb{R}^{d_Y}$, in this work we restrict ourselves to $X = \mathbb{R}$, because time is scalar). It is assumed that these observations were generated by some function

[1]`https://github.com/MIC-DKFZ/gpconvcnp`

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:939–949.

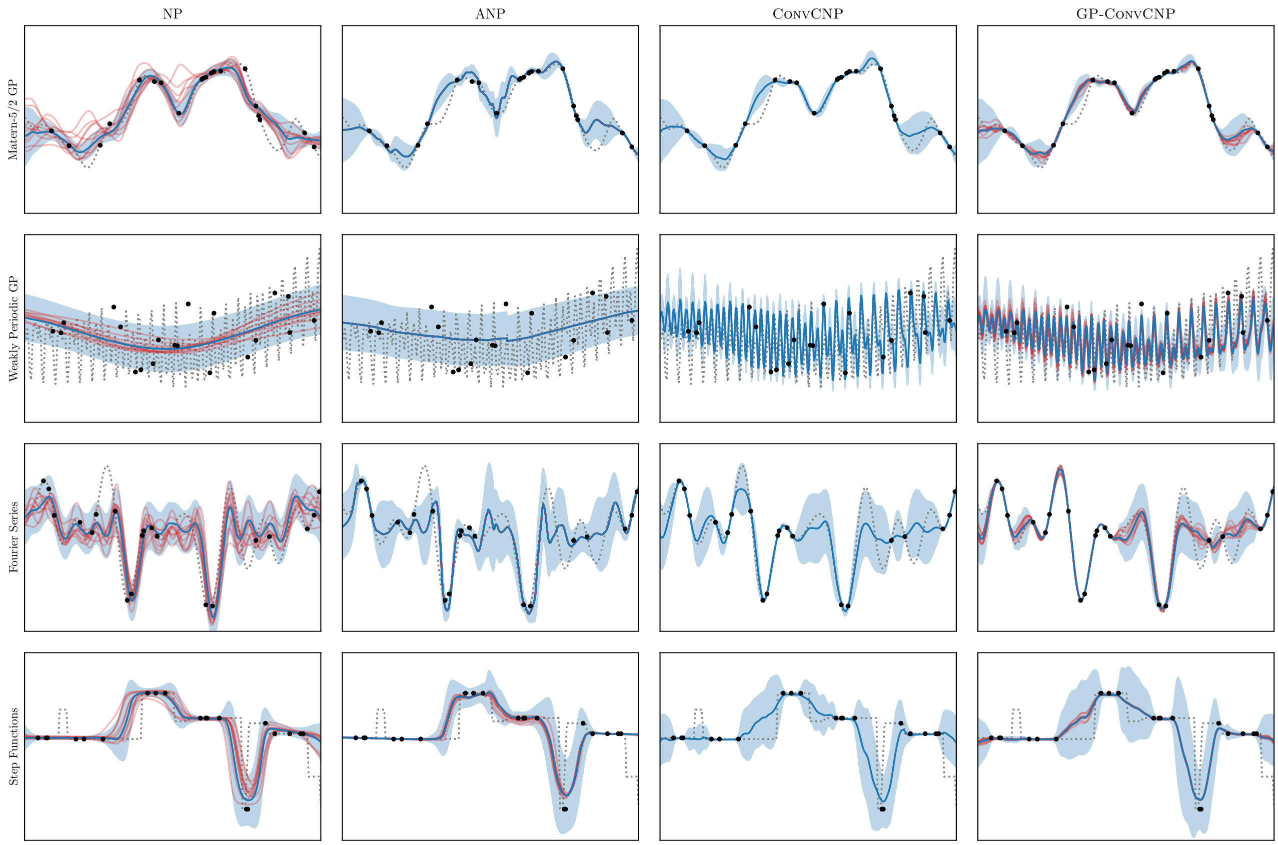

Figure 1: Our work proposes GP-ConvCNP, an extension of ConvCNP that reintroduces sampling and improves generalization on time series data. Shown here are examples for the different synthetic time series and methods evaluated in this work (mean prediction in blue, samples in red). While the mean predictions from ConvCNP and GP-ConvCNP look similar—and significantly better than those from Neural Processes (NP) and Attentive Neural Processes (ANP)—only GP-ConvCNP combines high quality predictions (a feature of ConvCNP) with the ability to sample (a feature of NP and ANP). While synthetic data measures in-distribution performance, we evaluate generalization capabilities on real data.

$f : X \rightarrow Y$, i.e. $y_c = f(x_c)$, and our goal is to infer f from C so that we may evaluate it at arbitrary new input locations x_t. In reality, this will most likely mean we have collected a number of measurements over time and are interested in an f that lets us interpolate and extrapolate those measurements. Note that when we speak of predictive performance, we refer to both of those cases and not in a temporal sense. The problem is ill-posed without placing further assumptions on f, which is why we typically restrict it to some family $\mathcal{F}$: polynomials of some order, a combination of oscillating functions with different frequencies, etc.. However, in many cases it is undesired or even impossible to manually specify $\mathcal{F}$, so Neural Processes propose to use neural networks to learn an approximate representation of $\mathcal{F}$ by observing many examples $f \in \mathcal{F}$. The latter are typically represented as a context set C (the measurements we have) and a *target* set $T = \{(x_t, y_t)\}_{t=1}^{M} =: (\mathbf{x_t}, \mathbf{y_t})$ (the measurements we're interested in). By learning to reconstruct the examples f from a limited number of context points a model should implicitly form a representation of $\mathcal{F}$, which leads to the following learning objective:

$$\max_{\theta} \sum_{f \in \mathcal{F}} \log p_\theta(\mathbf{y_t}|\mathbf{x_t}, \mathbf{x_c}, \mathbf{y_c}) \tag{1}$$

$$= \max_{\theta} \sum_{f \in \mathcal{F}} \sum_{t} \log \mathcal{N}(y_t; g_\theta^\mu(Z, x_t), g_\theta^\sigma(Z, x_t)) \tag{2}$$

This objective is common to all approaches we evaluate in our work, and the second line formalizes the fact that we choose to always model the output as a diagonal Gaussian, parametrized by mean and variance functions $g_\theta^\mu, g_\theta^\sigma$ that seek to maximize the log-likelihood of the targets $\mathbf{y_t}$. The output variance can also be fixed, but Le et al. [2018] show that a learned output variance is preferable. Z is a representation of the context $(\mathbf{x_c}, \mathbf{y_c})$, i.e. there is a mapping $E : X, Y \rightarrow Z$. The implementation of E is where the members of the Neural Process family differ most, and we visualize them in Fig. A.1.

2.1 (ATTENTIVE) NEURAL PROCESSES

The original Neural Processes [Garnelo et al., 2018b] implement E as a neural network that *encodes* individual context observations $(\mathbf{x_c}, \mathbf{y_c})$ into a finite-dimensional space. These representations are then averaged to form the global representation Z. Similar to Eq. (2), Z parametrizes a Gaussian distribution, which enables NP to sample from this latent space and produce diverse predictions; we do not consider the deterministic NP variant [Garnelo et al., 2018a] in this work. NP are trained by maximizing a lower bound on Eq. (2), similar to variational autoencoders. In our NP implementation E and $(g_\theta^\mu, g_\theta^\sigma)$ are symmetric 6-layer MLP, with a representation size of 128. Attentive Neural Processes [Kim et al., 2019] are motivated by the observation that NP poorly reconstruct the provided context, i.e. the predictions seem to miss the context points, as seen for example in Fig. 1. To mitigate this effect, ANP augment NP with an additional deterministic encoder-decoder path. Instead of averaging the individual representations, a learned attention mechanism combines them, conditioned on a target point x_t. So while NP need to compress representations to a single point in Z, ANP don't have this bottleneck, which likely contributes to their improved performance. In our ANP implementation, the deterministic path mirrors the variational path, with both the representation dimension and the embedding dimension of the attention mechanism being 128. Le et al. [2018] evaluated several hyperparameter configurations for NP and ANP and our implementation matches their best performing one.

2.2 FROM CONVCNP TO GP-CONVCNP

With the goal of enabling translation equivariance (i.e. independence of the value range of $\mathbf{x_c}$ and $\mathbf{x_t}$) in Neural Processes, the authors of Convolutional Conditional Neural Processes (ConvCNP) [Gordon et al., 2020] approach their work from the perspective of *learning on sets* [Zaheer et al., 2017]. While NP and ANP map the context set into a finite-dimensional representation, ConvCNP map it into an infinite-dimensional function space. The authors show that in this scenario translation equivariance (as well as permutation invariance) can only be achieved if the mapping E can be represented in the form

$$E(\mathbf{x_c}, \mathbf{y_c}) = \rho(E'(\mathbf{x_c}, \mathbf{y_c})) \tag{3}$$

$$E'(\mathbf{x_c}, \mathbf{y_c}) = \sum_c \phi(y_c)\psi(\cdot - x_c) \tag{4}$$

where $\phi : Y \to \mathbb{R}^2$ and $\psi : X \to \mathbb{R}$, so that E' defines a function and ρ operates in function space and must be translation equivariant. The similar naming of E, E' is deliberate, because herein lies a key difference to NP (and also ANP): NP learn a powerful mapping (i.e. neural network) from the context to a representation and then another one from this representation to the output space, whereas ConvCNP employs a very simple mapping to another representation (to function space, because ϕ and ψ are defined with kernels, see below). A powerful approximator is then learned that operates *within* this representation space, as ρ is a CNN operating on a discretization of E'. The mapping back to output space is again a simple one, usually also ψ combined with a linear map. In this sense, both E and E' can be thought of as representations when we make the connection to NP. See also Fig. A.1 for a visualization of these differences. In Gordon et al. [2020], ψ is chosen to be a simple Gaussian kernel, and ϕ such that the resulting E' has two components:

$$E'(\mathbf{x_c}, \mathbf{y_c}) = \left(\sum_c k(\cdot, x_c) \ , \ \sum_c \frac{y_c k(\cdot, x_c)}{\sum_{c'} k(\cdot, x_{c'})} \right) \tag{5}$$

which is the combination of a kernel density estimator and a Nadaraya-Watson estimator. This estimate is discretized on a suitable grid and a CNN ρ is applied, the result of which is again turned into a continuous function by convolving with the (Gaussian) kernel ψ. We use the official implementation[2] in our experiments. Note that k in Eq. (5) is the same as ψ in the implementation.

In this work, we propose GP-ConvCNP, a model that replaces the deterministic kernel density estimate E' in ConvCNP with a Gaussian Process posterior [Rasmussen and Williams, 2006]. Gaussian Processes (GP) are a popular choice for time series analysis [Roberts et al., 2013], but typically require a lot of prior knowledge about a problem to choose an appropriate kernel. We will find that this is not the case for GP-ConvCNP, which is even able to learn periodicity when the chosen kernel is not periodic.

The posterior in a GP is a normal distribution with a mean function $m(\mathbf{x_t})$ conditioned on the context and a covariance function $K(\mathbf{x_t})$ specified by some kernel k:

$$m(\mathbf{x_t}) = k_{tc}^T \left(k_{cc} + \sigma^2\mathbb{I}\right)^{-1} \mathbf{y_c} \tag{6}$$

$$K(\mathbf{x_t}) = k_{tt} + \sigma^2 - k_{tc}^T \left(k_{cc} + \sigma^2\mathbb{I}\right)^{-1} k_{tc} \tag{7}$$

where $k_{tc} = k(\mathbf{x_t}, \mathbf{x_c})$ etc. and σ^2 is a noise parameter that essentially determines how close the prediction will be to the context points. We make this parameter learnable. Note that Eq. (6) is very similar to Eq. (5): it corresponds to the second component of the Nadaraya-Watson estimator with only a changed denominator.

The first obvious benefit of this model is that we can sample from the GP posterior distribution and thus also from

[2] https://github.com/cambridge-mlg/convcnp

our model, recovering one very compelling property of NP that ConvCNP lacks. Another advantage we see is that by working with a distribution instead of a deterministic estimate as input to the CNN, the data distribution is implicitly smoothed. It has been established that such smoothing reduces overfitting and improves generalization, e.g. by adding noise to inputs [Bishop, 1995, p.347] or more generally doing data augmentation [Volpi et al., 2018]. Working with a distribution instead of a deterministic estimate, we need to perform Monte-Carlo integration to get a prediction from our model. During training, however, we only use a single sample, as is commonly done e.g. in variational autoencoders when training with mini-batch stochastic gradient descent. To facilitate comparison, the kernel we use in our GP is the same as in ConvCNP, i.e. a Gaussian kernel with a learnable length scale.

Note that our model retains all desirable characteristics of the competing approaches, in particular permutation invariance with respect to the inputs (present in all prior art) and translation equivariance (present in ConvCNP)[3]. For details on the various optimization parameters etc. we refer to the provided implementation.

[3] As in ConvCNP, this obviously requires a stationary kernel.

3 EXPERIMENTS

We design our experiments with the purpose of evaluating how well members of the Neural Process family, including the one we propose, are suited for the task of learning distributions over functions, i.e. stochastic processes, specifically for time series data. Like the works we compare ourselves with, we evaluate both predictive performance (*How good is our prediction between context points?*) via the predictive log-likelihood and the reconstruction performance (*How good is our prediction at the context points?*) via the root-mean-square error (RSME), because predictions directly at the context points are usually extremely narrow Gaussians, leading to unstable likelihoods.

As outlined in the introduction, one defining aspect of successfully learning a distribution over functions is a model's ability to generalize. This can mean several things, for example independence with respect to the input value range,

Table 1: Results for synthetically created data. Test data was generated with the same parameters as the training data, so we're looking at *in-distribution* performance. ↑/↓ indicate that higher/lower is better. Errors represent 1 standard deviation over 5 runs with different seeds (standard error of the mean for GPs, because seed influence is negligible), where each run was evaluated with $102\,400$ ($30\,720$ for W_2) samples. Bold indicates that the method(s) are significantly better than all non-bold methods, i.e. when the difference is larger than the root sum of squares of the standard deviations. Overall, GP-ConvCNP outperforms the competing approaches, especially in terms of predictive log-likelihood and sample diversity (compared to an oracle) where applicable. In terms of reconstruction error, our method outperforms prior art on three datasets, but is on par with ConvCNP on two of those. Interestingly, the EQ-GP, which is what our model uses as an initial estimate, performs rather poorly in all but the first example. In the first example, where the EQ-GP is already a decent estimate, our approach leverages that information and matches the oracle GP in predictive performance! The reconstruction error and W_2 of the oracle are zero, so we don't show them here. The dependence of model performance on the number of context points is visualized in Fig. A.2 for the two GP examples.

		Matern-5/2 GP	Weakly Per. GP	Fourier Series	Step Functions
Predictive LL↑	GP (EQ)	1.031 ± 0.075	-8.034 ± 2.260	-0.241 ± 0.752	-2×10^{17}
	GP (Oracle)	1.933 ± 0.095	1.876 ± 0.026		
	NP	-0.496 ± 0.027	-1.161 ± 0.007	-1.743 ± 0.020	-3.287 ± 0.491
	ANP	0.723 ± 0.046	-1.047 ± 0.008	-0.976 ± 0.028	-65.141 ± 60.979
	ConvCNP	1.710 ± 0.038	-0.153 ± 0.033	0.372 ± 0.065	$\mathbf{-0.522 \pm 0.163}$
	GP-ConvCNP	$\mathbf{1.930 \pm 0.031}$	$\mathbf{-0.090 \pm 0.021}$	$\mathbf{1.632 \pm 0.079}$	$\mathbf{-0.532 \pm 0.044}$
Recon. Error↓	GP (EQ)	0.001 ± 0.001	0.028 ± 0.001	0.004 ± 0.001	0.097 ± 0.001
	NP	0.027 ± 0.001	0.500 ± 0.003	0.845 ± 0.074	0.292 ± 0.010
	ANP	$\mathbf{0.008 \pm 0.002}$	0.491 ± 0.004	0.181 ± 0.018	0.284 ± 0.013
	ConvCNP	0.025 ± 0.020	0.109 ± 0.077	$\mathbf{0.042 \pm 0.027}$	$\mathbf{0.121 \pm 0.017}$
	GP-ConvCNP	0.013 ± 0.002	$\mathbf{0.061 \pm 0.007}$	$\mathbf{0.040 \pm 0.023}$	$\mathbf{0.116 \pm 0.017}$
W_2 ↓	GP (EQ)	4.294 ± 0.007	4.521 ± 0.003		
	NP	1.836 ± 0.021	2.745 ± 0.004		
	ANP	1.369 ± 0.048	2.708 ± 0.002		
	ConvCNP				
	GP-ConvCNP	$\mathbf{0.987 \pm 0.086}$	$\mathbf{1.800 \pm 0.045}$		

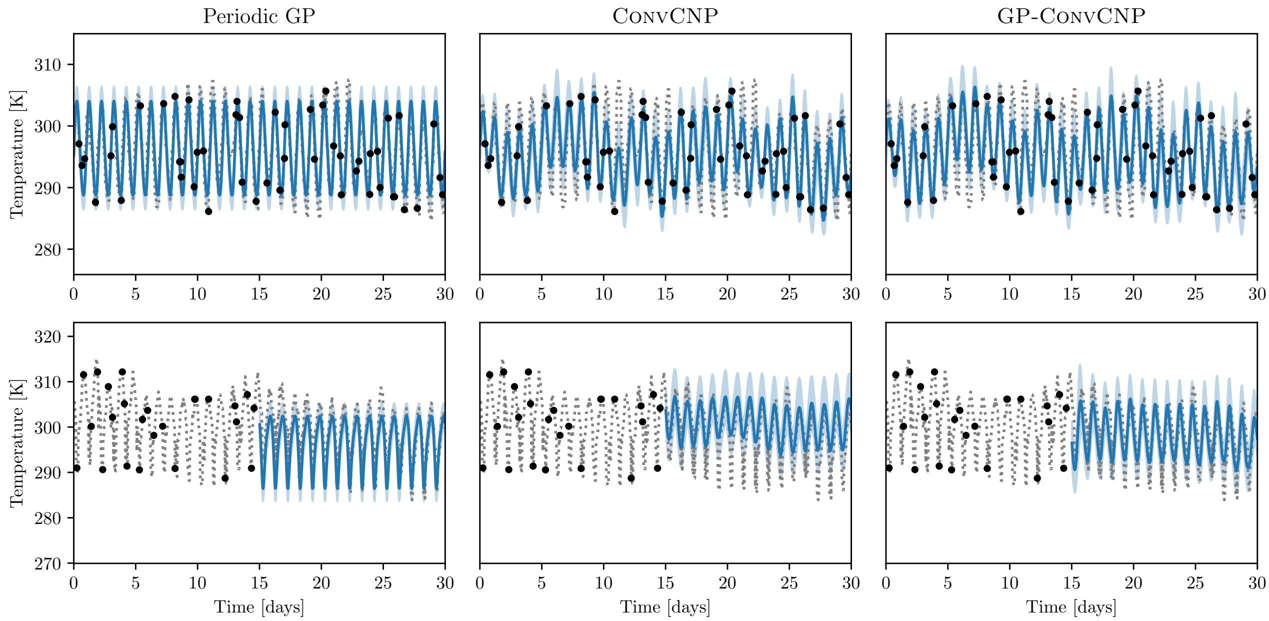

Figure 2: Examples from the temperature time series test set. For the interpolation task (top) we provide context points from the full sequence, for the extrapolation task (bottom) we provide context points in the first half of the sequence and evaluate the second. Both ConvCNP and GP-ConvCNP capture the periodicity of day/night changes in temperature well and are able to extrapolate it. We find that GP-ConvCNP often better matches the amplitude of the true signal, especially for the extrapolation task, which could explain its superior performance in Table 2. Because the temperature signal is periodic, we also show a periodic GP with a commonly used Exponential Sine-Squared kernel. NP and ANP are only shown in the appendix in Fig. A.3, as they were unable to fit the data (similar to the weakly periodic GP data in Fig. 1).

called translation equivariance. This is a key feature of ConvCNP (as long as a stationary kernel is used for interpolation), and we retain this property in GP-ConvCNP. We evaluate two further attributes of generalization, both on real world data: one is the ability to extrapolate the context information, i.e. to produce good predictions well into the future by inferring an underlying pattern; the other is the ability to deal with a distribution shift at test time, in our case a shift from simulated to real world data.

On top of the above, we are also interested in how well the distribution of samples from a model matches the ideal distribution. In general, the latter is not accessible, but for some synthetic examples we describe below, specifically those from a Gaussian Process, we do have access, simply by using the generating GP as an oracle. We can then compare this reference—a Gaussian distribution—with the distribution of samples from our model. Note that one sample is a prediction at all target points at once, as seen for example in Fig. 1. The majority of approaches that estimate differences between distributions fall into the categories of either *f-divergences* or *Integral Probability Measures* (for an overview see for example Sriperumbudur et al. [2009]). The former require evaluations of likelihoods for both distributions, while we only have individual samples from our model. We opt for a parameter-free representative of the IPM category, the Wasserstein distance W_2. We elaborate further on the definition and motivation in Appendix C.4.

We initially test our method on diverse synthetic time series. The first two have also been used in Gordon et al. [2020], and they allow us to evaluate the sample diversity, as outlined above: (1) Samples from a Gaussian Process with a Matern-5/2 kernel. (2) Samples from a Gaussian Process with a weakly periodic kernel. (3) Fourier series with a variable number of components, each of which has random bias, amplitude and phase. (4) Step functions, which were specifically chosen to challenge our model, as the kernel we employ introduces smoothness assumptions that are ill-suited for this problem. All of these are described in greater detail in Appendix C as well as the provided implementation. The size N of the context set is drawn uniformly from $[3, 100)$ and the size M of the target set from $[N, 100)$ following Le et al. [2018]. We further join the context set into the target set as done in Garnelo et al. [2018a,b]. Examples can be seen in Fig. 1.

The first real world dataset we look at are weather recordings for several different US, Canadian and Israeli cities. In particular we focus on temperature measurements in hourly intervals that have been collected over the course of 5 years (see Appendix C.2). Temperatures in each city are normalized by their respective means and standard deviations. We

Table 2: Results on the real world datasets. Again, ↑/↓ indicate that higher/lower is better and errors represent 1 standard deviation over 5 runs with different seeds (standard error of the mean for GP, because seed influence is negligible). (left) For the temperature interpolation task, context points are randomly sampled from the test interval, for the temperature extrapolation task we provide context points in the first half of the interval and measure performance on the second half (as seen in Fig. 2). For comparison, we also show a periodic GP with an Exponential Sine-Squared kernel. (right) For the population dynamics, models are trained on simulated data, so the real world data (also shown in Fig. 3) is likely out-of-distribution, as evidenced by the stark drop in performance. There is no obvious choice of kernel if one wanted to apply a GP to this problem.

		Temperature Time Series		Population Dynamics	
		interpolation	extrapolation	simulated	real
Predictive LL↑	GP (per.)	-2.075 ± 0.237	-46.611 ± 2.557		
	NP	-0.855 ± 0.003	-1.267 ± 0.011	0.527 ± 0.051	-33.070 ± 7.636
	ANP	-0.733 ± 0.008	-1.938 ± 0.381	1.027 ± 0.033	-29.714 ± 9.210
	ConvCNP	$\mathbf{-0.522 \pm 0.008}$	-1.261 ± 0.062	$\mathbf{1.374 \pm 0.017}$	-23.540 ± 12.441
	GP-ConvCNP	$\mathbf{-0.515 \pm 0.019}$	$\mathbf{-1.190 \pm 0.016}$	1.337 ± 0.029	$\mathbf{-5.382 \pm 2.625}$
Recon. Error↓	GP (per.)	0.274 ± 0.001			
	NP	0.238 ± 0.002		0.018 ± 0.001	1.053 ± 0.015
	ANP	0.198 ± 0.007		0.008 ± 0.004	0.772 ± 0.020
	ConvCNP	$\mathbf{0.106 \pm 0.002}$		$\mathbf{0.002 \pm 0.001}$	$\mathbf{0.374 \pm 0.019}$
	GP-ConvCNP	$\mathbf{0.123 \pm 0.018}$		0.004 ± 0.001	$\mathbf{0.411 \pm 0.026}$

randomly sample sequences of $\sim$1 month as instances and evaluate two tasks, taking US and Canadian cities as the training set and Israeli cities as the test set:

1. Interpolation, where we draw context points and target points randomly from the entire sequence (i.e. the same as in the synthetic examples).
2. Extrapolation, where context points are drawn from the first half of the sequence and performance is evaluated on the second half (as shown in Fig. 2). We can reasonably be sure that temperature changes between day and night occur in the future with the same frequency, so extrapolating this pattern is a good test of a model's ability to generalize.

The second real world dataset are measurements of a predator-prey population of lynx and hare. Such population dynamics are often approximated by Lotka-Volterra equations [Leigh, 1968], so we train models on simulated population dynamics and test on both the simulated and real world data. Gordon et al. [2020] used this dataset as well, but only to qualitatively show that ConvCNP can be applied to it. The analysis will allow us to quantify how robust the models are to a shift in distribution at test time, as the simulation parameters are almost certainly not an ideal fit for the real world data. For details on the simulation process we refer to Appendix C.3.

Finally, even though the focus of our work is on time series data, we include some image experiments, mainly for the purpose of a more nuanced direct comparison with ConvCNP. In particular, we compare the models on MNIST [Lecun et al.], CIFAR10 [Krizhevsky, 2009] and CelebA [Liu et al., 2015]. For the latter two, we work on resampled versions at 32^2 resolution. More details are given in Appendix E.

4 RESULTS

Table 1 shows results for the various synthetic time series. In this experiment the models are trained and tested on random samples generated in the same way, so these results measure *in-distribution* performance. We find that GP-ConvCNP is the overall best performing method, significantly so in terms of predictive performance for 3 out of the 4 time series and performing on par with ConvCNP on the other. Reconstruction performance is on par with ConvCNP in 3 out of 4 instances and significantly better in one. For reference, we also show results for a Gaussian Process with EQ kernel (what our model uses) and the oracle where available. Evidently, the initial GP estimate in our model doesn't have to be very good, but when it is, like in the Matern-5/2 case, our approach leverages this and even matches the oracle in performance. For examples originating from a Gaussian Process, we can evaluate the sample diversity with respect to the oracle GP, finding that GP-ConvCNP significantly outperforms the other methods in this regard. It is important to note, however, that this measure does not fully isolate the sample diversity. A low reconstruction error, for example, will also improve the W_2, which is likely the reason that ANP still performs better than NP, even though the former hardly displays any variation in its samples, as seen in Fig. 1.

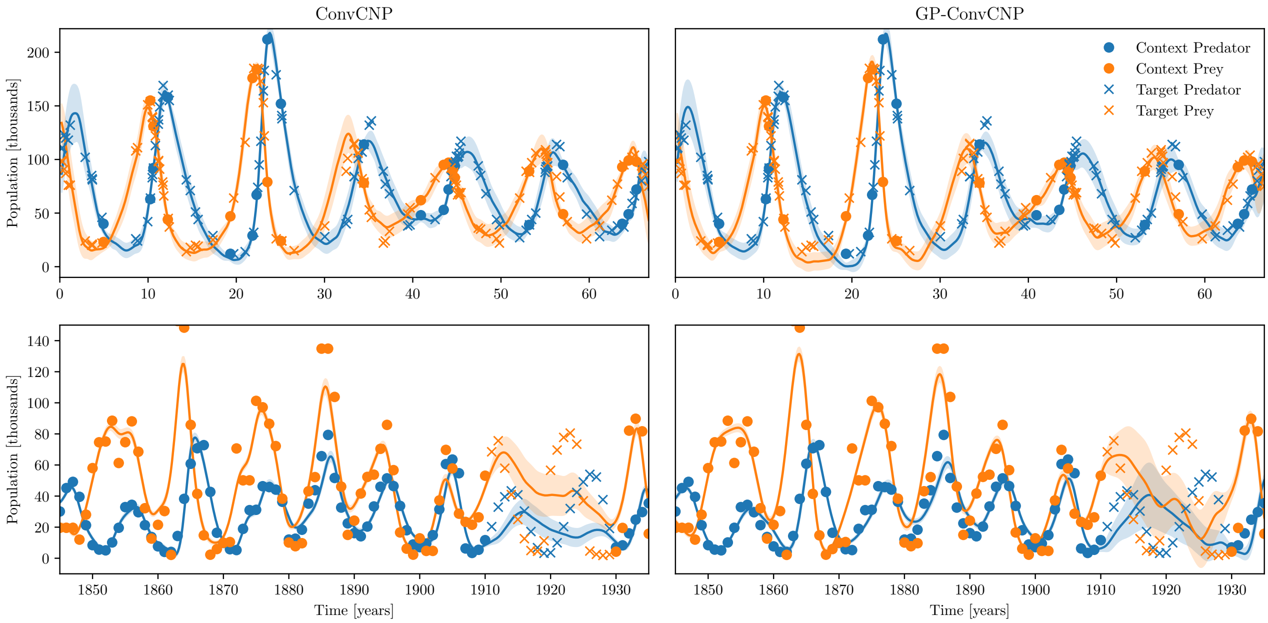

Figure 3: Example of CONVCNP and GP-CONVCNP applied to the simulated Lotka-Volterra population dynamics (top) and to the real Hudson Bay Company lynx-hare dataset (bottom). Both perform well on the simulated (i.e. in-distribution) data and seem to struggle fitting the test interval on the real world data. Not however how the predicted uncertainty is larger for GP-CONVCNP. We display the best out of 5 models in each case, and for CONVCNP the performance is much more volatile, as seen in Table 2. NP and ANP perform poorly on the real world data, the corresponding figure is Fig. A.4.

The figure also shows how NP and ANP struggle to fit high frequency signals, while CONVCNP and GP-CONVCNP are able to. The sample diversity in GP-CONVCNP is larger than in ANP, but samples are only significantly different from the mean prediction when further away from the context points in areas of high predictive uncertainty (shaded areas correspond to 1σ). In contrast, samples from the NP are more diverse throughout, at the expense of accurately matching the context points.

Table 3: Results for the image experiments, in terms of predictive log-likelihood (i.e. higher is better) on the respective test sets. Errors represent 1 standard deviation over 10 runs with different seeds. Bold indicates a significant difference, i.e. when the difference is larger than the root sum of squares of the standard deviations. GP-CONVCNP outperforms CONVCNP overall, with a slight (non-significant) advantage for CONVCNP on MNIST. Visual examples and more details on the image experiments are given in Appendix E.

	CONVCNP	GP-CONVCNP
MNIST	4.133 ± 0.057	4.077 ± 0.026
CIFAR10	2.462 ± 0.006	$\mathbf{2.744 \pm 0.008}$
CelebA	2.212 ± 0.006	$\mathbf{2.468 \pm 0.008}$

Examples from the temperature time series dataset can be seen in Fig. 2. The key characteristic of the signal is the temperature change between day and night, making it a high frequency signal not unlike the weakly periodic GP samples in the synthetic dataset. NP and ANP were not able to fit these signals, as can be seen in Fig. A.3. The top row of Fig. 2 shows an example of the regular interpolation task, the bottom row an example of the extrapolation task, which we deem an important aspect of generalization. CONVCNP and GP-CONVCNP are both able to interpolate as well as extrapolate the correct temperature pattern, but occasionally CONVCNP underestimates the amplitude when extrapolating. We also show an example of a periodic GP using an Exponential Sine-Squared kernel, which is a common choice for periodic signals. It fails to capture finer variations in the signal and often struggles to infer the right frequency, which results in its poor extrapolation performance in Table 2. We find that while CONVCNP and GP-CONVCNP perform on par for the interpolation task, GP-CONVCNP performs significantly better than the other methods on the extrapolation task.

To measure how robust the different members of the Neural Process family are to a distribution shift at test time, we train models on population dynamics simulated as Lotka-Volterra processes, and evaluate performance both on simulated (*in-distribution*) and real world (*out-of-distribution*) data. The real world dataset, along with a simulated example, can be seen in Fig. 3. While both CONVCNP and GP-CONVCNP fit the simulated data well, they struggle with the test interval on the real data. This is reflected in Table 2 as well, where we find that ConvCNP performs better than GP-ConvCNP (even significantly so, albeit not with a huge difference) on the simulated data. Applied to the real world dataset, all methods experience a large drop in performance, indicating that this is indeed a significant distribution shift. GP-ConvCNP is by far the best performing method here, which is likely because of a better estimate of the preditive uncertainty. Note how the uncertainty predicted by ConvCNP is smaller than that of GP-ConvCNP in Fig. 3 (the figure shows 1σ). The predictions we show here are from the best performing seed in each case, other ConvCNP models predicted an even narrower distribution. We selected this particular interval for testing because it's the same interval Gordon et al. [2020] show in the ConvCNP paper. We also evaluated with context points drawn randomly from the entire interval (i.e. the same way we evaluate on the simulated data), and GP-ConvCNP still performs significantly better than the competing approaches (see Table A.1).

ConvCNP also showed performance improvements compared to NP and ANP when applied to image data. While the focus of our work is on time series, we were also interested to see if our model yields any benefits in this domain. It does indeed, as seen in Table 3, where GP-ConvCNP outperforms ConvCNP on both CIFAR10 and CelebA (ConvCNP has a non-significant advantage on MNIST). Examples are given in Appendix E, where we don't see any meaningful difference in visual quality. The latter only "measures" the quality of the mean prediction, so we suspect that the performance improvement is due to a more accurate predictive uncertainty.

5 RELATED WORK

Neural Processes have inspired a number of works outside of the ones we discuss. Louizos et al. [2019] propose to not merge observations into a global latent space, but instead learn conditional relationships between them. This is especially suitable for semantically meaningful clustering and classification. Singh et al. [2019] and Willi et al. [2019] address the problem of overlapping and changing dynamics in the generating process of the data, a special case we do not include here. With a simple Gaussian kernel, we wouldn't expect our model to perform well in that scenario, but one could of course introduce inductive bias in the form of e.g. non-stationary kernels, when translation equivariance is no longer desired. NPs have also been scaled to extremely complex output spaces like in *Generative Query Networks* [Eslami et al., 2018, Rosenbaum et al., 2018], where a single observation is a full image. GQN directly relates to the problem of (3D) scene understanding [Sitzmann et al., 2019, Engelcke et al., 2020].

Gordon et al. [2020] build their work (ConvCNP) upon recent contributions in the area of *learning on sets*, i.e. neural networks with set-valued inputs [Zaheer et al., 2017, Wagstaff et al., 2019], which has mostly been explored in the context of point clouds [Qi et al., 2017b,a, Wu et al., 2019]. Especially the work of Wu et al. [2019] is closely related to Gordon et al. [2020], also employing a CNN on a kernel density estimate, but their application is not concerned with time series. *Bayesian Neural Networks* [Neal, 1996, Graves, 2011, Hernández-Lobato and Adams, 2015] also address the problem of learning distributions over functions, but often implicitly, in the sense that the distributions over the weights are used to estimate uncertainty [Blundell et al., 2015, Gal and Ghahramani, 2016]. We are interested in this too, but in our scenario we want to be able to condition on observations at test time.

The main limitation of *Gaussian Processes* is their computational complexity and many works are dedicated to improving this aspect, often via approximations based on inducing points [Snelson and Ghahramani, 2006, Titsias, 2009, Gardner et al., 2018, Wilson and Nickisch, 2015] but also other approaches [Deisenroth and Ng, 2015, Rahimi and Recht, 2007, Le et al., 2013, Cheng and Boots, 2017, Hensman et al., 2013, 2015, Salimbeni et al., 2018], even for exact GPs [Wang et al., 2019]. Rather than competing with these approaches, our model will be able to leverage developments in this area. Some of the above try to find more efficient kernel representations and are thus closely related to the idea of *kernel learning*, i.e. the idea to combine the expressiveness of (deep) learning approaches with the flexibility of kernel methods, for example Yang et al. [2015], Wilson et al. [2016b,a], Tossou et al. [2019], Calandra et al. [2016]. The key difference to our work is that these approaches attempt to learn kernels as an input to a kernel method, while we learn to make the output of a kernel method more expressive.

6 DISCUSSION

We have presented a new model in the Neural Process family that extends ConvCNP by incorporating a Gaussian Process into it. We show on both synthetic and real time series that this improves performance overall, but most markedly when generalization is required: our model, GP-ConvCNP, can better extrapolate to regions far from the provided context points and is more robust when moving to real world data after training on simulated data. We further retain translation equivariance, a key feature of ConvCNP, as long as a stationary kernel is used for the GP. The introduction of the latter also allows us to draw multiple samples from the model, where the distribution of samples from our model better matches the samples from an oracle than those from a regular Neural Process or an Attentive Neural Process do. Our model uses the prediction from a GP with an EQ-kernel as an initial estimate. Interestingly, this estimate needn't be very good—our model can learn periodicity even with a non-periodic input kernel—but when it is, our model can fully leverage it and even match the performance of an oracle, as seen in Table 1. An advantage all Neural Process flavors enjoy compared to many conventional time series prediction methods such as ARIMA models (see e.g. Hyndman and Athanasopoulos [2018]) is that they naturally work on non-uniform time series, with observations acquired at arbitrary times.

Of course, with the benefits of GPs we also inherit their limitations. GPs are typically slow, naively requiring $O(N^3)$ operations in the number of context observations, and our model inherits this complexity. While this was a non-issue on the time series data used in our work, GP-CONCNP was noticably slower than CONVCNP (roughly 1.5x) in the image experiments, which we included for a more complete comparison with CONVCNP. Our model still outperformed CONVCNP, but for larger images the improved performance will likely not be worth the additional cost. Making GPs faster is a very active research area, as outlined above. For our model specifically it seems reasonable to leverage work on deep kernels [Wilson et al., 2016b] or to learn mappings before the GP prediction like in Calandra et al. [2016] in order to learn more meaningful GP posteriors that capture information about the training distribution. We do expect that our model is well suited to also work with these approximate methods, as we modify the prediction from the GP with a powerful neural network that should be able to correct minor approximation errors. For example, KISS-GP Wilson and Nickisch [2015] only has linear complexity, so incorporating it or one of the many other efficient approximate methods into our model should allow it to scale to much larger datasets. We leave a verification of this for future work.

References

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. *arXiv:1402.5876 [cs, stat]*, 2016.

Ching-An Cheng and Byron Boots. Variational inference for gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems 30*, pages 5184–5194. 2017.

Marc Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490, 2015.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020.

S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Jacob Gardner, Geoff Pleiss, Ruihan Wu, Kilian Weinberger, and Andrew Wilson. Product kernel interpolation for scalable gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1407–1416, 2018.

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 *International Conference on Machine Learning*, pages 1704–1713, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018b.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24*, pages 2348–2356. 2011.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification.

In *Internation Conference on Artificial Intelligence and Statistics*, pages 351–360, 2015.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, volume 37, pages 1861–1869, 2015.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

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.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Quoc Le, Tamas Sarlos, and Alexander Smola. Fastfood - computing hilbert space expansions in loglinear time. In *International Conference on Machine Learning*, pages 244–252, 2013.

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

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 86(11): 2278–2324.

Egbert G Leigh. *Ecological role of Volterra's equations*. Lectures on mathematics in the life sciences. Princeton University, 1968.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. In *Advances in Neural Information Processing Systems 32*, pages 8743–8754. 2019.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer, 1996.

Charles R. Qi, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30*, pages 5099–5108. 2017b.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. 2007.

Carl Edward Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. 371(1984):20110550, 2013.

Dan Rosenbaum, Frederic Besse, Fabio Viola, Danilo J. Rezende, and S. M. Ali Eslami. Learning models for visual 3d localization with implicit mapping. In *NeurIPS Bayesian Deep Learning Workshop*, 2018.

Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. Orthogonally decoupled variational gaussian processes. In *Advances in Neural Information Processing Systems 31*, pages 8711–8720. 2018.

Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential neural processes. In *Advances in Neural Information Processing Systems 32*, pages 10254–10264. 2019.

Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems 32*, pages 1119–1130. 2019.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. 2006.

Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. On integral probability metrics, phi-divergences and binary classification. *arXiv:0901.2698 [cs, math]*, 2009.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv:1905.12131 [cs, stat]*, 2019.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems 31*, page 5339–5349, 2018.

Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Ingmar Posner, and Michael Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, 2019.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems 32*, pages 14622–14632. 2019.

Timon Willi, Jonathan Masci, Jürgen Schmidhuber, and Christian Osendorfer. Recurrent neural processes. *arXiv:1906.05915 [cs, stat]*, 2019.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems 29*, pages 2586–2594. 2016a.

Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (KISS-GP). In *International Conference on Machine Learning*, 2015.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, 2016b.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. A la carte – learning fast kernels. In *International Conference on Artificial Intelligence and Statistics*, pages 1098–1106, 2015.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems 30*, pages 3391–3401. 2017.