DECOUPLED KERNEL NEURAL PROCESSES: NEURAL NETWORK-PARAMETERIZED STOCHASTIC PROCESSES USING EXPLICIT DATA-DRIVEN KERNEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Processes (NPs) are a class of stochastic processes parametrized by neural networks. Unlike traditional stochastic processes (*e.g.*, Gaussian processes), which require specifying explicit kernel functions, NPs implicitly learn kernel functions appropriate for a given task through observed data. While this data-driven learning of stochastic processes has been shown to model various types of data, the current NPs' implicit treatment of the mean and the covariance of the output variables limits its full potential when the underlying distribution of the given data is highly complex. To address this, we introduce a new neural stochastic processes, Decoupled Kernel Neural Processes (DKNPs), which explicitly learn a separate mean and kernel function to directly model the covariance between output variables in a data-driven manner. By estimating kernel functions with self- and mixed attentive neural networks, DKNPs demonstrate improved uncertainty estimation in terms of conditional likelihood and diversity in generated samples in 1-D and 2-D regression tasks, compared to other concurrent NP variants. Also, maintaining explicit kernel functions, a key component of stochastic processes, allows the model to reveal a deeper understanding of underlying distributions.

1 INTRODUCTION

Neural processes (NPs) (Garnelo et al., 2018a;b) are a class of stochastic processes parametrized by neural networks. By embracing statistical properties in stochastic processes, NPs can effectively estimate the uncertainty of underlying distributions of functions with a set of realizations and their data points. Different from traditional stochastic processes (*e.g.*, Gaussian processes (GP) (Rasmussen & Williams, 2006)), NPs learn data-driven stochastic processes without a need to specify or keep an explicit form of kernel functions. As a result of their simplicity and flexibility, there have been numerous efforts to further develop improved variants of NPs (Kim et al., 2019; Lee et al., 2020; Gordon et al., 2020) and apply them to various downstream tasks (Singh et al., 2019; Requeima et al., 2019).

Though significant progress has been made in NPs, the current architectures of NPs either fails to capture output dependencies as in Conditional NPs (Garnelo et al., 2018a; Gordon et al., 2020), or indirectly capture the full stochasticity present in the traditional stochastic processes. For instance, different from GPs, conventional NPs reserve stochasticity in a global latent variable and output variables separately. The output variables estimate point-wise uncertainty, which corresponds to the diagonal elements of a kernel matrix. Similarly, the global latent variable takes charge of the functional uncertainty and diversity, represented by the full covariance matrix in GPs. Due to this inductive bias of conventional NPs, the role of estimating functional stochasticity is mainly assigned to a fixed-length vector (*i.e.* the global latent variable), and consequently, capturing the underlying distributions can be restricted in complex scenarios (*e.g.*, variable relationships are periodic or abruptly changing at a certain point). Although some approaches attempt to alleviate the problem by introducing attention (Kim et al., 2019) and bootstrapping (Lee et al., 2020) on top of conventional NPs, the problem still exists as the architectural limitation (*i.e.* implicit modeling of the mean and covariance) has not been addressed directly. Besides this, as NPs implicitly learn the kernel functions inside the model, the interpretability of kernels such as in GPs (Lloyd et al., 2014) is diminished.

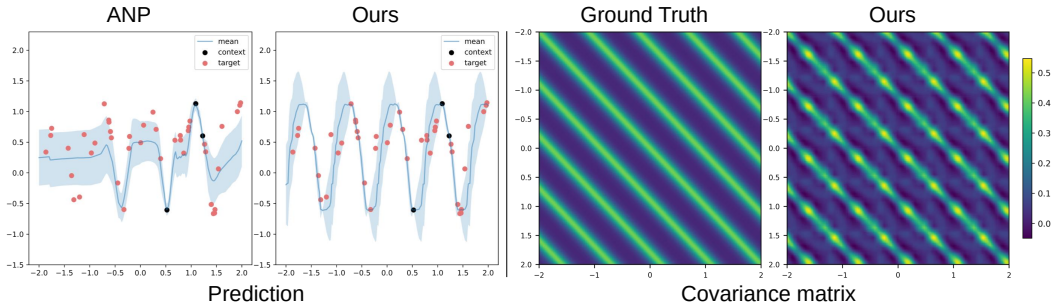

Figure 1: **(Left)** Comparison of predictions given context points (black dot) by the Attentive NP (ANP) and DKNP (Ours) after learning periodic kernels sampled from GPs. The mean (blue curve) and sigma (shaded blue area) predicted from the DKNP better represent the data, including target points (red dot) compared to the ANP. **(Right)** Visualization of kernel functions learned by the DKNP. As a result of kernel learning in the DKNP, the underlying prior distribution of data can be inferred (periodic kernel in this case).

To address this concern, we propose *Decoupled Kernel Neural Processes* (DKNPs), a neural stochastic processes that explicitly learn a separate mean and kernel function to directly model the covariance between output variables in a data-driven manner. Our experiments in 1-D and 2-D regression tasks reveal that the DKNP outperforms concurrent NP variants in terms of predictive likelihood, better global coherence of generated samples, and improved interpretability via explicitly learned kernels.

2 Background

2.1 Neural Process

Given a stochastic process sample consisting of n points, let us denote the input and output as $X = \{\boldsymbol{x}_i\}_{i=1}^n$ and $Y = \{\boldsymbol{y}_i\}_{i=1}^n$, respectively, where $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$ and $\boldsymbol{y}_i \in \mathbb{R}^{d_y}$. For a set of target input $X_T = \{\boldsymbol{x}_i\}_{i \in T} \subset X$, NPs model the conditional distribution of target outputs Y_T conditioned on the context set $(X_C, Y_C) = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in C}$ using factorized Gaussian distribution:

$$\log p(Y_T | X_T, X_C, Y_C) = \sum_{i \in T} \log p(\boldsymbol{y}_i | \boldsymbol{x}_i, X_C, Y_C). \tag{1}$$

For obtaining the predictive distribution $\log p(\boldsymbol{y}_i | \boldsymbol{x}_i, X_C, Y_C)$, NPs use an encoder-decoder architecture which ensures the permutation invariance of the context set (X_C, Y_C). The NP encoder consists of two separate paths, namely the deterministic path and the latent path. For the deterministic path, f_θ represents each context points in $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in C}$ as $\boldsymbol{r}_i \in \mathbb{R}^{d_r}$, *i.e.*, $\boldsymbol{r}_i = f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)$. Then, we aggregate the $\boldsymbol{r}_i$'s by averaging them across all context points, $\boldsymbol{r}_C = \frac{1}{n_c} \sum_{i \in C} \boldsymbol{r}_i$ where $n_c = |C|$. This vector $\boldsymbol{r}_C \in \mathbb{R}^{d_r}$ is the summarized representation of context points, and it is permutation invariant over the order of $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in (X_C, Y_C)$.

The latent path of the NP encoder operates in a similar fashion to the deterministic path, *i.e.*, $\boldsymbol{e}_C = \frac{1}{n_c} \sum_{i \in C} \boldsymbol{e}_i$ where $\boldsymbol{e}_i = g_\phi([\boldsymbol{x}_i; \boldsymbol{y}_i])$. Unlike the deterministic path, however, the latent path uses stochastic layers for obtaining a distribution of the latent variable $\boldsymbol{z}_C \in \mathbb{R}^{d_z}$; $q(\boldsymbol{z}_C | \boldsymbol{e}_C) = \mathcal{N}(\boldsymbol{z}_C; \boldsymbol{\mu}_{z_C}, \boldsymbol{\sigma}^2_{z_C})$ where $\boldsymbol{\mu}_{z_C}, \boldsymbol{\sigma}_{z_C}$ are the output of the additional fully-connected layer applied to $\boldsymbol{e}_C$. Finally, by concatenating these aggregated vectors $\boldsymbol{r}_C, \boldsymbol{z}_C$ with the target inputs $\boldsymbol{x}_i \in X_T$, the decoder h_ψ produces the predictive distribution $p(\boldsymbol{y}_t | \boldsymbol{x}_t, \boldsymbol{r}_C, \boldsymbol{z}_C) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_y, \boldsymbol{\sigma}^2_y)$ where $(\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y) = h_\psi(\boldsymbol{x}_t, \boldsymbol{r}_C, \boldsymbol{z}_C)$. Note that the $\boldsymbol{\sigma}_y$ is used for capturing the point-wise uncertainty. During the training phase, the parameters are learned by maximizing the evidence lower bound of $\log p(Y_T | X_T, X_C, Y_C)$ via the reparametrization trick (Kingma & Welling, 2014; Rezende et al., 2014) as follows:

$$\log p(Y_T | X_T, X_C, Y_C) \geq \mathbb{E}_{q(\boldsymbol{z}_T | X_T, Y_T)}[\log p(Y_T | X_T, \boldsymbol{z}_T)] - KL(q(\boldsymbol{z}_T | \boldsymbol{e}_T) \| q(\boldsymbol{z}_C | \boldsymbol{e}_C)) \tag{2}$$

where $q(\boldsymbol{z}_t | \boldsymbol{e}_T) = q(\boldsymbol{e}_t | X_T, Y_T)$ is a distribution of latent variable by encoding the set of target points $(X_T, Y_T) = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in T}$. This objective function consists of two parts: 1) the reconstruc-

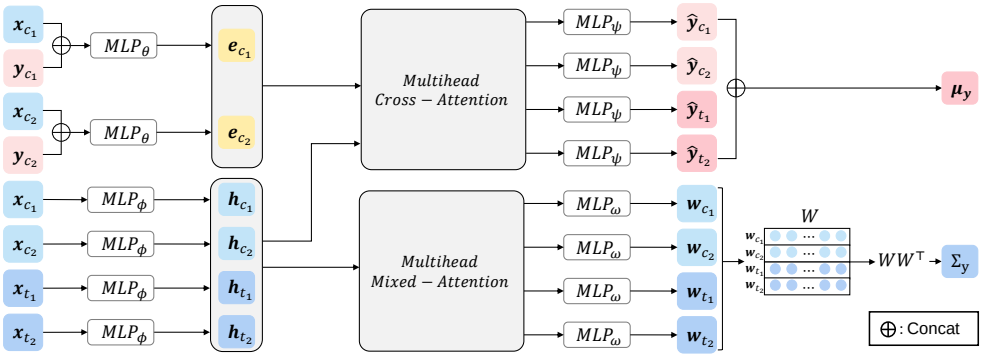

Figure 2: Model architecture of DKNPs. DKNPs estimate the predictive distribution $\mathcal{N}(y; \boldsymbol{\mu}_y, \Sigma_y)$ by using two attention-based deterministic paths where Multihead Cross-Attention (MCA) models the mean vector (upper path) and Multihead Mixed-Attention (MMA) models the full covariance matrix (lower path).

tion loss for the target points and 2) the KL divergence term, minimizing the divergence between two distributions $q(\boldsymbol{z}_t|X_T, Y_T)$ and $q(\boldsymbol{z}_c|X_C, Y_C)$. Note that, in practice, we assume that $X_C \subset X_T$ during the training phase. The KL divergence term encourages two distributions inferred by the context sets and target sets to be similar, which is reasonable because two sets are generated from the same function. Therefore, during the inference phase, the distribution of $q(\boldsymbol{z}_c|X_C, Y_C)$ captures the functional stochasticity which is demonstrated with the coherent sample generation. It can be thought that $\boldsymbol{z}_c$ learns to capture the correlation of output variables of the stochastic processes.

2.2 Attentive Neural Processes

Multihead Attention Given n key-value pairs of matrices $K \in \mathbb{R}^{n \times d_{model}}$ and $V \in \mathbb{R}^{n \times d_{model}}$, and m queries $Q \in \mathbb{R}^{m \times d_{model}}$, the scaled dot product attention is formulated as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^\top}{\sqrt{d_{model}}}\right)V \quad \in \mathbb{R}^{m \times d_{model}}, \tag{3}$$

where K, V, and Q are projected by learnable linear maps W_s^K, W_s^V, and W_t^Q from the source S and target T. When S and T are the same, attention is often called self-attention; otherwise, cross-attention.

The attention mechanism can be calculated from multiple subspaces, namely, multihead attention (MHA) (Vaswani et al., 2017). Denoting a single attention head as $Head_i = Attention(Q_i, K_i, V_i)$, the aggregate attention from multiple subspaces can be expressed as:

$$\text{MHA}(Q, K, V) = Concat(Head_1, ..., Head_H)W^O \quad \in \mathbb{R}^{m \times d_{model}}, \tag{4}$$

where W^O is the learnable linear map for aggretating the subspaces.

Attentive NPs Attentive NPs (ANPs) leverage attention to resolve the underfitting issue in NPs. Unlike NPs that produce a single variable $\boldsymbol{r}_C$ from the deterministic path, ANPs utilize a query-specific variable $\boldsymbol{r}_i^*$ by applying the attention score a_i for each $\boldsymbol{r}_i$ during the aggregation of the deterministic path, formulated as $\boldsymbol{r}_i^* = \sum_{i \in C} a_i \cdot \boldsymbol{r}_i$. The attention-based aggregation of ANPs resembles how the GPs utilize the correlation to estimate the predictive distribution of the context and the target points.

3 Decoupled Kernel Neural Processes

Decoupled Kernel Neural Processes use attention to explicitly learn a separate mean and kernel function so as to directly model covariances between output variables with related input variables in a data-driven fashion, which is contrary to conventional NPs that implicitly model the mean and kernel function through the latent variable $\boldsymbol{z}$. As shown in Figure 2, DKNPs estimate the predictive distribution as multivariate Gaussian $\mathcal{N}(y; \boldsymbol{\mu}_y, \Sigma_y)$ by using two attention-based deterministic paths.

Multihead Cross-Attention (MCA) models the mean vector (upper path) and Multihead Mixed-Attention (MMA) models the full covariance matrix (lower path). Here, attention modules are utilized since Le et al. (2018); Kim et al. (2019) have demonstrated attention was helpful in achieving low predictive uncertainty near the context points. With the predictive distribution $\mathcal{N}(y; \boldsymbol{\mu}_y, \Sigma_y)$, DKNPs are trained and evaluated based on the function likelihood.

The core design of DKNPs is motivated by the predictive posterior distribution of GPs, where X_T, X_C, Y_C are used for deriving the posterior mean,[1] but only X_T, X_C for deriving the posterior covariance as follows:

$$\textbf{GP:} \quad p(Y_T|X_T, X_C, Y_C) = \mathcal{N}\Big(Y_T; \Sigma_{X_C,X_T}^T \Sigma_{X_C,X_C}^{-1} Y_C, \;\; \Sigma_{X_T,X_T} - \Sigma_{X_C,X_T}^T \Sigma_{X_C,X_C}^{-1} \Sigma_{X_C,X_T}\Big), \tag{5}$$

$$\textbf{DKNP:} \quad p(Y_T|X_T, X_C, Y_C) = \mathcal{N}\Big(Y_T; \text{MLP}_\psi \circ \text{MCA}\big(\text{MLP}_\theta(X_C, Y_C), \text{MLP}_\phi(X_T, X_C)\big), \text{MLP}_\omega \circ \text{MMA}\big(\text{MLP}_\phi(X_T, X_C)\big)\Big), \tag{6}$$

where, in contrast to GPs, DKNPs' covariance is learned via attention.[2] This decoupled process allows DKNPs to explicitly learn the prior of the given dataset, and thus act as a true generative process, ensuring the global consistency of all points in the given stochastic process samples. Unlike DKNPs, NPs pack all information (X_C, Y_C, X_T) into latent variables to derive both mean and covariance, thus inherently becoming a conditional process that requires a sufficient amount of context points, unable to explicitly learn a prior.

Specifically, the DKNPs pass each context point $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i \in C}$, the concatenation of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, to MLP_θ and represent it as $\boldsymbol{e}_i$. Similarly, we produce the representation vector of x_i, $\boldsymbol{h}_i$, using another MLP_ϕ for all inputs $\{x_i\}_{i \in C \cup T}$. Then, $\boldsymbol{e}_i$ and $\boldsymbol{h}_i$ are passed to the MCA module to create the mean vector $\boldsymbol{\mu}_y$. All heads in the MCA module perform cross-attention, $Q = \{\boldsymbol{h}_i\}_{i \in C \cup T}, K = \{\boldsymbol{h}_i\}_{i \in C}$, and $V = \{\boldsymbol{e}_i\}_{i \in C}$. After the MCA, the last MLP_ψ generates the predictive mean vector $\boldsymbol{\mu}_y$ for each data point. Intuitively, this can be interpreted as predicting the target mean based on the context and the correlation, which resembles the estimation of the predictive distribution in GPs and ANPs.

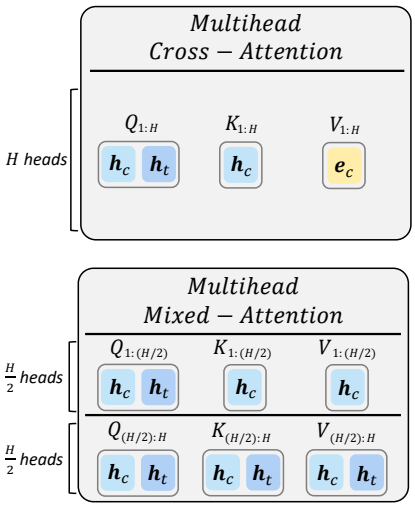

Figure 3: Query, key, value details of the multihead cross- and mixed- attention mechanisms shown in Figure 2.

Different from the NPs, DKNPs explicitly capture the correlation between the output variables using the multihead mixed attention (MMA). MMA contains multihead attention of H heads with both self-attention and cross-attention. Formally, $\frac{H}{2}$ heads perform self-attention between all input points $Q = K = V = \{\boldsymbol{h}_i\}_{i \in C \cup T}$, and the other $\frac{H}{2}$ heads perform cross-attention between context and target, $Q = \{\boldsymbol{h}_i\}_{i \in C \cup T}, K = V = \{\boldsymbol{h}_i\}_{i \in C}$. Then MLP_ω produces the representation vector $\boldsymbol{w}_i \in \mathbb{R}_w^d$ for each position, which are combined to generate the covariance matrix $\Sigma = WW^\top$, where $W_{i,:} = \boldsymbol{w}_i$ and $\Sigma_{ij} = \text{kernel}(x_i, x_j) = \boldsymbol{w}_i^\top \boldsymbol{w}_j$. Based on the posterior covariance of equation 5, it might seem sufficient to only use self-attention and let it learn the correlation. However, self-attention alone will produce Σ without distinguishing X_C and X_T, thus improperly modeling the stochastic process. Using a combination of self-attention and cross-attention ensures DKNPs to learn the correct correlation between data points. Note that without any context, only $\{\boldsymbol{h}_i\}_{i \in T}$ are fed to the self-attention heads to produce Σ, allowing us to view the prior learned by DKNPs, unlike conventional NPs.

To train the DKNPs, the obtained mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ act as parameters of a predictive distribution $\mathcal{N}(Y; \boldsymbol{\mu}_Y, \Sigma_Y)$. Instead of maximizing the lower bound of the log-likelihood

[1] We follow the typical GP formulation where the mean function is set to zero.

[2] Note that DKNPs assume $C \subset T$ during training, which makes Eq. 5 and Eq. 6 technically different. This difference comes from DKNPs using all data points for better learning the kernels, unlike fixed-kernel GPs. DKNPs and GPs, however, are different methods and superior empirical performance led to the current design choice.

Table 1: Results (log-likelihood) of the NP variants and DKNP in multiple 1D regression tasks

Methods	RBF	Periodic	Matérn 5/2
CNP	0.695 (±0.010)	-0.328 (±0.032)	0.558 (±0.006)
NP	0.577 (±0.015)	-0.619 (±0.005)	0.417 (±0.009)
BNP	0.754 (±0.004)	-0.018 (±0.042)	0.617 (±0.006)
ANP	1.086 (±0.001)	0.831 (±0.011)	1.020 (±0.000)
BANP	1.084 (±0.001)	0.821 (±0.018)	1.018 (±0.001)
Ours	**1.108** (±0.001)	**0.939** (±0.006)	**1.038** (±0.008)

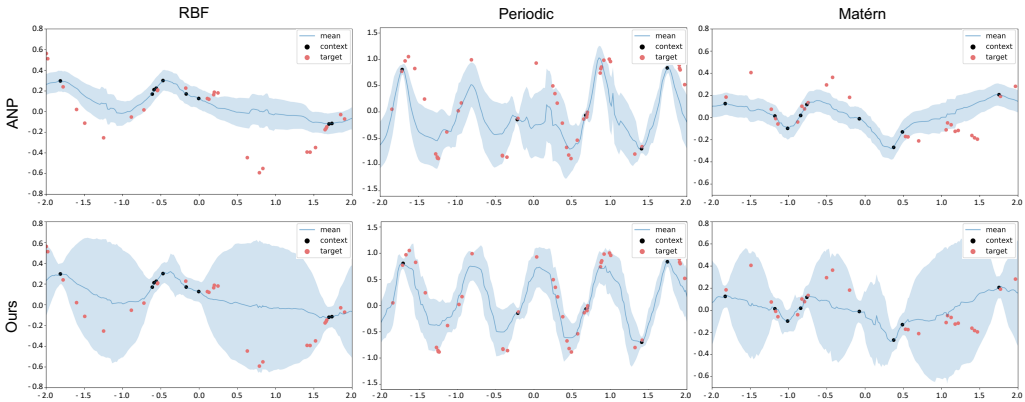

Figure 4: Comparison of predictions by ANP and DKNP. The dark blue line indicates the mean, and the light blue shade indicates one standard deviation range.

as in most NP models, the training objective of DKNPs is to maximize the tractable log-likelihood of the Gaussian as follows:

$$\log p(Y_T|X_T, X_C, Y_C) = \log \mathcal{N}(Y_T; \boldsymbol{\mu}_Y, \Sigma_Y) \text{ where } \Sigma_Y = WW^\top . \tag{7}$$

4 Experiments

We compare DKNP with diverse NP variants such as Conditional NP (CNP)(Garnelo et al., 2018a), NP, Bootstrapping NP (BNP)(Lee et al., 2020), ANP, and Bootstrapping ANP (BANP) on both 1D and 2D regression tasks. For a fair comparison, all NP variants use two paths for context encoding. CNP, BNP, and BANP have two deterministic paths, and NP and ANP have one deterministic and one stochastic path. Also, ANP and BANP have additional self-attention in both stochastic and deterministic paths. Following Lee et al. (2020), BNP and BANP were trained with 50 bootstrap context samples and tested with 4 samples. Note that we mostly followed the same hyperparameter setup used in Lee et al. (2020) and the details are described in Appendix A. Unlike the NP variants (except CNP) which are trained with the lower bound of conditional log-likelihood $\log p(Y_T|X_T, X_C, Y_C)$ or its slight modification, DKNP can directly evaluate it as well as maximize it during training. All results are also reported with the conditional log-likelihood to evaluate the methods' ability to model the given stochastic processes. Following (Le et al., 2018; Lee et al., 2020), we use importance weighting estimation (Burda et al., 2016) with 50 samples to evaluate the performance of NP variants that utilize z.

4.1 1D Regression with Gaussian Processes Data

We first conduct basic evaluation of DKNP in comparison to NP variants by testing its ability to model 1D stochastic process samples generated from Gaussian Processes of diverse kernels, namely RBF, Periodic and Matérn 5/2. In this task, we randomly sampled input x from [-2, 2] and generated y from GP kernels where the hyperparameters were also sampled both during training and testing. The number of context points n_c is sampled from $Unif(3, 25)$ and the target points is

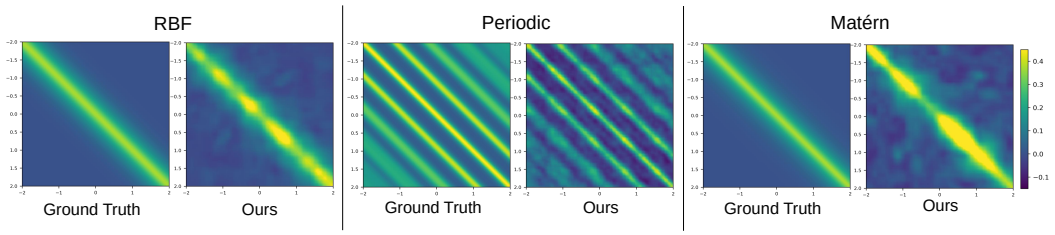

Figure 5: Visualization of the correlation matrices learned from the DKNP compared to the ground truths. We take a Monte Carlo estimate (averaging 1,000 kernel functions of the same type) to visualize the ground truth kernels for multiple hyperparameters at once.

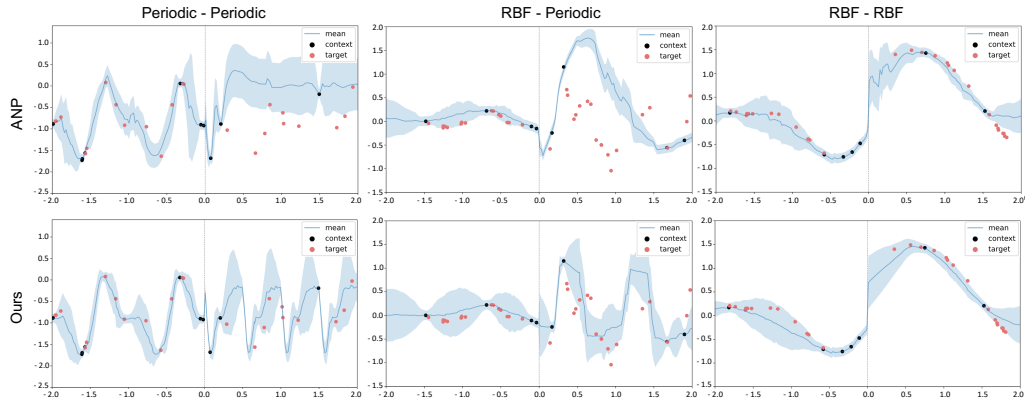

Figure 6: Comparison of predictions by ANP and DKNP. The dark blue line indicates the mean, and the light blue shade indicates one standard deviation range.

$n_c + Unif(3, 50 - n_c)$. Details about the data generation process (kernel hyperparameters), the training process, and the evaluation process are in Appendix B.1.

The results in Table 1 show that DKNP can better model the given stochastic processes compared to various NP variants in terms of the log-likelihood. Note that the attention mechanism dramatically increases all methods' ability to model GP samples, as can be seen by the clear divide between CNP, NP, BNP and the rest. The true merit of DKNP, however, is its capability to better estimate the uncertainty of the stochastic processes thanks to its explicit modeling of the full covariance matrix.

Figure 4, for example, clearly demonstrates the distinguishing feature of DKNP compared to previous NP variants, in this case ANP, which recorded the second highest likelihood in Table 1. In all kernel types, ANP fails to contain the target points within one standard deviation range while DKNP successfully captures the uncertainty in all unobserved points. Specifically, ANP is over-confident when modeling kernel types RBF and Matérn 5/2, where only a local structure exists. For periodic kernels, where both local and global structure exists, ANP fails to capture the underlying periodicity, due to its architecture that tries to estimate functional stochasticity with a fixed-size global latent variable. Unlike ANP, however, DKNP can correctly model the whole stochastic process thanks to its ability to explicitly model the correlations between all output variable.

Employing decoupled paths for modeling mean and covariance as in Figure 2, another advantage of DKNP is improved interpretability, as we can explicitly check the learned prior by visualizing the covariance matrix as described in Section 3. Such example is depicted in Figure 5, where we can readily compare the learned prior with the ground truth prior for all three kernel types. Note that DKNP not only model allows us to visually check the learned prior, but also demonstrate its ability to accurately learn the ground truth kernels for all kernel types.

4.2 1D Regression with Change Point

Next we test all methods in a more challenging setup, where the underlying dynamics of the stochastic process changes at midpoint. Specifically, we employ three different GP configurations: Periodic-

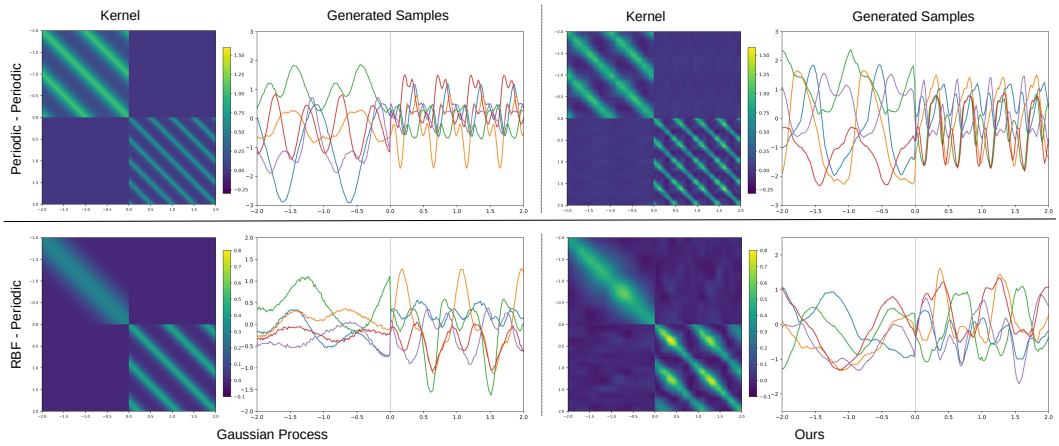

Figure 7: Visualization of the learned correlation matrices and the sampled data for Periodic-Periodic and RBF-Periodic compared to the ground truths.

Table 2: Results (log-likelihood) of the NP variants and DKNP for three different change point configurations.

Methods	Periodic - Periodic	Periodic - RBF	RBF - RBF
CNP	-0.423 ($\pm$0.016)	0.051 ($\pm$0.028)	0.966 ($\pm$0.006)
NP	-0.531 ($\pm$0.022)	-0.007 ($\pm$0.011)	0.872 ($\pm$0.017)
BNP	-0.358 ($\pm$0.022)	0.105 ($\pm$0.023)	0.984 ($\pm$0.011)
ANP	0.602 ($\pm$0.017)	0.883 ($\pm$0.005)	1.142 ($\pm$0.001)
BANP	0.605 ($\pm$0.026)	0.877 ($\pm$0.010)	1.145 ($\pm$0.001)
Ours	**0.905** ($\pm$0.047)	**0.945** ($\pm$0.008)	**1.259** ($\pm$0.000)

Periodic, Periodic-RBF, and RBF-RBF. Each configuration uses one kernel type (with a fixed hyperparamter setup) up to point 0, then another kernel type afterwards. As the correlations between output variables are more complex in this setup, we expect DKNP to demonstrate even more distinguishing performance than NP variants. The training and evaluation processes are the same as the previous experiments in Section 4.1. See Appendix B.1 for data generation details.

The results in Table 2, as expected, show wider gaps between DKNP and NP variants for all three configurations in terms of log-likelihood. Interestingly, unlike the the previous results (Table 1) where DKNP marginally outperformed NP variants for RBF kernels (but beyond one standard deviation margin), the performance gap is more distinguished even for RBF-RBF configuration in this setup. This observation indicates DKNP's ability to better capture complex correlations derived from a combination of rather simpler kernels, and we can predict DKNP will better adapt to even more change points composed of multiple kernel types than NP variants.

Figure 6 demonstrates predictions by ANP and DKNP for all three change point configurations. Compared to ANP which fails to correctly capture the changing dynamics of the given processes (especially for configurations including periodic kernels), DKNP shows its ability to correctly learn the underlying dynamics before and after the change point. In Figure 7, we compare the learned priors as well as the generated samples from them to the ground truths priors and samples for two change point configurations, Periodic-Periodic and RBF-Periodic. Note that DKNP can accurately learn two distinct kernels in both sides of the midpoint, (*i.e.* low and high frequency in the Periodic-Periodic case, smooth and periodic in the RBF-Periodic case) thus able to generate samples that are practically equivalent to the ground truth samples.

4.3 2D Image Completion

The purpose of the 2D image completion task is to test how well each model shows its functional flexibility to learn non-trivial kernels. Assuming that image data is generated by a stochastic process, the task is to regress pixels that are missing based on the provided pixels (*i.e.* context pixels) in the

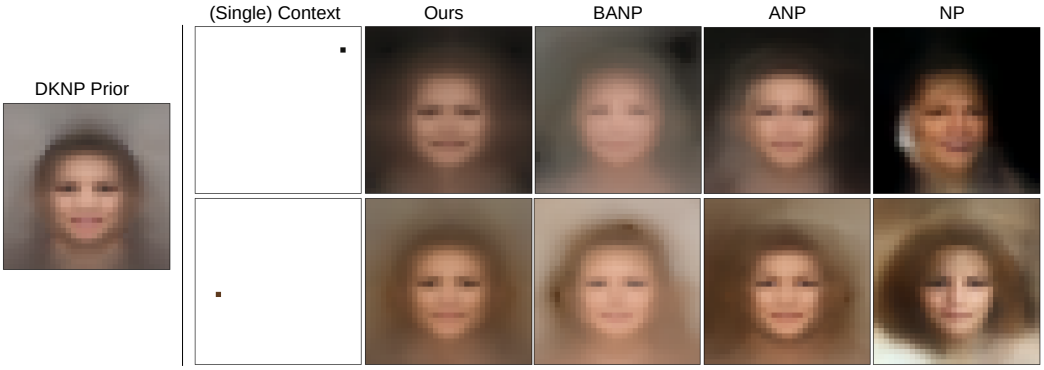

Figure 8: **(Left)** The prior learned by DKNP. **(Right)** Predicted means of various models given single context point shown in first column.

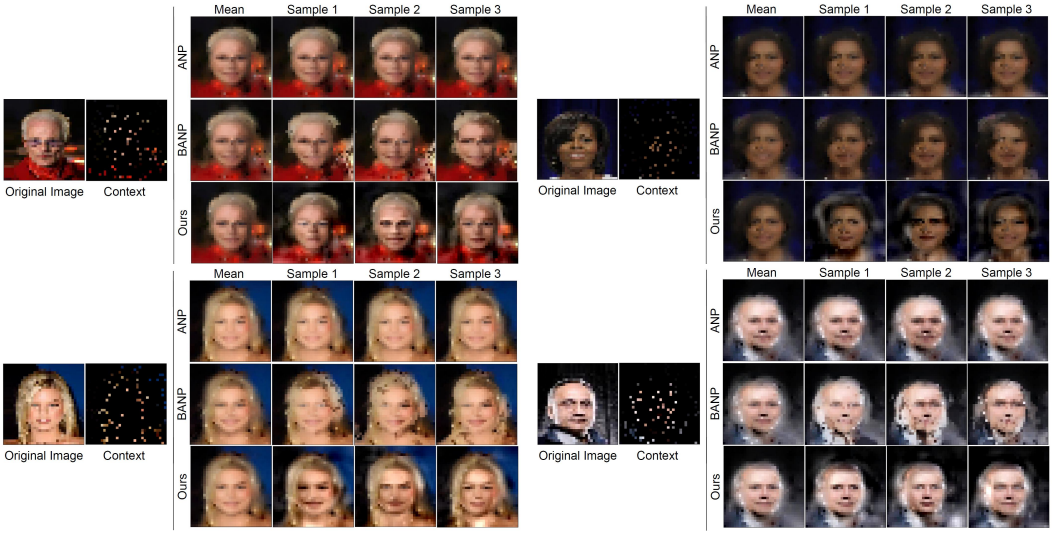

Figure 9: Examples of the image completion task. Based on the original image, roughly 10% of pixels (100 pixels) were given as the context. The filled images generated from DKNP (Ours) are much more diverse and clearer, compared to other baselines.

image. Specifically, we use the CelebA dataset (Liu et al., 2015), which consists of 202,599 number of facial images of celebrities of diverse ethnicities, gender, and age.

For both training and evaluation, the number of context points n_c is sampled from Unif(3, 200) and the target points is n_c + Unif(3, 200 - n_c). Details about the data generation process, the training process, and the evaluation process are in Appendix B.2.

In Table 3, consistent with the previous results, DKNP shows superior ability in modeling even two-dimensional stochastic processes compared to all NP variants. This is somewhat predictable based on two previous results (Table 1 and Table 2) where the performance gap between DKNP and NP variants was more visible for stochastic processes with complex underlying dynamics (where both local and global structure exist), and the fact that the facial images are likely to follow non-trivial dynamics. For example, the dynamics to generate the hair or the background, which are typically low frequency signals, would be quite different from the dynamics to generate the details of the face where eyes, nose, and mouth altogether compose higher frequency signals. And unlike the previous change point data where the boundary between two different dynamics (*i.e.* kernels) is a point in a 1D line, the boundary between different kernels in facial images is a (curved) line in a 2D plane (*e.g.* hairline). Therefore we can expect DKNP to significantly outperform all NP variants in estimating the correlations between output variables, as shown in Table 3.

Figure 8 illustrates the mean of the prior distribution learned by DKNP on the CelebA dataset and the predictive mean of the NP variants and DKNP when a single context point is given in the back-

ground. The posterior mean of DKNP demonstrates distinct behavior for the face and background, such that the given context uniformly affects the background while keeping the face practically intact. This shows that DKNP accurately captures the correlations between output variables by successfully learning kernel functions that behave differently for face and background. BANP, ANP, and NP, on the other hand, demonstrate unpredictable behaviors where the context point in the background also affects the hair or the clothes, or the background is only partially affected by the given context point. This behavior reveals that NPs, which implicitly learn the kernel function, have a harder time modeling the accurate correlation between output variables.

We conclude this section with qualitative examples in Figure 9 that highlight the DKNP's capability to generate diverse samples thanks to its explicit modeling of the covariance. Given 10% of the data points as context, ANP samples are generated by sampling the latent variable z, BANP samples are generated via bootstrapping, and DKNP samples are generated with the predicted mean and covariance. While ANP, BANP and DKNP all produce reasonable predictive means, ANP's generated samples heavily resemble one another, whereas DKNP's generated samples demonstrate considerable diversity such as different genders, facial expressions and hair styles. BANP also demonstrates more varied samples than ANP thanks to its bootstrapping process, but they sometimes contain noisy pixels.

Table 3: Results (log-likelihood) of NP variants and our model in the 2D image completion task.

Methods	Celeb A
CNP	2.280 ($\pm$0.010)
NP	2.374 ($\pm$0.024)
BNP	2.876 ($\pm$0.014)
ANP	3.471 ($\pm$0.005)
BANP	3.627 ($\pm$0.001)
Ours	**3.946** ($\pm$0.027)

5 Related Work

Many advances in NPs are made with the advent of neural network-based stochastic processes such as NPs (Garnelo et al., 2018b) and CNPs (Garnelo et al., 2018a). ANP (Kim et al., 2019) show a dramatic performance gain by leveraging attention mechanisms in the aggregating operation. Also, Le et al. (2018) confirm through extensive empirical evaluation on the design of NPs such as architecture and objective functions that combining attention and NPs improve the predictive log-likelihood marginally. Lee et al. (2020) extends NPs using the bootstrap technique for estimating functional uncertainty without maintaining a latent variable in the NP architecture. This notion of removing the latent variable part is inline with our motivation.

For modeling stationary stochastic processes, Gordon et al. (2020) introduced Convolutional CNP (ConvCNP) where translation equivariance in the data is explicitly encoded in the model. Convolutional NPs (ConvNP) Foong et al. (2020) are natural extension of ConvCNPs where a global latent variable was introduced to model dependencies of the predictive distribution. Gaussian Neural Processes (GNP) (Bruinsma et al., 2021) further generalize these class of models where on top of translation equivariance, captures predictive correlations by directly modeling covariance matrix of predictive distribution. While GNPs share similar motivation to our work, due to requiring translation equivariance, the way covariance matrix is parameterized are quite different (e.g. 1-D, 2-D convolution versus attention) from DKNPs. Moreover, covariance matrix of GNPs depend on Y_T, while for DKNPs, motivated by posterior of GPs, the covariance only depends on the inputs X_C, X_T (see equation 6).

6 Conclusion

We propose a new neural stochastic processes, Decoupled Kernel Neural Processes (DKNPs), that learn an explicit kernel function to better capture the correlation between output variables. By leveraging cross- and mixed attention mechanisms to model an explicit kernel function, DKNPs outperform the concurrent NP variants in terms of predictive likelihood and better global coherence of generated samples. By the novel model architecture of DKNPs, the learned prior can be accessible, which provides a deeper understanding of the underlying distributions of data. As future work, one could consider developing a method to manipulate a learned kernel or to impose a constraint on the kernel learning process of DKNPs with prior knowledge.

[Reproducibility Statement] We utilized the datasets and the baselines that are all available on Github (https://github.com/juho-lee/bnp). As for our proposed model and code, we will make a comment directed to the reviewer and area chairs once the discussion forums are open.

REFERENCES

Wessel Bruinsma, James Requeima, Andrew Y. K. Foong, Jonathan Gordon, and Richard E Turner. The gaussian neural process. In *Symposium on Advances in Approximate Bayesian Inference (AABI)*, 2021.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

Andrew Y. K. Foong, Wessel Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, and Richard E. Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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 (ICML)*, 2018a.

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.

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 (ICLR)*, 2020.

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 (ICLR)*, 2019.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

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.

Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, 2015.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. Association for the Advancement of Artificial Intelligence (AAAI), 2014.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* MIT Press, 2006.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. Sequential neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

A ARCHITECTURES DETAILS

We mostly followed the same model architectures in the paper (Lee et al., 2020). num_det_path and num_det_path indicate the number of deterministic and stochastic paths.

A.1 1D REGRESSION TASK

	CNP	NP	BNP	ANP	BANP	DKNP
dim_x	1	1	1	1	1	1
dim_y	1	1	1	1	1	1
dim_hid	128	128	128	128	128	128
dim_lat	-	128	-	128	-	-
num_det_path	2	1	2	1	2	2
num_stoch_path	0	1	0	1	0	0
enc_v_depth	-	-	-	4	4	4
enc_qk_depth	-	-	-	2	2	2
enc_pre_depth	4	4	4	4	4	4
enc_post_depth	2	2	2	2	2	2
dec_depth	3	3	3	3	3	3

Table 4: Hyperparameter setting for the 1D regression task

A.2 IMAGE COMPLETION TASK

	CNP	NP	BNP	ANP	BANP	DKNP
dim_x	2	2	2	2	2	2
dim_y	3	3	3	3	3	3
dim_hid	128	128	128	128	128	128
dim_lat	-	128	-	128	-	-
num_det_path	2	1	2	1	2	2
num_stoch_path	0	1	0	1	0	0
enc_v_depth	-	-	-	6	6	6
enc_qk_depth	-	-	-	3	3	3
enc_pre_depth	6	6	6	6	6	6
enc_post_depth	3	3	3	3	3	3
dec_depth	5	5	5	5	5	5

Table 5: Hyperparameter setting for the image completion task

B EXPERIMENTAL DETAILS

B.1 1D REGRESSION

Data generation from a single kernel We constructed training and test data generated from GPs with multiple kernels including RBF, periodic kernels, and Matérn 5/2. For each task, we generated $x \sim \text{Unif}(-2, 2)$ and y from GP kernels where the GP hyperparameters are sampled from uniform distributions. For RBF kernel, $k(x, x') = s^2 \exp(-||x - x'||^2/2l^2)$, we sampled $s \sim \text{Unif}(0.1, 1.0)$, $l \sim \text{Unif}(0.1, 0.6)$, and output additive noise $\mathcal{N}(0, 10^{-2})$. Given n, the size of the context C was drawn from $\text{Unif}(3, 47)$ and the size of the target is sampled from $\text{Unif}(3, 50 - n_c)$. For Matérn 5/2 kernel $k(x, x') = s^2(1 + \sqrt{5}d/l + 5d^2/(3l^2)) \cdot \exp(-\sqrt{5}d/l), (d = ||x - x'||)$, we sampled from $s \sim \text{Unif}(0.1, 1.0)$ and $l \sim \text{Unif}(0.1, 0.6)$. For periodic kernel, $k(x, x') = s^2 \exp(-2\sin^2(\pi||x - x'||^2/p)/l^2)$, we sampled from $s \sim \text{Unif}(0.3, 1.0)$, $l \sim \text{Unif}(0.6, 1.0)$, and $p \sim \text{Unif}(0.8, 1.0)$. We trained all models identically for 100,000 steps with training batch size of 100. We used Adam optimizer with initial learning rate of $5 \cdot 10^{-4}$ and decayed using cosine annealing scheme.

Data generation from two kernels Similar to a single kernel case, data points are sampled from a GP but with two types of kernels, RBF and periodic. The half of the points ranging from $x = -2$ and

$x = 0$ were sampled from one kernel and the rest from $x = 0$ to $x = 2$ were from another kernel. We generated samples from three scenarios: Periodic - Periodic, RBF - Periodic, and RBF - RBF. Note that we set the correlation between two kernels being zero. The training details are identical to the single kernel case.

B.2 IMAGE COMPLETION

CelebA32 Similar to 1-D regression tasks, we randomly sampled pixels of a given image at training as targets, and treat a subset of the points as contexts. The size of the contexts and targets is sampled from Unif$(3, 200)$ and n_c+Unif$(0, 200 - n_c)$. For preprocessing, x is rescaled to [-1, 1] and y is rescaled to [-0.5, 0.5]. We trained all models identically for 200 steps with training batch size of 100. We used Adam optimizer with initial learning rate of $5 \cdot 10^{-4}$ with cosine annealing scheme.