Improved Attentive Neural Processes

Srivatsan Srinivasan
SEAS
Harvard University
srivatsansrinivasan@g.harvard.edu

Anirudh Suresh
SEAS
Harvard University
anirudh_suresh@college.harvard.edu

Abstract

Neural Processes (NPs) (Garnelo et al., 2018) set up the regression problem as learning a distribution over regression functions given a set of input-output pairs as context, with each function modeling the distribution of the output given an input conditioned on the context. NPs can learn a wide family of distributions, work with arbitrary context sizes and scale linearly with the number of context points, but under-fit contexts due to mean aggregation of context representations. Attentive Neural Processes (ANPs) (Kim et al., 2019) solve NPs' typical under-fitting of context data by incorporating attention into NPs, creating a query-specific context representation for each input query. Compared to NPs, ANPs boast better prediction accuracy, less training time, and better flexibility in terms of modeling a wider range of functions. In this work, we further propose improvements to ANPs by (1) adding local latents for each context point and generating query-specific local latent representation via attention and (2) implementing self-attention in the decoder. We demonstrate the performance of these improvements in specifically constructed synthetic 1-D regression tasks and a few-shot regression problem for image completion using MNIST data.

1 Introduction

There are two broad learning methodologies for standard regression tasks–an easier and a more standard approach involves modeling the regression problem as learning the distribution of the response variable vector $\mathbf{y}$ given a predictor vector $\mathbf{x}$ via a deterministic class of functions such as neural networks, to which $\mathbf{x}$ is fed as an input. This approach tends toward black-box function approximators that allow for fast test-time predictions (scaling linearly in the number of target points) at the expense of limited interpretability and uncertainty estimation, extreme specialization, and reliance on abundant labeled data in the training phase. The second approach involves learning a distribution over possible functions that can map $\mathbf{x}$ to $\mathbf{y}$ and drawing samples from this distribution at test time whenever a new prediction needs to be made. The latter approach of conducting inference on a stochastic process, while harder to learn, provides two salient advantages. First, learning the posterior can allow us to systematically reason about uncertainties in the regression estimates. Furthermore, the posterior provides a natural way to capture the co-variablity of outputs given inputs. *Gaussian Processes* (GP) (Rasmussen and Williams, 2006), non-parametric models that learn distributions over functions, are one of the most popular choices of such stochastic process models. In addition to providing a full posterior of output predictions for given inputs, GPs are extremely data-efficient and have closed-form analytical solutions which involve a matrix inversion step and the model is invariant to the ordering of the training data. Despite these appealing properties, GPs suffer from computational inefficiency: their traditional form entails cubic scaling with respect to the number of total data points, and sparse approximations are only able to achieve quadratic scaling (Quinonero-Candela and Rasmussen, 2005). Moreover, traditional kernels which are used to express the co-variance structure are limited in terms of their expressiveness and can be quite

Final Project Report, CS282R (Spring-2019), Harvard University.

constrained away from the data, and additional selection and optimization procedures may be required for identification of a proper kernel for a given task (Pillonetto et al., 2014).

Neural processes (NPs) (Garnelo et al., 2018) are a method tailored to combine the computational efficiency of neural networks with the desirable properties of stochastic processes–namely, order invariance of training context data, possibility of learning a distribution over functions and uncertainty estimates, and good performance even with a small number of training points. NPs are particularly powerful in their ability to predict distributions of functions for an arbitrary number of target outputs conditioned on a small number of context input-output pairs. Moreover, NPs allow training on samples drawn from different realizations of a stochastic process and hence find a natural use case in few-shot learning. However, the learning process in NP depends on a single aggregate context embedding (usually mean) from the entire context set, which is hypothesized to serve as a bottleneck, since this representation is invariant to the query (point) the model decodes. In essence, as a result of the embeddings (mean) being the same across all targets, it is harder for the decoder to ascertain the relevance of particular context points to the target point in question. To counter this, Attentive Neural Process (ANP) (Kim et al., 2019) adds self-attention and cross-attention modules to the NP framework in order to account for differential relevance of context points toward particular target points and better learn internal strucutre of context data. The resulting model is better equipped to produce crisper distributional predictions as opposed to collapsing to a mean prediction and outperforms NP with regard to reconstruction of context information.

Despite the advancements of ANP over NP, ANP still assumes a single global latent variable that is parametrized by an aggregation (mean) of output embeddings of the entire context data. Kim et al. (2019) posit that this global latent variable is key toward recognizing any global structure in the task arising out of dependencies between context points. Acknowledging this argument, we continue to retain the global latent variable and rather, add an additional cross-attention mechanism over a set of local latents learned from the context embeddings in order to exploit the differential relevance of the particular local structure of context points with respect to target queries. Additionally, we replace the multilayer perceptron (MLP) in the decoder with a self-attention module, replacing all instances of MLPs in the ANP with attention. Our model, called ANP^{++}, performs at least as well as ANP, if not better, on a set of few-shot regression tasks - different 1-D synthetic regression tasks and 2-D MNIST image completion task. Particularly, the model's predictive performance trumps that of ANP in cases where it is important to capture the varying local structural properties across the domain of each task in order to enhance predictive performance.

2 Background

The first important technical background necessary to understand the role of NPs and our contribution is Gaussian Processes (GPs), and the Appendix contains a brief description of GPs.

2.1 Neural Processes

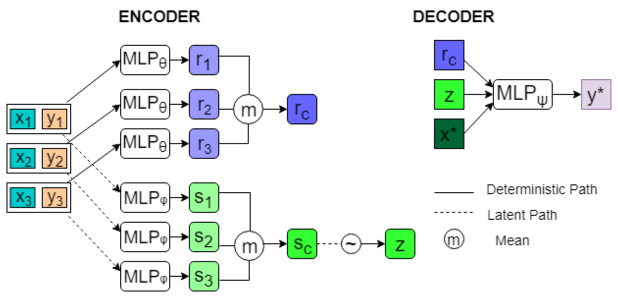

Figure 1: A sketch of the NP model

NP - Neural Processes (Garnelo et al., 2018) combine the desirable predictive properties of GPs while maintaining linear computational complexity in terms of the number of train and test points. Given a set of observed context points $\{(x_i, y_i)\}_{i \in C}$ and a set of target inputs $\{x_j\}_{j \in T}$, NP predicts a distribution over the associated target outputs in a manner that is invariant to the order of the context and target points. Typically, we assume that $C \subseteq T$–that is, the target points we wish to reconstruct encompass the context points we are provided. The encoder in NP consists of two paths–a deterministic path and a latent path. The deterministic path uses an MLP layer to map each $(x_i, y_i) \subset C$ to an embedded representation r_i. The r_is are then aggregated into a single r_C that represents the output of the encoder's deterministic path for the context C. The latent path meanwhile uses another MLP to map each $(x_i, y_i) \subset C$ to an embedded representation s_i. The s_is are similarly aggregated into a single s_C, which parametrizes a distribution for the global latent variable z. In particular, $z \sim \mathcal{N}(\mu(s_C), \sigma(s_C))$, where μ and σ are neural networks that output the mean and standard deviation, respectively, of the distribution of z. The aggregation rule of the r_is and s_is is typically a simple mean.

The decoder in NP consists of another MLP network that maps r_C, z, and the target $\{x_j\}_{j\in T}$ to learn the predictive means and variances of the target outputs - $\{y_j\}_{j\in T}$. Each sample of z serves to achieve a different stochastic realization of the data-generating process that the Neural Process learns to model. This framework, seen in Figure 1, carries several advantages. NP achieves scalable training and inference, scaling linearly with respect to the sum of the number of context and target points. Moreover, NP performs really well in cases where only few context points are available and also performs order-invariant inference from a context set. One downside of the setup of NP–and subsequently a motivation for the ANP model–is the characteristic context reconstruction error that is a by-product of aggregating the embeddings into a single common context representation for all targets and in the process, creating an information bottleneck.

Neural Processes use variational approximations for the latent variable inference and updates the model parameters by maximizing the following ELBO loss (Kim et al., 2019):

$$\log p(Y_T|X_T, X_C, Y_C) \geq \mathbb{E}_{q(z|s_T)}\big[\log p(Y_T|X_T, r_C, z) - KL\big(q(z|s_T)\,||\,q(z|s_C)\big)\big] \quad (1)$$

where $X_C = \{x_i\}_{i\in C}$, $X_T = \{x_j\}_{j\in T}$, $Y_C = \{y_i\}_{i\in C}$, and $Y_T = \{y_j\}_{j\in T}$. $q(z|s_C)$ refers to the prior over z, which is provided by the set of context points via the latent path, and $q(z|s_T)$ is the posterior update to the latent variables on the evidence of target sets. We are motivated here to reconstruct the context set and construct new target outputs with a high likelihood while simultatneously ensuring the consistency between the prior and the posterior update via the KL term. This serves to ensure model generalizability from train time to test time, since no target outputs would be available during test time.

2.2 Attention

Given a set of a set of key-value pairs $\{(k_\ell, v_\ell)\}_{\ell\in\mathbb{L}}$, an attention mechanism uses some notion of similarity (measured by distance metrics) between the query and the keys to decide the weights of each key with respect to the query and aggregate the values with these weights in order to generate a query specific value aggregate. Also, the final attention weights sum to 1 and hence serve as a linear interpolation of the values and thus, the computation of these weights is invariant to the ordering of the key-value pairs.

Attention mechanisms come in many forms. Given n key-value pairs of the form $K \in \mathbb{R}^{n\times d_k}, V \in \mathbb{R}^{n\times d_v}$, and m queries $Q \in \mathbb{R}^{m\times d_k}$, the following are the examples of some common attention mechanisms (same mechanisms used by (Kim et al., 2019)):

- **Laplacian**$(Q, K, V) := WV \in \mathbb{R}^{m\times d_v}$; $W_{ij} = \text{softmax}(-\sum_{r=1}^{d_k} |Q_{ir} - K_{jr}|)$
- **DotProduct**$(Q, K, V) := \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \in \mathbb{R}^{m\times d_v}$
- **MultiHead**$(Q, K, V) := \text{concat}(head_1, \ldots, head_H)W \in \mathbb{R}^{m\times d_v}$
  - $head_h := \text{DotProduct}(QW_h^Q, KW_h^K, VW_h^V)$; $W \in \mathbb{R}^{n\times d_v}$

The Laplacian attention depends on a kernel that uses the ℓ_1 norm of the difference between individual query components and keys to assess similarity, while dot-product attention uses the cosine distance between the queries and the keys in the kernel. Multihead attention (Vaswani et al., 2017) is a parameterized version of attention that uses multiple "heads" of dot-product attention before linearly combining the results of these individual dot-product attentions to arrive at a final set of values.

Self-Attention Self-attention is a special case of attention where the query, key, and value matrices are the same ($Q = K = V$). In such a case, regardless of the exact attention mechanism chosen, the query attends to its own set (the set which contains the keys and the query), and the goal is to learn the dependencies between the inputs themselves and use that information to capture the inherent internal structure (Vaswani et al., 2017). This is in contrast to cross-attention, which refers to the general use of attention where the query, keys, and values are not necessarily the same.

2.3 Attentive Neural Processes

As mentioned above, an Attentive Neural Process (Kim et al., 2019) addresses the issue of underfitting that NPs suffer from, in particular with regard to provided context points. NPs' use of aggregated deterministic path and latent path embeddings that are invariant to the specified target points makes it difficult for the model to recognize the differential relevance that context points may have for the target queries. The ANP model tackles this by integrating attention into the model, seen in Figure 2. The r_is and s_is in the ANP model are computed via application of self-attention to the context set. The self-attention mechanism allows the context set to attend to itself, thereby modeling interactions between the context points and accounting for redundancies and saliencies in the context set.

The deterministic path then replaces the NP model's mean aggregation step with a cross-attention mechanism that sets $Q = X_T, K = X_C, V = \{r_i\}_{i \in C}$. Whereas the deterministic path mean aggregation is removed in favor of cross-attention, the authors do not make the analogous replacement in the latent path. In particular, they continue to use a global latent with the aim of capturing global dependencies and structural properties of the task–thus, the deterministic path becomes specialized toward learning local structure in the context data while the latent path focuses entirely on assessing a larger global structure. The innovation of ANP then lies in the deterministic cross-attention mechanism and its ability to filter how relevant particular context input-output pairs may be for the target query of interest.

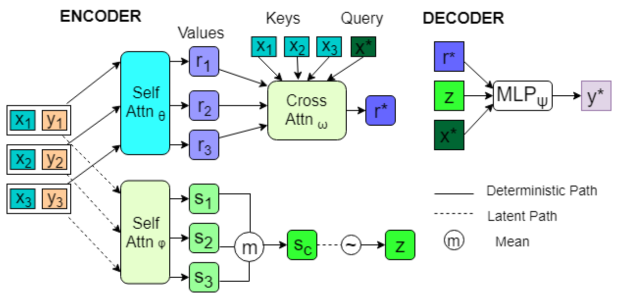

Figure 2: A sketch of the ANP model

3 Methods

3.1 Model: Improvements to the Attentive Neural Process

Remember from the previous section that the global latent variable z in the NP models serves to enforce certain global properties of the task across all target predictions.Each single sample of z is meant to represent the selection of a single function from the distribution of functions that explain the observed data. Hence, one cannot completely dispense away with z in any Neural Process model. Having said that, we believe that finding means to further incorporate attention into the latent path can help to capture local latent structure better. We thus add a cross-attention mechanism into the latent path. Individual s_is are still generated similar to ANP from the context input-output pairs via self-attention, and a mean aggregation is still used to get a single s_C from these s_is: s_C once again parametrizes a distribution from which z is drawn. Simultaneously, each s_i parametrizes a local latent distribution from which a c_i is drawn. Formally, we have $z \sim \mathcal{N}(\mu_z(s_C), \sigma_z(s_C)), c_i \sim \mathcal{N}(\mu_c(s_i), \sigma_c(s_i)) \, \forall \, i \in C$, where μ_z and σ_z are the learned mean and variance parameter functions, respectively, for the global latent and μ_c and σ_c are the learned mean and variance parameter functions, respectively, for the local latents.

Each local latent c_i (local latent for the i^{th} context point (x_i, y_i)) can be thought in a similar vein as the global latent by assuming only that particular context pair (x_i, y_i) was provided as the entire context set–that is, each local latent would be analogous to a global latent if the context set were a singleton set consisting of only the corresponding context point. Intuitively, then, the cross-attention scheme that is applied over the c_is (with X_C serving as the keys and X_T as the query) attends to these local latents and determines the relevance of the target queries to these latent samples. Cross-attention yields a query-specific c_{lat} for each query $x_j \in X_T$, and we pass this output through a network along with z to yield a final local latent path output c^*. We pass this local latent path output along with the global latent path output z to the decoder. We believe that this formulation is especially relevant when the tasks have strong local structures (captured by the c_is) in addition to a general global structure. Using $\mathcal{A}(k, q, v)$ to indicate an attention module and NN to refer to a feed-forward net, we have

$$c_j^* = \text{NN}(\ \mathcal{A}(X_C,\ x_j^*,\ \{c_i\}_{i=1}^{|C|}),\ z),\ r_j^* = \mathcal{A}(X_C,\ x_j^*,\ \{c_i\}_{i=1}^{|C|})$$

The ANP^{++} decoder is also endowed with self-attention, amending the MLP-only ANP decoder. This change is motivated by the idea that the self attention module can force the decoder to adjust to the the decoded targets' internal dependencies with respect to each other similar to the Transformer model (Vaswani et al., 2017). The self-attention mechanism takes as its input a representation of the concatenation of $x_j \in X_T, r^*, c^*$, and z. A sketch of the ANP^{++} model can be seen in Figure 3.

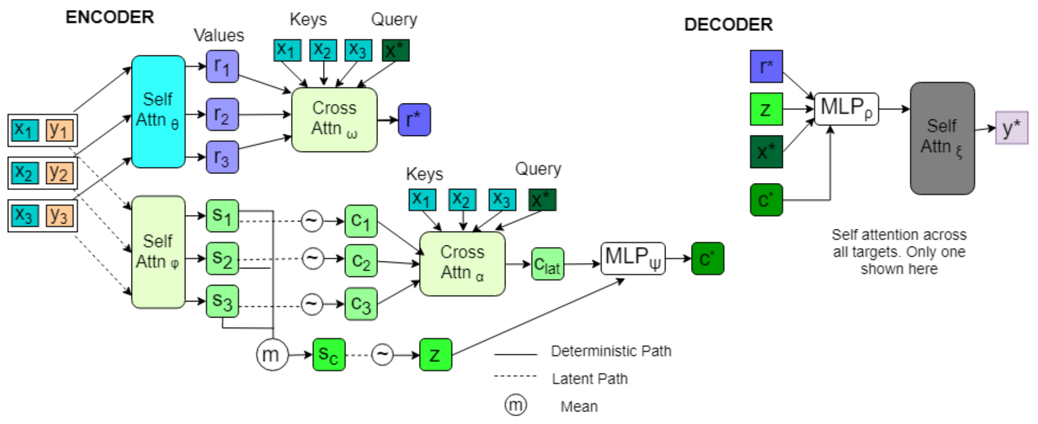

Figure 3: A sketch of the ANP^{++} model

3.2 Loss and Inference

Using Gaussian variational approximations for the latent variable inference, one can derive the ELBO loss function for the ANP^{++} model provided in Equation 2. One can easily observe the similarities between this loss term and the NP loss function (Equation 1) or the ANP loss function (Kim et al., 2019) and notice the addition of more KL terms owing to the local latents.

$$\underbrace{\mathcal{L}}_{\text{ELBO}} = \underbrace{R}_{\text{(likelihood)}} + \underbrace{G}_{\text{(global KL)}} + \underbrace{L}_{\text{(local KL)}}$$

$$\begin{aligned} R &= \mathbb{E}_{q(z, \{c_i\}_{i=1}^{|C|} \mid s_T)} \left[\sum_{j=1}^{|T|} \log p(y_j | x_j, r_j^*, c_j^*, z; \theta) \right] \\ G &= D_{KL}\ \big(\ q(z|s_T; \phi)\ ||\ q(z|s_C; \phi)\ \big) \\ L &= \sum_{i=1}^{|C|} D_{KL} \big(\ q(c_i|s_T; \lambda)\ ||\ \underbrace{q(c_i|s_i; \lambda)}_{\substack{\text{local latent for} \\ \text{the } i^{th} \text{ context}}}\ \big) \end{aligned} \tag{2}$$

The original ANP loss (Kim et al., 2019) has only the first two components–reconstruction and global latents–$R + G$ (and that too, without the need for local latents c_is in the reconstruction term). Also, it is important to note that all latent variable parametrizations are from the Gaussian family.

$$\begin{aligned} q(z|s_C;\phi) &= \mathcal{N}(\mu(s_C), \sigma(s_C);\phi) \\ q(z|s_T;\phi) &= \mathcal{N}(\mu(s_T), \sigma(s_T);\phi) \\ q(c_i|s_T;\lambda) &= \mathcal{N}(\mu(s_T), \sigma(s_T);\lambda) \\ q(c_i|s_i;\lambda) &= q(c_i|s_C;\lambda) = \mathcal{N}(\mu(s_i), \sigma(s_i);\lambda) \end{aligned}$$

Inference is done via simple backpropagation and we use an Adam optimizer in all our models. We use the re-parameterization trick to generate latent variable samples while still maintaining the computation graph fully differentiable. (Kingma and Welling, 2013).

4 Related Work

Gaussian Processes The key advantage of regression learning with GP is the model's ability to incorporate similarity between points through the co-variance structure using kernel functions. While NPs do not have this implicit ability, ANPs learn the similarity between points in their domain via the attention mechanisms that are a part of the model. Also, the choice of kernel is arguably a detrimental factor in the training of GPs, whereas all members of the neural process family learn the kernels directly from the context data. On the other hand, GPs are more principled in the sense that their posterior predictive covariances and marginal variances can be expressed in a fully analytical form, while the family of neural processes has no such guarantees about the learned distributions. There are other works that learn data generating processes with algorithms that lie somewhere in the spectrum between neural networks and Gaussian Processes. Variational implicit processes (Ma et al., 2018), for instance define the stochastic process with a similar decoder and a finite dimensional latent setup as NP but ultimately learn the posterior with a GP approximation. Similar to NPs, matching networks (Vinyals et al., 2016) and deep kernel learning (Wilson et al., 2015) are two other methods that extract representations directly from data, but both pass these representations to an explicit kernel and perform learning in a GP framework. Along the spectrum from neural networks to GPs, the neural process family resides closer to neural networks in terms of inference and computational complexity but possess an ability to learn a distribution over functions.

Meta-Learning Meta learning models share the fundamental motivations of NPs as they shift some workload from training time to test time. Assuming we are given input-output pairs from any function at test time, NPs have the ability to probabilistically reason about this function conditioning on the given input-output pairs, making them ideal candidates for few-shot density estimations. On one hand, few-shot classification has received extensive focus from the research community, including attention based models in the last few years, such as Chen et al. (2019); Jimenez Rezende et al. (2016); Reed et al. (2017). On the other hand, few-shot pairwise regression is comparatively less explored (e.g. Bachman et al. (2018)). In this work, we use attention mechanisms to learn the similarity between domain points in the few-shot pairwise regression setting ,building on top of the works of Garnelo et al. (2018); Kim et al. (2019) with the addition of cross attention across local latents and a more expressive decoder compared to ANP.

Conditional Latent Variable Models One can group all the models similar to the family of neural processes under the canopy of conditional latent variable models. Such models learn the conditional distribution $p(X_T|X_C, z)$ where z is a latent variable that is sampled to generate different stochastic realizations. The well-known conditional VAEs (Kingma and Welling, 2013) are an example of this class of models which use a conditional input apart from a latent variable sample in order to generate an output. A more sophisticated version of CVAE that uses a heirarchical latent structure with global and local latents is the Neural Statistician (Edwards and Storkey, 2016), which uses z to capture global uncertainty, but instead of representing a distribution over possible functions, it learns simply an unconditional distribution over the sets in the input space. The NS does not generate different y values conditioned on inputs in a pairwise setting like a GP or NP but rather generates different samples x using the global latent z and additionally sampling local latents z_T. Notably, unlike in NPs, the prior of the global latent variable in the NS is not conditioned on z. Another variant of the NS model is the variational homoencoder (Hewitt et al., 2018) , which has a similar hierarchical structure as (Edwards and Storkey, 2016) but uses a separate subset of data points for the context and predictions. In this family, neural processes have their own standing due to their ability to perform more targeted sampling of the latent distribution using x in a regression setting, while some of the other models described earlier have been demonstrated to be useful only in classification settings. An ability to do this kind of targeted sampling has potential interesting applications — e.g. conditional generation and completion tasks (image, natural language etc.) that do not fit fully well into the framework provided by other models. The distinction across the family of neural processes as conditional latent variable models has already been extensively discussed in Section 2.

Attention and Transformer Models The idea of attention in sequence-to-sequence models has been extensively pursued as a mechanism to aggregate value representations based on the similarity between a query and a set of keys that correspond to the values Bahdanau et al. (2014); Kim et al. (2017); Xu et al. (2015). Transformers (Vaswani et al., 2017) use a combination of self-attention and cross-attention modules for sequence transduction, thus creating a fully attention based model, dispensing with the usual recurrent encoders and decoders used for this purpose. Image transformers (Parmar et al., 2018) extend this work to a sequence-modeling formulation of image generation with a tractable likelihood. In this work, we leverage self-attention modules in both the encoder and the decoder and two cross-attention modules between the encoder and the decoder, producing (just) the model architecture similar to the transformer models, yet applied in entirely different settings.

5 Experiments and Results

5.1 1-D Regression on data generated by a GP

The first few-shot regression experiments we run are on synthetic 1-D data generated from a GP with varying kernel parameters for each task (curve). This experiment is performed in order to benchmark ANP against ANP^{++} and to assert that whether ANP^{++} is able to effectively learn a wider class of functions. We also experiment with two different GP kernels–squared exponential (SE) and periodic (PER), defined as follows:

$$k^{SE}(x,x') = \sigma^2 \exp\big(-\frac{(x-x')^2}{2l^2}\big)$$
$$k^{PER}(x,x') = \sigma^2 \exp\left(\frac{-2\sin^2\big(\pi\frac{x-x'}{p}\big)}{l^2}\right)$$

In all these experiments, except the one in which we compare different attention mechanisms, we use multihead attention, which is the best-performing mechanism according to Kim et al. (2019). The experiment setup is similar to that of Kim et al. (2019) to offer clear comparisons: for each task (curve) in these experiments, the targets are 400 evenly spaced inputs in the range $[-2, 2]$. The contexts are chosen randomly over the same range, and the number of context points is chosen uniformly between 3 and 50 while the number of targets remain fixed at 400. During training time, the kernel parameters (length scales and variance) are chosen randomly for each training batch, ensuring that the model trains over a diverse class of functions (tasks). We terminate the process at 100,000 iterations and measure the context and target NLL on test batches of $m = 16$ curves. The results can be seen in Figure 4. Context NLL and target NLL quantify the likelihood of observing y conditional on x and the latent samples drawn and can be formally defined as

$$\text{Context NLL} = -\frac{1}{|C|}\sum_{i=1}^{|C|}\log(p(y_i \mid x_i, z, r_i^*, c_i^*)$$
$$\text{Target NLL} = -\frac{1}{|T|}\sum_{i=1}^{|T|}\log(p(y_i \mid x_i, z, r_i^*, c_i^*)$$
$$\text{where} \quad \text{Global latent} \;\; z \sim q(z|s_C;\phi)$$
$$\text{Local latent} \;\; c_i \sim q(c|s_i;\lambda) \;\; i = 1, 2, \ldots, |C|$$

From the plots in Figure 4, we see that ANP^{++} typically shows a faster decrease in reconstruction error and lower loss at convergence. Overall, one can see that ANP^{++} marginally outperforms ANP in terms of both context and target likelihoods, meaning that it is both able to reconstruct the contexts and, at the same time, generalize across the target points during test time even with the limited context examples that it has seen in the past. This difference in performance for the kernel hyperparameter settings provides evidence that the model is at least as expressive, if not more, when compared to the baseline ANP in terms of learning a wide range of functions.

Different attention mechanisms We also experiment with different attention mechanisms along the lines of Kim et al. (2019) but with the ANP^{++} model. The results (seen in Table 1) are similar to the original ANP results in that multihead performs better than dot-product, which in turn performs better than Laplacian. Similar to Kim et al. (2019), we attribute the success of multi-head attention to the multiple heads that are able to smooth out the dot-product interpolations better and hence lead to better predictive performance than simple dot-product attention.

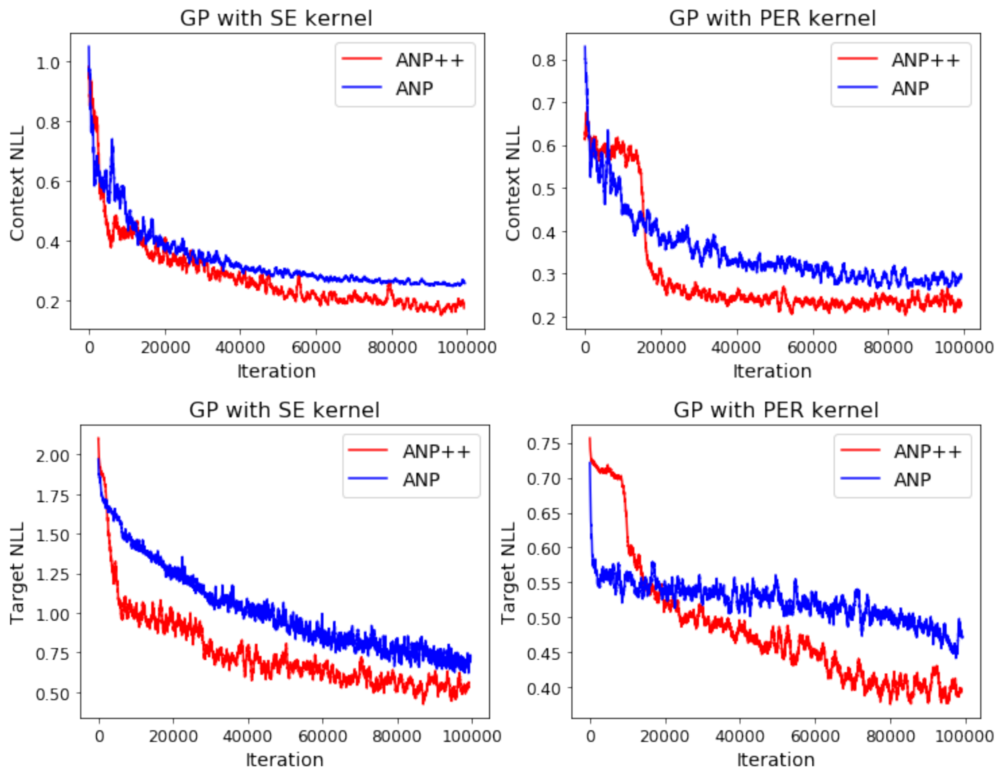

Figure 4: Context and Target Negative Log Likelihood for regression with GP data (the closer to 0, the better)

Attention	Context NLL	Target NLL
Laplacian	0.32	0.81
Dot Product	0.23	0.56
Multihead	**0.18**	**0.49**

Table 1: Impact of Attention Mechanisms in ANP^{++}

5.2 1-D Regression on data generated by non-stationary sines

In this experiment, we test the models on points from synthetic 1-D curves which have varying local structures across different portions of the curve. As one such example, we run experiments on the data sampled from a non-stationary linear combination of sine functions.

$$y(x_i) = \sum_{k=1}^{K} w_k(x_i) \sin(\gamma_k \cdot \pi \cdot x_k)$$

In this specific formulation, the weights of the linear combination are a function of the position x, and thus, we hypothesize that learning global statistics with a global latent vector z alone may not sufficiently capture this position-specific linear combination. All the experiment settings (context, target sizes, metrics, etc.) are the same as in the experiments with GPs. Figure 5 shows the context and target NLL for ANP and ANP^{++} under this setting. In this experiment, ANPs still perform reasonably well because of the deterministic path (the r_i embeddings) that accounts for local information to some extent. Yet, it is obvious from the likelihood plots that ANP^{++} performs better context reconstruction and target prediction compared to ANP. We believe the addition of the local latents and query-specific local-latent combination via cross-attention allow the model to learn how the local structure at the query position x relates to the local structure at each context point. For example, given two sine functions that are linearly combined with a simple weight scheme (e.g. $w(x) = [1, 0]\, \forall x < 0, w(x) = [0.5, 0.5]\, \forall x > 0$), the model should be able to learn that $x < 0$ should focus more on the local structure around negative context points and less so on positive context points. As a sanity check, Figure 6 shows one sample output of the ANP and ANP^{++} in which we see that our model has better predictive performance and similar context reconstructive performance as the ANP model.[1]

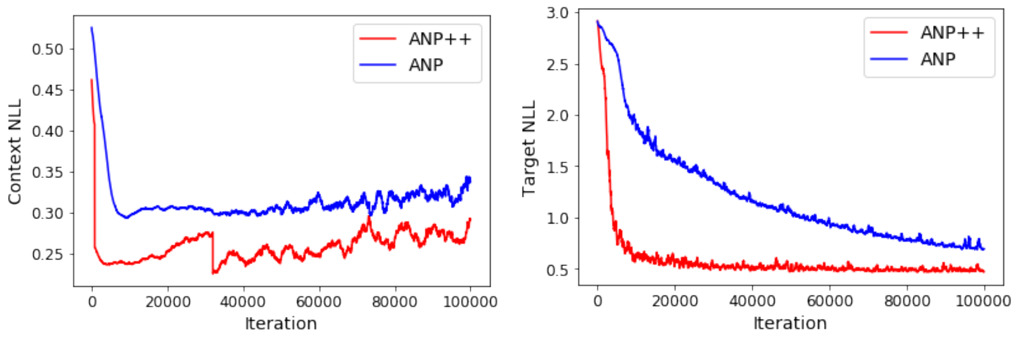

Figure 5: Context and Target Negative Log Likelihood for regression with non-stationary sines data (the closer to 0, the better)

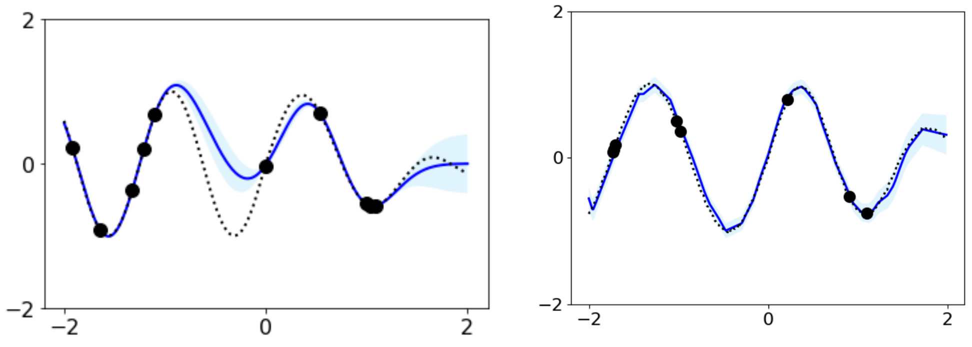

Figure 6: Sample curve generated from ANP (left) and ANP^{++} (right)

[1]Due to our software implementation, in which each model runs its own synthetic data generation step every time it is trained, it is hard to exactly standardize the context generation process across different models. Hence, it is difficult to share specific examples (same context points and same curves) to compare ANP^{++} performance against ANP learning. Having said that, better target and context NLL indicate better performance of ANP^{++} on average compared to ANP.

Besides these experiments, we also run simple tests to see whether our query-specific cross-attended latent learns a different representation based on the position of the query. To do so, we set a fixed weight function–$w(x) = [1, 0]\,\forall\, x < 0, w(x) = [0.5, 0.5]\,\forall\, x > 0$–for each curve in our training data, collect the query-specific local latent representation learned in the ANP^{++} model, and compare it with samples of the global latent representation z that is used by ANP. Figure 7 shows a 2-dimensional (t-SNE) projection of these weights using 400 randomly picked query points across multiple test curves. We observe that the latent representations produced by ANP^{++} are more distinct (at least on this lower-dimensional projection) for $x > 0$ versus $x < 0$ when compared with the global latent samples from ANP (which are not specific to the query position). To round off this batch of experiments, we also run a simple classification of query-specific latent samples into two classes: $x < 0$ and $x > 0$, and a simple supervised classifier is able to better distinguish the two sets of samples produced by ANP^{++} than ANP. These toy experiment results ascertain that the latent representations learned by the ANP^{++} model are different when the local structure changes as function of the position whereas the ANP's global latent learns some notion of a global average.

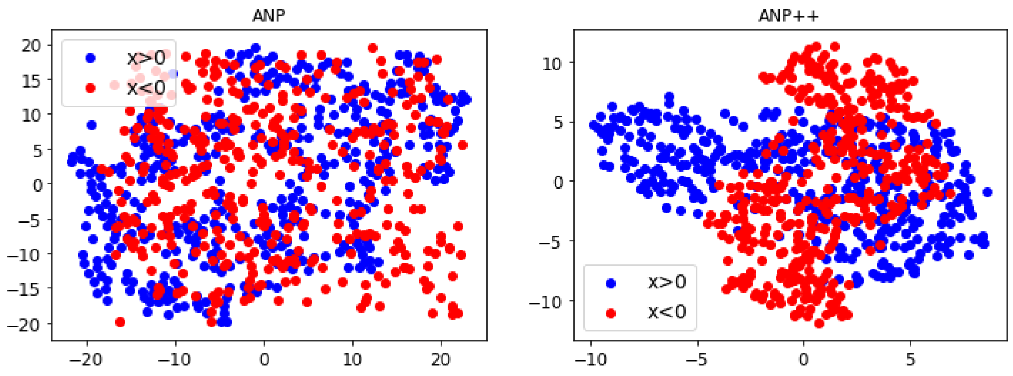

Figure 7: A t-SNE visualization of the latent variables used in the decoder for ANP(z) and ANP^{++} (c^*). The latent variable plotted is the global z in the ANP model (left) and the query-specific cross-attended local latent representation c^* in the ANP^{++} (right). The plots indicate that ANP^{++} learns different latents for points $x < 0$ and $x > 0$. Recall that in this example, the weights for the two sine periods are different for $x < 0$ and $x > 0$.

5.3 2-D Regression with MNIST

To see if model performance translates from a synthetic dataset to a real dataset, we also run an 2-D image completion task with MNIST data. MNIST data provides us two advantages: it is easily trainable with a simple feed-forward net, and the properties of the data are well-known. This problem can be thought of as a pairwise regression problem between the 2-dimensional position x_i and the corresponding pixel value y_i. Figure 8 shows some realizations of our model for different numbers of context points. We see that the model is capable of performing few-shot regression well, even with very few context points ($|C| = 10$). As a sanity check, we also test the model's reconstructive ability when the entire context is provided ($|C| = 784$) and we see that the output realizations almost entirely resemble the input contexts.

Another factor to ascertain in the ANP^{++} is whether the global latent variable z provides diverse stochastic realizations of the data-generating process. To do so, we generate samples with 10 context points as inputs and generate decoded samples with a different global latent variable z sampled each time. Figure 9 shows that the model predicts different digits for the same input context. This example indicates that the latent dimensions of the global latent variable z are expressive enough to induce decoding of diverse samples with limited context information[2].

[2]Appendix contains some more samples of the actual curves (as some form of sanity check) learned in all the three experiments

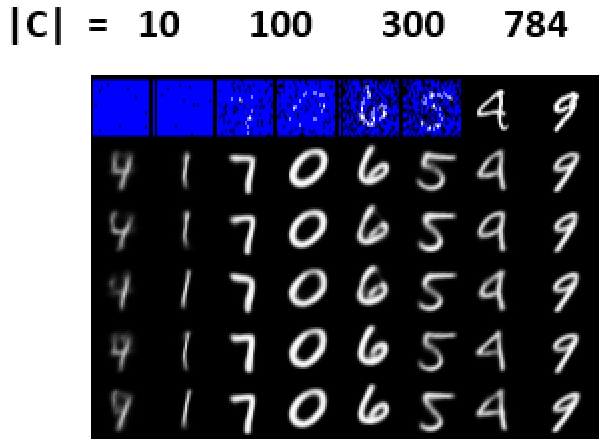

Figure 8: MNIST image completion results for different numbers of context points. The top row of the table indicates the context passed to the model. The samples in each column are generated by the predicted means and standard deviations of the model. Note that each column has been sampled with a single global latent vector z.

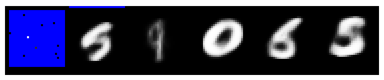

Figure 9: MNIST image completion results for the same context, but with different realizations of the global latent z.

6 Discussion

Time Complexity The significantly low computational complexity at inference time is what distinguishes NPs from GPs (linear versus quadratic/cubic). Yet, as witnessed both in this work and Kim et al. (2019), attention mechanisms (both self- and cross-attention) are necessary to prevent NPs' drastic under-fitting of context points. One downside of the incorporation of attention into the NP framework is the supralinear scaling–assuming n_C context points and n_T "different" target points (members of $T\backslash C$), the cross-attention mechanism in the deterministic path entails $n_C + n_T$ queries attending to n_C key-value pairs. The complexity of this operation is $O(n_C(n_C + n_T))$; with the ANP^{++}, the self-attention in the decoder entails a slightly increased complexity of $O((n_C + n_T)^2)$. Therefore, we lose the desirable linear scaling of NPs; however, this quadratic scaling is not a major concern as the attention operations are all some form of matrix multiplications which could be massively parallelized with a GPU. Thus, the actual runtime (wall-clock) does not climb so dramatically in practice. Furthermore, our experiments indicate that the more sophisticated model ($ANP^{++} > ANP > NP$) converges in a fewer number of iterations, thus offsetting any time complexity overheads caused by these attention mechanisms.

NP vs. GP We have already established the fact that ANPs are more flexible (besides being computationally more efficient) compared to GPs and that they implicitly learn a “kernel" from the data. On the other hand, these advantages come at the cost of the of any neural process being just an approximation of the conditionals of the true stochastic process. In the 1-D synthetic experiments with a GP data, we also compared the learned uncertainties of the neural process models with respect to the original data generating Gaussian Process. A commonality we observe (largely by visual inspection of the plots) is that all the neural processes (NP, ANP, ANP^{++}) have lower variance estimates for the targets than the posterior variance of the GP for the same context points. In other words, the NPs seem to be overconfident at the target points compared to the data generating Gaussian Process. We believe that this could be explained by a few reasons which need further investigation: (1) the perceived overconfidence is an artifact of Gaussian variational approximations and performing backpropagation with reparameterization — similar to the observations in a BNN trained with Bayes by Backprop (Blundell et al., 2015); (2) the similarity among the different training tasks (curves) could be perceived by the model to be high enough and this could lead to this overconfidence.

Marginal Contributions of the components Both the ANP and ANP^{++} have a deterministic and a latent path with ANP^{++} having an additional set of local latents. Since ANP^{++} also learns latent representations for each of the context points c_i, the deterministic embeddings learned for the context points r_i are expected to have a significant informational overlap with the learned local latents. To verify this, we ran a quick last-minute performance check by turning off the deterministic path in the 1-D synthetic data experiment with a squared-exponential kernel, and the context NLL and target NLL numbers seem fairly similar with and without the deterministic embeddings r_i. This observation means that the marginal value of deterministic embeddings and local latents needs to be investigated more systematically in future work. Neither the original ANP work (Kim et al., 2019) nor our work does any systematic ablation study in order to understand the incremental value each component (self-attention in deterministic path, self-attention in latent path, self-attention in the decoder, cross-attention in deterministic path, cross-attention in latent path). Besides removing redundant components and trimming computational overheads, such studies are also necessary to determine the real scope of our work and find fitting real-world applications.

Extensions and Future Work Our work in this paper provides few very initial steps at further improving the existing ANP model and the initial experiments show promise to an extent. Having said that, there is huge scope for future investigation about the properties of ANP^{++}. Beyond the simple 1-D GP and sine curve experiments, more targeted experiments should continue to be conceived in order to build stronger evidence regarding this proposed advantage of the novel features of ANP^{++}. One other key missing element in our work is the absence of experiments to understand the marginal effects of our two additions - self attention in decoder and cross attention across local latents. Further experiments need to be designed and executed to study the individual contributions of each of these changes as they both cause computational overheads to the model. In terms of real-life experiments on image completion, we wish to run the model on larger datasets such as CelebA and also, run experiments to check how the model behaves when the context points are not provided at random, but rather based on a predetermined pattern - for e.g. top half of the image or one half a time series data. Furthermore, one can also think about applications of the ANP^{++} model toward other domains–natural language (e.g. sentence completion) and reinforcement learning (e.g. transfer applications, exploration in bandits and RL) are two such categories of problems for which these neural process models could be a good fit [3].

7 Conclusion

We introduce ANP^{++}, an iteration on the vanilla ANP that i) further incorporates an attention module into the latent path in order to better capture local structure of the tasks and ii) improves the expressiveness of the decoder with self-attention. We show that ANP^{++} performs at least as well as vanilla ANP on various synthetic tasks, and we demonstrate specific experiments in which ANP^{++} outperforms ANP as a result of better accommodating local structure in the data. In a nutshell, ANP^{++} shows faster learning (in terms of number of iterations), better context reconstruction and target prediction likelihood compared to ANP.

Acknowledgements

We wish to thank the CS 282R course staff for their support and advice throughout the course. We especially want to thank David Belanger for his involved guidance as well as suggestions on various experiments, application areas, and future directions.

[3] In the appendix, we also describe another specific application that could use the uncertainty estimates learned by NPs to model the acquisition costs for collecting new samples of data based on the samples observed till now

References

Bachman, P., Islam, R., Sordoni, A., and Ahmed, Z. (2018). VFunc: a Deep Generative Model for Functions. *arXiv e-prints*, page arXiv:1807.04106.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, page arXiv:1409.0473.

Betancourt, M. (2018). A Conceptual Introduction to Hamiltonian Monte Carlo. *arXiv e-prints*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight Uncertainty in Neural Networks. *arXiv e-prints*.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. (2019). A Closer Look at Few-shot Classification. *arXiv e-prints*, page arXiv:1904.04232.

Edwards, H. and Storkey, A. (2016). Towards a Neural Statistician. *arXiv e-prints*, page arXiv:1606.02185.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Whye Teh, Y. (2018). Neural Processes. *arXiv e-prints*, page arXiv:1807.01622.

Hernandez-Lobato, J. M. and Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *arXiv e-prints*.

Hewitt, L. B., Nye, M. I., Gane, A., Jaakkola, T., and Tenenbaum, J. B. (2018). The Variational Homoencoder: Learning to learn high capacity generative models from few examples. *arXiv e-prints*, page arXiv:1807.08919.

Jimenez Rezende, D., Mohamed, S., Danihelka, I., Gregor, K., and Wierstra, D. (2016). One-Shot Generalization in Deep Generative Models. *arXiv e-prints*, page arXiv:1603.05106.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Whye Teh, Y. (2019). Attentive Neural Processes. *arXiv e-prints*, page arXiv:1901.05761.

Kim, Y., Denton, C., Hoang, L., and Rush, A. e. M. (2017). Structured Attention Networks. *arXiv e-prints*, page arXiv:1702.00887.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114.

Ma, C., Li, Y., and Hernández-Lobato, J. M. (2018). Variational Implicit Processes. *arXiv e-prints*, page arXiv:1806.02390.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., and Tran, D. (2018). Image Transformer. *arXiv e-prints*, page arXiv:1802.05751.

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., and Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50:657–682.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression.

Ranganath, R., Gerrish, S., and Blei, D. M. (2013). Black Box Variational Inference. *arXiv e-prints*.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*.

Reed, S., Chen, Y., Paine, T., van den Oord, A., Eslami, S. M. A., Rezende, D., Vinyals, O., and de Freitas, N. (2017). Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions. *arXiv e-prints*, page arXiv:1710.10304.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv e-prints*.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching Networks for One Shot Learning. *arXiv e-prints*, page arXiv:1606.04080.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2015). Deep Kernel Learning. *arXiv e-prints*, page arXiv:1511.02222.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv e-prints*, page arXiv:1502.03044.

A Background : Gaussian Processes

A Gaussian process (GP) is a type of stochastic process–it is a collection of random variables such that every finite subset of those random variables has a multivariate Gaussian distribution. The GP itself has a joint distribution over all random variables in the collection, and if there are infinitely many random variables in the collection, the GP is a distribution over functions with a continuous domain. GPs are often formulated in terms of a prior and a likelihood, assuming white noise, and the posterior and posterior predictive can be computed analytically. The GP prior for a real process f over a set of inputs X can be written as

$$p(f(X)) \sim \mathcal{GP}(\mu(X), k(X))$$

where μ is the prior mean function and k is the prior covariance/kernel function, meaning $k(X)$ is a $n \times n$ matrix where n refers to the number of inputs in X. The kernel function can be selected from a variety of choices depending on the expected shape of or dependencies within the process (Pillonetto et al., 2014). Given a Gaussian likelihood assumed to demonstrate observation white noise with variance of σ^2_{obs}, the likelihood takes the form $p(Y|X, f) \sim \mathcal{N}(f(X), \sigma^2_{\text{obs}} I_n)$. The GP posterior then takes the form

$$p(f|X, Y) \sim \mathcal{GP}(\mu', k')$$
$$\mu'(x) = k(x, X)\big[k(X) + \sigma^2_{\text{obs}} I_n\big]^{-1} Y$$
$$k'(x, x') = k(x, x') - k(x, X)\big[k(X) + \sigma^2_{\text{obs}} I_n\big]^{-1} k(X, x'),$$

and, given n' inputs in X', the posterior predictive proceeds as follows:

$$p(Y'|X', X, Y) \sim \mathcal{N}(\mu'(X'), k'(X') + \sigma^2_{\text{obs}} I_{n'})$$

GPs are often used in regression tasks with smaller train and test datasets, for they offer several advantages over neural networks and other black-box approximators. For one, they conduct lazy learning, meaning training is conducted at inference time as opposed to distinctly before inference as in the case of neural networks. A natural result of their lazy learning procedure is order/permutation invariance of the train and test data: regardless of how the train and test data is ordered, the outputted distributions for the test inputs will be the same. This is in contrast to a neural network, for which order of provision of the data in ante-inference training impacts the outputted predictions. Additionally, GPs' distinction between a mean output and a variance via the kernel function for each prediction yields a natural notion of uncertainty in the outputted distribution, whereas traditional neural networks only provide point estimates. Bayesian neural networks present one method of imbuing neural network architecture with a distributional output over functions but are forced to employ training procedures that suffer from either costly computational scaling (Betancourt, 2018) or extreme variance of results across runs (Blundell et al., 2015; Hernandez-Lobato and Adams, 2015; Ranganath et al., 2013). On a related note, GPs are fairly flexible in cases of low training data, naturally providing adequate uncertainty in regions without much training data. Yet in spite of these appealing properties, GPs suffer from supralinear runtime (cubic in the number of points) scaling as well as quadratic storage cost in the number of train and test points; this arises as a result of the covariance matrix inversion in the posterior. These computational costs limit the scalability of traditional GPs and motivate the search for substitutes. Even the more state-of-the-art approximate inference methods for GPs which have good predictive performance scale at least quadratically.

B Future Extension Proposal : Data Collection Application

On a more detailed note, we wish to investigate the applicability of our model to the following specific problem (significantly drawing from the inputs of David Belanger in our discussions) : suppose we have a device that can collect data for a signal at fixed, evenly spaced intervals. In particular, at each collection timestep, we know the cost of collection and can decide whether or not to collect. Given that we have collected data for some signal f at some previous timesteps and that our goal is to accurately predict the value of the signal at a later timestep t, the goal is to determine whether the cost of collecting data at a certain intermediate time t' is worth the estimation benefit afforded by having that data. Using the framework of the family of neural processes, we are able to compute the posterior of $f(t)$ conditional on just the past measurements and to compute the marginal posterior of $f(t)$ conditional on the past measurements *and* the information that a measurement is made at t' (but without knowing the value of $f(t')$). The latter involves marginalizing over the value of $f(t')$, which can be done by drawing samples for $f(t')$ from the neural process and computing the posteriors for $f(t)$ given these samples for $f(t')$. This problem fits the meta-learning glove well, given the need to adapt to a new signal based off an arbitrary number of context points, and will fit in the requirements of neural processes very well.

C ANP^{++} Graphical Model

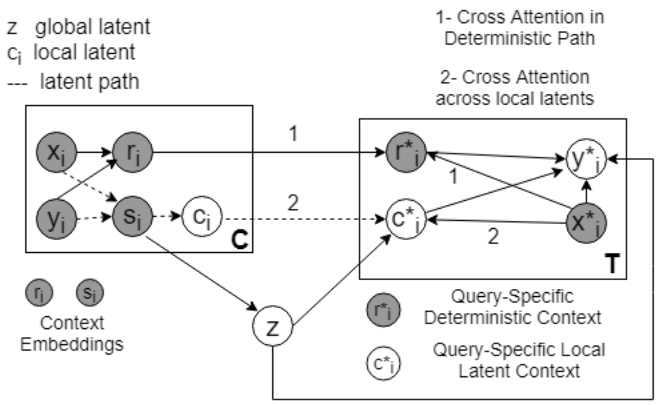
z global latent
c_i local latent
--- latent path
1- Cross Attention in Deterministic Path
2- Cross Attention across local latents
x_i
r_i
y_i
s_i
c_i
C
1
2
r*_i
y*_i
c*_i
x*_i
T
z
Context Embeddings
Query-Specific Deterministic Context
Query-Specific Local Latent Context

Figure 10: A sketch of the underlying graphical model for the ANP^{++}

D More samples for sanity checks

D.1 Gaussian Processes

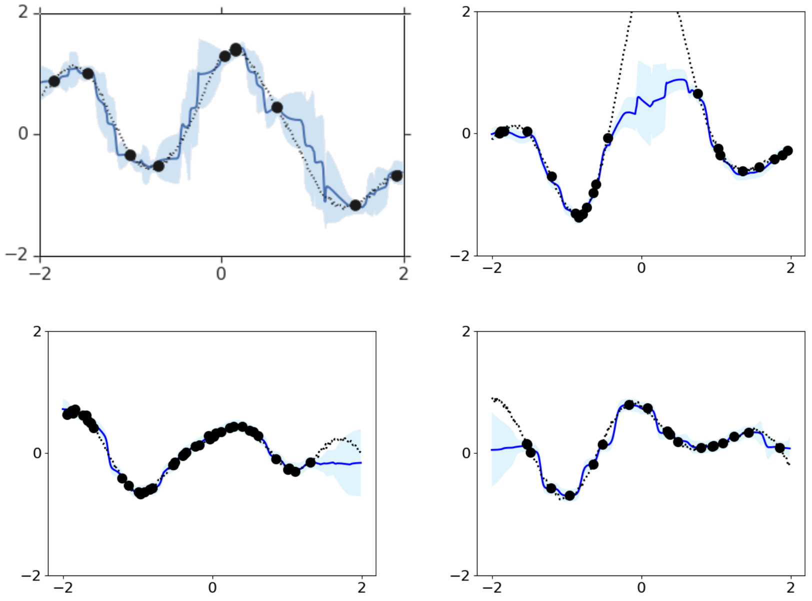

Figure 11: Sanity check : Some sample predictions of ANP^{++} (top) and ANP (bottom)

D.2 MNIST

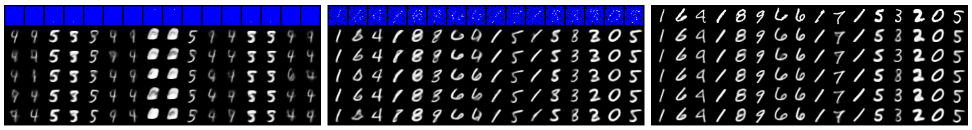

Figure 12: Sanity check : Some sample predictions for ANP^{++} with MNIST data for 0, 100 and 784 context points. In each column, the model uses the same latent z variable (one single sample).

D.3 Periodic NS Sines

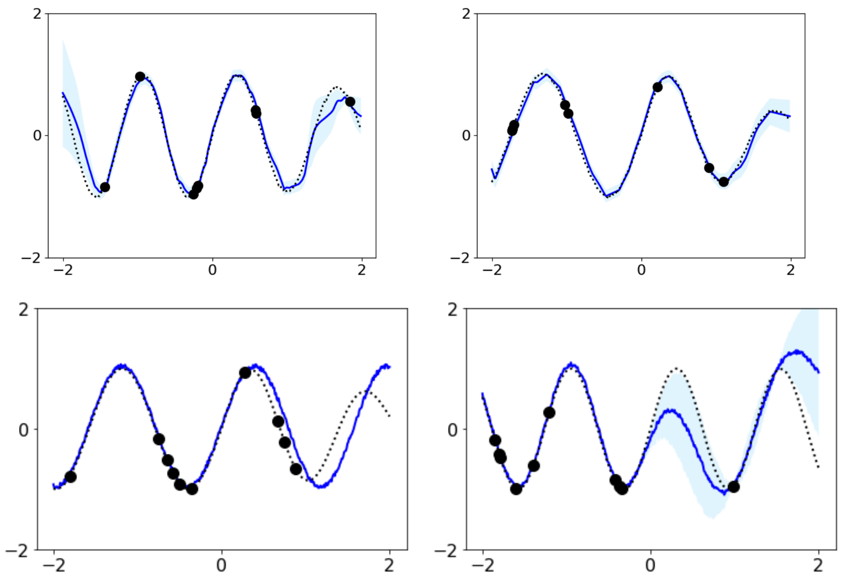

Figure 13: Sanity check : Some sample predictions of ANP^{++} (top) and ANP (bottom)