Residual Neural Processes

Byung-Jun Lee,[1] **Seunghoon Hong,**[1] **Kee-Eung Kim**[1,2]
[1]School of Computing, KAIST, Republic of Korea
[2]Graduate School of AI, KAIST, Republic of Korea
bjlee@ai.kaist.ac.kr, {seunghoon.hong, kekim}@kaist.ac.kr

Abstract

A Neural Process (NP) is a map from a set of observed input-output pairs to a predictive distribution over functions, which is designed to mimic other stochastic processes' inference mechanisms. NPs are shown to work effectively in tasks that require complex distributions, where traditional stochastic processes struggle, e.g. image completion tasks. This paper concerns the practical capacity of set function approximators despite their universality. By delving deeper into the relationship between an NP and a Bayesian last layer (BLL), it is possible to see that NPs may struggle in simple examples, which other stochastic processes can easily solve. In this paper, we propose a simple yet effective remedy; the Residual Neural Process (RNP) that leverages traditional BLL for faster training and better prediction. We demonstrate that the RNP shows faster convergence and better performance, both qualitatively and quantitatively.

Introduction

Inferring with stochastic processes, such as Gaussian Processes (GPs), provides a powerful probabilistic learning framework. Despite its computational cost, it is still widely used due to the unique strengths that usual function approximators are not equipped with. One important strength is that they do not require a costly training phase of parameters: after tuning a small set of hyper-parameters, GPs can be directly applied to any set of observations to infer the posterior distribution of functions.

Neural Processes (NPs) (Garnelo et al. 2018a; 2018b) are a novel attempt to achieve such strength using the combination of neural network function approximators. It is defined by two components: a permutation invariant encoder that processes query-observations pairs (which are called *contexts*) and yields an approximate posterior of function embeddings, and a decoder that takes the function embedding and query point as inputs and yields the prediction on the query point (which is called *targets*). Training is done by feeding the model with random *contexts* and *targets* from random functions and maximizing the lower bound of the log probability of the predictive distribution. Attentive Neural Processes (ANPs) (Kim et al. 2019) are an improvement over NPs by adopting the attention mechanism to be more flexible and accurate, at a certain computational cost. After the training, an NP incurs prediction with computational cost linear in the number of the contexts (quadratic in case of an ANP), which is favorable over a GP which requires a cubic computational cost.

The success of NPs is partially due to a recently blooming line of research on the set function approximators, based on the permutation invariant neural networks such as Deep Sets (Zaheer et al. 2017; Bloem-Reddy and Teh 2019). Nevertheless, given the finite capacity, it has been reported that not every permutation invariant continuous function is reasonably well approximated. Wagstaff et al. (2019) argues that even if the function approximators, after and before the sum pooling, are flexible enough, the dimension of the embedding to be summed should be at least the cardinality of the input set to represent any permutation invariant function with neural networks. In other words, when the embedding dimension is not large enough to match the input cardinality, Deep Set based architectures can approximate only some of the permutation invariant functions (e.g. averaging function). In practice, the function approximators after and before the sum pooling are typically not flexible enough, and Deep Set fails for some of the tasks even though the embedding dimension is larger than the input set cardinality (Murphy et al. 2019). These imply the importance of building a task-specific set function approximator, as ANPs improved from NPs by a large margin.

In this paper, we delve further into the structural similarity between the ANPs and traditional stochastic processes, namely the Bayesian last layer (BLL) (Calandra et al. 2016; Weber et al. 2018; Harrison, Sharma, and Pavone 2018), as ANPs are designed to mimic BLL's behavior efficiently. It turns out that the self-attention layers in an ANP are not expressive enough, and simple cases where the underlying functions lie in the space spanned by a feature extractor, which can be exactly modeled with BLL, might not be efficiently learned. This motivates us to extend the NP to explain the residual part of the prediction that the BLL cannot model, which we call *Residual Neural Processes*. This also allows us to improve the variety of function samples by borrowing the idea of kernel learning, i.e. learning the approximate posterior of feature functions with the implicit distribution. We show that such extension improves convergence speed and asymptotic performance significantly.

Background

The task that we are handling throughout this paper is regression in the context of meta-learning setting, although the method is not limited to regression tasks. A regression problem can be defined as approximating a mapping f from input variables $\mathbf{x} \in \mathbb{R}^{d_x}$ to continuous output variables $\mathbf{y} \in \mathbb{R}^{d_y}$, i.e. $\mathbf{y} = f(\mathbf{x})$. Given the finite training samples of input and output variables, *contexts* $\mathbf{X}_C, \mathbf{Y}_C := \{\mathbf{x}_c\}_{c\in C}, \{\mathbf{y}_c\}_{c\in C}$, one should predict *target* outputs $\mathbf{Y}_T := \{\mathbf{y}_t\}_{t\in T}$ from $\mathbf{X}_T := \{\mathbf{x}_t\}_{t\in T}$. With a slight abuse of notations, we will write $f(\mathbf{X}_C) = \{f(\mathbf{x}_c)\}_{c\in C}$ and $f(\mathbf{X}_T) = \{f(\mathbf{x}_t)\}_{t\in T}$.

Using a stochastic process, a prior distribution over function values $p_\theta(f(\mathbf{X}_C), f(\mathbf{X}_T)|\mathbf{X}_C, \mathbf{X}_T)$ is defined. With a likelihood $p_\theta(\mathbf{Y}_C, \mathbf{Y}_T|f(\mathbf{X}_C), f(\mathbf{X}_T))$, a posterior predictive distribution $p(\mathbf{Y}_T|\mathbf{Y}_C, \mathbf{X}_T, \mathbf{X}_C)$ can be computed. Meta-learning with a stochastic processes then will be the task of finding out the best θ in the prior and likelihood, with the predefined task generator, i.e. $(\mathbf{X}_C, \mathbf{Y}_C, \mathbf{X}_T, \mathbf{Y}_T) \sim G$.

On the other hand, if we only have to learn how to map the input and the output of the inference, the training task can be alternatively defined by a black box that gives the conditional predictive distribution $p_\theta(\mathbf{Y}_T|\mathbf{Y}_C, \mathbf{X}_T, \mathbf{X}_C)$. Similarly, the best θ should be found given the task generator G. The subscript θ will be omitted afterward for the sake of brevity.

Bayesian Last Layer (BLL)

A classical way of meta-learning with a stochastic process would be the *Bayesian last layer* (BLL) method (Weber et al. 2018; Harrison, Sharma, and Pavone 2018), which fits into the former way of defining the task. Using the fact that the last layers of many neural networks are usually (generalized) linear models, a straightforward way of building a flexible stochastic process is to set a Gaussian prior on the weights of the last layer.

After the training phase with random functions (or maximizing the marginal likelihood of a fixed data set), previous layers are treated as a fixed feature function $\phi(\mathbf{x}) \in \mathbb{R}^{d_h}$, and the closed form predictive distribution can be inferred conditioned on an arbitrary context set since the inference is equivalent to the Bayesian linear regression $y = \mathbf{w}^\top\phi(\mathbf{x})+\epsilon$ (when $d_y = 1$ for simplicity), where $\mathbf{w}$ is the weights of the last linear layer. It is also equivalent to a GP with specific kernel choice, which is called a *manifold GP* (Calandra et al. 2016). The posterior inference of BLL can be specified as:

$$
\begin{aligned}
p(\mathbf{w}) &= \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{I}), \quad p(\epsilon) = \mathcal{N}(\epsilon; 0, \sigma_N^2), \\
p(\mathbf{w}|\mathbf{X}_C, \mathbf{Y}_C) &= \mathcal{N}(\mathbf{w}; \mathbf{m}_w, \mathbf{S}_w), \\
\mathbf{m}_w &= \mathbf{\Phi}_C(\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\mathbf{Y}_C, \\
\mathbf{S}_w &= \mathbf{I} - \mathbf{\Phi}_C(\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\mathbf{\Phi}_C^\top, \\
p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{Y}_C, \mathbf{X}_C) & \\
&= \mathcal{N}(\mathbf{Y}_T; \mathbf{m}_w^\top\mathbf{\Phi}_T, \mathbf{\Phi}_T^\top\mathbf{S}_w^{-1}\mathbf{\Phi}_T + \sigma_N^2\mathbf{I}), \quad (1)
\end{aligned}
$$

where we denote the feature matrices of the context and the target input sets by $\mathbf{\Phi}_C = [\phi(\mathbf{x}_{c_1}), \phi(\mathbf{x}_{c_2}), ...]_{c_1,c_2,...\in C}$ and $\mathbf{\Phi}_T = [\phi(\mathbf{x}_{t_1}), \phi(\mathbf{x}_{t_2}), ...]_{t_1,t_2,...\in T}$.

The main difference between the BLL and the Bayesian linear regression is that the BLL trains the feature function $\phi(\cdot)$ as well, maximizing the model evidence $\mathbb{E}_{\mathbf{Y}_C,\mathbf{X}_C}[\log p(\mathbf{Y}_C|\mathbf{X}_C, \phi(\cdot))]$, while it is usually fixed in a linear regression. In a meta-learning context, we can either maximize the log probability of the predictive distribution Eq. (1) with a random context and target set sampled from the task generator G.

Due to the matrix inversion, the computation of the predictive distribution takes $O(|C|^2 d_h + |C|d_h^2 + \min(d_h^3, |C|^3))$, where the minimum depends on whether we invert $|C|\times|C|$ matrix or $d_h \times d_h$ matrix using the Woodbury matrix identity.

Neural Processes (NPs)

Recently, Garnelo et al. (2018a) proposed a novel methodology to the abovementioned problem: learning the whole inference procedure using neural networks. To mimic stochastic processes, two main characteristics of general probability distributions are encoded into the neural architecture for $p(\mathbf{Y}_T|\mathbf{Y}_C, \mathbf{X}_T, \mathbf{X}_C)$:

- *Exchangeability*: $p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_2, \mathbf{x}_1)$
- *Consistency*: $p(\mathbf{x}_1) = \int p(\mathbf{x}_1, \mathbf{x}_2)d\mathbf{x}_2$

To achieve *exchangeability*, a permutation invariant neural network as Deep Set (Zaheer et al. 2017) is adopted to be invariant to the ordering of the contexts and the targets. While summarizing contexts into a finite dimensional vector $\mathbf{r}_C := r(\mathbf{X}_C, \mathbf{Y}_C) \in \mathbb{R}^{d_h}$, to make the summarizing function $r(\cdot)$ permutation invariant, each context $(\mathbf{x}_c, \mathbf{y}_c)$ is fed as the input to the neural network $g(\cdot)$, and its outputs are aggregated by taking an average to form the context embedding $\mathbf{r}_C$:

$$\mathbf{r}_C = \frac{1}{|C|}\sum_{c\in C} g(\mathbf{x}_c, \mathbf{y}_c).$$

In the deterministic version of NPs (Garnelo et al. 2018a), the conditional predictive distribution is directly modeled as $p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C) = p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{r}_C)$. It is then modeled as a factorized Gaussian across the targets $(\mathbf{x}_t, \mathbf{y}_t)_{t\in T}$ for *consistency*:

$$p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{r}_C) = \prod_{t\in T}\mathcal{N}(\mathbf{y}_t|\mu(\mathbf{x}_t, \mathbf{r}_C), \mathrm{diag}[\sigma(\mathbf{x}_t, \mathbf{r}_C)]^2), \quad (2)$$

where both $\mu(\cdot)$ and $\sigma(\cdot)$ are the functions of $\mathbf{r}_C$ and $\mathbf{x}_t$ modeled by neural networks. The NP is then optimized by maximizing the log predictive probability of targets given contexts,

$$\mathcal{L} = \mathbb{E}_{\mathbf{X}_C,\mathbf{Y}_C,\mathbf{X}_T,\mathbf{Y}_T}[\log p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{r}_C)].$$

Garnelo et al. (2018b) later proposed to extend the deterministic NP with latent variables to enable *global sampling* of functions. It introduces the *stochastic* context embedding $\mathbf{z} \in \mathbb{R}^{d_h}$ that provides implicit stochasticity to the posterior of functions. Following the common choice of latent variable generative models in amortized variational inference, $\mathbf{z}$ is modeled by the factorized Gaussian, with statistics (mean

and variance, denoted as $\mathbf{s}_C$) being permutation invariant using the function $r(\cdot)$,

$$p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C) \approx \int p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z})q(\mathbf{z}|\mathbf{s}_C)d\mathbf{z},$$
$$\mathbf{s}_C = \frac{1}{|C|}\sum_{c\in C} g(\mathbf{x}_c, \mathbf{y}_c),$$
$$q(\mathbf{z}|\mathbf{s}_C) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{s}_C), \text{diag}[\sigma(\mathbf{s}_C)]^2).$$

Observation likelihood $p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z})$ is parameterized as in Eq. (2) where $\mathbf{z}$ replaces $\mathbf{r}_C$. The model is learned by maximizing the following approximated ELBO,

$$\begin{aligned}&\log\ p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C)\\ &\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{s}_{TC})}[\log p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z})] - D_{KL}(q(\mathbf{z}|\mathbf{s}_{TC})\|p(\mathbf{z}|\mathbf{s}_C))\\ &\approx \mathbb{E}_{q(\mathbf{z}|\mathbf{s}_{TC})}[\log p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z})] - D_{KL}(q(\mathbf{z}|\mathbf{s}_{TC})\|q(\mathbf{z}|\mathbf{s}_C))\end{aligned}$$

via the reparametrization trick (Kingma and Welling 2014). The main strength of NPs compared to other stochastic processes is their linear time complexity in the number of contexts, i.e. $O(|C|d_h^2)$, when performing the prediction.

Attentive Neural Processes

It turns out that NPs tend to underfit severely, i.e. not being able to accurately predict, even at context points. Kim et al. (2019) hypothesize that this is due to the bottleneck of fixed-length global summary $\mathbf{r}_C$ (or $\mathbf{z}$), whereas there are infinitely many potential targets $\mathbf{x}_t$ to be predicted. They draw inspiration from the locality of GPs, where the kernel forces the prediction $\mathbf{y}_t$ to be necessarily close to $\mathbf{y}_c$ when $\mathbf{x}_t$ is close to $\mathbf{x}_c$ and propose Attentive Neural Processes (ANPs) that implement the locality via the attention mechanism.

In an ANP, the summary of a context $\mathbf{r}_C$ is no longer global, and there exists a local summary $\mathbf{r}_{t|C}$ per each target. This is made possible by the attention mechanism, which takes key, query, and value as its input. In this case, the attention mechanism computes similarities between the target input embeddings $\mathbf{\Phi}_T$ (queries) and the context input embeddings $\mathbf{\Phi}_C$ (keys), and aggregates context embeddings $\{g(\mathbf{x}_c, \mathbf{y}_c)\}_{c\in C}$ (values) with these similarities to predict the embedding corresponding to the target.

By denoting N key-value pairs arranged as matrices as $K \in \mathbb{R}^{N\times d_h}, V \in \mathbb{R}^{N\times d_v}$, and M queries as $Q \in \mathbb{R}^{M\times d_h}$, the (scaled) *dot-product* attention (Graves 2012), one of the most widely used attention mechanisms, can be written as:

$$\textbf{DotProduct}(Q, K, V) = \text{softmax}(QK^\top/\sqrt{d_h})V$$

The *Multi-head* attention (Vaswani et al. 2017) is an extension that linearly transforms keys, values, and queries, and then applies dot-product attention in each head. It tends to yield smoother predictions than the dot-product attention by performing ensemble over multiple dot-product attention:

$$\textbf{MultiHead}(Q, K, V) := \text{concat}(\text{head}_1, ..., \text{head}_H)W^O$$
$$\text{where head}_i := \textbf{DotProduct}(QW_i^Q, KW_i^K, VW_i^V).$$

Furthermore, ANPs adopt the *self-attention* instead of the simple neural network $g(\cdot)$, i.e. an attention with $Q = K = V$, particularly effective for obtaining a richer representation by modeling interactions among context points. Higher-order interactions can be also modeled by simply stacking multiple alternating self-attention layers and dense layers, as in *transformer* (Vaswani et al. 2017).

BLL	$\mathbf{R}_C = (\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\left[\mathbf{Y}_C, \mathbf{\Phi}_C^\top\right]$ $\mathbf{r}_{t\|C} = \phi(\mathbf{x}_t)^\top\mathbf{\Phi}_C\mathbf{R}_C$ $p(y_t\|\mathbf{x}_t, \mathbf{Y}_C, \mathbf{X}_C)$ $= \mathcal{N}\left(\mathbf{r}_{t\|C}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}, \left(\phi(\mathbf{x}_t)^\top - \mathbf{r}_{t\|C}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\right)\phi(\mathbf{x}_t)\right)$
ANP	$\mathbf{R}_C = \text{SA}([\mathbf{X}_C, \mathbf{Y}_C])$ $\mathbf{r}_{t\|C} = \frac{1}{\sqrt{d_k}}\text{softmax}(\phi(\mathbf{x}_t)^\top\mathbf{\Phi}_C)\mathbf{R}_C$ $p(y_t\|\mathbf{x}_t, \mathbf{Y}_C, \mathbf{X}_C)$ $= \mathcal{N}\left(\mu(\mathbf{r}_{t\|C}, \mathbf{x}_t), [\sigma(\mathbf{r}_{t\|C}, \mathbf{x}_t)]^2\right)$

Table 1: Comparison of the conditional predictive distributions of a Bayesian Last Layer (top) and a single-headed Attentive Neural Process (bottom).

ANPs use *both* the deterministic path $\mathbf{r}_{t|C}$ and the stochastic path $\mathbf{z}$. It gets one context dependent summary of contexts $\mathbf{r}_{t|C}$ and one global summary of contexts $\mathbf{s}_C$ by

$$\mathbf{r}_{t|C} = \textbf{MultiHead}(\phi(\mathbf{x}_t), \mathbf{\Phi}_C, \text{SA}([\mathbf{X}_C, \mathbf{Y}_C])),$$
$$\mathbf{s}_C = \frac{1}{|C|}\sum_{c\in C}[\text{SA}([\mathbf{X}_C, \mathbf{Y}_C])]_c, \quad (3)$$

where $\text{SA}([\mathbf{X}_C, \mathbf{Y}_C]) \in \mathbb{R}^{|C|\times d_h}$ are trainable self-attention networks. Note that the feature extractor $\phi(\mathbf{x}_c) \in \mathbb{R}^{d_h}$ is adopted to perform attention over diverse features of inputs. To summarize, the conditional predictive distribution of an ANP is given by:

$$p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C) \approx \int p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z}, \mathbf{r}_{t|C})q(\mathbf{z}|\mathbf{s}_C)d\mathbf{z},$$
$$p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z}, \mathbf{r}_{t|C}) = \prod_{t\in T}\mathcal{N}(\mathbf{y}_t|\mu(\mathbf{x}_t, \mathbf{r}_{t|C}, \mathbf{z}), \text{diag}[\sigma(\mathbf{x}_t, \mathbf{r}_{t|C}, \mathbf{z})]^2),$$
$$q(\mathbf{z}|\mathbf{s}_C) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{s}_C), \text{diag}[\sigma(\mathbf{s}_C)]^2). \quad (4)$$

By adopting the attention mechanisms, ANPs are shown to be less affected by the finite dimension bottleneck, yielding much more accurate predictions than BLLs or NPs. However, using the self-attention network leads to an increased prediction time complexity, $O(|C|^2d_h + |C|d_h^2)$, which is asymptotically equivalent to that of BLLs.

Residual Neural Processes

Despite the asymptotically equivalent time complexity, we find in practice that ANPs outperform BLLs with the same bottleneck width d_h, and the performance gap gets larger as

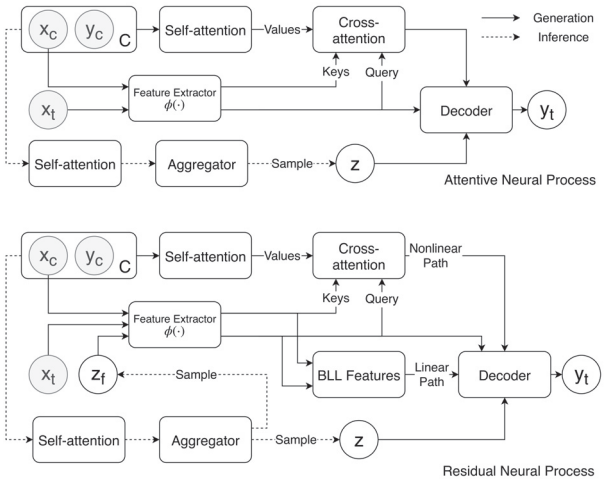


Figure 1: Model architectures of an Attentive Neural Process (top) and an Residual Neural Process (bottom)

the problem gets more complex. Where does the additional modeling power come from?

By considering the most basic form of ANP (deterministic path-only, $d_y = 1$ and $H = 1$), it can be easily observed that ANPs and BLLs exhibit a structural similarity (see Table 1). Both summarize contexts into the matrix $\mathbf{R}_C$ of $|C|$ rows, and compute the similarities between target/context input embeddings $\phi(\mathbf{x}_t)^\top \mathbf{\Phi}_C$ to build target-dependent context embeddings $\mathbf{r}_{t|C}$. It is then fed into a function along with the target input to yield the predictive mean/variance.

One major difference is that the context summary of a BLL is linear in $\mathbf{Y}_C$ whereas that of an ANP is not. Although the conditional mean of a BLL can be the best loss minimizer among functions in the feature space by the representer theorem, the capacity of the feature space becomes a bottleneck due to the cubic complexity $O(d_h^3)$ in the inversion operation. Consequently, having a conditional mean that is non-linear in $\mathbf{Y}_C$ is evidently helpful to increase performance without increasing d_h.

Nevertheless, considering the fact that Deep Set based architectures including the self-attention network turned out to be not omnipotent (Murphy et al. 2019; Wagstaff et al. 2019), there are also drawbacks in learning flexible nonlinear prediction respect to $\mathbf{Y}_C$ since the optimal summary $\mathbf{R}_C$ can be very difficult to learn. Even in the cases when functions lie in the space spanned by feature function so that BLLs can perfectly predict, the self-attention network is required to learn how to solve a linear system. While the scaling $\frac{1}{\sqrt{d_k}}$ and the softmax transformation over the similarities help to fix output scale of $\mathbf{R}_C$ independent of $|C|$ or d_h, it can be easily shown empirically that the self-attention network cannot yield a reasonably approximate solution to an arbitrary linear system: We tried to train a *self-attention* network such that $SA(\mathbf{\Phi}_C) = \hat{\mathbf{\Phi}}_C^{-1} \approx (\mathbf{\Phi}_C^\top \mathbf{\Phi}_C)^{-1}\mathbf{\Phi}_C^\top$. It ended up with $\frac{1}{|C|^2}\|\mathbf{I} - \hat{\mathbf{\Phi}}_C^{-1}\mathbf{\Phi}_C\|_F^2 \geq 0.6$ when $[\mathbf{\Phi}_C]_{ij} \sim \mathcal{N}(0,1)$, where the off-the-shelf pseudo-inverse library achieved an error of $1e^{-12}$. Stacking *self-attention* layers multiple times or adjusting parameters $\{d_h, |C|\}$ did not help improve the error.

Motivated from the above, for better summarization of contexts $\mathbf{R}_C$, it can be naturally proposed to train ANP to predict a nonlinear residual of a context summary respect to $\mathbf{Y}_C$. Instead of explicitly learning the residual as in ResNet (He et al. 2016), we compute the exact linear contexts summary–BLL statistics–and additionally feed it to the conditional predictive distribution to make the network more flexible. It is also expected to learn a good feature function faster since the gradients from the exact linear path avoids a backpropagation through a large number of parameters of the self-attention network. We hence name our model a *Residual Neural Process* (RNP), defined by

$$
\begin{aligned}
\mathbf{R}_C =& \text{SA}([\mathbf{X}_C, \mathbf{Y}_C]),\\
\mathbf{r}^{nl}_{t|C} =& \textbf{MultiHead}(\phi(\mathbf{x}_t), \mathbf{\Phi}_C, \text{SA}([\mathbf{X}_C, \mathbf{Y}_C])),\\
\mathbf{r}^{l}_{t|C} =& \begin{bmatrix} \phi(\mathbf{x}_t) \\ \mathbf{\Phi}_C(\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\mathbf{Y}_C \\ \phi(\mathbf{x}_t) - \mathbf{\Phi}_C(\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\mathbf{\Phi}_C^\top\phi(\mathbf{x}_t) \end{bmatrix},\\
p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z}, \mathbf{r}^{nl}_{t|C}, \mathbf{r}^{l}_{t|C}) =& \qquad (5)\\
\prod_{t\in T} \mathcal{N}(\mathbf{y}_t; &\mu(\mathbf{x}_t, \mathbf{z}, \mathbf{r}^{nl}_{t|C}, \mathbf{r}^{l}_{t|C}), \text{diag}[\sigma(\mathbf{x}_t, \mathbf{z}, \mathbf{r}^{nl}_{t|C}, \mathbf{r}^{l}_{t|C})]^2).
\end{aligned}
$$

where the $\mathbf{r}^{l}_{t|C}$ term above is carefully designed to adopt adequate features used for the prediction of a BLL; we can recover (1) by learning the dot product between the first and second (predictive mean), and between the first and third (predictive variance).

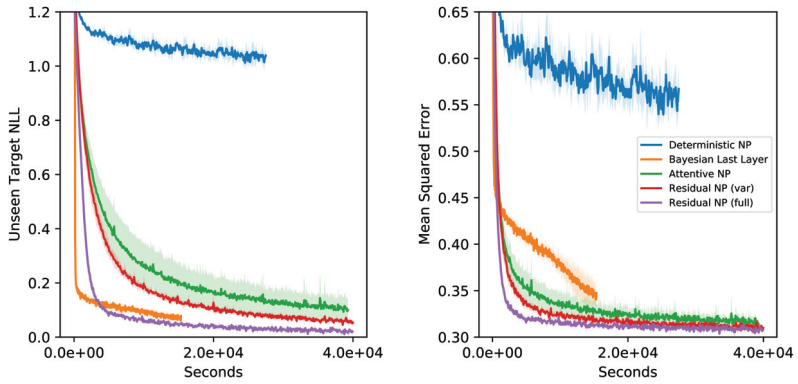
While the number of parameters does not increase significantly from original ANPs, RNPs take a few more computation steps for prediction; however, since a BLL takes $O(|C|^2 d_h + |C| d_h^2 + \min(d_h^3, |C|^3))$, the asymptotic time complexity does not increase when compared to an ANP given either cases of $|C| > d_h$ or $d_h > |C|$.

Stochastic Feature Extractor

A natural extension to a BLL for more accurate uncertainty prediction would be defining a prior and inferring a posterior on not only on the last layer but on the feature function as well. The exact predictive distribution will not be tractable anymore, but it has been shown that with approximate inference it is possible to express a far richer family of distributions over functions, e.g. Bayesian Neural Networks (Graves 2011; Kingma, Salimans, and Welling 2015). It has not been applied to a meta-learning context, as it is challenging to avoid costly training phases with the approximate inference algorithms proposed so far.

Inheriting a spirit of NPs, however, it is possible to approximately infer the posterior of feature function in a meta-learning context. We introduce the additional stochastic context embedding $\mathbf{z}_f$ for feature extractor in BLL, such that,

$$
\begin{aligned}
p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C)& \\
&\approx \int p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{X}_C, \mathbf{Y}_C, \phi_{\mathbf{z}_f}) q(\mathbf{z}_f|\mathbf{s}_C) d\mathbf{z},
\end{aligned}
$$



	Target NLL	MSE
NP	1.030 (±0.028)	0.557 (±0.020)
BLL	0.068 (±0.011)	0.344 (±0.012)
ANP	0.106 (±0.044)	0.316 (±0.005)
RNP (var)	0.051 (±0.009)	**0.310** (±0.003)
RNP (full)	**0.021** (±0.010)	**0.309** (±0.004)

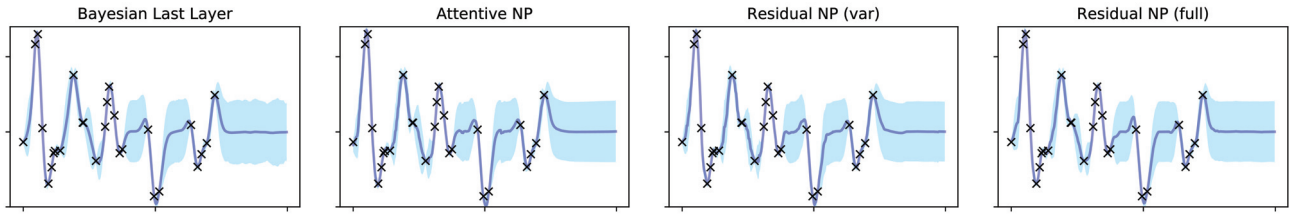


Figure 2: Learning curves, converged results and example predictions of deterministic models are shown. (top left) Wall-clock time v.s. unseen target negative log likelihood / mean squared error averaged over 5 random seeds. 95% confidence bounds are also shown as shades. (top right) Converged results after 2×10^6 iterations. 95% confidence bounds are also reported. (bottom) Predictive mean and variance of different models given the same context.

where $\phi_{\mathbf{z}_f}(\cdot)$ is implemented with a neural network of a concatenated input $\phi_{\mathbf{z}_f}(\cdot) = \phi([\cdot, \mathbf{z}_f])$ and $q(\mathbf{z}_f|\mathbf{s}_C)$ follows Eq. (3) and Eq. (4).

Such an extension can also be applied to ANPs and RNPs. A problem of having only a separate stochastic path is that the stochasticity induced by $\mathbf{z}$ does not affect the way how a deterministic target-dependent context summary $\mathbf{r}_{t|C}$ is computed. In the case where the correlations of $\mathbf{x}$s vary significantly over tasks (e.g. variable length-scale), the difference between having different context summarizing mechanism per task and having fixed mechanism over tasks will be maximized. With both the stochastic path and the stochastic feature extractor, we optimize over the modified objective,

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_f|\mathbf{s}_{TC}),q(\mathbf{z}|\mathbf{s}_{TC})}[\log p(\mathbf{Y}_T|\mathbf{X}_T, \mathbf{z}, \phi_{\mathbf{z}_f}, \mathbf{r}^{nl}_{t|C}, \mathbf{r}^{l}_{t|C})] - D_{KL}(q(\mathbf{z}|\mathbf{s}_{TC})\|q(\mathbf{z}|\mathbf{s}_C)) - D_{KL}(q(\mathbf{z}_f|\mathbf{s}_{TC})\|q(\mathbf{z}_f|\mathbf{s}_C)).$$

Experimental Results

Following the training method of an NP, we train on multiple realizations of the underlying data generating process. We sample a function per batch, and select random points to be the target and the context. Note that we did not forcefully include the context points among the target points as done in the previous NP benchmarks (Garnelo et al. 2018b; Kim et al. 2019) since such a training method introduces an additional bias. The convergence speed is hence a little slower when compared to what reported before. We used the architecture reported in (Kim et al. 2019) for an ANP structure; except for the fact that we used a feature extractor $\phi(\mathbf{x})$ of 2 hidden layers with d_h units each (d_h depends on the task) and with skip connections. Adam optimizer with a learning rate of 5e-5 is used throughout all experiments.

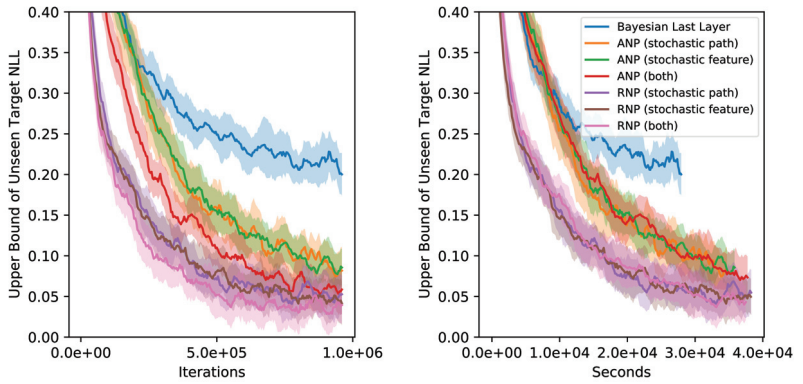
As a quantitative result we mainly report an unseen target NLL or an upper bound of it: however, note that it is significantly affected by the lower bound of predictive variance. We used the lower bound 10^{-2} of predictive variance, the lower bound used in previous NP researches. In the case of BLLs, we used the lower bound of observation variance of 10^{-4}, but augmented the predictive distribution to have 10^{-2} lower bound for a predictive variance for a fair comparison of NLL. For a stochastic feature extractor, we used $\mathbf{z}_f \in \mathbb{R}^5$ for all experiments[1].

1D Function Regression with Deterministic NPs

First, we demonstrate the idea of the RNP with deterministic path only models. The functions to train are generated from a Gaussian Process with a squared exponential kernel and small likelihood noise, with hyper-parameters fixed. The number of contexts and the number of targets is chosen randomly ($|C|, |T| \sim U[3, 100]$). Both X_C and X_T are also drawn uniformly in $[-20, 20]$. In this experiment, we used $d_h = 150$. This is just an illustrative example, and there is no need to use a known stochastic process (e.g. a GP) for training NPs.

In Figure 2 we show the running average of unseen target negative log-likelihood (NLL) and mean squared er-

[1]Code used for experiments can be found at : https://github.com/dlqudwns/Residual-Neural-Process



	Target NLL
BLL	0.212 ($\pm$0.024)
ANP (stochastic path)	0.079 ($\pm$0.026)
ANP (stochastic feature)	0.082 ($\pm$0.024)
ANP (both)	**0.059** ($\pm$0.025)
RNP (stochastic path)	**0.048** ($\pm$0.025)
RNP (stochastic feature)	**0.047** ($\pm$0.025)
RNP (both)	**0.038** ($\pm$0.023)

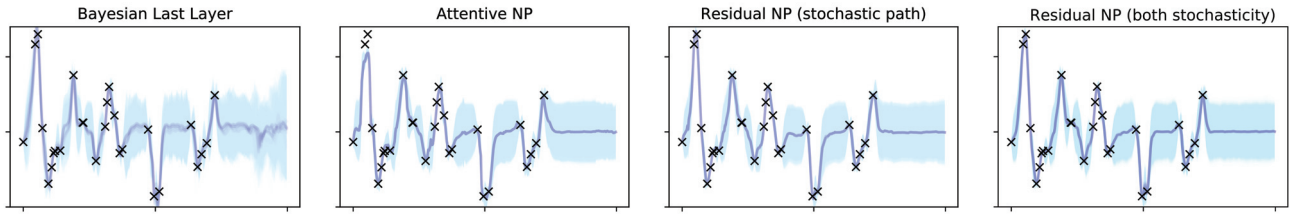


Figure 3: Learning curves, converged results, and example predictions of stochastic models are shown. (top left) Iterations / Wall-clock time v.s. an upper bound of the unseen target negative log-likelihood averaged over 10 random seeds. 95% confidence bounds are also shown as shades. (top right) Converged results after 1×10^6 iterations. 95% confidence bounds are also reported. (bottom) 20 samples of predictive mean and variance of different models given the same context.

ror (MSE). Deterministic NP, deterministic ANP, and BLL methods are included in the plot as baselines, and two different deterministic RNP models are demonstrated. One denoted with *Residual NP (var)* uses:

$$\mathbf{r}_{t|C}^{var} = \begin{bmatrix} \phi(\mathbf{x}_t) \\ \phi(\mathbf{x}_t) - \mathbf{\Phi}_C(\sigma_N^2\mathbf{I} + \mathbf{\Phi}_C^\top\mathbf{\Phi}_C)^{-1}\mathbf{\Phi}_C^\top\phi(\mathbf{x}_t) \end{bmatrix},$$

instead of full $\mathbf{r}_{t|C}^{l}$ as from Eq. (5).

It can be observed in the left figure that the RNPs (full) outperform the ANPs and the BLLs, showing much rapid decrease and better convergence point in terms of both NLL and MSE. The two learning curves are plotted against wall-clock time and show that the improvement of performance persists even with the increased running time of the RNPs. The RNPs with variance statistics only, the RNPs (var) improve from ANP but do not show dramatic improvement as the RNPs (full). Even though the performance of the BLLs alone is not always better than the ANPs, the results prove that adopting BLLs statistics and making attention based contexts summary to predict the residuals do help for better prediction overall. The performance at convergence is shown on the right, where the results of the RNPs required 15% of more time than the ANPs.

At the bottom of Figure 2, predictions of models are shown. With the help of the BLL statistics, the RNP seems to alleviate the discontinuities of mean prediction that occur in the ANP plot due to the attention mechanism.

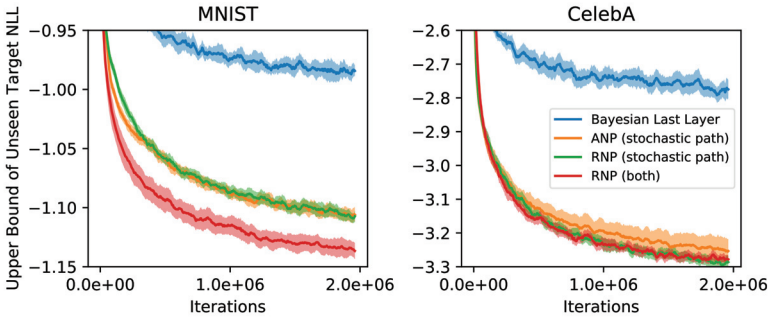
1D Function Regression with Stochastic NPs

We then demonstrate the idea of incorporating stochasticity in a feature extractor. Unlike previous experiment, we made kernel hyper-parameters to be also sampled randomly $l \sim U[0.1, 0.6]$, $\sigma_f \sim U[0.1, 1]$. In the experiments with stochastic models, we resort to evaluating approximate lower bounds:

$$\begin{aligned} &\log p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{X}_C, \mathbf{Y}_C) \geq \\ &\mathbb{E}_{q(\mathbf{z}_i|\mathbf{X}_C,\mathbf{Y}_C)}\left[\log\left(\frac{1}{K}\sum_{i=1}^{K}\frac{p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{z}_i)p(\mathbf{z}_i|\mathbf{X}_C,\mathbf{Y}_C)}{q(\mathbf{z}_i|\mathbf{X}_C,\mathbf{Y}_C)}\right)\right] \\ &\approx \mathbb{E}_{q(\mathbf{z}_i|\mathbf{X}_C,\mathbf{Y}_C)}\left[\log\left(\frac{1}{K}\sum_{i=1}^{K}p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{z}_i)\right)\right], \end{aligned} \tag{6}$$

as suggested in (Burda, Grosse, and Salakhutdinov 2016; Le et al. 2018). We used $K = 50$ and 2000 estimates are running averaged. The results are shown in Figure 3. $d_h = 150$ is used in this experiment. We compared the ANPs and the RNPs with different stochasticities: the *stochastic path* model is what used in the ANPs, where decoder is fed by $\mathbf{z}$. The *stochastic feature* model is what we suggest, where the feature extractor is fed by latent variable $\mathbf{z}_f$. We can also use two methods in the same time, and is denoted as *both*. The BLL method here is implemented with stochastic feature, and shown as baseline.

It can be observed that the performance of the BLL is relatively worse: apparently, the inclusion of stochastic feature slows down the learning of feature extractor. Nevertheless, the RNPs are getting benefit from the BLL part of the model



	MNIST NLL	CelebA NLL
BLL	-0.985 (±0.006)	-2.776 (±0.028)
ANP	-1.106 (±0.006)	-3.252 (±0.036)
RNP	-1.108 (±0.004)	**-3.289** (±0.017)
RNP (both)	**-1.135** (±0.006)	**-3.277** (±0.013)

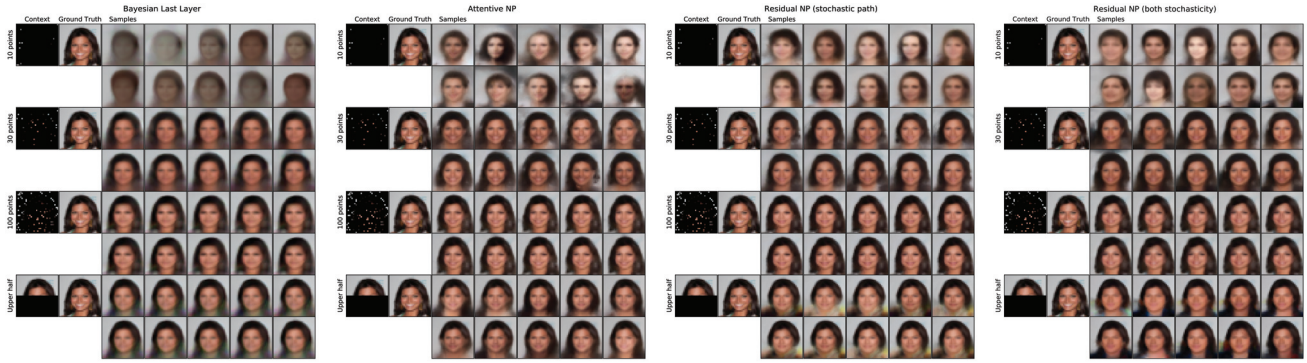


Figure 4: Learning curves, converged results and example predictions on image completion tasks are shown. (top left) Iterations v.s. an upper bound of the unseen target negative log likelihood averaged over 5 random seeds. 95% confidence bounds are also shown as shades. (top right) Converged results after 2×10^6 iterations. 95% confidence bounds are also reported. (bottom) Given four different contexts (10 points, 30 points, 100 points and upper half), 10 samples of a predictive mean of each process are shown. Predicted variances are not presented in this figure.

and shows increased learning speed from that of the ANPs. Either having one of the stochastic path or the stochastic feature shows similar performance, whereas having both shows better NLL.

On the right of the figure, we pick 4 models and presented their predictive distributions. Unlike ANPs, the RNPs did not miss any of the contexts. With both the stochastic path and the stochastic feature, the RNPs show more smooth and stable prediction when compared to the stochastic path only, which we suspect the variable attention that can learn varying length-scales with $\mathbf{z}_f$.

2D Function Regression on Image Data

If we treat images as a function that maps pixel location $\mathbf{x} \in \mathbb{R}^2$ to its pixel intensity $\mathbf{y}$ ($\mathbf{y} \in \mathbb{R}^3$ for RGB, $\mathbf{y} \in \mathbb{R}$ for gray-scale), the whole dataset becomes a group of functions sampled from some stochastic process. We trained and compared BLL, ANP, and RNP models on MNIST (LeCun et al. 1998) and sub-sampled 32×32 CelebA (Liu et al. 2015). We used random sizes of contexts and targets ($|C|, |T| \sim U[3, 200]$). The $\mathbf{x}$ and $\mathbf{y}$ are re-scaled to $[-1, 1]$ and $[-0.5, 0.5]$ respectively. $d_h = 250$ is used in this experiment.

The quantitative results reporting the upper bound of Target NLL is shown in Figure 4. $K = 50$ samples are used to evaluate the bound and 2000 bounds are running averaged. Compared to previous experiments, the performance gap between the BLL and the other methods is the largest, meaning that the nonlinear mapping from $\mathbf{Y}_C$ is crucial in the image completion tasks. While the RNPs with both stochasticity show improved results in both domains, the improvements come from different reasons: MNIST benefits from additional stochasticity while leveraging BLL features helps in CelebA. For the convergent results, the RNPs (both) required 50% of more wall-clock time than the ANPs. The bottom figure shows the predictive means for ten different samples of $\mathbf{z}$ and $\mathbf{z}_f$ given different contexts. The RNPs tend to show more diverse and realistic samples than the ANPs, especially when not enough samples were provided.

Conclusion

In this paper, we presented an RNP, which leverages a BLL for efficient residual learning. A stochastic feature extractor is also presented to approximately infer a posterior of feature extractor, complementing the stochasticity that was previously independent to the context summary. As a result, the RNPs with both types of stochasticities improves from the ANPs, both quantitatively (convergence speed and asymptotic performance of target NLL) and qualitatively (sample quality and diversity).

One of the directions for future work would be to investigate other various architectures of set function approxima-

tors that can better represent inference procedures. While we explicitly included BLL statistics to the decoder input in this work, a self-attention network that inherently capture them would be more desirable for both approximation accuracy and model simplicity.

Acknowledgments

This work was supported by the National Research Foundation (NRF) of Korea (NRF-2019M3F2A1072238 and NRF-2019R1A2C1087634), the Information Technology Research Center (ITRC) program of Korea (IITP-2019-2016-0-00464), and the Ministry of Science and Information communication Technology (MSIT) of Korea (IITP No. 2019-0-00075-001 and IITP No. 2017-0-01779 XAI).

References

Bloem-Reddy, B., and Teh, Y. W. 2019. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*. To appear in Jounral of Machine Learning Research.

Burda, Y.; Grosse, R.; and Salakhutdinov, R. 2016. Importance weighted autoencoders. In *International Conference on Learning Representations*.

Calandra, R.; Peters, J.; Rasmussen, C. E.; and Deisenroth, M. P. 2016. Manifold gaussian processes for regression. In *International Joint Conference on Neural Networks*.

Garnelo, M.; Rosenbaum, D.; Maddison, C.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y. W.; Rezende, D.; and Eslami, S. A. 2018a. Conditional neural processes. In *International Conference on Machine Learning*.

Garnelo, M.; Schwarz, J.; Rosenbaum, D.; Viola, F.; Rezende, D. J.; Eslami, S.; and Teh, Y. W. 2018b. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*.

Graves, A. 2012. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*. Springer. 5–13.

Harrison, J.; Sharma, A.; and Pavone, M. 2018. Meta-learning priors for efficient online bayesian regression. In *Workshop on the Algorithmic Foundations of Robotics*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Kim, H.; Mnih, A.; Schwarz, J.; Garnelo, M.; Eslami, A.; Rosenbaum, D.; Vinyals, O.; and Teh, Y. W. 2019. Attentive neural processes. In *International Conference on Learning Representations*.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*.

Le, T. A.; Kim, H.; Garnelo, M.; Rosenbaum, D.; Schwarz, J.; and Teh, Y. W. 2018. Empirical evaluation of neural process objectives. In *NeurIPS Workshop on Bayesian Deep Learning*.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*.

Murphy, R. L.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Wagstaff, E.; Fuchs, F. B.; Engelcke, M.; Posner, I.; and Osborne, M. 2019. On the limitations of representing functions on sets. In *International Conference on Machine Learning*.

Weber, N.; Starc, J.; Mittal, A.; Blanco, R.; and Màrquez, L. 2018. Optimizing over a Bayesian last layer. In *NeurIPS Workshop on Bayesian Deep Learning*.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. In *Advances in Neural Information Processing Systems*.