NEURAL BOOTSTRAPPING ATTENTION FOR NEURAL PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Processes (NP) learn to fit a broad class of stochastic processes with neural networks. Modeling functional uncertainty is an important aspect of learning stochastic processes. Recently, Bootstrapping (Attentive) Neural Processes (B(A)NP) propose a bootstrap method to capture the functional uncertainty which can replace the latent variable in (Attentive) Neural Processes ((A)NP), thus overcoming the limitations of Gaussian assumption on the latent variable. However, B(A)NP conduct bootstrapping in a non-parallelizable and memory-inefficient way and fail to capture diverse patterns in the stochastic processes. Furthermore, we found that ANP and BANP both tend to overfit in some cases. To resolve these problems, we propose an efficient and easy-to-implement approach, Neural Bootstrapping Attentive Neural Processes (NeuBANP). NeuBANP learns to generate the bootstrap distribution of random functions by injecting multiple random weights into the encoder and the loss function. We evaluate our models in benchmark experiments including Bayesian optimization and contextual multi-armed bandit. NeuBANP achieves state-of-the-art performance in both of the sequential decision-making tasks, and this empirically shows that our method greatly improves the quality of functional uncertainty modeling.

1 INTRODUCTION

Neural Processes (NP) (Garnelo et al., 2018b) define distributions over functions given a set of observations, and are trained via a meta-learning framework so that it can adapt to new functions rapidly. NP learns to model a wide range of stochastic processes and can estimate the uncertainty over the predictions with less computational effort, compared to Gaussian Processes (GP) (Rasmussen, 2003). However, NP frequently suffers from a fundamental drawback of underfitting. As a remedy of the underfitting issue, Attentive Neural Processes (ANP) (Kim et al., 2018) applies the attention modules to the encoder network. Despite this modification, a single Gaussian latent variable of (A)NP has a limitation in inducing *functional uncertainty* (Louizos et al., 2019), a global uncertainty that decides the distribution over the space of trajectories or functions.

Appropriate modeling of functional uncertainty in stochastic processes improves the predictive performance and diversity in function realizations (Le et al., 2018), thus provides a principled way to guide agents to find optimal candidates in sequential decision-making problems. In these tasks, a model needs to approximate a function well and estimate variance correctly to optimize a black-box function whose analytic information, such as gradients, is not given. Although GP is widely used for these tasks, these are the promising area for the application of NP algorithms because GP is computationally expensive, and it can be hard to choose an appropriate prior. Recently, Bootstrapping (Attentive) Neural Processes (B(A)NP) (Lee et al., 2020) modify (A)NP to induce more robust uncertainty estimation by employing the residual bootstrapping. However, B(A)NP underperforms in capturing a functional uncertainty because the residual bootstrapping works in a homoscedastic way, removing the connection between the feature and the label in its bootstrapped samples. Furthermore, the residual bootstrapping used in B(A)NP demands a higher computational burden compared to (A)NP since it requires multiple computations of the encoder network, the adaptation layer, and additional heuristics.

As shown in Figure 1, ANP and BANP tend to overfit for a simple regression task rather than learning the underlying heteroscedasticity. We observed that this problem is a by-product of the attention

modules used in ANP and BANP, allowing the model to fit such a complex biased pattern (see Figure 6). When ANP replaces the mean aggregation step in the encoder with attention modules, it was to resolve underfitting of NP, but our observation implies that there is a need for proper regularization to attention modules to prevent overfitting. Additionally, as explained in Section 4, ANP and BANP often tend to estimate homogeneous uncertainty regardless of the area where the observation is given.

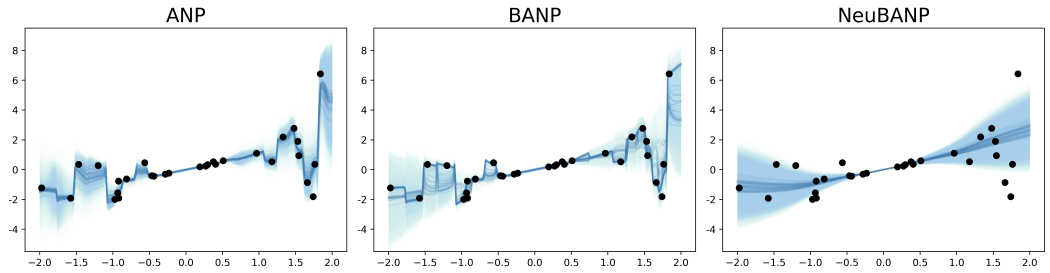

Fig. 1. Each plot shows predictions given by ANP, BANP, and the proposed method in a linear regression. The ground-truth function is a simple linear function with heterogeneous variance: $y = x + \beta\epsilon(x)$ where $\epsilon(x) \sim N(0, \sigma^2(x))$ and $\sigma(x) = \sqrt{x^2 + 10^{-5}}$. We used the official code provided by Lee et al. (2020).

To resolve the problems of previous NPs, we introduce a novel bootstrapping method for neural processes, **Neu**ral **B**ootstrapping **A**ttentive **N**eural **P**rocesses (NeuBANP). Motivated from the recent work on efficient bootstrapping of the neural network, Neural Bootstrapper (Shin et al., 2021), we introduce bootstrapping of the attention in a computationally efficient way by simple modification of the input of attention modules and the loss function, instead of memory-inefficient resampling and contrived heuristics employed in BANP. NeuBANP learns the connection between random bootstrap weights and input data, with properly guided loss function adopted from the random weight bootstrapping (Præstgaard & Wellner, 1993). This modification is simple, but gives a strong consistency between the bootstrapped samples and the representations from the attention modules, unlike BANP whose encoder inputs bootstrapped samples which lack the connection between their features and labels because of the residual bootstrapping. Besides, our bootstrapping method, which implements the concatenation and multiplication of random bootstrap weights, operates as a regularizer on the attention networks, thus preventing overfitted predictions observed in previous attention-based NP algorithms. NeuBANP is trained to generate a valid predictive distribution by utilizing the uncertainty inherent in observations by bootstrapping instead of the uncertainty that depends on the prior assumption on the latent variable as in (A)NP. This leads to the success in capturing heteroscedasticity of the data and modeling functional uncertainty in stochastic processes. As a result, NeuBANP achieves state-of-the-art performance in sequential decision-making problems such as Bayesian optimization and contextual multi-armed bandit. The experimental results demonstrate that our model provides promising capabilities as an efficient neural approximation of stochastic processes.

Contributions We propose NeuBANP, an easy-to-implement and computationally efficient method for bootstrapping ANP. The proposed method has a novelty in learning a generator for bootstrapping stochastic processes under a meta-learning framework. NeuBANP resolves overfitting problem of attention modules and shows robust performance on heteroscedastic models without heuristics like the extra adaptation layer in BANP. NeuBANP estimates functional uncertainty better than BANP, and achieves state-of-the-art performance in stochastic optimization problems, including multi-dimensional Bayesian optimization and contextual multi-armed bandit.

2 Preliminaries

2.1 Meta-Learning Framework of Neural Processes

Consider data $\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$, the pairs of inputs $x_i \in \mathcal{X}$ and outputs $y_i \in \mathcal{Y}$. Let $\mathcal{P}$ be a probability distribution over functions $f \in \mathcal{F}$; $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_i^2)$, hence $\mathcal{P}$ determines the distribution of $\mathcal{D}$. For disjoint index sets $\mathcal{C}$ and $\mathcal{T}$ satisfying $\mathcal{C} \cup \mathcal{T} = [n]$, define *context* $\mathcal{D}_\mathcal{C} = (X_\mathcal{C}, Y_\mathcal{C}) = \{(x_c, y_c)\}_{c \in \mathcal{C}}$ and *target* $\mathcal{D}_\mathcal{T} = (X_\mathcal{T}, Y_\mathcal{T}) = \{(x_t, y_t)\}_{t \in \mathcal{T}}$, so

that $\mathcal{D} = \mathcal{D}_\mathcal{C} \cup \mathcal{D}_\mathcal{T}$. The task is to learn the neural processes p_θ that fits the stochastic processes $f \sim \mathcal{P}$ given $\mathcal{D}_\mathcal{C}$ when $\mathcal{C}$ consists of various number of uniformly chosen numbers from $[n]$, as follows:

$$\theta_\star = \operatorname*{argmin}_\theta \mathbb{E}_{f\sim\mathcal{P}}\Big[\mathbb{E}_{\mathcal{C}\sim[n]}\Big[-\log p_\theta(Y|X,\mathcal{D}_\mathcal{C})\Big]\Big]. \tag{1}$$

This meta-learning framework allows p_θ to learn diverse patterns in $\mathcal{F}$ via a prior distribution $\mathcal{P}$; hence NP can predict target points conditioned on contexts adaptively in the inference phase.

2.2 (Bootstrapping) Attentive Neural Processes

ANP ANP is a variant of NP equipped with attention modules in the encoder part. See Appendix A for the detailed definition of attention operations. Let the context $\mathcal{D}_\mathcal{C}$ is given, and the model aims to infer y for a given feature $x \in \mathcal{X}$. The encoder network of ANP maps $(x, \mathcal{D}_\mathcal{C})$ into a pair of representation vectors $\mathbf{r} = (\mathbf{z}, \mathbf{h})$ as follows:

$$\{s_c\}_{c\in\mathcal{C}} = \text{SelfAttn}(\mathcal{D}_\mathcal{C}), \quad \mathbf{s}_\mathcal{C} = \text{mean}(\{s_c\}_{c\in\mathcal{C}}), \quad \mathbf{z} \sim \mathcal{N}(z|\mu_z(\mathbf{s}_\mathcal{C}), \sigma_z^2(\mathbf{s}_\mathcal{C})) \tag{2}$$

$$\{h_c\}_{c\in\mathcal{C}} = \text{SelfAttn}(\mathcal{D}_\mathcal{C}), \quad \mathbf{h} = \text{CrossAttn}(x, X_\mathcal{C}, \{h_c\}_{c\in\mathcal{C}}) \tag{3}$$

Here, μ_z and σ_z are single linear layers that map $\mathbf{s}_\mathcal{C}$ to mean and standard deviation of the latent variable $\mathbf{z}$, respectively. In ANP, $\mathbf{z}$ and $\mathbf{h}$ refer to latent path and deterministic path, respectively. The deterministic path models the overall skeleton of the encoder network, while the latent path models the functional uncertainty using a stochastic global latent variable (Garnelo et al., 2018b; Kim et al., 2018). Then the decoder network takes the representations $\mathbf{r}$ and target data x as inputs to predict $\mu(x, \mathbf{r})$ and $\sigma(x, \mathbf{r})$, the parameters of the conditional predictive distribution $p(y|x, \mathcal{D}_\mathcal{C}) = \mathcal{N}(y|\mu, \sigma^2)$.

BANP The global latent variable in ANP potentially limits the flexibility in expressing functional uncertainty. BANP proposes a method that utilizes paired bootstrapping and residual bootstrapping to model stochasticity in a data-driven way. First, they resample pairs of (x_c, y_c) with replacement to construct $\hat{\mathcal{D}}_\mathcal{C}^{(b)} = (\hat{X}_\mathcal{C}^{(b)}, \hat{Y}_\mathcal{C}^{(b)})$. The resampled context is encoded into the representation vector $\hat{\mathbf{r}}^{(b)} = (\hat{\mathbf{z}}^{(b)}, \hat{\mathbf{h}}^{(b)})$ by replacing $\mathcal{D}_\mathcal{C}$ in (2) and (3) by $\hat{\mathcal{D}}_\mathcal{C}^{(b)}$. Then BANP conducts the residual bootstrapping to construct bootstrapped contexts again $\tilde{\mathcal{D}}_\mathcal{C}^{(b)} = (\tilde{X}_\mathcal{C}^{(b)}, \tilde{Y}_\mathcal{C}^{(b)})$. By using $\mathcal{D}_\mathcal{C}$ and $\tilde{\mathcal{D}}_\mathcal{C}^{(b)}$ in (2) and (3) separately, BANP gets the representations of contexts $(\mathbf{r}, \tilde{\mathbf{r}}^{(b)})$. Finally, BANP uses an adaptation layer to merge $(\mathbf{r}, \tilde{\mathbf{r}}^{(b)})$ and obtain $\mu^{(b)}, \sigma^{(b)}$ through the decoder network (see Figure 2). Repeating the above procedure for $b = 1, \ldots, B$, BANP gets an ensembled distribution:

$$p(y|x, \mathbf{r}, \tilde{\mathbf{r}}^{(b)}) = \mathcal{N}(y|\mu^{(b)}, (\sigma^{(b)})^2), \quad p(y|x, \mathcal{D}_\mathcal{C}) \approx \frac{1}{B}\sum_{b=1}^{B} p(y|x, \mathbf{r}, \tilde{\mathbf{r}}^{(b)}). \tag{4}$$

Due to the residual bootstrap step, BANP should conduct the encoder computation three times and the decoder computation twice for a single forward propagation. These additional calculations cause the computational bottleneck in the training and inference of BANP.

2.3 Neural Bootstrapper

Repetitions of training restrain the practical use of bootstrap procedures in deep neural networks due to their high computational burden. To alleviate this, Shin et al. (2021) proposes Neural Bootstrapper (NeuBoots). NeuBoots circumvents multiple training of networks by learning a bootstrap generator.

Random Weight Bootstrapping Let $\sum_{c\in\mathcal{C}} \ell(f(x_c), y_c)$ be the loss function of interest for a neural network f. In standard bootstrap procedures, a bootstrapped neural network can be obtained by minimizing the loss function weighted by a random bootstrap weight $\mathbf{w}_\mathcal{C} := \{w_c : c \in \mathcal{C}\}$:

$$L(\mathbf{w}_\mathcal{C}, f, \mathcal{D}_\mathcal{C}) = \sum_{c\in\mathcal{C}} w_c \ell(f(x_c), y_c). \tag{5}$$

According to the choice of the distribution on $\mathbf{w}_\mathcal{C}$, various bootstrap procedures can be represented under the form of (5); e.g., the paired bootstrap by $\mathbf{w}_\mathcal{C} \sim \text{Multinomial}(n; 1/n, \ldots, 1/n)$ and the

Random Weight Bootstrapping (RWB) (Præstgaard & Wellner, 1993; Newton & Raftery, 1994) by $\mathbf{w}_{\mathcal{C}} \sim |\mathcal{C}| \times \text{Dirichlet}(1, \ldots, 1)$. NeuBoots utilizes RWB to avoid the data discard problem which can occur in the standard bootstrapping. We then compute bootstrapped neural networks $\{\widehat{f}^{(b)} : b = 1, \ldots, B\}$ via minimization of (5) for sampled $\mathbf{w}_{\mathcal{C}}^{(1)}, \ldots, \mathbf{w}_{\mathcal{C}}^{(B)}$.

Learning To Generate Bootstrap Distribution The main idea of NeuBoots is to construct a single generative network that models the bootstrapped neural networks with varying bootstrap weights in (5). This formulation modifies the backbone network in a form of $f(x, \mathbf{w}_{\mathcal{C}})$ that inputs both feature x and bootstrap weight $\mathbf{w}_{\mathcal{C}}$. Shin et al. (2021) show that the minimizer of the following loss generates valid bootstrap evaluations that match the results of the standard bootstrap procedure:

$$\mathcal{L}(f, \mathcal{D}_{\mathcal{C}}) = \mathbb{E}_{\mathbf{w}_{\mathcal{C}} \sim |\mathcal{C}| \times \text{Dirichlet}(1,\ldots,1)} \left[L(\mathbf{w}_{\mathcal{C}}, f(\cdot, \mathbf{w}_{\mathcal{C}}), \mathcal{D}_{\mathcal{C}})) \right],$$

We call this weighted bootstrapping loss. Once this generator is trained via a single optimization procedure, we can efficiently generate bootstrapped predictions by plugging random bootstrap weights in the trained generator; i.e., for a feature of interest x, the trained generator inputs $\{\mathbf{w}_{\mathcal{C}}^{(b)}\}_{b=1}^{B}$ and produces bootstrapped predictions $\hat{y}^{(b)} = \hat{f}(x, \mathbf{w}_{\mathcal{C}}^{(b)})$ for $b = 1, \ldots, B$.

3 Neural Bootstrapping Attentive Neural Processes

We propose a novel class of NP, called Neural Bootstrapping Attentive Neural Processes (NeuBANP). Aligned with the previous formulation of NP families, we can define our model p_θ as:

$$p_\theta(Y|X, \mathcal{D}_{\mathcal{C}}) = \int p_\varphi(Y|X, \mathbf{h}, \mathbf{z}) q(\mathbf{z}|\mathcal{D}_{\mathcal{C}}) \mathrm{d}\mathbf{z} = \int \prod_{i=1}^{n} p_\varphi(y_i|x_i, \mathbf{h}, \mathbf{z}) q(\mathbf{z}|\mathcal{D}_{\mathcal{C}}) \mathrm{d}\mathbf{z} \tag{6}$$

$$\approx \prod_{i=1}^{n} p_\varphi(y_i|x_i, \mathbf{h}, \mathbf{z}) \text{ where } \mathbf{z} \sim q(\mathbf{z}|\mathcal{D}_{\mathcal{C}}). \tag{7}$$

Here p_φ is the decoder network and q denotes the posterior distribution. Since the above integral is intractable, we approximate the predictive distribution by sampling $\mathbf{z}$ from the bootstrap distribution $q(\cdot|\mathcal{D}_{\mathcal{C}})$ given the context $\mathcal{D}_{\mathcal{C}}$, instead of Gaussian distribution as in (A)NP. Precisely, we train a generative encoder network g_ϕ, which outputs bootstrapped representation pairs $(\mathbf{h}, \mathbf{z}) = \{(\mathbf{h}^{(b)}, \mathbf{z}^{(b)})\}_{b=1}^{B} = g_\phi(X, \mathcal{D}_{\mathcal{C}}, \{\mathbf{w}_{\mathcal{C}}^{(b)}\}_{b=1}^{B})$. Through the meta-learning framework (1), our model learns to generate bootstrapped predictions for an arbitrarily given pattern $f \sim \mathcal{P}$. Thus, our approach can be regarded as a *learn-to-bootstrap* method for stochastic processes. Note that NeuBANP can obtain a number of bootstrapped predictions by simply plugging different bootstrap weights into g_ϕ, while BANP needs repetitive data resampling from scratch. Figure 2 shows the difference between BANP and NeuBANP in detail.

3.1 Neural Bootstrapping Contexts with Attention Modules

To train a generative network, which outputs the bootstrapped representations, we modify the encoder network in ANP to take both $(x, \mathcal{D}_{\mathcal{C}})$ and bootstrap weight $\mathbf{w}_{\mathcal{C}}$ as inputs. We introduce the *posterior* and the *prediction* paths in the encoder network, analogous to NP's latent and deterministic paths. NeuBANP is designed to leverage RWB, which is theoretically proven as a valid bootstrap method. In detail, the posterior and prediction path in NeuBANP take the random bootstrap weight as an auxiliary input. This input provides enough randomness into the network so that our model can successfully capture the functional uncertainty. It can replace BANP's bootstrap method without imposing additional overhead costs on the architecture.

Posterior path We tag each context $(x_c, y_c) \in \mathcal{D}_{\mathcal{C}}$ with a bootstrap weight $w_c^{(b)}$ to construct bootstrapped contexts $\mathcal{D}_{\mathcal{C}}^{(b)} := \{(x_c, y_c, w_c^{(b)})\}_{c \in \mathcal{C}}$. Then the posterior path receives $\mathcal{D}_{\mathcal{C}}^{(b)}$ as input and outputs a latent variable $\mathbf{z}^{(b)}$. In detail, we apply self-attention to $\mathcal{D}_{\mathcal{C}}^{(b)}$ and multiply the resultant representation $\mathbf{z}_{\mathcal{C}}^{(b)}$ by the bootstrap weight $\mathbf{w}_{\mathcal{C}}^{(b)}$ before mean aggregation. This path connects the contexts with weighted bootstrapping loss during the training and allows the bootstrap weights to model bootstrapped posterior distribution q by controlling the magnitude of each context representation.

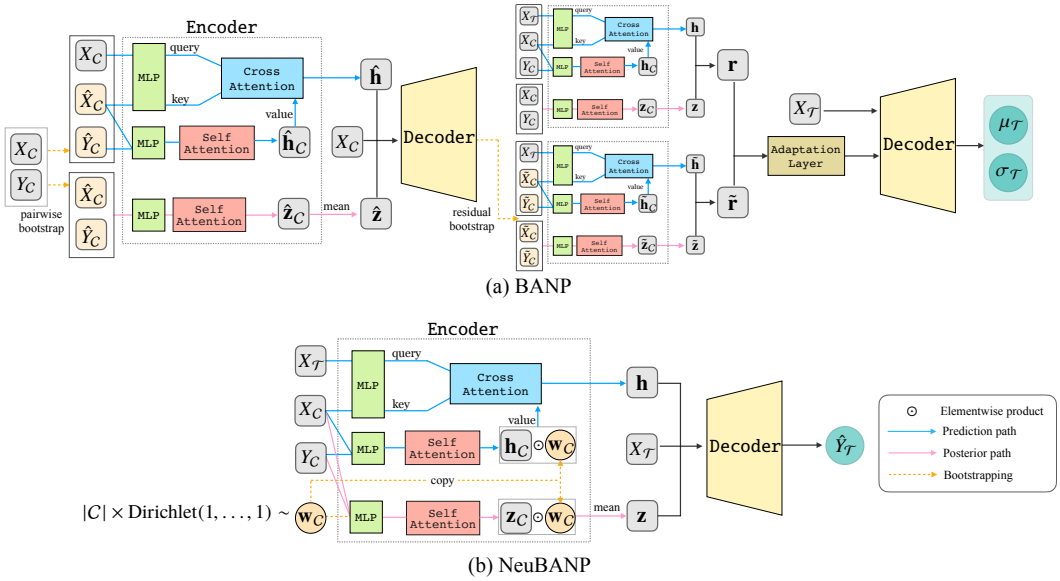

Fig. 2. A single forward computation of (a) BANP and (b) NeuBANP.

Prediction path The prediction path outputs a target-specific representation $\mathbf{h}^{(b)}$ that is relevant for the prediction. We apply self-attention to contexts $\mathcal{D}_\mathcal{C}$ and multiply the bootstrap weight $\mathbf{w}_\mathcal{C}^{(b)}$ element-wisely to compute another bootstrapped representation. Cross-attention module uses $X_\mathcal{C}$ and this bootstrapped representation as *key-value* pairs to which the target *query* x attends. Consequently, reflecting the bootstrap weights shared with the posterior path, the prediction path models interactions between the given context and the target input.

We summarize the above posterior and prediction paths as follows:

$$\mathbf{z}_\mathcal{C}^{(b)} = \{z_c^{(b)}\}_{c\in\mathcal{C}} = \text{SelfAttn}(\mathcal{D}_\mathcal{C}^{(b)}), \quad \mathbf{z}^{(b)} = \text{mean}(\mathbf{z}_\mathcal{C}^{(b)} \odot \mathbf{w}_\mathcal{C}^{(b)}) \tag{8}$$

$$\mathbf{h}_\mathcal{C} = \{h_c\}_{c\in\mathcal{C}} = \text{SelfAttn}(\mathcal{D}_\mathcal{C}), \quad \mathbf{h}^{(b)} = \text{CrossAttn}(x, X_\mathcal{C}, \mathbf{h}_\mathcal{C} \odot \mathbf{w}_\mathcal{C}^{(b)}) \tag{9}$$

where $\odot$ denotes element-wise multiplication. Concatenation of random bootstrap weights and the input of the posterior path, as to make $\mathcal{D}_\mathcal{C}^{(b)}$, allows $\mathbf{z}$ to contain bootstrap information and hence the decoder can generate the proper bootstrap distribution. Additionally, by multiplying the random weights to the representations from both paths, as in (8) and (9), the representations are consistent with the weights and maximize bootstrapping effect. We think that these multiplications also provide the effect of regularization, which mitigates the overfitting tendency which ANP and B(A)NP show, for example, in a simple regression experiment (see Figure 1).

Decoder and Bootsrapped Prediction The decoder takes $(\mathbf{z}^{(b)}, \mathbf{h}^{(b)})$ from the encoder network and target $x \in \mathcal{X}$ as inputs to generate a prediction $\hat{y}^{(b)} = \text{MLP}(x, \mathbf{z}^{(b)}, \mathbf{h}^{(b)})$. We can generate bootstrap samples $\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(B)}$ by plugging $\mathbf{w}_\mathcal{C}^{(1)}, \mathbf{w}_\mathcal{C}^{(2)}, ..., \mathbf{w}_\mathcal{C}^{(B)}$ into the encoder $g_\phi(x, \mathcal{D}_\mathcal{C}, \cdot)$, respectively. We estimate the predictive mean and standard deviation using bootstrap samples:

$$\mu = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^{(b)}, \quad \sigma = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{y}^{(b)} - \mu)^2} \tag{10}$$

In the previous NP algorithms, the decoder directly outputs the parameters of the predictive distribution, but the decoder of NeuBANP outputs the stochastic predictions $\{\hat{y}^{(b)}\}_{b=1}^{B}$. This design of output is a distinction of our model from other NP algorithms and allows the nonparametric estimation. Due to this structure, existing NPs set the lower bound of standard deviation for robust performance. We found that the performance was susceptible to the lower bound value. However, NeuBANP naturally obtains parameters of predictive distribution using bootstrap predictions and shows better performance without such heuristics. It also has the advantage of calculating higher-order statistics without changing the structure.

3.2 TRAINING

Weighted Bootstrapping Loss We train NEUBANP with weighted loss similar to that of NeuBoots as demonstrated in Section 2.3. Only context data has the corresponding bootstrap weight in our setting, but the model still has to fit target data. Thus we designed loss function as a sum of weighted context loss $\mathcal{L}_{\text{context}}$ and non-weighted target loss $\mathcal{L}_{\text{target}}$ as follows:

$$\mathcal{L}_{\text{total}} = \underbrace{\frac{1}{B}\sum_{b=1}^{B}\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\Big(-w_c^{(b)}\log\mathcal{N}(y_c|\mu_c,\sigma_c^2)\Big)}_{\mathcal{L}_{\text{context}}} + \underbrace{\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\Big(-\log\mathcal{N}(y_t|\mu_t,\sigma_t^2)\Big)}_{\mathcal{L}_{\text{target}}} \tag{11}$$

where μ_c, σ_c, μ_t, and σ_t are computed by (10) for context $(x_c, y_c) \in \mathcal{D}_{\mathcal{C}}$ and target $(x_t, y_t) \in \mathcal{D}_{\mathcal{T}}$. Averaging weighted context loss with multiple bootstrap weights improves the robustness of a model by showing various bootstrap datasets during training. We trained the model with negative log-likelihood (NLL). One can replace the NLL with a different loss function, such as cross-entropy according to the target tasks.

4 EXPERIMENTS

We conducted experiments to compare NEUBANP with the previous NP methods for regression and sequential decision-making problems. For regression tasks, we conducted one-dimensional (1D) regression experiments on random functions generated from GP prior and image completion tasks as two-dimensional (2D) regression (see Appendix B.4 for image completion). For sequential decision-making problems, we evaluated each method in Bayesian optimization (BO) and Contextual Multi-Armed Bandit (CMAB).

4.1 NONPARAMETRIC REGRESSION AND UNCERTAINTY ESTIMATION

Settings We followed the settings in Lee et al. (2020). To obtain meta-training datasets, we sampled batches of random functions from GP prior with RBF kernel, and context and target points were chosen randomly from each function. In addition, kernel parameters of GP were randomly sampled so that the models could learn about various functions. NEUBANP was trained with 10 bootstrap samples, and we confirmed that it is robust to the number of samples. Please refer to Appendix B.1 for details.

Results The numerical results are summarized in Table 2. ANP and BANP tend to estimate homogenous uncertainties for all target points, regardless of the context-given area (see Figure 3). In the case of BANP, homoscedasticity occurs even when the number of contexts is small. Furthermore, we argue that residual bootstrap is not an appropriate method to model functional uncertainty. In Figure 3, the samples of mean prediction of BANP are very similar, not like those of the other methods. NEUBANP resolves these problems without a computational bottleneck.

4.2 BAYESIAN OPTIMIZATION

Since NP can approximate a class of arbitrary functions, it can replace GP, the commonly used surrogate model of BO. It is crucial to approximate the objective function from the given observations using the surrogate model, but evaluating the acquisition function and determining the subsequent samples are vital for efficient exploration. We evaluated the proposed method on various black-box functions, which may be unobserved in the meta-training step (see Algorithm 1).

Settings For 1D, we followed the same setting in Lee et al. (2020). We set objective functions generated from GP with RBF, Matérn 5/2, and Periodic kernels and applied the models trained in Section 4.1. Furthermore, we demonstrated the BO performance of NEUBANP for multi-dimensional settings (2D and 3D). We set objective functions as various benchmark functions used in the optimization literature (Kim, 2020; Kim & Choi, 2017). See Appendix B.2 for details. A simple regret measured the performance of each model, and the mean performance over 100 experiments is reported for reliable evaluations.

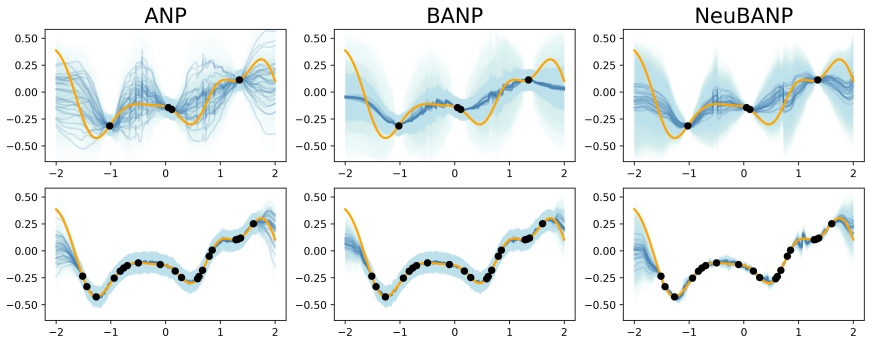

Fig. 3. Comparison of ANP, BANP, and NeuBANP in 1D regression given 4 context points (top) and 20 context points (bottom). Orange lines represent the ground-truth function. Blue lines are predictive mean given by each model and shaded region denotes the standard deviation (amount of uncertainty). To visualize the quality of functional uncertainty, we overlapped multiple shaded areas obtained with 30 sampled outputs for each input.

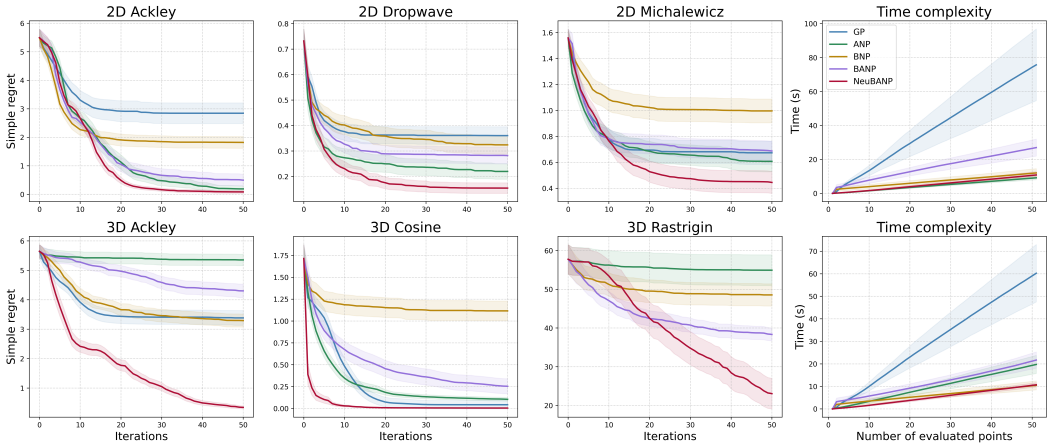

Fig. 4. Left: Multi-dimensional Bayesian optimization results on various benchmark functions with UCB as an acquisition function. Bold lines represent the mean performance over 100 experiments. We indicate 20% of the standard deviation. **Right most:** Time complexity of each model as the number of observations increases.

Results NeuBANP outperformed the other methods in BO experiments (see Figure 4 and 7). Table 3 and 4 shows numerical results. For multi-dimensional BO, NeuBANP achieved the best results for every function except the Hartmann-3D when using Upper Confidence Bound (UCB) as an acquisition function. For the Hartmann-3D, the performance of NeuBANP is statistically comparable to the best method (ANP) when considering the standard deviation. For Goldsteinprice and Rastrigin functions, GP records poor performance due to its numerical errors during the optimization procedure (see Appendix B.2). The rightmost plots of Figure 4 show the time complexity according to the number of observation points. NeuBANP has the fastest decreasing rate of regrets in terms of iterations in both cases. We also conducted BO experiments using Expected Improvement (EI) to test the performance of NeuBANP to be independent of the selection of acquisition functions (Figure 8).

Figure 5 demonstrates the better exploration strategy of NeuBANP compared to BANP. The second and fourth columns show the explored points by BANP and NeuBANP, respectively. In most cases, NeuBANP converged to the optimum faster than BANP. The third and fifth column shows the contour plots of acquisition functions of each method. Note that NeuBANP accurately infers the potential area of the optimum compared to BANP. As explained above, NeuBANP can estimate heterogeneous uncertainty, and thus it is able to handle well for the exploration-exploitation trade-off, which is essential in sequential decision-making problems.

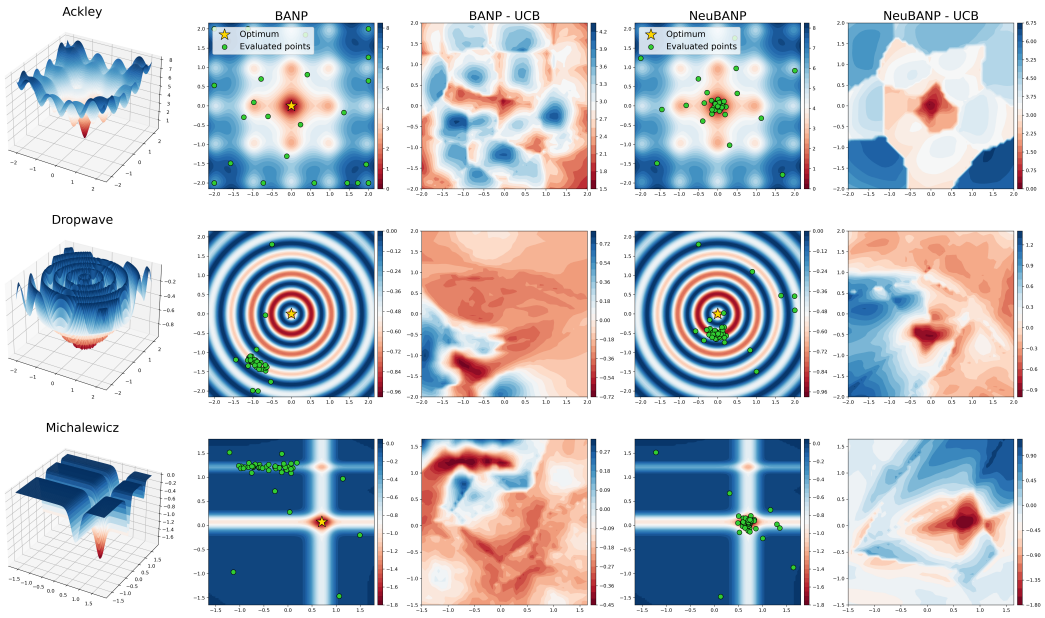

Fig. 5. First column: 2D objective functions **Second & Fourth columns:** Contour plots of functions and the evaluated points during Bayesian optimization. **Third & Fifth columns:** UCB value at the last iteration.

4.3 Contextual Multi-Armed Bandit

We conducted a CMAB experiment, the *wheel bandit* problem, as in Garnelo et al. (2018b) to show that NeuBANP works efficiently as well as BO based on its performance of uncertainty estimation.

Settings We followed the same environment in Garnelo et al. (2018b), but used UCB policy. For more details, please refer to Appendix B.3. The parameter δ determines the environment of the wheel bandit problem. As δ increases, high-reward observation becomes sparse, which makes the problem more difficult. We set the baselines of CMAB experiment to MAML (Finn et al., 2017), Neural Linear (Riquelme et al., 2018) and NP. We measured cumulative regrets and simple regrets for 2,000 iterations to demonstrate the performance of each model. We conducted each experiment 50 times, and the results were normalized to the performance of the uniform policy.

Results NeuBANP performed well for various δ values (see Table 1). Note that NeuBANP showed better performance in challenging environments with sparse high rewards. NeuBANP showed the ability to learn various reward distributions based on appropriate functional uncertainty modeling. The result demonstrates that NeuBANP can utilize a small number of context information and make accurate estimations.

5 Related Works

Neural Processes CNP (Garnelo et al., 2018a) uses a pair of encoder and decoder networks to produce a predictive posterior distribution over functions. NP (Garnelo et al., 2018b) introduces a global latent variable to embed functional uncertainty in the deterministic architecture of CNP and predicts various outputs given the same context data. ANP (Kim et al., 2018) then enhances predictive accuracy by replacing MLP modules in NP with the attention modules. BNP (Lee et al., 2020) proposes the bootstrap method to model uncertainty in stochastic processes without the assumption of a single latent variable on which previous methods rely. In addition to these works, there are many attempts to use NPs in various tasks like modeling a sequence of stochastic processes (Singh et al., 2019; Yoon et al., 2020) or approximation of translation-equivariant functions (Gordon et al., 2019).

Regret	Method	$\delta = 0.5$	$\delta = 0.7$	$\delta = 0.9$	$\delta = 0.95$	$\delta = 0.99$
Cumulative	Uniform	100.00 ± 0.08	100.00 ± 0.09	100.00 ± 0.25	100.00 ± 0.37	100.00 ± 0.78
	Neural Linear	0.95 ± 0.02	1.60 ± 0.03	4.65 ± 0.18	9.56 ± 0.36	49.63 ± 2.41
	MAML	2.95 ± 0.12	3.11 ± 0.16	4.84 ± 0.22	7.01 ± 0.33	22.93 ± 1.57
	NP	1.60 ± 0.06	1.75 ± 0.05	3.31 ± 0.10	5.71 ± 0.24	22.13 ± 1.23
	NeuBANP	**0.85** ± 0.22	**1.02** ± 0.27	**1.85** ± 0.56	**3.04** ± 0.88	**9.76** ± 1.93
Simple	Uniform	100.00 ± 0.45	100.00 ± 0.78	100.00 ± 1.18	100.00 ± 2.21	100.00 ± 4.21
	Neural Linear	**0.33** ± 0.04	**0.79** ± 0.07	2.17 ± 0.14	4.08 ± 0.20	35.89 ± 2.98
	MAML	2.49 ± 0.12	3.00 ± 0.35	4.75 ± 0.48	7.10 ± 0.77	22.89 ± 1.41
	NP	1.04 ± 0.06	1.26 ± 0.21	2.90 ± 0.35	5.45 ± 0.47	21.45 ± 1.3
	NeuBANP	0.86 ± 0.06	1.04 ± 0.08	**1.88** ± 0.14	**3.09** ± 0.23	**9.96** ± 0.70

Table 1. Results of the wheel bandit problem according to the value of δ. Mean and standard deviation for cumulative regret and simple regret over 50 runs are reported.

Bootstrapping Neural Networks Bootstrap method (Efron, 1987) is a reliable approach to estimate predictive uncertainty (Lakshminarayanan et al., 2017; Osband et al., 2016). However, it is computationally inefficient to go through the feed-forward computation as much as the number of bootstraps; hence, it discourages the practical application of bootstrap in neural networks. There have been several works to circumvent this issue by approximating bootstrapped distribution. Amortized bootstrap (Nalisnick & Smyth, 2017) approximates bootstrap distribution over model parameters by using amortized inference and implicit models. Generative Bootstrap Sampler (Shin et al., 2020) proposes a computational bootstrap procedure that constructs a generator function of bootstrap evaluations for classical statistical models. Neural Bootstrapper (Shin et al., 2021) suggests a simple recipe for generating bootstrapped predictive distributions of MLPs and convolutional neural networks.

Meta-Learning based Stochastic Optimization NPs are trained with a meta-learning framework to solve various tasks related to the data generation process through a single optimization. Santoro et al. (2016); Chen et al. (2017) follow the same training procedure of NP. However, Santoro et al. (2016) proposes an memory-augmented network for robust meta-learning, while Chen et al. (2017) proposes the method to produce an algorithm for black-box optimization using recurrent networks. Sharaf & Daumé III (2019) presents a meta-learning algorithm for learning a good exploration policy in the contextual bandit. Ravi & Beatson (2019) also solves contextual bandits based on the Bayesian framework by inferring a posterior on weights of neural networks. Galashov et al. (2019) introduces a unified framework for applying NP to a wide range of sequential decision-making problems.

6 Conclusion

We have proposed NeuBANP, a novel bootstrap method for a family of NP to model functional uncertainty appropriately. Instead of the standard bootstrap, NeuBANP learns to construct a generator function that produces valid bootstrapped distribution without resampling which can be considered as a *learn-to-bootstrap* method. NeuBANP successfully worked in a meta-learning framework, providing diverse trajectories of underlying data-generating processes, consistent to any given context. In addition, NeuBANP estimates the local uncertainty accurately, resolving overfitted prediction and variance overestimation problems observed in both ANP and BANP. We replace the additional layer and repetitive computations in BANP with the simple attachment of bootstrap weights to the model, which leads to lower computations and smaller memory.

NeuBANP shows superior performance to previous NP methods in regression and stochastic optimization tasks, including multi-dimensional setting, which has been the desired application of NP. However, due to the numerical instability of GP, there is a limit in sampling high-dimensional stochastic processes for the meta-learning framework. We suggest that this is a primary task for scalable applications of NP in stochastic optimization, as a challenging research direction.

7 REPRODUCIBILITY

For reproducibility of experimental results, we provide a link to anonymous github that contains our source code in Appendix B. The source code includes the implementation of our model, data generation, and experiments. Additionally, the data generation steps are thoroughly explained in Appendix B.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient Monte-Carlo bayesian optimization. In *Advances in Neural Information Processing Systems*, 2020.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, pp. 748–756. PMLR, 2017.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks*, pp. 2921–2926. IEEE, 2017.

Bradley Efron. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185, 1987.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Alexandre Galashov, Jonathan Schwarz, Hyunjik Kim, Marta Garnelo, David Saxton, Pushmeet Kohli, S.M. Ali Eslami, and Yee Whye Teh. Meta-learning surrogate models for sequential decision making. *arXiv preprint arXiv:1903.11907*, 2019.

Jacob Gardner, Geoff Pleiss, Kilian Q. Weinberger, David Bindel, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, volume 31, pp. 7576–7586, 2018.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S.M. Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018a.

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

Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2019.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2018.

Jungtaek Kim. Benchmark functions for bayesian optimization. https://github.com/jungtaekkim/bayeso-benchmarks, 2020.

Jungtaek Kim and Seungjin Choi. BayesO: A Bayesian optimization framework in Python. https://bayeso.org, 2017.

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

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical evaluation of neural process objectives. In *Advances in Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2018.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, pp. 3730–3738, 2015.

Christos Louizos, Xiahan Shi, Klamer Schutte, and Max Welling. The functional neural process. *arXiv preprint arXiv:1906.08324*, 2019.

Eric Nalisnick and Padhraic Smyth. The amortized bootstrap. In *International Conference on Machine Learning 2017 Workshop on Implicit Models*, 2017.

Michael A. Newton and Adrian E. Raftery. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1): 3–26, 1994.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *arXiv preprint arXiv:1602.04621*, 2016.

Jens Præstgaard and Jon A Wellner. Exchangeably weighted bootstraps of the general empirical process. *The Annals of Probability*, pp. 2053–2086, 1993.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2019.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2016.

Amr Sharaf and Hal Daumé III. Meta-learning for contextual bandit exploration. *arXiv preprint arXiv:1901.08159*, 2019.

Minsuk Shin, Lu Wang, and Jun S Liu. Scalable uncertainty quantification via generativebootstrap sampler. *arXiv preprint arXiv:2006.00767*, 2020.

Minsuk Shin, Hyungjoo Cho, Hyun-seok Min, and Sungbin Lim. Neural bootstrapper. In *Advances in Neural Information Processing Systems*, 2021.

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

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

Jaesik Yoon, Gautam Singh, and Sungjin Ahn. Robustifying sequential neural processes. In *International Conference on Machine Learning*, pp. 10861–10870. PMLR, 2020.

A MODEL

A.1 BASIC OPERATION

Muti-Layer Perceptron $\text{MLP}(d_i, d_h, d_o, n_l)$ denotes the multi-layer perceptron consisting of n_l linear transformations and ReLU activations between them. Let parameters d_i, d_h, d_o the dimension of input, hidden and output feature, repectively. We used same d_h for every hidden layers. $\text{Lin}(p, q)$ denotes the linear transformation of input with feature dimension p into output with feature dimension q.

$$\text{MLP}(d_i, d_h, d_o, n_l)(X) = \text{Lin}(d_h, d_o) \circ (\text{ReLU} \circ \text{Lin}(d_h, d_h))^{n_l - 2} \circ \text{ReLU} \circ \text{Lin}(d_i, d_h)(X) \tag{12}$$

Dot Product Attention DotProdAttn (Vaswani et al., 2017) denotes the attention operation with attention score based on cosine similarity. Dot product of query (Q) and key (K) tensors calculates the similarity of each vectors in query tensor relative to each vectors in key tensor. Attention score is calculated through softmax operation of normalized similiarity. Let d_k the feature dimension of key and query tensors, and d_v that of value tensors.

$$\text{DotProdAttn}(Q, K, V) = \text{softmax}\big(Q^T K / \sqrt{d_k}\big) V \tag{13}$$

Multi-Head Attention MHA denotes multi-head attention with dot product attention. The input is pre-processed with MLP and the output of attention is post-processed with layer normalization. For simplicity, we omit parameters of each MLP layers. $\text{split}(X, n)$ denotes splitting of tensor X with respect to feature axis into a tuple of n tensors $(X'_i)_{i=1}^n$ which have same dimension in feature axis. $[X_1, X_2, ..., X_n]$ denotes the concatenation of tensors $X_1, X_2, ..., X_n$ with respect to feature axis. LayerNorm denotes the layer normalization introduced in Ba et al. (2016). Let n_{head} the number of heads in multi-head attention.

$$(Q'_i)_{i=1}^{n_{head}} = \text{split}(\text{MLP}_{qk}(Q), n_{head}) \tag{14}$$

$$(K'_i)_{i=1}^{n_{head}} = \text{split}(\text{MLP}_{qk}(K), n_{head}) \tag{15}$$

$$(V'_i)_{i=1}^{n_{head}} = \text{split}(\text{MLP}_{v}(V), n_{head}) \tag{16}$$

$$\text{MHA}(Q, K, V) = \text{LayerNorm}([(\text{DotProdAttn}(Q'_i, K'_i, V'_i))_{i=1}^{n_{head}}]) \tag{17}$$

Self-Attention We used self-attention to calculate efficient representations of context $\mathcal{D}_\mathcal{C} = (X_\mathcal{C}, Y_\mathcal{C})$. We define self-attention based on multi-head attention as follows:

$$\mathcal{D}'_\mathcal{C} = [X_\mathcal{C}, Y_\mathcal{C}] = ([x_c, y_c])_{c \in \mathcal{C}}, \quad \text{SelfAttn}(\mathcal{D}_\mathcal{C}) := \text{MHA}(\mathcal{D}'_\mathcal{C}, \mathcal{D}'_\mathcal{C}, \mathcal{D}'_\mathcal{C}) \tag{18}$$

Cross-Attention We used cross-attention to calculate representation of context specific to target feature x in interest, when original representations $(h_c)_{c \in \mathcal{C}}$ is given. We define cross-attention based on multi-head attention as follows:

$$\text{CrossAttn}(x, X_\mathcal{C}, (h_c)_{c \in \mathcal{C}}) := \text{MHA}(x, X_\mathcal{C}, [(h_c)_{c \in \mathcal{C}}]). \tag{19}$$

A.2 ARCHITECTURE

For fair comparison, we used the same architecture of all models as in Lee et al. (2020). For NEUBANP, we increased the input dimension of SelfAttn in the posterior path by one to take bootstrap weight as additional input. Please refer to Lee et al. (2020) for detailed model architecture of ANP and BANP.

B EXPERIMENT

Implementation[1] of NPs other than NEUBANP was borrowed from the source code of Lee et al. (2020)[2]. Regression and Bayesian optimization experiment was done in single *GeForce RTX 2080 Ti* GPU with the memory of $11,019$ MiB. Multi-dimensional regression including image completion was done in *Tesla V100* GPU with the memory of $32,480$ MiB.

[1] https://anonymous.4open.science/r/neubanp_initial

[2] https://github.com/juho-lee/bnp, MIT License.

Method	RBF		Matérn 5/2		Periodic	
	context	target	context	target	context	target
CNP	1.17 $\pm$ 0.08	0.87 $\pm$ 0.36	1.06 $\pm$ 0.11	0.65 $\pm$ 0.39	-0.31 $\pm$ 0.41	-2.05 $\pm$ 1.17
NP	1.11 $\pm$ 0.09	0.78 $\pm$ 1.47	0.99 $\pm$ 0.11	0.56 $\pm$ 0.50	-0.28 $\pm$ 0.37	-1.73 $\pm$ 1.09
BNP	1.20 $\pm$ 0.07	0.92 $\pm$ 0.34	1.09 $\pm$ 0.09	0.72 $\pm$ 0.35	-0.18 $\pm$ 037	**-1.16** $\pm$ 0.56
ANP	1.38 $\pm$ 0.00	1.08 $\pm$ 0.41	1.38 $\pm$ 0.00	0.94 $\pm$ 0.47	0.21 $\pm$ 0.76	-6.82 $\pm$ 2.83
BANP	1.38 $\pm$ 0.00	1.12 $\pm$ 0.33	1.38 $\pm$ 0.00	0.99 $\pm$ 0.38	0.28 $\pm$ 0.69	-5.69 $\pm$ 2.37
NeuBANP	**3.17** $\pm$ 0.28	**1.38** $\pm$ 0.60	**3.09** $\pm$ 0.29	**1.13** $\pm$ 0.64	**1.56** $\pm$ 0.73	-11.49 $\pm$ 8.08

Table 2. Log-Likelihood of NPs for 48,000 different evaluations of context and target.

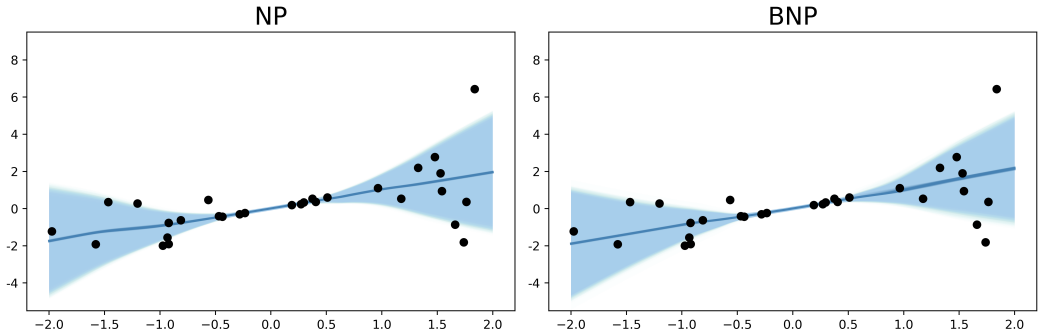

Fig. 6. Each plot shows predictions given by NP, BNP in a linear regression. The ground-truth function is a simple linear function with heterogeneous variance: $y = x + \beta\epsilon(x)$ where $\epsilon(x) \sim N(0, \sigma^2(x))$ and $\sigma(x) = \sqrt{x^2 + 10^{-5}}$. We used the official code provided by Lee et al. (2020). Unlike the case of ANP and BANP, considering that overfitting does not occur in NP and BNP, we can notice that attention modules are the leading cause of overfitting.

B.1 1D Regression

Training For all models, training dataset consists of randomly sampled context and target from functions following GP with RBF kernel $k(x, y) = s^2 \cdot \exp(-||x - y||^2/(2l^2))$. Parameters of kernel are randomly sampled with $s \sim \text{Uniform}(0.1, 1.0)$ and $l \sim \text{Uniform}(0.1, 0.6)$. Feature values $(x_i)_{i \in \mathcal{C} \cup \mathcal{T}}$ is chosen uniformly at random in $[-2, 2]$. The size of context and target are randomly sampled with $|\mathcal{C}| \sim \text{Uniform}(3, 47)$ and $|\mathcal{T}| \sim \text{Uniform}(3, 50 - |\mathcal{C}|)$. We trained all models for 100,000 iterations and used Adam optimizer (Kingma & Ba, 2015). For stable learning, we used the cosine annealing scheduler with initial learning rate 5×10^{-4}.

Additional Results Table 2 shows log-likelihood of NPs for various evaluation dataset sampled from GP with RBF, Matérn 5/2, Periodic kernel. As in generation of training dataset, parameters of Matérn 5/2 kernel $k(x, y) = s^2(1 + \sqrt{5}||x - y||^2/(3l^2)) \exp(-\sqrt{5}||x - y||/l)$ and Periodic kernel $k(x, y) = s^2 \exp(-2 \sin^2(\pi||x - y||^2/p)/l^2)$ was randomly sampled with $s \sim \text{Uniform}(0.1, 1.0)$, $l \sim \text{Uniform}(0.1, 0.6)$ and $p \sim \text{Uniform}(0.1, 0.5)$. For RBF and Matérn 5/2 dataset, NeuBANP showed state-of-the-art performance both in fitting context and predicting target.

B.2 Bayesian Optimization

One-dimensional case Table 3 shows the performance of GP and various NPs for 1D Bayesian optimization task. RBF, Matérn 5/2, and Periodic kernels are used to generate evaluation dataset.

Multi-dimensional case In multi-dimensional BO experiments, we used `GPyTorch` [3] (Gardner et al., 2018) for scalable GP regression, and `BoTorch` [4] (Balandat et al., 2020) for overall BO process

[3] https://github.com/cornellius-gp/gpytorch, MIT License.
[4] https://github.com/pytorch/botorch, MIT License.

Algorithm 1: Neural Process based Bayesian Optimization.

Input : Target function $f^\star$; Acquisition function $\mathcal{U}$; Observed data $\mathcal{D}_0 = \{(x_0, f^\star(x_0))\}$; Maximum evaluation step T.

Meta-train a neural process p_θ on $f \sim \mathcal{P}(\mathcal{F})$.

for $t = 1, \ldots, T$ **do**

  Find x_t by optimizing acquisition function: $x_t = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, \mathcal{U}\left(p_\theta(y|x, \mathcal{D}_{t-1})\right)$

  Evaluate $f^\star(x_t)$ and update the observed data: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(x_t, f^\star(x_t))\}$

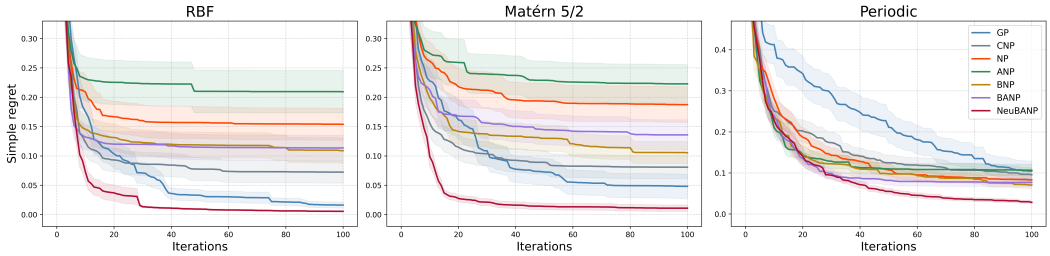

Fig. 7. 1D Bayesian optimization results. Bold lines show the mean of simple regrets over 100 experiments. We also report 10% of the standard deviation.

(e.g., optimization of acquisition functions). GP was set to the default setting of `BoTorch`. In detail, GP model was parameterized with Matérn 5/2 kernel with ARD and constant mean function, and prior distribution for hyperparameters was set as Gamma$(3, 6)$ for length scale l and Gamma$(2, 0.15)$ for output scale s. For three-dimensional BO experiment in Figure 4, the overall time complexity of ANP and BANP is almost the same. This result is seemingly in contrast with the fact that ANP takes a shorter time for prediction than BANP. However, since the BO algorithm contains the optimization of the acquisition function, the qualities of acquisition functions obtained by the model predictions may affect the overall time complexity. We conjecture that BANP gives an acquisition function easier to optimize than that of ANP so that the overall time complexities of both models are similar. Additionally, we did not report the results of GP for the two functions. Specifically, we omitted the Goldstein-Price since the simple regret value of GP was too large (the performance was poor) compared to other models and omitted the Rastrigin since the 46 errors occurred out of 100 experiments. An error may occur when ill-conditioned data is given during the kernel training process of the GP, and the data recommended by the UCB during the exploration (or exploitation) process seems to correspond to this condition. If we report the performance ignoring the numerical error, the simple regret for the Goldstein-Price was 52049.13, and the simple regret for the Rastrigin was 18.48. For the Rastrigin function, GP with UCB is numerically unstable, but like EI, it achieved the best performance.

Method	RBF	Matérn 5/2	Periodic
GP (RBF)	0.016 $\pm$ 0.052	0.048 $\pm$ 0.206	0.104 $\pm$ 0.242
CNP	0.072 $\pm$ 0.188	0.081 $\pm$ 0.198	0.096 $\pm$ 0.166
NP	0.154 $\pm$ 0.273	0.187 $\pm$ 0.303	0.083 $\pm$ 0.121
ANP	0.209 $\pm$ 0.364	0.223 $\pm$ 0.328	0.107 $\pm$ 0.142
BNP	0.109 $\pm$ 0.214	0.105 $\pm$ 0.188	0.071 $\pm$ 0.091
BANP	0.114 $\pm$ 0.216	0.136 $\pm$ 0.256	0.077 $\pm$ 0.11
NeuBANP	**0.006** $\pm$ 0.011	**0.011** $\pm$ 0.055	**0.028** $\pm$ 0.035

Table 3. 1D Bayesian optimization results. Mean and standard deviations of simple regrets over 100 runs are reported.

Dim	Target	GP	ANP	BNP	BANP	NeuBANP
2D	Ackley	2.84 $\pm$ 1.82	0.19 $\pm$ 0.53	1.82 $\pm$ 1.03	0.50 $\pm$ 1.03	**0.08** $\pm$ 0.27
	Dropwave	0.36 $\pm$ 0.17	0.22 $\pm$ 0.15	0.32 $\pm$ 0.18	0.28 $\pm$ 0.17	**0.15** $\pm$ 0.10
	Goldsteinprice	-	475.52 $\pm$ 469.15	2098.22 $\pm$ 1509.88	80.67 $\pm$ 65.85	**30.33** $\pm$ 25.42
	Michalewicz	0.67 $\pm$ 0.45	0.61 $\pm$ 0.40	1.00 $\pm$ 0.46	0.69 $\pm$ 0.38	**0.45** $\pm$ 0.42
3D	Ackley	3.39 $\pm$ 1.25	5.36 $\pm$ 0.97	3.29 $\pm$ 1.11	4.30 $\pm$ 1.15	**0.34** $\pm$ 0.26
	Cosine	0.04 $\pm$ 0.24	0.10 $\pm$ 0.10	1.12 $\pm$ 0.57	0.25 $\pm$ 0.43	**0.005** $\pm$ 0.003
	Hartmann	0.42 $\pm$ 0.77	**0.33** $\pm$ 0.50	1.94 $\pm$ 0.86	0.93 $\pm$ 0.98	0.39 $\pm$ 0.39
	Rastrigin	-	54.94 $\pm$ 19.84	48.55 $\pm$ 14.34	38.36 $\pm$ 8.20	**23.06** $\pm$ 19.77

Table 4. Multi-dimensional Bayesian optimization results. Mean and standard deviations of simple regrets over 100 runs are reported.

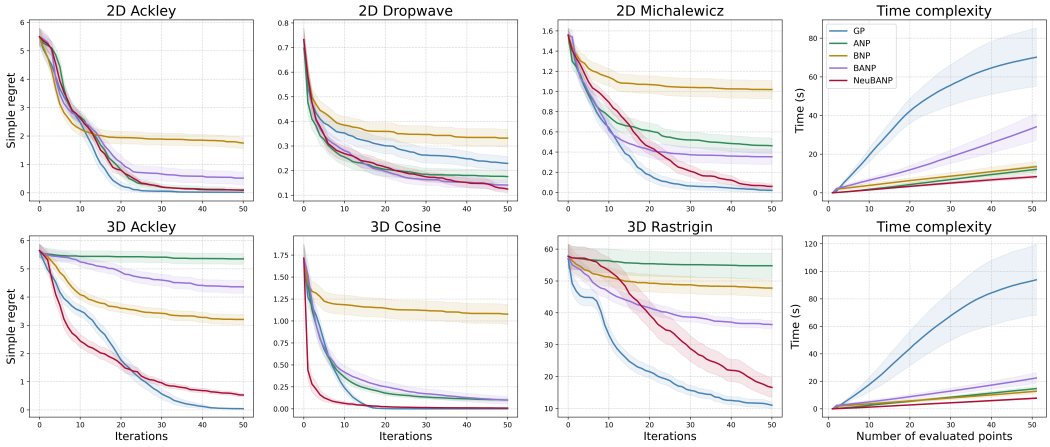

Fig. 8. Left: Multi-dimensional Bayesian optimization results on various benchmark functions with EI as an acquisition function. Bold lines represent the mean performance over 100 experiments. We indicate 20% of the standard deviation. **Right most:** Time complexity of each model as the number of observations increases.

B.3 Contextual Multi-Armed Bandit

Setting We followed wheel bandit setting introduced in Riquelme et al. (2018). At every step $t\,(< T)$, a two-dimensional point (x_t, y_t) inside the unit circle $R_{\texttt{unit}} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$ is given as a context c_t. The algorithm chooses an action $a_t \in \{1, 2, 3, 4, 5\}$. The stochastic reward $r_t = r(a_t, c_t)$ is sampled from the reward distribution. Let $R_1, R_2, R_3, R_4, R_5 \subset R_{\texttt{unit}}$ the disjoint sets (regions) of unit circle as follows:

$$R_1 = \{(x, y) : x^2 + y^2 < \delta\} \tag{20}$$

$$R_2 = \{(x, y) : \delta \leq x^2 + y^2 \leq 1, x > 0, y > 0\} \tag{21}$$

$$R_3 = \{(x, y) : \delta \leq x^2 + y^2 \leq 1, x < 0, y > 0\} \tag{22}$$

$$R_4 = \{(x, y) : \delta \leq x^2 + y^2 \leq 1, x < 0, y < 0\} \tag{23}$$

$$R_5 = \{(x, y) : \delta \leq x^2 + y^2 \leq 1, x > 0, y < 0\} \tag{24}$$

where the constant δ determines the size of R_1 relative to other regions. Each action results in rewards following different distribution according to the region to which the given context belongs, where $\mathcal{N}$ denotes the normal distribution.

$$r(1, c) \sim \mathcal{N}(1.2, 0.01^2) \tag{25}$$

$$r(a, c) \sim \begin{cases} \mathcal{N}(50, 0.01^2), & \text{if } c \in R_a \\ \mathcal{N}(1, 0.01^2), & \text{otherwise} \end{cases} \quad \forall a \in \{2, 3, 4, 5\} \tag{26}$$

Note that action $\{1\}$ always produces a moderate reward, but the other actions $\{2, 3, 4, 5\}$ sometimes produce a very high reward when the context is sampled from the corresponding high-reward region.

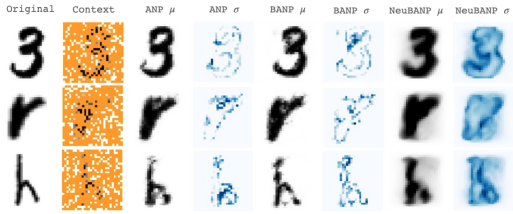

Fig. 9. Qualitative result of EMNIST image completion.

Method	Seen classes (0-9)		Unseen classes (10-46)	
	context	target	context	target
CNP	0.926 ± 0.007	0.751 ± 0.005	0.766 ± 0.009	0.498 ± 0.012
NP	0.948 ± 0.006	0.806 ± 0.005	0.808 ± 0.005	0.600 ± 0.009
ANP	1.383 ± 0.000	0.993 ± 0.005	**1.383** ± 0.000	0.894 ± 0.004
BNP	1.004 ± 0.008	0.880 ± 0.005	0.883 ± 0.010	0.722 ± 0.006
BANP	1.383 ± 0.000	1.010 ± 0.006	1.382 ± 0.000	0.942 ± 0.005
NeuBANP	**1.475** ± 0.345	**1.337** ± 0.224	1.333 ± 0.516	**1.119** ± 0.388

Table 5. Quantitative result of EMNIST image completion. Mean and standard deviationd of likelihoood over 5 experiments.

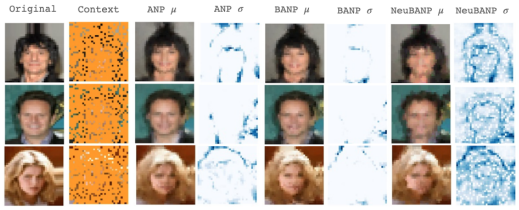

Fig. 10. Qualitative result of CelebA image completion.

	context	target
CNP	2.975 ± 0.013	2.199 ± 0.003
NP	3.066 ± 0.011	2.492 ± 0.014
ANP	4.150 ± 0.000	2.731 ± 0.006
BNP	3.269 ± 0.008	2.788 ± 0.005
BANP	4.149 ± 0.000	**3.129** ± 0.005
NeuBANP	**13.946** ± 0.590	2.870 ± 0.021

Table 6. Quantitative result of CelebA results. Mean and standard deviationd of likelihoood over 5 experiments.

Thus, learning different reward distributions for actions $\{2, 3, 4, 5\}$ by apprehension of context information is critical to bandit performance. As δ increases, the high-reward regions for each actions become smaller. Then the model should learn from rare observation of such high reward, which means the problem becomes more difficult.

Training and Evaluation When pre-training NeuBANP, as in Garnelo et al. (2018b), 8 training batches of 512 contexts and 50 targets were generated from the environment with hyperparameter randomly sampled; $\delta \sim \text{Uniform}(0, 1)$. We consider two-dimensional context point c_i as feature $x_i \in \mathbb{R}^2$ and five rewards for actions $(r(1, c_i), r(2, c_i), r(3, c_i), r(4, c_i), r(5, c_i))$ as label $y_i \in \mathbb{R}^5$. At evaluation, only rewards for chosen actions are observed by the model. Thus, we replace other unobserved rewards with *dummy* values randomly sampled from $\mathcal{N}(0, 1)$, following the usual strategy.

B.4 Image Completion

Settings We compared the baseline NPs and NeuBANP on image completion tasks. Following Lee et al. (2020), we trained all models on EMNIST (Cohen et al., 2017) and CelebA (Liu et al., 2015) which was resized to 32×32. For EMNIST, we used only 10 classes for training and reported the evaluation results for both seen classes and unseen classes separately. NeuBANP was trained with 10 samples, and the other baselines were trained with 4 samples. For evaluation, we used 50 samples for all methods. Similar to 1D regression experiment, we randomly select the pixels of a given image as context/target, and the number of context/target were drawn randomly from Uniform distribution. However, in this case, we increased the maximum number of given points; i.e., $|\mathcal{C}| \sim \text{Uniform}(3, 197)$, $|\mathcal{T}| \sim \text{Uniform}(3, 200 - |\mathcal{C}|)$. x values were rescaled to $[-1, 1]$ and the corresponding y values were rescaled to $[-0.5, 0.5]$. We trained all models for 200 epochs and set a initial learning rate of 5×10^{-4} using the Adam optimizer (Kingma & Ba, 2015) with cosine annealing scheduler for learning rate decay.

Results Figure 9 and 10 show the mean prediction and uncertainty estimation of ANP, BANP, and NeuBANP for test images in unseen classes. For both datasets, though our model shows noisy mean prediction due to the random weights, we can demonstrate the advantage of NeuBANP in uncertainty estimation. NeuBANP estimated the uncertainty correctly in the area where the color of the pixel changes and thus possesses significant uncertainty. This leads to the overall improvement in quantitative performance (see Table 5 and 6).