Neural Process for Black-Box Model Optimization Under Bayesian Framework

Zhongkai Shangguan,[1] Lei Lin,[2] Wencheng Wu,[2] Beilei Xu[2]

[1 2]Rochester Data Science Consortium, University of Rochester, Rochester, NY 14604 USA
[1]zshangg2@ur.rochester.edu [2]{Lei.Lin, Wencheng.Wu, Beilei.Xu}@rochester.edu

Abstract

There are a large number of optimization problems in physical models where the relationships between model parameters and outputs are unknown or hard to track. These models are named as "black-box models" in general because they can only be viewed in terms of inputs and outputs, without knowledge of the internal workings. Optimizing the black-box model parameters has become increasingly expensive and time consuming as they have become more complex. Hence, developing effective and efficient black-box model optimization algorithms has become an important task. One powerful algorithm to solve such problem is Bayesian optimization, which can effectively estimates the model parameters that lead to the best performance, and Gaussian Process (GP) has been one of the most widely used surrogate model in Bayesian optimization. However, the time complexity of GP scales cubically with respect to the number of observed model outputs, and GP does not scale well with large parameter dimension either. Consequently, it has been challenging for GP to optimize black-box models that need to query many observations and/or have many parameters. To overcome the drawbacks of GP, in this study, we propose a general Bayesian optimization algorithm that employs a Neural Process (NP) as the surrogate model to perform black-box model optimization, namely, Neural Process for Bayesian Optimization (NPBO). In order to validate the benefits of NPBO, we compare NPBO with four benchmark approaches on a power system parameter optimization problem and a series of seven benchmark Bayesian optimization problems. The results show that the proposed NPBO performs better than the other four benchmark approaches on the power system parameter optimization problem and competitively on the seven benchmark problems.

Introduction

A complex physical model can be viewed as a black-box model since it can only be viewed in terms of its inputs (parameters) and outputs (observations) when internal workings are hard to track. Optimizing the parameters of such a black-box model is a common problem in engineering fields (Xiao et al. 2015; Cassioli 2013; Zhang and Zhang 2010). For examples, power system models need regular calibrations to reflect the current power system status, identify and mitigate potential issues (Lin et al. 2020); geometric structure parameter optimization of an antenna aims at reaching an optimal gain in working frequency band (Gustafsson 2016). In many cases, this optimization process is done manually by experts based on a series of experiments. However, this process requires expert domain knowledge and a number of experiments to be conducted, which can be both expensive and time consuming. In order to reduce human effort and the number of required experiments, automated optimization algorithms with varying computational complexity and scalability have been proposed.

Conventional automated black-box model optimization algorithms include grid search and random search (Bergstra and Bengio 2012). In grid search, the problem is defined in a high-dimensional grid space where each grid dimension corresponds to a parameter, and each grid point corresponds to a parameter combination. We then evaluate the model on all parameter combinations defined by the grids, and select the parameter combination that yields the best performance of the model. One drawback of grid search is that the number of grid points grows exponentially as the number and value range of the parameters increase. On the other hand, random search approach can potentially explore the parameter space more extensively through randomly generates parameter combinations in the parameter space. However, both of these two methods do not utilize the prior sampled information, so that the search is blind. Recently, more advanced automated optimization algorithms have been introduced, including evolutionary optimization (Cheng 2018), population-based optimization (Jaderberg et al. 2017), and Bayesian optimization (Frazier 2018). These algorithms construct the relationship between parameter combinations and the performance of black-box models, and provide guidance for the next selection of parameter combination for evaluation, which make those methods more efficient to find the global optimal parameter combination.

In this paper, we focus on the Bayesian optimization framework. Bayesian optimization implements a surrogate model that predicts the black-box model performance for a specific parameter combination; and an acquisition function, which trades off exploration and exploitation to query the next observation point. An accurate surrogate model that can predict the black-box model performance as well as measure

the uncertainty is crucial to the performance of Bayesian optimization. Gaussian process (GP) has been the most widely used surrogate model for Bayesian optimization (Rasmussen and Williams 2006) due to its expressiveness, smoothness and well-calibrated uncertainty estimation of the model response. However, GP has $\mathcal{O}(N^3)$ time complexity, where N is the number of training samples (Rasmussen and Williams 2006), so it is computationally expensive. Another drawback of GP is its poor scalability to high parameter dimensions (Liu et al. 2020), i.e., with the increase in the dimension of parameters, the performance of GP becomes worse. Hence, GP cannot be applied to problems that require to query many observations and/or have many parameters.

To overcome these issues, we propose a new algorithm for black-box model optimization by employing Neural Process as a powerful and scalable surrogate model under the Bayesian optimization framework, namely, Neural Process for Bayesian Optimization (NPBO). Neural Process (NP) (Garnelo et al. 2018b) is an algorithm to simulate stochastic process using neural networks (NNs). It combines the advantages of stochastic process and NNs so that it has the ability to capture uncertainty while predicting the black-box model's performance accurately. The performance of the proposed algorithm is evaluated on a power system parameter optimization problem and seven benchmark problems for Bayesian optimization (Klein et al. 2017). The results show that the proposed NPBO outperforms the other four benchmark approaches including GP-based Bayesian optimization, random forest based Bayesian optimization, Deep Networks for Global Optimization (DNGO) and Bayesian Optimization with Hamiltonian Monte Carlo Artificial Neural Networks (BOHAMIANN) on the power system parameter optimization problem and performs competitively on the benchmark problems.

The rest of the paper is arranged as follows. **Related Work** introduces state-of-the-art methods in parameter optimization under the Bayesian optimization framework. The methodology of NPBO is introduced in detail in **Methodology**. We compare and discuss the results of NPBO and the benchmark algorithms in the **Experiment** section. Finally, the paper concludes our work and discusses future steps.

Related Work

This section will introduce the framework of Bayesian optimization using GP, then discuss previous proposed approaches (Snoek et al. 2015; Springenberg et al. 2016) that use NNs as alternative surrogate models.

Bayesian Optimization Framework

Bayesian optimization is a state-of-the-art optimization framework for black-box model optimization problems. It has great power in parameter optimization of physical models (Duris et al. 2020; Muehleisen and Bergerson 2016) and hyperparameter optimization in training machine learning (ML) models (Chen et al. 2018). A detailed tutorial can be found in (Archetti and Candelieri 2019).

The aim of Bayesian optimization is to find the input $\boldsymbol{x}$ that maximizes an unknown nonlinear function $f(\boldsymbol{x}) : R^d \rightarrow R$. It can be formulated as (1):

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in A \subset R^d}{\operatorname{argmax}} f(\boldsymbol{x}), \tag{1}$$

where d is the dimension of $\boldsymbol{x}$, A is the constrain that defined for $f(\boldsymbol{x})$, $f(\cdot)$ is the nonlinear function that is expensive to evaluate, and $\hat{\boldsymbol{x}}$ is the estimation of the input parameter. One assumption of Bayesian optimization is that only the outputs $f(\boldsymbol{x})$ can be observed while its derivatives cannot be obtained. Hence, $f(\cdot)$ is a black-box model and the optimization problem cannot be solved using gradient descent algorithm. Bayesian optimization repeatedly executes the following steps until a satisfactory input parameter combination is found: (i) fit a surrogate model to the current observations to get a prior distribution; (ii) convert the prior to the posterior distribution and predict where the next input parameter combination is by maximizing an acquisition function; (iii) obtain the observation on the suggested parameter combination and add the result to the observation set. The third step is usually the most expensive step since it needs to generate the observation on the expensive black-box model.

There are two main components in Bayesian optimization: a surrogate model that simulates the black-box model and an acquisition function that trades off exploration and exploitation in order to decide the next query inputs. GP is a classical model that is widely employed as the surrogate model by Bayesian optimization. A GP is defined as a stochastic process indexed by a set $\mathcal{X} \subseteq \mathcal{R}^d$: $\{f(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$ such that any finite number of random variables of the process has a joint Gaussian distribution. Instead of inferring a distribution over the parameters, GP can be used to infer a distribution over the function directly. For example, considering we sample a finite set $D = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, $D \in \mathcal{X}$, GP is completely defined by its mean and covariance functions as (2) shows

$$p(\boldsymbol{f}|D) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K}) \tag{2}$$

where $\boldsymbol{f} = (f(\boldsymbol{x}_1), ...f(\boldsymbol{x}_n))$ is the distribution over the black-box model; $\boldsymbol{\mu} = (m(\boldsymbol{x}_1), ..., m(\boldsymbol{x}_n))$ where m is the mean function, and $\boldsymbol{K} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ represents the covariance function (also known as kernel) such as Radial basis function kernel (Görtler, Kehlbeck, and Deussen 2019). So for a specific $\boldsymbol{x}$, GP predicts a mean and a variance that completely define a Gaussian distribution over $f(\boldsymbol{x})$, i.e., $f(\boldsymbol{x}) \sim \mathcal{N}(f(\boldsymbol{x})|m(\boldsymbol{x}), K(\boldsymbol{x}))$.

Expected improvement (EI) is one of the popular acquisition functions in Bayesian optimization (Frazier 2018). EI is computed as the expectation taken with respect to the posterior distribution. It is defined as:

$$\alpha_{EI}(\boldsymbol{x}) := (\mu(\boldsymbol{x}) - \tau)\Phi\left(\frac{\mu(\boldsymbol{x}) - \tau}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{\mu(\boldsymbol{x}) - \tau}{\sigma(\boldsymbol{x})}\right) \tag{3}$$

where Φ and ϕ represent the cumulative distribution function and probability distribution function of standard normal distribution, τ is the current best observed outcome, μ and σ are the mean and variance.

Bayesian Optimization with Neural Networks

To address the downsides of the standard GP, i.e., scaling cubically with the number of queried observations and not performing well with high dimensional data, alternative methods based on sparse GP approximations (Snelson and Ghahramani 2006; Lázaro-Gredilla et al. 2010; McIntire, Ratner, and Ermon 2016) have been proposed. These methods approximate the full GP by using subsets of the original dataset as inducing points to build a covariance function. However, they are not accurate in uncertainty estimation and still have poor scalability in high dimensional parameter space (Hebbal et al. 2019). Since NNs are very flexible and scalable, adapting NNs to the Bayesian optimization framework is highly desirable. However, NNs are deterministic models that do not have the ability to measure the uncertainty. Hence, combining the flexibility and scalability of NNs with well-calibrated uncertainty estimations is crucial in this procedure.

More recently, Bayesian optimization methods based on NNs have been proposed. Snoek et al. (2015) propose a Deep Networks for Global Optimization (DNGO) framework that uses NNs as the feature extractor to pre-process the inputs, and then adapts a Bayesian linear regression (BLR) model to gain the uncertainty. Specifically, the authors first train a deterministic neural network with fully-connected-layers, where the output of the penultimate layer is regarded as many basis functions and the final layer is regarded as only a linear combination of these basic functions. Then they freeze the parameters of neural network and feed the basis functions generated by the penultimate layer to a probabilistic model, i.e., BLR, to measure the uncertainty. This method successfully embeds neural networks into a probability model but uncertainty measurement only takes a small part in the whole procedure, which make it not perform well in uncertainty estimation.

Springenberg et al. (2016) apply a Bayesian Neural Network (BNN) as the surrogate model and train it with stochastic gradient Hamiltonian Monte in the Bayesian optimization framework (BOHAMIANN). In detail, they treat the weights of a neural network as a probability distribution so as to model it in a probabilistic approach. In inference, they sample several models from the probabilistic model to acquire the uncertainty. This method combines the advantages of neural networks and Bayesian models, but in the inference stage, it need to sample several times to obtain the uncertainty. Hence, if only a few models are sampled, the uncertainty estimation will not be accurate; on the other hand, sampling too many models will make the algorithm time consuming.

Methodology

In this section, we show how NP can be used as the surrogate model of Bayesian optimization. We first summarize the general formalism behind NP, then derive the training process for the proposed NPBO algorithm.

Neural Process for Bayesian Optimization

NP is a neural network based approach to represent a distribution over functions. It builds neural networks to model functions as a random process f. Given a set of observations $((x_1, y_1), ..., (x_N, y_N))$, NP first defines $\rho_{x_{1:N}}$ as the marginal distribution of $(f(x_1), ..., f(x_N))$, i.e.

$$\rho_{x_{1:N}}(y_{1:N}) := \rho_{x_1,...,x_N}(y_1, ..., y_N) \tag{4}$$

Assuming that the independence among samples and observation noise exists, i.e. $y_i = f(x_i) + \varepsilon_i \sim \mathcal{N}(f(x_i), \sigma^2(x_i))$, in order to build a model on (4), the conditional distribution can be written as

$$p(y_{1:N}|x_{1:N}, f) = \prod_{i=1}^{N} \mathcal{N}(y_i|f(x_i), \sigma^2(x_i)) \tag{5}$$

where p denotes the probability distribution. Then, in order to build the stochastic process f using neural network, we assume $f(x) = g(x, \boldsymbol{z})$, where $\boldsymbol{z}$ is a latent vector that is sufficient to represent f, and g is a general function defined for f. We assume $p(\boldsymbol{z})$ obeys a multivariate standard normal, then the random process f becomes sampling of $\boldsymbol{z}$. For example, assuming f is a GP, then $\boldsymbol{z}$ should be a vector containing the mean and variance that can fully define the GP. Hence, a NN can be employed to output $\boldsymbol{z}$ so as to model the random process f. Replacing $f(x)$ with $g(x, \boldsymbol{z})$, we have

$$p(\boldsymbol{z}, y_{1:N}|x_{1:N}) = p(z) \prod_{i=1}^{N} \mathcal{N}(y_i|g(x_i, \boldsymbol{z}), \sigma^2(x_i)) \tag{6}$$

Modeling and Training

To model the distribution over random functions defined by (6), we build a NPBO that includes three components: a probabilistic encoder, a deterministic encoder, and a decoder. The mean function is applied after both probabilistic encoder and deterministic encoder by adding the vectors extracted from each encoder together and take the average. The overall network architecture in our implementation of NP is shown in Fig. 1. As it can be seen, the probabilistic encoder is used to generate $\boldsymbol{z}$ that can be used to define the random process f. In this procedure, we use a NN to output μ and σ, which are used to build the multivariate normal distribution, then sample $\boldsymbol{z}$ from this normal distribution. The deterministic encoder with outcome $\boldsymbol{r}$ is to help improve the model stability. Finally, the decoder takes x, $\boldsymbol{r}$ and $\boldsymbol{z}$ as the inputs and predicts $\hat{y} = f(x)$.

NP modify the evidence lower-bound (ELBO) (Yang 2017) to design the loss function. ELBO is given by:

$$\begin{aligned} &\log p(y_{1:N}|x_{1:N}) \\ &\geq E_{q(\boldsymbol{z}|x_{1:N}, y_{1:N})} \left[\sum_{i=1}^{N} \log p(y_i^*|f_{\boldsymbol{z}}(x_i)) + \log \frac{p(\boldsymbol{z})}{q(\boldsymbol{z}|x_{1:N}, y_{1:N})} \right] \end{aligned} \tag{7}$$

where $q(\boldsymbol{z}|x_{1:N}, y_{1:N})$ is a posterior of the latent vector $\boldsymbol{z}$.

In the training process, we split the data in each batch $\{x_1, ..., x_n\}$ into two sets: context points $(X_c, Y_c) = \{(x_1, y_1), ..., (x_m, y_m)\}$ and target points $(X_t, Y_t) = \{(x_{m+1}, y_{m+1}), ..., (x_n, y_n)\}$. The context points are fed to

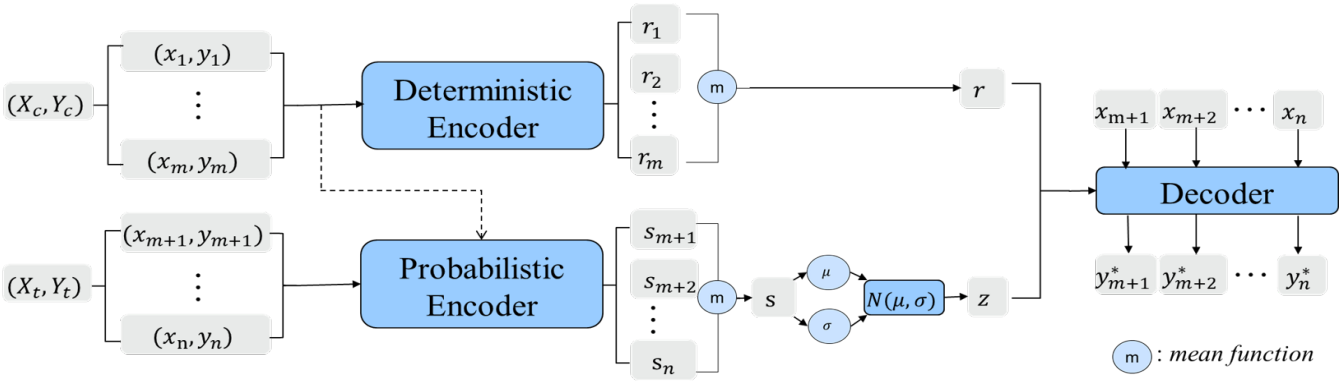

Figure 1: Flowchart of Neural Process.

the deterministic encoder to produce the global representation of $\boldsymbol{r}$ and the target points are fed to the probabilistic encoder to generate $\boldsymbol{s}$ that generates $\boldsymbol{z}$. In this process, we assume that we only have the information of the context points, and want to predict the target points. After this data splitting process, ELBO becomes:

$$\begin{aligned}&\log p(Y_t|X_t, X_c, Y_c)\\ &\geq E_{q(\boldsymbol{z}|X_t,Y_t)}\left[\sum_{i=m+1}^{n}\log p(y_i^*|f_{\boldsymbol{z}}(x_i)) + \log\frac{p(\boldsymbol{z}|X_c,Y_c)}{q(\boldsymbol{z}|X_t,Y_t)}\right]\end{aligned} \quad (8)$$

The final loss function is modified from the ELBO defined by (8). Because it is impossible to estimate the prior distribution of latent vector, i.e. $p(\boldsymbol{z})$, so we use the posterior of context points $q(\boldsymbol{z}|X_c, Y_c)$ to approximate $p(\boldsymbol{z}|X_c, Y_c)$:

$$\begin{aligned}&\log p(Y_t|X_t, X_c, Y_c)\\ &\geq E_{q(\boldsymbol{z}|X_t,Y_t)}\left[\sum_{i=m+1}^{n}\log p(y_i^*|f_{\boldsymbol{z}}(x_i)) + \log\frac{q(\boldsymbol{z}|X_c,Y_c)}{q(\boldsymbol{z}|X_t,Y_t)}\right]\end{aligned} \quad (9)$$

We use the lower bound of (9) as the loss function. In order to obtain $q(\boldsymbol{z}|X_c, Y_c)$, the context points will be fed into probabilistic encoder for only the forward process. Note that (9) contains two terms. The first is the expected log-likelihood over the target points. To calculate the first term, we need the context parameter r and a sample from the latent space $z \sim q(z|X_t, Y_t)$, then feed $\boldsymbol{z}$, $\boldsymbol{r}$ and X_t to the decoder to get the prediction of the model performance as well as its uncertainty. The second term evaluates the approximate negative Kullback–Leibler (KL) divergence (Joyce 2011) between $q(\boldsymbol{z}|X_c, Y_c)$ and $q(\boldsymbol{z}|X_t, Y_t)$, since we replace the prior of $\boldsymbol{z} \sim p(\boldsymbol{z})$ to the posterior $\boldsymbol{z} \sim q(\boldsymbol{z}|X_c, Y_c)$. In the inference process, we use all points observed as the context points, and z is generated using this context points instead of target points; then perform the forward step of the training process to get the prediction which will be fed to Equ. (3) to determine the next point to query.

Similar to GP, NP models distributions over functions and provides uncertainty estimations. Therefore, it is very suitable to be applied as the surrogate model under the Bayesian optimization framework. In other words, NP learns an implicit kernel from the data directly, which reduces the human effort to design the kernel function in GP, and leads to uncertainty estimations over unknown parameters. NP also combines benefits of neural networks so that is scaled linearly with respect to the number of observations.

Experiments

In this section, we compare the performance of NPBO on a power system parameter optimization problem with random search, GP-based Bayesian optimization, DNGO and BOHAMIANN. We further compare the NPBO with benchmark problems including GP-based Bayesian optimization, Random forest based Bayesian optimization, DNGO and BOHAMIANN on seven Bayesian optimization benchmark problems.

Parameter Optimization for Power System

Accurate and validated machine models are essential for reliable and economic power system operations. Machine models need to be regularly calibrated to ensure their accuracy for planning purpose and real-time operation (Huang et al. 2017). In recent years, power generation is facing substantial changes to its power grid with increasing additions of renewable energy sources and generators. Consequently, it is critical to the system operators to have efficient calibration methods and tools in order to reduce the time and effort required in machine calibration. To test our proposed NPBO method, an IEEE 14-bus system (Yk 2020) with a 14-parameter generator model, namely the ROUND ROTOR GENERATOR MODEL (GENROU) shown in Fig.2, is simulated using the power system simulation tool, PSS®E (Weber 2015). The simulator takes the 14-dimensional parameter of a generator model as the input, where their physical meanings are shown in Table 1, to generate a 4-dimensional output measured on the bus terminal that is connected to the target generator, i.e., Bus Voltage Magnitude (voltage), Bus Voltage Frequency (frequency), Real Power Injection (P) and Reactive Power Injection (Q). To reduce the parameter dimension, the Design of Experiments (DOE), a well-established statistical approach, has been applied to select a subset of four out of 14 parameters (Gunawan, Lau et al. 2011). That is, our goal is to optimize the selected four parameters so that the simulated output matches the target observations. The ranges of input parameters to be optimized

Table 1: Input parameters and their physical meaning

$T'do$	d-axis OC Transient time constnat
$T''do$	Subtransient
$T'qo$	Transient
$T''qo$	Subtransient
H	H Inertia
D	D speed Daping
Xd	Direct Axis Reactance
Xq	Quadrature Axis Reactance
$X'd$	Direct Axis Transient Reactance
$X'q$	Quadrature Axis Transient Reactance
$X''d/X''q$	Subtransient Reactance
XI	Leakage Reactance
$S(1.0)$	Saturation First Point
$S(2.0)$	Saturation Second Point

Table 2: Input and Output range of the power system

		Min	Max
Input	$T'do$	5.625	9.375
	Xd	1.425	2.375
	Xq	1.35	2.25
	$X'd$	0.315	0.525
Output	P	-2.4612	1.8012
	Q	-0.5418	8.3188
	$frequency$	-8.5615	6.8365
	$voltage$	0.9987	1.0001

and observations are shown in Table 2.

We use Mean Square Error (MSE) of the estimated parameters to evaluate the performance:

$$MSE = \frac{1}{Tr \times D} \sum_{i=1}^{Tr} \sum_{j=1}^{D} (P_{ij} - \hat{P}_{ij})^2 \qquad (10)$$

where Tr is the number of trials (e.g., each trial is initialized with a new set of ground-truth parameters), D is the number of parameters to be optimized, i.e., the parameter dimension, P_{ij} and $\hat{P}_{ij}$ represent the ground-truth and estimated j^{th} parameter in i^{th} trial, respectively. The term ground-truth denotes to the parameter combination that generates expected outputs. In our experiment, the objective function that needs to be minimized is defined by

$$D = \sum_{i=1}^{m} ||O_i - \hat{O}_i||_2 \qquad (11)$$

where $m = 4$ is the dimension of the output, O_i and $\hat{O}_i$ are vectors of length 452 that represent outputs of ground-truth and of estimated parameters respectively. We set $Tr = 100$, and in each run, we query 500 observations. The experiments are run on an Intel i7-9700k, and the results are shown in Table 3. As residual block (He et al. 2016) has seen great success in NNs, we re-implement a DNGO with two residual blocks and add it for comparison. As the table shows,

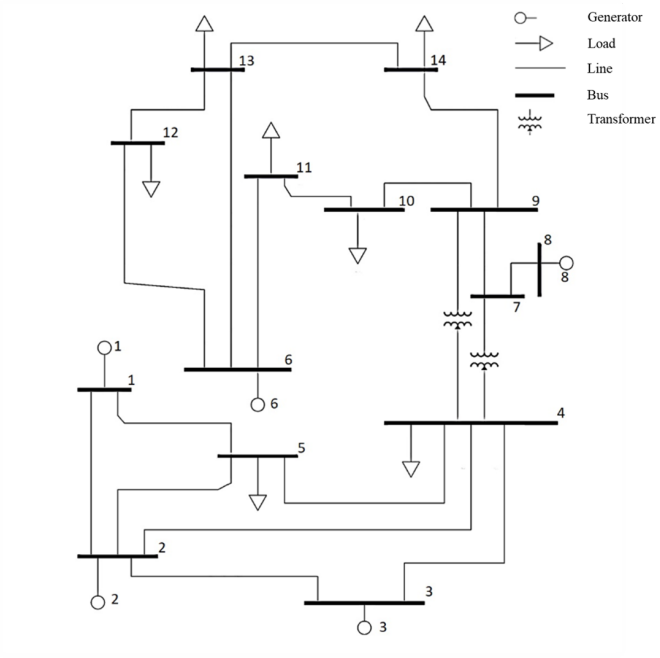

Figure 2: The IEEE 14 bus system.

Table 3: Evaluation of Different Parameter Optimization Methods for power system

Experiment	Time(s)	MSE
Random Search	20	2.001
GP	349	1.759×10^{-1}
DNGO	409	1.504×10^{-1}
DNGO(Residual)	997	1.758×10^{-1}
BOHAMIANN	1672	2.718×10^{-2}
NPBO	157	5.182×10^{-3}

NPBO has the most accurate parameter calibration among all the models with a very short execution time.

We further compare the four-dimensional outputs of the power system with the optimized parameters using NPBO to the observed target outputs in Fig. 3 . As it can be seen, there is only a very small difference between our optimized output and the target output, which indicates, with only a few observations, we can still obtain accurate and satisfactory parameter values.

Seven Benchmark Problems

To demonstrate the effectiveness of our approach, we compare NPBO to the state-of-the-art Bayesian optimization methods using different surrogate models on a set of synthetic functions (Klein et al. 2017). Besides GP-based Bayesian optimization, DNGO and BOHAMIANN, we also use Random Forest (Hutter, Hoos, and Leyton-Brown 2011) as the surrogate model in comparison. The goal is to find the parameters that minimize the synthetic functions. The results based on seven benchmark problems (Eggensperger

Table 4: Evaluation of Different Surrogate Models on global optimization benchmarks

Experiment	Gaussian Process	Random Forest	DNGO	BOHAMIANN	NPBO
Branin	**0.3996**	0.4562	0.4019	**0.3979**	**0.3980**
Camelback	**−1.011**	−0.8085	**−1.026**	**−1.027**	−0.9999
Hartmann3	−1.028	−0.998	**−3.862**	**−3.861**	**−3.498**
Forrester	**−6.021**	**−6.021**	−5.846	**−6.021**	−5.301
GoldsteinPrice	**4.916**	27.69	**6.379**	11.39	**8.654**
Hartmann6	**−3.255**	−3.132	**−3.249**	**−3.264**	−3.214
SinOne	**0.04292**	0.06472	**0.04292**	0.04292	**0.04292**

Top-3 best algorithms for each benchmark problem are bolded.

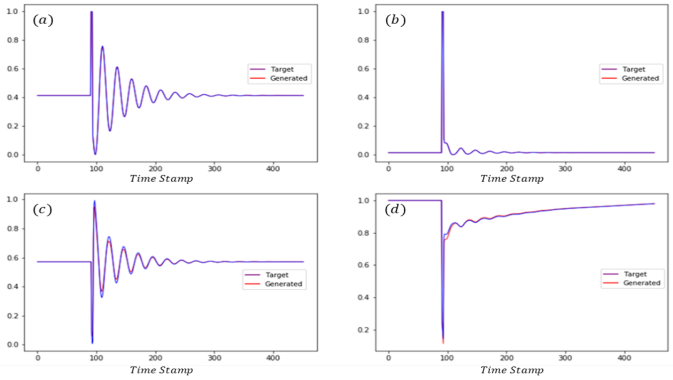

Figure 3: Comparison of Output from Calibrated Model and Target Output. (a): Real power injection; (b): Reactive power injection; (c): Frequency; (d) Voltage. Outputs are normalized using Min-Max Normalization method.

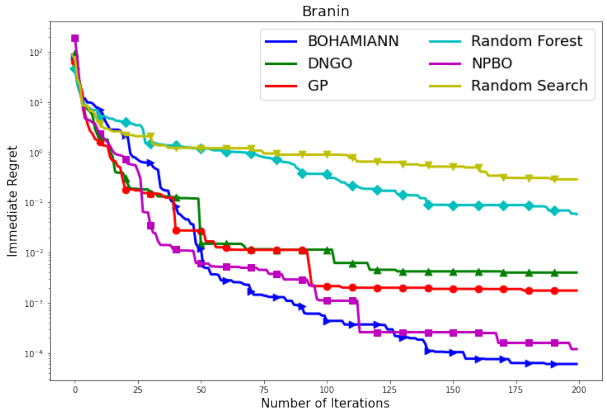

Figure 4: Immediate regret of different surrogate models applied to Bayesian Optimization on the Branin benchmark. Result averaged over 10 runs.

et al. 2013) are shown in Table 4. These benchmark problems are popular synthetic functions with the number of parameters range from one to six, e.g., Branin and Hartmann function (Surjanovic and Bingham 2013). As the table shows, all surrogate models with Bayesian optimization achieved acceptable performance. Overall, among the seven benchmark problems, NPBO performs competitive to BOHAMIANN, DNGO and GP based Bayesian optimization on four problems.

We further show the optimization process of *Branin* in detail in Fig. 4, where the performance is measured by immediate regret defined by (12)

$$I = |\hat{f}^i_{opt} - f_{opt}| \tag{12}$$

where $\hat{f}^i_{opt}$ is the optimal observed value found in i^{th} iteration, and f_{opt} represents the theoretical optimal value. As it can be seen, NPBO performs competitively with BOHAMIANN and GP based Bayesian optimization, and its performance exceeds random search and Bayesian optimization based on random forest.

Conclusions and Future Work

In this paper, we propose Neural Process for Bayesian Optimization (NPBO) as a scalable parameter optimization method. NPBO has the ability to efficiently identify the optimal parameter combination of black-box models. The proposed model preserves the advantage of the GP such as flexibility and estimation of uncertainty, while reduces the time complexity from cubic to linear and improves the accuracy in uncertainty estimation. NPBO is applied to optimize the parameters of a complex 14-parameter generator models in an IEEE 14-bus power system and the results show that NPBO outperforms the other benchmark algorithms, i.e., Gaussian Process, Random Forest, Deep Neural network for Global Optimization (DNGO) and Bayesian Optimization with Hamiltonian Monte Carlo Artificial Neural Networks (BOHAMIANN). We further compared the performance of NPBO on seven common benchmark problems with different surrogate models and the results show NPBO has competitive performance with benchmark approaches.

We consider three aspects in our future work: i) we are going to apply NPBO in different scenarios, e.g., accelerating experiments in the physical science (Ermon 2020); ii) we will test the performance of variants of NP as the surrogate model, such as Conditional Neural Process (Garnelo et al. 2018a) and Attentive Neural Process (Kim et al. 2019); iii) acquisition function could also be replaced by NN to perform the trade-off strategy under Bayesian optimization framework.

References

Archetti, F.; and Candelieri, A. 2019. *Bayesian Optimization and Data Science*. Springer.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research* 13(1): 281–305.

Cassioli, A. 2013. A Tutorial on Black–Box Optimization. [Online]. Available: https://www.lix.polytechnique.fr/~dambrosio/blackbox_material/Cassioli_1.pdf.

Chen, Y.; Huang, A.; Wang, Z.; Antonoglou, I.; Schrittwieser, J.; Silver, D.; and de Freitas, N. 2018. Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855* .

Cheng, J. 2018. Evolutionary optimization: A review and implementation of several algorithms. [Online]. URL https://www.strong.io/blog/evolutionary-optimization. Available: https://www.strong.io/blog/evolutionary-optimization [Accessed: Jun.23, 2020].

Duris, J.; Kennedy, D.; Hanuka, A.; Shtalenkova, J.; Edelen, A.; Baxevanis, P.; Egger, A.; Cope, T.; McIntire, M.; Ermon, S.; et al. 2020. Bayesian optimization of a free-electron laser. *Physical Review Letters* 124(12): 124801.

Eggensperger, K.; Feurer, M.; Hutter, F.; Bergstra, J.; Snoek, J.; Hoos, H.; and Leyton-Brown, K. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, 3.

Ermon, S. 2020. Bayesian Optimization and Machine Learning for Accelerating Experiments in the Physical Sciences. [Online]. Available: https://www.youtube.com/watch?v=5n7UinKrMLk&list=PL1e3Jic2_DwwJQ528agJYMEpA0oMaDSA9&index=18.

Frazier, P. I. 2018. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811* .

Garnelo, M.; Rosenbaum, D.; Maddison, C. J.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y. W.; Rezende, D. J.; and Eslami, S. 2018a. Conditional neural processes. *arXiv preprint arXiv:1807.01613* .

Garnelo, M.; Schwarz, J.; Rosenbaum, D.; Viola, F.; Rezende, D. J.; Eslami, S.; and Teh, Y. W. 2018b. Neural processes. *arXiv preprint arXiv:1807.01622* .

Görtler, J.; Kehlbeck, R.; and Deussen, O. 2019. A Visual Exploration of Gaussian Processes. *Distill* 4(4): e17.

Gunawan, A.; Lau, H. C.; et al. 2011. Fine-tuning algorithm parameters using the design of experiments approach. In *International Conference on Learning and Intelligent Optimization*, 278–292. Springer.

Gustafsson, M. 2016. Antenna current optimization and optimal design. In *2016 International Symposium on Antennas and Propagation (ISAP)*, 132–133. IEEE.

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*, 770–778.

Hebbal, A.; Brevault, L.; Balesdent, M.; Talbi, E.-G.; and Melab, N. 2019. Bayesian optimization using deep Gaussian processes. *arXiv preprint arXiv:1905.03350* .

Huang, R.; Diao, R.; Li, Y.; Sanchez-Gasca, J.; Huang, Z.; Thomas, B.; Etingov, P.; Kincic, S.; Wang, S.; Fan, R.; et al. 2017. Calibrating parameters of power system stability models using advanced ensemble Kalman filter. *IEEE Transactions on Power Systems* 33(3): 2895–2905.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, 507–523. Springer.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* .

Joyce, J. M. 2011. *Kullback-Leibler Divergence*, 720–722. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-04898-2. doi:10.1007/978-3-642-04898-2_327. URL https://doi.org/10.1007/978-3-642-04898-2_327.

Kim, H.; Mnih, A.; Schwarz, J.; Garnelo, M.; Eslami, A.; Rosenbaum, D.; Vinyals, O.; and Teh, Y. W. 2019. Attentive neural processes. *arXiv preprint arXiv:1901.05761* .

Klein, A.; Falkner, S.; Mansur, N.; and Hutter, F. 2017. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian Optimization Workshop*.

Lázaro-Gredilla, M.; Quiñonero-Candela, J.; Rasmussen, C. E.; and Figueiras-Vidal, A. R. 2010. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research* 11: 1865–1881.

Lin, L.; Wu, W.; Shangguan, Z.; Wshah, S.; Elmoudi, R.; and Xu, B. 2020. HPT-RL: Calibrating Power System Models based on Hierarchical Parameter Tuning and Reinforcement Learning. In *IEEE International Conference on Machine Learning Applications*.

Liu, H.; Ong, Y.-S.; Shen, X.; and Cai, J. 2020. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems* .

McIntire, M.; Ratner, D.; and Ermon, S. 2016. Sparse Gaussian Processes for Bayesian Optimization. In *UAI*.

Muehleisen, R. T.; and Bergerson, J. 2016. Bayesian Calibration - What, Why And How. In *INTERNATIONAL HIGH PERFORMANCE BUILDINGS CONFERENCE*.

Rasmussen, C.; and Williams, C. 2006. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press.

Snelson, E.; and Ghahramani, Z. 2006. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, 1257–1264.

Snoek, J.; Rippel, O.; Swersky, K.; Kiros, R.; Satish, N.; Sundaram, N.; Patwary, M.; Prabhat, M.; and Adams, R. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, 2171–2180.

Springenberg, J. T.; Klein, A.; Falkner, S.; and Hutter, F. 2016. Bayesian optimization with robust Bayesian neural networks. In *Advances in neural information processing systems*, 4134–4142.

Surjanovic, S.; and Bingham, D. 2013. Virtual library of simulation experiments: test functions and datasets. *Simon Fraser University, Burnaby, BC, Canada, accessed May* 13: 2015.

Weber, J. 2015. Description of Machine Models GENROU, GENSAL, GENTPF and GENTPJ. [Online]. Available: https://www.powerworld.com/files/GENROU-GENSAL-GENTPF-GENTPJ.pdf.

Xiao, J.-k.; Li, W.-m.; Li, W.; and Xiao, X.-r. 2015. Optimization on black box function optimization problem. *Mathematical Problems in Engineering* 2015.

Yang, X. 2017. Understanding the variational lower bound.

Yk, B. 2020. IEEE 14 bus System. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/46067-ieee-14-bus-system.

Zhang, P.; and Zhang, P. 2010. Industrial control system simulation routines. *Advanced Industrial Control Technology; Elsevier: Oxford, UK* 781–810.