Prediction of Core Neutronics Parameters Using Neural Network Hyperparameter Optimization Methods

Authors: Zhang Fan, Zhang Junda, Sun Qizheng, Xiao Wei, Liu Xiaojing, Zhang Tengfei, Tengfei Zhang

Date: 2024-05-21T00:00:00+00:00

Abstract

Neural networks can learn the relationships between input and output variables from large datasets, exhibit powerful fitting capabilities, and are commonly utilized as surrogate models for computational codes in various fields, including nuclear engineering calculations. Neutron transport calculation, as a core component of neutronics simulation, suffers from prohibitive computational time, which can be addressed through the adoption of neural network models. However, neural network models involve numerous hyperparameters that require configuration, and manual tuning of these hyperparameters is labor-intensive, repetitive, and relies exclusively on empirical expertise; moreover, these hyperparameters are not transferable when solving different problems. To overcome these limitations, this paper proposes a methodology that employs Bayesian Optimization algorithms to tune neural network hyperparameters, integrating learning rate decay and loss function optimization techniques. This approach can automatically search for optimal hyperparameter combinations tailored to datasets from different problems to achieve superior performance, demonstrating high flexibility, efficiency, and strong generalization capability. This study fits key core parameters obtained from the TAKEDA benchmark problem, with results indicating that the average error of the effective multiplication factor keff is within 150 pcm, while the average and maximum error rates of the regional integrated flux Φ on the TAKEDA1 dataset are 1.72% and 7.56%, respectively. This research can provide valuable reference for the application of artificial intelligence in core physics calculation theory.

Full Text

Preamble

Vol. XX, No. X, XXX 20XX, NUCLEAR TECHNIQUES ChinaXiv

Research on Core Neutronic Parameter Prediction Based on Neural Network Hyperparameter Optimization Method

ZHANG Fan[1], ZHANG Junda[2], SUN Qizheng[2], XIAO Wei[2], LIU Xiaojing[2], ZHANG Tengfei[2]

[1] College of Smart Energy, Shanghai Jiao Tong University, Shanghai 200240, China
[2] School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Abstract

[**Background**] Neural networks possess powerful fitting capabilities and can learn relationships between input and output variables from large datasets, often serving as surrogate models for physical programs in engineering calculations, including nuclear engineering. Neutron transport calculations, as a core component of neutronics simulations, suffer from lengthy computational times —a challenge that can be addressed through neural network models. However, neural network models require setting numerous hyperparameters, and manual adjustment is laborious, repetitive, and relies solely on experience. Moreover, these hyperparameters are not transferable across different problems.

[**Purpose**] This research seeks to develop a surrogate model for VITAS and provide reference for artificial intelligence applications in core physics calculation theory.

[**Methods**] This paper proposes using Bayesian Optimization algorithms to adjust neural network hyperparameters, combined with learning rate decay and loss function optimization methods.

[**Results**] By fitting key core parameters obtained from VITAS calculations of the TAKEDA benchmark problem, results show that the average error of the effective multiplication factor (k ff) remains within 150 pcm, while the average error rate of regional integral flux (f) on the TAKEDA1 dataset is 1.72% with a maximum error rate of 7.56%.

[**Conclusions**] This approach can automatically search for optimal hyperparameter combinations for different datasets to achieve peak performance, demonstrating high flexibility, efficiency, and strong generalization capability.

Keywords: Bayesian optimization for hyperparameter tuning, FCNN, Neutron transport computation

Classification: TL329

1. Introduction

Neutron transport problems in nuclear reactor physics calculations are primarily solved using two classes of methods: Monte Carlo methods (MC) and deterministic methods. Common neutron transport methods include the Finite Element Method (FEM), Method of Characteristics (MOC), and Variational Nodal Method (VNM). These traditional neutron transport methods are characterized by low computational efficiency and long processing times, making them impractical for frequent core design optimization and refueling design tasks.

Machine learning and artificial intelligence offer advantages in processing complex big data compared to traditional Monte Carlo and deterministic neutron transport methods. In recent years, interdisciplinary applications between nuclear physics and machine learning have emerged with broad prospects. Song et al. developed a core parameter prediction program based on BP (back propagation) artificial neural networks with relative errors within 10%, sparking subsequent research interest. Akkoyun used artificial neural networks to predict total fusion and fusion-evaporation reaction cross-sections, while Guo et al. employed a hybrid artificial neural network to simulate the thermodynamic behavior of the Sequoia nuclear power plant.

Among these approaches, the Fully Connected Neural Network (FCNN) represents a relatively simple artificial neural network architecture belonging to the feedforward neural network family. FCNNs consist only of an input layer, hidden layers, and an output layer, with multiple neurons possible in each hidden layer. This versatile learning approach can be applied to nearly all tasks, including classification, regression, and unsupervised learning. Machine learning has developed rapidly due to its ability to fit complex analytical calculation processes while saving labor and computational costs.

VITAS is a general-purpose computational program for accurately solving neutron transport problems based on VNM. It integrates multiple calculation methods, utilizes matrix operations and numerical integration techniques, and can handle multi-dimensional, multi-group, steady-state, and transient neutron transport problems with various mesh types. To address the low efficiency and long computational times of traditional transport methods, this paper develops a surrogate model for core transport calculations based on neural network methods to replace VITAS program functionality.

However, neural networks require setting a series of hyperparameters that largely determine FCNN performance. Manual hyperparameter tuning demands substantial effort, requires experience, and incurs high costs. With the continuous development of machine learning, deep learning models involve far more hyperparameters than traditional machine learning methods, making manual tuning inadequate for both precision and time requirements. In recent years, Bayesian optimization has been increasingly applied to black-box function problems and has become the mainstream method for hyperparameter optimization. Bayesian optimization, also known as Sequential

Kriging Optimization (SKO), Sequential Model-based Optimization (SMBO), or Efficient Global Optimization (EGO), is a global optimization approach that uses surrogate models to fit expensive-to-evaluate objective functions. It actively selects the most "promising" evaluation points based on the surrogate model and utilizes complete historical information to improve search efficiency, obtaining optimal solutions for complex functions with minimal evaluations. Consequently, Bayesian optimization is also termed active optimization.

This paper employs Bayesian optimization methods, combined with learning rate decay and loss function optimization, to optimize FCNN hyperparameter selection. Using experimental data calculated by VITAS for the TAKEDA benchmark problem as datasets, we evaluate the prediction accuracy of FCNN with optimized hyperparameters and compare results with manually tuned networks. This aims to provide more efficient methods for solving core refueling optimization problems and explore further applications of artificial intelligence in the nuclear industry.

1.1 Bayesian Optimization Method

The Bayesian optimization framework consists of two key components: (1) using probabilistic models to surrogate complex objective functions with expensive evaluation costs, and (2) constructing an active selection strategy (acquisition function) based on posterior information from the surrogate model. In practical applications, appropriate models must be selected for specific problems. This section introduces the definition of hyperparameter optimization problems and the steps of Bayesian optimization, concluding with the selected FCNN hyperparameters and their value ranges for this study.

The hyperparameter optimization problem is defined as:

$$\arg\min_{x \in X} f(x)$$

where x represents a set of hyperparameter values and X is the hyperparameter search space. $f(x)$ is the objective to be optimized in hyperparameter tuning. In this study, the loss function is used as $f(x)$. Comparison and selection of loss functions are discussed in Section 1.2 (2) on loss function optimization. The goal of hyperparameter algorithms is to find the global optimum as quickly as possible.

The steps for Bayesian optimization of hyperparameters are as follows:

1. Randomly initialize n_0 groups of hyperparameters x_{init} in the hyperparameter search space X;
2. Obtain their corresponding function values $f(x_{\text{init}})$ to form the initial point set $D_0 = \{(x_{\text{init}}, f(x_{\text{init}}))\}$;
3. Construct a surrogate model $g(x)$ based on the currently obtained point set distribution;

4. Based on the surrogate model $g(x)$, maximize the acquisition function to obtain the next evaluation point: $x_t = \arg\max_{x \in X} \text{acquisition}(x)$;
5. Obtain the function value $f(x_t)$ of evaluation point x_t, add it to the current evaluation point set: $D_t = D_{t-1} \cup \{(x_t, f(x_t))\}$;
6. If t is less than the maximum iteration number N, return to step (3);
7. After reaching the maximum iteration number, output the optimal evaluation point $\{x^*, f(x^*)\}$.

This study implements the Bayesian optimization process by calling TensorFlow' s Bayesian Optimization function. By inputting hyperparameters and their value ranges to initiate the optimization function, the selected FCNN hyperparameters and their ranges are shown in .

** Table 1 The selected hyperparameters and their ranges**

Hyperparameters	Ranges
Batch size	[1, 1000]
Number of hidden layers	[1, 5]
Threshold (min_{delta})	[1e-6, 1e-4]
Decay factor	[0.1, 0.9]
Number of neurons in hidden layers	[1, 2000]
Loss function parameter (loss_{delta})	[0.1, 10]

All hyperparameters listed in the table are critical parameters in FCNN, as introduced below.

1.2 Fully Connected Neural Network

For the surrogate model selection in neutron transport solving, this work adopts the Fully Connected Neural Network (FCNN). FCNN is a relatively simple artificial neural network architecture belonging to the feedforward neural network family, consisting only of an input layer, hidden layers, and an output layer, with multiple neurons possible in each hidden layer, as shown in [Figure 1: see original paper].

The entire network is built using TensorFlow, an open-source framework for deep learning platforms developed by Google in 2015. The following parameters are involved in FCNN training:

Epoch: The process of feeding the entire training set into the neural network model for one complete training cycle is called an epoch.

Batch Size: Due to computational limitations or other constraints, when data cannot pass through the neural network all at once, the dataset must be divided into multiple batches. A small portion of data samples from the training set is used for one backpropagation parameter update.

Learning Rate (lr) refers to the step size used to adjust fitting parameters during neural network training, determining the adjustment magnitude for each gradient update. Learning rate selection directly impacts model performance and training effectiveness.

(1) Adaptive Learning Rate Adjustment

If the learning rate is set too large, the step size increases, leading to oscillations that make it difficult to find high-precision solutions. [Figure 2: see original paper] (left) shows loss function oscillations caused by an excessively large learning rate, resulting in unstable convergence. If the learning rate is too small, convergence speed slows down, potentially yielding only local optimal solutions. Based on this, this paper proposes an adaptive learning rate adjustment method that uses a larger learning rate for rapid approximation initially, then gradually reduces the learning rate to accommodate model parameter update patterns, eliminating late-stage loss function oscillations and stabilizing the loss curve. [Figure 2: see original paper] (right) shows the loss function decline curve after adaptive learning rate adjustment, which effectively eliminates oscillations.

This study uses the ReduceLROnPlateau function from TensorFlow, a learning rate scheduler that automatically adjusts the learning rate based on monitored metric changes. When performance metrics on the validation set stop improving, ReduceLROnPlateau gradually reduces the learning rate to facilitate better model convergence. It includes the following parameters:

- **lr**: Initial learning rate value
- **factor**: The factor by which the learning rate is reduced each time, with the learning rate decreasing as lr $\times$ factor
- **patience**: When patience epochs pass without model performance improvement, the learning rate reduction is triggered
- **min_{delta}**: Threshold; when the model performance improvement is less than min_{delta}, it is considered no improvement

Both factor and min_{delta} parameters are selected as hyperparameters for Bayesian optimization in this study.

(2) Loss Function Optimization

Mean Square Error (MSE) refers to the average of squared distances between model predicted values $f(x)$ and sample true values y, as shown in Equation (2):

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2$$

Mean Absolute Error (MAE) refers to the average of absolute distances between model predicted values $f(x)$ and sample true values y, as shown in Equation (3):

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - f(x_i)|$$

In Equations (2) and (3), y_i and $f(x_i)$ represent the true and predicted values of the i-th sample, respectively, and m is the number of samples.

A major issue with using MAE as a loss function for neural network training is that its gradient remains consistently large, potentially causing the model to miss minima when using gradient descent. In contrast, MSE' s gradient gradually decreases as the loss approaches its minimum, enabling more accurate convergence. This problem can be addressed using Huber Loss. Huber Loss is a loss function for regression problems that combines characteristics of both MSE and MAE. It can reduce gradients near minima to address MAE' s issue of missing minima while being more robust to outliers than MSE. This paper uses Huber Loss as the loss function, as shown in Equation (4):

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

Huber Loss synthesizes MSE and MAE, containing a hyperparameter δ that determines its emphasis on MSE versus MAE. When $\delta \to 0$, Huber Loss approaches MAE; when $\delta \to \infty$, Huber Loss approaches MSE. In , δ corresponds to loss_{delta}.

The settings of these parameters determine FCNN performance, so they are selected as hyperparameters for Bayesian optimization, with their search spaces provided in .

1.3 Dataset Acquisition

The datasets used in this paper are generated through VITAS calculations of the TAKEDA1 and TAKEDA2 benchmark problems, comprising 10,000 and 20,000 data groups, respectively. Each data sample consists of a core arrangement pattern and two core parameters (effective multiplication factor k ff and regional integral flux f). Input dimensions are determined by the reactor core arrangement, while output dimensions correspond to core parameters. TAKEDA1 is a 1/8 symmetric light water reactor model, and TAKEDA2 is a 1/4 fast breeder reactor model.

TAKEDA1 contains three assembly types: Control Rod (CR), Reflector, and Core. Its three-dimensional schematic and x-y cross-section core arrangement are shown in [Figure 3: see original paper]. To ensure physical meaningfulness, the outermost reflector layer in the 1/4 x-y cross-section is fixed in position while other assemblies are shuffled, then mirrored to form a complete core, as illustrated in [Figure 4: see original paper]. For FCNN computation convenience, the three assembly types (CR, Reflector, Core) are mapped to -1, 0, and 1,

respectively. This generates 10,000 different random arrangements, where each input vector' s sequence of -1, 0, 1 corresponds one-to-one with the shuffled assembly arrangement. The input dimension is 16. Two independent FCNNs are used to predict two physical quantities: k ff has an output dimension of 1, while f has an output dimension of 6.

TAKEDA2 contains three assembly types: Control Rod (CR), Axial Blanket, and Radial Blanket, as shown in [Figure 5: see original paper]. To ensure physical meaningfulness, the outermost three layers of Radial Blanket in the 1/4 x-y cross-section are fixed while other assemblies are shuffled, then mirrored to form a complete core, as shown in [Figure 6: see original paper]. For FCNN computation convenience, the three assembly types (CR, Axial Blanket, Radial Blanket) are mapped to -1, 0, and 1, respectively. This generates 20,000 different random arrangements. Each input vector' s sequence of -1, 0, 1 corresponds one-to-one with the shuffled assembly arrangement, with an input dimension of 121. Two independent FCNNs predict two physical quantities: k ff has an output dimension of 1, while f has an output dimension of 20.

During FCNN training, raw data must be transformed into a standard dataset. Assuming N training samples, each training sample should be: $T_k = (X_k, Y_k)$, where T_k is the k-th training sample containing the k-th input variable X_k and output variable Y_k. Here, $X = \{x_1, x_2, \ldots, x_N\}$, $Y = \{y_1, y_2, \ldots, y_N\}$, where i and j represent the dimensions of input and output variables, respectively. For TAKEDA1, $k = 10,000$, input vector X is a sequence of -1, 0, 1 with each input having dimension $i = 16$. Two independent FCNNs predict two physical quantities, with output variables Y being k ff and f, where k ff has dimension $j = 1$ and f has dimension $j = 6$. For TAKEDA2, $k = 20,000$, input vector X is a sequence of -1, 0, 1 with each input having dimension $i = 121$. Two FCNNs predict two physical quantities, with output variables Y being k ff and f, where k ff has dimension $j = 1$ and f has dimension \$j = 20.

1.4 Model Validation

This study uses 10,000 TAKEDA1 and 20,000 TAKEDA2 data groups, with inputs being reactor core arrangements and outputs being two parameters (k ff and f). The data are divided into training and validation sets at a 6:4 ratio. The algorithm flowchart is shown in [Figure 7: see original paper].

Using TAKEDA1 as an example, the table below provides values for a manually set hyperparameter group and a group output by Bayesian optimization:

** Table 2 Hyperparameter selection for TAKEDA1**

Hyperparameters	Manual Setting	Bayesian Optimization
Batch_{size}	64	100
Hidden_{layers}	4	5
Min_{delta}	5.5e-5	4.5e-5

Hyperparameters	Manual Setting	Bayesian Optimization
Factor	0.5	0.8
Num_{neurons}	800, 900, 1600, 800	400, 800, 1800, 600, 50
Loss_{delta}	1.248e-5	8.927e-05

After substituting into FCNN training, the training process curves for TAKEDA1' s two prediction parameter models are presented below:

[**Figure 8: see original paper**] Comparison of loss function curves between Bayesian optimization (left) and manual tuning (right) for k ff

As shown in [Figure 8: see original paper], as the learning rate decreases, the training set loss function gradually declines. The test set loss function in [Figure 8: see original paper] (left) stabilizes after 400 epochs, while the test set loss function in [Figure 8: see original paper] (right) for manual tuning first decreases, then slowly increases, and finally stabilizes.

[**Figure 9: see original paper**] Comparison of loss function curves between Bayesian optimization (left) and manual tuning (right) for f

In [Figure 9: see original paper] (left), the loss function for the Bayesian-tuned group drops rapidly at the beginning of training, while in [Figure 9: see original paper] (right), the test set loss function actually increases when the training set loss function decreases, indicating overfitting in the manual tuning group.

2. Results and Discussion

Evaluation Metrics for Prediction Problems: Definitions of MAE and MSE are given in Equations (2) and (3). Mean Absolute Percentage Error (MAPE) and R^2 (coefficient of determination) are defined as:

$$\text{MAPE} = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

where y_i represents true values, $\bar{y}$ represents sample mean, and $\hat{y}_i$ represents predicted values. R^2 measures the proportion of variance in the dependent variable explained by independent variables, ranging from 0 to 1. An R^2 value closer to 1 indicates better regression fit.

**** and **** compare errors between FCNN outputs after Bayesian optimization and manual hyperparameter setting for the two output variables (k ff and f) using TAKEDA1 and TAKEDA2 as datasets:

** Table 3 Comparison of errors between Bayesian optimization and manual hyperparameter tuning for k ff**

Metric	TAKEDA1	TAKEDA2
	Manual Setting	Bayesian Optimization
MAE (pcm)	397	118
MSE	6.78996e-05	4.258e-11
R^2	0.985	0.999

As shown in , Bayesian optimization hyperparameter combinations demonstrate clear advantages over manual settings in both error magnitude and goodness-of-fit. The MAE for k ff obtained through Bayesian optimization is 118 pcm for TAKEDA1 and 132 pcm for TAKEDA2, compared to 397 pcm and 1625 pcm from manual tuning. On TAKEDA1, Bayesian optimization' s MAE is one-third of manual tuning; on TAKEDA2, it is one-twelfth, representing significant accuracy improvements. The R^2 values from Bayesian optimization are approximately 0.999, exceeding those from manual tuning and indicating better FCNN fitting.

** Table 4 Comparison of errors between Bayesian optimization and manual hyperparameter tuning for f**

Metric	TAKEDA1	TAKEDA2
	Manual Setting	Bayesian Optimization
MAPE (%)	15.47	1.7187
Max Percentage Error (%)	378.92	7.5585
R^2	0.85	0.99

Bayesian optimization yields a MAPE of 1.7187% and maximum percentage error of 7.5585% for f on TAKEDA1, and 0.8213% and 11.130% respectively on TAKEDA2, achieving satisfactory accuracy. Compared to manual tuning, Bayesian optimization' s MAPE is one-ninth on TAKEDA1 and one-fifth on TAKEDA2, while maximum percentage errors are one-fiftieth and one-eighth respectively, demonstrating substantial accuracy improvements. The R^2 values from Bayesian optimization are approximately 0.99, exceeding manual tuning results and indicating superior FCNN fitting.

[**Figure 10: see original paper**] and [**Figure 11: see original paper**] show error distribution histograms for k ff and f after Bayesian hyperparameter tuning:

[**Figure 10: see original paper**] Error distribution histogram for k ff

As shown in [Figure 10: see original paper], errors in k ff for both TAKEDA1 and TAKEDA2 follow a normal distribution with concentrated distribution. Ninety percent of errors are within 500 pcm, and fifty percent are within 150 pcm.

[**Figure 11: see original paper**] Percentage error distribution histogram for f

As shown in [Figure 11: see original paper], percentage errors for f in both TAKEDA1 and TAKEDA2 follow a normal distribution. TAKEDA1 shows a more dispersed error distribution, while TAKEDA2 exhibits a "tall and narrow" distribution with more concentrated errors. For TAKEDA1, 90% of percentage errors for f are within 3.5%; for TAKEDA2, 90% are within 1%, achieving good accuracy.

3. Conclusions

To efficiently solve core refueling optimization problems, this paper proposes a Bayesian optimization algorithm for neural network hyperparameters as a surrogate model for core transport calculations. Using datasets obtained from TAKEDA1 and TAKEDA2 benchmark problems, we trained the model and compared results with manual tuning, yielding the following findings:

1. FCNN with Bayesian-optimized hyperparameters can effectively fit VITAS calculation results. The average error of k ff obtained through Bayesian optimization is within 150 pcm. The MAPE for f is 1.7187% on TAKEDA1 and 0.8213% on TAKEDA2, with errors within acceptable ranges.
2. The FCNN model established with Bayesian-optimized hyperparameters significantly outperforms manually tuned models in both error metrics and goodness-of-fit.
3. The hyperparameters and corresponding FCNN used in this study are only applicable to TAKEDA1 and TAKEDA2 datasets calculated by VITAS. For predicting other core parameters, the corresponding dataset can be replaced and new hyperparameter combinations can be obtained following the flowchart in [Figure 7: see original paper] for FCNN training, replacing manual tuning steps and demonstrating universality.
4. This study validates the feasibility and advantages of neural network-based parameter prediction methods in reactor physics calculations. Further exploration of additional possibilities for different neural network types in reactor physics calculations is warranted.

Author Contributions

ZHANG Fan: Simulation calculations and manuscript writing; ZHANG Junda, SUN Qizheng, XIAO Wei: Technical support; LIU Xiaojing: Funding support; ZHANG Tengfei: Funding support and manuscript revision.

References

1. Kazuteru SUGINO, Toshikazu TAKEDA. An Improvement of the Transverse Leakage Treatment for the Nodal SN Transport Calculation Method

in Hexagonal-Z Geometry[J]. Journal of Nuclear Science and Technology, 1996(8). DOI:10.3327/jnst.33.620.

2. PALMIOTTI G, LEWIS E E, CARRICO C B. VARIANT: VARIational Anisotropic Nodal Transport for multidimensional Cartesian and hexagonal geometry calculation: ANL-95/40[R]. Argonne, IL, USA: Argonne National Laboratory, 1995.
3. ZHANG T F, WU H C, CAO L Z, et al. An improved variational nodal method for the solution of the three dimensional steady-state multi-group neutron transport equation[J]. Nuclear Engineering and Design, 2018, 419-427. DOI:10.1016/j.nucengdes.2018.07.009.
4. ZHANG T F, ZHU L, WU H C, et al. Development and validation of high flux engineering test reactor fuel management platform HEFT[C]. Atomic Energy Science and Technology, 2013, 47(S1): 467-471.
5. SONG Meicun, CAI Qi. Reactor power prediction based on BP Neural Network[J]. Atomic Energy Science and Technology, 2011, 45(10):1242-1246. DOI:10.7538/yzk.2011.45.10.1242.
6. AKKOYUN S. Estimation of fusion reaction cross-sections by artificial neural networks[J]. Nuclear Instruments and Methods in Physics Research B, 2020, 462: 51-54. DOI:10.1016/j.nimb.2019.11.014.
7. GUO Z, UHRIG R E. Use of artificial neural networks to analyze nuclear power plant performance[J]. Nuclear Technology, 1992, 99(1): 36-42. DOI:10.13182/NT92-A34701.
8. RUMELHART D E, HINTON G E, WILLIAMS R J. Learning representations by back-propagating errors[J]. Nature, 1986, 323: 533-536. DOI:10.1038/323533a0.
9. ZHANG T F, XIAO W, YIN H, et al. VITAS: A multi-purpose simulation code for the solution of neutron transport problems based on variational nodal methods.[J]. Annals of Nuclear Energy, 2022. DOI:10.1016/j.anucene.2022.109335.
10. ZHANG Tengfei, YIN Han, SUN Qizheng, XIAO Wei, et al. Application Research on VITAS—a General-purpose Neutron Transport Code[J]. Nuclear Power Engineering, 2023, 44(02):15-23. DOI:10.13832/j.jnpe.2023.02.0015.
11. CUI Jiaxu, YANG Bo. Survey on Bayesian Optimization Methodology and Applications[J]. Journal of Software, 2018, 29(10):3068-3090. DOI:10.13328/j.cnki.jos.005607.
12. Ghahramani. Probabilistic machine learning and artificial intelligence[J]. Nature, 2015, 521:452-459. DOI:10.1038/nature14541.
13. MARTIN A, BARHAM P, CHEN J, et al. TensorFlow: A system for large-scale machine learning[J]. USENIX Association, 2016. DOI:10.48550/arXiv.1605.08695.