Development of Deep Residual Neural Networks for Gear Pitting Fault Diagnosis Using Bayesian Optimization

Jialin Li, Renxiang Chen, Xianzhen Huang, and Yongzhi Qu, *Member, IEEE*

***Abstract*— In recent years, the application of deep neural networks containing directed acyclic graph (DAG) architectures in mechanical fault diagnosis has achieved remarkable results. In order to improve the fault diagnosis ability of the networks, researchers have been working on developing new network architectures and optimizing the training process. However, this approach requires sufficient time and empirical knowledge to try a potential optimal framework. Furthermore, it is time-consuming and laborious to retune the network architecture and hyperparameter values when faced with different operating conditions or diagnostic tasks. To avoid these drawbacks, this article proposes an automated network architecture search (NAS) method and performs hyperparameter optimization. The adjacency matrix is used to define the architecture search space, the Bayesian optimization is used as the architecture search strategy, and a network test error is used for architecture evaluation. Seven types of convolutional layers and pooling layers are used as basic components to build fault diagnosis models. The gear pitting fault experiment, including seven gear pitting types, was established and used to validate the diagnostic model. The experimental results show that the diagnostic results of the network model automatically constructed by the proposed method are better than those of the general network model. It can be concluded that the proposed method can indeed replace the manual construction of an effective and practical gear pitting fault diagnosis model.**

***Index Terms*— Bayesian optimization, directed acyclic graph (DAG) network, gear pitting faults, hyperparameter optimization, network architecture search (NAS).**

I. Introduction

NOWADAYS, there are a lot of works on developing fault diagnosis models based on deep neural networks (DNNs) [1]. The neural network architecture designing and hyperparameter optimization are crucial for the feature representation of fault data and the final diagnosis results [2]. However, the design of neural architectures relies heavily on the researchers' prior knowledge and experience. Moreover, due to the limitation of inherent human knowledge, it is difficult for people to jump out of the original way of thinking and design the optimal model. Therefore, an intuitive idea is to reduce human intervention as much as possible and let the algorithm automatically design the neural network architecture and do hyperparameters optimization. In this way, a revolutionary neural architecture search (NAS) [3], [4], [5], [6], [7] algorithm emerges as the times require, and its related research work is complex and rich. The current network architecture search (NAS) algorithm can be divided into: search space, search strategy, and evaluation strategy according to the key components of NAS [8].

The search space defines the set of network architecture to be searched, which is crucial to the subsequent neural NAS. The size of the search space can be reduced by incorporating prior knowledge of the typical properties of the applicable task. However, this introduces inherent human thinking that may be detrimental to the discovery of better new architectures [9]. The traditional neural network architecture (search space), such as the traditional convolutional neural network (CNN) [10], has a chain structure, that is, it can be expressed as a sequence of n layers, and the ith layer network receives the information of the $i-1$ layer and transmits it to the $i+1$ layer. In this way, the search space of the chained network can be jointly represented by the maximum number of layers n, the type of operation performed at each layer, and the hyperparameters of the operation performed [11]. Influenced by multitype residual neural networks recently, the NAS search space has also shifted to a multistructured direction, that is, the ith layer network not only receives the information of the $i-1$ layer but can also randomly receive all the information from the 1 to the $i-1$ layer. In the face of multinode information input, two methods can be used: summation and splicing. The typical representative of the former is ResNet, and the typical representative of the latter is DenseNet [12], [13], [14]. In addition, in order to improve the search efficiency, the search space is sometimes limited or simplified, that is, by dividing the network into basic units and using the stacking of these units to form more complex networks [15]. The search efficiency can also be improved by limiting the overall topology of the neural network by learning from the human experience in designing

Manuscript received 29 August 2022; revised 2 October 2022; accepted 25 October 2022. Date of publication 4 November 2022; date of current version 22 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 51975110 and Grant 51975079, in part by the Liaoning Revitalization Talents Program under Grant XLYC1907171, in part by the Fundamental Research Funds for the Central Universities under Grant N2003005, in part by the Chongqing Postgraduate Joint Training Base under Grant JDLHPYJD2021007, in part by the Chongqing Engineering Laboratory for Transportation Engineering Application Robot Open Fund under Grant CELTEAR-KFKT-202103, and in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJQN202200732 and Grant KJQN201900721. The Associate Editor coordinating the review process was Dr. Yang Song. *(Corresponding author: Jialin Li.)*

Jialin Li and Renxiang Chen are with the Chongqing Engineering Laboratory for Transportation Engineering Application Robot, Chongqing Jiaotong University, Chongqing 400074, China (e-mail: jialinli_neu@163.com).

Xianzhen Huang is with the School of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China.

Yongzhi Qu is with the Department of Mechanical and Industrial Engineering, The University of Minnesota Duluth, Duluth, MN 55804 USA.

Digital Object Identifier 10.1109/TIM.2022.3219476

neural networks. However, these practices also limit the search diversity of NAS algorithms.

The search strategy details how to explore the search space, which is often exponential or even unbounded. Therefore, there is a tradeoff between exploration and mining. On the one hand, we want to quickly find an architecture with good performance, and on the other hand, we should avoid getting stuck in the local optimal architecture region. Currently known search methods include random search, Bayesian optimization [16], genetic algorithm (GA) [17], reinforcement learning [18], and gradient-based algorithm [19]. Among them, reinforcement learning, genetic learning, and gradient-based optimization are the current mainstream algorithms. The NAS algorithm based on reinforcement learning regards the neural network architecture design as a reinforcement learning problem, and the learning purpose is to obtain an optimal strategy for generating the network architecture [20]. Since the length of the structural parameters of the neural network is not fixed, a learning method that takes a variable-length data chain as input is required. A recurrent neural network is used as a controller to generate chains that describe the architecture of the subnetworks, thereby determining several subnetwork architectures, and these networks are trained on the training set and their accuracy is calculated on the validation set. The idea of using GA to solve NAS is to encode various network architectures into binary strings and run the GA to obtain the network architecture with the largest fitness function value, which is the optimal solution. Regarding how to encode the architecture of a neural network into a fixed-length binary string, Xie and Yuille [21] divided the network into multiple levels, encoding the topology of each level. The gradient-based algorithm is to make the discrete problem of architecture search continuous. In this way, the gradient descent method is used to solve the problem, which can efficiently search the neural network architecture and obtain the weight parameters of the network at the same time. DARTS [22] expresses the network architecture and network unit as a directed acyclic graph (DAG), relaxes the architecture search problem, and transforms it into a continuous variable optimization problem. In this way, the objective function is derivable and can be solved by gradient descent.

Evaluation strategy refers to the process of evaluating performance. The simplest approach is to sequentially train and validate numerous architectures, but doing so is computationally expensive and limits the number of architectures that can be explored [23]. Therefore, many recent studies have focused on developing methods to reduce the cost of these performance evaluations, such as reducing training time (the number of iterations), training on a subset of training samples, training on low-resolution images, or reducing the number of convolution kernels for certain layers during training. However, these practices may lead to biases in performance estimates while reducing computational costs. Using the characteristics of the unit block to predict the performance of the entire network and weight sharing may solve this contradiction [24], [25], that is, using the weight of the trained subnetwork as the initial weight of the subnetwork to be evaluated can effectively improve the training speed, accelerate the convergence, and avoid training from scratch. ENAS [26] and DARTS directly let each subnetwork share the same set of weight parameters.

This article aims to address network development and hyperparameter optimization problems when diagnosing gear pitting faults using deep DAG [27], [28], [29] neural networks. The Bayesian optimization is used to determine the network architecture and hyperparameters of the deep DAG neural network. Seven types of convolutional layers and pooling layers are used as basic components to build fault diagnosis models. The DAG network architecture is represented using an adjacency matrix, and the Bayesian optimization is used to search for the best DAG architecture and hyperparameter values. This article is arranged as follows. Section I presents related research on DNN architecture search and hyperparameter optimization methods. Section II describes the current problem and proposes a deep DAG neural NAS and hyperparameter optimization method based on a Bayesian optimization strategy. Section III describes the gear pitting test rig, hyperparameter information, and optimization procedures. Section IV compares and analyzes the diagnostic results of the deep DAG neural network constructed by the proposed method. Finally, Section V gives a conclusion.

II. Current Problem and Proposed Method

This section summarizes the difficulties in developing a gear pitting fault diagnosis model based on residual neural network and proposes a method for developing a deep DAG fault diagnosis model and hyperparameter optimization based on the Bayesian optimization for current problems.

A. Current Problem

The application of DNNs containing DAG architectures in mechanical fault diagnosis has achieved remarkable results. However, the following problems need to be solved when automatically building a fault diagnosis model based on the DAG neural network.

1) Global Search Space: At present, the strategy adopted by most NAS is to search the entire network space, which means that NAS needs to search for an optimal network architecture in a very large search space.
2) Discrete Search Strategy: The search space of early NAS is discrete, which means that the gradient cannot be calculated, and the gradient strategy cannot be used to adjust the network model architecture.
3) Search From Scratch: Each model is trained from scratch, which will not take full advantage of the architecture of the existing network model and the parameters that have been trained.

B. Proposed Method

In this article, a Bayesian optimization strategy is proposed to develop a DAG network for gear pitting fault diagnosis. The proposed method focuses on the DAG NAS and training hyperparameter optimization using the Bayesian optimization. The overall framework of the proposed method is shown in Fig. 1. Also, the overall process is given in the following four steps.

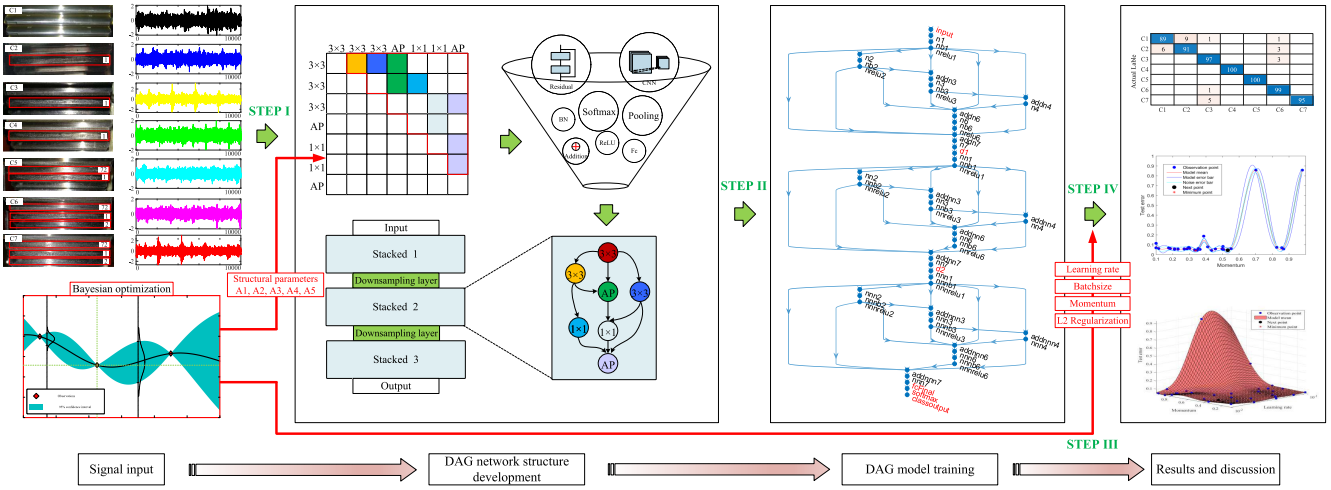


Fig. 1. Framework of the proposed method.

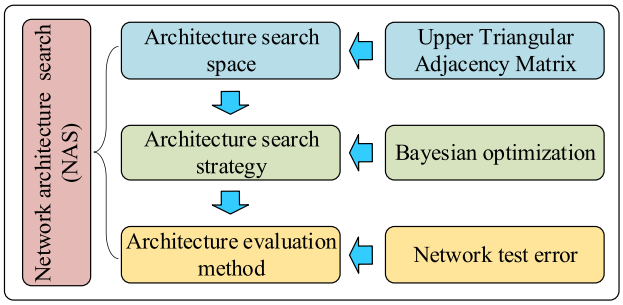


Fig. 2. Block diagram of NAS theory.

1) *STEP I:* First, seven kinds of gear pitting fault vibration signals are processed. The vibration signal of 90 000 signal points of each type of gear fault is divided into 100 segments, each containing 900 data points. After that, each segment of 900 data points is cut into a 30 × 30 2-D image as input.
2) *STEP II:* Search of DAG network architecture based on the Bayesian algorithm. The block diagram of NAS theory is presented in Fig. 2. It can be known from the figure that the NAS mainly includes architecture search space, architecture search measurement, and architecture evaluation method. The representation of the DNN architecture space has always been a difficult point. In this article, the adjacency matrix is used to represent the nodes and edges of the DAG network, so as to obtain the basic layer of the DAG network. The final fault diagnosis model is then constructed by stacking three times and using the link of the output layer of the downsampling layer. The Bayesian optimization algorithm generates the connection matrix by converting the $A1$–$A5$ parameters from decimal to binary and then constructs the DAG network architecture. Finally, the test error of the constructed network is used as a metric for network architecture evaluation.
3) *STEP III:* Training the gear pitting fault diagnosis model. Find the best combination of training hyperparameters (learning rate, batch size, momentum term, and regularization coefficient) through the Bayesian optimization. The number of Bayesian observation points is set to 30, and the acquisition function expected improvement (EI) and kernel function Matern 5/2 are used. Considering the uncertainty of multiple training results of deep residual neural network, the influence of noise is considered to the Gaussian process fitting.
4) *STEP IV:* Finally, gear pitting fault classification is carried out using a well-trained DAG neural network. In addition, the impact of training hyperparameters is discussed and the optimization results of various optimization algorithms are compared.

C. Definition of the Architecture Search Space

Compared with the traditional top-to-down single-channel neural network architecture, the unique feature of the DAG neural network is that it contains a DAG architecture, that is, a data flow branch is created beside the main road. In this way, the data can skip the calculation of the main road if necessary, which can deepen the network depth and enhance the network computing ability.

Fig. 3 shows the proposed fault diagnosis model of the stacked DAG structure. To reduce the architecture search space [30], [31], a DAG network unit with the same structure is used to stack three times to build a diagnostic model. The DAG unit is composed of seven types of convolutional layers and pooling layers.

The seven base layers for building DAGs are described in Table I, where the base layers are ordered according to human experience. In order to further narrow the search space of the network architecture, it is stipulated that the data transmission between layers can only be carried out backward from the leading order. Also, doing so can avoid the emergence of a closed-loop neural network architecture.

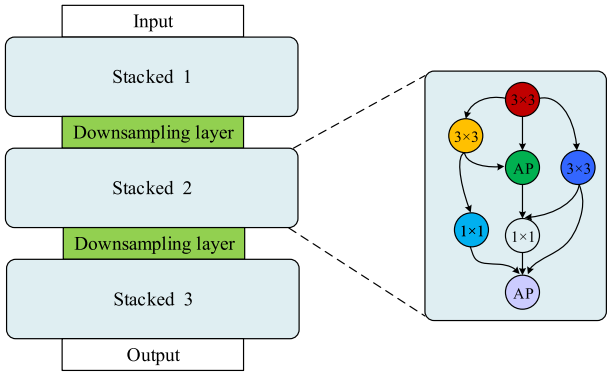


Fig. 3. Demonstration of the stacked diagnostic model.

TABLE I
DESCRIPTION OF THE BASED LAYERS FOR DAG CONSTRUCTION

♯	Layers	Filter size	Stride
n1	Convolutional(3×3)	[3 3]	[1 1]
n2	Convolutional(3×3)	[3 3]	[1 1]
n3	Convolutional(3×3)	[3 3]	[1 1]
n4	Average pooling(AP)	[3 3]	[1 1]
n5	Convolutional(1×1)	[1 1]	[1 1]
n6	Convolutional(1×1)	[1 1]	[1 1]
n7	Average pooling(AP)	[3 3]	[1 1]

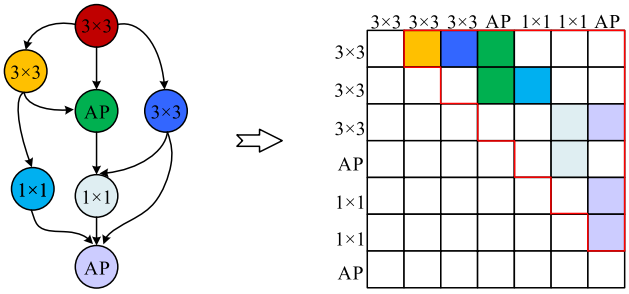


Fig. 4. Example of network architecture converted to adjacency matrix.

In order to carry out the NAS strategy, the DAG structure must be digitally transformed. As shown in Fig. 4, it is an example in which the DAG structure is represented by an adjacency matrix [32], [33]. Due to the previous search space reduction strategy, the adjacency matrix is an upper triangular matrix. The 0 or 1 value at the adjacency matrix position corresponds to whether the construction layer is connected.

The red area of the adjacency matrix in Fig. 4 corresponds to the edge information. There are 21 edges in total, and each edge has two states, so there are 2^{21} types of DAG structures in total.

Faced with such a huge NAS space, it is impossible to try it from scratch. Therefore, the search strategy of the network architecture is very important. This article adopts the Bayesian optimization algorithm as the search strategy, which is described in detail in Section II-D. To cater to the Bayesian optimization algorithm, the 21 binarized edge information in the adjacency matrix needs to be parameterized. However, in order to enrich the network architecture, to ensure that

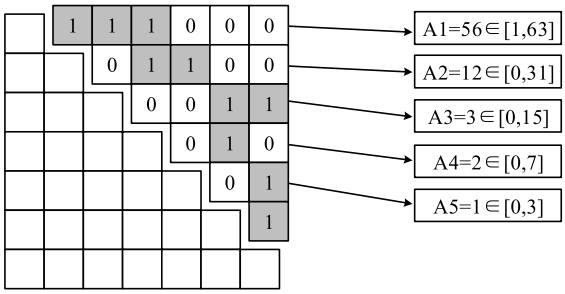


Fig. 5. Decimal conversion of binary adjacency matrix.

layer "n6" is always used, the parameter in the sixth row of the adjacency matrix is fixed to 1. As shown in Fig. 5, we convert the binarized strings of each row of the adjacency matrix into decimal values as parameters [34] so that we get five parameters $A1$–$A5$. The range of parameters is determined by the length of the binarized string (the number of edges). Since it is necessary to ensure that layer "n1" has data output, the value of $A1$ cannot be 0.

It is worth noting that only the main layers are shown when constructing the DAG in Table II and Fig. 3. In fact, each convolutional layer is followed by a batch normalization (BN) layer and an activation layer (ReLU). Moreover, in the DAG network structure, the multi-input data are fused by summation, which requires all convolutional layers [35] in the DAG network to have the same output size and number of filters. The mathematical is implemented as follows:

$$\boldsymbol{x}_{l+1}^{m} = \sum_{i=1}^{I} \boldsymbol{W}_m * \boldsymbol{x}_l^c + b_m \tag{1}$$

where $\boldsymbol{W}_m$ is the mth kernel matrix, $\boldsymbol{x}_l^c$ is the input data matrix extracted by the sliding filter, * is the element-wise multiplication, b_m is the offset of the mth convolution kernel, and $\boldsymbol{x}_{l+1}^{m}$ is the mth feature map. An activation function ReLU [36] performs the $f(\cdot)$ operation, which is expressed as

$$f\left(\boldsymbol{x}_{l+1}^{m}\right) = \max\left(0, \boldsymbol{x}_{l+1}^{m}\right). \tag{2}$$

The BN operation [37] is designed to adjust the distribution of the output data of the convolution operation to speed up network training. It can be expressed as

$$\tilde{\boldsymbol{x}}^{m} = \frac{\boldsymbol{x}^{m} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3}$$

where $\boldsymbol{x}^m$ is the output of convolution, μ and σ^2 are the mean and the variance for each observation in each channel m, respectively, ϵ is a very small positive number to improves numerical stability, and $\tilde{\boldsymbol{x}}^m$ is the normalized output.

In this article, the addition of DAG network channels is fused by addition as

$$\boldsymbol{y} = \mathrm{ReLU}(f(\boldsymbol{x_1} + f(\boldsymbol{x_2})) \tag{4}$$

where x_1 and x_2 are the multiple inputs.

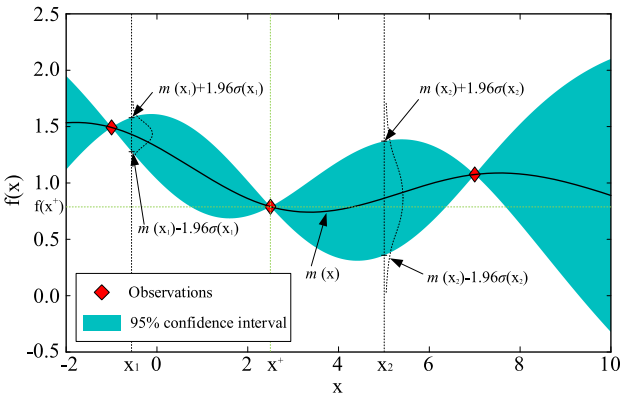


Fig. 6. Example of 1-D Gaussian process with three observations.

D. Architecture Search Strategy

In this article, a Bayesian optimization algorithm is used as the search strategy of NAS. The parameters $A1$–$A5$ are selected in the range by means of Bayesian optimization. The Bayesian optimization algorithm [38], [39] realizes the optimization of hyperparameters by continuously adding observation points and performing the Gaussian process to fit the objective function. The mathematical expression of the optimal hyperparameters x_{best} of the objective function is given as follows:

$$x_{\text{best}} = \text{argmin}_{x\in\chi} f(x) \tag{5}$$

where $f(x)$ is the objective function and χ is the hyperparameter space.

Fig. 6 shows a 1-D Gaussian process with three observation points. The multiple observation points conform to the joint Gaussian distribution. The mathematical expression is given as follows:

$$\begin{cases} f(x) \sim GP\big[m(x), k\big(x, x'\big)\big] \\ m(x) = E[f(x)] \\ k\big(x, x'\big) = E\big[\big(f(x) - m(x)\big)\big(f\big(x'\big) - m\big(x'\big)\big)\big] \end{cases} \tag{6}$$

where the mean $m(x)$ and variance $k(x, x')$ of the Gaussian distribution are also given above.

According to the calculation process of the joint Gaussian distribution in (6), it can be inferred that the joint Gaussian distribution containing n observation points can be expressed as follows:

$$\begin{cases} f(x_{1:n}) \sim GP[m(x_{1:n}), \mathbf{K}] \\ m(x_{1:n}) = E[f(x_{1:n})] \\ \mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \end{cases} \tag{7}$$

where the value of the kernel function $k(x_i, x_j)$ changes between [0, 1] with the distance between points x_i and x_j from far to near. The commonly used kernel functions [40], squared exponential kernel, improved squared exponential kernel, Matern 3/2, and Matern 5/2, are given as follows:

$$k(x_i, x_j) = \begin{cases} \exp\left(-\dfrac{(x_i - x_j)^2}{2}\right) \\ \sigma_f^2 \exp\left(-\dfrac{(x_i - x_j)^2}{\sigma_l^2}\right) \\ \sigma_f^2\left(1 + \dfrac{\sqrt{3}r}{\sigma_l}\right)\exp\left(-\dfrac{\sqrt{3}r}{\sigma_l}\right) \\ \sigma_f^2\left(1 + \dfrac{\sqrt{5}r}{\sigma_l} + \dfrac{5r^2}{3\sigma_l^2}\right)\exp\left(-\dfrac{\sqrt{5}r}{\sigma_l}\right). \end{cases} \tag{8}$$

Then, the next observation point x_{n+1} and the existing observation points $x_{1:n}$ still follow the joint Gaussian probability distribution, so we can get:

$$\begin{cases} \begin{bmatrix} f(x_{1:n}) \\ f(x_{n+1}) \end{bmatrix} \sim N\left(m\left(\begin{bmatrix} x_{1:n} \\ x_{n+1} \end{bmatrix}\right), \begin{bmatrix} \mathbf{K} & k_* \\ k_*^T & k(x_{n+1}, x_{n+1}) \end{bmatrix}\right) \\ k_* = \big[k(x_{n+1}, x_1), k(x_{n+1}, x_2), \cdots k(x_{n+1}, x_n)\big]. \end{cases} \tag{9}$$

Therefore, the probability distribution of the observation point x_{n+1} in the entire coordinate system can be easily obtained as

$$\begin{cases} P(f(x_{n+1})|D_{1:n}, x_{n+1}) \sim N\big(\mu(x_{n+1}), \sigma^2(x_{n+1})\big) \\ \mu(x_{n+1}) = k_*^T \mathbf{K}^{-1} f(x_{1:n}) \\ \sigma^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k_*^T \mathbf{K}^{-1} k_* \end{cases} \tag{10}$$

where $D_{1:n}$ represent the positions of observation points $\{x_{1:n}, f(x_{1:n})\}$. The position of the next observation point can be determined according to the obtained probability distribution of x_{n+1} at each location and combined with the acquisition function

$$x_{\text{next}} = \text{argmax}_{x\in\chi} \text{Acq}(x) \tag{11}$$

where the next observation point x_{next} corresponds to get the maximum value in the function acquisition function.

The commonly used acquisition function [41] lower confidence bound (LCB), upper confidence bound (UCB), probability of improvement (PI), and EI, are given as follows:

$$\text{Acq}(x) = \begin{cases} k\sigma_Q(x) - \mu_Q(x) \\ k\sigma_Q(x) + \mu_Q(x) \\ \Phi\big(\big(f\big(x^+\big) - \mu_Q(x) - \xi\big)/\sigma_Q(x)\big) \\ \big(f\big(x^+\big) - \mu_Q(x) - \xi\big)\Phi(Z) + \sigma_Q(x)\phi(Z) \end{cases} \tag{12}$$

where x^+ is the lowest value among existing observation points.

In order to deal with the uncertainty of the results during DNN training, artificial noise $\epsilon \sim N(0, \sigma_{\text{noise}}^2)$ [42] is added to the prior results when using the Bayesian optimization. Then, the new objective function $g(x)$ and the joint probability distribution of $x_{1:n}$ are given as follows:

$$\begin{cases} g(x) = f(x) + \epsilon \\ g(x_{1:n}) \sim GP\big[m(x_{1:n}), \mathbf{K} + \sigma_{\text{noise}}^2 I\big]. \end{cases} \tag{13}$$

TABLE II
DESCRIPTION OF THE OPTIMIZED HYPERPARAMETERS

Hyperparameters	Range	Type
Initial learning rate(L_r)	[0.01 ∼ 0.1]	Non-integer
Batchsize(B_s)	[10 ∼ 150]	Integer
Momentum(M_o)	[0.1 ∼ 0.98]	Non-integer
L2 Regularization(L_2)	[1e-10 ∼ 0.01]	Non-integer

Thus, the probability distribution of the point x_{n+1} is updated as

$$\begin{cases} P(g(x_{n+1})|D_{1:n}, x_{n+1}) \sim N\left(\mu(x_{n+1}), \sigma^2(x_{n+1}) + \sigma^2_{\text{noise}}\right) \\ \mu(x_{n+1}) = k_*^T\left[\mathbf{K} + \sigma^2_{\text{noise}} I\right]^{-1} g(x_{1:n}) \\ \sigma^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k_*^T\left[\mathbf{K} + \sigma^2_{\text{noise}} I\right]^{-1} k_*. \end{cases} \tag{14}$$

E. Architecture Evaluation Strategy

When using the search strategy to find the optimal architecture in the NAS space, an evaluation result should be fed back to the tried network architecture. This is done on the one hand to evaluate the performance of the network architecture for selecting and on the other hand to provide a reference for the Bayesian optimization acquisition function to generate the next try network. This article evaluates the accuracy of structural performance by using the network to diagnose gear pitting faults. The network error is calculated as follows:

$$E_{\text{NAS}} = 1 - S_g / S. \tag{15}$$

where S_g is the number of correctly diagnosed samples and S is the total sample size.

Then, the obtained network architecture evaluation error is used in (12) to calculate the next generation parameters $A1$–$A5$, thereby generating a new network architecture. In order to speed up the NAS search speed, the total number of network evaluation samples S can be appropriately reduced.

III. HYPERPARAMETERS AND DATASETS

This article aims to build a DAG network using the Bayesian optimization, including two major aspects: NAS and network training hyperparameter optimization. The NAS was introduced in Section II. Here, the optimization of hyperparameters and the development of gear pitting experiments are introduced.

A. Hyperparameters

There are similarities between using Bayesian optimization for network training hyperparameter optimization and searching for network architecture. When searching for a network architecture, it is necessary to convert the network architecture into an adjacency matrix first and then convert the binarized adjacency matrix into decimals as parameters $A1$–$A5$. However, when optimizing training hyperparameters, these steps are omitted. The operation is performed directly using the Bayesian optimization, and the training hyperparameter types and ranges are shown in Table II.

TABLE III
DESCRIPTION OF THE EARLY GEAR PITTING TYPES

Label	Gear pitting type		
	Tooth 1	Tooth 2	Tooth 3
C1	Healthy	Healthy	Healthy
C2	Healthy	10% in middle	Healthy
C3	Healthy	30% in middle	Healthy
C4	Healthy	50% in middle	Healthy
C5	10% in middle	50% in middle	Healthy
C6	10% in middle	50% in middle	10% in middle
C7	30% in middle	50% in middle	10% in middle

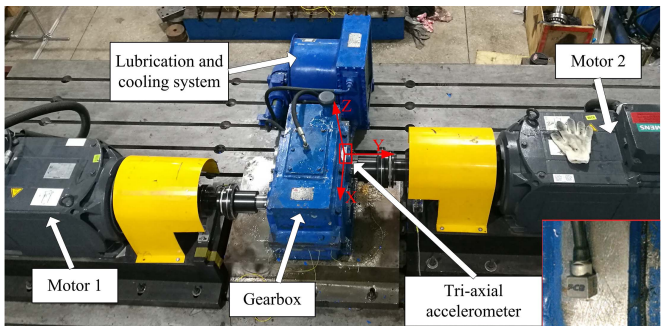


Fig. 7. Experimental test rig of the early gear pitting faults.

Here, the training hyperparameters, such as initial learning rate and batch size, are all set to an approximate range. It is worth noting that the data type of the hyperparameter largely determines the choice of the optimization algorithm.

B. Experiment Setup and Data Processing

The gear pitting vibration signal used for model validation was obtained from the gearbox test rig in Fig. 7. The gearbox test rig is driven by two 45-kW Siemens servo motors: motor 1 is the driving motor and motor 2 is the loading motor. Also, the gearbox contains a pair of spur gears with 40 teeth and 72 teeth, the pinion gear is the driving gear (including 40 teeth, module 3 mm), and the large gear is the driven gear (including 72 teeth and module 3 mm). The three-axis vibration accelerometer sensor was installed on the bearing house (the red box in the figure) of the gearbox to collect vibration signals with 10 240 and Hz sampling rate. The specific descriptions of the seven gear pitting types (C1–C7) set in the experiment are given in Table III. Each gear type performs five times signal acquisition independently and a total of 35 sets of vibration signal for seven types of gears. In the experiment, the vibration signal was collected under a variety of working conditions. The speed and torque range of the motor were 100–1000 r/min and 50–500 Nm, respectively. Pictures of the seven types of gear faults and the corresponding vibration signals are shown in Fig. 8.

In this article, seven kinds of gear pitting fault vibration data under 1000-r/min–100-Nm working conditions are used to validate the diagnosis model. A total of 35 sets of vibration signals collected in the above experiment. Among them, 28 sets of signals are used as training data, and the remaining

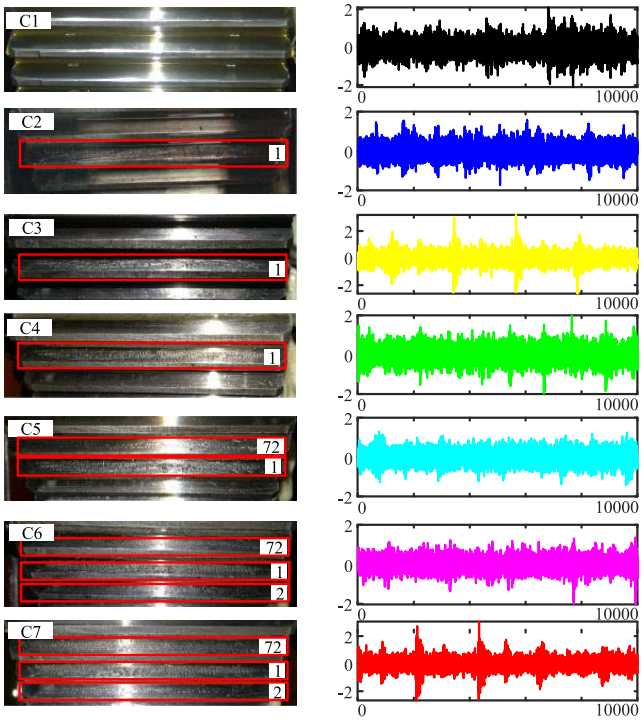


Fig. 8. Gear pitting fault picture and corresponding vibration signal.

seven sets of signals are used as test data. In order to facilitate the collection of signals as the input of the developed residual DNN, the collected vibration signals are cut into several segments of 900 data points. Thus, the size of the training data and test data is $30 \times 30 \times 2800$ and $30 \times 30 \times 700$, respectively.

IV. Results and Analysis

In this article, the Bayesian optimization is used to construct a DAG network to achieve high-accuracy diagnosis of gear pitting faults. The Bayesian optimization of the DAG process is divided into two parts: 1) DAG NAS and 2) network training hyperparameter optimization. This section presents the results of these two parts and expands the discussion.

A. Results of DAG Architecture Searching

This article uses the Bayesian optimization for regular NAS. The seven types of convolutional layers and pooling layers ($n1$–$n7$) used to construct the DAG unit are first specified in Table I. In order to reduce the NAS search space, these basic neural network layers are used to build the DAG unit, then stack it three times, and cooperate with two downsampling layers ($d1$ and $d2$) to build a gear pitting fault diagnosis model, as shown in Fig. 3.

To search the DAG unit architecture using the Bayesian optimization, the DAG architecture must be parameterized. In this article, the binary adjacency matrix is first used to represent the nodes and edges of the DAG unit, and then, each row of the adjacency matrix is converted from binary to decimal to obtain the parameters $A1$–$A5$, as shown in Fig. 5. Finally, the Bayesian optimization is based on the structural parameters $A1$–$A5$ to carry out optimization, that is, to repeat the process: build network according to parameters $A1$–$A5 \rightarrow$ get network performance error $\rightarrow$ generate new parameters $A1$–$A5$ based on the prior knowledge.

In this article, fixed hyperparameters are used to train the model in the NAS and the network performance error is obtained. The "sgdm" algorithm is used for model correction, the initial learning is set to 0.02 with a 50% drop in every 60 epochs, the number of mini-batch samples is set to 50, and the momentum term (M_o) and regularization coefficient $L2$ are 0.5 and 0.001, respectively.

Table IV shows the 30-iteration process of searching for the best network architecture using the Bayesian optimization and presents 30 DAG unit network architectures, as well as the corresponding training time, performance accuracy, and evaluation results. It can be seen that the s24 has the best performance among the 30 network architectures.

The network architecture in the table is represented by the architecture parameters $A1$–$A5$, which is not intuitive enough. To this end, Fig. 9 shows part of the DAG network architectures in Table IV. The network architecture in the figure is completely built on the Bayesian generation parameters, which can abandon the inherent thinking of people and develop a better network architecture. These DAG units not only contain seven base layers ($n1$–$n7$) but also add BN layers (nb) and activations (nrelu) to the base layers. It includes the first five network architectures of s1–s5, and the four network architectures obtained by s7, s9, s19, and s24 for the "Best" evaluation. Combining Table IV and Fig. 9, it can be found that the more complex the DAG unit architecture, the longer the network training time. For example, the training time of network architecture s1 is nearly twice that of s2.

The complete gear pitting diagnostic model is shown in Fig. 10, which is based on the best performing DAG unit stacked three times. The average-pooling layers $d1$ and $d2$ are used to connect the DAG units, the filter size [4, 4], and the stride [2, 2]. In addition, the model output contains a fully connected layer of seven neurons and a softmax layer.

This article also compares the diagnostic results of the fully connected DNN, the standard CNN, and the proposed method for seven gear types. The accuracy of C1–C7 gear fault diagnosis is given in Table V, and the gear pitting fault test confusion matrix is shown in Fig. 11.

B. Results of Bayesian Optimization Hyperparameters

This section mainly discusses the influence of training hyperparameters on the diagnosis results. Use the network architecture in Fig. 10 as the basis to test the effect of Bayesian optimization on training hyperparameters. The Bayesian optimization process for solving the training hyperparameters is the same as for solving the structural parameters $A1$–$A5$. The 30 times hyperparameter set explorations were performed, and the DAG model training settings were kept the same. The ranges of the four training hyperparameters are shown in Table II, and the optimization results are shown in Table VI. The test accuracy of the best model in Table VI is 0.9857,

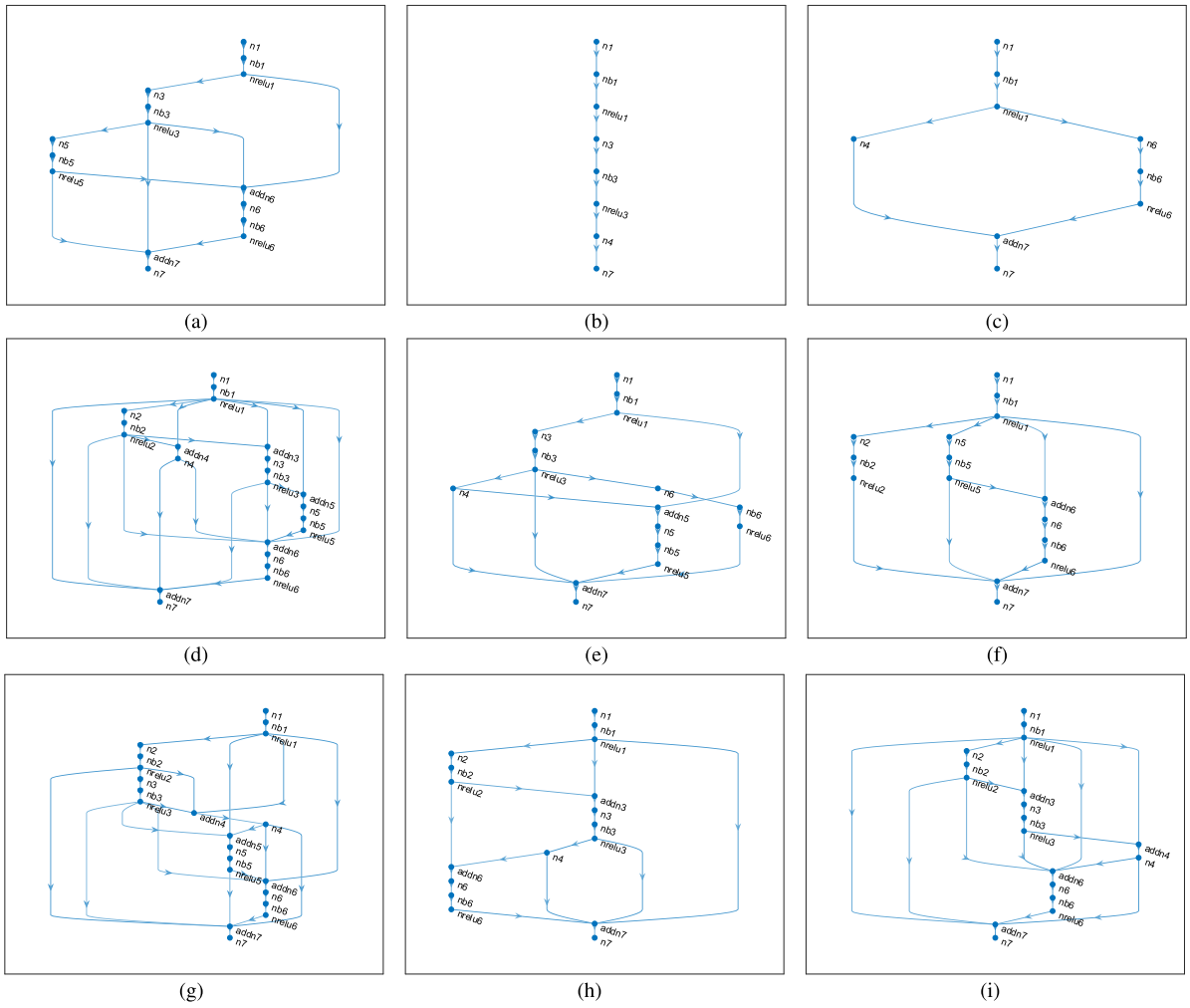


Fig. 9. Partial network architecture units display of Bayesian optimization search. (a) s1. (b) s2. (c) s3. (d) s4. (e) s5. (f) s7. (g) s9. (h) s19. (i) s24.

which is improved compared to Table IV. It can be concluded that training hyperparameter optimization training can improve the accuracy of the trained model to a certain extent. Moreover, based on the optimal hyperparameter combination obtained in Table VI, the influence of one single hyperparameter on the diagnostic results is also discussed one by one in this section.

First, the training initial learning rate is discussed. Fig. 12(a) shows the process of using the Bayesian optimization to find the best initial learning rate value. The figure shows trying various learning rates to find the optimal value based on the optimal network architecture. It can be seen that the optimal model architecture is very robust to the learning rate value. The results show that it is easier to get a small error model when the learning rate value is around 0.06. Fig. 12(b) shows the effect of batch size on training the model. It is easier to get good training models with batch sizes of around 40. The trend in the figure is probably that smaller batch size values result in smaller error values. However, the sample batch size will directly affect the training time of the model. Setting a smaller sample batch size will reduce the training speed of the model. Therefore, the batch size should be selected considering the tradeoff between model accuracy and training speed.

Fig. 12(c) shows the impact of the momentum term on model training. It can be seen from the figure that the fluctuation is small in the range of 0.1–0.5, and the stability of model training is poor after it is greater than 0.6. This is also because the Bayesian optimization strategy focuses more on the momentum term mining when the model error is small. Fewer attempts in the 0.6–1 range are prone to fluctuations. Fig. 12(d) shows the effect of the regularization coefficient $L2$ on model training. The curve in the figure is arched, and the search results show that it is easier to train an excellent DAG diagnostic model when $L2$ is 0.01.

All subgraph curves in Fig. 12 are obtained after 30 times verification and fitting using a Gaussian process distribution. Considering the uncertainty of deep residual neural network training, the effect of noise error is added in curve fitting. It can be seen from Fig. 12 that the dispersion of verification points is large, and the test errors of similar hyperparameter combination values are not similar. It is precise because the number of validation points is limited and the results of repeatedly training the deep residual network are not fixed, the fit curve does not well reflect the hyperparameter–test error relationship.

TABLE IV
RESULTS OF STRUCTURAL PARAMETER OPTIMIZATION USING THE BAYESIAN OPTIMIZATION METHOD

♯	Assess	Structural parameters					Time	Accuarcy
		A1	*A2*	*A3*	*A4*	*A5*		
s1	Best	18	1	7	5	3	204.32	0.9200
s2	Accept	16	19	12	1	0	116.89	0.9057
s3	Accept	10	24	1	1	2	124.30	0.8886
s4	Accept	63	27	7	3	2	254.78	0.9129
s5	Accept	20	1	11	5	1	198.58	0.9186
s6	Accept	2	3	2	5	0	113.46	0.8871
s7	Best	39	1	9	7	3	195.70	0.9357
s8	Accept	1	22	9	2	0	80.07	0.9000
s9	Best	46	25	15	7	3	249.51	0.9471
s10	Accept	62	5	15	7	3	251.29	0.9200
s11	Accept	46	11	15	6	0	168.32	0.9429
s12	Accept	47	1	5	7	0	163.82	0.9229
s13	Accept	41	24	15	7	3	247.98	0.9271
s14	Accept	49	15	15	6	0	203.94	0.9357
s15	Accept	46	0	15	4	0	113.95	0.8900
s16	Accept	49	31	15	7	1	256.68	0.9357
s17	Accept	44	5	14	6	0	159.51	0.9086
s18	Accept	48	19	5	7	0	192.41	0.9457
s19	Best	49	22	13	7	0	204.79	0.9514
s20	Accept	49	20	10	7	0	207.79	0.9314
s21	Accept	2	2	3	0	0	115.43	0.8400
s22	Accept	36	20	2	7	0	181.59	0.9214
s23	Accept	1	28	11	7	0	81.62	0.9014
s24	**Best**	**59**	**23**	**14**	**7**	**0**	**213.50**	**0.9586**
s25	Accept	61	24	13	7	0	202.53	0.1429
s26	Accept	2	14	10	6	2	115.25	0.8786
s27	Accept	20	23	3	5	1	190.99	0.9171
s28	Accept	55	14	12	3	1	250.95	0.9429
s29	Accept	59	14	5	2	2	249.77	0.9129
s30	Accept	13	2	11	0	1	115.28	0.9000

TABLE V
COMPARISON OF THE RESULTS OF THREE DIAGNOSTIC METHODS

Methods	DNN	CNN	The proposed method
C1	0.33	0.80	0.89
C2	0.29	0.90	0.91
C3	0.35	0.87	0.97
C4	0.43	1	1
C5	0.69	1	1
C6	0.37	0.80	0.99
C7	0.48	0.86	0.95
Average	**0.42**	**0.89**	**0.9586**

In addition to discussing the effects of a single hyperparameter, this section also discusses the combined effect of two hyperparameters such as the learning rate and momentum both acting together to network feedback error correction. Therefore, fixing other hyperparameters and only discussing the impact of a single hyperparameter on test error have obvious disadvantages. Therefore, as shown in Fig. 13, the impact of the joint hyperparameter group on the network test error is discussed. The results show that the network test error fluctuates greatly with the hyperparameters, but the optimal hyperparameter combination can still be roughly selected.

C. Comparison of Those Optimization Methods

In order to compare with the Bayesian optimization results, this section uses five hyperparameter comparison optimization

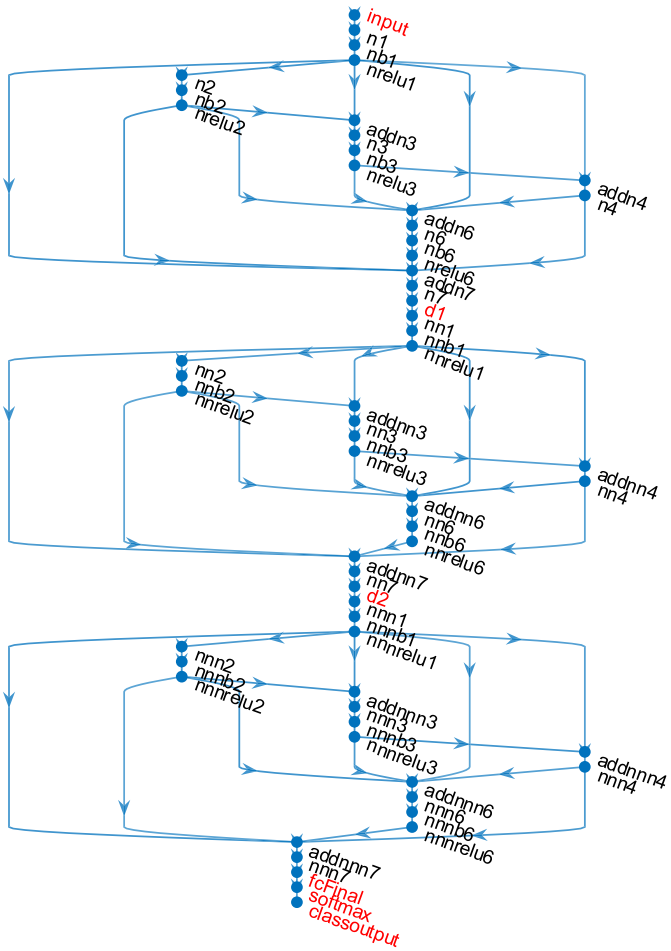


Fig. 10. Deep DAG fault diagnosis model developed by the Bayesian optimization.

methods such as grid search, random search, particle swarm optimization (PSO), GA, and grey wolf optimization (GWO) to optimize the DNN gear pitting model. The principles of these optimization methods are different, and the advantages and disadvantages of each method are compared and analyzed according to the optimization process and optimization results. Since the trial values generated by the random search, PSO, and GWO are all noninteger, they are not suitable for integer hyperparameter optimization with a small range of values. Therefore, this section does not compare and discuss these methods for NAS. In order to compare with the best results of Bayesian optimization in Table VI, all the comparison methods in this section are established under the DAG network architecture, as shown in Fig. 10. The best hyperparameter groups and fault diagnosis results obtained by the six optimization methods are shown in Table VII. The six optimization methods are described by item as follows.

1) Grid Search: The grid search method trains the model by regularly selecting several parameter combinations. This section selects three values for each hyperparameters, $L_r = [0.01, 0.05, 0.1]$, $B_s = [20, 80, 150]$, $M_o = [0.1, 0.5, 0.98]$, and $L_2 = [0.001, 0.005, 0.01]$. There are four hyperparameters, and each hyperparameter has three values, so the number of grid search attempts is $3^4 = 81$. This method is simple and easy to implement, but the search range is small and the

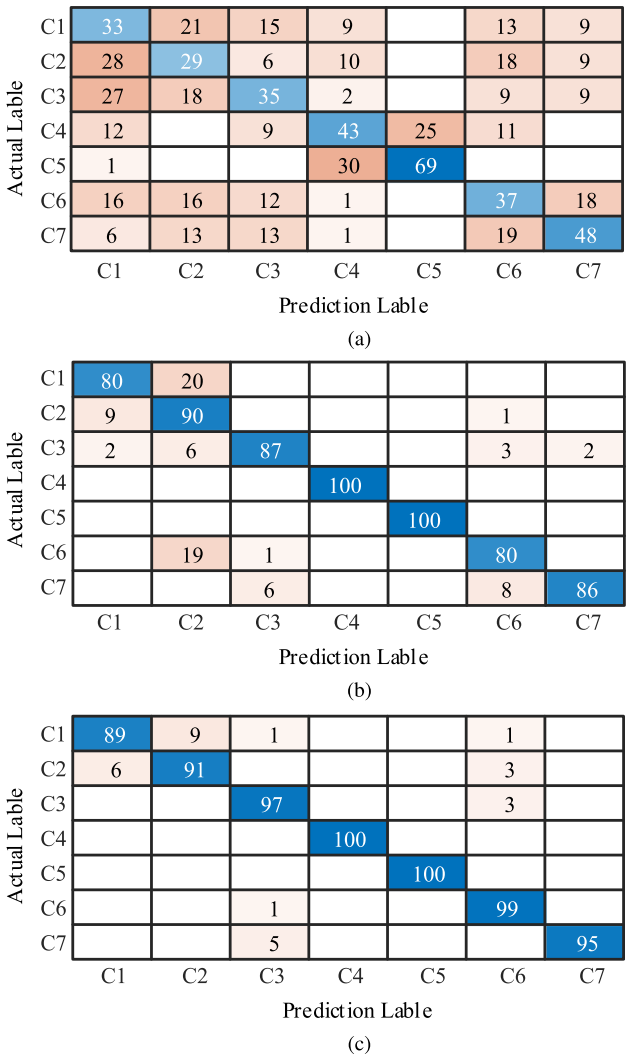


Fig. 11. Gear pitting fault test confusion matrix. (a) DNN. (b) CNN. (c) Proposed method.

TABLE VI

RESULTS OF HYPERPARAMETER OPTIMIZATION USING THE BAYESIAN OPTIMIZATION METHOD

♯	Assess	Hyperparameters				Time	Accuarcy
		L_r	B_s	M_o	L_2		
1	Best	0.0361	25	0.7070	8.36E-04	382.90	0.9486
2	Accept	0.0955	149	0.8386	6.20E-09	66.98	0.1429
3	Accept	0.0525	54	0.3029	5.10E-09	176.26	0.9471
4	Accept	0.0210	148	0.9297	1.96E-03	68.71	0.9457
5	Accept	0.0260	81	0.4625	2.36E-04	119.98	0.9343
6	Accept	0.0100	54	0.6416	7.50E-09	177.03	0.9386
7	Accept	0.0139	147	0.7350	1.45E-07	70.64	0.9271
8	Accept	0.0448	132	0.1008	4.18E-08	76.30	0.9386
9	Accept	0.0181	20	0.1988	2.27E-08	478.65	0.9471
10	Accept	0.0321	148	0.8536	8.81E-10	66.94	0.9457
11	Accept	0.0112	148	0.9259	1.10E-04	66.66	0.9429
12	Accept	0.0193	149	0.8556	2.00E-06	67.74	0.9371
13	Accept	0.0495	149	0.8072	1.15E-06	66.50	0.1429
14	Accept	0.0102	148	0.1165	8.20E-06	67.24	0.8400
15	Accept	0.0288	149	0.6852	5.34E-07	66.60	0.9386
16	Accept	0.0273	71	0.9307	1.37E-03	134.19	0.9457
17	Accept	0.0383	47	0.1167	2.18E-05	211.64	0.9257
18	Accept	0.0366	149	0.3806	2.73E-03	66.77	0.9271
19	Accept	0.0629	148	0.1027	1.15E-07	67.97	0.9357
20	Accept	0.0240	147	0.1087	2.86E-07	70.33	0.9386
21	Accept	0.0153	144	0.9763	5.22E-05	69.69	0.8743
22	Accept	0.0136	148	0.3428	1.34E-05	67.10	0.9171
23	Accept	0.0190	146	0.4802	7.72E-04	70.53	0.9171
24	Accept	0.0968	139	0.1243	1.49E-07	73.13	0.9271
25	Accept	0.0149	144	0.1039	3.31E-05	69.32	0.9114
26	Accept	0.0100	145	0.9774	5.55E-06	69.46	0.8857
27	**Best**	**0.0530**	**69**	**0.1280**	**7.90E-06**	**136.88**	**0.9857**
28	Accept	0.0799	136	0.1079	1.16E-08	73.14	0.9214
29	Accept	0.0113	149	0.7465	2.39E-09	67.46	0.9300
30	Accept	0.0102	149	0.4235	1.38E-09	67.02	0.9143

TABLE VII

COMPARISON OF THE BEST HYPERPARAMETERS OPTIMIZATION RESULTS OF DIFFERENT METHODS

Methods	Hyperparameters				Accuracy
	L_r	B_s	M_o	L_2	
Grid search	0.0100	20	0.5000	1.000E-02	0.9429
Random search	0.0242	48	0.3616	6.486E-03	0.9429
Particle swarm optimization	0.0282	116	0.8038	6.650E-03	0.9386
Genetic algorithm	0.0746	59	0.2692	8.763E-03	0.9457
Grey wolf optimization	0.0438	41	0.4962	1.000E-02	0.9443
Bayesian optimization	**0.0530**	**69**	**0.1280**	**7.900E-06**	**0.9857**

number of training attempts is too many. From Fig. 14, it can be found that the four hyperparameters have a greater impact on the diagnostic accuracy. In order to ensure a high level of stable test accuracy of the diagnostic model, the value of the hyperparameter group should be selected as $L_r = 0.01$, $B_s = 20$, $M_o = 0.5$, and $L_2 = 0.01$.

2) Random Search: Compared with grid search, random search randomly selects hyperparameter points in the range and combines them. The selection range of hyperparameters is the same as that of Bayesian optimization in Table II. This section conducts 30 times random hyperparameter search trial. Compared with grid search, it can be seen from Table VII that random search has the same results as the grid search method. However, the number of search attempts is less than half of the former, and there is no need to manually determine the grid point value. The disadvantage is that the results are highly random, and there is no regularity between multiple attempts. From Table VII and Fig. 15, it can be seen that random search can obtain a higher diagnostic accuracy rate with fewer attempts but cannot infer the optimal hyperparameter range according to its optimization process.

3) Particle Swarm Optimization: The particle swarm algorithm finds the optimal hyperparameter group through multiple iterations according to the guidance of the optimal particles in the swarm. It solves the nature of grid search and random search without result feedback. As shown in the figure, the particle swarm has undergone five iterations, and the circles with five colors of blue, red, orange, purple, and green are used to correspond to the particles of the first generation to the fifth generation. It can be seen that the range of the particle swarm is gradually concentrated with the iterative search. In Fig. 16, the number of particles is set to 10, and the number of iterations is 5. The optimal selection range of hyperparameters for optimization results is that L_r is between

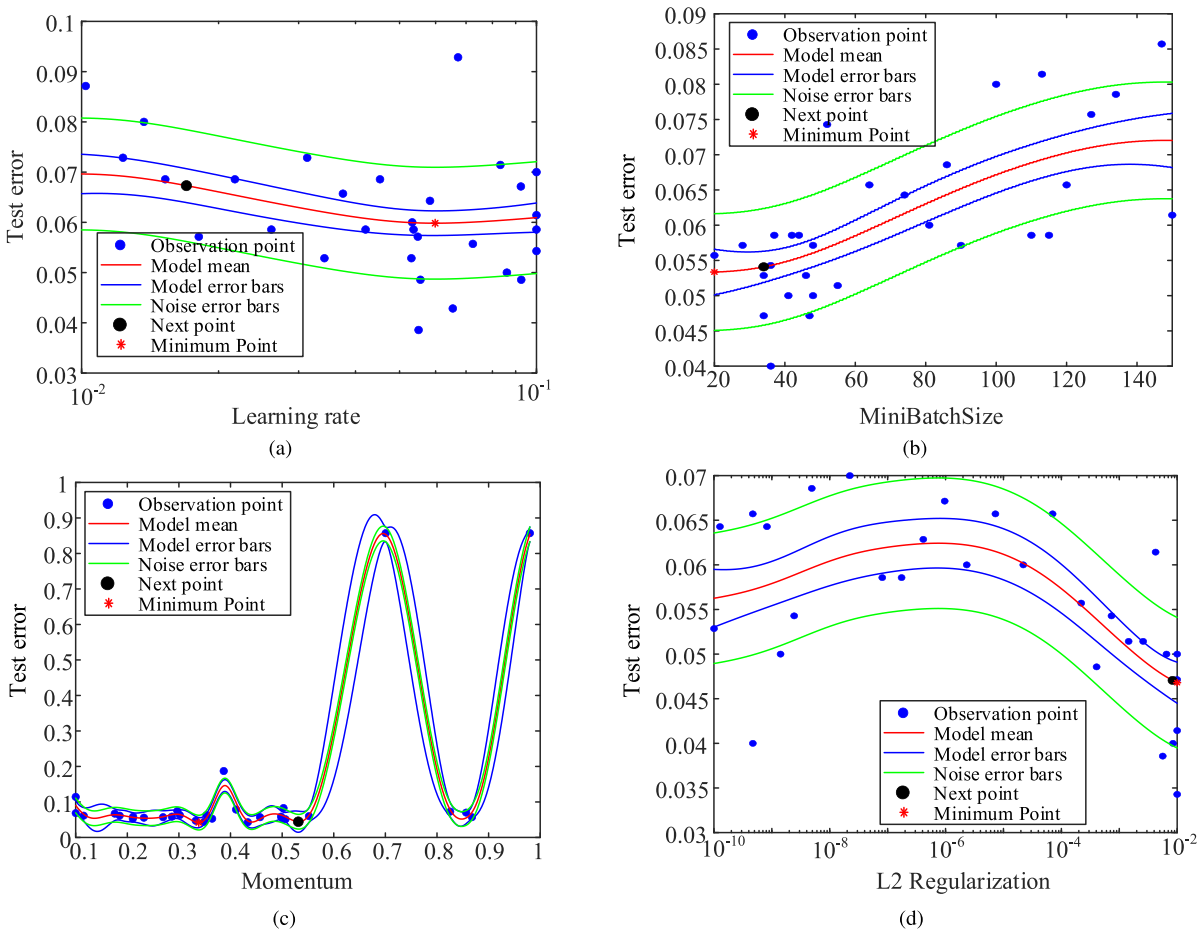


Fig. 12. Effect of a single hyperparameter on test error. (a) Learning rate. (b) Mini-Batchsize. (c) Momentum. (d) L_2 Regularization.

TABLE VIII
COMPARISON OF ADVANTAGES AND SHORTCOMINGS OF SIX TYPES OF OPTIMIZATION METHODS

♯	Methods	Accuracy	Attempts	Advantage	Shortcoming
1	Grid search	0.9429	81	1.Simple and easy to implement.	1.Takes too many attempts. 2.Independence between attempts.
2	Random search	0.9429	30	1.Simple and easy to implement. 2.Takes lower attempts. 3.Larger search scope.	1.High randomness of the results. 2.Independence between attempts.
3	Particle swarm optimization	0.9386	50	1.Regularly doing attempts.	1.Takes too many attempts. 2.Unable to integer hyperparameters. 3.Unbalance exploitation and exploration.
4	Genetic algorithm	0.9457	50	1.Search in more detail.	1.Heavily depend on primary genes. 2.Small searching scope. 3.Unable to integer hyperparameters.
5	Grey wolf optimization	0.9443	50	1.Larger search scope.	1.Takes large search group. 2.Unable to integer hyperparameters. 3.Unbalance exploitation and exploration.
6	Bayesian Optimization	**0.9857**	30	1.Considering DNN result noise error. 2.Infer validation points from priors.	1.The algorithm is complicated.

0.02 and 0.04, B_s is around 120, M_o is around 0.7, and the L_2 coefficient is between 0.004 and 0.008.

4) Genetic Algorithm: It selects individuals with excellent performance as parents for genetic cross inheritance. The advantage of this method is that the search for hyperparameter combinations is more detailed, and the disadvantage is that the gene type completely depends on the randomly generated first-generation genes. Gene mutation operations can appropriately

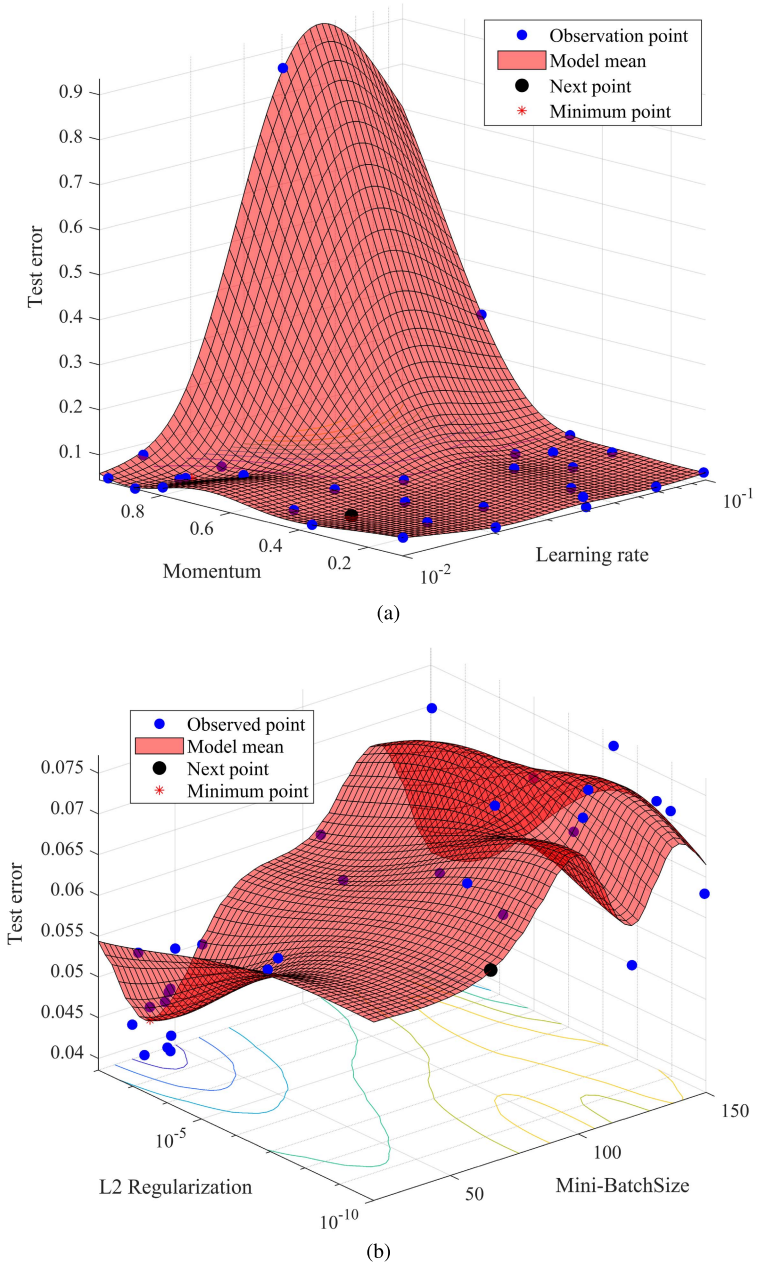


Fig. 13. Effect of joint hyperparameters on test error. (a) Joint effect of initial learning rate and momentum term on test error. (b) Joint effect of initial minibatch size and L_2 regularization term on test error.

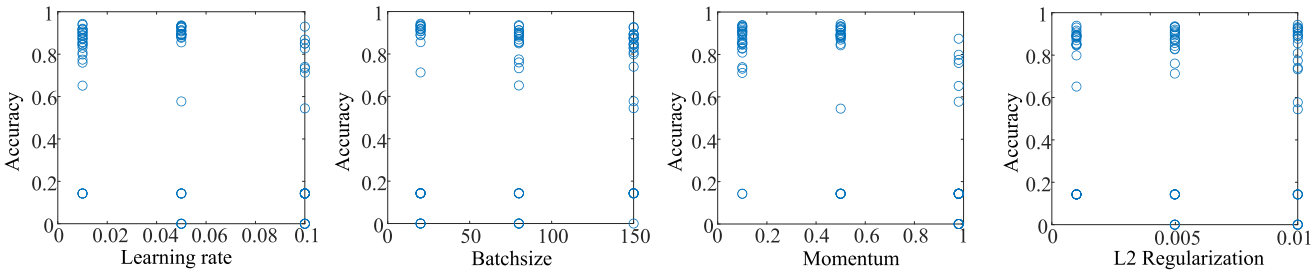


Fig. 14. Hyperparameter optimization process and results using a grid search method.

improve the situation with fewer gene types. In this article, the initial number of genes for each type is set to 10 and processes five generations of gene mutation and elimination. As shown in Fig. 17, the final remaining genes are $L_r = [0.0221, 0.0386, 0.0513, 0.0828]$, $B_s = [26, 58, 64]$, $M_o = [0.2074, 0.2636, 0.4296]$, and $L_2 = [5.864\text{E}^{-3}, 8.743\text{E}^{-3}]$.

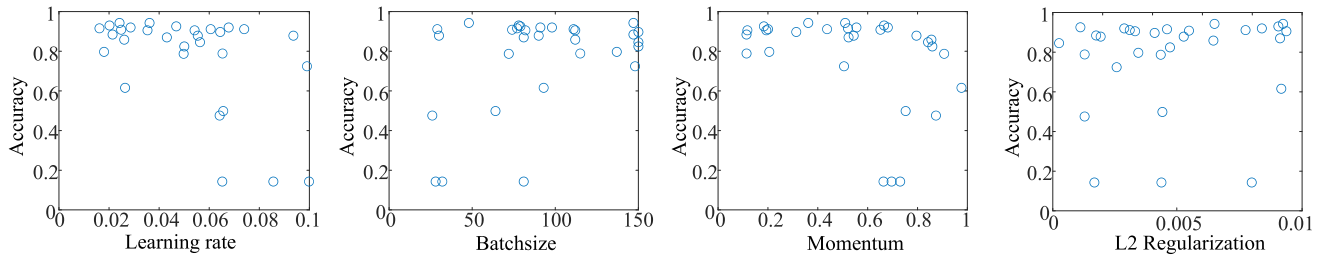


Fig. 15. Hyperparameter optimization process and results using a random search method.

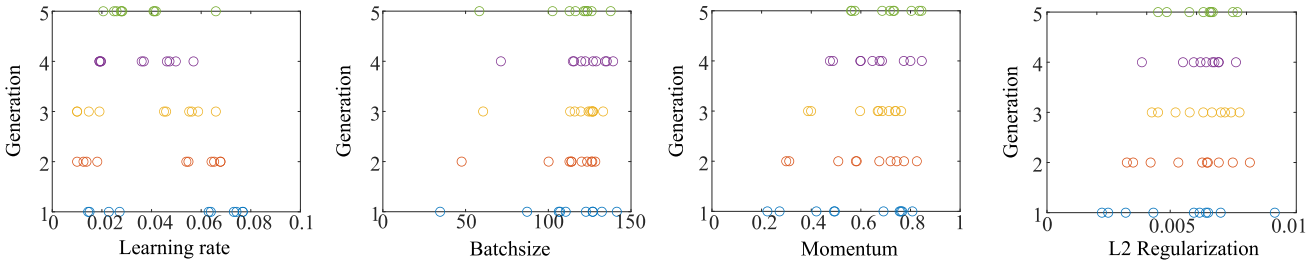


Fig. 16. Hyperparameter optimization process and results using PSO.

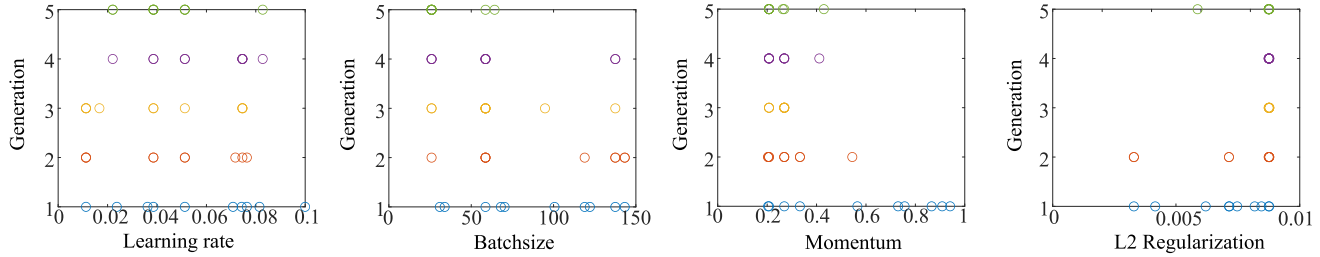


Fig. 17. Hyperparameter optimization process and results using GA.

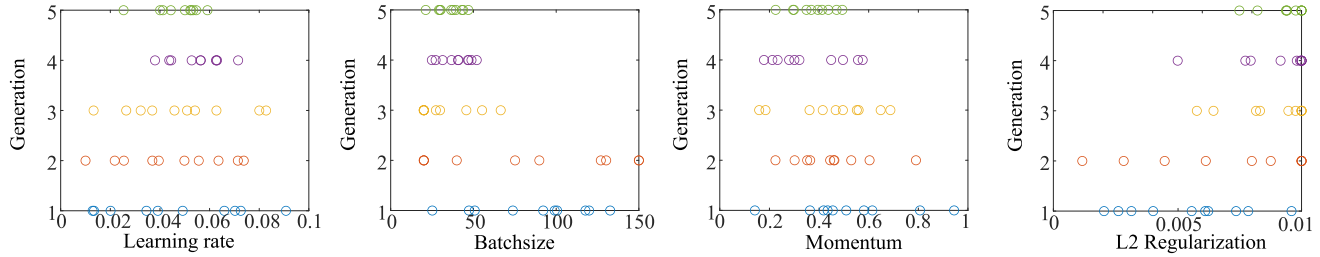


Fig. 18. Hyperparameter optimization process and results using GWO.

5) Gray Wolf Optimization: The optimization idea of the gray wolf algorithm is similar to that of the particle swarm algorithm. They all through excellent individuals to guide the group movement to find the best hyperparameters. The difference is that the gray wolf algorithm increases the number of command individuals from 1 to 3. This makes the iterative optimization process more stable but requires a larger search population. The wolf pack size is set to 10, after five wolf pack location updates. Comparing the search process in Figs. 16 and 18, it can be found that the search range of GWO is larger, and the convergence speed is not as good as that of the PSO algorithm.

6) Bayesian Optimization: It builds an objective function from prior data to find the best set of hyperparameters. Compared with other algorithms, the biggest advantage is that it considers the error when training the DNN model multiple times with the same hyperparameters.

According to the description of the above points, Table VIII provides a brief comparison of the advantages and shortcomings of the six optimization methods.

V. Conclusion

This article addressed the problems of model development and hyperparameter selection when training gear pitting fault diagnosis model based on deep DAG neural networks. The Bayesian optimization method is used to search the best deep DAG neural network architecture and optimize the training hyperparameters such as learning rate (L_r), batch size (B_s),

momentum (M_o), and $L2$ coefficients (L_2). In addition, the grid search, random search, PSO, GA, and GWO are used to compare and analyze the influence of training hyperparameters on model testing errors. The advantage of the proposed method is that it can avoid the limitations of human thinking and automatically construct a network structure with stronger processing power. The disadvantage is that the network structure search time is affected by the search strategy, and the results are greatly affected by the definition of the search space. The detailed conclusions obtained are given as follows.

1) The DAG network architecture can be digitized through the binary adjacency matrix transformed by the decimal parameters, and then, Bayesian optimized NAS can be realized.
2) The diagnostic accuracy of the best DAG neural network model searched by the Bayesian optimization reaches 0.9586, which is better than traditional DNN and CNN. Also, after Bayesian optimization training hyperparameters, a higher network diagnosis accuracy of 0.9857 is obtained.
3) The reason why the Bayesian optimization algorithm is better than grid search and random search is that it includes result feedback and tries point inference. The reason why Bayesian optimization is superior to algorithms such as PSO, GA, and GWO is that it considers the error situation of multiple results of DNN training. Of course, random search may perform better than the proposed method when the number of attempts is small.
4) In this article, experimental data under one working condition are used to validate the proposed method that has certain limitations. Therefore, future work should pay attention to different working condition validation and adaptability improvement. Moreover, the network search space and search strategy should also be further developed.

REFERENCES

[1] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, "Applications of machine learning to machine fault diagnosis: A review and roadmap," *Mech. Syst. Signal Process.*, vol. 138, Apr. 2020, Art. no. 106587.

[2] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020, *arXiv:2003.05689*.

[3] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019, *arXiv:1905.01392*.

[4] C. Liu et al., "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[5] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.

[6] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. 35th Uncertainty Artif. Intell. Conf.*, 2020, pp. 367–377.

[7] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 7588–7598.

[8] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[9] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Aug. 6, 2021, doi: 10.1109/TNNLS.2021.3100554.

[10] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2019.

[11] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[12] T. Elsken, J. Hendrik Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," 2018, *arXiv:1804.09081*.

[13] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 4780–4789.

[14] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[15] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2015.

[17] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, pp. 185–212, Apr. 2009.

[18] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[19] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.

[20] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. Porikli, "Dynamical hyperparameter optimization via deep reinforcement learning in tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1515–1529, May 2021.

[21] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[22] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[23] A. Zela, J. Siems, and F. Hutter, "NAS-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search," 2020, *arXiv:2001.10422*.

[24] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12239–12248.

[25] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[26] X. Li et al., "Improving one-shot NAS by suppressing the posterior fading," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 13836–13845.

[27] V. Thost and J. Chen, "Directed acyclic graph neural networks," 2021, *arXiv:2101.07965*.

[28] J. Li, X. Li, and D. He, "A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction," *IEEE Access*, vol. 7, pp. 75464–75475, 2019.

[29] L. D'Arcy, P. Corcoran, and A. Preece, "Deep Q-learning for directed acyclic graph generation," 2019, *arXiv:1906.02280*.

[30] Y. Ci, C. Lin, M. Sun, B. Chen, H. Zhang, and W. Ouyang, "Evolving search space for neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6659–6669.

[31] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Densely connected search space for more flexible neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10628–10637.

[32] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2020, pp. 660–676.

[33] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20309–20319.

[34] V. T. Rhyne, "Serial binary-to-decimal and decimal-to-binary conversion," *IEEE Trans. Comput.*, vol. C-19, no. 9, pp. 808–812, Sep. 1970.

[35] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015, *arXiv:1511.08458*.

[36] A. Fred Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.

[37] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.

[38] W. J. Maddox, M. Balandat, A. G. Wilson, and E. Bakshy, "Bayesian optimization with high-dimensional outputs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.
[39] R. Astudillo and P. I. Frazier, "Thinking inside the box: A tutorial on grey-box Bayesian optimization," 2022, *arXiv:2201.00272*.
[40] Y. Pan, X. Zeng, H. Xu, Y. Sun, D. Wang, and J. Wu, "Evaluation of Gaussian process regression kernel functions for improving groundwater prediction," *J. Hydrol.*, vol. 603, Dec. 2021, Art. no. 126960.
[41] H. Wang, B. van Stein, M. Emmerich, and T. Back, "A new acquisition function for Bayesian optimization based on the moment-generating function," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 507–512.
[42] L. Fröhlich, E. Klenske, J. Vinogradska, C. Daniel, and M. Zeilinger, "Noisy-input entropy search for efficient robust Bayesian optimization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2262–2272.

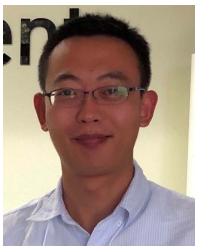
Jialin Li received the B.E. degree from the School of Mechanical Engineering, Shenyang University of Technology, Shenyang, China, in 2015, and the M.S. and Ph.D. degrees from the School of Mechanical Engineering and Automation, Northeastern University, Shenyang, in 2017 and 2021, respectively.

He is currently an Assistant Professor of mechatronics and vehicle engineering with Chongqing Jiaotong University, Chongqing, China. His current research interests include data-driven methods to fault diagnosis, pattern recognition, deep learning, remaining useful life estimation, and their application to critical components of mechanical equipment.

Renxiang Chen received the B.E. and Ph.D. degrees in mechanical engineering from Chongqing University, Chongqing, China, in 2007 and 2012, respectively.

His main research interests include mechanical and electrical equipment security service, life prediction and health management, mechanical signal processing, intelligent manufacturing, machinery condition monitoring, and fault diagnosis.

Xianzhen Huang is currently a Full Professor with the School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China. He has published more than 60 articles. His main research interests include mechanical reliability, system reliability, and mechanical dynamics.



Yongzhi Qu (Member, IEEE) received the bachelor's degree in measurements and control and the master's degree in measurement and testing from the Wuhan University of Technology, Wuhan, China, in 2008 and 2011, respectively, and the Ph.D. degree in industrial engineering and operations research from the University of Illinois at Chicago, Chicago, IL, USA, in 2014.

He is currently an Assistant Professor in mechanical and industrial engineering with the University of Minnesota at Duluth, Duluth, MN, USA. His research interests include machine learning for industrial artificial intelligence, signal processing, systems diagnostics and prognostics, smart sensing, and digital twin. His recent work focuses on data-driven discovery of system dynamics and using machine learning methods to approximate differential equations for in situ dynamic system modeling and in-process quality control for smart manufacturing.