Adaptive Knowledge Distillation-Based Lightweight Intelligent Fault Diagnosis Framework in IoT Edge Computing

Yanzhi Wang, Ziyang Yu, Jinhong Wu, Chu Wang, Qi Zhou, and Jiexiang Hu

***Abstract*—Intelligent fault diagnosis of mechanical equipment is crucial to ensure reliable operation. However, cloud-based fault diagnosis methods often encounter challenges, such as time delays and data loss. Therefore, edge computing-based fault diagnosis has emerged as a promising alternative. However, the limited hardware resources of edge devices in the industrial Internet of Things (IoT) pose significant challenges in striking a balance between diagnostic capabilities and operational efficiency. This article introduces a novel lightweight intelligent fault diagnosis method, which is tailored for IoT edge computing scenarios. Optimal weights are trained on cloud computing and inference is performed on edge computing to ensure timely diagnosis. Based on adaptive knowledge distillation, fault knowledge is transferred from a cloud-based deep neural network model (T-model) to an edge-based lightweight model (S-model). By dynamically adjusting the distillation temperature, the S-model effectively acquires and deeply understands the knowledge representation from the T-model. Additionally, we explore practical considerations and potential challenges in the application of the proposed approach. Verification experiments were conducted on two experimental devices, and the NVIDIA Jetson Xavier NX suite was selected as the edge computing platform. The proposed method exhibited significant enhancements in diagnostic accuracy, demonstrating an average improvement of 10.7% compared to existing methods. In lightweight tests, our method achieved an average 25.5% increase in inference speed compared to current approaches. Furthermore, our method reduced memory usage by 96.58% compared to the T-model, concurrently boosting processing speed by a factor of 8.79.**

***Index Terms*—Deep learning (DL), edge computing, fault diagnosis, knowledge distillation, lightweight neural network.**

I. Introduction

FAULT diagnosis is integral to ensuring the reliable operation of industrial systems and machinery. A prompt and precise fault diagnosis can help prevent potential failures and reduce downtime, ultimately leading to an enhancement in overall system performance [1], [2], [3], [4].

Manuscript received 9 February 2024; revised 15 March 2024; accepted 7 April 2024. Date of publication 10 April 2024; date of current version 26 June 2024. This work was supported in part by the National Key Research and Development Program of China Young Scientists Project under Grant 2022YFC2204700. *(Corresponding author: Jiexiang Hu.)*

Yanzhi Wang, Jinhong Wu, Chu Wang, Qi Zhou, and Jiexiang Hu are with the the School of Aerospace Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: wangyz2050@gmail.com; wujh2019@gmail.com; faralleywc@gmail.com; qizhou@hust.edu.cn; hu_jiexiang@hust.edu.cn).

Ziyang Yu is with the School of Automation, China University of Geosciences, Wuhan 430074, China (e-mail: yzy15171688019@163.com).

Digital Object Identifier 10.1109/JIOT.2024.3387328

Over the years, various fault diagnosis techniques have been explored, ranging from traditional rule-based approaches to data-driven machine learning methods [5], [6]. Deep learning (DL)-based fault diagnosis has shown promising results because of its capacity to autonomously extract complex features and patterns from raw data [7], [8]. Yuan et al. [9] presented a generic data-driven, end-to-end manufacturing system monitoring framework. The framework utilizes DL techniques to detect and even predict failures and wear and tear. Xu et al. [10] presented a novel zero-shot fault semantics learning approach, which is trained on a single fault category and aims to identify previously unknown compound faults. Li et al. [11] proposed an intelligible wavelet packet kernel-constrained convolutional network to address the issues of inadequate interpretability and susceptibility to noise observed in DL-based techniques for fault diagnosis. Chen et al. [12] introduced a novel relational conduction graph network to address the challenge of scarce fault samples in real-world scenarios. The network was trained on a laboratory-generated data set and effectively identified fault types in real-world environments.

In terms of deployment and implementation, the cloud-based method has become a common approach to monitoring the health of industrial systems [13]. The method involves collecting real-time operational data from devices via sensors and uploading it to the cloud for inference via the industrial Internet of Things (IoT). However, in the case of mechanical equipment, which often operates in harsh environments, this method is prone to data loss and inference latency. These challenges significantly impact the real-time capability of fault diagnosis and hinder its effectiveness [14].

Recently, there has been academic interest in edge computing. This novel computing paradigm performs computation at the IoT edge instead of centralized cloud servers [15]. This strategy reduces system response time, minimizes transmission bandwidth usage, reduces storage demand, and computes resources within the cloud infrastructure [16], [17], [18], [19], [20]. Shi et al. [21] proposed a comprehensive definition of edge computing and researched various application scenarios, identifying several challenges and prospects within the realm of edge computing. Liu et al. [22] proposed an IoT network dynamic clustering solution using the emerging deep reinforcement learning for efficient data processing in IoT edge systems. Janjua et al. [23] introduced an abnormal monitoring system, capable of employing unsupervised machine

learning methods directly on gateways situated at the IoT edge. This system is used for data analysis and fault diagnosis in proximity to target devices, which allows for real-time feedback on results. Jing et al. [24] introduced a cloud-edge collaboration framework based on DL, resulting in more accurate predictions of remaining useful life and a significant reduction in model training time. Zhang et al. [25] utilized a stacked autoencoder deep neural network to construct a fault diagnosis model during the training phase. Subsequently, a transfer learning strategy was employed to deploy this model at the edge, facilitating expedited fault localization. Ren et al. [26] introduced a collaborative adaptation approach between cloud and edge computing for fault diagnosis tailored to equipment specific to various manufacturing scenes within cloud manufacturing systems. Qizhao et al. [27] introduced an efficient asynchronous federated learning technique. It mitigates resource demands at the edge, minimizes communication overhead, and enhances training speed within heterogeneous edge environments. The combination of edge computing and DL techniques shows potential for addressing the challenges of real-time fault diagnosis scenarios, positioning it as an increasingly important area of research in edge computing applications [28], [29], [30].

Currently, intelligent fault diagnosis in the context of edge computing faces a series of challenges, mainly related to the following issues [13], [31].

1) These models typically have many parameters, resulting in excessively large model sizes that are not appropriate for operation on resource-limited edge devices.
2) The high computational complexity of these models requires significant computational resources on edge devices, potentially leading to real-time responsiveness and efficiency issues.
3) Edge devices have limited available memory, which affects model design. This is not conducive to balancing the timeliness and accuracy of fault diagnosis.

To address these challenges, researchers have proposed various model-lightweight methods in recent years. These approaches are designed to diminish the computation complexity and memory demands of models while upholding accuracy, rendering them well-suited for fault diagnosis in IoT edge computing scenarios. These lightweight methods include model pruning, parameter quantization, low-rank decomposition, depth-wise separable convolutions, and more. Model pruning reduces the size and computational complexity of the model by removing redundant parameters and connections [32]. Parameter quantization represents model parameters in lower precision formats, reducing memory requirements [33]. Low-rank decomposition further reduces the number of model parameters by decomposing weight matrices into multiple low-rank matrices [34]. Depth-wise separable convolutions split the traditional convolution operation into depth-wise convolutions and pointwise convolutions, reducing the computation and the parameter count [35].

By selecting appropriate lightweight methods, we can effectively adapt complex DL models to resource-constrained edge devices, enabling efficient and accurate fault diagnosis. However, when applied to different tasks, the aforementioned lightweight techniques often require manual parameter adjustments or the design of complex algorithms. This increases the workload for engineering applications. Knowledge distillation is a model compression and transfer learning technique that effectively reduces the workload of algorithm design [36], [37]. It involves transferring knowledge from a pretrained T-model to a new S-model to achieve lightweight and efficiency. Specifically, the T-model is typically a larger and more accurate DL model, while the S-model is a lightweight model suitable for operation on edge devices. Through knowledge distillation, the S-model can learn the complex knowledge representation from the T-model, thereby reducing the number of model parameters and computational complexity, while retaining high diagnostic accuracy [38]. In this article, an adaptive knowledge distillation-based lightweight intelligent fault diagnosis framework is proposed for IoT edge computing scenarios. The following are the main contributions of this article.

1) The fault diagnosis process is designed based on the concept of edge computing. The optimal weights with high diagnostic accuracy are trained and distributed based on the hardware resources on the cloud computing side. The inference is performed at the edge computing side to ensure the timeliness of the diagnosis.
2) Knowledge distillation is employed to transfer the fault diagnosis expertise from the T-model to the S-model, realizing knowledge transfer from the cloud-based deep neural network model to the lightweight edge model.
3) The distillation temperature is adaptively adjusted to effectively guide S-models in learning the knowledge of T-models. This enabled the S-model to better understand and generalize complex knowledge representations, thereby achieving superior fault diagnosis results.
4) Comparative tests and ablation experiments ablation experiments were carried out on two experimental platforms. The results show that the method proposed in this article can transfer knowledge effectively. The S-model exhibited better diagnostic capability, a smaller memory footprint, and higher speed.

This article is structured as follows. Section I presents the application background of the proposed method. Section II introduces the technical basis of the proposed method. Details of the proposed method are presented in Section III. Experiments and evaluations are demonstrated in Section IV. Section V presents practical considerations and potential challenges in the application of the proposed approach. Finally, Section VI presents the conclusions and future work.

II. Preliminary

In this article, we introduce a novel knowledge distillation methodology aiming to optimize model size and enhance computational efficiency. This is achieved by transferring knowledge from a large and intricate neural network, referred to as the teacher model (T-model), to a smaller and simpler neural network known as the student model (S-model) [38], [39]. The process contains three key elements.

A. Distillation Temperature

The distillation temperature is used to weigh the output of the T-model to control the entropy size of the probability distribution of the T-model output. N classes of samples are fed into the model. With n representing the number of classes and b denoting the index of classes. The probability of the b-category output of the model can be expressed as

$$Q^b = \frac{\exp(z^b/T)}{\sum_{b=1}^{n} \exp(z^b/T)} \tag{1}$$

where Q^b is the soft label for category b, T is the temperature in the knowledge distillation, and z^b indicates the output of the fully connected layer of the neural network via category b. The optimal balance between classification accuracy and confidence in the S-model can be achieved by adjusting the size of T. The output of the T-model has a smoother probability distribution when T is large, and the trained S-model is more focused on the confidence level of the classification. When T is small, the output of the T-model is sharper, the probability distribution is concentrated, and the trained S-model is more focused on the accuracy of the classification. When T is equal to 1, indicating the standard Softmax function.

B. Soft Label

In knowledge distillation, soft label (Q) is used as a label to guide the training of the S-model. Specifically, the soft label of the T-model for a sample is a vector of probability distributions, where each dimension represents the probability that the sample belongs to a different label. Based on (1), the soft label is obtained by dividing the probability distribution based on the T-model predictions by T and performing the Softmax exponentiation. Compared with traditional hard labels (one-hot labels), soft labels can provide more supervised information and help the S-model to learn better from the T-model. In addition, soft labels can solve the problem of the discrete output of the T-model, it makes the output of the S-model more continuous.

C. Loss Functions

The goal of knowledge distillation is to make the S-model learn the knowledge of the T-model. Therefore, loss functions are required to guide the learning of the S-model.

Hard loss is used to ensure the similarity of the S-model output to the sample label. For the sample (x_i, l_i), the probability of the student network predicting class b by the Softmax function is $p(b, x_i)$, as $T = 1$. The distribution $q(l_i, x_i)$ of hard labels l_i can be expressed as

$$q(l_i, x_i) = \begin{cases} 1, & b = l_i \\ 0, & b \neq l_i. \end{cases} \tag{2}$$

The standard cross-entropy loss function is used as the hard loss in this process, which can be written as

$$\begin{aligned} \text{Loss}_{\text{hard}} &= -\sum_{b=1}^{n} q(l_i, x_i) \log\, p(b, x_i) \\ &= -\log\, p(b = l_i, x_i). \end{aligned} \tag{3}$$

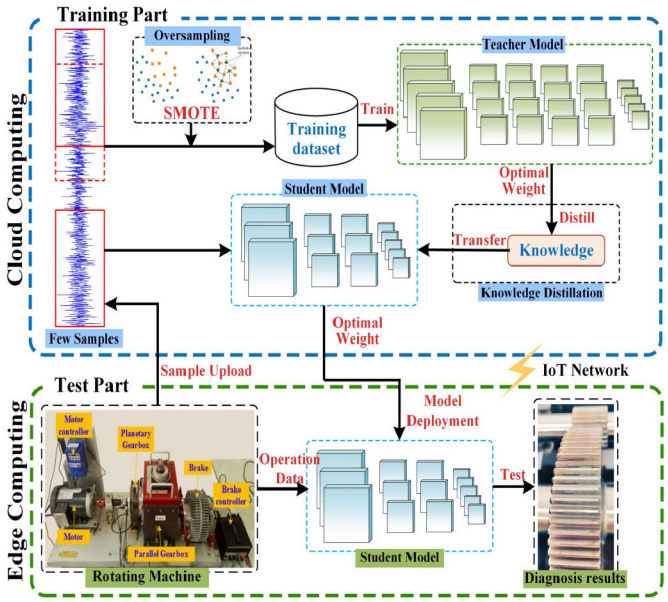

Fig. 1. Process of intelligent fault diagnosis method based on adaptive knowledge distillation.

Soft loss measures how close the predicted outcome of the S-model is to the soft label of the T-model. With setting $T = t$ in (1), put the sample (x_i, l_i) into the teacher network, and get the soft label $Q^T(b, x_i, t)$ of the sample. The output soft label $Q^S(b, x_i, t)$ of the student network can be obtained in the same way. The Kullback–Leibler divergence loss function is used as the soft loss in this process, which can be written as

$$\text{Loss}_{\text{soft}} = \sum_{b=1}^{n} Q^T(b, x_i, t) \log \frac{Q^T(b, x_i, t)}{Q^S(b, x_i, t)}. \tag{4}$$

Due to the random initialization of parameters at the onset and the absence of hard labels, the model often struggles to converge. Both hard and soft labels are used to supervise knowledge distillation with the following loss function:

$$\text{Loss} = b * \text{Loss}_{\text{hard}} + (1 - b) * \text{Loss}_{\text{soft}} \tag{5}$$

where β is a hyperparameter in knowledge distillation, which is used to balance the knowledge obtained from the teacher network through knowledge distillation (soft label) and the tagged knowledge of the sample (hard label). When the value of β is close to 0, the model emphasizes more on accurate prediction of true labels and misses the learning of T-model information. On the contrary, when the β value is close to 1, the model emphasizes more on the accuracy of the T-model prediction and pays less attention to the accuracy of the true label. In applications, β should be set according to actual task conditions.

III. Proposed Method

A. Framework of the Proposed Approach

To address the fault diagnosis of rotating machinery in IoT edge computing scenarios, this study introduces an adaptive knowledge distillation-based intelligent fault diagnosis method. The approach comprises two parts: 1) a training part located on the cloud and 2) a testing part on the edge. Fig. 1

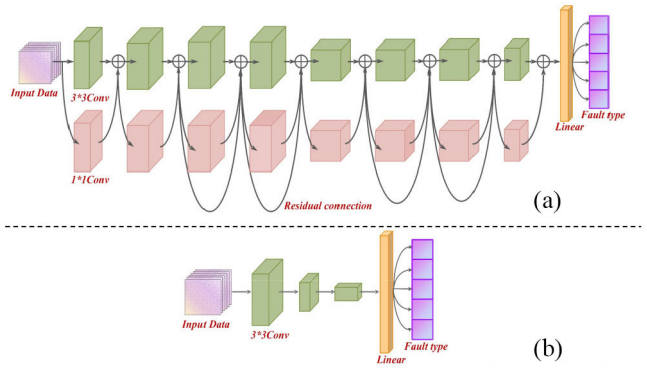

Fig. 2. Network structure of the (a) T-model and the (b) S-model.

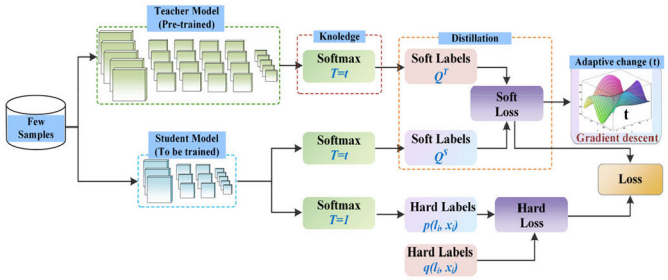

Fig. 3. Process of the proposed knowledge distillation method.

illustrates the primary flowchart of the method introduced in this study.

On the cloud, first, a small number of samples are received from the edge through the IoT network. Next, the samples are oversampled to construct the training data set by the synthetic minority over-sampling technique (SMOTE). The T-model is trained iteratively based on the training data set. Knowledge is supplied by the T-model that has been trained, and the simple S-model is trained by adaptive distillation to acquire the knowledge in the T-model. This procedure achieves the transfer of knowledge from the complex T-model to the simple S-model. Finally, the optimal weights of the S-model are sent down to the edge through the IoT network after training.

On the edge, the S-model is deployed in the edge computing device, which has imported the best weights of the model distributed from the cloud. Sampling real-time data from rotating machinery and importing the samples into the S-model for testing, achieving fast and accurate fault diagnosis results on the edge.

In conclusion, the most important aspect of the proposed method is the transfer of knowledge from the complex T-model to the simple S-model which is achieved by adaptive knowledge distillation.

B. Architecture of Network

To achieve the lightweight deployment of the model in IoT edge computing scenarios, we designed a neural network with a relatively complex structure as the T-model and a neural network with a simpler structure as the S-model. The network structures are shown in Fig. 2.

The network structure of the T-model consists of eight basic modules connected in series, as shown in Fig. 2(a). Each basic module consists of a 3×3 convolution, a 1x1 convolution, and a BN layer. Specifically, for modules in which the number of channels and feature maps in the input domain and output domain remain unchanged, a residual connection is added. The structure of the S-model is very simple, as shown in Fig. 2(b), consisting of three 3×3 convolution layers and BN layers in series. When designing the structures of the teacher and S-models, the following issues were primarily taken into consideration.

1) *Better Feature Extraction Capability for the T-model:* The multibranch architecture model can be considered as an implicit collection of multiple shallow models, which can improve the feature extraction ability of the network. At the same time, the residual connection can prevent the disappearance of the gradient and is more conducive to training.
2) *The Lightweight of the S-model:* Compared with the multibranch structure of the T-model, the single-branch structure of the S-model can significantly reduce memory costs. At the same time, the three 3×3 convolution layers connected in series allow the model to achieve faster inference speed.

C. Training of T-Model

Training fault diagnosis models in the context of edge computing are often challenged by insufficient or imbalanced training samples. The proposed method oversamples the training set data to augment the training set based on SMOTE [40]. This approach enhances the ability of the model to learn from minority samples during the training of the T-model, thereby improving the diagnostic capability of the T-model. The specific steps are as follows.

1) For each sample in the data set $D_{\text{sample}} = \{x_1, x_2, \ldots . x_i\}$, select one sample x_c from c nearest neighbors.
2) Set a random number γ, $\gamma \in (0, 1)$.
3) Generate a new synthetic sample using the following formula:

$$x_{\text{new}} = x_i + \gamma * (x_c - x_i). \tag{6}$$

4) Repeat the above steps $N - i$ times to generate $N - i$ new samples.

By employing this oversampling method, an augmented training data set is obtained: $D_{\text{train}} = \{x_1, x_2, \ldots, x_i, \ldots, x_N\}$.

Training the T-model using the augmented data set D_{train}. During training, the cross-entropy loss function is used to quantify the difference between the real distribution and the output distribution of the model

$$\text{Loss}_{\text{teacher}} = -\sum_{i=1}^{N} (L_{\text{train}} * \log L_{\text{teacher}}) \tag{7}$$

where $L_{\text{train}} = \{l^1_{\text{train}}, l^2_{\text{train}}, \ldots, l^N_{\text{train}}\}$ is the true distribution of the data set D_{train}, and $L_{\text{teacher}} = \{l^1_{\text{teacher}}, l^2_{\text{teacher}}, \ldots, l^N_{\text{teacher}}\}$ is the output domain of the T-model.

D. Adaptive Knowledge Distillation Method

Fig. 3 depicts the proposed adaptive knowledge distillation framework. The proposed method modifies the output layer of the trained T-model by adding a parameter $T = t$. This model processes z training samples in a training batch and produces the T-model's soft label set $L_{Q^T} = \{Q_1^T, Q_2^T, \ldots, Q_z^T\}$ for that batch. Similarly, set $T =$ t in the S-model to obtain the S-model's soft label set $L_{Q^S} = \{Q_1^S, Q_2^S, \ldots, Q_z^S\}$ for the same batch of training samples.

Based on (4), the soft loss between L_{Q^T} and L_{Q^S} are calculated. Set T in the S-model to 1 to obtain the student hard label set $L_p = \{p_1, p_2, \ldots, p_z\}$ for the same batch of training samples. Calculate the hard loss by using the (3) between L_p and the label set $L_q = \{q_1, q_2, \ldots, q_z\}$ of the batch of samples. Finally, the loss is obtained from (5). The gradients of the loss concerning the parameters of the S-model are then calculated through backpropagation. The Adam optimizer adjusts the parameters of the S-model using the computed gradients.

It is worth noting that the proposed method adopts a dynamic adjustment approach for the T during the distillation process. This process can change the degree of softness of the prediction distribution between the T-model and the S-model. This allows the S-model to better learn from the knowledge of the T-model. Specifically, the process involves the following steps.

1) *Initialisation of the Distillation Temperature:* At the beginning of the training, an initial temperature of T0 is set, which can be set based on the empirical values of the pretrained.
2) *Calculate the Softened Predictive Distributions:* In each iteration, the predictions of the T-model and the S-model on the training data are obtained first. The predictions of the T-model and S-model are softened using the current temperature T. The predictions are converted into probability distributions by applying the Softmax function.
3) *Calculate the* $\text{Loss}_{\text{soft}}$*:* Using the softened probability distributions, compute the KL divergence loss between the prediction distributions of the T-model and the S-model.
4) *Calculate the Gradient of T:* By calculating the gradient of the concerning the temperature T, obtain the loss gradient concerning T

$$\text{Gradient\_}T = \text{Gradient}(\text{Loss}_{\text{soft}}, T). \tag{8}$$

5) *Update Temperature T:* Dynamically adjust the value of T based on the direction and magnitude of the loss gradient. Specifically, increase or decrease the temperature value so that the average of the loss gradient is close to zero. The purpose of this step is to reduce the Loss further. This avoids instability problems caused by excessively high or low temperatures during the training process, and enables the S-model to learn more knowledge of the T-model

$$T = T - \text{KD\_lr} * \text{mean}(\text{Gradient\_}T) \tag{9}$$

where mean() is the average algorithm, and KD_lr is the learning rate (LR) used to update the temperature during distillation with an initial value of 0.1.

6) *Update Temperature LR:* The following formula adjusts the temperature LR during distillation:

$$\text{KD\_lr} = \begin{cases} \text{KD\_}lr_{\min} + \frac{1}{2}(\text{KD\_}lr_{\max} - \text{KD\_}lr_{\min})\left[1 + \cos\left(\frac{E_C - E_W}{E_M - E_W}\pi\right)\right], & E_C \geq E_W \\ \text{KD\_}lr_{\max}\frac{E_C}{E_W}, & E_C < E_W \end{cases} \tag{10}$$

where E_C, E_W, and E_M are the current epoch, warm-up epoch, and maximum epoch, respectively. $\text{KD\_}lr_{\max}$ and $\text{KD\_}lr_{\min}$ are the maximum LR and minimum LR in the adjustment process, respectively.

Repeat steps 2–6 during the distillation process until the specified number of iterations is reached. Through the aforementioned process, the temperature parameter T can be dynamically adjusted based on the current model state and loss conditions, enabling a more effective and stable knowledge distillation process. In addition, it allows better adaptation to different data sets and model structures, leading to improved performance and generalization capability of the knowledge distillation method.

IV. Case Studies

A. Experimental Background

This article employed a computer system and a Jetson Xavier NX kit for conducting all case studies. The computer system consists of an Intel Core i7-11700K@ 3.60 GHz processor, an NVIDIA GeForce RTX 3070 GPU, and 32.0 GB of RAM. On the other hand, the Jetson Xavier NX kit includes an NVIDIA Carmel ARM CPU, an NVIDIA Volta GPU with 48 Tensor Cores, and 8.0 GB of RAM. These devices were used as the cloud device and the edge device, respectively. The cloud device, operating on the Windows platform, hosted the models' dependency environment. In contrast, the edge device utilized Docker, an open-source application container engine, for deployment. The models were developed using Python 3.9 and PyTorch 1.11.0.

For the experimental procedure, the data set was partitioned into a training data set, a validation data set, and a testing data set. The training and verification data sets were stored in the cloud device, while the test data set was stored in the edge device. The models were trained on the cloud device, enabling the acquisition of optimal weights. These weights were then transferred to the edge device via file streams. On the edge device, the models loaded these weights and performed tests to obtain results.

The proposed method is compared with the typical methods using rotating machinery failure simulation (RMFS), which mainly involves the diagnostic accuracy, feature extraction capability, complexity metrics, and the inference time of the model. This evaluation aims to validate its efficiency in fault diagnosis and lightweight performance in IoT edge computing scenarios. Furthermore, to validate the effectiveness of the proposed adaptive knowledge distillation method, a series

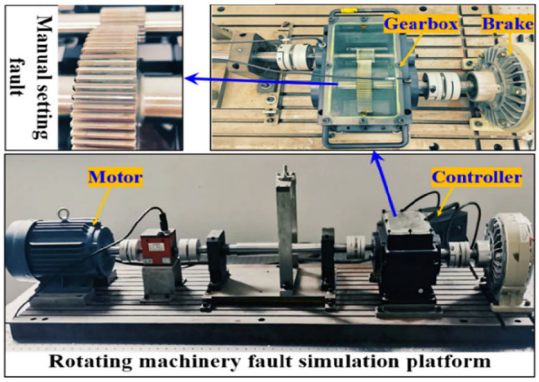

Fig. 4. Experimental equipment of RMFS.

TABLE I
FAULT SETTING METHOD OF RMFS

Fault label	Fault type	Fault processing method
0	Normal	-
1	Broken	Cut off 1/2 of a single whole tooth
2	Miss	A tooth completely missing
3	Root	A crack at the gear root Crack specification: width 0.2 mm, depth 1mm
4	Pitting	Make four pitting corrosion by an electric spark Specification of single place: five times of electric spark

of ablative experiments were performed using the drivetrain dynamics simulator (DDS).

B. CASE 01

1) Experimental Method: The experimental for case 01, focusing on the gearbox, was obtained using a custom-designed RMFS platform from Huazhong University of Science and Technology, as shown in Fig. 4. This comprehensive rotating machinery fault simulator includes a motor, controller, bearing, gearbox, and brake. It allows the introduction of multiple fault scenarios for bearings and gears, along with adjustments to operating conditions by varying speed and load. For this study, we selected the gearbox from this platform. Specifically, the gearbox utilized is a ZDY80 parallel shaft gearbox, which is capable of simulating five different operating conditions: 1) normal; 2) broken; 3) miss; 4) root; and 5) pitting. The fault location is in the large gear, as shown in Table I.

Acceleration sensors were mounted above the vertical axis of both the high-speed and low-speed end shafts of the parallel gearbox to record acceleration signals in the x, y, and z directions at these two positions. A sliding window approach was used for sampling, with a fixed window length of 1024, and a window overlap of 0 was employed. The samples of the five fault conditions were labeled as 0–4 during the experiments. To fully verify the advantage of the proposed method, three sample data sets of experiments are conducted with 5, 10, and 20 randomly selected samples for each fault category, denoted as: $D^{I}_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-5}\}$, $D^{II}_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-10}\}$, and $D^{III}_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-20}\}$, respectively. Based on the D^{I}_{sample}, D^{II}_{sample}, and D^{III}_{sample}, the training data sets: $D^{I}_{\text{train}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-100}\}$, $D^{II}_{\text{train}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-100}\}$, and $D^{III}_{\text{train}} = \{x_{0-1}, x_{0-2}, \ldots, x_{4-100}\}$are constructed by SMOTE oversampling, respectively. During testing, 100 samples were randomly selected from each fault category to form the testing data set $D_{\text{test}} = \{x_1, x_2, \ldots, x_{100}\}$. To further validate the diagnostic performance of the proposed method and its applicability in IoT edge computing scenarios, the following models were selected for comparative experiments.

1) *Mobilenet:* Mobilenet is a lightweight DL model specifically designed for efficient deployment on mobile and embedded devices. It utilizes depth-wise separable convolutions to reduce computation while maintaining reasonable accuracy, making it ideal for resource-constrained environments [41].
2) *Mnasnet:* Mnasnet is a mobile-friendly architecture optimized for edge devices with limited computational resources. It employs a differentiable neural architecture search to automatically design the network, striking a balance between model accuracy and computational efficiency [42].
3) *Xception:* Xception is a deep convolutional neural network architecture. It employs depth-wise separable convolutions in an extreme version, achieving superior accuracy in classification tasks [35].
4) *Resnet:* Resnet is a seminal DL model architecture that introduced residual connections. This design innovation effectively addresses the vanishing gradient problem in very deep neural networks, enabling the creation of even deeper models for stronger feature extraction capabilities [43].
5) *T-Model:* The T-model described in the previous section. Which is iteratively trained to obtain optimal weights based on D_{train}.
6) *Proposed Method:* Based on the optimal weights of the T-model and the D_{sample}, the S-model undergoes adaptive knowledge distillation training, resulting in the optimal weights for the S-model. Subsequently, the S-model is evaluated on the D_{test}.

MobileNet and MnasNet have been popular lightweight network models in recent years. Comparing the proposed method with these two networks to evaluate its lightweight effect. On the other hand, Xception and Resnet are efficient convolutional neural networks in recent years. Comparing the proposed method with these two approaches to evaluate its diagnostic capacity. Furthermore, the comparison of the proposed method with the T-model to measure the effectiveness of knowledge transfer in the proposed approach.

2) Results and Discussion: The input samples for each model are reshaped into a matrix of size [6], [13]. For training, the batch size is set to 32, taking into account convergence and memory considerations. When training the MobileNet, MnasNet, Xception, Resnet, and T-model, the epoch is set to 50, and when training the S-model by the knowledge

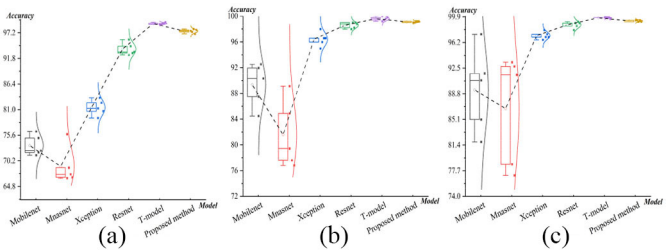

Fig. 5. Test accuracy in case one. (a) Test accuracy of group I. (b) Test accuracy of group II. (c) Test accuracy of group III.

TABLE II
TEST ACCURACY OF DIFFERENT MODELS IN CASE ONE (%)

Model	I	II	III
Mobilenet	73.43	89.36	89.35
Mnasnet	69.03	81.57	86.65
Xception	81.41	96.41	97.18
Resnet	93.75	98.59	98.75
T-model	99.15	99.59	99.76
Proposed method	**97.54**	**99.11**	**99.32**

distillation, this parameter is set to 500. This setting method ensures that each model can converge during the training process. Cosine annealing LR is used to adjust the LR during training. To ensure the robustness and reliability of the results, *I*, *II*, and *III* groups of experiments were independently repeated five times and the average values were used for the subsequent analysis.

The experimental test accuracy is shown in Fig. 5, where the line connects the averages of five repeated experiments and the curves represent the normal distribution of test accuracy for each model. Table II shows the averages of repeated experiments in three sets of experiments. The three sets of experiments shown in Fig. 5 have similar trends. Only the T-model and the proposed method maintain an accuracy of over 97% across different data scenarios. Furthermore, their normal distribution curves are concentrated, indicating the excellent diagnostic ability of the T-model. The average accuracy of Mobilenet and Mnasnet is significantly lower than the other models, and their normal distribution curves are more scattered. This suggests that traditional lightweight models are less likely to achieve excellent fault diagnosis performance in IoT edge computing scenarios with limited real-world samples. As for Xception and Resnet, the average diagnostic accuracy in group *I* experiments is 81.41% and 93.75%, respectively, which is significantly lower than that of the T-model and the proposed method. This indicates a significant decrease in the diagnostic capability of classical models when facing limited real samples in IoT edge computing scenarios.

To further analyze the feature extraction performance of different models under different sample sizes and to verify the diagnostic performance of the proposed method, we use t-distributed stochastic neighbor embedding (t-SNE) to visualize the impact of different techniques. The features extracted by the model are represented by the output in front of the fully connected layer. Comparing the feature extraction results of the different models shown in Fig. 6(a), we observe

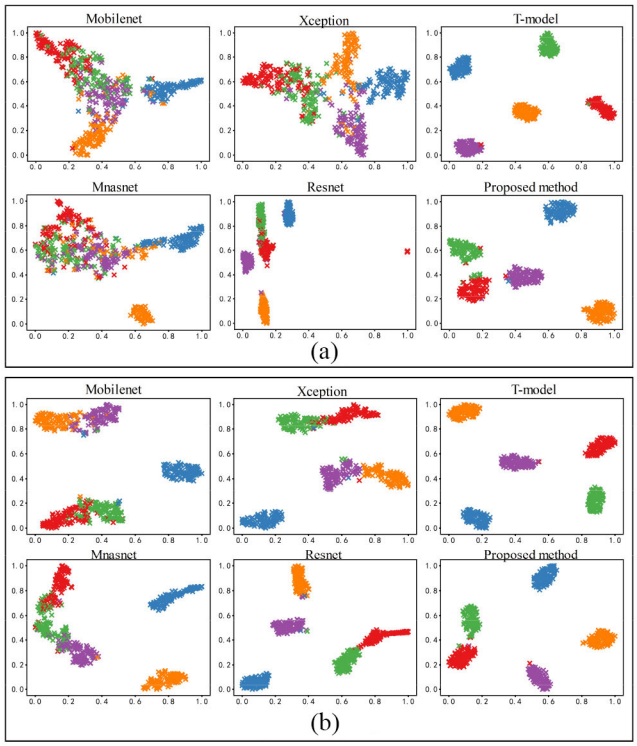

Fig. 6. t-SNE of different models in case one. (a) Group I. (b) Group III.

that for Mobilenet, Mnasnet, and Xception, samples from different fault classes overlap and there is no clear boundary between fault types. In the case of Resnet, although the data distribution within each fault class is more compact, some individual fault types show indistinct features and partial samples from different fault classes appear to be stacked. For the T-model and the proposed method, clear boundaries are maintained between each class, with only a few instances of slight feature misalignment. Fig. 6(b) shows the t-SNE results of group *III*, we find that the feature stacking of samples from different classes has improved for Mobilenet, Mnasnet, and Xception. However, some individual faults still lack distinct boundaries and exhibit adhesion. In contrast, Resnet maintains clear boundaries between each class, with a few samples showing feature misalignment. Clear class boundaries are observed for the T-model and the proposed method. The above analysis indicates that under both initial sample conditions, the T-model and the proposed method demonstrate excellent feature extraction capabilities, surpassing the performance of the corresponding control models.

Furthermore, based on the detailed diagnostic information in the test results, an analysis of the diagnostic performance of the models was conducted. Specifically, the model weights corresponding to the median test accuracy of each model in the group *II* were selected, tested in D_{test}, and the classification results were visualized as a percentage stacked bar chart, as shown in Fig. 7. The graph shows the number of correctly classified samples and the total number of misclassifications for each category. In the graph, the number of misclassifications for T-model and the proposed method are 1 and 3, respectively. Resnet and Xception maintain misdiagnosis

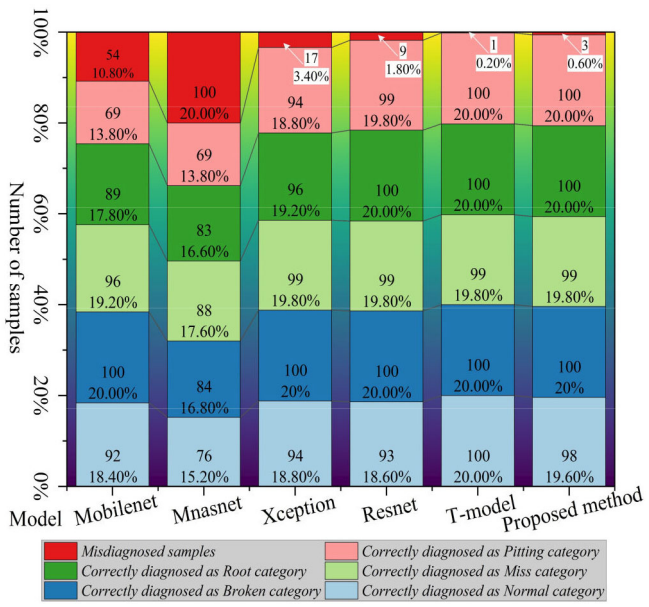

Fig. 7. Number of correctly diagnosed samples in each state and number of misdiagnosed samples in group II.

counts at a relatively low level but are noticeably higher than the proposed method and T-model. Mobilenet has a misdiagnosis count of 54, with an error rate reaching 10.8%. Another lightweight model, Mnasnet, has a misdiagnosis count of 100, which exceeds the number of correctly diagnosed samples for any single category. The results indicate that in the context of this work, the T-model can effectively learn fault diagnosis knowledge from operational data. The proposed adaptive knowledge distillation achieves successful knowledge transfer, enabling accurate fault diagnosis on edge devices.

In the previous sections, the proposed method and the comparative approaches have been evaluated in terms of classification accuracy and feature extraction capability for the specified tasks. Furthermore, it is essential to evaluate the network complexity of the models. These indicators of model complexity help determine the suitability of the models for use in IoT edge computing scenarios. The following parameters of different models were compared in the experiments.

1) *Params:* The total number of parameters to be trained in the model training. It is used to measure the size of the model and to calculate the space complexity.
2) *Memory:* The amount of memory required for model inference.
3) *Floating Point Operations (FLOPs):* The theoretical amount of floating point arithmetics is the amount of computation in the neural network. A high FLOPs in the model indicates that the model has a higher model capacity, allowing it to learn more information and capture more complex features.
4) *Inference Time:* Inference time of a single sample in the edge-end device.

Table III shows the parameters of each model, including params, memory, flops, and the average inference time per single sample. The statistics for the params, memory, and flops

TABLE III
TEST RESULTS OF THE COMPLEXITY OF EACH MODEL

Methods	Params	Memory (MB)	Flops (MFlops)	Inference time (ms)
Mobilenet	610,213	1.05	7.31	4.17
Mnasnet	2,048,781	0.96	5.11	9.04
Xception	20,818,061	2.6	88.94	23.13
Resnet	21,300,869	1.47	198.94	28.68
T-model	9,357,573	33	3540	20.92
Proposed method	**1,481,797**	**1.13**	**193.78**	**2.38**

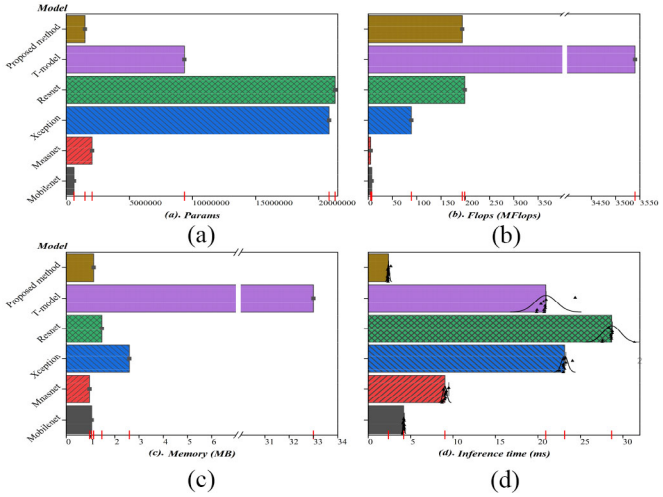

Fig. 8. Test results of the complexity of each model. (a) Params. (b) Flops (MFlops). (c) Memory (MB). (d) Inference time (ms).

parameters were obtained using torchstat, a lightweight neural network analyzer based on PyTorch. Inference times were measured on the NVIDIA Jetson Xavier NX platform using the model weight from group *III*. To ensure the reliability of the results, the measurement experiment was repeated ten times. To reduce the impact of code writing style on inference time, we monitored the model's computation time on the GPU using CUDA Event. In addition, the GPU was preheated at the same intensity before each speed measurement. Fig. 8 provides a visual representation of these results.

Compared to the T-model, the S-model of the proposed method has params and memory which are 15.84% and 4.03% of the T-model, respectively. This indicates that the complexity of the S-model of the proposed method is lower, which makes it easier to be trained with less hardware overhead at runtime. The inference time is 11.40% of the T-model, demonstrating that the proposed method achieves faster inference speed on edge computing devices. Notably, the FLOPs of the T-model are 3540, much higher than the classical deep neural network models Xception and Resnet. This indicates its large model capacity, and as a T-model, it can transfer a significant amount of information to the S-model.

Xception and Resnet have over 14 times more parameters than the S-model in the proposed method. This indicates that classical deep neural network models have a significantly higher number of parameters, leading to higher complexity and making them less amenable to training. The memory of

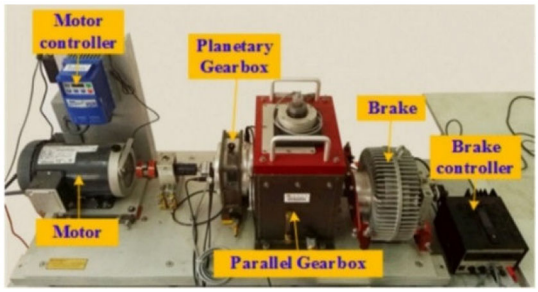

Fig. 9. Experimental equipment of DDS.

the proposed method is also significantly smaller than that of Xception and Resnet, indicating reduced hardware overhead at runtime. The FLOPs of the proposed method are in the same order of magnitude as those of the two classical deep neural networks. This indicates that the S-model of the proposed approach has a significant model capacity and can learn a considerable amount of information. In terms of inference time, the proposed method is 8.31% and 10.31% of that of Xception and Resnet, respectively, showing a clear speed advantage on edge devices.

The S-model of the proposed method has parameters that are in the middle range compared to Mobilenet and Mnasnet. The three models have similar memory requirements, which makes them suitable for edge computing devices. However, the FLOPs of the S-model in the proposed method are 2650.89% and 3792.17% of those of Mobilenet and Mnasnet, respectively, indicating a relatively larger information capacity in its model. Compared with the two classical lightweight models, the inference time of the proposed method is 57.13% and 28.38%, respectively. This shows a significant speed advantage, which enables faster prediction results on edge devices, thus assisting IoT monitoring systems in timely responses.

It can be found through the experiment of CASE1. Compared with Mobilenet and Mnasnet, representatives of lightweight neural networks, in terms of complexity indicators, the proposed method can achieve the same level of lightweight; in terms of inference time, the proposed method is ahead of Mobilenet and Mnasnet; in terms of diagnostic effect, the proposed method is stronger than Mobilenet and Mnasnet. Compared with the two classic neural networks, Xception and Resnet, the proposed method has an obvious lightweight effect and strong diagnostic ability. Compared with the T-model, the proposed method can achieve lightweight while ensuring diagnostic performance. The above results show that the proposed method can achieve relatively ideal fault diagnosis results under IoT edge computing.

C. CASE 02

1) Experimental Method: The DDS at Southeast University, China was used to acquire the experimental data for Case 1. Fig. 9 depicts the simulation platform, where the DDS is capable of simulating the health and multiple fault operating conditions of the bearing and gearbox. The operating condition data of the gear and gearbox in this data set were selected for the case study [44].

The acceleration signals along the x, y, and z axes of the experimental platform were acquired by using acceleration sensors attached to the parallel and planetary gearboxes. A sliding window approach was used for sampling, with a fixed window length of 1024, and a window overlap of 0 was employed. In this study, random Gaussian white noise was introduced to the original experimental signal to enhance it. The signal-to-noise ratio (SNR) of the noise signal was set to -2.

The experimental apparatus can simulate nine fault conditions, namely 1)health, 2)gear teeth broken, 3)gear teeth missing, 4)root crack, 5)surface pitting corrosion, 6)bearing ball, 7)bearing comb, 8)bearing inner ring, and 9)bearing outer ring. They are labeled as 0–8 during the experiments. To fully verify the effectiveness of the proposed method, three sets of experiments are conducted with 6, 12, and 20 randomly selected samples for each fault category, denoted as: $D^A_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-6}\}$, $D^B_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-12}\}$, and $D^C_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-20}\}$, respectively. Based on the D^A_{sample}, D^B_{sample}, and D^C_{sample}, training data sets: $D^A_{\text{train}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-100}\}$, $D^B_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-100}\}$, and $D^C_{\text{sample}} = \{x_{0-1}, x_{0-2}, \ldots, x_{8-100}\}$, are constructed by SMOTE oversampling, respectively. During testing, each fault category contains 100 samples to form the testing data set $D_{\text{test}} = \{x_1, x_2, \ldots, x_{100}\}$. In addition, to validate the application effectiveness of the modules of the proposed method, the following models are selected for ablation experiments.

1) *S-Model:* The S-model is iteratively trained using D_{train} to obtain optimal weights, and tested on the D_{test}.
2) *T-Model:* Based on D_{train}, the T-model is iteratively trained to obtain optimal weights and tested on the D_{test}.
3) *T-Model_W/O-SMOTE:* Based on D_{sample}, the T-model is iteratively trained to obtain optimal weights and tested on the D_{test}.
4) *KD_T0:* The temperature T during distillation is set to three fixed values, $T = 5$, $T = 10$, and $T = 20$. Take the five results with the highest accuracy among all test results as the final result.
5) *Proposed Method:* Based on D_{train}, the T-model is trained to obtain optimal weights. Then, knowledge distillation was performed based on the optimal weights of the T-model and the D_{sample}. The temperature adaptive regulation is used for the process of knowledge distillation. Then, the S-model is tested on the D_{test}.

In the above-mentioned methods, the comparison of the proposed method with the S-model allows us to verify the improvement of the model performance brought by the proposed approach. Comparing the proposed method with the T-model allows us to verify the retention of the T-model knowledge during the knowledge distillation process in the proposed approach. By comparing the T-model with T-model_W/O-SMOTE, the improvement of SMOTE on the training effect of the T-model in the proposed method can be verified. Finally, the comparison of the proposed method with the KD_T0 allows us to validate the improvement resulting from the adaptive temperature changes in the proposed approach.

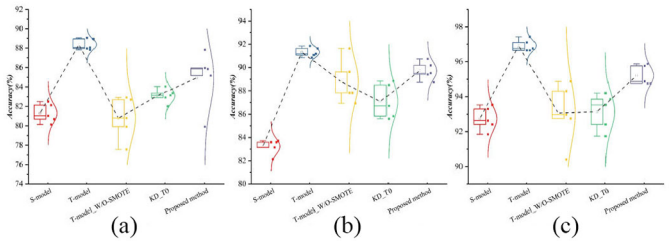

Fig. 10. Test accuracy in case two. (a) Test accuracy of group A. (b) Test accuracy of group B. (c) Test accuracy of group C.

TABLE IV
TEST ACCURACY OF DIFFERENT MODELS IN CASE TWO (%)

Model	*A*	*B*	*C*
S-model	81.3	83.24	92.74
T-model	88.37	91.32	96.92
T-model_W/O-SMOTE	80.78	88.77	93.06
KD_T0	83.3	87.1	93.15
Proposed method	**86.12**	**89.71**	**95.2**

2) Results and Discussion: The input samples for each model are reshaped into a matrix of size [6], [13]. For training, the batch size is set to 32, taking into account convergence and memory considerations. In regular training, Epoch is set to 50, while in knowledge distillation training, this parameter is 500, this setting method ensures that each model can converge during the training process. Cosine annealing LR is used to adjust the LR during training. To ensure the robustness and reliability of the results, A, B, and C groups of experiments were independently repeated ten times and the average values were used for subsequent analysis.

The experimental test accuracy is shown in Fig. 10, where the line connects the averages of five repeated experiments and the curves represent the normal distribution of test accuracy for each model. Table IV shows the averages of repeated experiments in three sets of experiments. The experiments presented in Table IV share common trends. The test results of the S-model were significantly lower than those of the T-model and the Proposed method in the three groups, indicating that its diagnostic ability is limited. The T-model achieves the highest accuracy in each group, demonstrating its excellent diagnostic performance. In the three test groups, the diagnostic accuracy of the T-model is improved by 7.07%, 8.08%, and 4.18%, respectively, compared to the S-model, showing a significant advantage. On the other hand, in the three groups, compared with the T-model_W/O-SMOTE method, the diagnostic accuracy of the T-model increased by 7.59%, 2.55%, and 3.86%, respectively. It shows the gain effect of SMOTE on the training of the T-model. Furthermore, after applying the proposed adaptive knowledge distillation method, the diagnostic results of the proposed method are improved by 4.82%, 6.47%, and 2.46%, respectively. Compared to the S-model in the three respective groups, the results indicate the effective transfer of knowledge from the T-model to the S-model. In addition, the temperature used in the knowledge distillation process is adaptively adjusted in the proposed method. Compared to the traditional knowledge distillation with a fixed temperature (KD_T0), the diagnostic results of the proposed method are further improved by 2.82%, 2.61%, and 2.05% in the three groups, respectively. Furthermore, after applying the proposed adaptive knowledge distillation method, the diagnostic results of the proposed method are improved by 4.82%, 6.47%, and 2.46% compared to the S-model in the three respective groups, also indicating the effectiveness of the knowledge transfer process. In addition, the temperature used in the knowledge distillation process in the proposed method is adaptively adjusted. Compared to the knowledge distillation with a fixed temperature (KD_T0), the diagnostic results of the proposed method are further improved by 2.82%, 2.61%, and 2.05% in the three groups, respectively. This shows that the use of the temperature-adaptive adjustment method can improve the effectiveness of knowledge distillation.

To conduct further analysis of the classification performance of various models under different sample sizes, confusion matrices for test results are constructed using data from groups A and C. Select the classification results corresponding to the diagnostic median of each model in the five repeated experimental results, and draw the confusion matrix as shown in Fig. 11. The vertical axis represents the true labels of the samples, while the horizontal axis denotes the predicted results. The prediction accuracy and the corresponding sample counts are annotated within the matrix cells. In groups A and C, the S-model, shows significant improvements in diagnostic accuracy for individual classes after undergoing adaptive knowledge distillation. In Fig. 11(a), compared with T-model_W/O-SMOTE, the diagnostic accuracy of the T-model for the “Surface” and “Root” categories has increased by 17.17% and 45.57%, respectively. It shows that using SMOTE during the training process has significantly improved the classification effect of the T-model. In Fig. 11(b), the proposed method improves the diagnostic accuracy of the Root class from 66.33% to 85.86%, while the accuracy of the “Comb” class increases from 94.95% to 100%. Notably, all class accuracies of the proposed method reach 85% in group C experiments, whereas some class accuracies remain as low as 69% in the results of knowledge distillation with a fixed temperature (KD_T0). This further illustrates that the proposed method can improve the diagnostic accuracy for certain class samples based on the knowledge transferred from the T-model during training.

To further investigate the impact of the proposed temperature-adaptive variation method on the knowledge distillation process, we plotted the variations of the loss function during the knowledge distillation training for both the proposed method and KT_T0, as shown in Fig. 12. It can be seen that the loss curve of the proposed method shows better convergence in both experimental groups, while the KT_T0 curve shows more pronounced oscillations. These observations explain the superior diagnostic performance of the proposed method compared to KT_T0. Due to the adaptive adjustment of the distillation temperature, the training process can more effectively guide the S-model to learn from the knowledge of the T-model, enabling the S-model to better understand and generalize complex knowledge representations, thus achieving

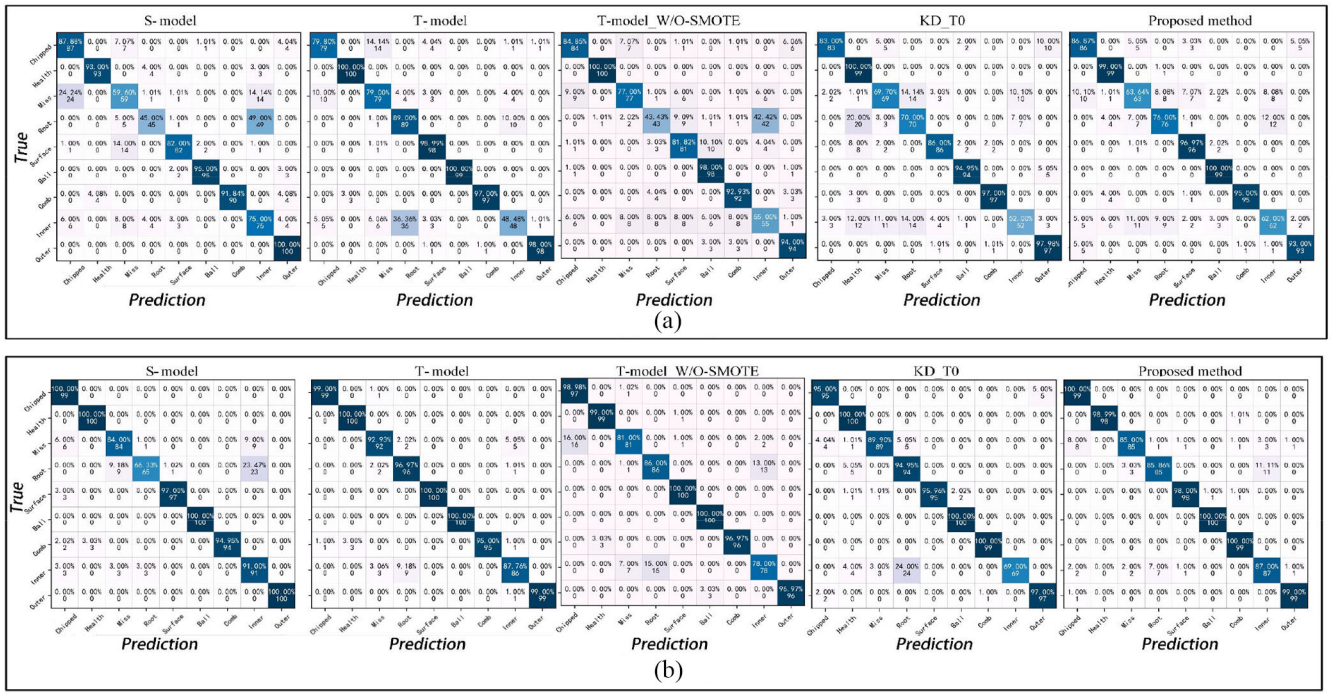

Fig. 11. Confusion matrix of the annealing experiment in case two. (a) Group A. (b) Group C.

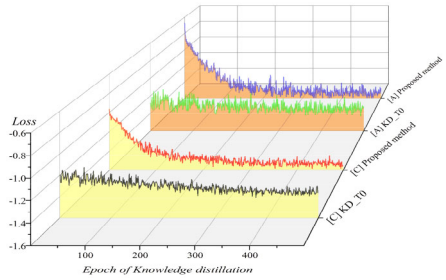

Fig. 12. Loss variation of the proposed method and KD_T0 during the knowledge distillation process.

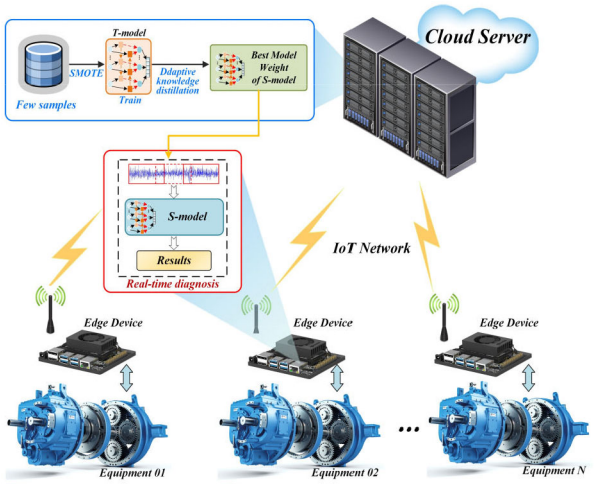

Fig. 13. Distributed framework for deployment and implementation of the proposed method.

superior diagnostic results. In contrast, the fixed distillation temperature of KT_T0 may cause the S-model to struggle to capture crucial knowledge from the T-model at certain points, which in turn affects its final diagnostic performance.

V. Practical Considerations and Challenges

Despite the outstanding performance demonstrated by the proposed method in the previous chapters, introducing it into real industrial scenarios involves various practical considerations and potential challenges. We will focus on practical issues related to implementation infrastructure, method integration, and framework generalization to ensure the feasibility and effectiveness of the proposed framework in practical applications [15], [31], [45].

A. Implementation Infrastructure

The distributed framework for deployment and implementation of the proposed method is shown in Fig. 13, which is mainly composed of cloud servers, edge devices, and IoT networks. When implementing the deployment of the proposed framework, a series of practical considerations will be faced when it comes to the real industrial IoT environment. When selecting the training part of the proposed method for cloud server deployment, give priority to using the platform as a service (PaaS) form of the cloud. While providing high development efficiency, it can facilitate the secondary development of the proposed framework. Also, it is critical to consider the number of service devices and the level of task concurrency when selecting edge devices with appropriate processing power to ensure real-time diagnostics. When constructing the IoT network, the choices should be made based on actual application scenarios, including the selection

of suitable communication networks (Ethernet, 5G, etc.), communication protocols (MQ Telemetry Transport, Modbus, LoRaWAN, etc.), and the appropriate network topology. It is worth noting that the privacy and security of data should be taken into account when selecting infrastructure, and the private cloud service and the encryption of data protocols should be considered. In addition, the energy consumption of edge devices deserves equal attention.

B. Methodology Integration

System integration is also worth emphasizing when deploying the application. First, the compatibility of the proposed method with the data center, control algorithm, and other functional components in terms of operating system and algorithm language should be considered. Especially in terms of data, the sampling window of the model should be adjusted according to the actual sampling rate of the sensor and the network transmission rate. At the same time, the components should focus on real-time demand and performance compatibility in the process of mutual collaboration. In addition, an effective error-handling mechanism should be introduced.

C. Challenges in Generalizing Frameworks

When deploying and implementing the proposed method, in addition to considering the above factors, the proposed method also faces potential challenges of generalization. The proposed framework focuses on studying how to transfer knowledge from the T-model to the S-model. In actual application scenarios, more complex problems, such as changing working conditions and category drift may be faced, resulting in a decrease in model generalization ability. The T-model and S-model should be improved in a targeted manner based on the proposed adaptive knowledge distillation framework based on the actual situation.

VI. Conclusion

Reliable operation of mechanical equipment heavily depends on intelligent fault diagnosis. However, traditional cloud-based fault diagnosis methods face challenges, such as time delays and data loss. This has led to the emergence of edge computing-based fault diagnosis as a promising alternative. The hardware resources of edge devices in the industrial IoT are limited, which makes it challenging to balance diagnostic capabilities and operational efficiency. This study proposes a lightweight intelligent fault diagnosis framework specifically designed for IoT edge computing scenarios, using adaptive knowledge distillation. By integrating fault diagnosis with edge computing, real-time diagnostic capabilities are guaranteed. Knowledge transfer from a cloud-based deep neural network model to a lightweight edge-based model is facilitated through knowledge distillation. The S-model effectively learns knowledge from the T-model through dynamic adjustment of the distillation temperature, achieving a deeper understanding and generalization of complex knowledge representations. Experiments were conducted on a customized experimental platform. The proposed method achieved diagnostic accuracy improvements of 6.99% and 1.63% compared to Xception and Resnet, respectively. Additionally, it achieved an average improvement of 14.61% and 19.57% compared to Mobilenet and Mnasnet, respectively. In the lightweight testing, the proposed method achieved inference speeds that were 10.29% and 8.30% of Xception and Resnet, respectively, and 57.07% and 26.33% of Mobilenet and Mnasnet, respectively. Compared to the T-model, the memory decreased by 3.42% and the speed increased by 11.38%. Additionally, experiments were conducted on a publicly available data set, where the proposed method achieved an average diagnostic accuracy improvement of 4.58% compared to the S-model and an average improvement of 2.49% compared to the method without temperature-adaptive adjustments.

In summary, the application of the temperature-adaptive change method in knowledge distillation training effectively improves the training effect of the S-model and enhances its diagnostic ability in complex samples. This provides an effective way to optimize the deployment of lightweight intelligent fault diagnosis in IoT edge computing scenarios, further promoting the promotion and application of intelligent fault diagnosis in practical applications. However, traditional methods of knowledge refinement only learn the output of the teacher network, resulting in the loss of knowledge in the middle layer. Future work could attempt to exploit the information contained in the intermediate model layer by designing different knowledge representations, rather than only using the output information.

References

[1] J. Lin, H. Shao, X. Zhou, B.-P. Cai, and B. Liu, "Generalized MAML for few-shot cross-domain fault diagnosis of bearing driven by heterogeneous signals," *Expert Syst. Appl.*, vol. 230, Nov. 2023, Art. no. 120696.

[2] G. Li, J. Wu, C. Deng, Z. Chen, and X. Shao, "Convolutional neural network-based Bayesian Gaussian mixture for intelligent fault diagnosis of rotating machinery," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, May 2021.

[3] H. Zhao et al., "Intelligent diagnosis using continuous wavelet transform and Gauss convolutional deep belief network," *IEEE Trans. Rel.*, vol. 72, no. 2, pp. 692–702, Jun. 2023.

[4] H. Zhao, H. Liu, Y. Jin, X. Dang, and W. Deng, "Feature extraction for data-driven remaining useful life prediction of rolling bearings," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, Feb. 2021.

[5] Z. Chen, Z. Li, J. Wu, C. Deng, and W. Dai, "Deep residual shrinkage relation network for anomaly detection of rotating machines," *J. Manuf. Syst.*, vol. 65, pp. 579–590, Oct. 2022.

[6] H. Shao, X. Zhou, J. Lin, and B. Liu, "Few-shot cross-domain fault diagnosis of bearing driven by task-supervised ANIL," *IEEE Internet Things J.*, early access, Jan. 24, 2024, doi: 10.1109/JIOT.2024.3360432.

[7] S. Y. Zhang, L. Su, J. F. Gu, K. Li, L. Zhou, and M. Pecht, "Rotating machinery fault detection and diagnosis based on deep domain adaptation: A survey," *Chin. J. Aeronaut.*, vol. 36, no. 1, pp. 45–74, Jan. 2023.

[8] S. Endo and S. Yokogawa, "Analysis of the trends between indoor carbon dioxide concentration and plug-level electricity usage through topological data analysis," *IEEE Sensors J.*, vol. 22, no. 2, pp. 1424–1434, Jan. 2022.

[9] Y. Yuan et al., "A general end-to-end diagnosis framework for manufacturing systems," *Nat. Sci. Rev.*, vol. 7, no. 2, pp. 418–429, Feb. 2020.

[10] J. Xu, S. K. Liang, X. Ding, and R. Q. Yan, "A zero-shot fault semantics learning model for compound fault diagnosis," *Expert Syst. Appl.*, vol. 221, Jul 2023, Art. no. 119642.

[11] S. A. Li, T. F. Li, C. Sun, X. F. Chen, and R. Q. Yan, "WPConvNet: An interpretable wavelet packet kernel-constrained convolutional network for noise-robust fault diagnosis," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 15, 2023, doi: 10.1109/TNNLS.2023.3282599.

[12] Z. Y. Chen, X. Q. Wang, J. Wu, C. Deng, and D. D. Zhang, "Relational conduction graph network for intelligent fault diagnosis of rotating machines under small fault samples," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, Apr. 2023.
[13] T. M. Hewa, A. Braeken, M. Liyanage, and M. Ylianttila, "Fog computing and blockchain-based security service architecture for 5G Industrial IoT-enabled cloud manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7174–7185, Oct. 2022.
[14] X. J. Kong, Y. H. Wu, H. Wang, and F. Xia, "Edge computing for Internet of everything: A survey," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23472–23485, Dec. 2022.
[15] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
[16] S. S. Gill et al., "AI for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, Aug. 2022, Art. no. 100514.
[17] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[18] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4921–4934, Jun. 2019.
[19] Y. Wang, Y. Wang, C. Shi, L. Cheng, H. Li, and X. Li, "An edge 3D CNN accelerator for low-power activity recognition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 918–930, May 2021.
[20] X. Yu, X. Yang, Q. Tan, C. Shan, and Z. Lv, "An edge computing based anomaly detection method in IoT industrial sustainability," *Appl. Soft Comput.*, vol. 128, 2022, Art. no. 109486.
[21] W. S. Shi, J. Cao, Q. Zhang, Y. Li, and L. Y. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[22] Q. Liu, T. Xia, L. Cheng, M. van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in IoT edge systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1491–1502, Jun. 2022.
[23] Z. H. Janjua, M. Vecchio, M. Antonini, and F. Antonelli, "IRESE: An intelligent rare-event detection system using unsupervised learning on the IoT edge," *Eng. Appl. Artif. Intell.*, vol. 84, pp. 41–50, Sep. 2019.
[24] T. Jing, X. Tian, H. Hu, and L. Ma, "Deep learning-based cloud–edge collaboration framework for remaining useful life prediction of machinery," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7208–7218, Oct. 2022.
[25] K. L. Zhang, W. Huang, X. Y. Hou, J. H. Xu, R. D. Su, and H. Y. Xu, "A fault diagnosis and visualization method for high-speed train based on edge and cloud collaboration," *Appl. Sci.* vol. 11, no. 3, p. 1251, Feb. 2021.
[26] L. Ren, Z. Jia, T. Wang, Y. Ma, and L. Wang, "LM-CNN: A cloud-edge collaborative method for adaptive fault diagnosis with label sampling space enlarging," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9057–9067, Dec. 2022.
[27] W. Qizhao, Q. Li, K. Wang, H. Wang, and Z. Peng, "Efficient federated learning for fault diagnosis in industrial cloud-edge computing," *Computing*, vol. 103, no. 10, pp. 2319–2337, 2021.
[28] T. Wang, Y. Liang, X. Shen, X. Zheng, A. Mahmood, and Q. Z. Sheng, "Edge computing and sensor-cloud: Overview, solutions, and directions," *ACM Comput. Surv.*, vol. 55, no. 13, pp. 1–37, 2023.
[29] G. Qian, S. Lu, D. Pan, H. Tang, Y. Liu, and Q. Wang, "Edge computing: A promising framework for real-time fault diagnosis and dynamic control of rotating machines using multi-sensor data," *IEEE Sensors J.*, vol. 19, no. 11, pp. 4211–4220, Jun. 2019.
[30] Y. Wang, J. Wu, Z. Yu, J. Hu, and Q. Zhou, "A structurally re-parameterized convolution neural network-based method for gearbox fault diagnosis in edge computing scenarios," *Eng. Appl. Artif. Intell.*, vol. 126, Nov. 2023, Art. no. 107091.
[31] S. Lu, J. Lu, K. An, X. Wang, and Q. He, "Edge computing on IoT for machine signal processing and fault diagnosis: A review," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11093–11116, Jul. 2023.
[32] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[33] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[34] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," 2014, *arXiv:1404.0736*.
[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 1800–1807.
[36] C. Xu, W. Gao, T. Li, N. Bai, G. Li, and Y. Zhang, "Teacher-student collaborative knowledge distillation for image classification," *Appl. Intell.*, vol. 53, pp. 1997–2009, 2022.
[37] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4388–4403, Aug. 2022.
[38] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021.
[39] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[40] Q. Zhu, X.-W. Wang, N. Zhang, Y. Xu, and Y. He, "Novel K-medoids based SMOTE integrated with locality preserving projections for fault diagnosis," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–8, Nov. 2022.
[41] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[42] M. Tan et al., "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.
[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[44] S. Y. Shao, S. McAleer, R. Q. Yan, and P. Baldi, "Highly accurate machine fault diagnosis using deep transfer learning," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2446–2455, Apr. 2019.
[45] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

Yanzhi Wang received the M.E. degree in naval architecture and marine engineering from Huazhong University of Science and Technology, Wuhan, China, in 2021, where he is currently pursuing the Ph.D. degree in mechanical engineering with the School of Aerospace Engineering.

His current research interests include deep learning, edge intelligence, and intelligent monitoring and operation of equipment.

Ziyang Yu received the bachelor's degree in electrical engineering and its automation from Wuhan Engineering University, Wuhan, China, in 2022. He is currently pursuing the master's degree in electronic information with the School of Automation, China University of Geosciences, Wuhan.

His current research interests include data processing, machine learning, and deep neural networks.

Jinhong Wu received the B.S. degree in flight vehicle design and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2021, where he is currently pursuing the M.E. degree in mechanical engineering with the School of Aerospace Engineering.

His current research interests include deep learning, physics-informed neural networks, and fault diagnosis.

Chu Wang received the B.S. degree in aircraft design and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2022, where he is currently pursuing the M.E. degree in mechanical engineering with the School of Aerospace Engineering.

His current research interests include deep learning and physics-informed machine learning that utilizes neural operators for partial differential equations solving.

Qi Zhou received the Ph.D. degree in mechanical engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2018.

He is an Associate Professor with the School of Aerospace engineering, HUST. He has authored or co-authored over 100 papers in peer-reviewed journals and conference proceedings. His research focuses on machine learning, intelligent optimal design of equipment, and digital twins.

Dr. Zhou was awarded the National Defense Innovation Award from China, in 2018 and the Outstanding Young Researcher from HUST.

Jiexiang Hu received the Ph.D. degree in mechanical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2019.

From 2019 to 2022, he worked as a Postdoctoral Researcher with the School of Aeronautics and Astronautics, Huazhong University of Science and Technology, where he is an Assistant Professor with the School of Aerospace Engineering. He has authored or co-authored over 30 papers in peer-reviewed journals and conference proceedings. His main research interests are intelligent optimal design methods, model verification, and validation.