Engineering Applications of Artificial Intelligence 126 (2023) 107091

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A structurally re-parameterized convolution neural network-based method for gearbox fault diagnosis in edge computing scenarios

Yanzhi Wang[a], Jinhong Wu[a], Ziyang Yu[b], Jiexiang Hu[a], Qi Zhou[a,*]

[a] School of Aerospace Engineering, Huazhong University of Science and Technology, Wuhan, 430074, China
[b] School of Automation, China University of Geosciences, Wuhan, 430074, China

ARTICLE INFO

Keywords:
Lightweight neural network
Edge computing
Fault diagnosis
Structural re-parameterization
Noisy environments
Deep learning

ABSTRACT

Gearboxes operate in harsh environments. Cloud-based techniques have been previously adopted for fault diagnosis in Gearboxes. Cloud-based fault diagnosis methods are prone to time delays and loss of information. Therefore, edge computing-based fault diagnosis becomes an option. However, with limited hardware resources for edge devices, balancing the diagnostic capabilities of the model with operating performance becomes a challenge. This paper proposes a lightweight convolutional neural network for gearbox fault diagnosis in edge computing scenarios to achieve an accurate diagnosis and lightweight deployment of models. By constructing the Mel-Frequency Cepstral Coefficients (MFCC) feature matrix of input data, the methodology can suppress noise interference and improve diagnostic accuracy. Based on the structural re-parameterization, the model structure transforms from multiple branches at training time to a single branch at inference time. This improves the inference speed of the model and reduces the hardware cost when the model is deployed while ensuring that the diagnostic capability of the model remains unchanged. Validation experiments were conducted on a public dataset and a custom experimental device, using the NVIDIA Jetson Xavier NX kit as the edge computing platform. According to the experiment result, after extracting the MFCC feature matrix, the average diagnostic accuracy rate in the noisy environment of the presented methodology is improved by 12.22% and 9.44%, respectively. After structural re-parameterization, the Memory of the model decreases by 52.58%, and the inference speed is increased by 38.83%.

1. Introduction

The gearbox is a vital component in industrial systems as it plays an indispensable and critical role in power transmission. Its high load capability and efficiency make it an extensively used component in various sectors, including energy and transportation. However, the high intensity of usage makes it easy to fault. Fault diagnosis can determine the type, severity, and location of the fault in time. Which can help identify the root cause and target repair work before a catastrophic failure occurs. To assure the reliability and stability of industrial systems, it is necessary to research the fault diagnosis of gearboxes (Kumar, et al., 2020; Kumar et al., 2022; Kumar et al., 2021; Lei et al., 2014; Shao et al., 2021).

Gearboxes often operate in harsh environments with multiple types of noise interference, which creates many difficulties for fault diagnosis. Therefore, ensuring the effectiveness of fault diagnosis methods in harsh environments is a key issue in gearbox health condition monitoring research (Chen, et al., 2016; Huo et al., 2022; Tang et al., 2022; Wang et al., 2019; Yu et al., 2022). Zhao et al. (Zhao, et al., 2019) proposed the deep residual shrinkage network. The network uses soft thresholding as a nonlinear transformation layer, which can eliminate features that are not important for the diagnosis process and improves the feature learning capability for high noise vibration signals. By experimenting with various types of noise, high fault diagnosis accuracy is achieved. Zhang et al. (Zhang, et al., 2020) proposed a fault diagnosis methodology based on signal processing where the model takes the raw time signal directly as input without any noise reduction pre-processing. In addition, the acquired signals containing noise often have strong nonlinear characteristics, which can lead to relatively low diagnostic accuracy when directly applied to the original input. Therefore, Wang et al. (Wang, et al., 2022) proposed an approach to fault diagnosis based on time-frequency representation with DRL. By converting the vibration signal into TF maps of uniform size, a diagnostic agent is established under the DRL framework, and the methodology is experimentally

* Corresponding author.
E-mail address: qizhouhust@gmail.com (Q. Zhou).

https://doi.org/10.1016/j.engappai.2023.107091
Received 3 April 2023; Received in revised form 10 August 2023; Accepted 31 August 2023
Available online 6 September 2023

verified to have good generalization and stability.

The above fault diagnosis methods can achieve good results. However, most of the existing studies focus on improving the accuracy of fault diagnosis and ignore resource efficiency. Their success relies mainly on increasing the number of network layers or adding certain network structures to improve the feature extraction capability of the model, which usually increases the model parameters and hardware overhead (Li, et al., 2022; McDonald et al., 2012; Wu et al., 2021). In engineering applications, industrial system health monitoring is often performed using cloud computing (Hewa, et al., 2022). The method collects the operational data of gearboxes and uploads them to the server on the cloud computing side through the network. Based on the sufficient computing and storage resources on the cloud computing side, the algorithm model is iteratively optimized and the inference of the gearbox health status is realized in the cloud. Typically, gearboxes operate in harsh environments, thus cloud computing (which requires multiple levels of data transfer) often faces data loss and time delays. Therefore, the use of cloud computing can affect the accuracy and real-time fault diagnosis (Shi, et al., 2016).

In the next-generation application paradigm of artificial intelligence, edge computing is emerging as a key enabling technology that is attracting widespread academic attention. As a new computing paradigm, it moves the computing of applications, data, and services from the central node of the network (cloud servers, etc.) to the edge nodes of the network (edge gateways, intelligent cards, etc.). Edge computing disaggregates the large services processed by the cloud into smaller and more manageable parts, which are distributed to the edge for processing. It can reduce the response time of the system, reduce the load on the transport network, and reduce the memory and computing overhead of cloud computing (Chiang and Zhang, 2016; Gill et al., 2022; Kong et al., 2022; Samie et al., 2019). Shi et al. (Shi, et al., 2016) propose a definition of edge computing, conduct research for several application cases, and present several challenges and opportunities in the field of edge computing. Compared with cloud servers, edge devices provide extremely limited resources such as computation and storage, and it is a challenging problem for edge computing to fully utilize the resources of edge devices to satisfy the most complex services possible (Zeng, et al., 2022). Gao et al. (Gao, et al., 2020) proposed a lightweight EdgeDRNN, which uses incremental network algorithms to exploit temporal sparsity in RNNs and can run on the cheapest FPGAs, reducing DRAM-weighted memory accesses by a factor of 10. Zhao et al. (Zhao, et al., 2020) proposed an INES technique using deep learning to reduce the computational cost by building lightweight deep neural networks through deeply separable convolution. The methodology can be deployed based on an edge cloud collaboration architecture, reducing the bandwidth load and achieving high test accuracy in the field at a tenth of the operating cost of a centralized server. In the area of industrial health monitoring, edge computing offers new methods for processing and mining monitoring data. Jing et al. (Jing, et al., 2022) proposed a deep learning-based framework for cloud-based collaboration, using the collaboration between the cloud and the edge for more accurate RUL prediction and significantly reduced model training time. Zhang et al. (Zhang, et al., 2021) build a fault diagnosis model for the train, using a stacked autoencoder deep neural network, which is then deployed at the edge with a migration learning strategy to achieve fast fault localization. However, related studies mainly focus on developing algorithmic models with the concept of edge computing for the research object to improve the effectiveness of the model in relevant aspects (Errandonea, et al., 2023; Huong et al., 2021; Khalil et al., 2021; Liang et al., 2022; Yao et al., 2020; Zhang et al., 2023). Due to the high number of parameters and hardware overhead of traditional fault diagnosis models, the computational resources of edge devices are limited and limited in deployment and application. Therefore, it is important to develop a lightweight gearbox fault diagnosis-based approach.

To address the above problems, this article introduces a gearbox fault diagnosis methodology using a lightweight convolutional neural network with edge computing. The following are the main contributions of this paper:

A. A 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.
B. To prevent the noise in the monitoring signal from interfering with the diagnostic accuracy, noise interfering with monitoring data is suppressed by extracting the MFCC feature matrix, and the fault features can be enhanced at the same time.
C. Based on the principle of structural re-parameterization, the model is transformed from multi-branching during training to single-branching during inference. Which can improve the inference speed and reduce the hardware overhead of computing devices while ensuring the fault diagnosis capability.
D. By comparing and analyzing the test results on two experimental devices and the relevant parameters of the model. It is demonstrated that the proposed method in this paper has better network extraction capability, smaller memory footprint, and faster speed.

The rest of the paper is structured as follows. Section 2 presents the background of the technique from related work. The details of the proposed method are presented in Section 3. Experiments and evaluation are demonstrated in Section 4. Finally, Section 5 presents the conclusions and future work.

2. Related works

The related works are briefly reviewed in this section, including work on edge computing, lightweight neural network, and Mel-frequency cepstral coefficients feature.

2.1. Edge computing-based fault diagnosis

Edge computing is closer to the monitored device, allowing for faster processing and fewer delays. Applying this distributed architecture to the fault diagnosis of the device, data analysis, and diagnosis results generation are closer to the target device. Therefore, helps the device to make immediate feedback to the diagnosis results. And it can filter most of the rubbish data during device operation, effectively reducing the cloud loading. Making large-scale device connectivity and large-scale data processing possible in monitoring industrial device health. Qian et al. (Qian, et al., 2019) propose a method for real-time fault diagnosis and dynamic control of rotating machinery based on edge computing. Sensor signals are collected in parallel by a designed edge computing node. Then, feature extraction and fault diagnosis are performed. The motor can be controlled when an emergency fault is diagnosed. Ren et al. (Ren, et al., 2022) propose a cloud-edge collaborative adaptation approach for fault diagnosis for scene-specific equipment in cloud manufacturing systems. The variety of diagnosable faults is extended by a sampling space expansion method. Wang et al. (Qizhao, et al., 2021) propose an efficient asynchronous federated learning method. This method allows edge nodes to select part of the model from the cloud for asynchronous updates based on local data distribution, thereby reducing computation and communication. This method can reduce resource requirements at the edge, reduce communication, and improve training speed in heterogeneous edge environments. Therefore, the benefits of edge computing are as follows:

A. Low delay: computing power is deployed close to the device, and device requests are responded to in real-time.
B. Operating at low bandwidth: Moving work closer to users or equipment, can reduce the impact of bandwidth limitation.

C. Privacy protection: Data is collected locally, analyzed locally, and processed locally, reducing the exposure of data to public networks and protecting data privacy effectively.

Edge computing offers a wealth of benefits. However, its adoption comes with several significant challenges. Notably, one of the key challenges is the inherent limitation of computing capacity at edge nodes. Often constrained by size, power, and resource availability, these devices can struggle to handle resource-intensive computing tasks.

2.2. Lightweight neural network

In edge computing scenarios, there are usually abundant software and hardware resources available in the cloud for model training, but limited resources in the edge for model inference. Therefore, we aim to have a more complex model structure for training to achieve higher accuracy and a smaller model structure for inference that preserves the same level of accuracy. Dai et al. (Dai, et al., 2021) propose an improved knowledge distillation method, which enables the transfer of the complex mapping functions learned by cumbersome models to relatively simpler models. The method can be effectively applied to intelligent edge computing. Fang et al. (Yu, et al., 2021) propose a novel global pruning method. Formulating the pruning problem as a performance improvement sub-problem and a global pruning sub-problem by introducing an alternating direction method of multipliers. The method can compress and accelerate the DNNs for efficient edge computing. Huang et al. (Huang, et al., 2023) propose an integrated cloud-edge-device framework that connects the edge, the remote cloud, with the device through cross-platform web technology for adaptive deep learning services, achieving lower latency, lower mobile power, and higher system throughput. However, whether pruning or knowledge distillation, it requires secondary training and has some difficulty in training.

Since a set of training parameters of a neural network corresponds to a network structure, a set of parameters of one network structure can be transformed into another set of parameters, and the transformed parameters can be used for the other structure. The two network structures are equivalent as the conversion of the parameters is equal, which is called structural re-parameterization (Ding et al., 2019; Ding et al., 2021a,b). Structural re-parameterization is implemented by first constructing a network structure for training and a network structure for inference, then converting the parameters from training equivalently to another set of parameters for inference.

2.3. Mel-Frequency Cepstral Coefficients feature

MFCC can simulate the nonlinear characteristics of the human ear with a Mel filter bank and is usually used to extract signal features in sound signal processing. The core idea of MFCC is based on the fact that human has different auditory sensitivities to the incoming sound of different frequencies, i.e., the lower frequency with the higher resolution. MFCC feature coefficients describe the spectral envelop characteristics of the sound signal and can suppress interference band information, which has a high recognition rate in practical applications (Li, et al., 2022; Yan et al., 2022). The extraction steps of the MFCC feature are shown in Fig. 1. The following will be introduced separately:

Step 1: Framing & windowing. Because MFCC is usually used in automatic speech, and the audio signal is constantly changing, the signal is framed by the window function to obtain a stable frame

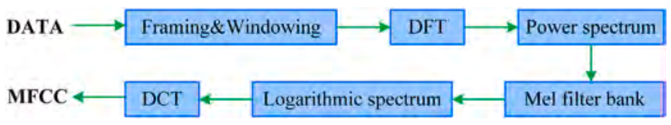


Fig. 1. MFCC feature extraction and MFCC feature matrix construction process.

signal. The commonly used window function is the Hann and Hamming window, whose expression is:

$$H(n) = a_0 - (1 - a_0)\cos\left(\frac{2\pi x}{L}\right), (n = 0, 1, \ldots, L-1) \tag{1}$$

where L refers to how many data points are contained in every frame. If a_0 is set as 0.54 or 0.5, the above expression is named Hamming window or Hann window respectively. Here we utilize the Hamming window function.

Step 2: Discrete Fourier transforms (DFT). As a result of the inherent limitations in analyzing signal characteristics in the time domain, it is customary to convert signals into energy distributions in the frequency domain for analysis. The spectrum *T(k)* of the signal fragment *H(n)* is extracted by applying the discrete Fourier transform to the signal fragment *H(n)* by the formula:

$$T(k) = \sum_{n=0}^{L-1} H(n) e^{-\frac{j2\pi kn}{L}}, (n = 0, 1, \ldots, L-1) \tag{2}$$

Step 3: Power spectrum calculations. Since the power spectrum can reflect the main signal features, the power spectrum *Pt* is obtained by calculating the modular square:

$$Pt = T(n).\ \hat{T}(n) \tag{3}$$

Step 4: Mel triangle filtering. Motivated by human hearing that the cochlea is a series of filter banks, which solely focuses on some specific frequencies. Some band-pass filters are utilized to separate them, i.e., the Mel filter bank. The first filter is very narrow since the low-frequency space possesses a lot of information. As the frequencies get higher the filters get wider, which is consistent with the human ear. To offset this feature, the Mel scale is utilized to convert frequency *f*, which is written as:

$$f_{mel} = 2595\ \lg(1 + f\ /\ 700) \tag{4}$$

The Mel scale describes the nonlinear characteristics of human auditory frequency perception and tells us exactly how to space the filter banks and how wide to make them. Next, the information in each frequency band is filtered using T triangular filters, and the process is as follows:

$$H_t(n) = \begin{cases} 0, & \\ \dfrac{2 \times (n - f(t-1))}{(f(t+1) - f(t-1)) \times (f(t) - f(t-1))}, & n < f(t-1) \\ & f(t-1) \le n \le f(t) \\ \dfrac{(2 \times f(t+1) - n)}{(f(t+1) - f(t-1)) \times (f(t+1) - f(t))}, & f(t) \le n \le f(t+1) \\ & n \ge f(t+1) \\ 0, & \end{cases} \tag{5}$$

where t = 1,2,3,$\cdots$ 24. which means 24 independent T triangular filters, $f(t)$ is the t^{th} central frequency, n = 0,1, … , L/2 – 1 is the Mel scale in each filter. Here, we have:

$$\sum_{t=0}^{T-1} H_t(n) = 1 \tag{6}$$

Step 5: The logarithmic spectrum *S(t)*. To make the data processing results more robust to noise, the output power spectrum is taken as a logarithmic operation to obtain the logarithmic spectrum *S(t)* for each filter in Step 4, and the formula is:

$$S(t) = \ln\left(\sum_{n=0}^{L-1} |P_t|^2 H_t(n)\right)(0 \le t \le T) \tag{7}$$

Where: $H_t(n)$ is the t^{th} filter bank, $S(t)$ is the logarithmic spectrum, P_t is the dispersive power spectrum obtained in step 3, and T is the number of filter banks. The logarithmic operation reflects that the human does not hear loudness on a linear scale. 8 times the energy needs to be put in if we want to double the sound volume.

Step 6: Discrete cosine transforms (DCT). Because the filter banks are overlapping, the above energies are quite correlated with each other. The transformation is used to calculate the spectral components of different frequency bands, making the dimensional vectors of each band independent of each other, thus obtaining the information of each frequency band of the signal by the following formula:

$$c(x) = \sum_{t=0}^{L-1} S(t)\cos\left(\frac{x\pi(t-0.5)}{T}\right), 0 \le x \le T \tag{8}$$

where $c(x)$ is the MFCC feature of the input data sample. S(t) is the t^{th} logarithmic spectrum.

3. Proposed method

In this section, the main procedure of the fault diagnosis methodology using edge computing and the methodology of constructing the MFCC feature matrix are introduced. The proposed lightweight convolutional neural network and the implementation principle of structure re-parameterization are elaborated on in detail.

3.1. Edge computing-based fault diagnosis method for gearboxes

To solve the problems of inconspicuous fault characteristics of gearbox monitoring signals and lightweight deployment of models, the main flow chart of the proposed edge computing-based gearbox fault diagnosis methodology is given in Fig. 2. The method is divided into two parts: the cloud computing side, which is far from the gearbox, and the edge computing side, which is close to the gearbox. At the cloud computing side, the monitoring data from the edge side is firstly labeled, the data is divided into the training set and validation set, the signal samples are obtained by sliding window sampling, the MFCC feature matrix is extracted from the signal samples, the MFCC is imported into the SrepCNN-T neural network, and iterative training is performed based on the powerful computing performance of cloud computing. Finally, the optimal weights of the neural network are sent down to the edge side after training. At the edge computing side, firstly, the collected real-time monitoring data is sampled by sliding window and the MFCC feature matrix of samples is extracted, and then the MFCC feature matrix of the sample is imported into the SrepCNN-I neural network, and the SrepCNN-I neural network has imported the optimal weights of the neural network issued from the cloud computing side, then the accurate fault diagnosis results can be obtained quickly at the edge side. It is worth mentioning that in engineering applications, the training of models can be done periodically on the cloud, depending on the actual situation. The updated model weights are sent down to the edge through file streams, message queues, etc. In summary, the edge computing-based gearbox fault diagnosis methodology consists of two links: MFCC matrix acquisition and Srep-CNN for fault diagnosis.

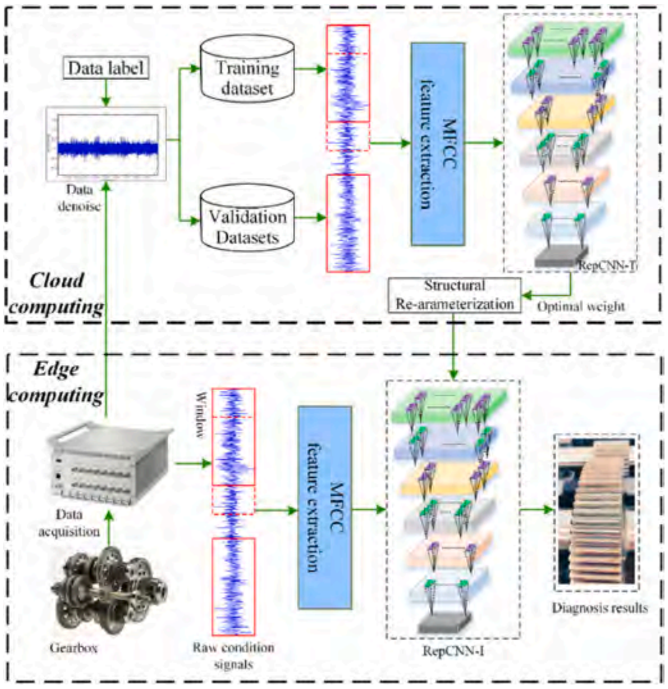


Fig. 2. The procedure of gearbox fault diagnosis method based on edge computing.

3.2. MFCC feature matrix construction

The monitoring signals are full of noise and other useless information due to the working conditions and environment, which affects the accuracy of the model heavily. The MFCC feature matrix involves MFCC feature extraction and MFCC feature matrix construction.

Since the MFCC features only describe the static spectral envelope information of signals, in reality, the fault information of gearboxes is mostly hidden in the dynamic information. Firstly, the MFCC features of gearbox monitoring data are extracted based on the method introduced in 2.1. More fault information is obtained from the MFCC features using first-order difference calculation and second-order difference calculation. The calculation process of the first-order difference and the second-order difference is as follows.

$$c_1(x) = \frac{1}{\sqrt{\sum_{i=-n}^{i=n} i^2}} \sum_{i=-n}^{i=n} i \times c(x+i) \tag{9}$$

$$c_2(x) = \frac{1}{\sqrt{\sum_{i=-n}^{i=n} i^2}} \sum_{i=-n}^{i=n} i \times c_1(x+i) \tag{10}$$

The MFCC feature matrix M_{data} constructed in this section is obtained by combining the obtained $c(x)$, $c_1(x)$ and $c_2(x)$, which is performed as:

$$M_{\text{data}} = [c(x)c_1(x)c_2(x)] \tag{11}$$

Where: [•] indicates the first and last concatenation of the different inputs.

3.3. Structural Re-parameterized convolutional neural network

3.3.1. Network architecture

To achieve a lightweight deployment of the network, we designed a structurally re-parameterized convolutional neural network called SrepCNN, and the network structure is shown in Fig. 3. The network adopts a modular design, and the number of layers of module A and module B can be set according to the actual application scenario. By adjusting the depth of it, the feature extraction capability can be changed. The network is divided into a train state and an inference state, and the network in the two states is called SrepCNN-T and SrepCNN-I, respectively.

SrepCNN-T consists of multiple network stages in series, each network module consisting of a 3*3 convolutional branch, a 1*1

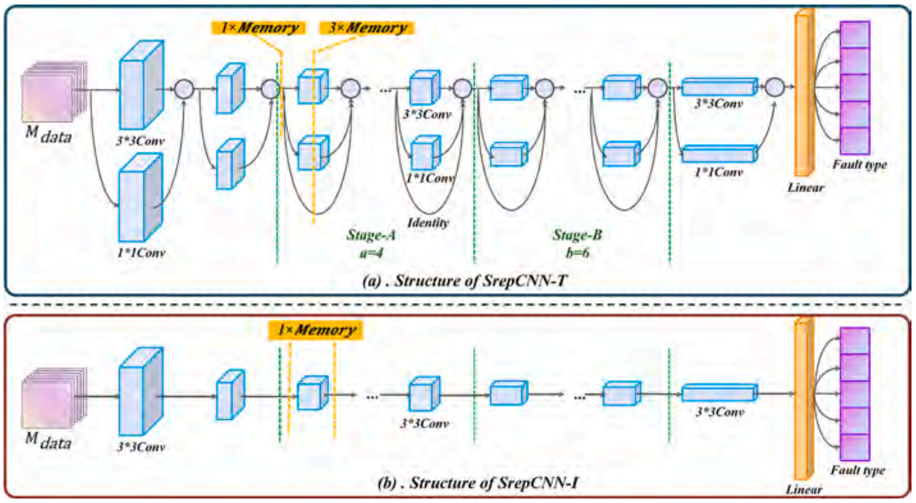


Fig. 3. Structure of structurally re-parameterized convolutional neural networks.

convolutional branch, and an identity branch, followed by a convolutional kernel and a batch normalization (BN) layer. and the SrepCNN-I consists of multiple 3*3 convolutional modules in series.

When the model is trained iteratively on the cloud side and is in the training session, the SrepCNN-T network is used. Since the identity branch can be considered as degenerate 1*1 conv and 1*1 conv can be further considered as degenerate 3*3 conv. Therefore, after the model training is finished, a simple algebraic transformation is applied to the model weights to remove the identity branch and 1*1 conv branch through structure re-parameterization, and the new weights are saved and deployed to the edge for testing.

When the model has been deployed on the edge computing platform and is in inference session, the SrepCNN-I network is used. The structural re-parameterization method is used to transform the network structure, mainly considering the following reasons:

A. **Better network extraction capability:** Inspired by Resnet, the multi-branch architecture can be seen as a collection of many paths of differing lengths, making the model an implicit collection of many shallower models. Which can increase the feature extraction faculty of the network. Residual networks introduce short paths which can carry a gradient throughout the extent of very deep networks. Which avoids the vanishing gradient problem, and makes it more conducive to training (Belongie, et al., 2016; He et al., 2016).
B. **Smaller memory usage:** Considering the limited resources of edge-side devices and the increased memory peak due to the feature maps of each branch need to be saved until the summation is completed, the multi-branch network structure is memory inefficient as shown in Fig. 3. Compared with the multi-branch structure of T-network, the single-branch structure of I-network can significantly reduce the cost of memory.
C. **Faster speed:** SrepCNN-I uses only 3*3 conv because it is optimized in GPU and CPU by many computing libraries (e.g., Nvidia cuDNN & Intel MKL), and the theoretical computing density of 3*3 conv is about 4 times higher than other convolutional methods, so using 3*3 conv can effectively improve the model's inference speed on edge devices (Chetlur, et al., 2014; Lavin and Gray, 2016).

3.3.2. *Implementation of structural re-parameterization*

When the model is iteratively trained to obtain the optimal weights, the structural re-parameterization methodology is applied to convert the parameters of the SrepCNN-T network structure into another set of parameters coupled with the SrepCNN-I network structure. We can equivalently replace the former with the latter, thus achieving a change in the network architecture. The conversion process is shown in Fig. 4.

Fig. 4. (a) depicts the changes in the structure of the model during the conversion of the trained SrepCNN-T basic module into a single 3*3 conv layer in the SrepCNN-I network, and Fig. 4. *(b)* depicts the parameter changes of the neural network. We use $M^{(k)} \in [C_1, C_2, k, k]$ to describe a convolutional layer with a convolutional kernel $k*k$, where C_1 and C_2 are the number of input channels and output channels, respectively. The computational process for the BN layer is described using the following equation:

$$BN(X_{BN}, \alpha, \beta, \gamma, \mu) = \frac{X_{BN} - \alpha}{\sqrt{\beta^2 + \varepsilon}} \cdot \gamma + \mu \tag{12}$$

Where X_{BN} is the input for the BN layer, $\alpha, \beta, \gamma, \mu$ indicates the cumulative mean, standard deviation, and learned scale factor of the BN layer and bias, respectively, and ε is a very small constant that can prevent the denominator from being zero. The BN layer in each branch of the basic module of the SrepCNN-T network can be further transformed by:

$$BN(X_{BN}, \alpha, \beta, \gamma, \mu) = \frac{\gamma}{\beta} \cdot X_{BN} + \left(-\frac{\alpha \cdot \gamma}{\beta} + \mu\right) = M'^{(k)} * X_{Conv} + \mu^{(k)} \tag{13}$$

Where $M'^{(k)}$ refer to the "virtual convolutional layer" of the branch in the process of structural reparameterization. This transformation also applies to the identity, since the identity can be considered as a 1*1 convolution with the unit matrix as the convolution kernel. Then the result of a SrepCNN-T basic module can be expressed as:

$$\begin{aligned} W(X) &= F(X) + D(X) + Identity(X) \\ &= BN\left(M'^{(3)} * X, \alpha^{(3)}, \beta^{(3)}, \gamma^{(3)}, \mu^{(3)}\right) \\ &\quad + BN\left(M'^{(1)} * X, \alpha^{(1)}, \beta^{(1)}, \gamma^{(1)}, \mu^{(1)}\right) \\ &\quad + BN\left(X, \alpha^{(0)}, \beta^{(0)}, \gamma^{(0)}, \mu^{(0)}\right) \\ &= M'^{(3)} * X + \mu^{(3)} + M'^{(1)} * X + \mu^{(1)} + X + \mu^{(0)} \\ &= M' * X + \mu' \end{aligned} \tag{14}$$

Where M' refer to the "virtual convolutional layer" of three-branch after reparameterizing. The three additive numbers in the formula represent the output of the 3*3 conv branch, 1*1 conv branch, and identity in the basic module of SrepCNN-T, respectively. The identity and 1*1 conv branches are filled with 0 to be able to transform to the size of 3*3. Further, the 1*1 kernel is summed to the centroid of the 3*3 kernel to obtain the final 3*3 kernel, and the three deviation vectors are summed to get the total deviation, completing the constant transformation of integrating the three branches into one 3*3 conv. Thus the SrepCNN-I can be used to equivalently replace the SrepCNN-T.

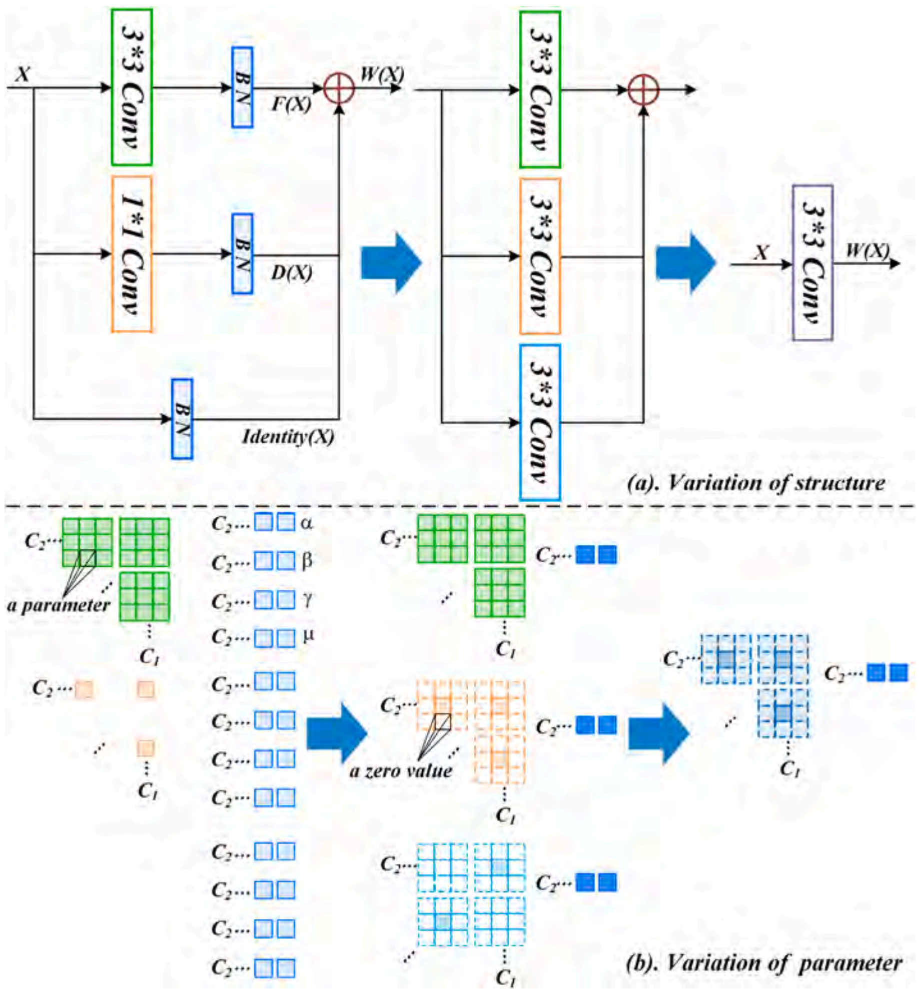


Fig. 4. Implementation process of structural re-parameterization.

4. Case studies

4.1. Experimental method

In this paper, all case studies were conducted on a computer [IntelCore i7-11700K@3.60GHz processor, NVIDIA GeForce RTX 3070 GPU, 32.0 GB of RAM], and a Jetson Xavier NX kit [NVIDIA Carmel ARM CPU, NVIDIA Volta GPU with 48 Tensor Cores, 8.0 GB RAM]. The first one is used as the cloud device and the second one as the edge device. On the cloud device, the dependency environment of models is based on Windows. on the edge device, it is deployed via Docker (an open-source application container engine). In addition, models were written in Python 3.9 and Pytorch 1.11.0.

The experimental procedure is shown in Fig. 5. The experimental data set is divided into a training data set, a verification data set, and a test data set. The training and verification data sets are saved in the cloud device. The test data set is saved in the edge device. Models are trained on the cloud device to obtain the best weight. The weight is downlinked to the edge device via file streams. On the edge device, models load the weight and run to get the test results. It is worth noting that in the engineering application of the proposed method, the updating and distribution of the best weight can be done periodically. In the experiments, since there is no data update in real-time during the training process, the downlink of model weight is performed only once.

The MFCC-SrepCNN-T model is first trained on the cloud device, the structure is re-parameterized based on the best weights, and the re-parameterized weights are sent down to the edge devices. The MFCC-SrepCNN-I model is deployed on edge devices. Given the difficulty of the experiments, the network structure used in the two application cases in this paper, a and b in Stage-A and Stage-B take values of 4 and 6, respectively. The Hyper-parameters of MFCC-SrepCNN in the experiment are shown in Table 1. To verify the efficiency of the proposed method, the following models are used for comparison experiments:

A. **Resnet:** The degradation problem of deep networks is solved by residual connectivity, enabling the training of deeper networks (He, et al., 2016).
B. **Xception:** The network uses residual connectivity to enhance feature extraction, and uses depthwise separate convolution to achieve complete decoupling of cross-channel correlation and spatial correlation (Chollet, 2017).
C. **Deep residual shrinkage networks (DRSN):** Soft thresholding in the network structure can filter noise-related features, effectively improving the ability to identify features from noisy data. It is a classical model used in recent years for fault diagnosis in noisy environments (Zhao, et al., 2019).
D. **SrepCNN:** The samples are reshaped and fed directly into SrepCNN-T for iterative training, and tested using SrepCNN-I. The difference between this method and the proposed method is that the MFCC matrix is not extracted in this method.
E. **MFCC-SrepCNN-T:** The MFCC feature matrix of samples is extracted and then input to SrepCNN-T for iterative training, and the test session still uses SrepCNN-T. The difference between this method

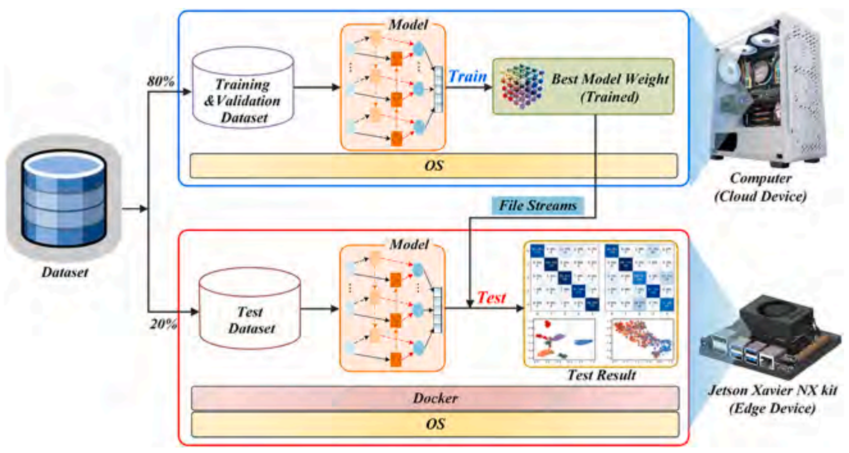


Fig. 5. The procedure of experimental.

Table 1
The hyper-parameters of MFCC-SrepCNN.

Number of modules		Layer Name	Size		Layer Name	Size
1	MFCC-SrepCNN-T	MFCC	channels:6, mels:64	MFCC-SrepCNN-I	MFCC	6 channels, 64 mels
1		3*3 Conv	channels:48, stride:1		3*3 Conv	48 channels, stride 1
		1*1Conv	channels:48, stride:1			
1		3*3 Conv	channels:48, stride:2 channels:48, stride:2		3*3 Conv	48 channels, stride 2
		1*1Conv				
4		3*3 Conv	channels:96, stride:2 → 1[a]		3*3 Conv	channels:96, stride:2 → 1
		1*1Conv	channels:96, stride:2 → 1			
		Identity	/			
6		3*3 Conv	channels:192, stride:2 → 1		3*3 Conv	channels:192, stride:2 → 1
		1*1Conv	channels:192, stride:2 → 1			
		Identity	/			
1		3*3 Conv	channels:1280, stride:2		3*3 Conv	channels:1280, stride:2
		1*1Conv	channels:1280, stride:2			
1		Linear	out_features:5		Linear	out_features:5

[a] Stride:2 → 1: The stride of the first module in this stage is 2, the other values are 1.

and the proposed method is that this method does not have structural re-parameterization.

F. **MFCC-SrepCNN:** The proposed method. The MFCC feature matrix of samples is extracted and then input to SrepCNN-T for iterative training, the model is structurally re-parameterized, and SrepCNN-I is used for the testing session.

Xception, Resnet, and DRSN are the classical and efficient convolutional neural networks in recent years. These models are chosen to validate the overall performances of the suggested methods in a noisy environment. In addition, SrepCNN is chosen as a comparison method to verify the gain effect of MFCC on the features extract capability of the method in a noisy environment. MFCC-SrepCNN-T is chosen as a comparison method to verify the effect of structural re-parameterization in MFCC-SrepCNN on the diagnostic ability of the method and the acceleration effect of the method. To verify the advantages of the proposed method, two real gearbox datasets are used to compare the performance metrics of each model, and model complexity metrics such as Params, Flops, Memory, and inference speed of a single sample are calculated for each model.

4.2. Case one

4.2.1. Experimental data description

The experimental gearbox data for Case 1 was obtained from the drivetrain dynamic simulator (DDS) at Southeast University, China. The simulation platform is shown in Fig. 6, the DDS can simulate a variety of bearing and gear operating conditions, and the gear operating condition data in this dataset were selected for the case study (Shao, et al., 2019).

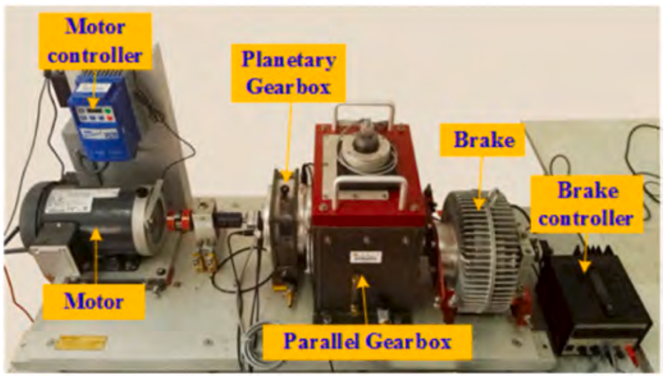


Fig. 6. Experimental equipment of DDS.

The acceleration signals in the x, y, and z directions of the experimental platform were collected using acceleration sensors attached to the parallel gearbox and planetary gearbox housings. Sampling was performed using a sliding window approach with a window overlap of 0. The window length was fixed at 3136, taking into account the rotation speed of the device and the sampling rate of the sensor. Ensuring that each sample contained at least one revolution of the gear. The data are subjected to a de-singularization and normalization operation. In this paper, the original experimental signal is augmented with multiple noise samples by incorporating random Gaussian white noise. The signal-to-noise ratio (SNR) is used as a metric to quantify the magnitude of the added noise relative to the original experimental signal. The SNR

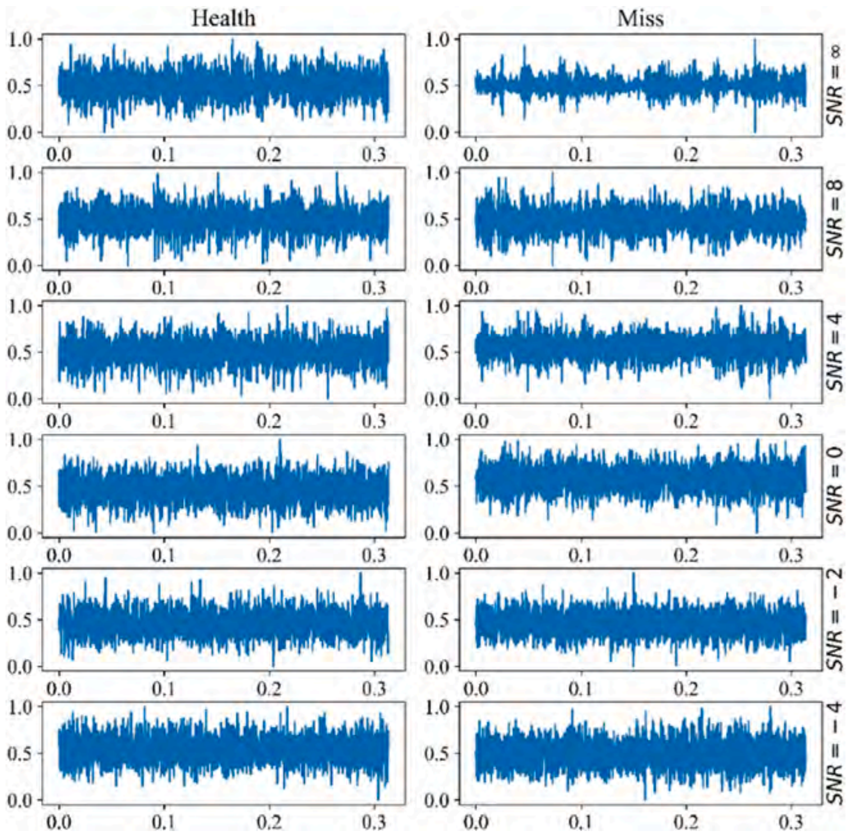


Fig. 7. The data of six different SNRs in case one.

function is expressed as:

$$SNR = 10\log_{10}(E_s / E_n)(\mathrm{dB}) \tag{15}$$

Where E_s is the energy of the data and E_n is the energy of Gaussian white noise. In general, SNR $= \infty$ indicates the case of no added noise.

To observe the effect of different noise levels on samples, Health and Miss states were examined as an example. Five different SNRs were applied to the same sample. Fig. 7 shows that without added noise, there is a clear distinction in the signal amplitude characteristics between the two states. However, as the noise level increases, it becomes increasingly challenging to identify the amplitude characteristics of the two states.

The completed dataset contains 668 samples for each operating state, which are randomly disrupted and subsequently partitioned into the training, validation, and testing sets using a ratio of 7:1:2. The different working states Health, Chipped, Miss, Root, and Surface are also labeled with 0–4 respectively.

4.2.2. *Results & discussion*

The samples input to each network are reshaped into a matrix of size [6, 56, 56], respectively. The Hyper-parameters of the model training during the experiments are shown in Table 2.

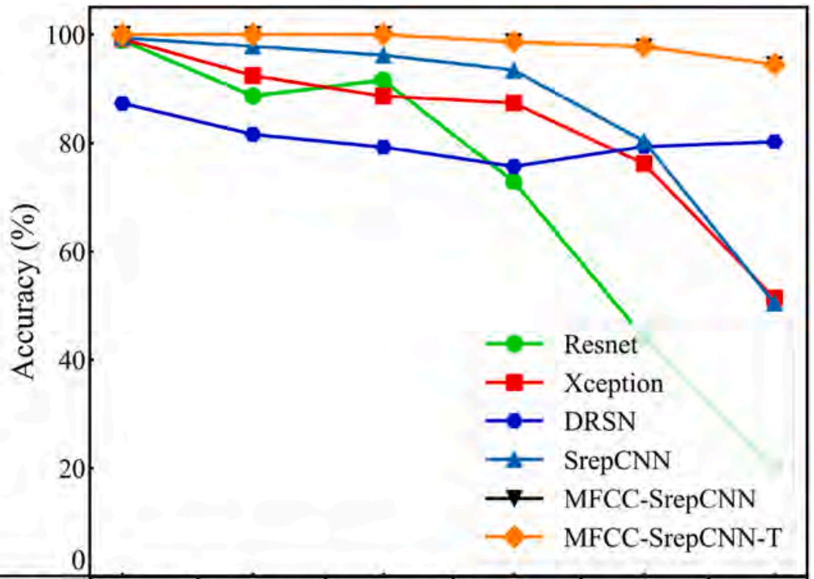
The Batch size is set to 64 considering the convergence and memory. Epoch is set to 50 according to the convergence of each model, in the pre-experiment. And CrossEntropy, which is the frequent loss function. CosineAnnealingLR is used to modify the learning rate (LR) of the network during training. The method adjusts the LR by the cosine function, which can make the learning take the lead in a slow decline, then accelerate the decline, and then slowly decline again. This method yields advantageous results in accelerating model convergence and enhancing model efficacy. Considering the differences of each model, three sets of experiments were set up according to the maximum LR. The group with the highest average accuracy was chosen as the final experimental result of the model.

To evaluate the proposed methodology in comparison with other classical techniques, three replicate experiments were conducted to record the classification accuracy under normal conditions and different noise environments, The experimental outcomes are presented in Fig. 8, where the prediction accuracy values of the model represent mean values of the three replicate experiments. It can be seen that the accuracy of MFCC-SrepCNN overlaps with that of MFCC-SrepCNN-T under each noise condition, indicating that the structural re-parameterization does not change the final test accuracy of the MFCC-SrepCNN model. The accuracy fold of MFCC-SrepCNN is above the accuracy fold of SrepCNN in each noisy environment, especially the accuracy of MFCC-SrepCNN is 43.99% higher than that of SrepCNN at SNR $= -4$, indicating that MFCC can significantly increase the diagnostic capability of the approach in the presence of noise. MFCC-SrepCNN is significantly ahead of Resnet, Xception, and DRSN, suggesting that the proposed model exhibits good diagnostic capability in noisy environments compared with the classical model. In addition, the mean diagnostic accuracy of SrepCNN surpasses that of Resnet, Xception, and DRSN, and the diagnostic accuracy stays above 90% in the noiseless and low-noise environments, indicating that the multi-branch structure used in SrepCNN also gains the feature extraction capability of the network.

To analyze the gain effect of MFCC on the classification ability of the model under different noise environments in a more intuitive way. The confusion matrices of the post-test sample classification results of SrepCNN and MFCC-SrepCNN are selected for plotting. Fig. 9 shows the

Table 2
The Hyper-parameters in the training process.

Epoch	Batch size	Loss Function	LR			
			LR _max			LR _min
50	64	CrossEntropy	0.01	0.005	0.001	LR _max*1e-2



Noise Intensity	∞	8	4	0	-2	-4
Resnet	*98.9*	*88.67*	*91.57*	*72.91*	*44.06*	*20.36*
Xception	*99.2*	*92.4*	*88.63*	*87.33*	*76.15*	*51.39*
DRSN	*87.28*	*81.59*	*79.24*	*75.65*	*79.34*	*80.24*
SrepCNN	*99.40*	*97.81*	*96.16*	*93.41*	*80.31*	*50.47*
MFCC-SrepCNN	***100***	***100***	***100***	***98.65***	***97.75***	***94.46***
MFCC-SrepCNN-T	*100*	*100*	*100*	*98.65*	*97.75*	*94.46*

Fig. 8. The test accuracy of different models across different noise environments in Case one.

confusion matrix corresponding to the intermediate values of the results of three replicate experiments conducted by the two models under different noise environments. Where the vertical indicates the actual labels of the samples, the horizontal labels indicate the prediction results. The prediction accuracy and the corresponding number of samples are labeled in the matrix.

Comparing the classification of each category of SrepCNN in Fig. 9 *(a)*, the classification accuracy of each fault category is more than 95% when no noise and environmental noise SNR = 8 are added. As the size of the environmental noise increases to SNR = 4 and SNR = 0, the accuracy of some fault categories decreases to less than 90%, and when the environmental noise increases to SNR = −2, the diagnostic accuracy of the model for Miss is only 66.15%. When the environmental noise SNR = −4, the accuracy of the model for all categories except the Chipped category is below 60%, and the data with the real category of Miss has lost the diagnostic ability. Therefore, the SrepCNN model can achieve better accuracy for classification in the noise-free and low-noise environments, and the fault recognition ability of the model decreases as the noise in the samples increases, and the model has lost the diagnostic ability for each category when SNR = −4. Fig. 9 *(b)* shows the classification of the MFCC-SrepCNN model for each fault category under different environmental noise. When no environmental noise is added and the environmental noise SNR = 8, SNR = 4, the classification accuracy of each fault category is 100%, and when the environmental noise is increased to SNR = 0, SNR = −2, SNR = −4, the classification accuracy of the fault category decreases to The classification accuracies of fault categories drop to more than 97%, 95%, and 91% when the environmental noise increases to SNR = 0, SNR = −2, and SNR = −4, respectively. When compared with the fault classification of the SrepCNN model, it can be seen that MFCC can well improve the classification ability of the neural network in noisy environments.

The aforementioned experiments demonstrate the beneficial impact of MFCC on the model's fault classification performance across diverse noise environments. To further validate the enhanced feature extraction capability of MFCC for distinct fault categories, we employ t-distributed stochastic neighbor embedding (t-SNE) to visualize the impact of various techniques. The features extracted by the model are represented by the output preceding the fully connected layer.

As shown in Fig. 10 *(a)*, when no noise is added and the environmental noise SNR = 8, the data distribution of each homogeneous fault category is compact and the boundaries between different categories are more obvious. As the size of the environmental noise increases to SNR = 4, SNR = 0, and SNR = −2, although the data distribution of each homogeneous fault category is more compact, some samples of different fault categories have been stacked. When the ambient noise increases to SNR = −4, the samples of different fault categories are stacked together and there is no clear boundary between the fault types. Therefore, the SrepCNN model can have good feature extraction ability in the noise-free and low-noise environment, and as the noise in the samples increases, the fault feature extraction ability of the model decreases, and when the SNR = −4, the model can no longer extract the fault features of different categories well. Fig. 10 *(b)* shows the extraction of each fault feature by the MFCC-SrepCNN model under different environmental noises. In each noise environment, the data distribution of each same fault category is compact, and the boundaries between different fault categories are more obvious. When the ambient noise is high, some sample data show a small amount of misalignment or stacking. After comparing with the fault feature extraction of the SrepCNN model, it shows that MFCC can well enhance the feature extraction capability of the neural network under a noisy environment.

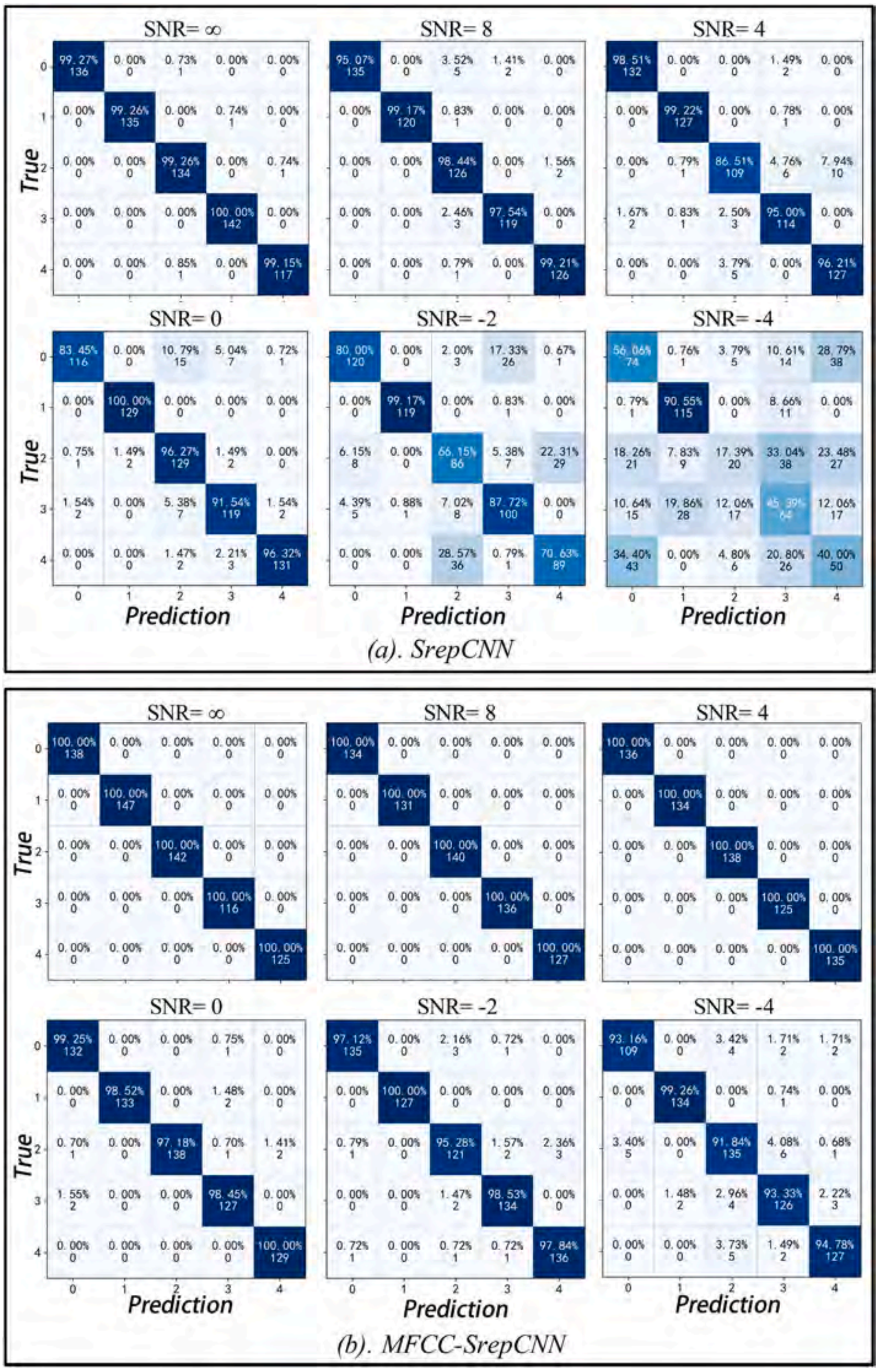


Fig. 9. Confusion matrix of SrepCNN and MFCC-SrepCNN in different noise environments.

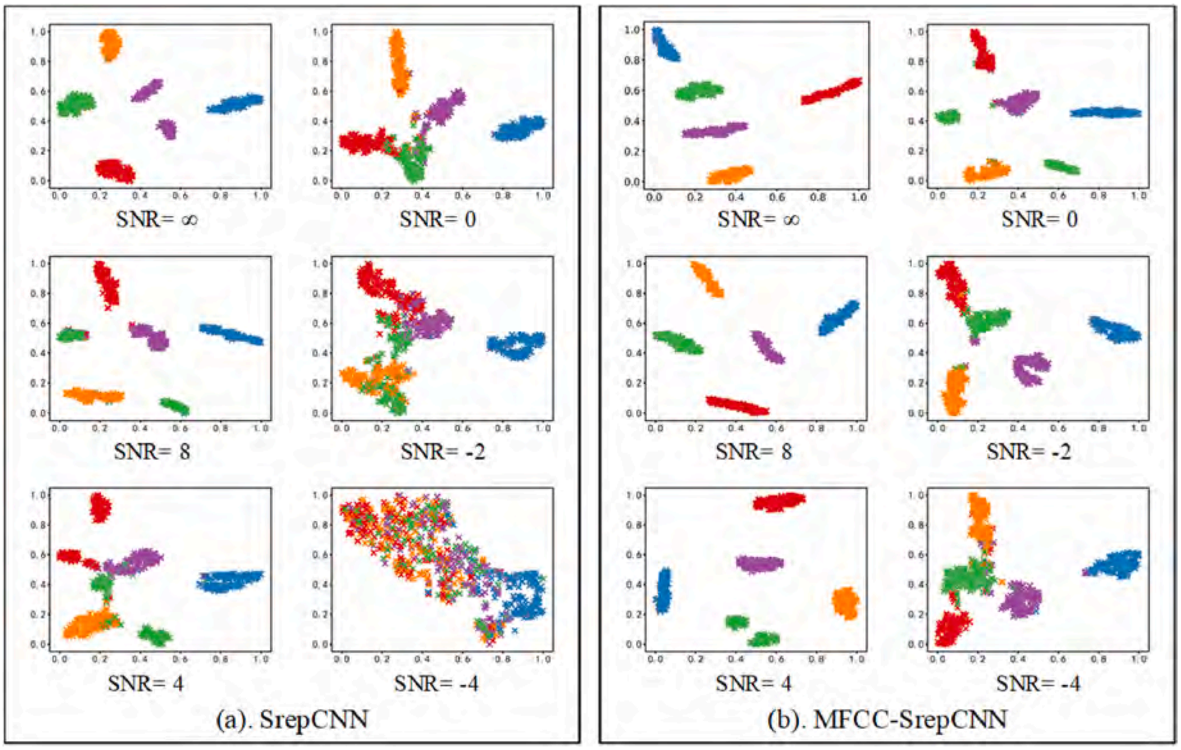


Fig. 10. t-SNE maps of SrepCNN and MFCC-SrepCNN in different noise environments.

4.3. *Case two*

4.3.1. *Experimental data description*

The experimental data for case two involving the gearbox were sourced from Huazhong University of Science and Technology and obtained using a bespoke Rotating Machinery Fault Simulation (RMFS) platform, as illustrated in Fig. 11. This platform is an all-encompassing simulator for rotating machinery faults, consisting of a motor, controller, bearing, gearbox, and brake. It can be set up with multiple faults for bearings and gears respectively and can be set up with different operating conditions by changing the speed and load. In this paper, the gearbox in this platform is selected for the case study. The gearbox type is ZDY80 parallel shaft gearbox, which can simulate 5 types of operating conditions: Normal, Broken, Miss, Root, and Pitting, and the fault location is located in the large gear, as shown in Table 3.

Acceleration sensors were affixed above the vertical of both the high-speed and low-speed end shafts of the parallel gearbox to capture acceleration signals in the x, y, and z directions at the two locations. Sampling was performed using a sliding window approach with a window overlap of 0. The window length was fixed at 3136, taking into account the rotation speed of the device and the sampling rate of the sensor. Ensuring that each sample contained at least one revolution of the gear. The data were subjected to de-singularization and normalization. The dataset was partitioned into the training, validation, and testing sets using a ratio of 7:1:2, wherein the testing dataset comprised 1135 samples.

In this case, multiple noise samples were generated by introducing random Gaussian white noise to the original experimental signal at varying SNRs of 10, 8, 6, 4, and 2. Normal and Miss states were examined as an example to observe the effect of different noise levels on signals. Fig. 12 shows that when no or little noise is added, there is a clear distinction in the sample amplitude characteristics between the two states. However, as the noise level increases, making it gradually more difficult to identify the amplitude characteristics of the two states.

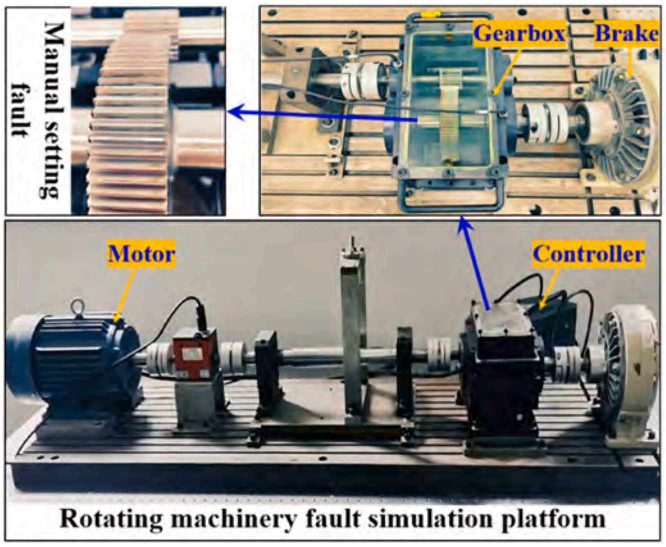


Fig. 11. Experimental equipment of RMFS.

Table 3
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 1 mm
4	Pitting	Make four pitting corrosion by an electric spark Specification of single place: five times of electric spark

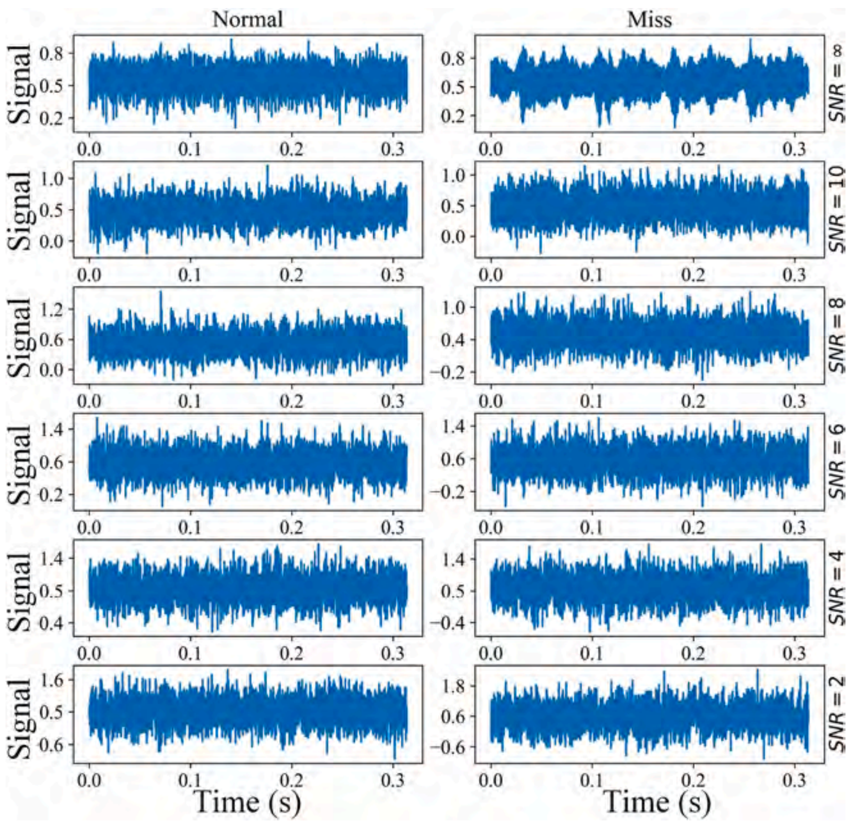


Fig. 12. The data of six different SNRs in case two.

4.3.2. *Results & discussion*

The samples are reshaped into a matrix of size [6,56,56] and input to each model respectively. The same training method as the above experimental was used for the training process.

The model was trained and validated on a computer for 50 epochs before deployment on the NVIDIA Jetson Xavier NX platform using Docker for testing. The prediction accuracy of the model in the test set is used as the performance indicator of the model. It was evaluated under normal conditions and varying levels of noise. The resulting experimental outcomes are presented in Fig. 13, where the prediction accuracy values are the mean values of three repetitions trials.

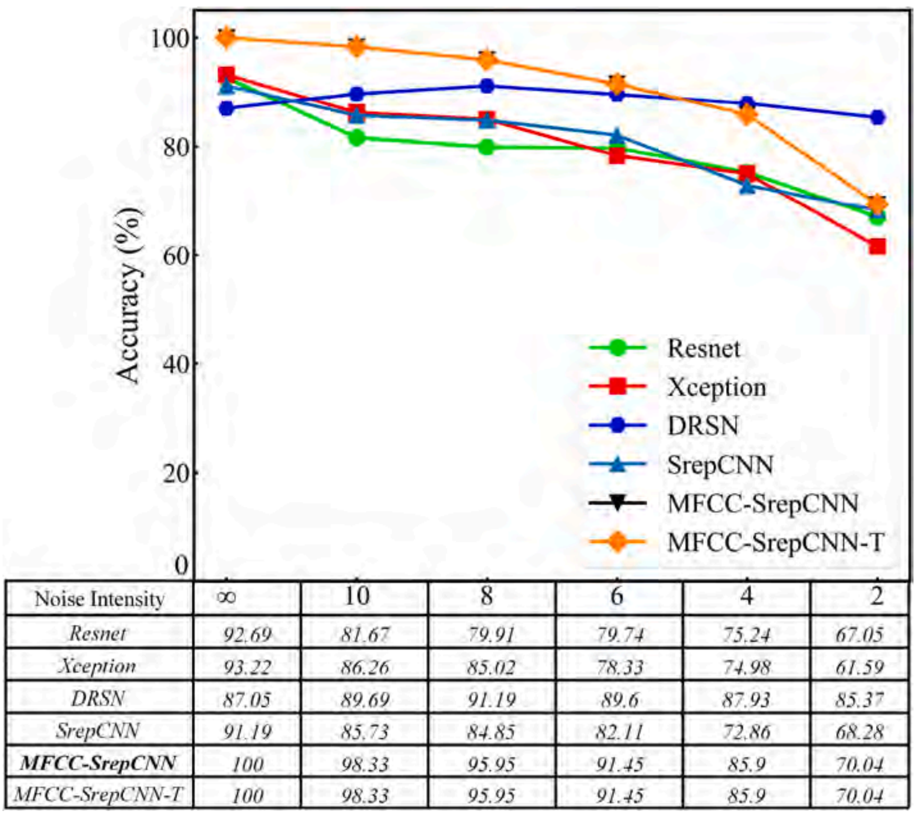
In Fig. 13, the accuracy fold of MFCC-SrepCNN overlaps with that of MFCC-SrepCNN-T, indicating that the final inference accuracy of the MFCC-SrepCNN model has not changed significantly after structural re-parameterization. In each noise environment, the accuracy fold of MFCC-SrepCNN is higher than that of SrepCNN, which can lead to an average specific inference accuracy of 9.44%. This indicates that MFCC can significantly improve the diagnostic ability of the model in the noise environment. MFCC-SrepCNN is significantly ahead of Resnet and Xception, highlighting the efficacy of the proposed model in noisy environments. Notably, although the DRSN network exhibits better diagnostic accuracy than MFCC-SrepCNN when the ambient noise intensity is SNR = 2, the model does not converge well due to the high computational effort of the DRSN and the relative complexity of the model, so the accuracy >95% does not appear even when there is no noise and low noise.

To further analyze the specific classification of the prediction results of different models under low and high noise, the confusion matrices of the test results are drawn by selecting the ambient noise intensity of SNR = 10 and SNR = 4. Fig. 14 shows the confusion matrix corresponding to the median of the results of three replicate experiments for each model under two noise environments, where the vertical indicates the actual labels of the samples, the horizontal labels indicate the prediction results, and the prediction accuracy and the corresponding number of samples are labeled in the matrix.

Comparing the confusion matrix of MFCC-SrepCNN and MFCC-SrepCNN-T in Fig. 14, it can be seen that the distribution is the same, indicating that the structural re-parameterization does not affect the final classification ability of the model. For Resnet, Xception, DRSN, and SrepCNN, the prediction accuracy of individual categories at SNR = 10 is above 65%, and when the noise increases to SNR = 4, the prediction accuracy of individual categories except DRSN decreases significantly, with the lowest being 26.67%, and the prediction in the noisy environment is poor. Meanwhile, the prediction accuracies of MFCC-SrepCNN for individual categories were above 95% and 75% at SNR = 10 and SNR = 4, indicating that MFCC can well enhance the classification capability of the model under a noisy environment.

The above experiments demonstrate the specific classification effects of different models under low and high noise. To further verify the feature extraction capability of MFCC-SrepCNN for different fault classes under low and high noise, t-SNE analysis is employed to visualize the impact of different models. Specifically, the output obtained before the fully connected layer is utilized to represent the features extracted by MFCC-SrepCNN and t-SNE plots for each of the models are generated, as depicted in Fig. 15.

In Fig. 15, regardless of whether SNR = 10 or SNR = 4, stacking occurs between the features of different faults extracted by Resnet, Xception, and SrepCNN, and the distinction between categories is not obvious, indicating that the model is less capable of extracting the features of sample faults due to the interference of environmental noise. By comparing MFCC-SrepCNN with other models, it is observed that the fault samples of the same category are more closely clustered and the boundaries between different categories are more distinct. This suggests that the incorporation of MFCC positively impacts fault classification ability. Moreover, a comparison of the feature maps extracted by MFCC-SrepCNN and MFCC-SrepCNN-T reveals that the data feature



Noise Intensity	∞	10	8	6	4	2
Resnet	*92.69*	*81.67*	*79.91*	*79.74*	*75.24*	*67.05*
Xception	*93.22*	*86.26*	*85.02*	*78.33*	*74.98*	*61.59*
DRSN	*87.05*	*89.69*	*91.19*	*89.6*	*87.93*	*85.37*
SrepCNN	*91.19*	*85.73*	*84.85*	*82.11*	*72.86*	*68.28*
MFCC-SrepCNN	*100*	*98.33*	*95.95*	*91.45*	*85.9*	*70.04*
MFCC-SrepCNN-T	*100*	*98.33*	*95.95*	*91.45*	*85.9*	*70.04*

Fig. 13. The test accuracy of different methods in different noise environments in Case Two.

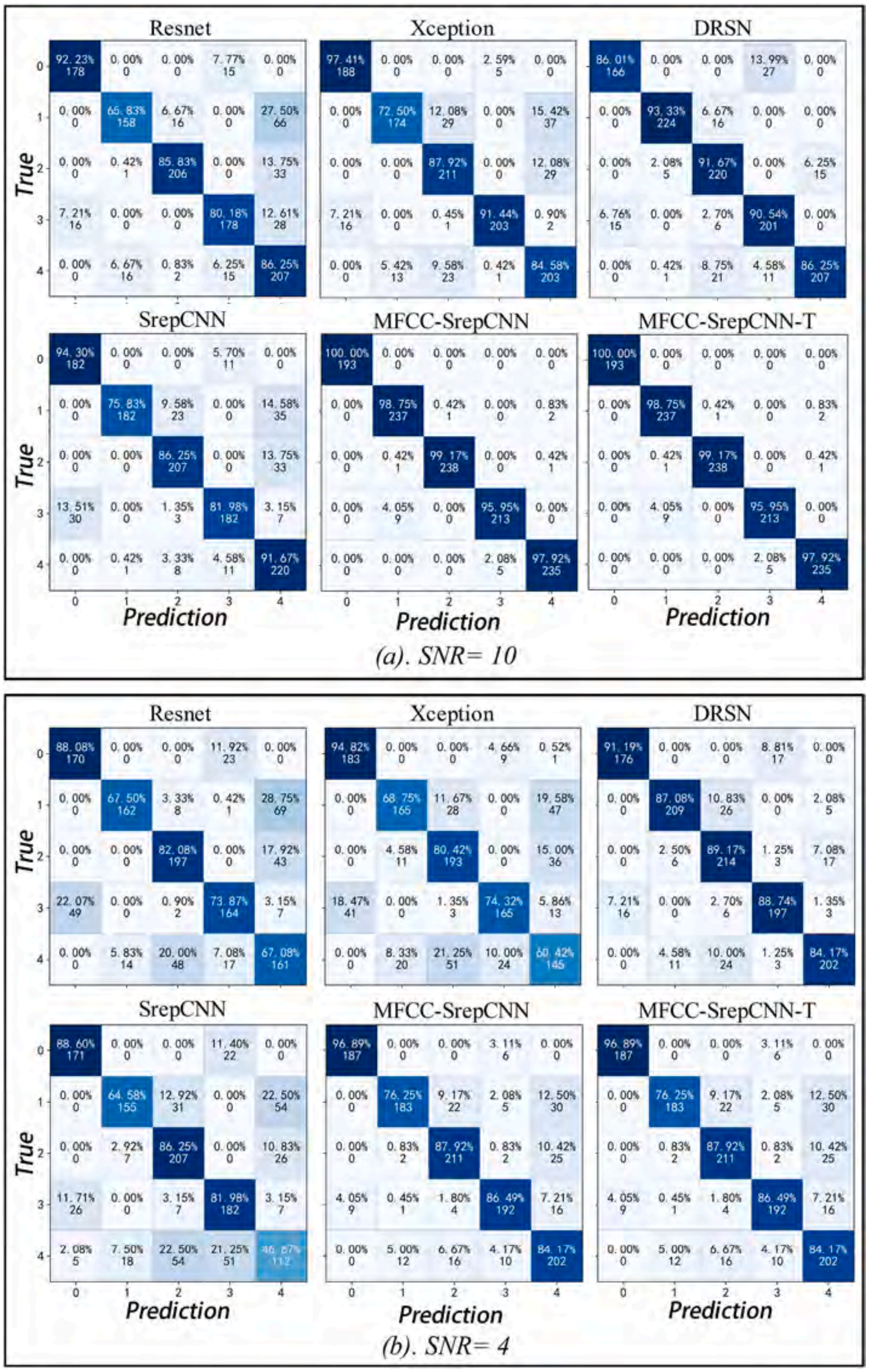


Fig. 14. Confusion matrix for each model in a noisy environment with SNR = 10 and SNR = 4.

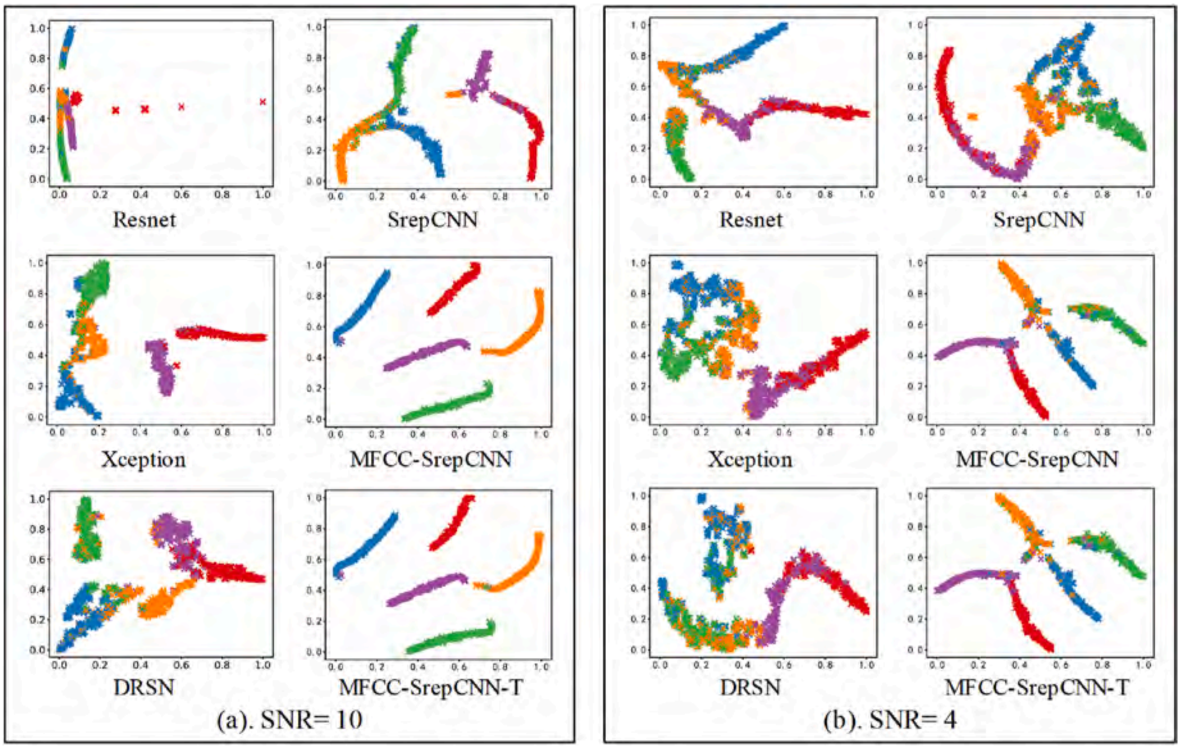


Fig. 15. t-SNE for each model in a noisy environment with SNR = 10 and SNR = 4.

distributions are identical. This further indicates that the structural re-parameterization does not affect the final feature extraction ability of the model. The DRSN in Fig. 15 shows a small amount of stacking of samples at both SNR = 10 and SNR = 4, indicating its superior feature extraction ability in noisy environments.

4.4. Model parameter analysis

In addition to evaluating the capability of the models in classification accuracy and feature extraction ability on the specific task, it is also necessary to assess their network complexity. Such parameters of the models are of great significance for considering the utility of the models in edge computing scenarios. Specifically, the following parameters of different models are compared.

A. **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.
B. **Memory:** The amount of memory required for model inference.
C. **Floating Point Operations (FLOPs):** The theoretical amount of floating point arithmetics is the amount of computation in the neural network.
D. **Inference time:** Inference time of a single sample in the edge-end device.

Table 4 presents the Params, Memory, Flops parameters, and average inference time per single sample for each mode. The statistics for Params, Memory, and Flops parameters were calculated using torchstat (A lightweight neural network analyzer based on PyTorch). The inference times were measured on the NVIDIA Jetson Xavier NX suite for the test dataset in Case 1. To exclude the effect of the code writing style on the inference time, To exclude the influence of the code writing style on the inference time, the model's computing time in the GPU is monitored via CUDA Event. Meanwhile, the GPU is warmed up with equal intensity before each speed measurement. Fig. 16 illustrates these results.

Compared with SrepCNN, the params of MFCC-SrepCNN are the

Table 4
Test results of the complexity of each model.

Methods	Params	Memory (MB)	Flops (MFlops)	Inference time (ms)
Resnet	21,300,869	2.96	275.79	27.72
Xception	20,818,061	9.59	338.62	22.19
DRSN	12,573,445	20.24	1710	18.43
SrepCNN	4,359,557	3.90	206.78	5.82
MFCC-SrepCNN	**4,359,557**	**2.65**	**144.09**	**9.45**
MFCC-SrepCNN-T	4,854,149	5.58	160.89	15.45

same as 4,359,557, but the Memory and Flops are decreased by 32.05% and 30.32%, respectively, which indicates that MFCC does not change the total number of parameters of the network, and the size of MFCC feature matrix is reduced compared with the original samples, which reduces the size of individual samples input to SrepCNN, which in turn reduces the amount of computation, resulting in less Memory for the MFCC-SrepCNN.

Compared with MFCC-SrepCNN-T, the Params, Memory, and Flops of MFCC-SrepCNN decreased by 26.67%, 52.51%, and 10.44%, respectively, indicating that the structural re-parameterization changed the network from multi-branch to single-way, which reduced the total number of FLOPs and parameters of the neural network, which in turn led to less Memory.

Meanwhile, the relevant network parameters of MFCC-SrepCNN are all significantly smaller than those of Resnet and Xception, indicating that the MFCC-SrepCNN model is less complex, easier to train, and has less hardware overhead at runtime. Although according to the previous paper, DRSN can be more stable in fault diagnosis accuracy under noisy environments, the Memory and Flops of this network are 20.24 MB and 1.71 GFlops, respectively, resulting in the large hardware overhead even when performing inference operations, and thus is not suitable for lightweight deployment in harsh environments full of disturbances.

The convolution kernels of MFCC-SrepCNN models are all 3*3 conv,

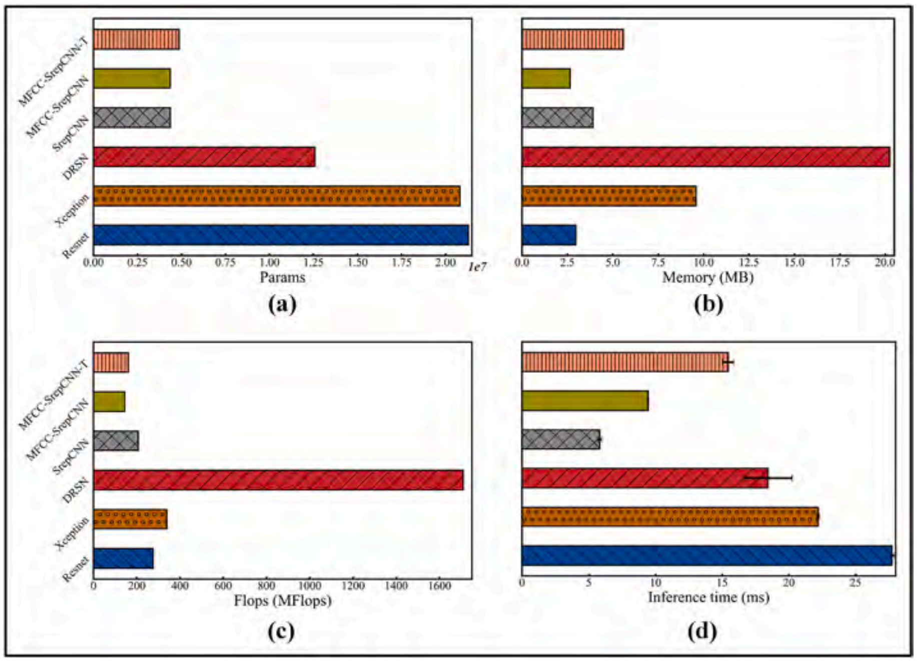


Fig. 16. Test results of the complexity of each model.

which can effectively improve the inference speed of the models on edge devices, so the effect of structural re-parameterization on the inference speed of the models is worth exploring. In three iterations of case one, the GPU runtime of each sample for inference is recorded, and the mean value was taken as the result of the test. Based on the results of 18 tests for each model, a visual error histogram of the inference speed of a single sample for different models was drawn. Analyzing Fig. 16 and Tables 4, it can be seen that the inference speed of the MFCC-SrepCNN model decreases from 15.45 ms to 9.45 ms than the MFCC-SrepCNN-T model. Meanwhile, compared with Resnet, Xception, and DRSN models, MFCC-SrepCNN has a significant advantage in model inference speed. This indicates that structural re-parameterization can enhance the inference speed of the method. Which can get the prediction results faster on the edge device, and then assist the monitoring system of the gearbox to make a timely response.

5. Conclusion

This study introduces a lightweight convolutional neural network-based approach for gearbox fault diagnosis in edge computing scenarios. The proposed method extracts the MFCC feature matrix from signals, effectively suppressing noise interference and improving diagnostic accuracy. To achieve improved inference speed and reduced hardware overhead on edge computing devices, we apply the principle of structural re-parameterization. By transforming the model from multiple branches during training to a single branch for inference, while maintaining its diagnostic capability. We conduct validation experiments on a public dataset and a custom test device using the NVIDIA Jetson Xavier NX suite as the edge computing platform. According to the experiment, after extracting the MFCC feature matrix, the average diagnostic accuracy rate in the noisy environment of the presented methodology is improved by 12.22% and 9.44%, respectively. After structural re-parameterization, the Memory of the model decreases by 52.58%, and the inference speed is increased by 38.83%. The results demonstrate that our proposed methodology offers better fault diagnosis capability, a smaller memory footprint, and faster inference speed in noisy environments.

Although, the proposed method achieves good diagnostic results while being lightweight. The experiments are conducted based on a large amount of labeled data for model training and sufficient data for various fault types, in the case study. And under the actual engineering application, there will be missing data labels or data imbalance among fault types. Therefore, we should investigate relevant semi-supervised or self-supervised learning to resolve the above issues in future work. In addition, the proposed method achieves lightweight in a way that can be effective in a specific network structure. The method has some limitations. Therefore, future work is expected to achieve a lightweight for generic models (rather than a specific network structure), through knowledge distillation, network pruning, etc.

Funding

This research has been partially supported by the National Key Research and Development Program of China (under Grant, 2022YFC2204700); and the Fundamental Research Funds for the Central Universities (under Grant YCJJ202201006).

CRediT authorship contribution statement

Yanzhi Wang: Conceptualization, Methodology, Software, Writing – original draft. **Jinhong Wu:** Validation, Writing – review & editing. **Ziyang Yu:** Investigation, Resources. **Jiexiang Hu:** Visualization, Data curation. **Qi Zhou:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Belongie, S., Wilber, M., Veit, A., 2016. Residual networks behave like ensembles of relatively shallow networks. In: 30th Conference on Neural Information Processing Systems (NIPS)Barcelona, SPAIN.

Chen, J.L., Li, Z.P., Pan, J., Chen, G.G., Zi, Y.Y., Yuan, J., Chen, B.Q., He, Z.J., 2016. Wavelet Transform Based on Inner Product in Fault Diagnosis of Rotating Machinery: A Review, Mechanical Systems, and Signal Processing, pp. 1–35. https://doi.org/10.1016/j.ymssp.2015.08.023, 70-71.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E., 2014. Cudnn: Efficient Primitives for Deep Learning arXiv preprint arXiv:1410.0759.

Chiang, M., Zhang, T., 2016. Fog and IoT: an overview of research opportunities. IEEE Internet Things J. 3, 854–864. https://doi.org/10.1109/jiot.2016.2584538.

Chollet, F., 2017. Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258.

Dai, C., Liu, X., Li, Z., Chen, M.-Y., 2021. A tucker decomposition based knowledge distillation for intelligent edge applications. Appl. Soft Comput. 101 https://doi.org/10.1016/j.asoc.2020.107051.

Ding, X.H., Guo, Y.C., Ding, G.G., Han, J.G., 2019. Ieee, ACNet: strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks. In: IEEE/CVF International Conference on Computer Vision (ICCV)Seoul, SOUTH KOREA, pp. 1911–1920.

Ding, X.H., Zhang, X.Y., Han, J.G., Ding, G.G., 2021a. S.O.C. Ieee comp, diverse branch block: building a convolution as an inception-like unit. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)Electr Network, pp. 10881–10890.

Ding, X.H., Zhang, X.Y., Ma, N.N., Han, J.G., Ding, G.G., Sun, J., 2021b. S.O.C. Ieee comp, RepVGG: making VGG-style ConvNets great again. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)Electr Network, pp. 13728–13737.

Errandonea, I., Ciáurriz, P., Alvarado, U., Beltrán, S., Arrizabalaga, S., 2023. Edge intelligence-based proposal for onboard catenary stagger amplitude diagnosis. Comput. Ind. 144 https://doi.org/10.1016/j.compind.2022.103781.

Gao, C., Rios-Navarro, A., Chen, X., Liu, S.C., Delbruck, T., 2020. EdgeDRNN: recurrent neural network accelerator for edge inference. Ieee Journal on Emerging and Selected Topics in Circuits and Systems 10, 419–432. https://doi.org/10.1109/jetcas.2020.3040300.

Gill, S.S., Xu, M.X., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A., Golec, M., Stankovski, V., Wu, H.M., Abraham, A., Singh, M., Mehta, H., Ghosh, S.K., Baker, T., Parlikad, A.K., Lutfiyya, H., Kanhere, S.S., Sakellariou, R., Dustdar, S., Rana, O., Brandic, I., Uhlig, S., 2022. AI for next generation computing: emerging trends and future directions. Internet of Things 19. https://doi.org/10.1016/j.iot.2022.100514.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Hewa, T., Braeken, A., Liyanage, M., Ylianttila, M., 2022. Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing. IEEE Trans. Ind. Inf. 18, 7174–7185. https://doi.org/10.1109/tii.2022.3140792.

Huang, Y., Qiao, X., Tang, J., Ren, P., Liu, L., Pu, C., Chen, J., 2023. An integrated cloud-edge-device adaptive deep learning service for cross-platform web. IEEE Trans. Mobile Comput. 22, 1950–1967. https://doi.org/10.1109/tmc.2021.3122279.

Huo, Z.Q., Martinez-Garcia, M., Zhang, Y., Shu, L., 2022. A multisensor information fusion method for high-reliability fault diagnosis of rotating machinery. IEEE Trans. Instrum. Meas. 71 https://doi.org/10.1109/tim.2021.3132051.

Huong, T.T., Bac, T.P., Long, D.M., Luong, T.D., Dan, N.M., Quang, L.A., Cong, L.T., Thang, B.D., Tran, K.P., 2021. Detecting cyberattacks using anomaly detection in industrial control systems: a Federated Learning approach. Comput. Ind. 132 https://doi.org/10.1016/j.compind.2021.103509.

Jing, T., Tian, X., Hu, H., Ma, L., 2022. Deep learning-based cloud–edge collaboration framework for remaining useful life prediction of machinery. IEEE Trans. Ind. Inf. 18, 7208–7218. https://doi.org/10.1109/tii.2021.3138510.

Khalil, R.A., Saeed, N., Masood, M., Fard, Y.M., Alouini, M.S., Al-Naffouri, T.Y., 2021. Deep learning in the industrial internet of things: potentials, challenges, and emerging applications. IEEE Internet Things J. 8, 11016–11040. https://doi.org/10.1109/jiot.2021.3051414.

Kong, X.J., Wu, Y.H., Wang, H., Xia, F., 2022. Edge computing for internet of everything: a survey. IEEE Internet Things J. 9, 23472–23485. https://doi.org/10.1109/jiot.2022.3200431.

Kumar, A., Gandhi, C.P., Zhou, Y., Vashishtha, G., Kumar, R., Xiang, J., 2020. Improved CNN for the diagnosis of engine defects of 2-wheeler vehicle using wavelet synchro-squeezed transform (WSST). Knowl. Base Syst. 208 https://doi.org/10.1016/j.knosys.2020.106453.

Kumar, A., Parkash, C., Vashishtha, G., Tang, H., Kundu, P., Xiang, J., 2022. State-space modeling and novel entropy-based health indicator for dynamic degradation monitoring of rolling element bearing. Reliab. Eng. Syst. Saf. 221 https://doi.org/10.1016/j.ress.2022.108356.

Kumar, A., Vashishtha, G., Gandhi, C.P., Tang, H., Xiang, J., 2021. Tacho-less sparse CNN to detect defects in rotor-bearing systems at varying speed. Eng. Appl. Artif. Intell. 104 https://doi.org/10.1016/j.engappai.2021.104401.

Lavin, A., Gray, S., 2016. Fast algorithms for convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4013–4021.

Lei, Y.G., Lin, J., Zuo, M.J., He, Z.J., 2014. Condition monitoring and fault diagnosis of planetary gearboxes: a review. Measurement 48, 292–305. https://doi.org/10.1016/j.measurement.2013.11.012.

Li, G.Q., Wu, J., Deng, C., Chen, Z.Y., 2022. Parallel multi-fusion convolutional neural networks based fault diagnosis of rotating machinery under noisy environments. ISA Trans. 128, 545–555. https://doi.org/10.1016/j.isatra.2021.10.023.

Li, H.F., Hu, G.Z., Li, J.Q., Zhou, M.C., 2022. Intelligent Fault diagnosis for large-scale rotating machines using binarized deep neural networks and random forests. IEEE Trans. Autom. Sci. Eng. 19, 1109–1119. https://doi.org/10.1109/tase.2020.3048056.

Liang, P., Wang, W., Yuan, X., Liu, S., Zhang, L., Cheng, Y., 2022. Intelligent fault diagnosis of rolling bearing based on wavelet transform and improved ResNet under noisy labels and environment. Eng. Appl. Artif. Intell. 115 https://doi.org/10.1016/j.engappai.2022.105269.

McDonald, G.L., Zhao, Q., Zuo, M.J., 2012. Maximum correlated Kurtosis deconvolution and application on gear tooth chip fault detection. Mech. Syst. Signal Process. 33, 237–255. https://doi.org/10.1016/j.ymssp.2012.06.010.

Qian, G., Lu, S., Pan, D., Tang, H., Liu, Y., Wang, Q., 2019. Edge computing: a promising framework for real-time fault diagnosis and dynamic control of rotating machines using multi-sensor data. IEEE Sensor. J. 19, 4211–4220. https://doi.org/10.1109/jsen.2019.2899396.

Qizhao, W., Li, Q., Wang, K., Wang, H., Peng, Z., 2021. Efficient federated learning for fault diagnosis in industrial cloud-edge computing, Computing. Archives for Informatics and Numerical Computation 103, 2319–2337.

Ren, L., Jia, Z., Wang, T., Ma, Y., Wang, L., 2022. LM-CNN: a cloud-edge collaborative method for adaptive fault diagnosis with label sampling space enlarging. IEEE Trans. Ind. Inf. 18, 9057–9067. https://doi.org/10.1109/tii.2022.3180389.

Samie, F., Bauer, L., Henkel, J., 2019. From cloud down to things: an overview of machine learning in internet of things. IEEE Internet Things J. 6, 4921–4934. https://doi.org/10.1109/jiot.2019.2893866.

Shao, H., Xia, M., Han, G., Zhang, Y., Wan, J., 2021. Intelligent Fault diagnosis of rotor-bearing system under varying working conditions with modified transfer convolutional neural network and thermal images. IEEE Trans. Ind. Inf. 17, 3488–3496. https://doi.org/10.1109/tii.2020.3005965.

Shao, S.Y., McAleer, S., Yan, R.Q., Baldi, P., 2019. Highly accurate machine fault diagnosis using deep transfer learning. IEEE Trans. Ind. Inf. 15, 2446–2455. https://doi.org/10.1109/tii.2018.2864759.

Shi, W.S., Cao, J., Zhang, Q., Li, Y.H.Z., Xu, L.Y., 2016. Edge computing: vision and challenges. IEEE Internet Things J. 3, 637–646. https://doi.org/10.1109/jiot.2016.2579198.

Tang, X., Xu, Y., Sun, X., Liu, Y., Jia, Y., Gu, F., Ball, A.D., 2022. Intelligent fault diagnosis of helical gearboxes with compressive sensing based non-contact measurements. ISA Trans. https://doi.org/10.1016/j.isatra.2022.07.020.

Wang, H., Xu, J.W., Sun, C., Yan, R.Q., Chen, X.F., 2022. Intelligent Fault diagnosis for planetary gearbox using time-frequency representation and deep reinforcement learning. Ieee-Asme Transactions on Mechatronics 27, 985–998. https://doi.org/10.1109/tmech.2021.3076775.

Wang, Y.R., Jin, Q., Sun, G.D., Sun, C.F., 2019. Planetary gearbox fault feature learning using conditional variational neural networks under noise environment. Knowl. Base Syst. 163, 438–449. https://doi.org/10.1016/j.knosys.2018.09.005.

Wu, P.L., Nie, X.Y., Xie, G., 2021. Multi-sensor signal fusion for a compound fault diagnosis method with strong generalization and noise-tolerant performance. Meas. Sci. Technol. 32 https://doi.org/10.1088/1361-6501/abc6e3.

Yan, H., Bai, H., Zhan, X., Wu, Z., Wen, L., Jia, X., 2022. Combination of vmd mapping MFCC and lstm: a new acoustic fault diagnosis method of diesel engine. Sensors 22. https://doi.org/10.3390/s22218325.

Yao, D., Liu, H., Yang, J., Li, X., 2020. A lightweight neural network with strong robustness for bearing fault diagnosis. Measurement 159. https://doi.org/10.1016/j.measurement.2020.107756.

Yu, F., Cui, L., Wang, P., Han, C., Huang, R., Huang, X., 2021. EasiEdge: a novel global deep neural networks pruning method for efficient edge computing. IEEE Internet Things J. 8, 1259–1271. https://doi.org/10.1109/jiot.2020.3034925.

Yu, X.L., Yang, Y., He, Q.B., Du, M.G., Peng, Z.K., 2022. Multiple frequency modulation components detection and decomposition for rotary machine fault diagnosis. IEEE Trans. Instrum. Meas. 71 https://doi.org/10.1109/tim.2021.3134334.

Zeng, Y.L., Song, C.Y., Ge, T.J., Zhang, Y., 2022. Reduction of large-scale graphs: effective edge shedding at a controllable ratio under resource constraints. Knowl. Base Syst. 240 https://doi.org/10.1016/j.knosys.2022.108126.

Zhang, K.L., Huang, W., Hou, X.Y., Xu, J.H., Su, R.D., Xu, H.Y., 2021. a fault diagnosis and visualization method for high-speed train based on edge and cloud collaboration. Applied Sciences-Basel 11. https://doi.org/10.3390/app11031251.

Zhang, L., Fan, Q., Lin, J., Zhang, Z., Yan, X., Li, C., 2023. A nearly end-to-end deep learning approach to fault diagnosis of wind turbine gearboxes under nonstationary conditions. Eng. Appl. Artif. Intell. 119 https://doi.org/10.1016/j.engappai.2022.105735.

Zhang, Z.Z., Li, S.M., Wang, J.R., Xin, Y., An, Z.H., Jiang, X.X., 2020. Enhanced sparse filtering with strong noise adaptability and its application on rotating machinery fault diagnosis. Neurocomputing 398, 31–44. https://doi.org/10.1016/j.neucom.2020.02.042.

Zhao, M., Zhong, S., Fu, X., Tang, B., Pecht, M., 2019. Deep residual shrinkage networks for fault diagnosis. IEEE Trans. Ind. Inf. 16, 4681–4690.

Zhao, Y., Yin, Y., Gui, G., 2020. Lightweight deep learning based intelligent edge surveillance techniques. Ieee Transactions on Cognitive Communications and Networking 6, 1146–1154. https://doi.org/10.1109/tccn.2020.2999479.