Automobile bearing fault diagnosis method based on Inverted Residuals and Linear Bottlenecks

HaiyueTian tian197@purdue.edu

Runjia Du du187@purdue.edu

Chieh-En Li mailto:li3261@purdue.edu

Abstract

Our team project aims at the problem that the amount of parameters and calculations is large by using convolutional neural network to analyze the automobile bearing fault condition, our team project proposes a method for automobile bearing fault diagnosis based on inverted residuals with linear bottlenecks (IRLB). First of all, the collected signals are used to enhance the bearing data of the driving end and the fan section by the overlapping sampling method to obtain 2*512 samples. Subsequently, the obtained sample is transformed into a 32*32 two-dimensional matrix as the input feature of IRLB, and the fault diagnosis and analysis are performed. By testing and comparing five kinds of optimizers, the ADAM optimizer with the best performance is selected as the optimizer of this model. Finally, the sample database was tested by IRLB. The results showed that the test accuracy of IRLB reached 99.4%, and the amount of calculation and parameters was only 2.7M and 0.92Mb, which was far less than the calculation amounts of 8.8M and 2008.6M, and the number of parameters convolutional neural networks with similar structures and deep convolutional neural networks of 3.2Mb and 223.5Mb. Apart from the machine learning part, we also implement real-time sensing, diagnosis, and edge device communication via Bluetooth 4.0.

Keywords: Automobile bearing fault diagnosis, Inverted Residuals with Linear Bottlenecks, the Overlapping Sampling Method, ADAM Optimizer, Exponential Decay Learning Rate, Convolutional Neural Network, Deep Convolutional Neural Network, Internet of Things, Edge Computing

0 Introduction

Nowadays, cars have become an indispensable part of people's life. It is of great significance for passengers whether cars can be driven safely. Rolling bearings appear in many parts of cars. If the rolling bearing failure will have a major safety hazard to the car. Therefore, it is crucial to study the fault diagnosis of rolling bearings for safe driving.

In recent years, the application of convolutional neural networks in fault diagnosis of rolling bearings has been widely used. The convolutional neural network can effectively extract fault feature information and classify diagnosis. Levent E et al. [1] extracted features and classified them through one-dimensional convolutional neural network. Nian Long GU et al. [2] proposed a time-multiple feature extraction method based on empirical modal decomposition (EMD) and combined it with convolutional neural network (CNNs) to establish a fault diagnosis method based on EMD-CNNS. Li S et al. [3] proposed a new bearing fault diagnosis algorithm IDSCNN based on integrated deep convolutional neural network and an improved evidence fusion algorithm based on the Dempster-Shafer theory. The convolutional neural network takes as input the ROOT mean square (RMS) graph of FFT (Fast Fourier Transform) characteristics of vibration signals from both sensors. The improved D-S evidence theory is realized by the evidence distance matrix and the improved Gini index. Zhang B et al. [4] proposed adversarial adaptive one-dimensional CNN (A2CNN) to solve this problem. It consists of a source feature extractor, target feature extractor, label classifier, and domain discriminator. In the training stage, the layer between the source feature extractor and the target feature extractor is partially unwrapped to give consideration to training efficiency and domain adaptation. It solves the problem of domain invariability and achieves high precision under different working conditions. Tonglin Gao et al. [5] used deep convolutional neural network to improve the accuracy to 100%. However, due to the huge amount of computation and reference, it is difficult to implement the convolutional neural network in the actual industrial

environment. Thus, our project presents a fault diagnosis method for automobile bearing based on inverted residuals with linear bottlenecks. Additionally, we also introduce the Internet of Things in our project to make expedite fault detection and we also manifest the imperative of edge devices communications through our lab-environment implementation. Our proposed method can be concluded as Fig. 0.

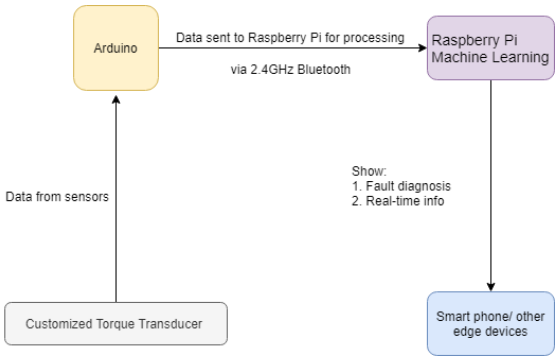

Fig. 0. The structure of edge device communication and their role

1 Inverted Residuals with Linear Bottlenecks

Inverted residuals with linear bottlenecks (IRLB) are a lightweight convolutional neural grid structure, which has the characteristics of small calculation and parameter amount and is more broadband friendly. This network structure has certain applications in the field of image recognition and has achieved good results [7]. IRLB is mainly composed of depthwise convolutional layer (DW) and pointwise convolutional layer (PW). Its main structure is shown in

Fig. 1. After the input feature map enters the network, it first passes through the PW layer to upgrade its dimension, then enters the DW layer to extract features, and finally passes through the PW layer to reduce its dimensionality. If the size of the input feature map and the output feature map are equal, the two are added together and output.

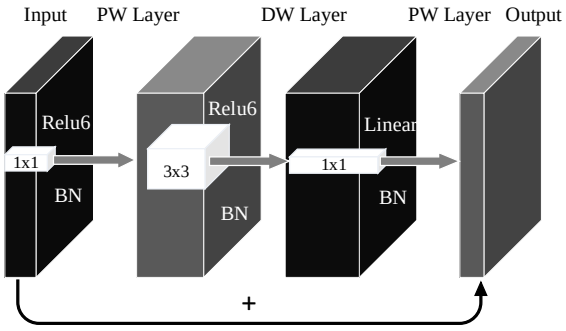

Fig. 1. The structure of IRLB

The DW and the PW layer are the core structures in IRLB, which can effectively compress the amount of calculation to one-ninth times the ordinary convolutional layer [7]. As shown in Fig. 2, unlike conventional convolutions, a single convolution kernel in DW is only responsible for a single channel, and a single channel can only be convolved by a single convolution kernel. DW in this article means that the input feature map is subjected to the DW operation without bias, then activate the output by the Relu or Relu6 activation function, and finally normalize the output through the BN Layer.

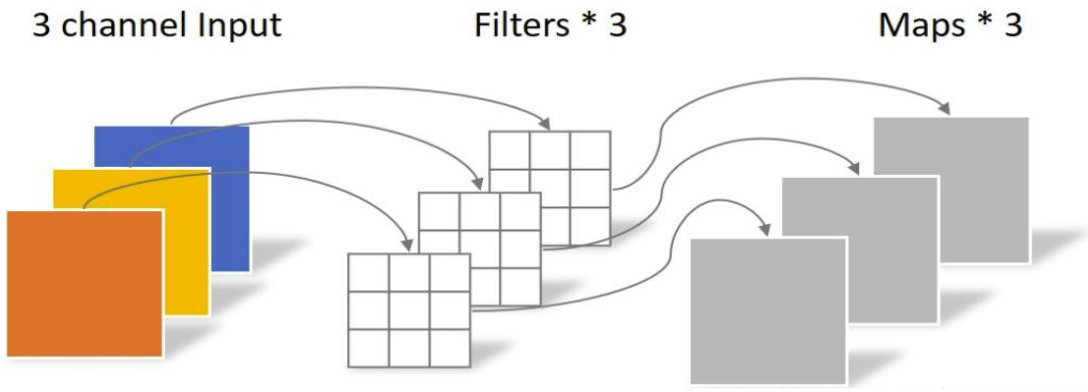

Fig. 2. Schematic diagram of DW

Since after DW, the number of output feature maps is the same as the number of input feature maps, the feature maps cannot be expanded, and the feature information of the same position on different channels is not used, so PW is required to combine new feature maps.

PW is very similar to ordinary convolution, but the difference is that the size of the convolution kernel of PW is 1*1*M, where M represents the depth of the input feature map. As shown in Fig. 3, PW effectively utilizes spatial information and extracts multiple different deep feature maps from the same feature map with shallow features. PW in this article means that the input feature map is first through PW operation without bias, then through the Relu, Relu6, or linear activation function, and finally through the BN layer to obtain the final output feature map.

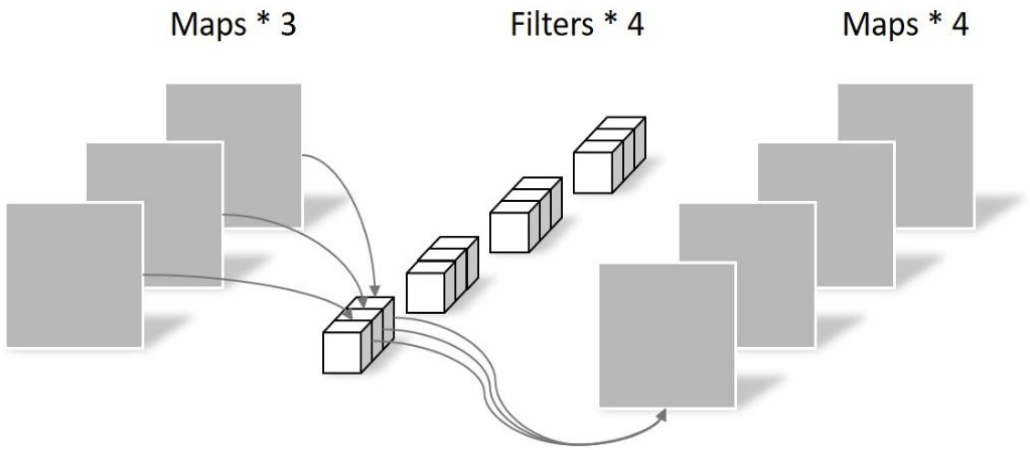

Fig. 3. Schematic diagram of PW

The calculation formula of DW and PW is shown in (1):

$$a_{d,i,j} = \sum_{d=0}^{D-1} \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} w_{d,m,n} * x_{d,i+m,j+n} \tag{1}$$

In (1), $a_{d,i,j}$ represent the pixel of the output feature map; D represents the number of channels; K represents the size of the convolution kernel; $w_{d,m,n}$ represents the weight of the convolution kernel; $x_{d,i+m,j+n}$ represents the pixel of the input feature map.

The Relu6 activation function is used in IRLB. The Relu6 activation function is a variant of the Relu activation function. When the input is less than 0, the output is 0; when the input is between 0~6, the output is the input itself; when the input is greater than 6, the output is 6, and the mathematical expression is shown in (2). The fitting curve is shown in Figure 5.

$$\mathrm{Relu6}(x) = \min(\max(x, 0), 6) \tag{2}$$

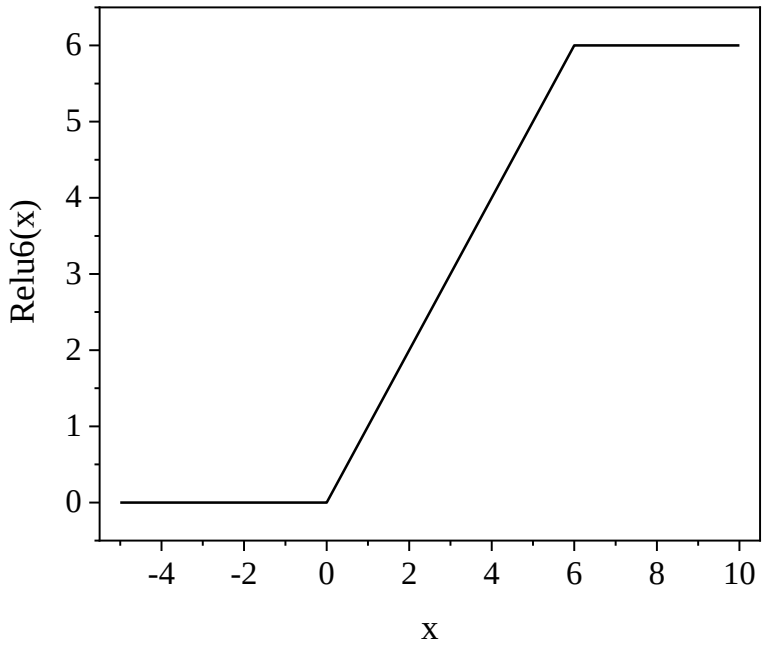

Fig. 4. Relu6 activation function

The Batch Normalization layer (BN) increases the correlation and coupling between the layers during training so that the weights do not need to be transitioned to adapt to the distribution changes caused by the different training batches, making the model more stable. While making the one-dimensional or two-dimensional features of each training batch meet the mean value of 0 and the standard deviation of 1, BN adds trainable parameters γ and β as transformation reconstruction to ensure that the feature distribution is not damaged. Mathematical expressions are shown in (3) and(4).

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \lambda}} \tag{3}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{4}$$

In(3), x_i represents the i-th input feature value; μ_B represents the mean of the training batch; σ_B^2 represents the standard deviation of the training batch. In(4), represents the output characteristic value.

The main function of the pooling layer is to down-sampling, reducing the size of the feature map, and reducing the amount of computation without losing too much feature information. Usually, There are two kinds of pooling layers, the maximum pooling layer, and the average pooling layer. Pooling layers take the maximum or average value of one or multiple adjacent pixel values as a single pixel of the output feature map. We select the maximum pooling layer, and the formula is shown in (5) in our project.

$$y_{i,j,k} = \max\left(x_{m,n,k} : i \le m \le i+p, j \le n \le j+q\right) \tag{5}$$

In the formula, p and q represent the width and height of the pooling window, and the output is the maximum value in the adjacent p rows and q columns in the input feature map x. This operation greatly reduces the amount of calculation for subsequent use of two-dimensional images.

2 Bearing fault diagnosis experiment

2.1 Experimental environment

This experiment was completed on the Ubuntu system. The basic data analysis and processing were processed by Matlab. The deep learning part, including the production of TFRecords, model establishment, and comparison experiments, were all implemented based on the Python environment and TensorFlow framework.

The experimental data of this subject are all from Case Western Reserve University [8]. As shown in Fig. 5, the test platform consists of four parts, including a 1.5kw (2HP) engine (shown on the left), torque sensor/decoder (shown in the figure), dynamometer (shown on the right), electronic controller, etc. The bearing to be tested is connected with the engine shaft, the drive end SKF6205, the sampling frequency is 12 kHz and 48 kHz, and the fan end SKF6203, the sampling frequency is 12 kHz. The damage in the experiment is single point damage of EDM. The damage diameters of SKF bearings are 0.1778, 0.3556, and 0.5334mm. The damage diameters of NTN bearings are 0.7112 and 1.016mm. The experiment collects samples of normal bearings, outer race fault, inner race fault, and ball fault.

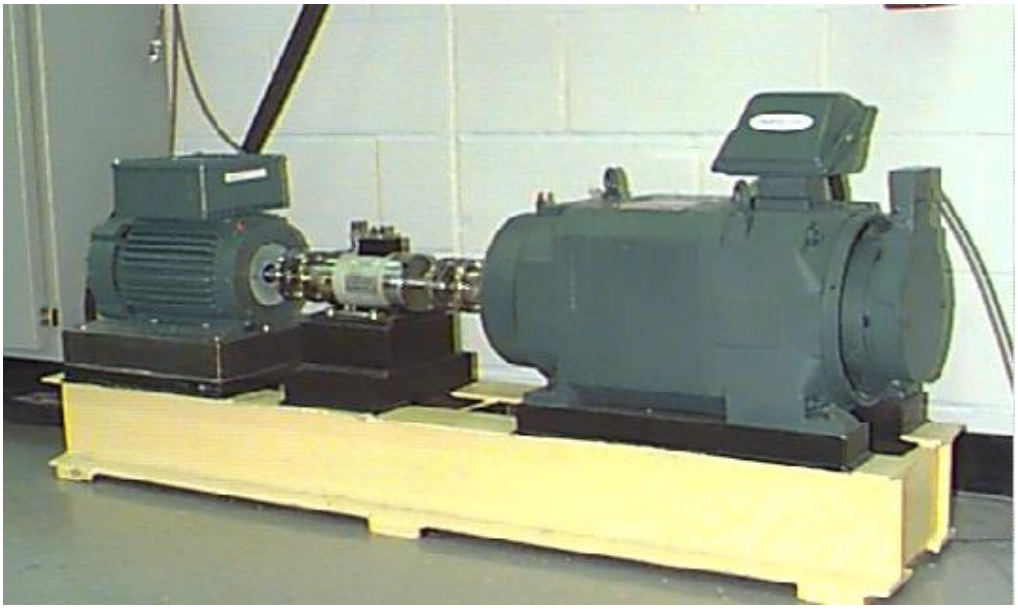

Fig. 5. The experiment platform

2.2 The overlap sampling method

Enhancing the robustness of deep learning can achieve better performance. The optimal solution is to enhance the robustness of the model by increasing the number of samples for training, that is, to enhance the data set. Because the original vibration signal of the bearing has a certain periodicity and time sequence, the data augmentation used in the computer field cannot be completely suitable for fault diagnosis. In this topic, the overlap sampling method [6] will be used to increase the data set samples. When collecting samples from the original vibration signal, there is an overlap between the previous signal and the next signal.

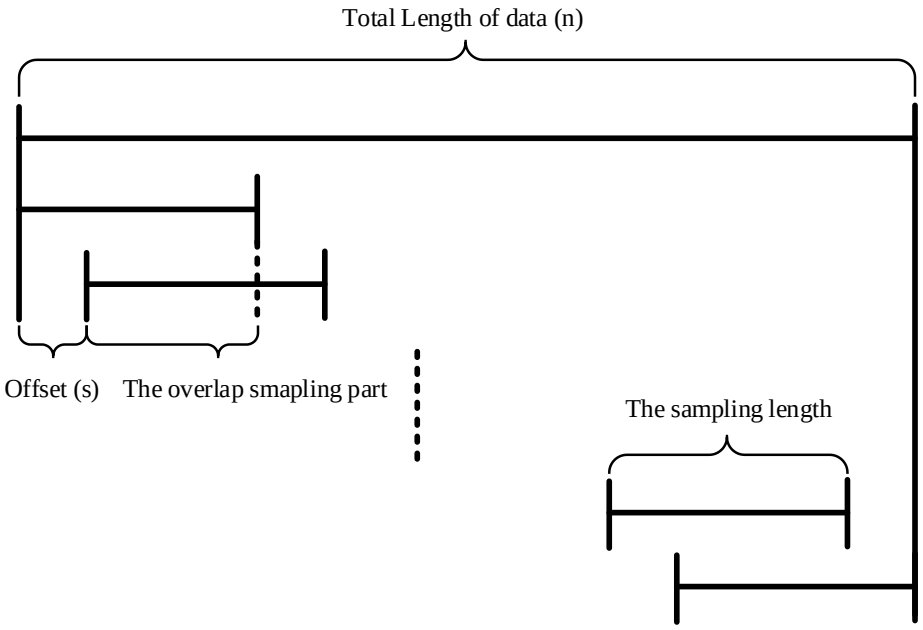

Fig. 6. The overlap sampling method

As shown in Fig. 6, assuming that a piece of the original signal has n data points, the length of the signal collected each time is l data points, and the offset is s data points. According to (6), a total of y training samples can be obtained. The specific situation is shown in Table 1, which can well meet the needs of model training.

$$y = \frac{n-l}{s} + 1 \tag{6}$$

Table 1

Overlapping and non-overlapping sampling analysis comparison

Sampling method	Total data	Single sample	Stride	Total number of samples
Non-overlap sampling	2*10240	2*512	512	20
Overlap sampling	2*10240	2*512	5	2048

2.3 Divide training set and test set

This subject will use 48 kHz data from Case Western Reserve University as the experimental object and divide it into four labels, namely normal bearings, outer rate fault bearings, inner rate fault bearings, and ball fault bearings. 70% of the data is used for training, 30% of the data is used for testing, and both use data augmentation to expand the training set samples and test set samples. The data set is saved in TFRecord format, as shown in Table 2.

Table 2

The number of samples in training set and testing set

Data set	Normal	Inner race fault	Outer race fault	Ball fault
Training set	23740	20318	27689	20323
Testing set	10153	8642	11783	8651

2.4 The structure of model

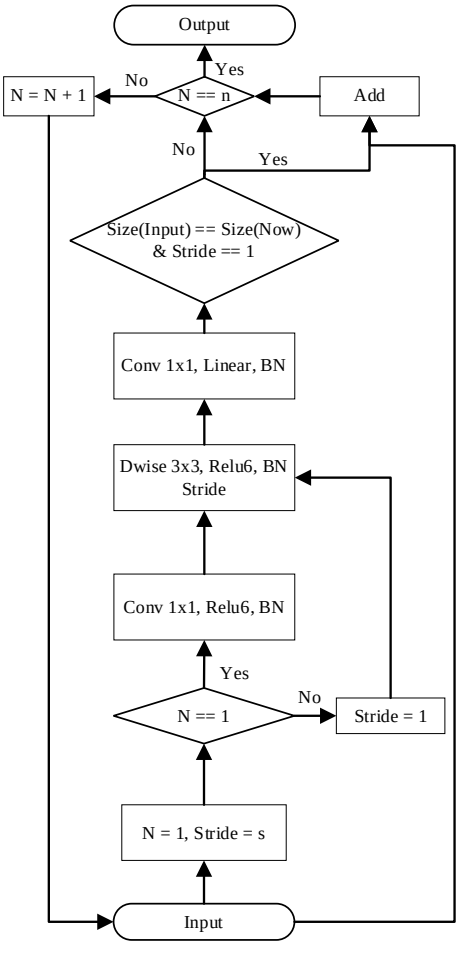

Fig. 7. The flow chart of IRLB

The model in our team project mainly uses IRLB to train the model. First, the input feature map goes through PW to increase the dimensionality to t times. If the original depth is 32, after this layer, the depth is 32t. Then it passes through DW, the input and output dimensions of this layer are the same, and the size of the convolution kernel is 3*3. If the data flow through this layer for the first time, the stride is s, otherwise, the stride is 1. Then, the feature maps go through PW to reduce the dimensionality to c. If the depth of the input feature map is equal to the depth of the output feature map, and the step size is 1, then the two are added. Finally, layers are looped n times. The specific process is shown in Fig. 7 and Table 3.

Table 3

The structure of IRLB

Input	Operation	Output
$h*w*k$	Pointwise, Relu6, BN	$h*w*(tk)$
$h*w*tk$	Depthwise 3x3, Relu6, BN	$\frac{h}{s}*\frac{w}{s}*(tk)$
$\frac{h}{s}*\frac{w}{s}*tk$	Pointwise, linear, BN	$\frac{h}{s}*\frac{w}{s}*c$

In the model in this article, first, convert the collected 2*512 original vibration signals of the bearing end and the fan end into a 32*32 two-dimensional matrix. Then, input this two-dimensional matrix into the convolution layer with the convolution kernel as 3*3*16 and stride as 2. Thus, a feature map with a size of 16*16*16 is obtained. The feature map is input to the IRLB structure. Since the dimensionality increase is 1 (t=1), the feature map is not input to PW for increasing the depth of the feature map, but directly through DW with the stride of 2 and the convolution kernel size of 3*3 and finally, dimensionality reduction through PW to obtain an 8*8*16 output feature map. After that, taking the output feature map as input, the input feature map go through PW to increase the depth, DW with a stride of 2, and PW to reduce the depth of the feature map. Then, obtain a feature map with a size of 4*4*32. Since the number of loops is 3 (n=3), the feature map needs to be re-input to the PW of this layer to increase the dimension. Because this is the second cycle, the stride of DW is 1, and the dimensionality reduction is performed by PW. Also, because the dimension of the input feature map is 4*4*32, and the output feature map is 4*4 *32, so it is necessary to add the input feature map and the output feature map to get the final output feature map of this cycle. Finally, this layer is recirculated once, and the feature map can be output to enter the next IRLB. After this layer, the size of the feature map is 2*2*64. This feature map passes through PW with a depth of 1280. This is to simplify the calculation while simulating a fully connected neural network. The feature map compresses the width and height of the feature map to 1*1 through the pooling layer. The purpose of the pooling layer is to transform the feature map into a 1*1*N matrix that is convenient for output classification. Finally, through PW, the feature map is output to the predicted label of the classification. The specific structure of each layer is shown in **Table 4** and Fig. 8.

Table 4

The model structure

Input	Operation	t	c	n	s
32*32*1	Conv2d, 3*3	-	16	-	2
16*16*16	IRLB	1	16	1	2
8*8*16	IRLB	6	32	3	2
4*4*32	IRLB	6	64	3	2
2*2*64	Pointwise	-	1280	-	1
2*2*1280	Pooling, 2*2	-	-	-	2
1*1*1280	Pointwise	1	4	-	1

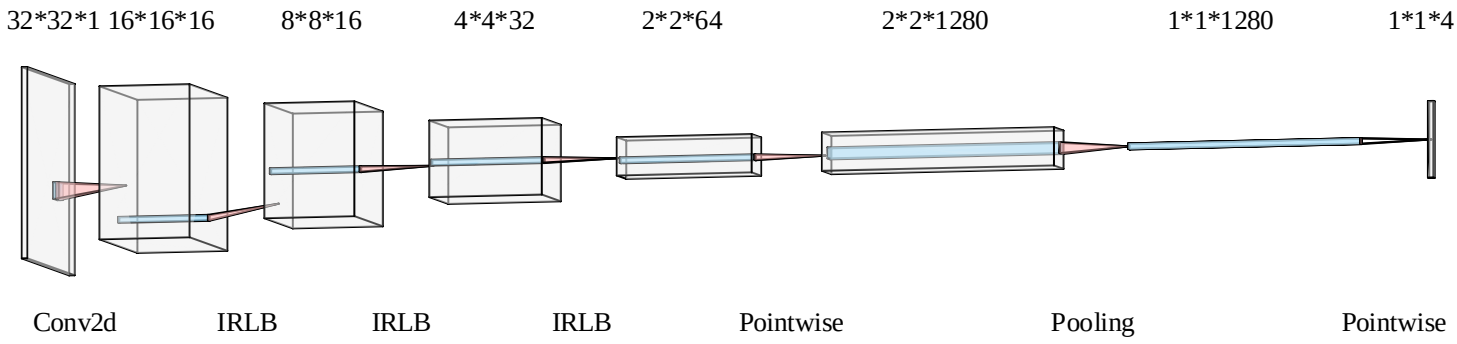

Fig. 8. The model structure

2.5 Optimizer selection

Deep neural networks use gradient descent methods to train and update weights. Choosing an appropriate optimization method is of great significance to improve the training speed of the network and the stability of the output results. This article uses the ADAGRAD optimizer, ADADELTA optimizer, ADAM optimizer, RMSPROP optimizer, and MOM optimizer to train the network, and test its accuracy and loss value (other parameters of the network will remain unchanged, batch-size is 128, the learning rate is the decay learning rate, and the training global step is 2000). To eliminate errors, each optimizer is trained and tested 5 times, and the results are averaged. The details are shown in **Table 5**

Table 5

The influence of the optimizer on the network training and test results

Optimizer	Average training accuracy (%)	Average training loss value	Average training time (min)	Average testing accuracy (%)	Average training loss value	Average testing time (s)
ADAM	99.38	0.2770	1.780	99.30	0.2758	7.704
ADAGRAD	99.69	0.2778	1.593	98.89	0.2840	7.266
ADADELTA	86.09	0.6330	1.507	79.58	0.7044	7.218
RMSPROP	99.84	0.2660	1.290	99.14	0.2767	7.554
MOM	99.69	0.2698	1.253	99.19	0.2757	7.124

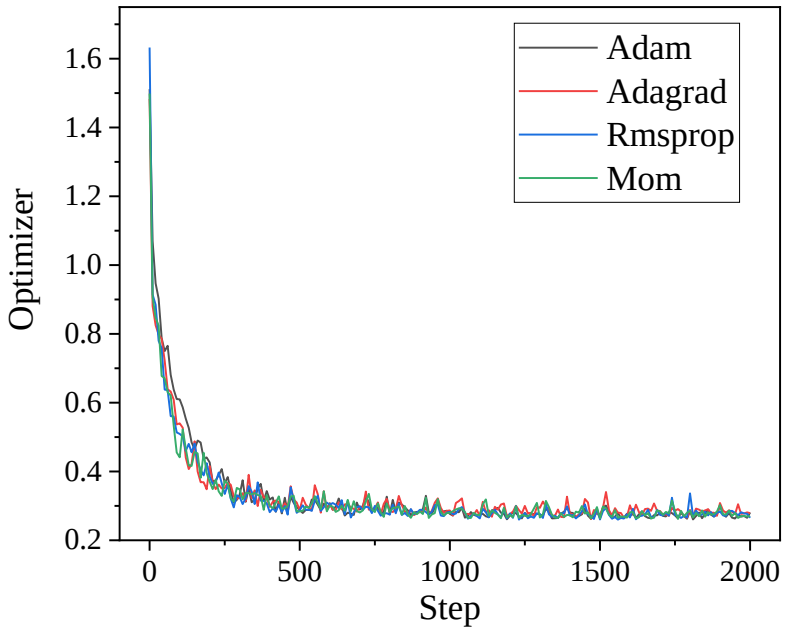

Fig. 9. Comparison chart of various optimizers

As shown in Table 5, compared with other optimizers, the average training loss and average test loss of the ADADELTA optimizer are both at least 127% higher, and the effect is the worst, so it is excluded. The parameters of the remaining four optimizers are relatively close, so compare the ADAGAD optimizer, RMSPROP optimizer, MOM optimizer, and ADAM optimizer.

As shown in Fig. 9, the convergence speed of each optimizer is faster than that of the ADAM optimizer, and after 1000 steps, all optimizers tend to converge, but the oscillation amplitude of other optimizers is significantly larger than that of the ADAM optimizer. The subsequent oscillation values after 1000 steps are shown in Table 6.

Table 6

Oscillation amplitude of each optimizer

Optimizer	Average loss	Loss variance ($\times 10^{-5}$)	Average accuracy （%）	Accuracy variance ($\times 10^{-5}$)
ADAM	0.2751	8.3	99.56	2.9
ADAGRAD	0.2859	22.1	98.97	8.8
RMSPROP	0.2765	16.1	99.46	4.8
MOM	0.2785	15.6	99.37	5.2

It can be seen from the above table that the loss average, loss variance, and accuracy variance of ADAM are all the minimum, and the accuracy average is the highest, so the ADAM optimizer performs the most stable, so the ADAM optimizer is selected as the optimizer of this model.

2.6 Learning rate

The learning rate is a very important super parameter in deep learning. The size of the learning rate greatly affects the speed of deep network convergence and determines whether the objective function can converge to the minimum and when it can converge to the minimum. If the learning rate is set too large, the loss will oscillate around the optimal value and it will be difficult to converge; if the learning rate is set too small, the convergence speed will be extremely slow and it will take longer to converge. Therefore, using a fixed learning rate is difficult to balance the relationship between the two, so this article will use exponential decay learning rate. Its advantage is fast convergence speed. The mathematical expression is shown in equation (7), and the description of the parameters is shown in Table 7.

$$decayed\_learning\_rate = learning\_rate * decay\_rate^{\frac{global\_step}{decay\_steps}} \tag{7}$$

Table 7

Exponential decay learning rate parameter description

Parameters	value
$learning\_rate$	0.1
$decay\_rate$	0.9
$decay\_steps$	50
$global\_step$	Change with the number of training steps

Fig. 10 shows the learning rate decay diagram. It can be seen from the figure that the learning rate starts at 0.1 and ends at 0.01, and the value of the learning rate decreases steadily. Fig. 9 also shows that the oscillation of the training iteration graph keeps decreasing as the training time increases. Taking the ADAM optimizer as an example, the total variance of the ADAM optimizer loss is 0.021, and the variance of the next 1000 steps is only 0.000085.

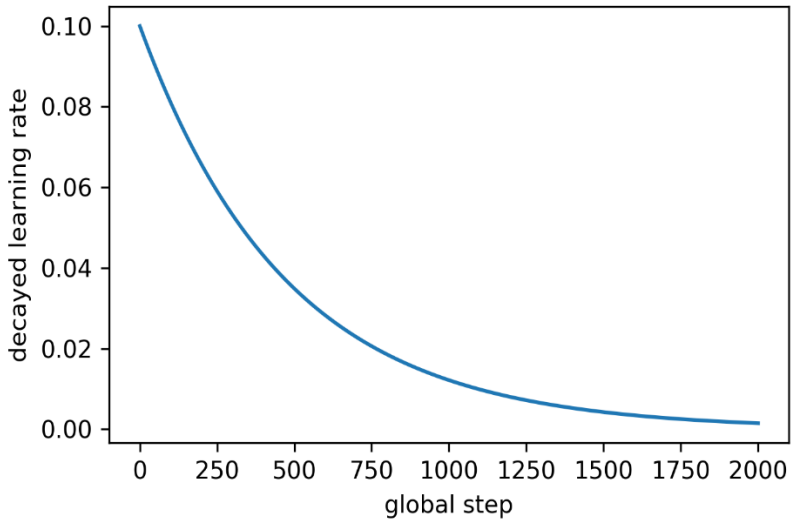

Fig. 10. Learning rate decay graph

3 Analysis of fault diagnosis results

3.1 Model Analysis

The accuracy of the model in the test set is 99.4%, which proves the feasibility of the model.

Fig. 11 shows the confusion matrix of this model. The X-axis represents the label value, and the Y-axis represents the predicted value. From the data shown in the normal bearing in the figure, it can be seen that the predictions of all bearings in normal working conditions are correct. In other words, in practical applications, the model can accurately predict whether a bearing fails, and its accuracy rate is as high as 100%. The poor performance in the confusion matrix is the ball fault, but is predicted to be an outer race fault, and the outer race fault is predicted to be the ball fault.

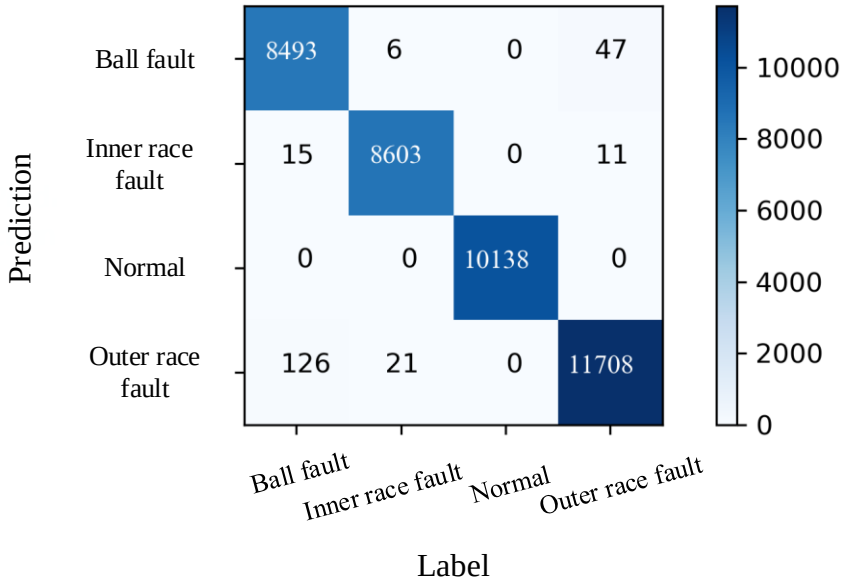

Fig. 11.

Confusion matrix

Fig. 12 shows the four P-R curves of the model. Fig. 12 (a) is the P-R curve in the Normal state. Its shape is a straight line, and the accuracy and recall rates are both 1. Fig. 12 (c) shows that the bearing with an outer rate fault has a slight jitter when the recall rate is 0.3, while Fig. 12 (b) and Fig. 12 (d) are relatively straight, and the turning point is near 0.99, showing a good performance.

F1-score is the harmonic average of recall rate and precision rate. The specific values are shown in **Table 8**.

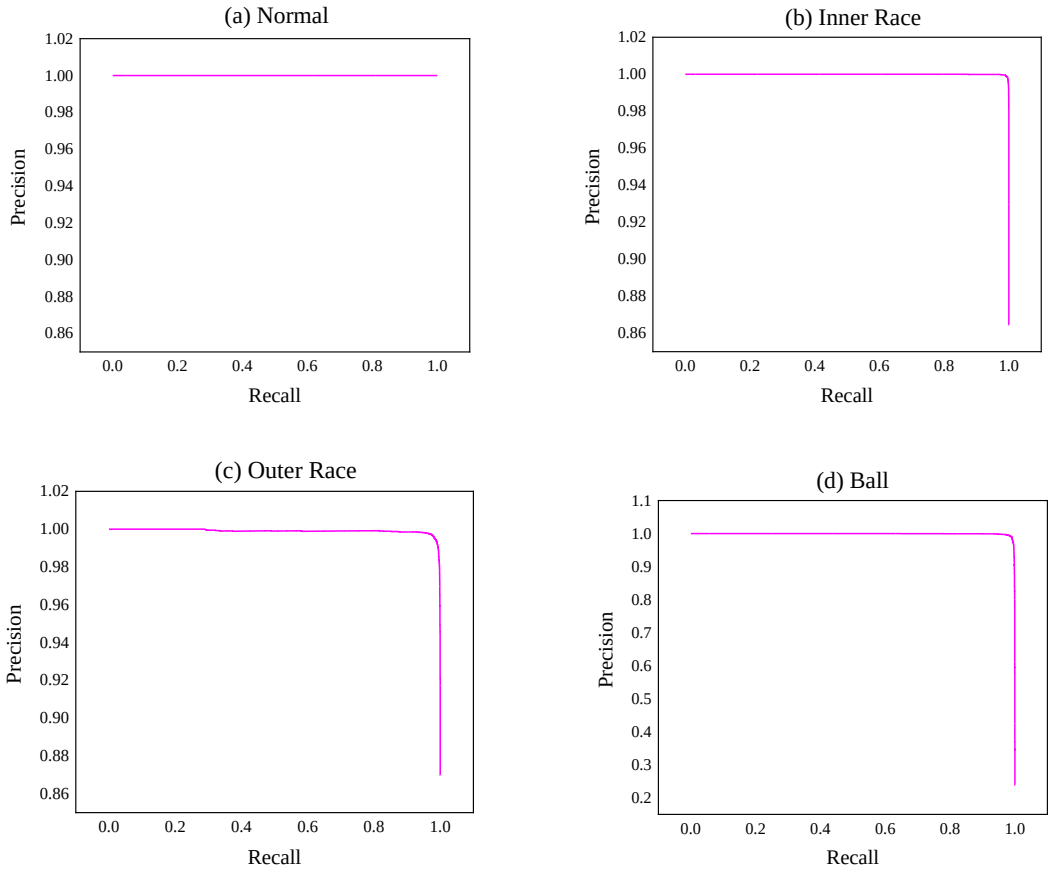

Fig. 12. P-R curve

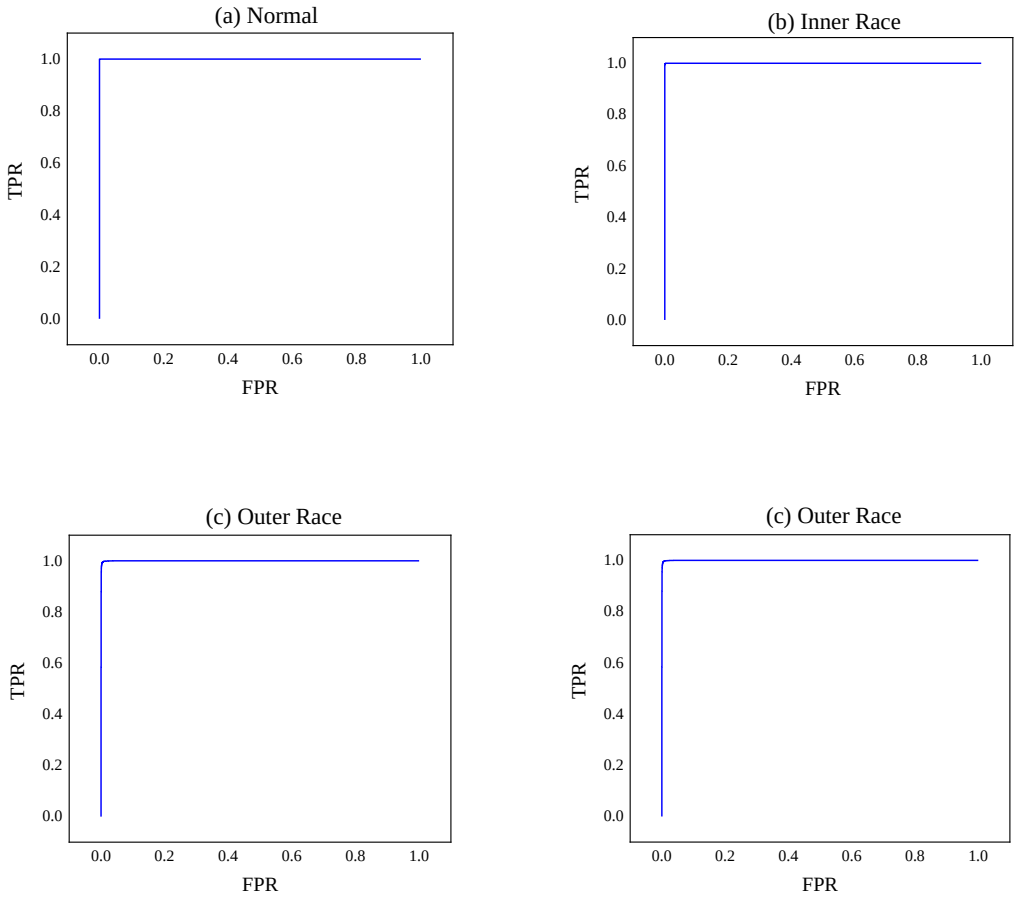

Fig. 13. ROC curve

The ROC curve represents the relationship between the true positive rate and the false positive rate. As shown in Fig. 13, it is the ROC curve in four states. **Table 8** is its corresponding AUC value.

Table 8

F1-score and AUC value

	Normal Condition	Inner Race Fault	Outer Race Fault	Ball Fault
F1-score	100%	99.69%	99.13%	98.87%
AUC	1	0.999982	0.999669	0.999447

3.2 Parameters and FLOPs of the model

The amount of parameters and FLOPs (Floating-point operations per second) of the model determines the memory size and computing speed required for the operation of the model. A model that is too large will have too high requirements for the operating environment, resulting in wasted resources, which will lead to increased costs.

Table 9 shows the FLOPS and parameter amount of IRLB network. Among them, Bottleneck in the first column represents the structure of IRLB. The parameter amount and FLOPs of IRLB are 0.9228 (Mb) and 2.7027 (M) respectively.

Table 9

Parameters and FLOPs of the model

Layer	Input feature map	params（Mb）	FLOPs（M）
Conv1	[32, 32, 1]	0.0007	0.0696
Bottleneck1	[16, 16, 16]	0.0018	0.0511
Bottleneck2	[8, 8, 16]	0.0792	0.7710
Bottleneck3	[4, 4, 32]	0.4944	1.1505
Pointwise1	[2, 2, 64]	0.3271	0.6502
Pool1	[2, 2, 1280]	-	-
Pointwise2	[1, 1, 1280]	0.0196	0.0102
Total	-	0.9228	2.7027

4 Comparison of experimental results

4.1 Corresponding Convolutional Neural Network

Firstly, adjust the corresponding CNN according to the structure of IRLB, that is, when the size of the input feature map and the output feature map is the same as the feature map of the corresponding IRLB, remove the loop structure of IRLB and establish a new neural network. Then calculate the number of FLOPs and parameters, and test its accuracy. The purpose of eliminating the loop structure is to simplify the number of parameters and FLOPs of CNN and effectively prevent the problem of gradient disappearance. The specific parameters are shown in **Table 10**.

Table 10

Parameters and FLOPs of the corresponding CNN

Layer	Input feature map	Params（Mb）	FLOPs（M）
Conv1	[32, 32, 1]	0.0007	0.0696
Conv2	[16, 16, 16]	0.0099	0.2949
Conv3	[8, 8, 16]	0.1187	0.8847
Conv4	[4, 4, 192]	0.2346	1.7695
Conv5	[4, 4, 32]	0.0786	0.1475
Conv6	[2, 2, 384]	0.9404	1.7695
Conv7	[2, 2, 64]	1.8896	3.9270
Pool1	[2, 2, 1280]	-	-
Conv8	[1, 1, 1280]	0.0196	0.0102
Total	-	3.2921	8.8730

As shown in **Table 12**, the parameter amount and FLOPs of the corresponding CNN are 3.56 times and 3.28 times that of IRLB, respectively, which proves that the parameter amount and FLOPs of IRLB are greatly reduced. If the corresponding CNN is to completely reproduce IRLB, the number of FLOPs and parameters will be increased to 32 times of IRLB.

As shown in **Table 12**, after the network is trained according to the method shown in section 3.6, the accuracy rate reaches 99.5%, which is an increase of 0.1% compared to 99.4% of IRLB.

4.2 Deep convolutional neural network

Tonglin Gao published a study on fault diagnosis of rolling bearings based on convolutional neural networks on December 19[5], using deep convolutional neural networks to increase the accuracy to 100%. Fig. 14[5] shows the schematic diagram of the network structure, according to which the corresponding FLOPs and parameter amount are calculated, as shown in Table 11.

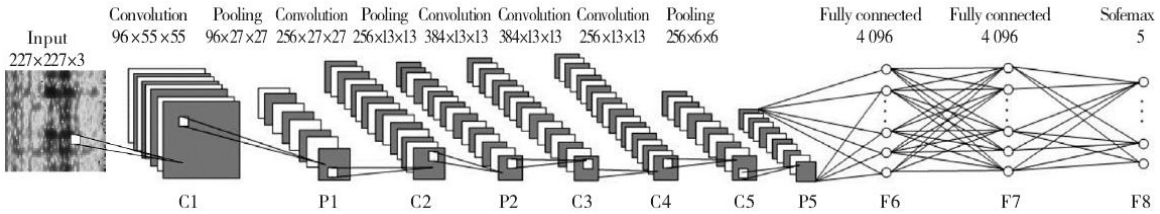

Fig. 14. Structure of the deep convolutional neural network

Table 11

Parameters and FLOPs of the deep CNN

Layer	Input feature map	Params（Mb）	FLOPs（M）
Conv1	[227, 227, 3]	0.13	210.54
Conv2	[55, 55, 96]	0.32	120.86
Conv3	[27, 27, 96]	0.85	332.30
Conv4	[27, 27, 256]	2.25	199.32
Conv5	[13, 13, 256]	3.38	298.98
Conv6	[13, 13, 384]	5.07	448.50
Conv7	[13, 13, 384]	3.39	299.00
Pool1	[13, 13, 256]	-	-
FC1	9216	144	75.49
FC2	4096	64	33.55
FC3	4096	0.08	0.04
Total	0	223.5	2008.57

Table 12

Network comparison

	IRLB	Corresponding CNN	Deep CNN
Params	0.9228Mb	3.2921	223.5Mb
FLOPs	2.7027M	8.8730	2008.57M
Accuracy	99.4%	99.5%	100%

This results in Table 12, which compares the parameters, FLOPs, and accuracy of IRLB, the corresponding CNN and the deep CNN. Compared with IRLB, the deep CNN has increased the parameter amount and FLOPs by 242 times and 743 times, respectively, while the accuracy rate has only increased by 0.6%. While IRLB effectively reduces the number of parameters and FLOPs, the accuracy is only reduced by 0.6%, which reflects the excellence of IRLB.

5 Combination to the Internet of Things

In our proposed experiment, we also develop our customized torque transducer constructed with Variable Resistors and 2 clock springs. After our customized torque transducer has been pushed with force, it will then send the signal to the Arduino with wire which is on the right top of the bread. Next, the Arduino will convert the voltage data into corresponding torque, and then send the data via Bluetooth 4.0 with a baud rate of 9600 transmit & receiver to a smartphone and Resiberry pi for further use. The torque transducer can halt with a command sent from Resibery pi and smartphone and then the Resibery pi will then further process the gathered data for further pragmatic prediction. Fig.15 shows our demonstration.

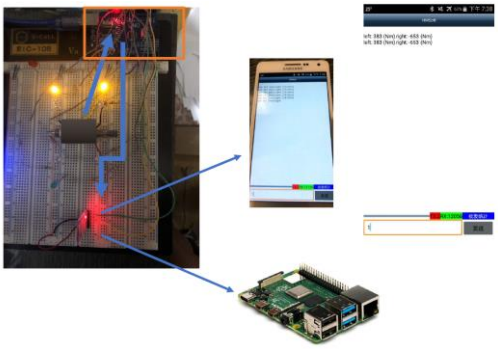

Fig. 15 Our lab-environment demonstration

6 Conclusion

Our project proposes a bearing diagnosis scheme based on IRLB, which can autonomously learn the features in the bearing vibration signal and classify the bearing state. The experimental results prove:

(1) The overlap sampling method is used as a data augmentation method, which effectively increases the number of training samples and test samples for deep learning. The only 920 training samples when the overlap sampling method is not used are increased to 92070 training samples, which satisfies the need for large amounts of data for deep learning.

(2) The 2*512 bearing vibration signal measured by the sensor at the bearing end and the fan end is transformed into a 32*32 two-dimensional input signal for fault diagnosis, which can effectively avoid human feature extraction and reduce the tester's reliance on prior knowledge and professional requirements.

(3) This article selects five optimizers for training models and compares them, namely the ADAM optimizer, ADAGRAD optimizer, ADADELTA optimizer, RMSPROP optimizer, and MOM optimizer. After training and comparing the results, it is found that the ADAM optimizer performs best when the training period is 2000 steps and the learning rate is exponentially decayed learning rate. After the model is trained to 1000 steps, the average loss of the model is 0.2751, the loss variance is $8.3*10^{-5}$, the accuracy average is 99.56%, and the accuracy variance is $2.9*10^{-5}$. Compared with other optimizers, the ADAM optimizer performs best. So choose the ADAM optimizer as the optimizer of this model.

(4) On the basis of selecting the best-performing ADAM optimizer, our team project proposes IRLB to learn and analyze bearing faults, and verify the feasibility of the model through the test set. The results show that IRLB effectively reduces the amount of parameters and FLOPs. The parameter amount is only 0.92Mb, FLOPs is 2.70M, and the accuracy rate is as high as 99.4%. While the corresponding CNN converted from IRLB and the deep CNN, although the accuracy rate is improved to 99.5% and 100%, the parameter amount is as high as 3.2Mb and 223.5Mb, and FLOPs reaches 8.8M and 20088.6 M, it is difficult to apply to the actual industrial environment. This reflects the effectiveness and practicality of the model established in this article.

(5) The method in our team project still needs attention and improvement. For example, the error rate of outer race fault and ball fault is relatively high, and the data set is not very large. Therefore, further analysis and research can be done in future research to increase the robustness and wider applicability of the model and data.

(6) We successfully implemented the information transfer between Raspberry Pi (simulate on a smart phone) and sensors can precipitate the information broadcasting of diagnosis of bearing real-time functionality.

References

[1] Levent E . Bearing Fault Detection by One-Dimensional Convolutional Neural Networks[J]. Mathematical Problems in Engineering, 2017, 2017:1-9.

[2] Nian-Long G U , Pan H , Peng H E . Bearing Fault Diagnosis Method Based on EMD-CNNs[J]. 2017.

[3] Li S, Liu G, Tang X, Lu J, Hu J. An Ensemble Deep Convolutional Neural Network Model with Improved D-S Evidence Fusion for Bearing Fault Diagnosis. Sensors (Basel). 2017;17(8):1729. Published 2017 Jul 28. doi:10.3390/s17081729

[4] Zhang B , Li W , Hao J , et al. Adversarial adaptive 1-D convolutional neural networks for bearing fault diagnosis under varying working condition[J]. 2018.

[5] [5] Gao Tonglin, Zhu Jianmin, Huang Zhiwen. Research on Fault Diagnosis of Rolling Bearing Based on Convolutional Neural Network [J]. Agricultural Equipment and Vehicle Engineering , 2019, 57(12):115-120.

[6] Zhang Wei. Research on Bearing Fault Diagnosis Algorithm Based on Convolutional Neural Network[D]. Harbin Institute of Technology. 2017.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks[J]. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520.

[8] Case Western Reserve University. Bearing Data Center, 2014, accessed: 2014-05-15 [EB/PL]. [2020-01.06] Available: https://csegroups.case.edu/bearingdatacenter.