Research Article

A New Generative Neural Network for Bearing Fault Diagnosis with Imbalanced Data

Wei You ¹, Changqing Shen ¹, Liang Chen ¹, Hongbo Que ², Weiguo Huang ¹
and Zhongkui Zhu ¹

¹School of Rail Transportation, Soochow University, Suzhou 215131, China

²CRRC Qishuyan Locomotive & Rolling Stock Technology Research Institute Co., Ltd., Changzhou 213011, China

Correspondence should be addressed to Changqing Shen; cqshen@suda.edu.cn

Received 18 September 2020; Revised 12 November 2020; Accepted 24 November 2020; Published 10 December 2020

Academic Editor: Dejan Gjorgjevikj

Copyright © 2020 Wei You et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Intelligent bearing fault diagnosis has received much research attention in the field of rotary machinery systems where miscellaneous deep learning methods are generally applied. Among these methods, convolution neural network is particularly powerful because of its ability to learn fruitful features from the original data. However, normal convolutions cannot fully utilize the information along the data flow while the features are being abstracted in deeper layers. To address this problem, a new supervised learning model is proposed for small sample size bearing fault diagnosis with consideration of imbalanced data. This model, which is developed based on a convolution neural network, has a high generalization ability, and its performance is verified by conducting two experiments that use data collected from a self-made bearing test rig. The proposed model demonstrates a favorable performance and is more effective and robust than other deep learning methods.

1. Introduction

Rolling element bearings (REBs) are common and important mechanic components used in various industries [1]. They are also the primary sources of mechanical faults in these devices [2]. REBs have three main components, namely, the inner race, the rolling element ball, and the outer race [3]. The REB faults that emerge during a running operation may lead to severe safety problems and huge maintenance costs [4]. Therefore, measuring and diagnosis of the healthy integrity and status of REBs, including their fault type, have attracted the attention of many researchers. Over the past few decades, many scholars have examined effective fault diagnosis algorithms for bearings and employed machinery vibration data for bearings diagnosis.

Data-driven intelligent fault diagnosis methods have rapidly developed over the past few years [5]. Many studies have adopted various machine learning algorithms, including K-nearest neighbor (KNN) [6], support vector machines (SVM) [7], random forest (RF) [8], extreme learning machine [9], and artificial neural networks (ANN)

[10]. These methods usually involve two procedures, namely, fault feature extraction and fault type classification. Feature extraction relies on prior knowledge and expertise background. The signal features of the time, frequency, or time-frequency domains are usually extracted from the raw vibration signals via short-time Fourier transform (STFT) [11], empirical model decomposition [12], local mean decomposition [13], or Hilbert–Huang transform [14]. In fault diagnosis procedure, these extracted features are inputted into a machine learning model to obtain the diagnosis results. However, one common drawback of these conventional intelligent methods is that an effective features extraction depends on specific domain knowledge and thereby requires much human effort.

With their increasing of computational power, deep learning-based methods can productively and rapidly utilize mechanical vibration signals [15] and generate accurate diagnosis results without requiring much expertise [16]. With their powerful representation learning ability, deep learning methods have attracted much attention in fault diagnosis research. These methods have also achieved a

state-of-the-art performance in bearing fault diagnosis. Zhang et al. [17] proposed a fault diagnosis model based on deep neural networks, and their experimental results showed that this model can efficiently recognize bearing fault types. Wang et al. [18] proposed a deep neural network with batch normalization for the diagnosis of bearings and gears. He and He [11] used an optimized deep learning model called large memory storage retrieval (LAMSTAR) neural network for bearing fault diagnosis and used acoustic emission signals to validate its classification performance. Lu et al. [19] applied a stacked denoising autoencoder (SDA) for bearing fault diagnosis. Shen et al. [20] proposed a method based on stacked CAE for automatic robust features extraction and fault diagnosis for bearings and gearbox and the method performed well under noise interference. Janssens et al. [21] developed a convolutional neural network and feature learning model to build a fault detection system for outer-raceway faults. Eren et al. [22] used a 1D convolutional neural network to achieve a generic real-time bearing fault diagnosis, and their experiment results showed that this network can demonstrate an excellent classification performance.

Although the above studies have applied different algorithms, they generally all assume that their employed datasets for validation are balanced. However, in reality, acquiring normal data is much more convenient than acquiring faulty ones, thereby resulting in imbalanced datasets. Such problem arises when classes of a dataset are not equally distributed [23]. One aim of classic algorithms is to maximize the classification accuracy. However, the employed accuracy metric is biased toward the majority class; that is, a classifier may achieve a very high classification accuracy without accurately predicting the minority class even if such class is often more important than the majority class in reality. Accordingly, an accurate prediction or detection of the minority class should be prioritized over the majority class [24]. Therefore, how to fully represent the minority class or how to classify an imbalanced data warrants further investigation.

The imbalanced classification problem has received much research attention across different fields. There are two main research streams, one is focused on the data and the other one is focused on the algorithm. Xu et al. [25] develop a new framework for fault diagnosis. They first designed and extracted features and then selected features for eliminating redundant features and identifying effective ones before using them for fault classification. Zhang et al. [26] proposed a synthetic oversampling approach called weighted minority oversampling (WMO) to balance the imbalanced data distribution and then used the balanced data to achieve the fault diagnosis. Mao et al. [27] used fast Fourier transform to get the frequency spectrum of original vibration signal. Then they applied a generative adversarial network to generate the synthetic minority samples and then used the synthetic balanced data into classifier for fault diagnosis. These data-based methods focus on the data distribution. Their key idea is trying to transfer imbalanced data into balanced ones. Usually, these methods consist of lots of work on feature extraction and selection. Zhao et al.

[28] used a normalized CNN to automatically and accurately identify the condition of rolling bearings. They used imbalanced data and tested the performance of their method on an open-source experimental dataset. Zhang et al. [29] applied the fast clustering algorithm and SVM for rotating machinery fault diagnosis with imbalanced data. However, they only considered four fault classes in their work. Studies on imbalanced data are also relatively fewer than those focusing on balanced data probably due to the fact that the most commonly employed methods are more suitable for balanced data than unbalanced ones. However, the percentage of normal samples is usually much larger than that of fault samples in reality. In this case, the diagnosis performance of proposed methods should be tested by using imbalanced data. This study then focuses on the development of a model that is applicable to imbalanced data and can deal with different cases.

This study proposes the generative convolutional neural network (GCNN) to diagnose bearing faults while taking imbalanced data into consideration by using small-scale samples. This model is inspired by both the convolutional neural network and the residual neural network [30] and comprises two main structures, namely, the basic network body and the skip connections between the encode and decode parts. This model can concatenate encoding feature maps to the corresponding decoding feature maps. When the features are being upsampled in the decode part, each layer connects to a corresponding layer in the encode part. In this way, GCNN prevents the loss of information during the convolution process without using superfluous parameters.

The rest of this article is organized as follows. Section 2 introduces the proposed model. Section 3 describes the experiment test rig and its setup, the model implementation, and the data preprocessing and training processes. Section 4 discusses the experimental situation, analyzes the experiment results for both balanced data and imbalanced data, explores the generalization ability of GCNN in three scenarios, and compares its performance with that of several deep learning methods. Section 5 concludes the paper.

2. Theoretical Background and the Proposed Model

2.1. Basic Theory of the Convolutional Neural Network. The convolutional neural network (CNN) is a deep learning model with a deep feed-forward neural network. The typical CNN architecture includes an input layer, several convolutional layers, several pooling layers, one or two fully connected layers, and an output layer as illustrated in Figure 1.

2.1.1. Convolutional Layer. The convolutional layer in a CNN comprises several convolution filters that aim to generate feature maps through the convolutional functions of each filter. Such convolutional function can be formulated as

$$y_j^l = f\left(\sum_{i=1}^k w_{i,j}^l * y_i^{l-1} + b_j^l\right), \quad (1)$$

where y_j^l represents the convolutional output of the j th channel of layer l , k is the number of filters in this layer, y_i^{l-1} represents the i th channel output in the previous layer $l-1$, $w_{i,j}^l$ represents the weights between this layer l and the previous layer, b_j^l represents the bias of the j th channel of convolutional layer l , $*$ represents the convolutional operation, and f represents the nonlinear activation function.

2.1.2. Pooling Layer. The pooling layer is another nonlinear downsampling layer that mainly reduces the dimensions of the convolutional layer output through downsampling. This layer lacks any weights or bias units, but its number of parameters can be reduced. Several pooling methods are available, including average pooling, max pooling, logarithmic pooling, and weight pooling, among which max pooling is frequently used in classification tasks. Max pooling can obtain the maximum value of a set area and can be formulated as

$$y_j^l = \max(w(a, b) \cap y_j^{l-1}), \quad (2)$$

where $w(a, b)$ represents the pooling filter, a and b represent the size of the filter, y_j^{l-1} represents the j th channel output in convolutional layer $l-1$, and $\cap$ represents the overlap between the output of the previous layer and the pooling filter.

2.1.3. Fully Connected Layer. After passing through the convolution and pooling layers, the extracted features enter the fully connected layer, where these features are further extracted and connected to the next layer (usually the output layer). All neurons in a fully connected layer are defined as

$$y^l = f(w^l \cdot y^{l-1} + b^l), \quad (3)$$

where w^l represents the weight matrix connecting the l th and $l-1$ th layers, b^l represents the bias vector, y^l and y^{l-1} represent the outputs of the l th and $l-1$ th layers, and f represents the nonlinear activation function.

2.1.4. Softmax Layer. The output layer uses a softmax function, which defines the probability of each class for multiclass classification. In softmax, the probability for one sample to belong to the i th class with the k -multiclass problem can be defined as

$$p(y = i|z) = \frac{e^{z_i}}{\sum_{i=1}^k e^{z_i}}, \quad \text{where } z_i = \sum_{i=1}^k x_i \cdot w + b. \quad (4)$$

If the output for a softmax neuron is close to 1, then the outputs for the other neurons will be close to 0. The sum of all predicted class probabilities is 1.

FIGURE 1: Typical architecture of a CNN.

2.2. The Proposed Method: GCNN. The proposed generative convolutional neural network (GCNN) is a new model built based on the architecture of CNN with extra skip connections. This model mainly involves convolution operations, max pooling operations, ReLU activations, concatenations (skip connection), and upsample convolution operations, all of which comprise the encode and decode parts of the model. Figure 2 shows the GCNN architecture.

The input data are fed into the network before they pass through the encode and decode parts. The atomically extracted features are then flattened into a single dimension before imputed into a classifier to obtain the output. The encode part learns features at different scales, whereas the decode part then combines information from the former layers to obtain highly comprehensive features.

The encode part has three 2D convolutional layers, with each layer having a set of parameters that form learnable kernels (also known as filters). The number of kernels increases at each subsequent layer, and each filter has a small receptive field whose depth is equal to that of the input volume of the layer. The convolution process involves element-wise multiplications where the entries of the filter are multiplied by the input matrix values. This procedure produces layer-by-layer learned feature maps.

The convolutional and fully connected layers use the rectified linear unit (ReLU) as their nonlinear activation functions. The ReLU is a nonsaturating activation function that is defined as

$$f(x) = \text{ReLU}(x) = \max(0, x). \quad (5)$$

The encode part comprises three superpositions of the convolution and max-pooling layers.

Another convolutional layer that serves as the model's center of GCNN is located between the encode and decode parts.

The decode part of GCNN is symmetric to the encode part. The convolutional layers in this part also have 3×3

FIGURE 2: GCNN architecture.

kernels but use a decreasing number of filters to make the feature map return to its original size.

At the end of the network, the learned features are flattened into a single dimension before being fed into the softmax classifier. The softmax layer can generate mutually probabilistic output categories.

GCNN has an additional skip path to create an expansion feature map. This skip path directly concatenates the encode and decode parts of the model with layers that share similar feature sizes. Afterward, extra rich features are added from the encode part to the corresponding expansion layer of the decode part. The concatenation can benefit the feature representations and retain some information from the prior layer in the encode part that may be lost after the data pass through the encode part.

GCNN can learn to associate the characteristics of the former encode part layer with those of the later decode part layer. The principal idea of this model is to extract features from all layers in the decode part separately and finally to use them in the classified layer for classification.

2.3. Initial Bias Setting. When the dataset is imbalanced, bias can be accurately initialized by considering the imbalance degree. Setting the bias correctly can accelerate convergence and this setting can just be applied in the final layer of the model.

The correct bias to set can be derived as

$$p_0 = \frac{\text{pos}}{\text{pos} + \text{neg}} = \frac{1}{1 + e^{-b_0}},$$

$$b_0 = -\log_e \left(\frac{1}{p_0 - 1} \right) = \log_e \left(\frac{\text{pos}}{\text{neg}} \right), \quad (6)$$

where p_0 is the initial probability, b_0 is the initial bias of the model's, pos represents the positive samples, and neg represents the negative samples.

When the initial bias is set this way, the model no longer needs to spend the first few epochs in learning the bias influenced by the positive examples, thereby saving training time and facilitating convergence. Therefore, setting the bias correctly provides an easy starting point for training the model and facilitating the feature learning process.

2.4. Adam Optimizer and Learning Rate Decay. Adaptive moment estimation (Adam) [31] is an algorithm for efficient stochastic optimization that only requires first-order

gradients and a small amount of computer memory. Adam updates any parameter with an individual learning rate, that is, each parameter in the network is associated with a specific learning rate. The parameters are updated as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \quad (7)$$

where θ_t and θ_{t+1} are the model parameters of the t and $t + 1$ timesteps. η is the learning rate, ϵ is assigned a small value to prevent the divisor from being equal to 0, and $\hat{m}_t$ and $\hat{v}_t$ are the bias corrected first and second moment estimates calculated as

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad (8)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

In these equations, β_1 and β_2 are the exponential decay rates for the moment estimates that belong to $[0, 1)$, m_t and m_{t-1} are the biased first moment estimates of the t and $t - 1$ timesteps, v_t and v_{t-1} are the biased second moment estimates of the t and $t-1$ timesteps, and g_t is gradients at timestep t .

Although Adam adapts to dynamic learning rates, using a learning rate decay schedule can substantially improve the performance of this algorithm. Learning rate decay is observed when the metric stops improving. For instance, using testing loss as a metric during the training process and when it decreases normally, such decrease has no effect on the learning rate. However, when the patience epoch is reached (i.e., testing loss does not decrease or directly increase), a learning rate decay takes place. The decayed learning rate can be computed as follows by multiplying the original learning rate by a factor:

$$\eta_{\text{decay}} = \eta \cdot \text{factor}. \quad (9)$$

2.5. Evaluation Metrics. The most direct way to evaluate the performance of a classifier is to use a confusion matrix. Consider an imbalance binary class classification

problem where the minority class is positive and the majority class is negative. As shown in Table 1, a confusion matrix has four basic elements, namely, true positive (TP), false negative (FN), false positive (FP), and true negative (TN). Among these elements, TP and TN are samples that have been correctly diagnosed or classified, whereas FP and FN are samples that have been incorrectly classified.

Accuracy is the most frequently used metric for evaluating classifier performance. This metric represents the proportion of correctly diagnosed samples to all samples. The confusion matrix defines accuracy as

$$\text{accuracy} = \frac{TP + TN}{TP + FN + FP + TN}. \quad (10)$$

Accuracy presents a good and simple way to measure the performance of a model with balanced data. However, when the class distribution varies, accuracy also greatly varies due to its sensitivity to changes in the data. The presence of imbalanced data also hinders relative analysis and prevents an algorithm from dealing with such data.

Precision, recall, F1-score, and AUC are frequently used as the evaluation metrics when dealing with imbalanced data [32]. Precision and recall measure exactness and completeness, respectively, and demonstrate an inverse relationship. These metrics can be computed as the percentage of positive predictions and classes that have been correctly classified, respectively:

$$\text{precision} = \frac{TP}{TP + FP}, \quad (11)$$

$$\text{recall} = \frac{TP}{TP + FN}. \quad (12)$$

F1-score combines precision and recall into a single measure. This metric is positively related to the performance of a model and may present a highly comprehensive description of such model. F1-score can be formulated as

$$F1 - \text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \quad (13)$$

Area under the curve (AUC) indicates the probability for how more confident of a classifier to randomly choose a positive example is actually positive compared to randomly choosing a negative example which is positive. The specific curve is the receiver operating characteristic (ROC) curve that plots the true positive rate (TPR) against the false positive rate (FPR). The TPR is defined similarly to Recall (see (12)), whereas the FPR is defined as

$$FPR = \frac{FP}{FP + TN}. \quad (14)$$

The performance of a model is positively related to the value of the AUC, which ranges from 0 to 1. A model with 100% correct predictions has an AUC value of 1.0.

This study employs accuracy, precision, recall, F1-score, and AUC to evaluate the performance of GCNN across different situations.

TABLE 1: Confusion matrix.

	Positive prediction	Negative prediction
Positive class	True positive (TP)	False negative (FN)
Negative class	False positive (FP)	True negative (TN)

2.6. Overall Flowchart of the Proposed Method. Figure 3 presents the flowchart of GCNN, in which main stages are summarized as follows.

Stage 1: data acquisition: the mechanical vibration signal data of bearings are detected by sensors and collected by a data acquisition system.

Stage 2: data preparation: without extra signal pre-processing and manual feature extraction, the original signal data are split into different dataset according to the dividing descriptions of these datasets (Section 3.2).

Stage 3: model building: the GCNN is constructed on a computer server and is compared with other deep learning models.

Stage 4: GCNN is trained by using the training set used for Experiment 1, Scenario 1, Case 1.

Stage 5: the trained GCNN is tested by using the testing set for Experiment 1, Scenario 1, Case 1, and the diagnosis results are obtained.

Stage 6: stages 4 and 5 are repeated for the rest of the experiment, scenarios, and cases.

3. Implementation and Experiments Setup

3.1. Experiment Details. The self-made bearing test experimental platform shown in Figure 4 was used for the vibration signal acquisition [33]. This platform comprises a drive motor, healthy bearings, acceleration sensors, a loading system, an NIPXle-1082 data acquisition system, and testing bearings. PCB 352C33 was used as the accelerometer and was installed on the testing bearing at a 12:00 direction. 6205-2RS SKF bearings were used as test bearings. The motor speed was set to 961 r/min (loaded 1 kN) and the sampling frequency was set to 10 kHz.

Four categories of rolling bearings, namely, normal (Norm), inner-race fault (IF), rolling ball fault (BF), and outer-race fault (OF), under different faulty severities were collected from the test platform. Test experiments were carried out under five bearing fault severity levels (0.2, 0.3, 0.4, 0.5, and 0.6 mm) for IF, BF and OF, separately. Figure 5 presents the vibration signals under different health conditions.

3.2. Dataset. Two datasets were used for the two experiments. Experiment 1 employed a balanced dataset, whereas experiment 2 employed an imbalanced dataset containing several different imbalance degree cases.

To evaluate the generalization of GCNN, each of the aforementioned datasets involved three scenarios, four health conditions (four classes), ten health conditions (ten

FIGURE 3: Flowchart of GCNN.

classes), and sixteen health conditions (sixteen classes). Case 1 represented the balance data situation. For all cases, the number of Norm samples for all cases was set to 100,

whereas the number of faulty samples was reduced from 100 to 45, thereby suggesting that the imbalance rates of normal samples to the faulty samples gradually increased.

FIGURE 4: Bearing test platform.

FIGURE 5: Vibration signals for different bearing health conditions. (1) Norm, (2) IF: 0.2 mm, (3) IF: 0.3 mm, (4) IF: 0.4 mm, (5) IF: 0.5 mm, (6) IF: 0.6 mm, (7) BF: 0.2 mm, (8) BF: 0.3 mm, (9) BF: 0.4 mm, (10) BF: 0.5 mm, (11) BF: 0.6 mm, (12) OF: 0.2 mm, (13) OF: 0.3 mm, (14) OF: 0.4 mm, (15) OF: 0.5 mm and (16) OF: 0.6 mm.

For each condition, 80% of the data were used for training, and the remaining 20% were used for testing. The dataset details are shown in Tables 2 and 3.

3.3. Implementation. The model was trained and tested on a laboratory computer server with a Windows 7 operating system. Python 3.6, CUDA 8.0, cudnn 7.6.0, Tensorflow 2.0.0

TABLE 2: Seven scenarios.

	Normal condition			Each faulty condition		
	Samples	Training samples	Testing samples	Samples	Training samples	Testing samples
Case 1	100	80	20	100	80	20
Case 2	100	80	20	90	72	18
Case 3	100	80	20	80	64	16
Case 4	100	80	20	70	56	14
Case 5	100	80	20	60	48	12
Case 6	100	80	20	50	40	10
Case 7	100	80	20	45	36	9

TABLE 3: Bearing dataset setting with three classes.

Condition	Faulty severity level (mm)	Scenario 1 label (four classes)	Scenario 2 label (ten classes)	Scenario 3 label (sixteen classes)
Norm	0	1	1	1
	0.2	2	2	2
	0.3	—	3	3
IF	0.4	—	4	4
	0.5	—	—	5
	0.6	—	—	6
BF	0.2	3	5	7
	0.3	—	6	8
	0.4	—	7	9
	0.5	—	—	10
	0.6	—	—	11
OF	0.2	4	8	12
	0.3	—	9	13
	0.4	—	10	14
	0.5	—	—	15
	0.6	—	—	16

[34], and Keras 2.3.1 [35] were among the software or packages used in the implementation. And Table 4 presents the parameters of GCNN.

3.4. Training. The model was trained by Adam with an initial learning rate of 10^{-4} , learning rate decay factor of 0.2, patience of 5, and minimum learning rate of 10^{-8} . During the training process, the validation loss was used as the training metric to adjust the learning rate. According to the learning rate decay strategy, if the loss does not decrease for five consecutive epochs, then the learning rate will be computed as the current learning rate multiplied by the decay factor. The training epoch was set to 100 for each round of training.

Each scenario, experiment, and case followed the same training methodology.

4. Results and Analysis

To evaluate the diagnosis performance and the generalization ability of GCNN, two experiments with balanced and imbalanced data were separately performed in three scenarios (four classes, ten classes, and sixteen classes). The imbalanced experiment included six imbalance degree cases.

TABLE 4: Configuration of the GCNN.

Layer name	Layer configuration
Convolution layer 1	3×3 , 32
Max pooling layer 1	2×2 , stride 2
Convolution layer 2	3×3 , 64
Max pooling layer 2	2×2 , stride 2
Convolution layer 3	3×3 , 128
Max pooling layer 3	2×2 , stride 2
Convolution layer 4	3×3 , 256
Upsample 1	3×3 , 128
Merge 1	—
Convolution layer 5	3×3 , 64
Upsample 2	3×3 , 64
Merge 2	—
Convolution layer 6	3×3 , 32
Upsample 3	3×3 , 32
Merge 3	—
Convolution layer 7	3×3 , 32
Flatten layer	—
Softmax layer	10 (or 4 or 16)

4.1. Evaluation with Balance Data: Diagnosis and Analysis. GCNN was initially trained with balanced data in the ten-classes scenario. The Case 1 dataset listed in Table 2 was used in this experiment. To compare GCNN with other artificial

intelligence methods, several deep learning methods were applied for diagnosis, including a wide convolutional neural network (WCNN), a narrow convolutional neural network (NCNN), and a classical five-layer neural network (FNN). WCNN has seven convolutional layers with ReLU activation functions. Each convolutional layer in this model was followed by a max pooling layer and fully connected and softmax layers can be found at the end of the model. NCNN has a similar architecture as WCNN yet only has two convolutional layers and two max pooling layers. Meanwhile, the FNN used here is a normal fully connected neural network and it has an input layer, three fully connected layers, and one softmax layer as its output layer. The nodes numbers of three hidden layers are 3072, 1536, and 500. The other hyperparameters setting are same as the other models.

Figure 6 shows the training and testing accuracies of the compared methods in the four-classes, ten-classes, and sixteen-classes scenarios. GCNN achieves a 100% classification accuracy across all scenarios in the training process (Figure 6(a)). GCNN also outperforms the other models in terms of testing accuracy across the three scenarios. Despite demonstrating a favorable diagnostic performance in all scenarios, WCNN reported 4%, 3%, and 4% lower accuracies than GCNN in the four-classes, ten-classes, and sixteen-classes scenarios, respectively. NCNN did not fare better than GCNN or WCNN, and FNN demonstrated the worst performance among all models.

Having more classification classes increases the difficulty for a model to generate good results. Such limitation reflects the limited generalization ability of a model. The performance of GCNN in the three scenarios validates its good generalization ability to deal with different classes and numbers of diagnosis.

GCNN was initially designed for fault diagnosis in the ten-classes scenario. Figure 7 plots the accuracy and loss curves of the four compared models during their learning process of the training and testing datasets. This plot facilitates the comparison of the performance and convergence rates of these models. As shown in Figure 7(a), the training loss of GCNN converged sharply and much faster than those of NCNN and FNN. The testing loss of GCNN also converged quickly and showed minimal vibrations. The training loss curves of WCNN were similar to those of GCNN yet took more epochs to converge. Meanwhile, the loss curves of NCNN converged slower than those of former two models, but its testing loss curve converged to a larger value. FNN encountered an overfitting problem. The trends shown in Figure 7(b) were opposite to the changes in the loss curve. All four models achieved high training accuracies with different convergence rates, but only GCNN achieved the highest accuracy quickly and stably.

Given that introducing more category labels will increase complexity, a traditional model cannot perform well in a different class situation. To check whether GCNN has such generalization ability and to evaluate the suitability of this model for different-class labels fault diagnosis, the results for the two classification scenarios are described as follows.

Figures 8 and 9 present the loss and accuracy change curves of the compared models during the learning

processes in the four-classes and sixteen-classes scenarios, respectively. Among all models, GCNN achieved the highest accuracy and lowest loss in both scenarios.

A balanced dataset was used for fault diagnosis in experiment 1. GCNN outperformed all the other models when dealing with balanced data and demonstrated a favorable generalization ability in diagnosing bearing faults in the four-classes, ten-classes, and sixteen-classes scenarios.

4.2. Evaluation with Imbalanced Data: Diagnosis and Analysis. The diagnosis performance of the GCNN with imbalanced data was investigated to further test its ability. Experiment 2 used an imbalanced dataset with different imbalance degrees. As shown in Table 2, six imbalanced cases (Cases 2 to 6) were implemented in experiment 2 to investigate the generalization ability and robustness of GCNN in dealing with different imbalance degree situations.

First, the proposed GCNN was compared with a classical method named the synthetic minority oversampling technique and support vector machine (SMOTE-SVM) designed for imbalanced dataset. The compared method was applied along the classical pipeline way; first we use the SMOTE algorithm to the imbalanced dataset to create a balanced dataset, then extract the signals' classical time and frequency features, and then applied SVM to do the classification. This comparison evaluation used the most severe imbalance degree case which is Case 7 as their dataset. And the parameters setting of each method were the same in three different scenarios.

Figure 10 is the t-SNE visualization of the diagnosis results for different scenarios. Figures 10(a)–10(c) are results of SMOTE-SVM for four-classes, ten-classes, and sixteen-classes fault diagnosis, respectively. Figures 10(d)–10(f) are results of GCNN for three different scenarios, respectively. It is clear that GCNN got better results than SMOTE-SVM in all three scenarios. As the number of classes increased, data distribution may vary minorly from each other, and SMOTE failed to generate robust samples for each faulty class. This brings complexity for the SVM classifying each class type and finally made the results in chaos. Conversely, the GCNN's relatively stable performance shown a good automatically feature extraction ability and good generalization. More studies on GCNN are then carried on in the following part.

Next, during the deep learning model evaluation, a confusion matrix was used to summarize the true and predicted labels for each model during the testing process. If the model had predicted everything perfectly, then the confusion matrix would be a diagonal matrix where the values off the main diagonal, which indicate incorrect predictions, would be equal to 0. A lighter color of the confusion matrix corresponded to fewer incorrectly labeled samples. Figures 11–14 show the confusion matrix of the GCNN and the three other compared models in the ten-classes scenario with different cases.

Figure 11 shows that the predicted labels of GCNN in the ten-classes scenario closely match the true labels in six different cases. The confusion matrices of almost all cases

FIGURE 6: Training and testing accuracies of the models across different scenarios. (a) Training accuracy and (b) testing accuracy.

FIGURE 7: Loss and accuracy curves of the models in experiment 1 with ten classes. (a) Loss. (b) Accuracy.

have clear diagonals. The true label 5 showed the worst match in different cases, especially in Cases 5 and 6. Label 5 in the ten-classes scenario represented BF with 0.2 mm fault size.

As shown in Figure 12, the confusion matrices of WCNN in different cases were similar to those of GCNN, thereby validating the favorable classification performance of the

former. Figures 13 and 14 show that NCNN and FNN encountered difficulties in dealing with all imbalance cases. While NCNN managed to handle the first few low imbalance level cases, FNN performed poorly in all cases.

Precision, recall, F1-score (calculated from precision and recall), and AUC were used as metrics in analyzing the results obtained with imbalanced data as discussed in

FIGURE 8: Loss and accuracy curves of the models in experiment 1 with four classes. (a) Loss. (b) Accuracy.

FIGURE 9: Loss and accuracy curves of the models in experiment 1 with sixteen classes. (a) Loss. (b) Accuracy.

Section 2.5. Among these metrics, AUC was mainly considered, with a larger value indicating a better performance. Table 5 compares the performance of the deep learning models in the ten-classes scenario.

As shown in Table 5, GCNN outperforms all the other models in terms of AUC. Specifically, the AUC values of GCNN were above 0.97 across all 6 cases, with the AUC value in Cases 2 and 3 nearing 1. All metrics decreased along with an increasing imbalance degree. A higher imbalance degree prevented a model from effectively diagnosing fault

class because most classifiers are sensitive to the distribution of samples. The values of these metrics also decreased along with an increasing imbalance degree. GCNN demonstrated the best performance even if some of its metrics were lower than those of WCNN.

To discover additional information, the changes in precision, recall, AUC, and loss during the learning process were plotted for visualization. Figure 15 presents the curves of GCNN under different cases. The training and testing curves of GCNN were better than those of the other

FIGURE 10: Visualization of the fault diagnosis results of GCNN and the SMOTE-SVM for different scenarios by t-SNE. (a) SMOTE-SVM four classes. (b) SMOTE-SVM ten classes. (c) SMOTE-SVM sixteen classes. (d) GCNN Case 7 four classes. (e) GCNN Case 7 ten classes. (f) GCNN case 7 sixteen classes.

FIGURE 11: Confusion matrices of GCNN for the ten-classes scenario in different cases. (a) Case 2. (b) Case 3. (c) Case 4. (d) Case 5. (e) Case 6. (f) Case 7.

FIGURE 12: Confusion matrices of WCNN for the ten-classes scenario in different cases. (a) Case 2. (b) Case 3. (c) Case 4. (d) Case 5. (e) Case 6. (f) Case 7.

FIGURE 13: Confusion matrices of NCNN for the ten-classes scenario in different cases. (a) Case 2. (b) Case 3. (c) Case 4. (d) Case 5. (e) Case 6. (f) Case 7.

FIGURE 14: Confusion matrices of FNN for the ten-classes scenario in different cases. (a) Case 2. (b) Case 3. (c) Case 4. (d) Case 5. (e) Case 6. (f) Case 7.

TABLE 5: Performance of different deep learning methods in ten-classes diagnosis.

Ten classes		Precision	Recall	F1-score	AUC
GCNN	Case 2	0.893	0.874	0.883	0.996
	Case 3	0.942	0.896	0.919	0.996
	Case 4	0.812	0.740	0.774	0.985
	Case 5	0.903	0.797	0.846	0.985
	Case 6	0.912	0.755	0.826	0.970
	Case 7	0.862	0.743	0.798	0.967
WCNN	Case 2	0.917	0.907	0.912	0.986
	Case 3	0.932	0.921	0.926	0.985
	Case 4	0.833	0.788	0.810	0.951
	Case 5	0.872	0.852	0.862	0.959
	Case 6	0.870	0.855	0.862	0.945
	Case 7	0.833	0.792	0.812	0.958
NCNN	Case 2	0.918	0.863	0.890	0.988
	Case 3	0.901	0.829	0.863	0.986
	Case 4	0.865	0.788	0.824	0.974
	Case 5	0.880	0.805	0.841	0.970
	Case 6	0.800	0.691	0.741	0.922
	Case 7	0.734	0.574	0.644	0.926
FNN	Case 2	0.642	0.522	0.576	0.829
	Case 3	0.640	0.543	0.587	0.837
	Case 4	0.542	0.438	0.485	0.777
	Case 5	0.563	0.453	0.502	0.814
	Case 6	0.563	0.409	0.474	0.783
	Case 7	22	0.485	0.544	0.812

methods. This model also achieved a fast-converging speed and low vibration for all metrics.

In sum, GCNN demonstrated a good performance when diagnosing with imbalanced data. This model also shows good robustness and generalization ability when dealing with different imbalance degree data in the ten-classes scenario.

Figure 16 presents the fault diagnosis AUC results of the models in the four-classes and sixteen-classes scenarios to further check whether GCNN can handle different complex scenarios, including the balanced dataset and six imbalanced cases. Tables 6 and 7 present the four metrics of all models in the four-classes and sixteen-classes scenarios of six imbalanced cases, respectively.

GCNN outperformed all models in both the four-classes and sixteen-classes scenarios and showed great robustness and generalization ability to deal with increasingly imbalanced data. For both the four-classes and sixteen-classes scenarios, WCNN and NCNN obtained poor results as the imbalance degree increased. FNN did not obtain an AUC value above 0.85.

Experiment 2 used an imbalanced dataset for fault diagnosis, and results highlight that GCNN can solve the imbalance diagnosis problem and achieve better results than the other models. In sum, GCNN has a good generalization ability to diagnose bearing faults in different scenarios with imbalanced data.

FIGURE 15: Metrics of GCNN for the ten-classes scenario across different cases. (a) Loss, (b) precision, (c) recall, and (d) AUC.

FIGURE 16: AUC of seven cases. (a) Four classes and (b) sixteen classes.

TABLE 6: Performance of different deep learning methods in four-classes diagnosis.

Four		Precision	Recall	F1-score	AUC
GCNN	Case 2	0.958	0.919	0.938	0.996
	Case 3	0.956	0.956	0.956	0.995
	Case 4	0.984	0.984	0.984	1.000
	Case 5	0.870	0.839	0.855	0.988
	Case 6	0.833	0.800	0.816	0.971
	Case 7	0.872	0.872	0.872	0.975
WCNN	Case 2	0.931	0.905	0.918	0.994
	Case 3	0.776	0.765	0.911	0.928
	Case 4	0.852	0.839	0.846	0.943
	Case 5	0.821	0.821	0.821	0.886
	Case 6	0.720	0.720	0.720	0.869
	Case 7	0.723	0.723	0.723	0.922
NCNN	Case 2	0.857	0.811	0.833	0.972
	Case 3	0.828	0.779	0.803	0.939
	Case 4	0.788	0.661	0.719	0.913
	Case 5	0.846	0.786	0.815	0.932
	Case 6	0.767	0.660	0.710	0.858
	Case 7	0.756	0.660	0.705	0.907
FNN	Case 2	0.662	0.608	0.634	0.744
	Case 3	0.567	0.559	0.563	0.702
	Case 4	0.672	0.661	0.667	0.771
	Case 5	0.589	0.589	0.589	0.710
	Case 6	0.587	0.540	0.563	0.725
	Case 7	0.674	0.660	0.667	0.753

TABLE 7: Performance of different deep learning methods in sixteen-classes diagnosis.

Sixteen		Precision	Recall	F1-score	AUC
GCNN	Case 2	0.746	0.690	0.717	0.993
	Case 3	0.761	0.723	0.742	0.985
	Case 4	0.765	0.709	0.736	0.973
	Case 5	0.801	0.745	0.772	0.969
	Case 6	0.705	0.576	0.634	0.935
	Case 7	0.672	0.516	0.584	0.933
WCNN	Case 2	0.878	0.869	0.873	0.979
	Case 3	0.834	0.831	0.832	0.978
	Case 4	0.796	0.783	0.789	0.957
	Case 5	0.815	0.815	0.815	0.944
	Case 6	0.774	0.765	0.769	0.917
	Case 7	0.757	0.723	0.739	0.910
NCNN	Case 2	0.757	0.676	0.714	0.975
	Case 3	0.797	0.708	0.749	0.978
	Case 4	0.679	0.570	0.619	0.953
	Case 5	0.645	0.545	0.591	0.921
	Case 6	0.684	0.535	0.601	0.892
	Case 7	0.655	0.477	0.552	0.875
FNN	Case 2	0.449	0.303	0.362	0.810
	Case 3	0.489	0.335	0.397	0.833
	Case 4	0.375	0.248	0.298	0.756
	Case 5	0.370	0.250	0.299	0.776
	Case 6	0.381	0.381	0.381	0.761
	Case 7	0.500	0.310	0.382	0.792

5. Conclusions

This paper proposes GCNN, an effective and efficient architecture for complex small sample size bearing fault diagnosis while considering imbalanced data. This model

benefits from the automatic feature learning ability of a CNN and the generalization capability of the skip connection that uses the features independently from the encode part to the decode part. Experiments are conducted on balanced and on imbalanced data to evaluate the performance of GCNN. Given that imbalanced data classification is an inherently difficult task, six different imbalance degree cases are used for verification in experiment 2. Results showed that GCNN outperforms all the other deep learning methods compared in this work. To sum up, GCNN has a robust feature representation capability and can use balanced or imbalanced data for either small or large classes fault diagnosis scenarios.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 51875376 and 51875375) and in part by the National Key R&D Program of China (Nos. 2017YFC0804803 and 2018YFB1600500).

References

- [1] Y. Qin, D. Chen, S. Xiang, and C. Zhu, "Gated dual attention unit neural networks for remaining useful life prediction of rolling bearings," *IEEE Transactions on Industrial Informatics*, vol. 99, p. 1, 2020.
- [2] G. Georgoulas, T. Loutas, C. D. Stylios, and V. Kostopoulos, "Bearing fault detection based on hybrid ensemble detector and empirical mode decomposition," *Mechanical Systems and Signal Processing*, vol. 41, no. 1-2, pp. 510-525, 2013.
- [3] Y. Qin, C. Li, F. Cao, and H. Chen, "A fault dynamic model of high-speed angular contact ball bearings," *Mechanism and Machine Theory*, vol. 143, p. 103627, 2020.
- [4] M. Cerrada, R.-V. Sánchez, C. Li et al., "A review on data-driven fault severity assessment in rolling bearings," *Mechanical Systems and Signal Processing*, vol. 99, pp. 169-196, 2018.
- [5] J.-Q. Sun, "Multi-Layer domain adaptation method for rolling bearing fault diagnosis," *Signal processing*, vol. 157, pp. 180-197, 2019.
- [6] J. Tian, C. Morillo, M. H. Azarian, and M. Pecht, "Motor bearing fault detection using spectral kurtosis-based feature extraction coupled with K-nearest neighbor distance analysis," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 3, pp. 1793-1803, 2016.
- [7] R. Jegadeeshwaran and V. Sugumaran, "Fault diagnosis of automobile hydraulic brake system using statistical features and support vector machines," *Mechanical Systems and Signal Processing*, vol. 52-53, pp. 436-446, 2015.
- [8] B.-S. Yang, X. Di, and T. Han, "Random forests classifier for machine fault diagnosis," *Journal of Mechanical Science and Technology*, vol. 22, no. 9, pp. 1716-1725, 2008.

- [9] W. Mao, L. He, Y. Yan, and J. Wang, "Online sequential prediction of bearings imbalanced fault diagnosis by extreme learning machine," *Mechanical Systems and Signal Processing*, vol. 83, pp. 450–473, 2017.
- [10] G. F. Bin, J. J. Gao, X. J. Li, and B. S. Dhillon, "Early fault diagnosis of rotating machinery based on wavelet packets-Empirical mode decomposition feature extraction and neural network," *Mechanical Systems and Signal Processing*, vol. 27, pp. 696–711, 2012.
- [11] M. He and D. He, "Deep learning based approach for bearing fault diagnosis," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 3057–3065, 2017.
- [12] X. Zhang, Y. Liang, J. Zhou, and Y. Zang, "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM," *Measurement*, vol. 69, pp. 164–179, 2015.
- [13] W. Lei, Z. Liu, M. Qiang, and Z. Xin, "Complete ensemble local mean decomposition with adaptive noise and its application to fault diagnosis for rolling bearings," *Mechanical Systems & Signal Processing*, vol. 106, pp. 24–39, 2018.
- [14] Z. Yuan, L. Zhang, L. Duan, and T. Li, "Intelligent fault diagnosis of rolling element bearings based on HHT and CNN," 2018.
- [15] R. Behnoud and S. Yi, "Deep learning for prognostics and health management: state of the art, challenges, and opportunities," *Measurement*, vol. 163, 2020.
- [16] D.-T. Hoang and H.-J. Kang, "A survey on Deep Learning based bearing fault diagnosis," *Neurocomputing*, vol. 335, pp. 327–335, 2019.
- [17] R. Zhang, Z. Peng, L. Wu, B. Yao, and Y. Guan, "Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence," *Sensors*, vol. 17, no. 3, p. 549, 2017.
- [18] J. Wang, S. Li, Z. An, X. Jiang, W. Qian, and S. Ji, "Batch-normalized deep neural networks for achieving fast intelligent fault diagnosis of machines," *Neurocomputing*, vol. 329, pp. 53–65, 2019.
- [19] C. Lu, Z.-Y. Wang, W.-L. Qin, and J. Ma, "Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification," *Signal Processing*, vol. 130, pp. 377–388, 2017.
- [20] C. Shen, Y. Qi, J. Wang, G. Cai, and Z. Zhu, "An automatic and robust features learning method for rotating machinery fault diagnosis based on contractive autoencoder," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 170–184, 2018.
- [21] O. Janssens, V. Slavkovikj, B. Vervisch et al., "Convolutional neural network based fault detection for rotating machinery," *Journal of Sound and Vibration*, vol. 377, pp. 331–345, 2016.
- [22] L. Eren, T. Ince, and S. Kiranyaz, "A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier," *Journal of Signal Processing Systems*, vol. 91, no. 2, pp. 179–189, 2019.
- [23] N. V. Chawla, "Data mining for imbalanced datasets: an overview," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., Springer, Berlin, Germany, 2009.
- [24] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: a review," *International Journal of Advances in Soft Computing and its Applications*, vol. 5, no. 3, pp. 176–204, 2013.
- [25] Q. Xu, S. Lu, W. Jia, and C. Jiang, "Imbalanced fault diagnosis of rotating machinery via multi-domain feature extraction and cost-sensitive learning," *Journal of Intelligent Manufacturing*, vol. 31, no. 6, pp. 1467–1481, 2020.
- [26] Y. Zhang, X. Li, L. Gao, L. Wang, and L. Wen, "Imbalanced data fault diagnosis of rotating machinery using synthetic oversampling and feature learning," *Journal of Manufacturing Systems*, vol. 48, pp. 34–50, 2018.
- [27] W. Mao, Y. Liu, L. Ding, and Y. Li, "Imbalanced fault diagnosis of rolling bearing based on generative adversarial network: a comparative study," *IEEE Access*, vol. 7, pp. 9515–9530, 2018.
- [28] B. Zhao, X. Zhang, H. Li, and Z. Yang, "Intelligent fault diagnosis of rolling bearings based on normalized CNN considering data imbalance and variable working conditions," *Knowledge-Based Systems*, vol. 199, p. 105971, 2020.
- [29] X. Zhang, D. Jiang, T. Han, N. Wang, W. Yang, and Y. Yang, "Rotating machinery fault diagnosis for imbalanced data based on fast clustering algorithm and support vector machine," *Journal of Sensors*, vol. 2017, Article ID 8092691, 15 pages, 2017.
- [30] W. You, C. Shen, X. Guo, X. Jiang, J. Shi, and Z. Zhu, "A hybrid technique based on convolutional neural network and support vector regression for intelligent diagnosis of rotating machinery," *Advances in Mechanical Engineering*, vol. 9, no. 6, pp. 168–176, 2017.
- [31] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings*, pp. 1–15, San Diego, CA, USA, May 2015.
- [32] L. Gautheron, A. Habrard, E. Morvant, and M. Sebban, "Metric learning from imbalanced data with generalization guarantees," *Pattern Recognition Letters*, vol. 133, pp. 298–304, 2020.
- [33] W. You, C. Shen, D. Wang, L. Chen, X. Jiang, and Z. Zhu, "An intelligent deep feature learning method with improved activation functions for machine fault diagnosis," *IEEE Access*, vol. 8, no. 1, pp. 1975–1985, 2020.
- [34] M. Abadi, "TensorFlow: large-scale machine learning on heterogeneous distributed systems," 2015, <http://arxiv.org/abs/1603.04467>.
- [35] F. Chollet and others, Keras, 2015, <https://keras.io>.