BEARING FAULT DIAGNOSIS USING DEEP LEARNING NEURAL NETWORKS WITH INPUT PROCESSING

by

Yuanyang Cai

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

Department of Electrical and Computer Engineering

Hammond, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Lizhe Tan, Chair

Department of Electrical and Computer Engineering

Dr. Constantin Apostoaia

Department of Electrical and Computer Engineering

Dr. Colin P Elkin

Department of Electrical and Computer Engineering

Approved by:

Dr. Lizhe Tan

ACKNOWLEDGMENTS

I would like to thank the committee chair, Dr. Tan Lizhe, for his guidance and meticulous care during the year and a half of the research process, especially during the special period of the global epidemic. I have learned how to become a better researcher while focusing on work ethic and professionalism. Dr. Li-Zhe Tan's overwhelming achievements and contributions to the scientific community showed me a different perspective on possibility with dedication.

I would also like to thank the committee members, Dr. Constantin Apostoaia and Dr. Colin P Elkin, for their endless patience and commitment in helping me throughout the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional neural network
LSTM	long-short term memory
STFT	Short-time Fourier transform
WPT	Wavelet packet transform
EMD	Empirical mode decomposition
CWRU	Case Western Reserve University
MCSA	Motor current signature analysis
ANN	Artificial neural networks
PCA	Principal component analysis
SVM	Support vector machines
K-NN	K-nearest neighbors
AE	Auto-encoder
RNN	Recurrent neural network
GAN	Generative adversarial network
WNN	wavelet neural network
RUL	Remaining useful life
EDM	Electro-discharge machining
IR	inner raceway
OR	outer raceway
IMF	Intrinsic mode functions
ML	Machine learning
DL	Deep learning
DFT	Discrete Fourier transform
HP	Horsepower
DE	Drive End
STD	Standard Deviation
MSE	Mean Square Error

ABSTRACT

The roller bearings are widely used in aviation cargo systems, engines, agriculture, heavy equipment and machinery, solar panels, medical equipment, automobile industry, powerhouses, and many others. Bearing faults during the operation process will result in downtime, economic loss, and even human injury. To prevent these from happening, rolling bearing fault diagnosis has become a mature discipline. Deep learning networks have been known as effective methods for bearing fault diagnoses. Deep learning neural networks such as the convolutional neural network (CNN) use the images as inputs. In contrast, the others, such as long-short term memory (LSTM), may apply data sequences as inputs. This thesis research work focuses on performance evaluations of deep learning networks according to the classification accuracy by utilizing various signal transforms to form the network inputs. CNN and LSTM are adopted as our deep learning network structures. Besides raw data, the algorithms for processing input signals include short-time Fourier transform (STFT), Cepstrum, wavelet packet transform (WPT), and empirical mode decomposition (EMD). In addition, this paper also applies three commonly used machine learning algorithms for comparison, namely K nearest neighbor (KNN), support vector machine (SVM), and random forest (RF). Finally, a one-dimensional CNN structure is designed and implemented. Our simulations validate the effectiveness for each network input formulation based on the Case Western Reserve University (CWRU) bearing dataset.

1. INTRODUCTION

1.1 Literature Review

The fault diagnosis of rotating machinery has been of great practical significance in industrial applications to avoid economic losses and enhance machine availability [1]. The vibration signal preprocess is essential in rolling bearing fault diagnosis and is most widely used. During the bearing operation process, the force transmitted from other parts to the bearing is unstable and changes with time. Hence, any precision bearing produces vibration signals. The contact force of a properly working bearing is time continuous in a normal condition. However, if bearing wear and scratches occur, they will generate periodic impulsive vibration signals different from those in a normal operation. These vibration signals can be captured via sensors and analyzed using signal processing techniques to extract signal features [2]-[10].

For vibration signal processing, the methods in time, frequency, and time-frequency domains are usually used to extract the sensitive characteristic index of the vibration signal. By feature extraction, one can identify the fault type more accurately. At present, bearing fault diagnosis methods are usually divided into three categories.

a. Conventional model based: vibration analysis, motor current signature analysis (MCSA), and principal component analysis (PCA).
b. Machine learning based: support vector machines (SVM), K-nearest neighbors (K-NN), and random forest (RF).
c. Deep learning based: Artificial neural networks (ANN), auto-encoder (AE), deep belief network (DBN), generative adversarial network (GAN), wavelet neural network (WNN), convolutional neural network (CNN), and recurrent neural network (RNN).

Among those machine-learning approaches, deep learning networks [11]-[16] have demonstrated significant efficiency with a high classification rate. However, most of the research focuses on improving network structures and the classification rate with less emphasis on the effects of processing signals for network inputs. In this thesis research, the short-time Fourier transform (STFT), Cepstrum, wavelet packet transform (WPT), and empirical mode decomposition (EMD) techniques are applied. Through the preprocessing and feature extraction of sensor signals using signal processing techniques to formulate feasible inputs for deep learning

networks, the condition of fault diagnosis can be evaluated. This thesis research will focus on two popular structures: 1D and 2D convolutional neural network (CNN) and long-short term memory (LSTM) and validate their performances in terms of accuracy and stability using various signal processing methods.

1.2 Research and Motivation

Fault diagnosis can usually be made through traditional signal processing techniques as well as artificial intelligence methods. So far, the application of deep learning networks in this field has been studied extensively and proved to be very effective. However, most of the research focuses on the modification of network structures for the performance improvement of the classification rate with less concern on the effects of processing signals for network inputs. Therefore, how deep learning networks respond to different signal processing of inputs still leaves room for us to investigate.

1.3 Organization of Thesis

This thesis work is organized as follows. Chapter 2 introduces the concept of the rolling bearing fault and the dataset used in the experiment. Chapter 3 describes five methods of preprocessing input signals. Chapter 4 applies the commonly used machine learning algorithm. Chapter 5 depicts several fault detection methods based on two deep learning networks combing with five signal processing techniques and evaluates each combination's results. Chapter 6 explores and demonstrates the results of machine fault detection using a one-dimensional convolutional neural network. Chapter 7 presents the conclusions and future work.

1.4 Contribution of Thesis

a. In this thesis work, various machine learning methods are studied and applied in bearing fault detection as the baseline.
b. Throughout the evaluation of deep learning neural networks with input processing for bearing fault diagnosis, a classification method (STFT-CNN and STFT-LSTM) with the highest accuracy is validated.

c. The feasibility of machine fault classification using a one-dimensional (1D) convolutional neural network is explored in order to reduce computational complexity.

2. BEARING FAULT DATASET

Data is the foundation for all the Artificial Intelligent algorithms. A good collection of datasets is of great importance to develop effective Machine learning (ML) and Deep learning (DL) algorithms for bearing fault diagnosis. Since the natural bearing degradation is a gradual process and may take many years, most people conduct the experiment and collect data either using bearings with artificially induced faults or with accelerated life testing methods. Common datasets currently available on the network include the Paderborn University dataset and the Intelligent Maintenance System (IMS) dataset, often used for bearing remaining useful life (RUL) predictions. In addition, there are PADERBORN dataset and Case Western University Dataset especially suitable for bearing fault detection.

2.1 Case Western Reserve University Bearing Dataset

In this work, the vibration data files from the Case Western Reserve University Bearing (CWRU) bearing fault dataset serves as a fundamental dataset to validate the performance of different ML and DL algorithms, are adopted, which were recorded and made publicly available on the CWRU bearing data center website [17], [18]. Figure 2-1 describes the experimental platform.

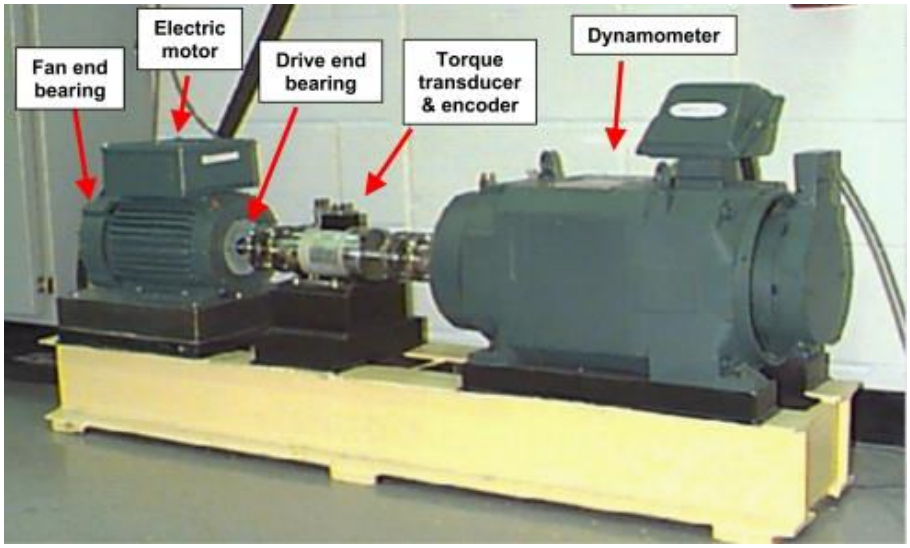

Figure 2-1 Experimental setup for CWRU bearing dataset [17]

The bearings support the motor shaft. The single-point artificial faults were seeded in the bearing's inner raceway (IR), outer raceway (OR), and rolling element (Ball) with an electro-discharge machining (EDM) operation [19]. Figure 2-2 depicts the structure of a rolling element bearing.

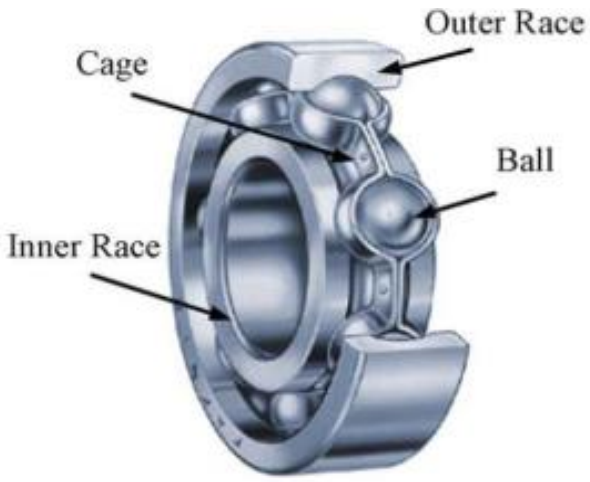

Figure 2-2 Structure of a rolling element bearing [19]

In our experiments for validation, we chose the vibration data files produced using three horsepower (HP) engine running at 1772 RPM with the motor shaft bearings having faults in depth from 0.000 (none), 0.007, 0.014, 0.021, and 0.028 inches and various fault locations (inner race, rolling element, and outer race). In the CWRU experiment, an accelerometer was attached at the 12'clock position at the motor housing's drive end (DE) to measure the vibrations. The acquired signals were sampled at 12 kHz. The data files contain one normal class and 11 single point fault classes after merging outer race positions relative to the load zone shown in Table 2-1.

Table 2-1 Classes in CWRU Bearing Dataset

Fault depth	Fault name	Class Name Used
0.028	0.028-Ball	No.1 (28-Ball)
	0.028-InnerRace	No.2 (28-IR)
0.021	0.021-Ball	No.3 (21-Ball）
	0.021-InnerRace	No.4 (21-IR）
	0.021-OuterRace12 0.021-OuterRace6 0.021-OuterRace3	No.5 （21-OR）
0.014	0.014-Ball	No.6 （14-Ball）
	0.014-InnerRace	No.7 （14-IR）
	0.014-OuterRace6	No.8 （14-OR）
0.07	0.007-Ball	No.9 （07-Ball）
	0.007-InnerRace	No.10 （07-IR）
	0.007-OuterRace3 0.007-OuterRace6 0.007-OuterRace12	No.11 （07-OR）
0	Normal	No.12 （Normal)

2.2 Experiment Data Setup

As shown in Table 2-1, there are 12 data categories in the experiments. For the training set, we used 180 groups of data for each type. For the category of outer race fault at different locations, we selected 60 groups for locations 3:00, 6:00, and 12:00, respectively, and merged them into one type.

For the testing set, we set 60 groups of data for each category. For the type of outer race fault with different locations, we chose 20 groups for locations of 3:00, 6:00, and 12:00, respectively, and merged them into one category. The ratio of the training set to the test set is 3:1. To adapt the input data to the convolutional neural network, the length of each sample data is 64x64, and there are totally 180x12 training samples and 60x12 testing samples, as shown in Figure 2-3.

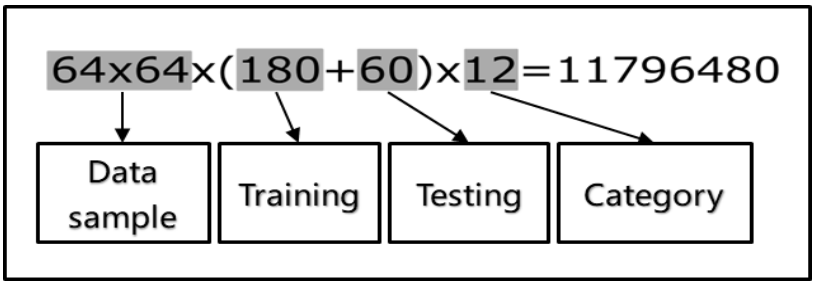

Figure 2-3 Selection of data

3. SIGNAL PREPROCESSING TECHNIQUES

3.1 Overall Steps of Signal Processing

When vibration signals are acquired via data acquisition, they need to be preprocessed to generate network inputs for training and testing. This process consists of random signal segmentation, transformation, and packing. We will explain how to convert the raw signal into the form of data that the deep learning networks can accept.

Figure 3-1 depicts the procedure to generate a training or a testing sample. It takes three steps.

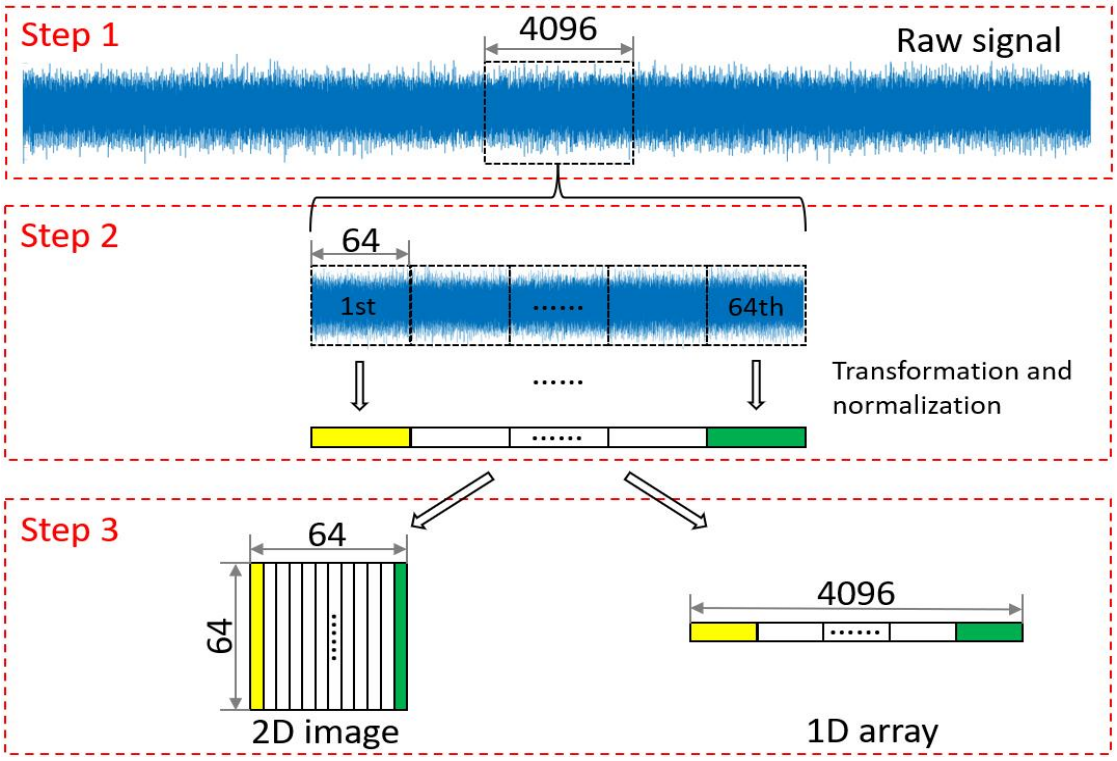

Figure 3-1 Transformed image and data sequence

Step 1: A raw signal is randomly segmented into a set of signal frames each with a size of $W \times W$ data samples. The random signal segmentation produces multiple random starting points and thus offers flexibility to generate sufficient signal frames.

Step 2: After signal segmentation, each frame is further split into W pieces of data blocks.

Step 3: For a deep learning network such as the 2-dimensional (2-D) CNN network, W raw data blocks or W transformed data blocks are normalized and stacked into 2-D images for the network inputs. For the LSTM network, W raw data blocks or W transformed data blocks are cascaded into an array to form the network input. In addition, the one-dimensional preprocessing signal can also be used as the input of the machine learning algorithm and one-dimensional CNN algorithm.

3.2 Signal Processing Methods

3.2.1 Cepstrum

Cepstrum analysis [20] is a nonlinear signal processing method that is defined as the inverse of discrete Fourier transform (DFT) of the logarithm of the absolute value of the DFT of the input signal. The Cepstrum is defined as follows.

$$\hat{x}_n = \mathrm{Re}\left\{\frac{1}{W}\sum_{k=0}^{W-1}\log_{10}|X(k)|e^{-2\pi kn/N}\right\} \quad n=0,1,2,...,W-1 \tag{3.1}$$

where $X(k)$ is the discrete Fourier transform of the signal $x(n)$.

This method is convenient for extracting and analyzing the periodic signals, which are difficult to recognize in the original spectrum. The Cepstrum indicates the concentration of the energy of the periodic frequency structure component on the original spectrum, and it gives higher weight to the low amplitude component in the logarithmic conversion of power, and lower weight to the high amplitude component, so the small-signal period is highlighted.

In the actual work, the failure of rotating machinery may be accompanied by the generation of the side frequency band. It is difficult to see the sideband structure in a complex spectral environment when multiple sidebands cross each other. Generally, it is impossible to quantitatively estimate the overall level of the side frequency in the power spectrum. Still, the Cepstrum can clearly show the periodic component in the power spectrum, so the original group's side frequency band spectral lines are simplified to a single spectral line, which is easy to observe. Meanwhile, the Cepstrum is not affected by the location of the sensor and the transmission path. For sensors arranged in different positions, their power spectra are different due to different transfer paths. However, the vibration effect of the signal source is separated from the effect of the transmission

path in the Cepstrum. In Cepstrum analysis, the attenuation and the calibration coefficient of the signal can be eliminated. This advantage is beneficial for fault identification.

3.2.2 Wavelet Packet Transform

Wavelet decomposition (wavelet transform) [21] decomposes a signal into low-frequency band A1 and high-frequency and D1. In the decomposition, the information lost in low-frequency information A1 is captured by high-frequency D1. Wavelet Packet Transform (WPT) method is a generalization of WT that offers more signal analysis. Wavelet packet atoms are waveforms indexed by three naturally interpreted parameters: position, scale, and frequency. The wavelet packets can be used for numerous expansions of a given signal. We then select the most suitable decomposition of a given signal with respect to an entropy-based criterion. Compared to WT, WPT decomposes not only the low-frequency part but also the high-frequency part. As shown in Figure 3-2, at the level-2 decomposition, A1 is further decomposed into two parts: low-frequency components of AA2 and high-frequency components of DA2. Note that the D1 signal undergoes a similar process to achieve AD2 and DD2. Therefore, wavelet packet decomposition is a more widely used wavelet decomposition method applied to signal decomposition, coding, denoising, compression, and other aspects.

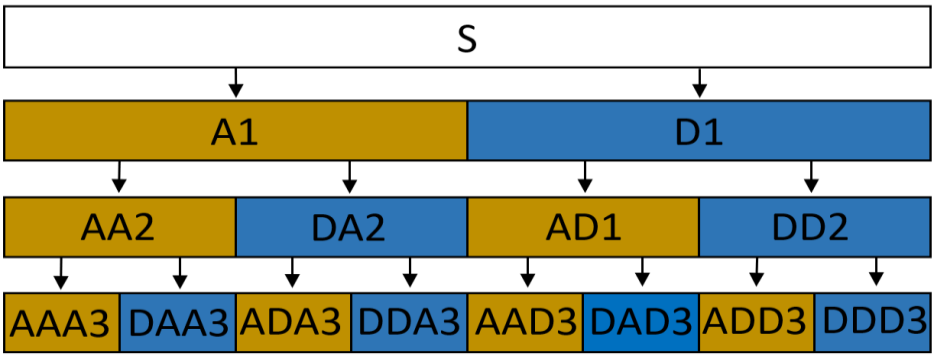

Figure 3-2 WPT framework

Two filters, corresponding to the wavelet, are the low-pass filter of $h(k)$ and high-pass filter of $g(k)$ each with a length of N, are given as

$$W_{2m}(n) = \sqrt{2}\sum_{k=0}^{2N-1} h(k)W_m(2n-k) \tag{3.2}$$

$$W_{2m+1}(n) = \sqrt{2}\sum_{k=0}^{2N-1} g(k) W_m(2n-k) \tag{3.3}$$

where m is designated as the decomposition level. Equation (3.2) and (3.3) represent the wavelet packet of low-frequency and high-frequency components, respectively. Note that $W_m(n), m = 0,1,2,...$ and $W_0(n) = x(n)$.

For our study, the wavelet packet transform (WPT) is used to transform each data block. Figure 3-3 displays a dyadic decomposition used.

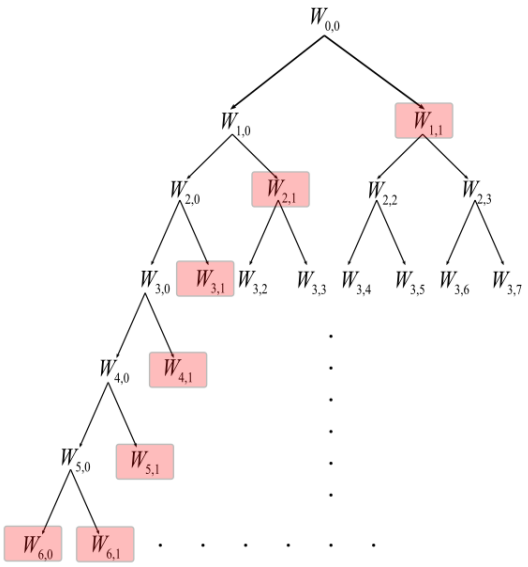

Figure 3-3 Wavelet packet selection

Compared with Figure 3-2, the WPT continues to decompose the lower frequency band and keep the higher frequency band at each level. In our study, Figure 3-3 shows the decomposition at level 6. The red boxes indicate the wavelet packet we need. Table 3-1 further lists the details of a WPT transformed data block with a size of 64 (wavelet coefficients).

Table 3-1 Number of Coefficients in Each Wavelet Packet

WP	$W_{1,1}$	$W_{2,1}$	$W_{3,1}$	$W_{4,1}$	$W_{5,1}$	$W_{6,0}$	$W_{6,1}$
No.	32	16	8	4	2	1	1

3.2.3 Empirical Mode Decomposition

The empirical mode decomposition (EMD) is an adaptive signal stabilization processing method proposed by Huang N E [22] in 1998. It can decompose the fluctuation or trend of different scales in the signal step by step as well as subdivide the signal into a finite number of intrinsic mode functions (IMF). This method is suitable for nonlinear and non-stationary signals. An IMF is an oscillatory function with an equal number of extrema and zero crossings while having envelopes symmetrical with respect to zero [23]. There are two conditions that an IMF must satisfy: (1) In the whole data set, the number of extrema and the number of zero-crossings must either be equal or differ at most by one. (2) At any point, the mean value of the envelope is defined by the local maxima, and the envelope defined by the local minima is zero.

Figure 3-4 shows an example in which a signal is decomposed into 9 IMFs with a display of the first 3 IMFs and a residue. The residue is essentially a portion from the original signal, which the EMD cannot further decompose.

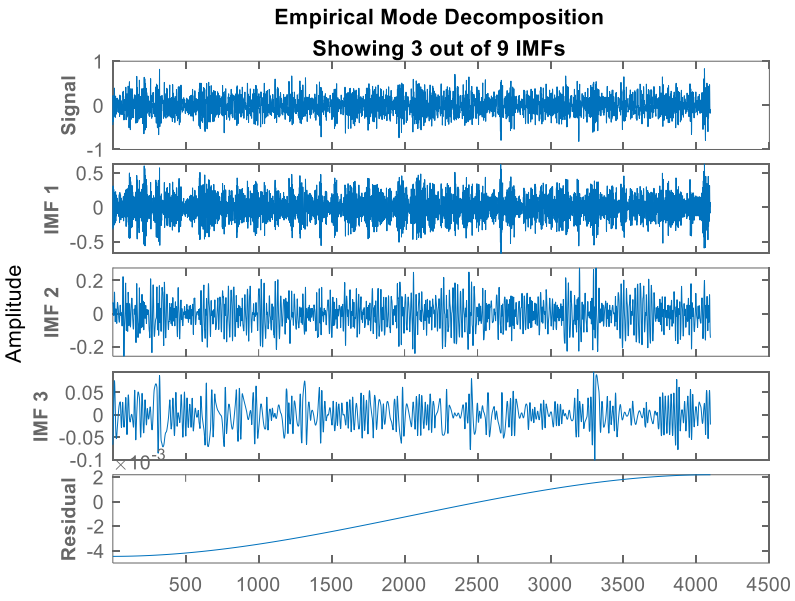

Figure 3-4 Composition of a signal after EMD

In our data formulation, the first 5 IMFs from decomposing a length of 4096 points raw data with by the EMD are obtained first. Among these 5 IMFs, the one having the largest cross-

correlation with the original raw signal is selected. Then the selected IMF is mixed with the original raw signal according to

$$y = (1-\lambda) * x + \lambda * IMF \tag{3.5}$$

where λ is chosen to be 0.3 in our study.

3.2.4 Short-time Fourier Transform

The Short-time Fourier transform (STFT) is a Fourier related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time [24]. In the discrete-time case, the data to be transformed could be broken up into frames (which usually have a fixed overlap). Each frame is Fourier transformed, and the complex result is added to a matrix, which records magnitude and phase for each point in time and frequency. This can be expressed as equation 4.5.

$$A_k = \frac{1}{W}\sum_{n=0}^{W-1}[x(n)w(n)]e^{-j2\pi kn/W} \quad k = 0,1,...,W-1 \tag{3.6}$$

where $x(n)$ is the original signal, $w(n)$ is the window function, which can reduce the leakage of the spectrum.

4. PERFORMANCE OF MACHINE LEARNING TECHNIQUES

Before applying the deep learning network, we tested several common machine learning algorithms and compared their performance in terms of classification accuracy. This chapter mainly introduces the following algorithms and tests the accuracy based on the one-dimensional 4096 STFT data.

4.1 K Nearest Neighbors (KNN)

The KNN algorithm is a nonparametric machine learning method used for classification in bearing fault detection [25]. In KNN classification, the output is the Class of an object, which is identified by a majority vote of its k nearest neighbors [26].

One early implementation of the KNN classifier on bearing fault diagnostics can be found in [27], where KNN acts as the ceramic bearing fault classifier using acoustic signals. After that, the other researchers [28]-[29] employ KNN to apply into a distance analysis on each new data sample and determine whether it belongs to a specific fault class.

The measurement of the distance between two points in the sample space indicates the degree of similarity between two sample points: the shorter the distance, the higher the degree of similarity; on the contrary, the degree of similarity is lower. To measure the distance between points *p* and *q* in a feature space, the Euclidean distance function is the most widely used one, which is shown (4.1). In Euclidean n-space, the distance between *p* and *q* is given by the training samples and its testing samples.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \,, \tag{4.1}$$

where $p = (p_1, p_2, p_3, ..., p_n)$, and *n* is the number of the space for a certain point.

Since each sample in the experimental data set belongs to a multi-space sample, it is difficult to display it through the figure. The following only uses a two-dimensional space data point double classification as an example.

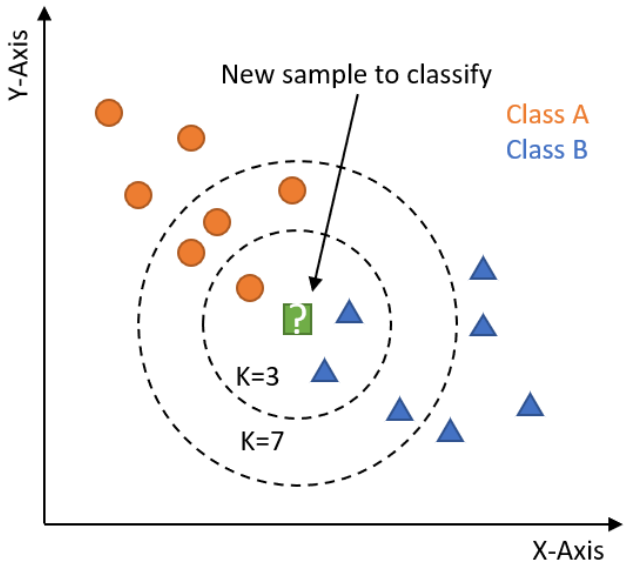

Figure 4-1 Two-classification

Figure 4-1 shows the two-class classification response for "categories" (A and B). If the k is set as 3, the three training samples nearest to the testing sample will be chosen. As indicated in Figure 3-1, there are one Class A sample and two Class B samples within the circled region. Thus, the testing sample belongs to Class B. Similarly, and when k is set as 7, the testing sample belongs to Class A.

The number of neighbors nearest to the new sample, k, is selected as an odd integer value because when an even number is chosen for k, there is a possibility of a tie, in which case it is possible or random to return a classification with uncertainty. In the classification phase, all test samples are unlabeled data. Each test sample will be assigned to its predicted label according to the most frequent training labels among k training samples closest to that test sample [30]-[32].

4.2 Support Vector Machine (SVM)

Support vector machine (SVM) is a supervised machine learning algorithm used as a margin classifier. The SVM finds an optimal hyperplane that separates data into two classes. The optimal hyperplane as depicted in Figure 4-2 is calculated to have the maximal margin (the distance from it to the nearest data point belonging to each lass) possible from patterns of each class. Most of the derivations in this section can be referred to [33]-[34].

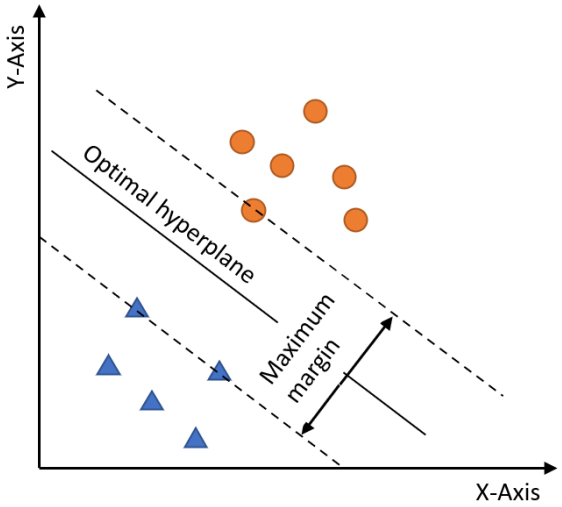

Figure 4-2 Optimal hyperplane

4.2.1 Hard Margin Problem

Assume training dataset D is linearly separable and belongs to two classes are given by:

$$D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{-1,1\}\}_{i=1}^{m} , \tag{4.2}$$

where x_i is the i^{th} training data vector with an n dimensions y_i represents the lass, and each data belongs to (-1 or 1). Any hyperplane can be written as the set of points x satisfying:

$$\omega \bullet x + b = 0, \tag{4.3}$$

In definition, ω is the weight vector and b is the bias. If the point (x, y) does not locate on the hyperplane, the value $\omega \bullet x + b$ could be positive or negative. For all the training data vectors, the goal is to obtain the closest point to the hyperplane. $\beta = |\omega \bullet x + b|$ can be calculated to find which point is closest to the hyperplane. This Equation is constrained by:

$$\min_{i=1...m} |\omega \bullet x + b| = 1. \tag{4.4}$$

In a canonical form, the separating hyperplane must satisfy the following constraints,

$$y_i(\omega \bullet x_i + b) \geq 1, i = 1,...,m \tag{4.5}$$

The distance from a point to a hyperplane is,

$$d = \frac{|\omega \bullet x + b|}{\|\omega\|} \tag{4.6}$$

The margin denoted by γ is given as,

$$\begin{aligned}\gamma &= \min_{x_i, y_i=1}(d) + \min_{x_i, y_i=-1}(d) \\ &= \min_{x_i, y_i=1}\left(\frac{|\omega \bullet x_i + b|}{\|\omega\|}\right) + \min_{x_i, y_i=-1}\left(\frac{|\omega \bullet x_i + b|}{\|\omega\|}\right) \\ &= \frac{2}{\|\omega\|}\end{aligned} \tag{4.7}$$

The optimal hyperplane is obtained by maximizing the margin γ, which is equivalent to minimizing $\frac{1}{2}\|\omega\|^2$ under the constraints.

The SVM optimization problem can be restated using the Lagrange multiplier method. When finding the minimum of f under the equality constraint g, the Equation of gradient is,

$$\nabla f(x) - \alpha \nabla g(x) = 0 \ , \tag{4.8}$$

where α can be defined as the Lagrange multiplier. In the SVM optimization problem, $f(\omega) = \frac{1}{2}\|\omega\|^2$, $g(\omega, b) = y_i(\omega \bullet x_i + b) - 1, i = 1, ..., n$. The Lagrangian function is expressed as,

$$L(\omega, b, \alpha) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\omega \bullet x_i + b) - 1] \ . \tag{4.9}$$

The dual problem can be introduced that Equation (4.9) must be minimized with respect to ω, b and be maximized with respect to α. The dual problem is shown as

$$\begin{aligned}&\min_{\omega, b} \max L(\omega, b, \alpha), \\ &\textit{subject to } \alpha_i \geq 0, i = 1...m .\end{aligned} \tag{4.10}$$

Taking the partial derivatives of $L(\omega, b, \alpha)$ to ω and b , respectively, and setting them to zero leads to:

$$\begin{aligned}\nabla_{\omega} L(\omega, b, \alpha) &= \omega - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \\ \nabla_{b} L(\omega, b, \alpha) &= -\sum_{i=1}^{m} \alpha_i y_i = 0\end{aligned} \tag{4.11}$$

Substituting (4.11) into (4.9), the dual problem can be given further by,

$$W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i x_j \tag{4.12}$$

The dual problem is thus stated as:

$$\max_{\alpha} W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i x_j \quad , \tag{4.13}$$

$$subject\ to\ \alpha_i \geq 0, i = 1...m, \sum_{i=1}^{m} \alpha_i y_i = 0 .$$

Note that the Lagrange multipliers method is extended to the KKT (Karush-Kuhn-Tucker) conditions because the constraints are inequalities. The complementary slackness condition of KKT conditions can be wittten as:

$$\alpha_i [y_i (\omega \bullet x^* + b) - 1] = 0 \tag{4.14}$$

where x^* are the points where we reach the optimal. The value of α is positive for these points. And the value of α value for the other points is close to zero. So $y_i(\omega \bullet x^* + b) - 1$ must be zero. These examples are called support vectors, which are the nearest points to the hyperplane.

Finally, ω and b can be computed as:

$$\begin{aligned} \omega &= \sum_{i=1}^{m} \alpha_i y_i x_i \\ b &= \frac{1}{S} \sum_{i=1}^{s} (y_i - \omega \bullet x) \end{aligned} \quad , \tag{4.15}$$

where S is the number of support vectors.

The classifier is then given as

$$f(x) = \text{sgn}(\omega \bullet x + b) \tag{4.16}$$

The Equation (4.16) is called Hard Margin SVM. The problem with Hard Margin SVM is that it does not tolerate outliers or work with non-linearly separable data because of outliers.

4.2.2 Soft Margin Problem

The problem with Hard Margin SVM is that it only works for linearly separable data. However, it is most likely that the data will contain some noise in practical cases and might not be linearly separable. The Soft Margin SVM will be introduced to solve this problem.

In Soft Margin SVM, the constraints are written as:

$$y_i (\omega \bullet x_i + b) \geq 1 - \xi_i, i = 1, ..., m \tag{4.17}$$

where ξ_i is a slack variable.

When minimizing the objective function, adding the slack variables makes it possible to satisfy the constraint even if the example does not meet the original constraint. When $\xi_i \geq 0$, the hyperplane is derived by:

$$\min_{\omega,b,\xi} \frac{1}{2} \| \omega \|^2 + C \sum_{i=1}^{m} \xi_i \tag{4.18}$$

$$subject\ to\ y_i(\omega \bullet x_i + b) \geq 1 - \xi_i, i = 1,...m, \xi_i \geq 0, i = 1,...m$$

where C is regularization parameter C to determine how the importance of ξ. Again, the optimization problem could be transformed into a dual problem as:

$$\max_{\alpha} W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i x_j \tag{4.19}$$

$$subject\ to\ 0 \leq \alpha_i \leq C, i = 1...m, \sum_{i=1}^{m} \alpha_i y_i = 0$$

Note that by using the Lagrange multipliers similar to the hard margin method, the above result can be obtained.

4.2.3 Kernel Trick

In some practical problems, it is often the case that the data is not linear, and a hyperplane cannot separate the datasets. The kernels are introduced to map the input data to a high-dimensional space non-linearly. The new mapping is then linearly separable. Now, let a kernel function defined as $K(x_i, x_j) = x_i \bullet x_j$, the dual problem can be rewritten as:

$$\max_{\alpha} W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{4.20}$$

$$subject\ to\ \alpha_i \geq 0, i = 1...m, \sum_{i=1}^{m} \alpha_i y_i = 0$$

Kernel trick (dot product) can transform the dimensionality of the data's feature space to define a similarity measure. For example, in Figure 4-3, the kernel trick increases the origin space (left) into a higher space (right), then the two classes become linearly separable. Multiple kernels could be used to classify the data. Some of the most popular ones are the linear kernel, polynomial kernel, and RBF kernel. In this test work, the linear kernel function is applied.

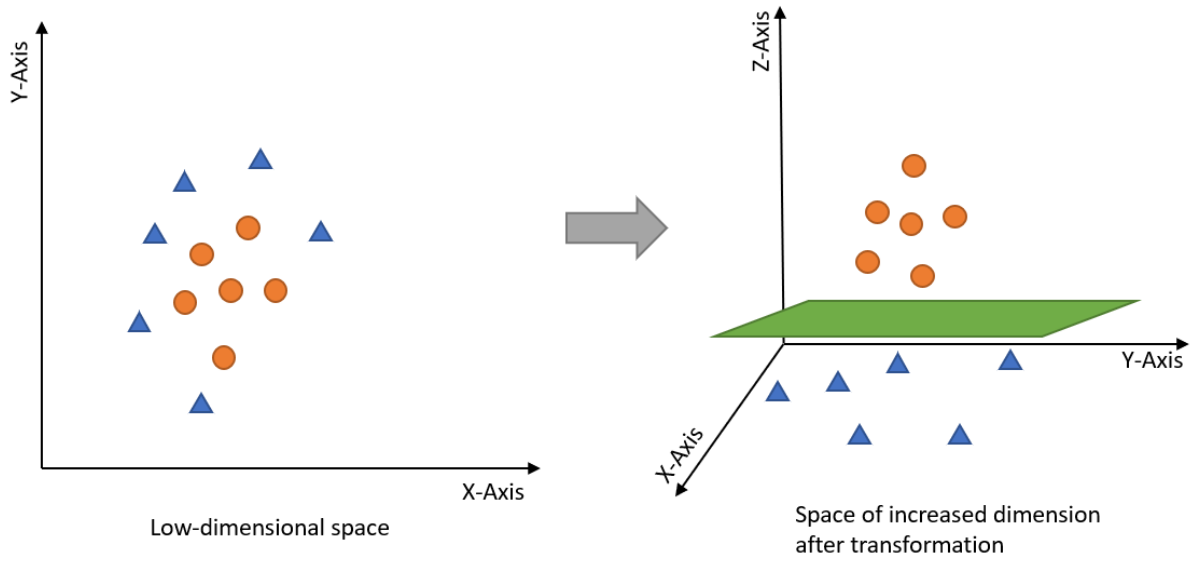

Figure 4-3 Importance of kernel trick

4.2.4 Multi-class SVM

Binary (two-class) classification using support vector machines (SVMs) is a well-developed technique. However, when applying the SVM to the problems with more than two classes, the better approach is to use a combination of several binary SVM. A multi-class problem can be decomposed into a series of two-class problems, which can be defined as the one-against-all methods. The one-versus-all method is used for distinguishing between one label and other labels. In this thesis work, twelve two-class classifiers need to be trained. Take an example. When the i^{th} sub-classifier is training, the samples belonging to the i^{th} class are labeled as positives while other samples are labeled as negatives. In the testing phase, all classifiers are applied to a test sample with an unknown class and predict its label, which is the most common among all classifier outputs. The function is given by

$$\hat{y} = \arg \max_{k \in \{1,\ldots,K\}} f_k(x), \tag{4.20}$$

where f_k are classifiers for $k \in \{1,\ldots,K\}$.

4.3 Random Forest

Random forest (RF) is one of the most used algorithms because of its simplicity. This section gives a brief overview of RF.

The 'forest', an ensemble of decision trees, is usually trained with the bagging method. The bagging method goes through various combinations of learning models and merges them to get better performance [35]. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each individual decision tree in the random forest will return a class prediction, and the class with the most votes becomes the model's prediction [36]. Each tree is grown as following rules [37]:

a. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
b. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M, and the best split on these M is used to split the node. The value of m is held constant during the forest growing.
c. Each tree is grown to the largest extent possible. There is no pruning.

To maximize the role of each model (tree) as a committee, we need to make sure that the behavior of each tree in a random forest has a low correlation with the other trees. It uses two methods that can be used:

a. Feature Randomness: Ensure that each tree in a random forest can only be selected from a random subset of features. Hence, there is more variation between trees in the model, leading to less correlation and greater diversity among trees.
b. Bagging (Bootstrap Aggregation): RF utilizes the sensitivity of decision tree structure to training data, allowing each tree to randomly extract replacement samples from the data set, resulting in different trees. This process is known as bagging. The general technique of bootstrap aggregating or bagging is applied to train the algorithm for random forests. Given the training set as $X = x_1, ..., x_N$ with the assigned classes, bagging can repeatedly select a random sample with replacement from a training set to these samples for N times.

4.4 Machine learning experiment results

4.4.1 Parameter Setting

According to our experiments, three machine learning methods as described in Sections 4.1-4.3 could not perform fault recognition satisfactorily using the raw fault signals without preprocessing. Therefore, one-dimensional STFT signals with a data length of 4096 were adopted in the ML experiments. The accuracy of each machine learning algorithm was obtained by averaging values from 10 independent runs. The specific parameter settings are described as follows. For the KNN algorithm, K was set to 3, and the distance criterion is Euclidean distance. For the Multi-SVM algorithm, the kernel function was selected as the linear kernel. For the Random Forest algorithm, the number of the decision tree was set to 100. The values of the fault classification accuracy are shown in Table 4-1.

Table 4-1 Accuracy of machine learning algorithms

Algorithm	Acc-max (%)	Acc-min (%)	Acc-avg (%)	STD
KNN	98.380	96.065	97.222	0.780
SVM	100	99.691	99.985	0.178
RF	100	98.611	99.537	0.577

It can be seen from the results that the signal after feature extraction can be used as the input of the machine learning algorithm to achieve relatively high accuracy, among which STFT-SVM has the highest precision and the best stability.

5. DEEP LEARNING NEURAL NETWORKS PERFORMANCE

5.1 Convolutional Neural Network

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and is used mainly for image processing, classification, segmentation, and other autocorrelated data. As one of the most effective deep learning models, CNN can complete the whole process of feature extraction, feature dimensionality reduction, and classifier classification through a neural network.

The CNN is a multistage neural network, including the filtering stage and classification stage. Among them, the filter stage is used to extract the features of input signals, the classification stage classifies the extracted features, and the network parameters of the two stages are obtained through joint training. The filter stage consists of three basic components: convolutional layer, pooling layer, and activation layer, while the classification stage is generally composed of the full connection layer. The purpose of these four layers can be described as:

a. Convolutional layer: the Convolutional layer uses the convolutional kernels to take convolution operations on the local region of the input signal and generate corresponding features. Weights sharing is the most important feature of the convolutional layer, that is, the same convolution kernel traverses the input once with a fixed stride. The first logits value is obtained by multiplying the corresponding coefficient of the convolution kernel and the neuron in the rolled region in the convolution process. Next, the convolution kernel is moved with step size, and the previous operation is repeated until the convolution kernel traverses all regions of the input signal.

b. Activation layer: the activation function non-linearly transforms the logits value of each convolution output. The purpose of the activation function is to map the original linear indivisible multidimensional feature to another space, in which the linear separability of the feature will be enhanced. The commonly used activation function in neural networks includes the sigmoid function, hyperbolic tangent function Tanh, and modified linear unit rectified linear unit (RELU).

c. Pooling layer: The pooling layer is used for downsampling, with the main purpose of reducing the neural network parameters. Pooling functions include average pooling and max pooling. Mean (average) pooling takes the mean value of neurons in the perception domain as the

output value, while maximum pooling takes the maximum value in the perception domain as the output value.

d. Full connection layer: the full connection layer classifies the features extracted from the filter stage. Specifically, the output of the last pooling layer is first spread out into a one-dimensional feature vector as the input of the full connection layer. The full connection layer is made of a full connection between the input and output.

In this test work, each of the input maps can be described as $W_1 \times W_1$ from the previous layer $(i-1)$ in convolutional or pooling layers. Hence, the output size after the layer i is

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \ , \tag{5.1}$$

where F is the size of kernel filters, S is the number of strides, and P is the number of padding zeros.

Figure 5-1 depicts the proposed Convolutional neural network (CNN) structure, and Table 5-1 lists the details.

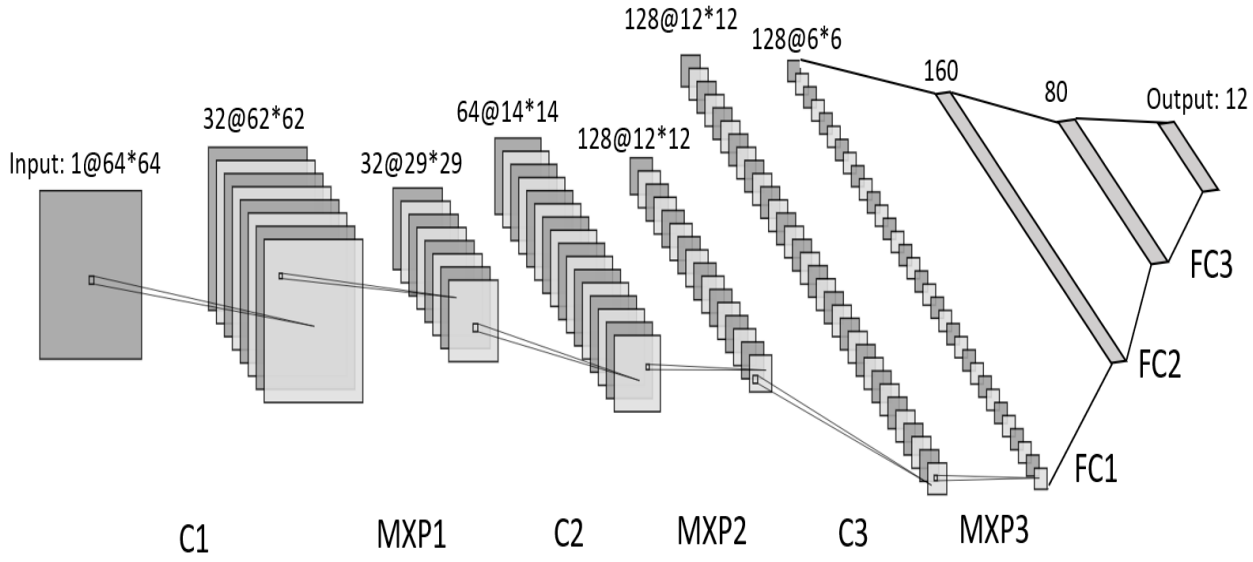

Figure 5-1 Structure of CNN

Table 5-1 Information of CNN Layers

Layer name	CNN Models (Kernel Filter, Activation Function, Strides, Padding)
C1	Conv (32*3*3, ReLU, 1, None)
MXP1	Maxpooling (2,2,1)
C2	Conv (64*3*3, ReLU, 1, None)
MXP2	Maxpooling (2,2,1)
C3	Conv (128*3*3, ReLU, 1, None)
MXP3	Maxpooling (2,2,1)
FC1	Fullyconnect (160, ReLU)
FC2	Fullyconnect (80, ReLU)
FC3	Fullyconnect (12, ReLU)
Output	Softmax, Classification

We adopt three successive convolutional and pooling layers to extract high-level features. The activation function of ReLU is used for each convolutional layer. The maximum pooling operation is applied to each poling layer. Finally, three fully connected layers, each with the ReLU activation function, are adopted. The classified output is achieved through a soft-max layer.

5.2 Long Short-Term Memory

The long short-term memory (LSTM) network is an artificial recurrent neural network (RNN) architecture used in deep learning. The RNN was firstly introduced [38] to solve time sequence learning problems. It applies the backpropagation algorithm for training. In principle, the RNN connects previous information to the present task. However, the RNN cannot capture the long-term dependencies, so it will face the problem of vanishing gradients in practice. Notice that the LSTM network is a special kind of RNN, capable of learning short-term dependencies [39]. A standard LSTM unit comprises a cell, an input gate, an output gate, and a forgetting gate. The cell

remembers values over arbitrary time intervals, whereas the three gates regulate the flow of information into and out of the cell. Figure 5-2 shows the components of the LSTM network.

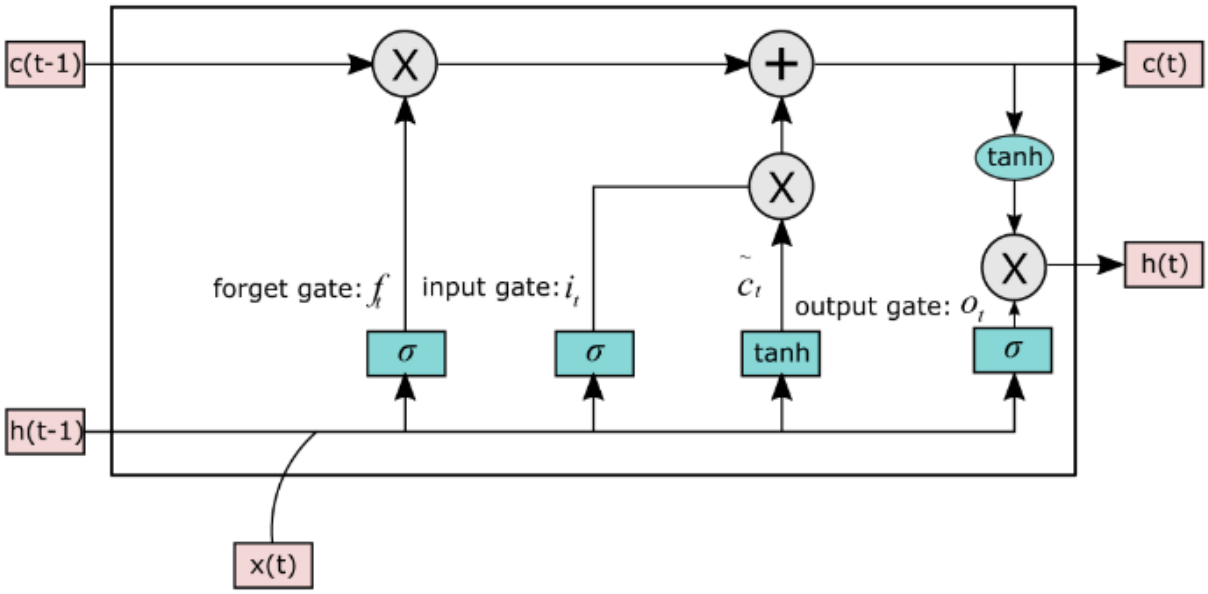

Figure 5-2 Structure of LSTM

Specifically, at each time step t, the hidden state h_t is updated by fusion of data at the same step x_t, input gate i_t, forgetting gate f_t, output gate o_t, memory cell $\tilde{c}_t$, and hidden state h_{t-1} at last time step. The updated equations are as follows:

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{5.2}$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{5.3}$$

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{5.4}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{5.5}$$

$$h_t = o_t \odot \sigma_h(c_t) \tag{5.6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5.7}$$

where W, U, and b are the weight matrices and a bias vector, respectively. $\tilde{c}_t$ is the cell input activation vector, while σ_c is the hyperbolic tangent function, σ_g is the sigmoid function. The operator $\odot$ denotes the Hadamard product.

Figure 5-3 displays the proposed LSTM frame, a 1-D feature vector with a size of 4096 is chosen as the input for the LSTM. Notice that the LSTM hidden layer has 500 units, and the rest includes a fully connected layer, soft-max layer, and classification layer to generate the classified output.

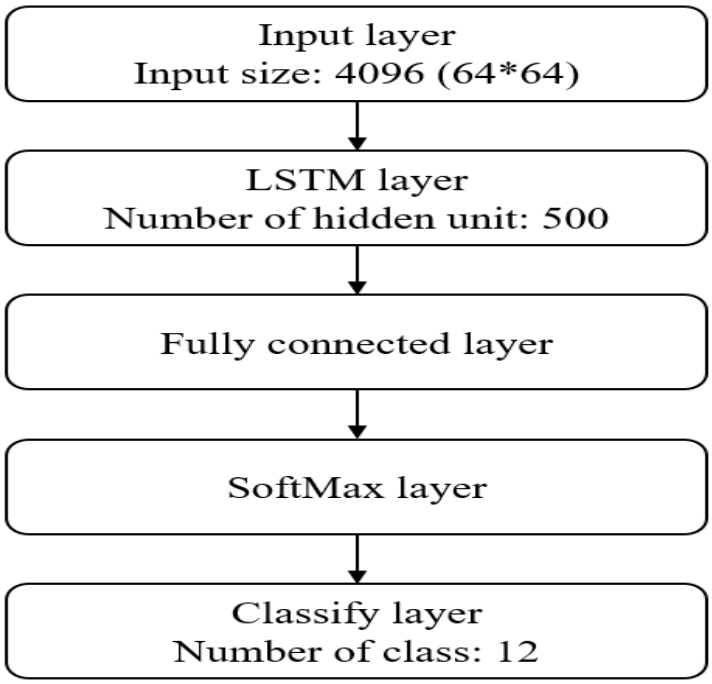

Figure 5-3 LSTM frame

5.3 Experiment Results

The proposed feature extraction methods and two fault detection models are built by using MATLAB R2019a. The training and testing process are all run by a laptop with a single NVIDIA 1070 Max-Q GPU.

The adaptive learning rate optimizer (Adam) with a learning rate is set to 0.0001 and categorical cross-entropy serves as a cost function for training the model. Each network will be trained and tested 10 times. This experiment will mainly focus on the classification accuracy in every type of fault. The final accuracy will be chosen based on the average accuracies.

Note that the numbers in the x and y label of the confusion matrix below can be matched to the type of fault in the 3rd column of table 2. The figures for the confusion matrix and training progress are both randomly selected from 10 trials.

To validate the feasibility of signal transformations, a raw data block was adopted for performance comparison.

After all the preparations are completed, we can begin the simulation validation.

5.3.1 Results for CNN

In CNN training model, the maximum number of epochs is set to 50, and the minimum batch size is set to 30 for all feature extraction methods.

(1) EMD-CNN

The average accuracy is 99.71%, and the STD is 0.202. It takes 2 mins and 4 sec to finish the training. Figures 5-4 and 5-5 show that the method has high classification accuracy and converges in about 27 epochs.

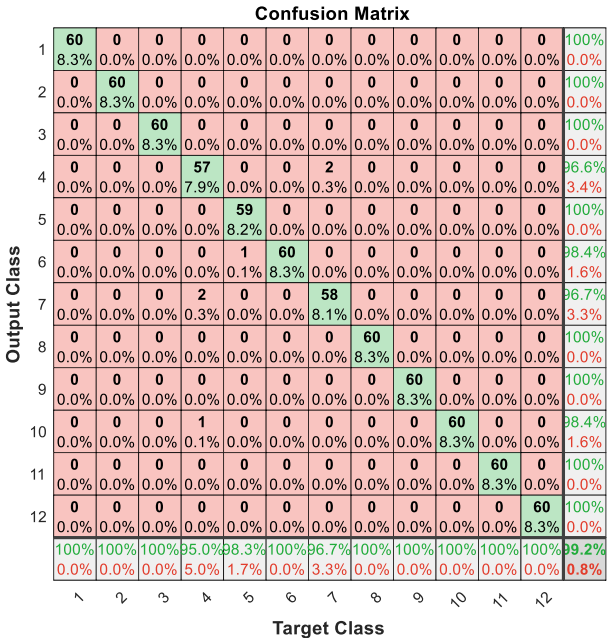

Figure 5-4 Confusion matrix for EMD-CNN

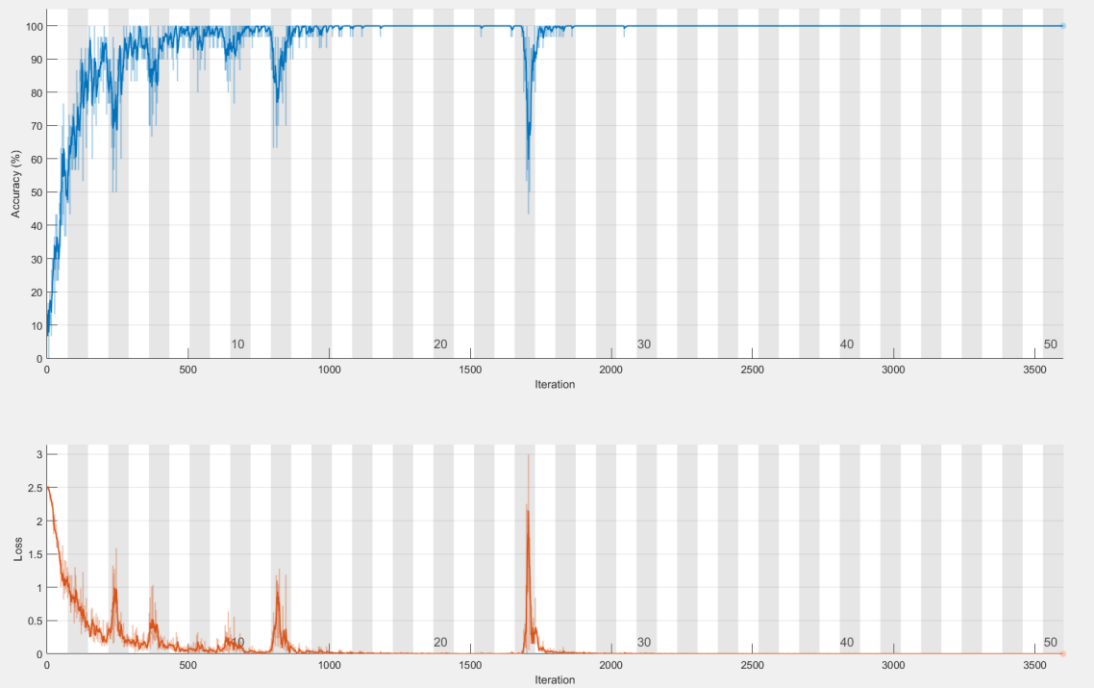

Figure 5-5 Training progress for EMD-CNN

(2) Cepstrum-CNN

The average accuracy is 99.16%, the STD is 0.111, and it will take 2 mins and 25 sec to finish the training. Figures 5-6 and 5-7 indicate that a small number of samples are misclassified, and the overall convergence speed is fast.

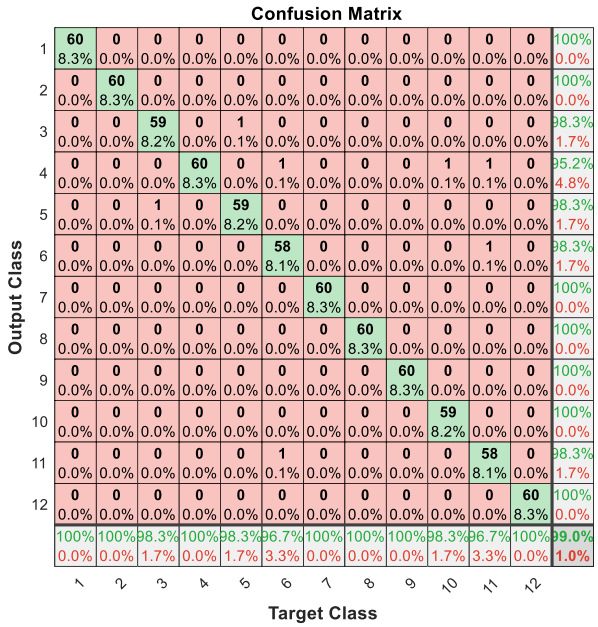

Figure 5-6 Confusion matrix for Cepstrum-CNN

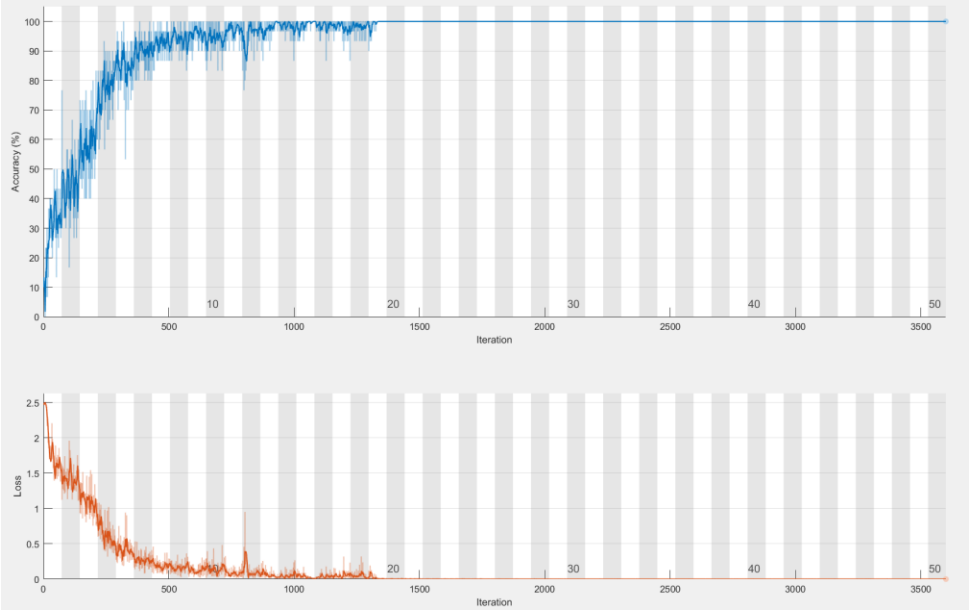

Figure 5-7 Training progress for Cepstrum-CNN

(3) WPT-CNN

The average accuracy is 99.9%, the STD is 0.077. Training time is 2 mins and 20 sec. The further results are shown in Figures 5-7 and 5-8.

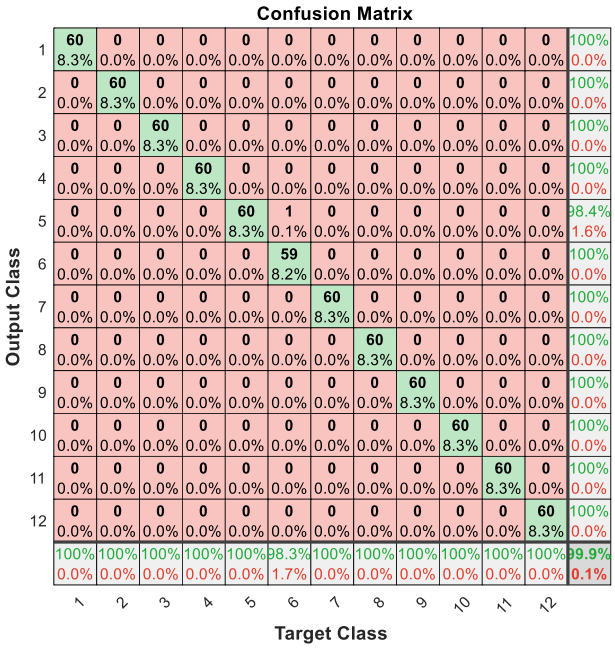

Figure 5-8 Confusion matrix for WPT-CNN

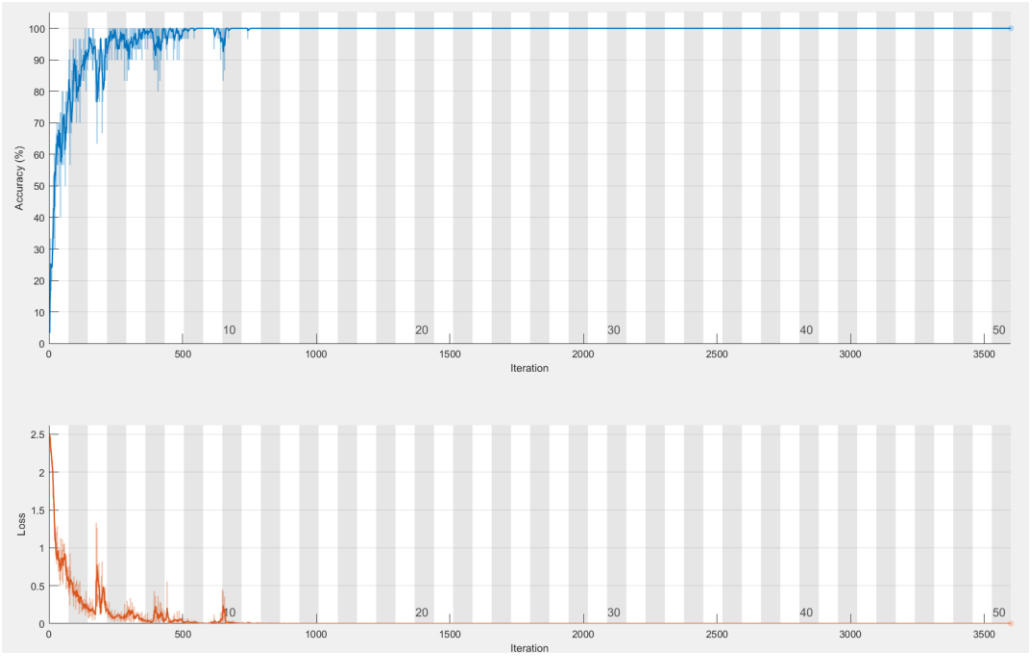

Figure 5-9 Training progress for WPT-CNN

(4) STFT-CNN

The average accuracy is 99.999%, the STD is 0.042, and it only takes 27 sec to finish the training. Figures 5-10 and 5-11 show that this method is accurate for each fault classification and has the fastest convergence speed.

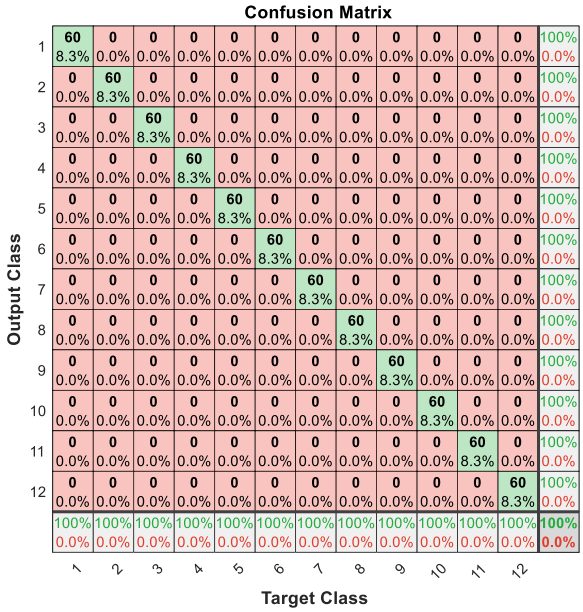

Figure 5-10 Confusion matrix for STFT-CNN

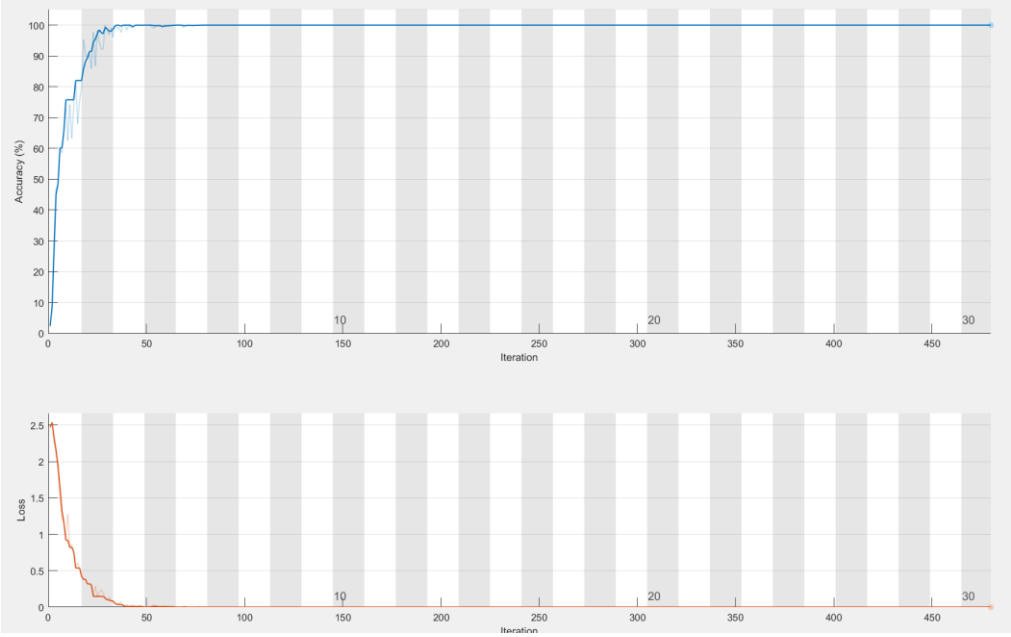

Figure 5-11 Training progress for STFT-CNN

(5) Raw-CNN

The average accuracy is 99.515%, the STD is 0.248, and it takes 44 sec to finish the training. The further results are shown in Figures 5-12 and 5-13.

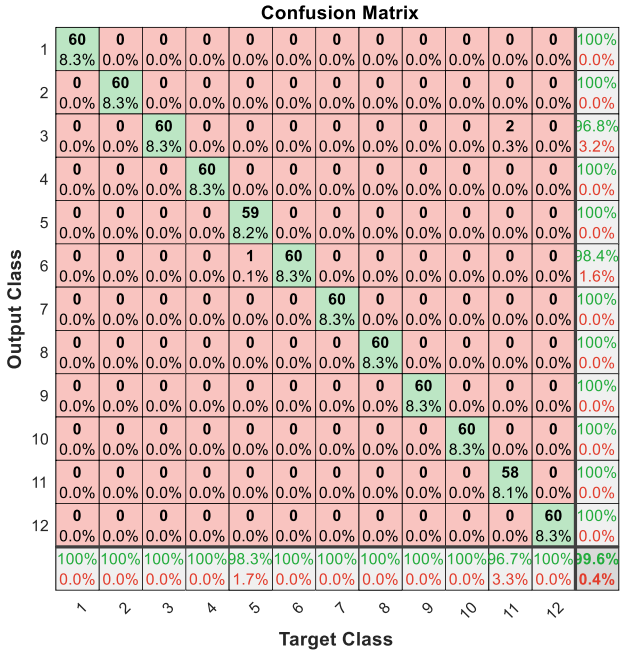

Figure 5-12 Confusion matrix for Raw-CNN

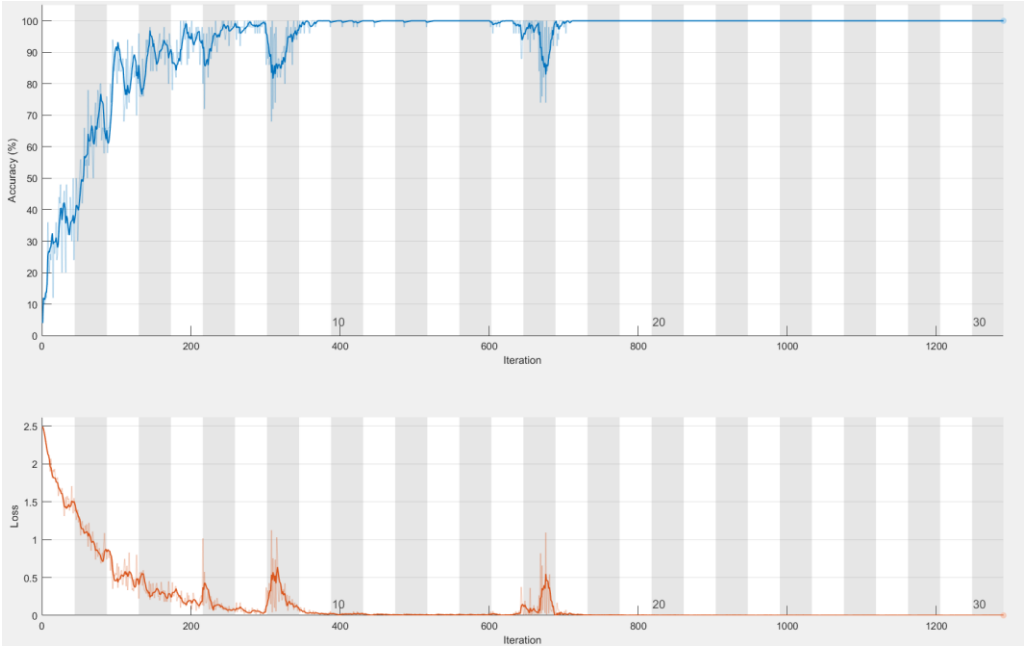

Figure 5-13 Training progress for Raw-CNN

(6) CNN results in comparison

Based on the accuracy in Table 4-3 and combined with STD and time consumption, STFT-CNN is the best CNN classification model. Every method except for Cepstrum has some progress in accuracy compared to the Raw-CNN. The Cepstrum-CNN has the disadvantage to adopt due to the cost of formulating input data while obtaining lower classification accuracy.

Table 5-2 CNN Classification Accuracy

Classifier Network	Acc-max(%)	Acc-min(%)	Acc-avg(%)	STD
Raw-CNN	100	99.31	99.515	0.248
STFT-CNN	100	99.986	99.999	0.003
Cepstrum-CNN	99.3	99	99.16	0.111
WPT-CNN	100	99.7	99.9	0.077
EMD-CNN	100	99.4	99.71	0.202

5.3.2 Results for LSTM

The maximum number of epochs are set to 100, and the minimum batch size is set to default (128) for all feature extraction methods. The gradient is clipped to 1 if it exceeds the value of 1.

(1) Cepstrum-LSTM

The average accuracy is 90.946%, the STD is 0.464, and it will take 1 min 40 sec to finish the training. The further results are shown in Figures 5-14 and 5-15.

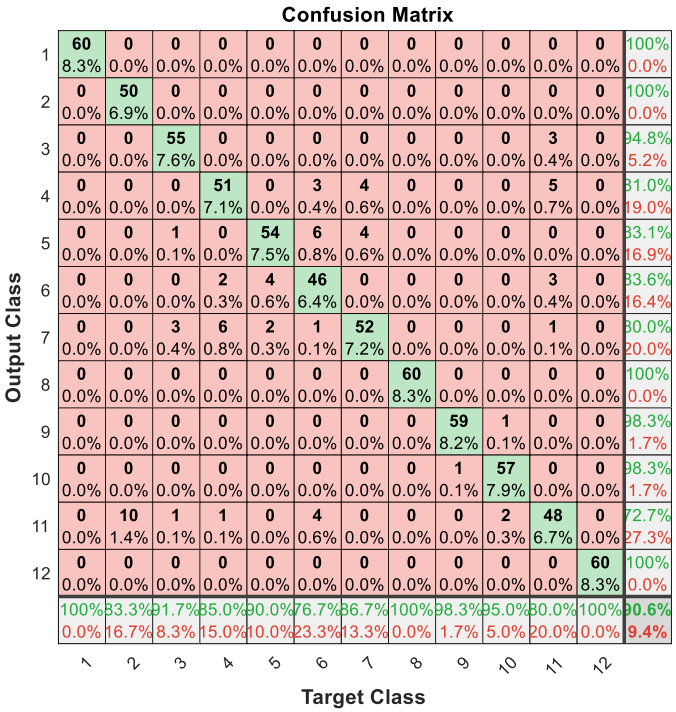

Figure 5-14 Confusion matrix for Cepstrum -LSTM

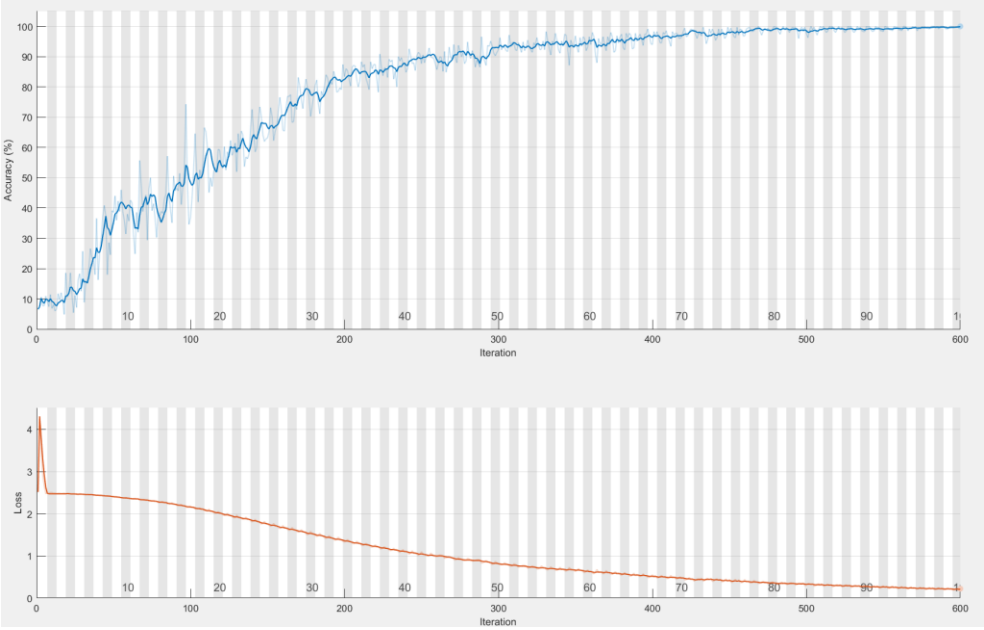

Figure 5-15 Training progress for Cepstrum-LSTM

(2) WPT-LSTM

The average accuracy is 93.48%, the STD is 0.611, and it will take 1 min 43 sec to finish the training. The further results are shown in Figures 5-16 and 5-17.

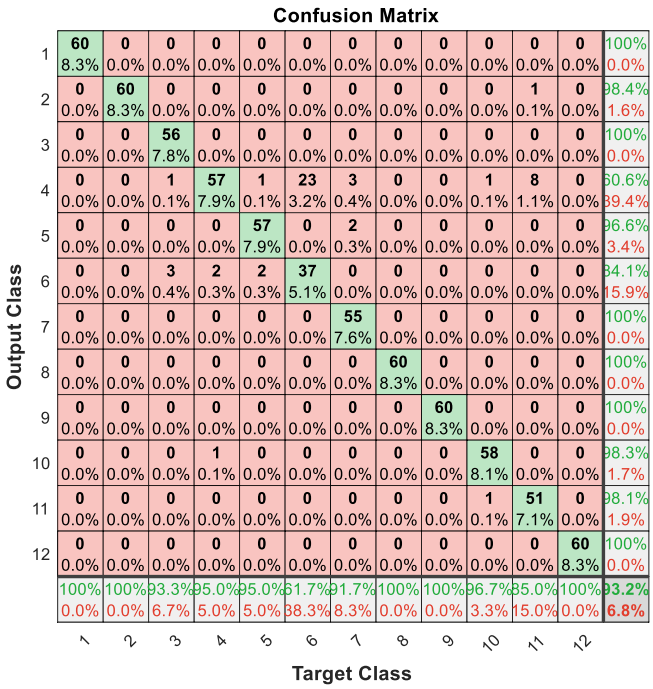

Figure 5-16 Confusion matrix for WPT-LSTM

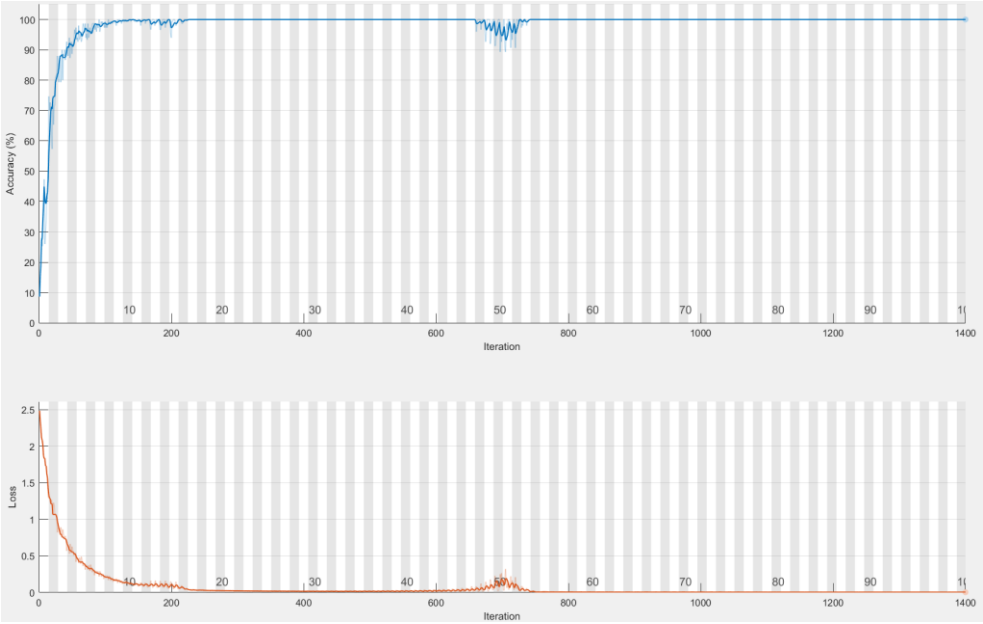

Figure 5-17 Training progress for WPT-LSTM

(3) STFT-LSTM

The average accuracy is 100%, the STD is 0, and it will take 1 min 40 sec to finish the training. The further results are shown in Figures 5-18 and 5-19.

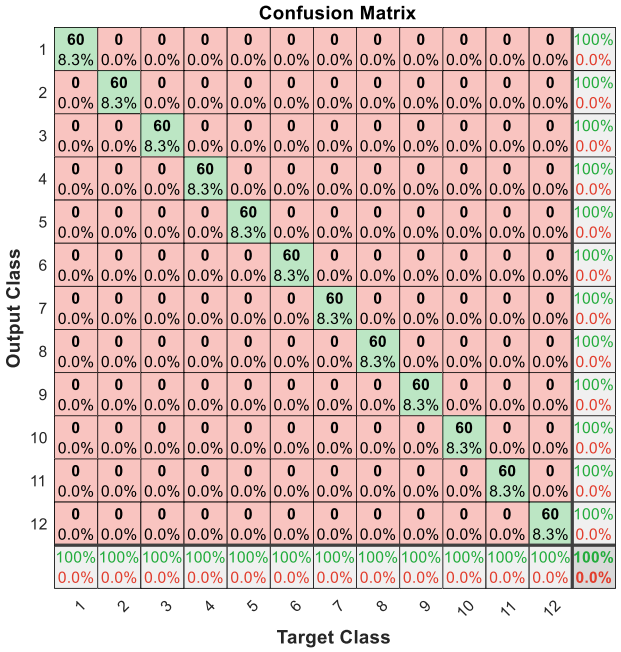

Figure 5-18 Confusion matrix for STFT-LSTM

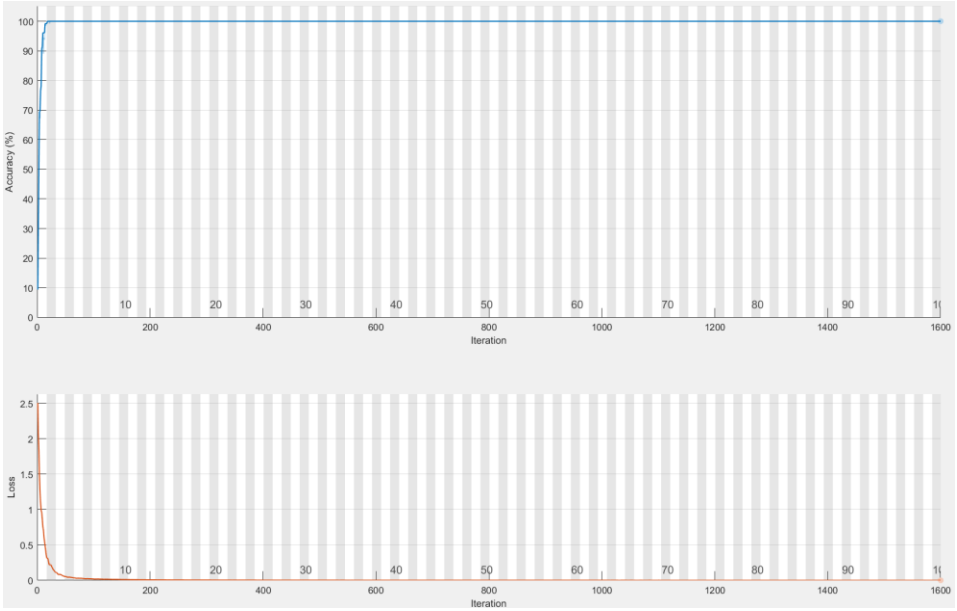

Figure 5-19 Training progress for STFT-LSTM

(4) Raw-LSTM

The average accuracy is 63.914%, the STD is 0.645, and it will take almost 12 mins to finish the training. The further results are shown in Figures 5-20 and 5-21. This combination has low test accuracy, but the accuracy of the training set is high, which is a situation of overfitting.

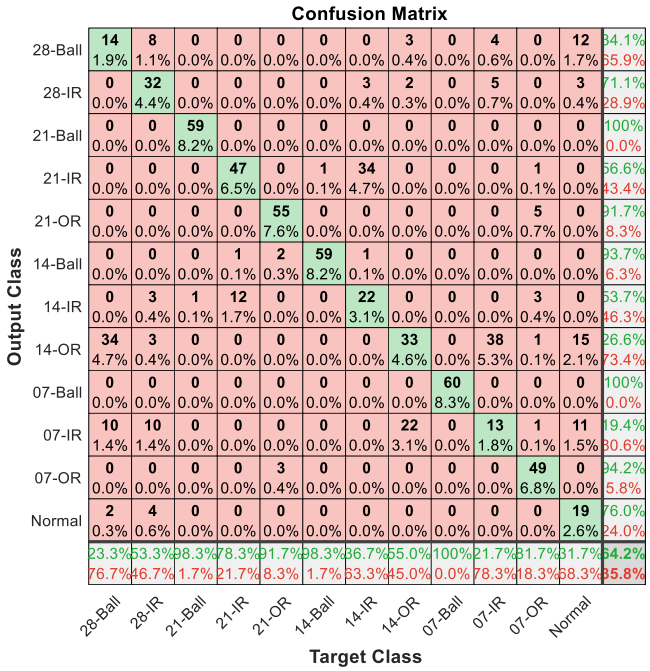

Figure 5-20 Confusion matrix for Raw-LSTM

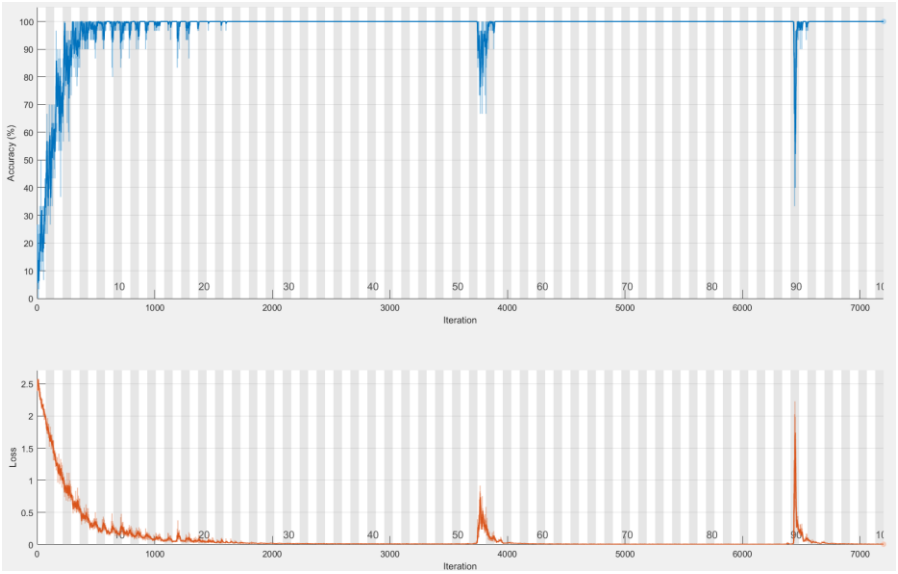

Figure 5-21 Training progress for Raw-LSTM

(5) LSTM result in comparison

Table 5-3 lists the LSTM comparison results. EMD-LSTM is not listed because the accuracy rate was much lower than expected. Compared with raw data results, STFT-LSTM, Cepstrum-LSTM, and WPT-LSTM achieve an accuracy average above 90%. The STFT-LSTM offers the best classification result.

Table 5-3 LSTM models Accuracy

Classifier Network	Acc-max(%)	Acc-min(%)	Acc-avg(%)	STD
Raw-LSTM	65.1	63.1	63.717	0.645
STFT-LSTM	100	99.986	99.999	0.042
Cepstrum-LSTM	91.53	90.3	90.946	0.464
WPT-LSTM	94.3	92.1	93.48	0611

5.3.3 Comparison and Evaluation

Figure 5-22 provides a general performance comparison of the CNN and LSTM networks. The CNN performs better than the LSTM. CNN has strong feature extraction ability, which enables raw signals to be accurately classified. Both STFT-CNN and STFT-LSTM achieve the same performance. STFT-CNN offers the highest accuracy in the CNN-based network, while the STFT-LSTM network obtains the highest accuracy in the LSTM based network. The EMD-LSTM network does not perform well.

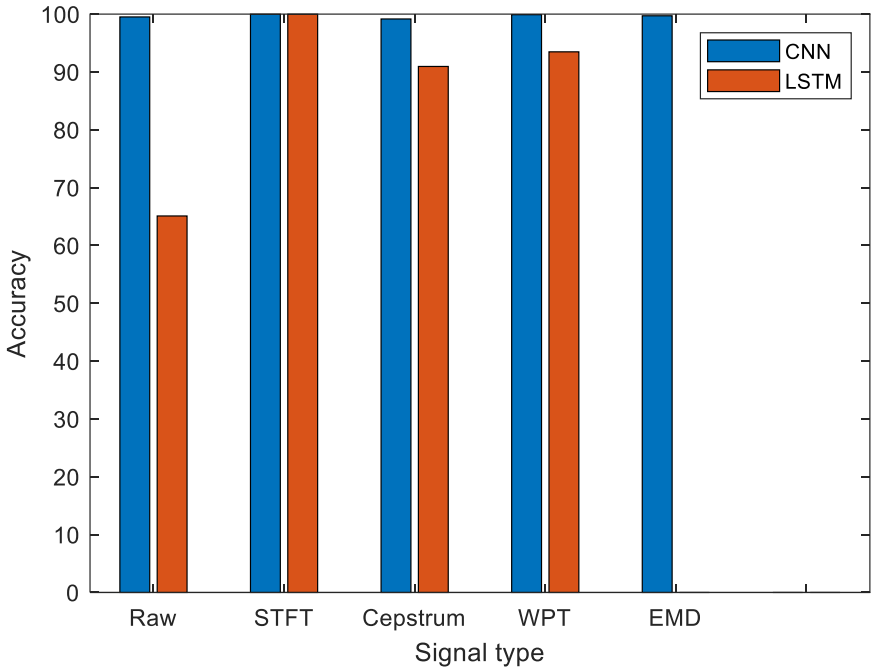

Figure 5-22 Accuracy versus signal processing methods

6. DESIGN OF 1D CONVOLUTIONAL NEURAL NETWORK

The experimental results in Chapter 4 have demonstrated that the CNN network structure has more advantages in machine fault classification. In this chapter, we attempt to develop a one-dimensional (1D) CNN network structure by modifying the existing 2D CNN to reduce the computational complexity of the original network. Meanwhile, we will continue to use the one-dimensional discrete-time Fourier transform of the fault signal as the input for testing.

The conventional CNN presented in the previous chapter is designed to operate on 2D data. Therefore, it is often defined as "2D CNN". The application of 1D CNN means that we can input one-dimensional vibration signals directly acquired by sensors into the network for operation.

The recently emerged 1D CNN immediately achieved impressive performance levels in several applications such as structural health monitoring, personalized biomedical data classification and early, anomaly detection and identification in power electronics, and motor-fault diagnosis. A significant advantage is that the compact and straightforward configuration of 1D CNN makes a real-time and low-cost hardware implementation feasible [40]-[41].

6.1 1D-CNN structure

This section serves as the development of one-dimensional CNN similar to [42], including the structure, forward propagation, and backward propagation algorithms. Figure 6-1 depicts the overall structure of the proposed 1D-CNN.

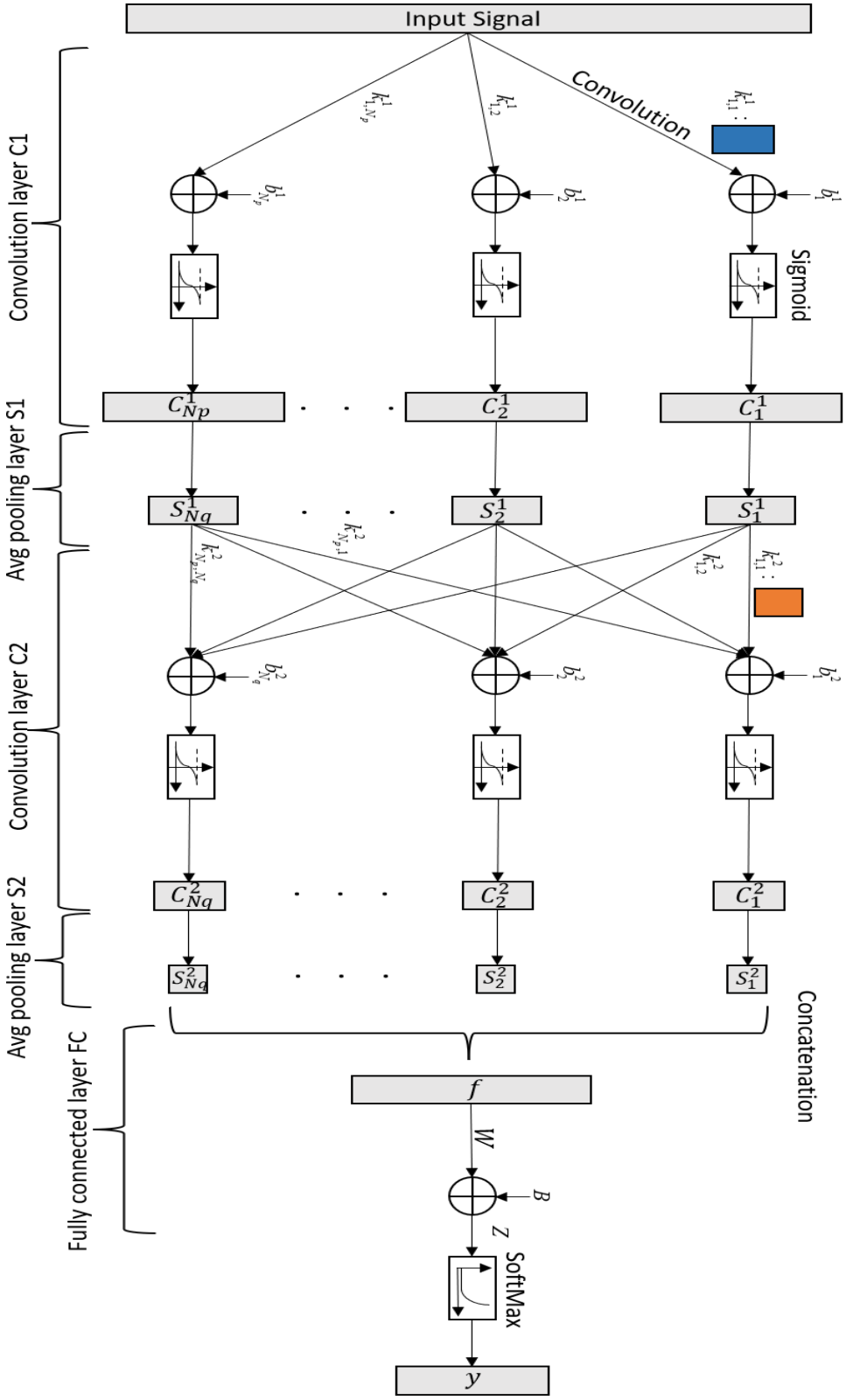

Figure 6-1 Structure of 1D-CNN

6.1.1 Forward Propagation

We define N_p as the number of the C1 feature map (p=1, 2, …, N_p). k^1 is the kernel of the C1 layer, b^1 is the basis, and I is the input signal. The convolution layer C1 can be expressed as,

$$C_p^1 = \sigma(I * k_{1,p}^1 + b_p^1) \ , \tag{6.1}$$

where σ is the sigmoid function

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{6.2}$$

The average pooling layer S1 can be obtained as

$$S_p^1(j) = \frac{1}{2}(C_p^1(2j-1) + C_p^1(2j)) \quad j = 1, 2, \cdots, N_j \, . \tag{6.3}$$

where N_j is the total number of columns of the feature map.

Then, we define N_q as the number of the C2 feature map, and q=1, 2, …, N_q. The convolution layer C2 is

$$C_p^2 = \sigma(\sum_{q=1}^{N_q} S_q^1 * k_{p,q}^2 + b_p^2) \tag{6.4}$$

The average pooling layer S2 is

$$S_q^2(j) = \frac{1}{2}(C_q^2(2j-1) + C_q^2(2j)) \tag{6.5}$$

Now, we let f denotes the concatenation of S_q^2

$$f \ = \left[S_1^2; \ S_2^2; \ ...; \ S_{N_q}^2 \right] \tag{6.6}$$

The size of the concatenation result of the network output can be obtained by

$$S = \frac{\frac{1}{2}(l - n_{k_1} + 1) - n_{k2}}{2} \cdot N_q, \tag{6.7}$$

where l is the length of the input signal, n_{k_1} and n_{k_2} are kernel size of the C1 and C2 layer, respectively.

For the FC layer, weight W is a (R x S) matrix, and bias B is a (R x1) matrix. R is the number of classes.

The output of the fully connected (FC) layer should be

$$Z = W^T f + B \tag{6.8}$$

The output of the SoftMax layer in a vector form is

$$y = \sigma_1(Z), \tag{6.9}$$

σ_1 is the SoftMax function, which is defined as

$$y_i = \sigma_1(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}, \tag{6.10}$$

where *m* is the dimension of the array, which is the number of fault types.

The loss function we used is the cross-entropy, which is given by

$$L = -\sum_{i}^{m} d_i \ln y_i, \tag{6.11}$$

where d_i is the training label and y_i the predicted probability for class *i*. For class *i*, *d* has the following vector form: $d = \begin{bmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}^T$.

6.1.2 Backpropagation

In the backpropagation, we update the parameters from the back to start, namely *W* and *B*, $k_{p,q}^2$ and b_q^2, $k_{1,p}^1$ and b_p^1.

Note that the optimization goal of the multi-classification problem is to minimize *L*. Therefore, we can define the gradient of *L* as

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \tag{6.12}$$

Taking partial derivative of *L* with respect to each component of *y* leads to

$$\frac{\partial L}{\partial y_i} = \begin{cases} -\frac{1}{y_i}, & i = o \\ 0, & i \neq o \end{cases} \tag{6.13}$$

where *o* is the index corresponding to the target category. The first term on the right-hand side of Equation (6.12) can be rewritten as

$$\frac{\partial L}{\partial y}=\begin{bmatrix}0\\0\\0\\-\frac{1}{y_o}\\0\\.\\0\end{bmatrix}, \tag{6.14}$$

the second term is a Jacob matrix

$$\frac{\partial y}{\partial z}=\begin{bmatrix}\frac{\partial y_1}{\partial z_1} & \frac{\partial y_1}{\partial z_2} & \frac{\partial y_1}{\partial z_3} & \cdots & \frac{\partial y_1}{\partial z_m}\\ \frac{\partial y_2}{\partial z_1} & \frac{\partial y_2}{\partial z_2} & \frac{\partial y_2}{\partial z_3} & \cdots & \frac{\partial y_2}{\partial z_m}\\ . & . & . & & .\\ \frac{\partial y_m}{\partial z_1} & \frac{\partial y_m}{\partial z_2} & \frac{\partial y_m}{\partial z_3} & \cdots & \frac{\partial y_m}{\partial z_m}\end{bmatrix} \tag{6.15}$$

When y_i takes the partial with respect to z_i

$$\frac{\partial y_i}{\partial z_i}=\begin{cases}y_i(1-y_i), & i=o\\ -y_i y_o, & i\neq o\end{cases}, \tag{6.16}$$

Finally, (6.15) can be expressed in matrix form

$$\frac{\partial y}{\partial z}=\begin{bmatrix}y_1(1-y_1) & -y_2y_1 & -y_3y_1 & \cdots & -y_my_1\\ -y_1y_2 & y_2(1-y_2) & -y_3y_2 & \cdots & -y_my_2\\ -y_1y_3 & -y_2y_1 & y_3(1-y_3) & \cdots & -y_my_3\\ -y_1y_m & -y_2y_m & -y_3y_m & \cdots & y_m(1-y_m)\end{bmatrix} \tag{6.17}$$

Then, we can combine (6.13) and (6.15), Equation (6.12) becomes

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z}$$

$$= \begin{bmatrix} y_1(1-y_1) & -y_2y_1 & -y_3y_1 & \dots & -y_my_1 \\ -y_1y_2 & y_2(1-y_2) & -y_3y_2 & \dots & -y_my_2 \\ -y_1y_3 & -y_2y_1 & y_3(1-y_3) & \dots & -y_my_3 \\ -y_1y_m & -y_2y_m & -y_3y_m & \dots & y_m(1-y_m) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{1}{y_o} \\ 0 \\ . \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ y_o - 1 \\ y_{o+1} \\ . \\ y_m \end{bmatrix} \begin{bmatrix} y_1 - 0 \\ y_2 - 0 \\ y_3 - 0 \\ . \\ y_o - 1 \\ y_{o+1} - 0 \\ . \\ y_m - 0 \end{bmatrix}$$

$$\therefore \frac{\partial L}{\partial z} = y - d \tag{6.18}$$

According to the chain derivation rule,

$$\Delta W = \frac{\partial L}{\partial W} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial W}$$

$$\therefore \Delta W = \frac{\partial L}{\partial z} \times f^T \tag{6.19}$$

$$\Delta B = \frac{\partial L}{\partial B} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial B}$$

$$\Delta B = \frac{\partial L}{\partial z} \tag{6.20}$$

$$\Delta f = \frac{\partial L}{\partial f} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial f}$$

$$\Delta f = \frac{\partial L}{\partial z} W^T \tag{6.21}$$

Before calculating $\Delta k_{p,q}^2$, we need to know the C2 layer's error ΔC_q^2. From Equation (6.6), we reshape the error array Δf into

$$\left\{\Delta S_q^2\right\}_{q=1,2,\ldots,N_q} = F^{-1}(\Delta f) \tag{6.22}$$

which is also the error of S2 layer. Then we get C2 layer's error by up sampling

$$\Delta C_q^2(j) = \frac{1}{2}\Delta S_q^2([j/2]) \tag{6.23}$$

Now $\Delta k_{p,q}^2$ can be represented by

$$\begin{aligned}\Delta k_{p,q}^2 &= \frac{\partial L}{\partial k_{p,q}^2} \\ &= \frac{\partial L}{\partial C_q^2(j)} \cdot \frac{\partial C_q^2(j)}{\partial k_{p,q}^2} \\ &= \Delta C_q^2(j) \cdot C_q^2(j)(1-C_q^2(j)) \cdot S_p^1(N_q+1-j)\end{aligned} \tag{6.24}$$

Then, we define

$$\Delta C_{q,\sigma}^2(j) = \Delta C_q^2(j) \cdot C_q^2(j)(1-C_q^2(j)) \tag{6.25}$$

$$S_{p,rot180}^1(j) = S_p^1(N_q+1-j) \tag{6.26}$$

Therefore, Equation (6.24) will turn into

$$\Delta k_{p,q}^2 = S_{p,rot180}^1 * \Delta C_{q,\sigma}^2 \tag{6.27}$$

$$\Delta b_q^2 = \sum_{j}^{\frac{1}{2}(n-n_{k_1}+1)-n_{k2}+1} \Delta C_{q,\sigma}^2(j) \tag{6.28}$$

The process of derivation of $\Delta k_{1,p}^1$ is similar to $\Delta k_{p,q}^2$, we can first get the error of S1 layer ΔS_p^1, and then derive error of C1 layer ΔC_p^1.

$$\Delta S_p^1 = \sum_{q=1}^{N_q} \Delta C_{q,\sigma}^2 * k_{p,q,rot180}^2 \tag{6.29}$$

$$\Delta C_p^1(j) = \frac{1}{2}\Delta S_p^1([j/2]) \tag{6.30}$$

$$\Delta C_{p,\sigma}^1(j) = \Delta C_p^1(j) \cdot C_p^1(j)(1-C_p^1(j)) \tag{6.31}$$

$\Delta k_{1,p}^1$ can be calculated as

$$\Delta k_{1,p}^{1} = I_{rot180} * \Delta C_{p,\sigma}^{1} \tag{6.32}$$

$$\Delta b_{p}^{1} = \sum_{j}^{n-n_{k_1}+1} \Delta C_{p,\sigma}^{1}(j) \tag{6.33}$$

6.1.3 Parameter Update

Finally, the updated parameters can be obtained by,

$$k_{1,p}^{1} = k_{1,p}^{1} - \alpha \cdot \Delta k_{1,p}^{1} \tag{6.34}$$

$$b_{p}^{1} = b_{p}^{1} - \alpha \cdot \Delta b_{p}^{1} \tag{6.35}$$

$$k_{p,q}^{2} = k_{p,q}^{2} - \alpha \cdot \Delta k_{p,q}^{2} \tag{6.36}$$

$$b_{q}^{2} = b_{q}^{2} - \alpha \cdot \Delta b_{q}^{2} \tag{6.37}$$

where α is the learning rate.

6.1.4 Initialization of Parameters

The input signal to the 1D CNN is a short-term Fourier transform (STFT) signal with a size of (1x1024). For the C1 layer, the kernel k^1 has the size of (1 x 5), and the bias b^1 has the size of (1 x 1). For the C2 layer, the kernel k^2 has the size of (1 x 7), and the bias b^2 has the size of (1 x 1). For the FC layer, weight *W* has the size of (12 x 3024), and bias *B* has the size of (12 x 1).

All bias, b_p^1 , b_q^2 , and *B*, are initialized to zero. The others are drawn randomly from a uniform distribution defined based on the kernel size and number of input and output maps on corresponding layers. $3.98*10^{-6}$

6.2 Experimental Results

In this experiment, the maximum number of epochs is set to 500, and the learning rate is selected as 0.01. The input signal to the 1D CNN is a one-dimensional signal with a length of 1024, which is obtained from a short-time Fourier transform (STFT). We achieved the average accuracy

of prediction as 94.444% through 10 independent runs. The training progress is shown in Figure 6-2.

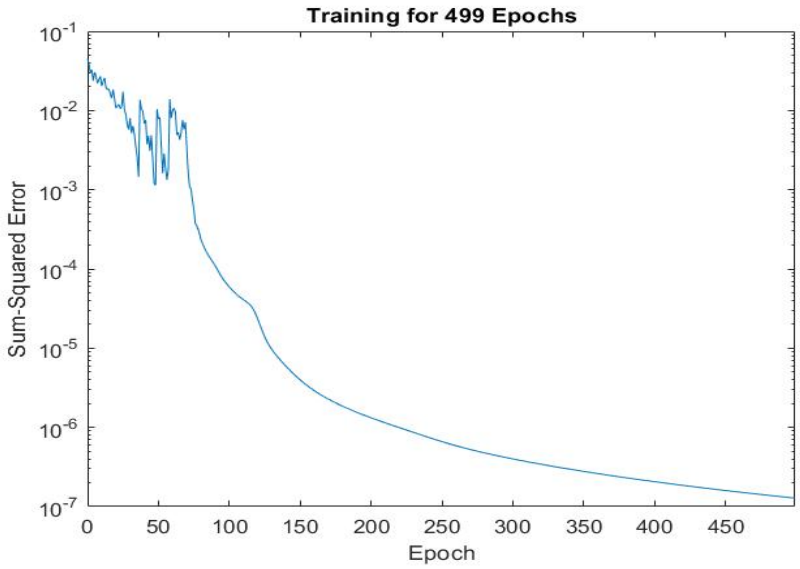

Figure 6-2 Training error of proposed 1D CNN

Table 6-1 Accuracy and MSE with different input data size

Input data size	Accuracy (%)	MSE
128	88.890	$3.02*10^{-4}$
256	93.333	$3.98*10^{-6}$
512	94.444	$4.78*10^{-7}$
1024	94.444	$3.56*10^{-8}$

Table 6-1 lists the performance of classification accuracy in terms of the input data length. As indicated in Table6-1, the accuracy rate does not increase further after the data length exceeds 512, although the mean square error is further decreased after training. The classification accuracy of the network can reach more than 90 percent. However, this 1D CNN network has performance degradation and cannot surpass the STFT-CNN model presented in Chapter 5 in terms of accuracy. Although the curve in Figure 6-2 does not converge ultimately after 500

training epochs, adding more training epochs and longer signal length could not improve the accuracy in our further experiments.

7. CONCLUSION AND FUTURE WORK

The bearing fault diagnosis methods based on traditional machine learning and deep learning are reviewed and summarized in this thesis work.

Three machine learning algorithms, K-nearest-neighbor (KNN), multi-class support vector machine (multi-SVM), and random forest (RF), are applied to bearing fault detection.

The performance of popular deep learning networks of convolutional neural networks (CNN) and long-short term memory (LSTM) networks for bearing fault diagnosis based on input signal processing have been validated. The signal processing methods include the raw signal (no processing), short-time Fourier transform (STFT), Cepstrum, wavelet packet transform (WPT), and empirical mode decomposition (EMD). The Case Western Reserve University's (CWRU) bearing dataset is adopted for our simulations. The STFT-CNN achieves the highest accuracy in the CNN-based networks, and the STFT-LSTM networks have the highest accuracy in the LSTM-based networks. The CNN-based networks perform better than the LSTM networks in general. The network input formulation using STFT offers the best performance. Our simulations validate the effectiveness of each network input formulation.

Moreover, the one-dimensional convolutional neural network is studied and constructed.

In the future, we will continue to explore the improved methods of one-dimensional CNN. Increasing the number of inputs channels using the signal decomposition and applying batch normalization may have potential. [43]-[45].

REFERENCES

[1] Y. Lei, Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery. Butterworth-Heinemann, pp. 67-174, 2017.

[2] R. B. Randall, “A history of Cepstrum analysis and its application to mechanical problems, Mechanical Systems and Signal Processing,” vol. 97, pp. 3-19, 2017.

[3] A. Rai, S. H. Upadhyay. "A review on signal processing techniques utilized in the fault diagnosis of rolling element bearings," Tribology International, vol 96, pp. 289-306. 2016.

[4] X. Zhang, Y. Liang, et al., “A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM,” Measurement, vol. 69, pp. 164-179, 2015.

[5] A. Soualhi, K. Medjaher, N. Zerhouni, “Bearing health monitoring based on Hilbert--Huang transform, support vector machine, and regression,” IEEE Transactions on Instrumentation and Measurement, vol. 64, no.1, pp. 52-62, 2015.

[6] D. Wang, “K-nearest neighbors-based methods for identification of different gear crack levels under different motor speeds and loads: Revisited,” Mechanical Systems and Signal Processing, vol. 70, pp. 201-208, 2016.

[7] R. Zhao, R. Yan, et al., “Deep learning and its applications to machine health monitoring,” Mechanical Systems and Signal Processing, vol. 115, pp. 213-237, 2019.

[8] L. Wen, X. Li, L. Gao, et al. “A new convolutional neural network-based data-driven fault diagnosis method,” IEEE Transactions on Industrial Electronics, vol 65, no 7, pp. 5990-5998, 2018.

[9] Y. Cai, L. Tan and J. Chen, “Evaluation of Deep Learning Neural Networks with Input Processing for Bearing Fault Diagnosis,” 2021 IEEE International Conference on Electro Information Technology (EIT), pp. 140-145, 2021.

[10] Y. Song, Y. Cai and L. Tan, “Video-Audio Emotion Recognition Based on Feature Fusion Deep Learning Method,” 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 611-616, 2021.

[11] D. Gray, Z. Zhang, C. Apostoaia and C. Xu, “A neural network based approach for the detection of faults in the brushless excitation of a synchronous motor,” 2009 IEEE International Conference on Electro/Information Technology, pp. 423-428, 2009.

[12] C. M. Apostoaia and M. Cernat, “Fault detection in synchronous motor drives, a co-simulation approach,” 2015 Intl Aegean Conference on Electrical Machines & Power Electronics (ACEMP), 2015 Intl Conference on Optimization of Electrical & Electronic Equipment (OPTIM) & 2015 Intl Symposium on Advanced Electromechanical Motion Systems (ELECTROMOTION), pp. 617-622, 2015.

[13] F. Jia, Y. Lei, J. Lin, et al. “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data,” Mechanical Systems and Signal Processing, vol. 72, pp. 303-315, 2016.

[14] M. Gan and C. Wang, “Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings,” Mech. Syst. Signal Process, vol. 72, pp. 92-104, May 2016.

[15] J. Chen, J, J. Jiang, X. Guo, L. Tan, “An efficient CNN with tunable input-size for bearing fault diagnosis,” International Journal of Computational Intelligence Systems, vol. 14, no. 1, pp. 625–634, 2021.

[16] J. Chen, J. Jiang, X. Guo, L. Tan, “A self-Adaptive CNN with PSO for Bearing Fault Diagnosis,” Systems Science & Control Engineering, vol. 8, no. 1, pp. 11-22, 2021.

[17] Case Western Reserve University (CWRU) Bearing Data Center. Accessed: Dec. 2018. [Online]. Available: https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-universitybearing-data-center-website

[18] W. A. Smith, and R. B. Randall, “Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study,” Mechanical Systems and Signal Processing, vol. 64, pp. 100-131, 2015.

[19] R. Randall, Vibration-Based Condition Monitoring: Industrial Aerospace and Automotive Applications. Wiley, 2011.

[20] Advantages of cepstrum analysis in gear fault diagnosis: https://www.zhygear.com/advantages-of-cepstrum-analysis-in-gear-fault-diagnosis/

[21] Wavelet Packets Introduction. Available: https://www. mathworks.com/help/wavelet

[22] N. Huang, Z., Shen, S. Long, et al, “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis,” Proceedings Mathematical Physical and Engineering Sciences, 454(1971), pp. 903-995, 1998.

[23] P. Zou, et al, “Bearing fault diagnosis method based on EEMD and LSTM,” International Journal of Computers Communications & Control, February 2020.

[24] L. Tan, J. Jiang, Digital Signal Processing: Fundamentals and Applications. Third Edition, Elsevier/Academic Press, 2018.

[25] Altman, N. S., “An introduction to kernel and nearest-neighbor nonparametric regression”. The American Statistician, vol.46, no. 3, pp. 175-185,1992.

[26] S. ZHANG, B. Wang, and T. HABETLER, "Deep Learning Algorithms for Bearing Fault Diagnostics—A Comprehensive Review," February 2020.

[27] D. He, R. Li, J. Zhu, and M. Zade, “Data mining based full ceramic bearing fault diagnostic system using AE sensors,” IEEE Trans. Neural Network, vol. 22, no. 12, pp. 2022–2031, Dec. 2011.

[28] M. M. Ettefagh, M. Ghaemi, and M. Y. Asr, “Bearing fault diagnosis using hybrid genetic algorithm K-means clustering,” in Proc. Int. Symp. Innov. Intell. Syst. Appli. (INISTA), Alberobello, Italy, pp. 84–89, Jun. 2014.

[29] 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 Trans. Ind. Electron., vol. 63, no. 3, pp. 1793–1803, Mar. 2016.

[30] T. W. Rauber, F. De Assis Boldt, and F. M. Varejao, “Heterogeneous feature models and feature selection applied to bearing fault diagnosis,” IEEE Trans. Ind. Electron., vol. 62, no. 1, pp. 637–646, Jan. 2015.

[31] K Nearest Neighbors Tutorial: https://people.revoledu.com/kardi/tutorial/KNN/

[32] Euclidean distance: https://en.wikipedia.org/wiki/Euclidean_distance

[33] Jingzhao Dai, "Sparse Discrete Wavelet Decomposition and Filter Bank Techniques for Speech Recognition," May 2019.

[34] Vikramaditya Jakkula, “Tutorial on Support Vector Machine (SVM),” School of EECS, Washington State University, Pullman 99164.

[35] A Complete Guide to the Random Forest Algorithm. Available: https://builtin.com/data-science/random-forest-algorithm

[36] Understanding Random Forest: How the Algorithm Works and Why it Is So Effective. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[37] Random Forests, Leo Breiman and Adele Cutler. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings

[38] S. Hochreiter, J. Chmidhuber, "Long short-term memory," Neural Computation, vol. 9 no. 8, pp. 1735-1780, November 1997.

[39] Zhang, Ran & Tao, Hongyang & wu, Lifeng & Guan, Yong. "Transfer Learning With Neural Networks for Bearing Fault Diagnosis in Changing Working Conditions." IEEE Access. pp. 1-1, 2017.

[40] Jin Woo Oh, Jongpil Jeong, "Data augmentation for bearing fault detection with a light weight CNN," vol. 175, pp. 72-79, 2020.

[41] Zhang, Zhifei. "Derivation of Backpropagation in Convolutional Neural Network (CNN)." July 2016.

[42] Sergey Ioffe and Christian Szegedy. "Batch normalization: accelerating deep network training by reducing internal covariate shift." In Proceedings of the 32nd International Conference on International Conference on Machine Learning, vol. 37, pp.448–456, 2015.

[43] T. Ince, S. Kiranyaz, L. Eren, M. Askar and M. Gabbouj, "Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks," in IEEE Transactions on Industrial Electronics, vol. 63, no. 11, pp. 7067-7075, Nov. 2016.

[44] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D. J. Inman, "1D convolutional neural networks and applications: A survey," Mechanical Systems and Signal Processing, vol.151, 2021.

[40] Xiaochen Zhang, Yiwen Cong, Zhe Yuan, Tian Zhang, Xiaotian Bai, "Early Fault Detection Method of Rolling Bearing Based on MCNN and GRU Network with an Attention Mechanism," Shock and Vibration, vol. 2021, pp.13, 2021.

[45] G. Jiang, H. He, J. Yan and P. Xie, "Multiscale Convolutional Neural Networks for Fault Diagnosis of Wind Turbine Gearbox," in IEEE Transactions on Industrial Electronics, vol. 66, no. 4, pp. 3196-3207, April 2019.

PUBLICATIONS

Conference Publication

Y. Cai, L. Tan and J. Chen, "Evaluation of Deep Learning Neural Networks with Input Processing for Bearing Fault Diagnosis," 2021 IEEE International Conference on Electro Information Technology (EIT), 2021, pp. 140-145.

Y. Song, Y. Cai and L. Tan, "Video-Audio Emotion Recognition Based on Feature Fusion Deep Learning Method," 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2021, pp. 611-616.