GRAPH NEURAL NETWORKS BASED ON MULTI-RATE SIGNAL DECOMPOSITION FOR BEARING FAULT DIAGNOSIS

by

Guanhua Zhu

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

May 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Lizhe Tan, Chair
Department of Electrical and Computer Engineering

Dr. Quamar Niyaz
Department of Electrical and Computer Engineering

Dr. Khair AI Shamaileh
Department of Electrical and Computer Engineering

Approved by:
Dr. Lizhe Tan

ACKNOWLEDGMENTS

I am grateful to Professor Lizhe Tan, the chair of my committee, for his theoretical and practical guidance throughout my year-and-a-half-long research journey. Thanks to his mentorship, I have been able to enhance both my research skills and theoretical foundation. I would also like to express my appreciation to Professor Quamar Niyaz and Professor Khair Al Shamaileh, who have provided me with their patient guidance and assistance in recent years.

A part of this thesis has been published in 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, February 2023.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Roller bearings are the common components used in the mechanical systems for mechanical processing and production. The running state of roller bearings often determines the machining accuracy and productivity on a manufacturing line. Roller bearing failure may lead to the shutdown of production lines, resulting in serious economic losses. Therefore, the research on roller bearing fault diagnosis has a great value. This thesis research first proposes a method of signal frequency spectral resampling to tackle the problem of bearing fault detection at different rotating speeds using a single speed dataset for training the network such as the one dimensional convolutional neural network (1D CNN). Second, this research work proposes a technique to connect the graph structures constructed from spectral components of the different bearing fault frequency bands into a sparse graph structure, so that the fault identification can be carried out effectively through a graph neural network in terms of the computation load and classification rate. Finally, the frequency spectral resampling method for feature extraction is validated using our self-collected datasets. The performance of the graph neural network with our proposed sparse graph structure is validated using the Case Western Reserve University (CWRU) dataset as well as our self-collected datasets. The results show that our proposed method achieves higher bearing fault classification accuracy than those recently proposed by other researchers using machine learning approaches and neural networks.

1. INTRODUCTION

1.1 Literature Review

As one of the most common industrial machinery components, the failure of rolling bearings will cause serious economic losses [1] [2]. Generally speaking, if the corresponding characteristic signals can be detected at the early stage of bearing failure by using related detection methods, the failure of rolling bearings can be avoided. Prevention can significantly improve the service life of machinery [3] [4] [5]. Techniques for bearing diagnosis can be roughly divided into methods based on signal processing, traditional machine learning methods, and deep learning methods. Among them, the process based on signal processing mainly includes, but not limited to, fast Fourier transform (FFT), wavelet transform (WT), empirical mode decomposition (EMD), and wavelet packet transform (WPT) on the vibration signal of the bearing. M. Cocconcelli et al. [6] used a short-time Fourier transform (STFT) to process the original vibration signal. They analyzed the results of the frequency domain transformation to detect bearing defects. Z. Peng et al. [7] used the WT to process the vibration signal of bearing to extract fault features. C. Castejón [8] further adopted the WPT to complete the extraction of fault features. Although the method based on signal processing can effectively extract fault features, since the judgment of features is still based on artificially formulated rules, when the fault features are not noticeable, this method will produce misjudgments. Aiming at the defects of signal processing methods, some research works to address these defects by introducing traditional machine learning algorithms. For example, A. Malhi and R. X. Gao [9] combined principal component analysis (PCA) with feed-forward neural networks (FFFNs) to extract and classify fault features. Y. S. Wang et al. [10] proposed an intelligent fault diagnosis method according to the support vector machine (SVM) algorithm. D. H. Pandya et al. [11] used the K-nearest neighbor algorithm (KNN) to determine the bearing defects. F. Shen et al. [12] used a transfer learning-based singular value decomposition (SVD) model to complete the diagnosis of bearing faults. Although traditional machine learning algorithms are sufficient to automatically learn and discover typical features of defect signals from sufficient data, the shallow model structure of conventional machine learning limits its ability to extract abstract features further. This feature limits the further improvement of the performance of such methods.

Because of the shortcomings of traditional machine learning, using a deeper model to extract and learn features has become a feasible solution. For example, M. Zhao et al. [13] proposed a method based on deep residual reduction networks, which significantly improves fault diagnosis accuracy. In our previous work [14] [15], we also proposed a diagnostic method combining an improved convolutional neural network (CNN) based on the particle swarm optimization (PSO) algorithm and an STFT with adjustable input size and CNN, respectively. W. Chen and K. Shi [16] used an end-to-end multi-scale convolutional neural network (MSCNN) to classify the types of bearing defects. Yao et al. [17] used a 1D convolutional layer to decompose the raw signal into multi-scale features, operating in an end-to-end fashion to improve practicality. In one of our studies [18]-[19], we have experimented with the combined effects of different signal processing methods and mainstream deep neural networks. Excellent fault classification results are obtained by introducing a multi-scale input model based on the best-performing combination. Again, based on our previous work in [19], we proposed a scheme which combines the sparse wavelet decomposition with a one-dimensional convolutional neural network [20]. Since the sparse wavelet decomposition can split the signal into multi-resolution features for different frequency bands, which are applied to multi-scale neural networks, the classification accuracy of the one-dimensional convolutional neural network has been further improved over the method proposed in [19]. Because the path of the sparse wavelet decomposition follows the Gray code, we propose to enhance the sparse WPT decomposition in our previous work [21]-[23] by finding an optimal sparse decomposition structure.

Although deep learning-based bearing diagnosis methods have achieved excellent performance, few studies on the relationship between different fault features are involved. Using graph data to concatenating various fault features to include their data relationships and learning them may be a feasible solution to improve performance of the model. In fact, after studying several cases of bearing diagnosis using graph neural networks [24] [25] [28], it is clear that the proposed methods were only processed based on the original signal, and no relationship is applied. Therefore, we propose a new graph data construction method to improve the classification performance for bearing fault diagnosis. In addition, it is noted that when the rotating speed of the bearing changes, its frequency spectrum will also deviate to a relative degree in the frequency axis correspondingly. This leads to that when using the trained neural network model for fault detection, it is necessary to adjust frequency spectrum using the bearing rotating speed so that the produced features are consistent with the one used in the training set. To solve this problem, we propose a frequency spectral resampling method to make bearing fault detection possible under the different rotating speed conditions.

1.2 Research and Motivation

By reviewing the methods proposed in the literature in recent years, it is known that by combing the signal processing and deep learning algorithms, many bearing fault detection methods have achieved good results. The excellent performance of these methods on multiple datasets demonstrates their effectiveness. However, all current fault identification methods are only aimed at bearing fault identification when the rotational speed is fixed. A change in the rotational speed of the bearing will result in a frequency shift of the characteristic signal in the frequency domain, thus causing the failure of the fault detection method for the bearing developed at the previous rotational speed. In this research work we propose a method based on bearing frequency spectral resampling, which can normalize the frequency spectrum of bearings at different speeds, thus solving the problem of bearing fault diagnosis of at the different rotating speeds. However, through observation, we found that with the increase of bearing speed, there will be changes in signal energy and unavoidable harmonics, and these problems still need further research and solution. In addition, our previous work mainly divides the bearing fault frequency range into three frequency bands in actual operation, namely the low frequency region, the natural bearing frequency region, and the high frequency region. We speculate that the impact of different bearing faults in these three areas may be related. If the deep learning algorithm can take this connection into account, it may improve the accuracy of fault diagnosis. Therefore, this research work proposes a connection method to construct a sparse graph structure for this problem. We first use the horizontal visibility graph (HVG) algorithm to connect the signals of each band into a graph structure, and then use our proposed sparse graph connection method to connect the graphs of the three bands as three subgraphs to form a sparse graph. Experimental results validate that our method has achieved the improved performance. For different data sets, our algorithm can increase the accuracy of fault classification by 5% to 10%.

1.3 Contribution of Thesis

The research work in this thesis has the following contributions:

(1) A frequency spectral resampling method for extracting features for bearing fault detection is proposed. This method solves the problem of fault identification and diagnosis for the same bearings with different rotating speeds using a single speed dataset for training the network such as the one-dimensional CNN.

(2) A sparse graph connection method for connecting the subgraphs constructed from different bands into a sparse graph structure is proposed. The sparse graph structure connected by this method with the graph deep learning algorithm can find the connection between the phenomena caused by the same fault in different frequency bands, thereby improving the accuracy of bearing fault identification.

1.4 Organization of Thesis

The organization of this thesis is as follows. In Chapter 2, the bearing fault frequency bands are introduced, along with the techniques of spectral resampling in frequency domain. Chapter 3 focuses on the diagnosis and classification of bearing faults using two of the most popular deep learning algorithms, that is, convolutional neural networks (CNN) and graph neural network (GNN). Chapter 4 mainly validates the practical results and the analysis. Finally, Chapter 5 summarizes the conclusions drawn from this study and outlines potential areas for future results.

2. BEARING FAULT DIAGNOSIS

2.1 Bearing Signal Fault Band

As shown in Figure 2.1, a bearing is composed of four parts: cage, ball, inner race and outer race.

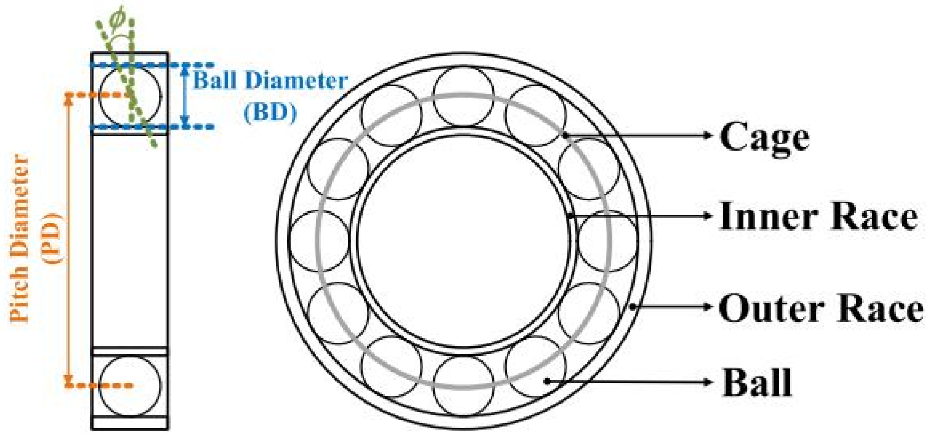


Figure 2.1. The structure of general bearing.

When the bearing rotates, the positions of these four parts in the full frequency band of the bearing signal can be calculated by formulas (1)-(4)[25]:

$$f_{OD} = \frac{n}{2} f_{rm}\left(1 - \frac{BD}{PD} cos\emptyset\right) \tag{1}$$

$$f_{ID} = \frac{n}{2} f_{rm}\left(1 + \frac{BD}{PD} cos\emptyset\right) \tag{2}$$

$$f_{BD} = \frac{PD}{2BD} f_{rm}\left(1 - \left(\frac{BD}{PD}\right)^2 cos^2\emptyset\right) \tag{3}$$

$$f_{Cage} = \frac{1}{2} f_{rm}\left(1 - \frac{BD}{PD} cos\emptyset\right) \tag{4}$$

where f_{OD}, f_{ID}, f_{BD} and f_{Cage} are the outer race, inner race, ball, and cage frequencies, respectively; n and f_{rm} are the number of balls and shaft speed, respectively; BD and PD are the ball diameter and rolling pitch diameter, respectively; $\emptyset$ is the contact angle between the ball and bearing as shown in Figure 2.1, which is approximately to be $0\sim10^{\circ}$.

As shown in Figure 2.2, the fault frequencies of bearings can be divided into four zones: low frequency, bearing defect frequency, bearing nature resonance frequency, and high frequency zones. In practice, we simplify these four regions into three frequency bands. As shown in Figure 2.2, band 1 mainly contains low-frequency information, ranging from $0.5f_{rm}$ to f_{rm}, which is considered to contain low-frequency characteristic information valuable for bearing fault diagnosis, including f_{Cage}. Band 2 is bounded by f_{BD} and f_{ID}, and contains information such as f_{BD}, f_{OD} and f_{ID}, which are mainly used for bearing fault identification. Band 3 takes f_{ID} as the lower bound and 60 times f_{rm} as the upper bound, including bearing nature frequency and high frequency zones. When the bearing fails, the abnormal fluctuation caused by the bearing at low frequency will also be reflected at high frequency, so the high frequency region is also considered valuable. It is worth noting that in order to ensure sufficient frequency margin to cover the desired frequency range, in practice we extend each band by $0.25f_{rm}$ from its edge.

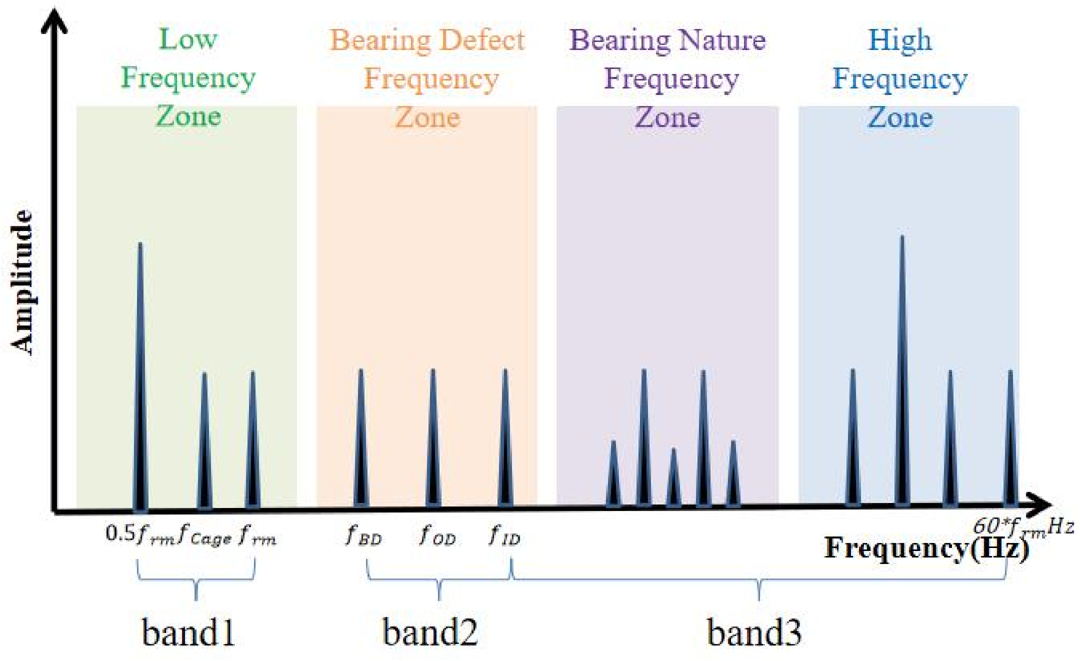


Figure 2.2. Four bearing characteristic frequency zones and three bands for bearing fault diagnosis.

2.2 Signal Processing Methods

According to Equations (1)-(4) in Section 2.1, it is not difficult to infer that the position of the fault frequency band of the bearing on the total frequency spectrum has a direct relationship with the bearing speed. As the rotational speed of the bearing increases, the frequency domain signal of the bearing will shift to high frequency as a whole, and the frequency space between the characteristic spectral components will also increase. This will lead to the fact that for the same type of bearing, the machine learning model trained with the inherent speed bearing signal data set cannot be directly applied to the same type of bearing with different speeds. Aiming at this problem, this thesis proposes a frequency domain resampling method, which can solve the problem of frequency shift when the bearing speed changes. This chapter will describe this method in detail.

2.2.1 Frequency Spectral Resampling Based on Linear Interpolation

In order to deal with fault diagnosis of bearings at different shaft speeds, it is first necessary to figure out the main ways in which the frequency characteristics of the same bearing shift at different speeds. According to the bearing characteristic frequency calculation in Equations (1)-(4) in Section 2.1, and the observation of the bearing frequency spectrum at different speeds, we found that the characteristic frequency of the bearing is in terms of the bearing speed. Therefore, we can resample the frequency spectrum of the bearing based on different fraction of the bearing's rotational speed frequency. Here we begin to introduce the most concise and fast resampling method, that is, the linear interpolation. As shown in Figure 2.3, it is assumed that the blue signal is the original frequency spectrum of the bearing, and the yellow signal is obtained by resampling with an accuracy of 0.5f_rm under the linear interpolation method.

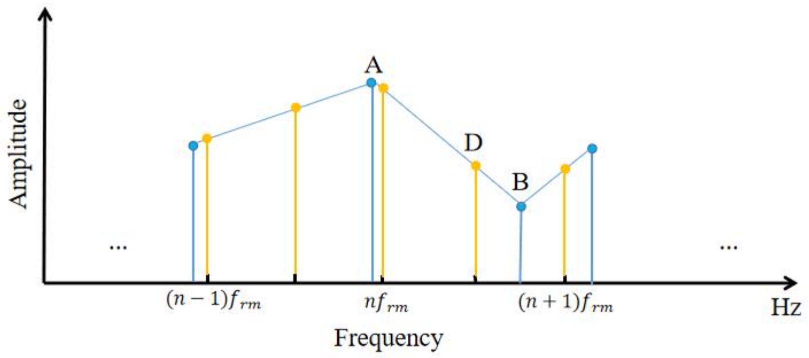


Figure 2.3. Frequency resampling based on linear interpolation.

Taking interpolation point D as an example, the interpolation frequency of point D is (n+0.5)f_{rm}. Notice that the closest original data points before and after the frequency of point D are A and B. Assuming that the frequency of original signal A is f_A, the amplitude is h_A, the frequency of original signal B is f_B, and the amplitude is h_B, the equation for determining the amplitude h_D at interpolation point D is given as follows:

$$h_D = \frac{h_B - h_A}{f_B - f_A} f_D + h_A - \frac{f_A(h_B - h_A)}{f_B - f_A} \tag{5}$$

For the given $f_D = (n + 0.5)f_{rm}$, Equation (5) becomes:

$$h_D = \frac{h_B - h_A}{f_B - f_A}(n + 0.5)f_{rm} + h_A - \frac{f_A(h_B - h_A)}{f_B - f_A}$$

The specific implementation method of the linear interpolation method is shown in the following pseudo code. Note that the reciprocal of the precision value is designated as the frequency resolution in terms of frm in the resampled spectrum, , frm is the bearing speed, F is the original frequency points, and f is the frequency points after the spectral resampling.

Algorithm 1 Linear Interpolation

```
1:  FUNCTION resampling(F, f, frm, precision=1)
2:    SET ff to 1/precision
3:    SET f to f/frm
4:    SET re_f to an empty array
5:    SET re_F to an empty array
6:    FOR i from 0 to length(F)-1 DO
7:        SET low_limit to f[i]
8:        SET high_limit to f[i+1]
9:        SET low_limit_value to F[i]
10:       SET high_limit_value to F[i+1]
11:       WHILE ff <= high_limit DO
12:           APPEND ff to re_f
13:           SET FF to ff×(high_limit_value-low_limit_value)/
                  (high_limit-low_limit) + (low_limit_value×high_limit-
                  high_limit_value×low_limit)/(high_limit-low_limit)
14:           APPEND FF to re_F
15:           SET ff to ff + 1/precision
16:       END WHILE
17:   END FOR
18:   RETURN re_f, re_F, f
19: END FUNCTION
```

2.2.2 Frequency Resampling Based on Centripetal Catmull-Rom Spline

Using the linear interpolation method described in the previous section to resample the spectrum is simple and fast, but the result of linear interpolation is often too rough to perfectly reflect the characteristics of the original signal. We need a spline function that is as smooth as possible as the basis for interpolation. Here we propose centripetal Catmull-Rom spline. Centripetal Catmull-Rom spline is an interpolating spline defined by four control points. As a variant of Catmull-Rom splines, centripetal Catmull-Rom splines have these three mathematical properties that are considered advantages over other types of Catmull-Rom methods. First, the centripetal Catmull-Rom spline will not form a loop or self-intersect in its line segment; secondly, the centripetal Catmull–Rom spline will never have a cusp; third, the centripetal Catmull-Rom spline will follow the control points more strictly [31].

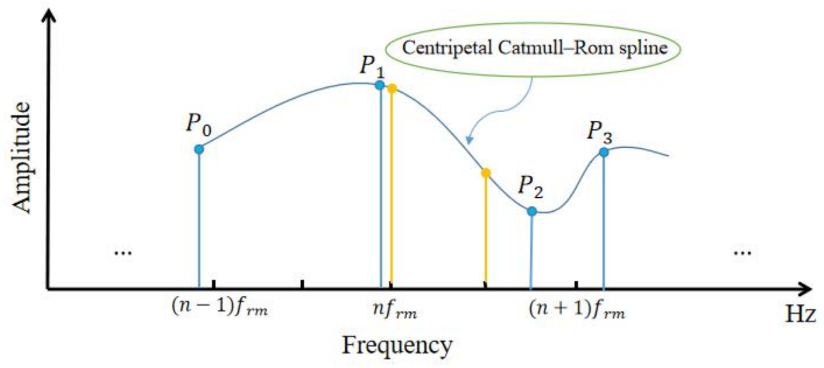


Figure 2.4. Frequency resampling based on centripetal Catmull-Rom spline.

As mentioned earlier, the centripetal Catmull-Rom spline is an interpolation method based on four-point control. As shown in Figure 2.4, to interpolate between P_1 and P_2 using the centripetal Catmull-Rom spline method, we first find the original data point P_0 before P_1 and the original data point of P_3 after P_2. As shown in Figure 2.5, the centripetal Catmull-Rom cpline method uses a pyramid structure to calculate the interpolation spline. The first step is to calculate four parameters t_0, t_1, t_2, t_3. t_0 defaults to 0, and $t_1 \sim t_3$ respectively represent the cumulative sum of the Euclidean distances from P_0 to P_1, as shown in Equation (12). Secondly, it is necessary to calculate three parameters of A_1, A_2 and A_3 at the first layer in the pyramid as shown in Figure 2.5. Equations (9)~(11) show the calculations of these three parameters. Afterwards, we use the calculated parameters A_1, A_2 and A_3 to calculate the parameters B_1 and

B_2 at the second layer of the pyramid. Equations (7)~(8) show the calculation of these two parameters. Finally, substituting parameters B_1 and B_2 into Equation (6), we obtain the height at interpolation point.

$$C = \frac{t_2 - t}{t_2 - t_1} B_1 + \frac{t - t_1}{t_2 - t_1} B_2 \tag{6}$$

$$B_1 = \frac{t_2 - t}{t_2 - t_0} A_1 + \frac{t - t_0}{t_2 - t_0} A_2 \tag{7}$$

$$B_2 = \frac{t_3 - t}{t_3 - t_1} A_2 + \frac{t - t_1}{t_3 - t_1} A_3 \tag{8}$$

$$A_1 = \frac{t_1 - t}{t_1 - t_0} P_0 + \frac{t - t_0}{t_1 - t_0} P_1 \tag{9}$$

$$A_2 = \frac{t_2 - t}{t_2 - t_1} P_1 + \frac{t - t_1}{t_2 - t_1} P_2 \tag{10}$$

$$A_3 = \frac{t_3 - t}{t_3 - t_2} P_2 + \frac{t - t_2}{t_3 - t_2} P_3 \tag{11}$$

$$t_{i+1} = \left[\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}\right]^{0.5} + t_i \tag{12}$$

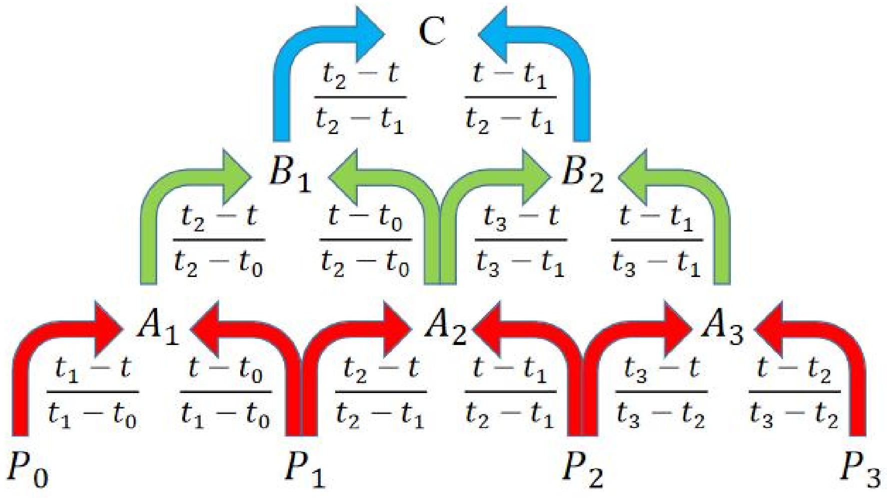

C
$\frac{t_2 - t}{t_2 - t_1}$
$\frac{t - t_1}{t_2 - t_1}$
B_1
B_2
$\frac{t_2 - t}{t_2 - t_0}$
$\frac{t - t_0}{t_2 - t_0}$
$\frac{t_3 - t}{t_3 - t_1}$
$\frac{t - t_1}{t_3 - t_1}$
A_1
A_2
A_3
$\frac{t_1 - t}{t_1 - t_0}$
$\frac{t - t_0}{t_1 - t_0}$
$\frac{t_2 - t}{t_2 - t_1}$
$\frac{t - t_1}{t_2 - t_1}$
$\frac{t_3 - t}{t_3 - t_2}$
$\frac{t - t_2}{t_3 - t_2}$
P_0
P_1
P_2
P_3

Figure 2.5. Calculation of centripetal Catmull-Rom spline.

Algorithm 2 depicts the specific implementation of centripetal Catmull-Rom spline, where x0~x3 are the frequencies corresponding to the abscissas of the four points on which the spline interpolation is based, and P0~P1 are the amplitudes corresponding to the ordinates of the four points. x is the frequency corresponding to the interpolation point, and the output C is the amplitude of the interpolation point.

Algorithm 2 Centripetal Catmull-Rom Spline

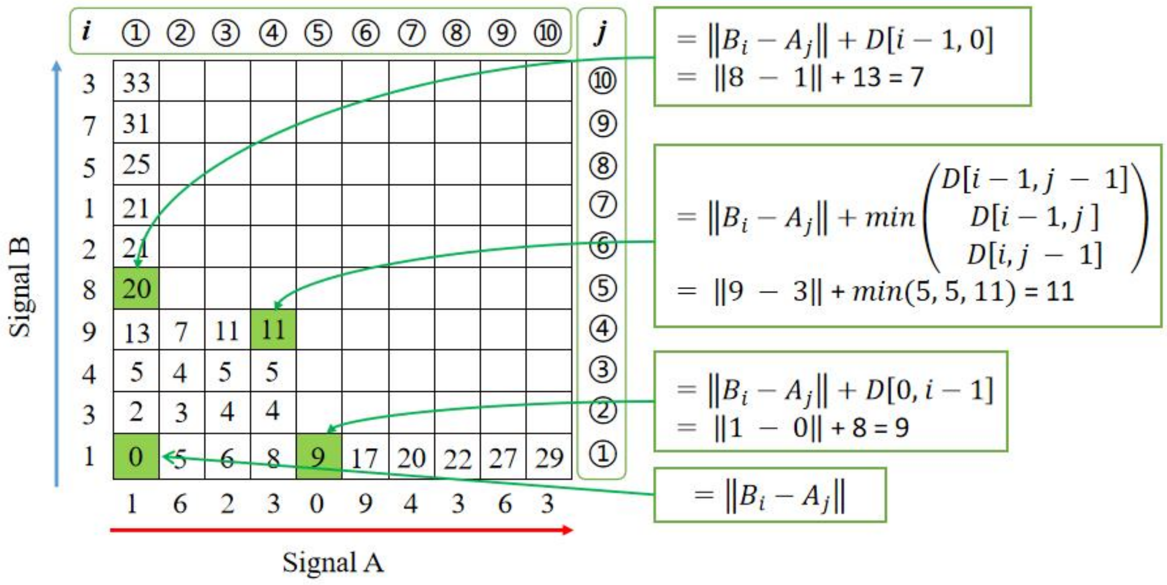
```
1:  FUNCTION catmullRom(x0, x1, x2, x3, P0, P1, P2, P3, x)
2:      SET t0 to 0
3:      SET t1 to square root of ((x1 - x0)^2 + (P1 - P0)^2)
4:      SET t2 to square root of ((x2 - x1)^2 + (P2 - P1)^2) + t1
5:      SET t3 to square root of ((x3 - x2)^2 + (P3 - P2)^2) + t2
6:      SET t to t1 + (x - x1) / (x2 - x1) × (t2 - t1)
7:      SET A1 to ((t1 - t) / (t1 - t0)) × P0 + ((t - t0) / (t1 - t0)) × P1
8:      SET A2 to ((t2 - t) / (t2 - t1)) × P1 + ((t - t1) / (t2 - t1)) × P2
9:      SET A3 to ((t3 - t) / (t3 - t2)) × P2 + ((t - t2) / (t3 - t2)) × P3
10:     SET B1 to ((t2 - t) / (t2 - t0)) × A1 + ((t - t0) / (t2 - t0)) × A2
11:     SET B2 to ((t3 - t) / (t3 - t1)) × A2 + ((t - t1) / (t3 - t1)) × A3
12:     SET C to ((t2 - t) / (t2 - t1)) × B1 + ((t - t1) / (t2 - t1)) × B2
13:     RETURN C
14: END FUNCTION
```

2.2.3 Dynamic Time Warping

When using the frequency spectral resampling method to generate a resampled frequency spectrum, the resampling accuracy must be considered first, that is, it is first necessary to decide a factor of f_{rm} to perform the interpolation. Let define an integer precision as the reciprocal of the factor. If the precision value is too small, it will cause the resampled spectrum to lose the characteristics that cannot reflect the original spectrum, resulting in signal distortion. Excessive precision will increase unnecessary calculations [31]. Thus, we use the dynamic time warping method to find the best accuracy for signal interpolation. Using the dynamic time warping method to calculate the matching degree between signals is divided into three steps.

First, the distance matrix needs to be calculated based on the signal strengths of the two signals to be matched. As shown in Figure 2.6, each element of the distance matrix can be calculated by Equation (13) [31].

$$D_{ij} = \left\| B_i - A_j \right\| + min(D[i-1, j-1], D[i-1, j], D[i, j-1]) \tag{13}$$

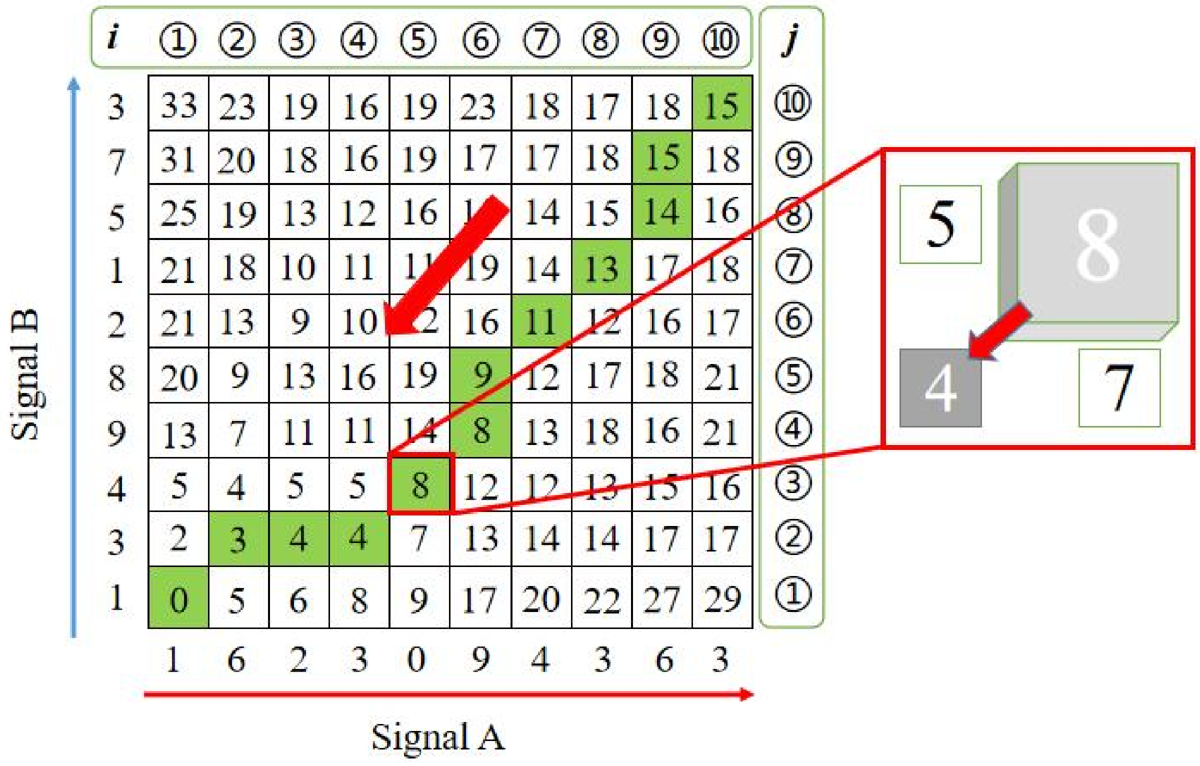


i
j
Signal B
Signal A
$= \|B_i - A_j\| + D[i-1, 0]$
$= \|8 - 1\| + 13 = 7$
$= \|B_i - A_j\| + min\begin{pmatrix} D[i-1, j-1] \\ D[i-1, j] \\ D[i, j-1] \end{pmatrix}$
$= \|9 - 3\| + min(5, 5, 11) = 11$
$= \|B_i - A_j\| + D[0, i-1]$
$= \|1 - 0\| + 8 = 9$
$= \|B_i - A_j\|$

Figure 2.6. Calculation of the distance matrix based on the input signal.

Since the distance matrix is calculated from the edge of the matrix, the adjacent elements will have vacancies when the element values near the edge are calculated. Taking the calculation of $D_{5,1}$ in Figure 2.6 as an example, when calculating the first column element of the distance matrix, it is only necessary to consider the absolute value of the signal amplitude difference at the corresponding position of the input signal plus the value of the element above it (row value table small direction). Correspondingly, take the calculation of $D_{1,5}$ in Figure 2.6 as an example. When calculating the first row of elements of the distance matrix, it is only necessary to consider the absolute value of the signal amplitude difference at the corresponding position of the input signal plus the value of the element on the left of it (column value table small direction). The intersection element $D_{1,1}$ of the first row and the first column of the distance matrix may be the absolute value of the difference of the input signals. It is worth noting that the length of the resampled signal is inconsistent with that of the original signal, and generally the length of the resampled signal is greater than that of the original signal. Therefore, in order to align the length of the input signal, we need to copy each value of the original signal by a certain multiple (the specific multiple is determined by the interpolation accuracy). See Algorithm 3 for details.

After the distance matrix is calculated, we need to find the shortest path in the distance matrix. It is necessary to find the shortest path from the last element of the distance matrix, and each time find the smallest element among the three elements above, obliquely above, and to the left of the current element as the shortest path element. Taking Figure 2.7 as an example, we start looking for the shortest path from $D_{10,10}$. When the current element is $D_{3,5}$, the values of the adjacent elements of $D_{3,5}$ in the upper, left, and upper oblique directions are 7, 5, and 4 respectively. Therefore, the next element to be added to the shortest path should be the value of the upper oblique element of $D_{3,5}$, that is, $D_{2,4}$.



3	33	23	19	16	19	23	18	17	18	15
7	31	20	18	16	19	17	17	18	15	18
5	25	19	13	12	16	1	14	15	14	16
1	21	18	10	11	1	19	14	13	17	18
2	21	13	9	10	2	16	11	12	16	17
8	20	9	13	16	19	9	12	17	18	21
9	13	7	11	11	14	8	13	18	16	21
4	5	4	5	5	8	12	12	13	15	16
3	2	3	4	4	7	13	14	14	17	17
1	0	5	6	8	9	17	20	22	27	29
	1	6	2	3	0	9	4	3	6	3

Figure 2.7. The shortest path in a distance matrix.

As shown in Figure 2.7, the green part is the shortest path included in the distance matrix calculated based on signal A and signal B. Whenever the signal is skewed in frequency, the shortest path is shifted horizontally or vertically in the distance matrix. As shown in Figure 2.8, all translation positions are marked in yellow. Let the number of shifts to occur on the shortest path is n, and the length of signal A and signal B is N, respectively, the warping degree of signal A and signal B can be expressed by Equation (14) below:

$$W_{A-B} = \frac{n}{N} \tag{14}$$

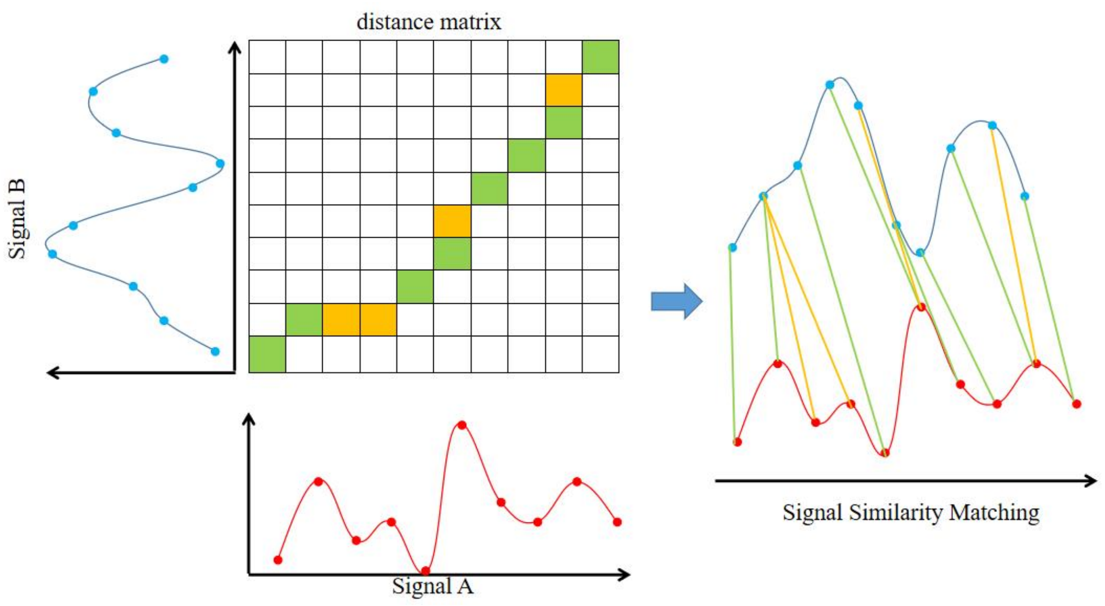


Figure 2.8. Utilization of dynamic time warping method to calculate the similarity between A and B signals.

Algorithm 3 Dynamic Time Warping

```
1:  FUNCTION DTW(F, re_F):
2:    // Align the original signal with the resampled signal
3:    SET time to integer division of len(re_F) by len(F)
4:    SET remain to len(re_F) minus len(F) times time
5:    SET FF to numpy array of zeros with length len(re_F)
6:    SET i to 0
7:    SET Add to False
8:    SET FF[0:time:time×len(F)] to F
9:    SET FF[0:time:time×len(F)+1] to F
10:   SET FF[len(re_F) - remain:end] to F[end]
11:
12:   // create the distance matrix
13:   SET D to numpy array of zeros with shape (len(re_F)+1, len(re_F)+1)
14:   SET all values in D's first row and column to infinity
15:   FOR each i in range from 1 to len(re_F)+1:
16:       FOR each j in range from 1 to len(re_F)+1:
17:          IF i and j = 0
18:             SET D[i, j] to absolute difference between re_F[i-1] and FF[j-1]
19:          ELSE:
20:             SET D[i, j] to absolute difference between re_F[i-1] and FF[j-1]
21:                 plus minimum of [D[i-1,j-1], D[i-1, j], D[i, j-1]]
```

```
            END IF
        END FOR
    END FOR
    // Calculate the matching degree of the original signal and the resampled signal
    SET dev to 0
    SET i to len(re_F)
    SET j to len(re_F)
    WHILE i is not 0 and j is not 0:
        SET submatrix to 2x2 submatrix of D with top-left corner at (i-1, j-1)
        SET submatrix's bottom-right element to infinity
        SET min_index to index of minimum element in submatrix
        SET i to min_index[0] plus i minus 1
        SET j to min_index[1] plus j minus 1
        IF min_index[0] is not 0 or min_index[1] is not 0:
            SET dev to dev plus 1
        END IF
    END WHILE
    SET Match to 1 minus dev divided by len(re_F)
    RETURN Match
END FUNCTION
```

2.2.4 Sparse Wavelet Decomposition

To extract the corresponding information of the characteristic and relevant frequency bands from the full-band information of the original signal, we also use a sparse wavelet decomposition method to extract the signals contained in band 1, band 2 and band 3.

Figure 2.9 describes a complete decomposition structure of the signal in the frequency band 0~200 Hz into three levels. "0" in the figure represents low-frequency decomposition, that is, the input signal from the upper layer is divided into two segments, and the low-frequency band is retained, which is marked as "L". "1" stands for high-frequency decomposition, which also divides the input signal from the previous layer into two segments, and retains the high-frequency band, marked as "H". Arrows indicate the direction of frequency distribution at the current decomposition level, with arrows pointing from low frequencies to high frequencies. Green arrows indicate no change in the direction of the frequency distribution, while red arrows indicate a change in the direction of the frequency distribution [27].

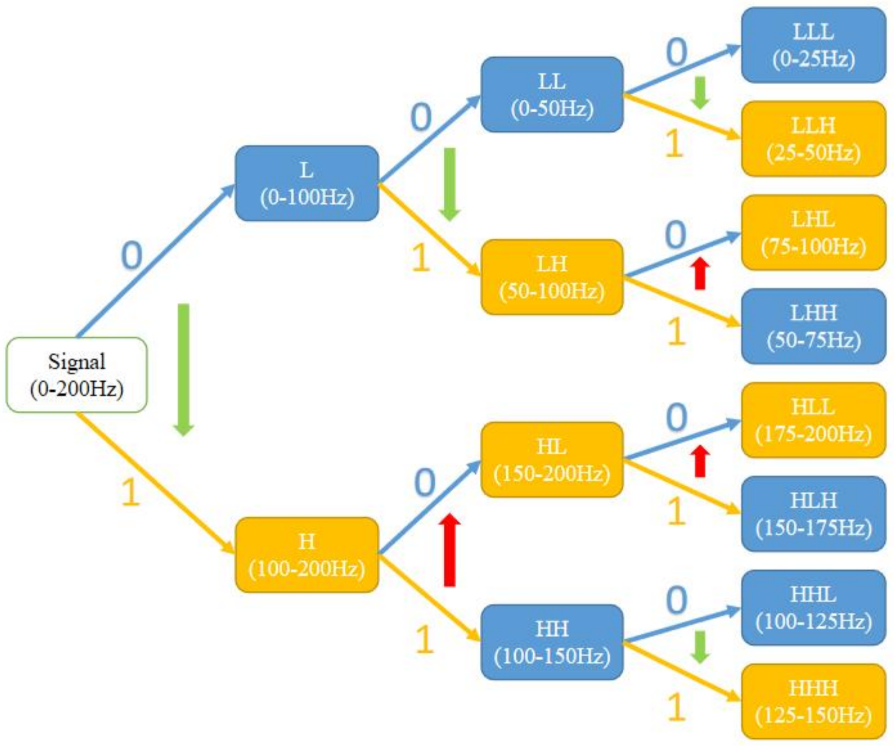


Figure 2.9. The subband tree structure using sparse wavelet decomposition.

Table 2.1 lists the results of decomposing the 0~200Hz signal into three levels using the sparse wavelet decomposition method.

Table 2.1 Complete Wavelet Decomposition

Frequency Band (Hz)	Gray Code	Decomposition Path
0 ~ 25	000	LLL
25~50	001	LLH
50~75	011	LHH
75~100	010	LHL
100~125	110	HHL
125~150	111	HHH
150~175	101	HLH
175~200	100	HLL

It is worth noting that, in order to use the sparse wavelet decomposition method to decompose the signal to find a specific band, the decomposition level *N* must first be calculated according to the upper and lower limits of the frequency of the band to be extracted. To calculate the decomposition level *N*, first we need to limit the upper and lower frequency edges N_{max} and N_{min} of the decomposition level. The calculation methods of N_{min} and N_{max} are shown in Equations (15) and (16):

$$N_{min} = log_2 \frac{0.5f_s}{f_H - f_L} \tag{15}$$

$$N_{max} = N_{min} + L_{max} \tag{16}$$

where f_s is the sampling frequency of the original signal. f_H is the upper frequency edges of the target frequency band. f_L is the lower frequency edges of the target frequency band. L_{max} is the number of maximum allowable steps.

Then, we can find the optimal decomposition level N_{best} by minimizing the difference err between the decomposition frequency band and the upper and lower frequency edges of the target frequency band. The calculation method of err is shown in Equations (17)~(20):

$$\text{err} = \min\{\max(f_L - f_{Lt}, f_{Ht} - f_H)\} \tag{17}$$

$$f_{Lt} = floor\left(\frac{f_L}{\Delta f}\right)\Delta f \tag{18}$$

$$f_{Ht} = ceil\left(\frac{f_H}{\Delta f}\right)\Delta f \tag{19}$$

$$\Delta f = \frac{f_s}{2^N},\ N \in [N_{min}, N_{max}] \tag{20}$$

where f_{Lt} and f_{Ht} are the lower frequency and the upper frequency edges of the frequency bands located by the sparse wavelet decomposition method. Δf given by equation (20) is the band resolution at decomposition of level *N*. define the desired minimum error as formula (21). The parameter λ is a percentage that determines the minimum error as the target frequency band.

$$err_{min} = \lambda(f_H - f_L) \tag{21}$$

The calculation method of Nbest is specifically shown in the pseudocode Algorithm 4.

Algorithm 4 Algorithm About Finding the Best Decomposition Level

```
1:  FUNCTION find_Nbest($N_{min}$, $N_{max}$, $err_{min}$, $f_s$, $f_H$, $f_L$)
2:    FOR $N$ from $N_{min}$ to $N_{max}$ DO
3:        SET $\Delta f$ to $f_s/2^N$
4:        SET $f_{Lt}$ to floor$(f_L/\Delta f)\times\Delta f$
5:        SET $f_{Ht}$ to ceil$(f_{Ht}/\Delta f)\times\Delta f$
6:        SET err to min$\{\max(f_L - f_{Lt}, f_{Ht} - f_H)\}$
7:        IF err < $err_{min}$ then
8:            SET $N_{best}$ to $N$
9:        ELSE:
10:           SET $N_{best}$ to $N_{max}$
16:        END IF
17:   END FOR
18:   RETURN $N_{best}$
19: END FUNCTION
```

In order to avoid unnecessary calculations caused by redundant decomposition, we also need to perform a merge operation on Gray codes. As shown in Figure 2-10, if we find the 25~150 Hz frequency band in the 0~200 Hz frequency band, the Gray code we need is [001, 011, 010, 110, 111]. But through observation, it is not difficult to find that the Gray code "011" and "010" have covered all the frequency bands represented by the Gray code "01". Similarly, the Gray code "110" and "111" have also covered the Gray code "11". All frequency bands indicated. Therefore, the Gray code [001, 011, 010, 110, 111] used to represent 25~150Hz can be simplified to [001, 01, 11].

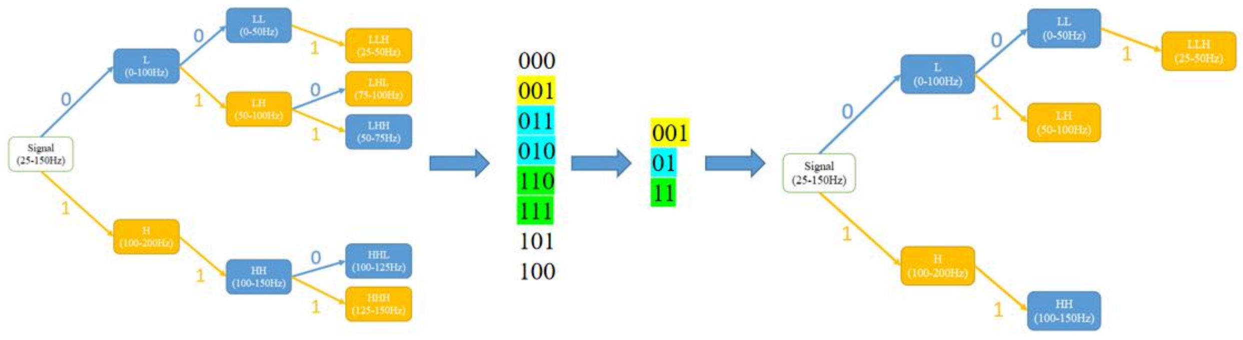


Figure 2.10. Merged operation based on Gray code.

3. SIGNAL FEATURE CLASSIFICATION

In Chapter 3, several signal processing methods are proposed, which intend to solve the problem of frequency spectral shift of bearings at different speeds. In this section, we will focus on how to use two important models such as , convolutional neural network (CNN) and graph neural network (GNN), to diagnose and classify bearing faults.

3.1 One-Dimensional Convolutional Neural Network

One-dimensional convolutional neural network (1D-CNN) is a commonly used deep neural network model for feature learning of one-dimensional signals. The general structure of the one-dimensional CNN is shown in Figure 3.1;and Table 3.1 lists the detailed information of each layer of the ID-CNN we constructed for classifying bearing faults.

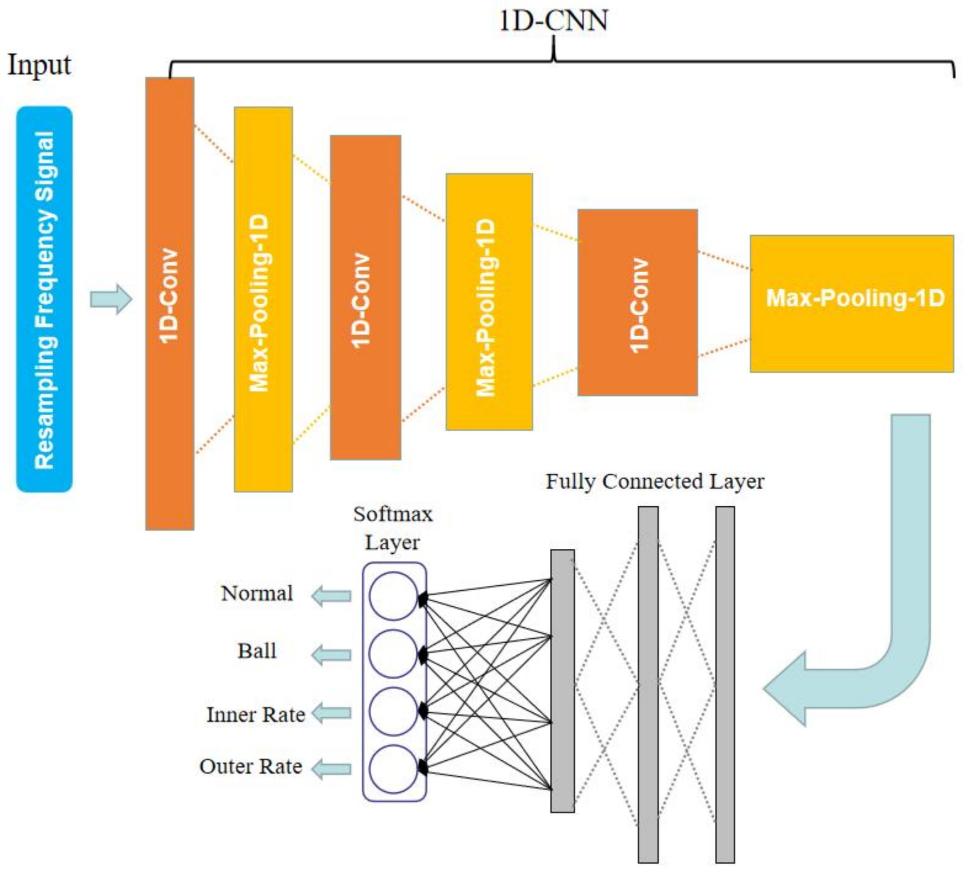


Figure 3.1 Bearing fault diagnosis based on 1D-CNN.

As shown in Figure 3.1, we input the frequency domain signal of the bearing after the frequency resampling method mentioned above as a one-dimensional matrix into the ID-CNN. The signal data undergoes three times of convolution and pooling processing to condense the main features and then next layer is the fully connected layer, and finally the Softmax layer determines the classification result. It is worth noting that in order to verify that our method can tacklethe problem of variable speed bearing fault diagnosis, the verification set must use another data set different from the training set that contains the signals generated by the same type of bearing at other speeds.. We collected a total of 4 classification datasets for the same bearing at four different rotational speeds. Experiments have show that using the frequency spectral resampling proposed in this research, the ID-CNN is trained only with a four-category data set of one rotational speed, and the other three rotational speed data sets are used in turn as the verification data set, we finally get close to 100 % correct rate. This validates that our method is successful for fault diagnosis for bearings at different speeds.

Table 3.1 Information of Layers in 1D-CNN Network

Layer Name	**Parameters (Kernel Size, Feature Maps, Activation Function, Strides, Padding)**
Conv-1	Conv1D(4×1, 32, ReLU, 240×1, Same)
Conv-1	Conv1D(4×1, 64, ReLU, Same)
Conv-1	Conv1D(4×1, 128, ReLU, Same)
Max-pooling-1	MaxPooling1D(4×1, None, None, 1, None)
Max-pooling-2	MaxPooling1D(4×1, None, None, 1, None)
Max-pooling-3	MaxPooling1D(4×1, None, None, 1, None)
Fully-Connected	Dense(256, ReLU, None, None)
Softmax(Output)	Dense(99, Softmax)

3.2 Graph Neural Network

Graph deep learning is a deep learning method based on a neural network with a graph structure. Because the graph structure itself has the function of connecting the ontology of things, combined with the use of graph deep learning to extract the feature vector of each node of the graph structure, this makes GNN a powerful relationship searcher. We believe that when using machine learning algorithms to diagnose and classify bearing faults, the algorithm will take into account the correlation between the effects of the same fault in the three frequency bands. Based on this idea, we propose a method to connect the frequency domain signals to construct a graph structure for each of three fault frequency bands and the sparse graph connection. In addition, we also tested different GNN algorithms and found a specific GNN algorithm suitable for extracting signal graph node features, that is, GraphSAGE.

3.2.1 Horizontal Visibility Graph

In order to connect the signals in each of the three frequency bands of the bearing information into a graph, it is first necessary to find a way to connect the signals into a graph. Here we use the horizontal visibility graph (HVG) algorithm.

HVG is a simplified and efficient method that can transform a time series into a graph in terms of the horizontal visibility relationship between data points while maintaining dynamic properties [24].

In general, a graph *G* can be represented by a triple *G* = (*V, E, A*). *V* = $\{v_1, v_2, \ldots v_N\}$ denotes the set of nodes in the graph *G,* v_i represents the attribute value of the *ith node*; *E* = $\{e_1, e_2, \ldots e_N\}$ designates the set of edges in the graph *G,* e_i is the attribute value of the *ith* edge; *A* represents the adjacency matrix, which defines the connection relationship between the nodes in the graph *G.* According to the HVG algorithm, an undirected edge between two nodes only exists when there is no middle data height intersecting the horizontal connection between v_1 and v_2. The adjacency matrix of *A* is given by Equation below [30]:

$$f(v_i, v_j) = \begin{cases} A_{ij} = 1, no\ value\ between\ v_i\ and\ v_j \\ A_{ij} = 0, otherwise \end{cases} \tag{22}$$

An illustration of the HVG graph is shown in Figure 3.2. Figure 3.2 depicts the bar chart of a time series {0.3, 0.7, 0.5, 0.9, 0.6, 0.7, 0.4} and indicates the connection status between the nodes. Taking the node 4 (marked red) in Figure 3.2 as an example, we can see that at node 4, nodes 2, 3, 5, 6 are horizontal visible. Thus, there should have undirected edges between node 4 to nodes 2, 3, 5, 6. But for node 1, node 2 and node 3's values are between node 1's value and node 4's value. The same is true for node 7. So nodes 1, and 7 are not horizontal visible to node 4. So there should not be any edges between nodes 1, 7 and node 4.

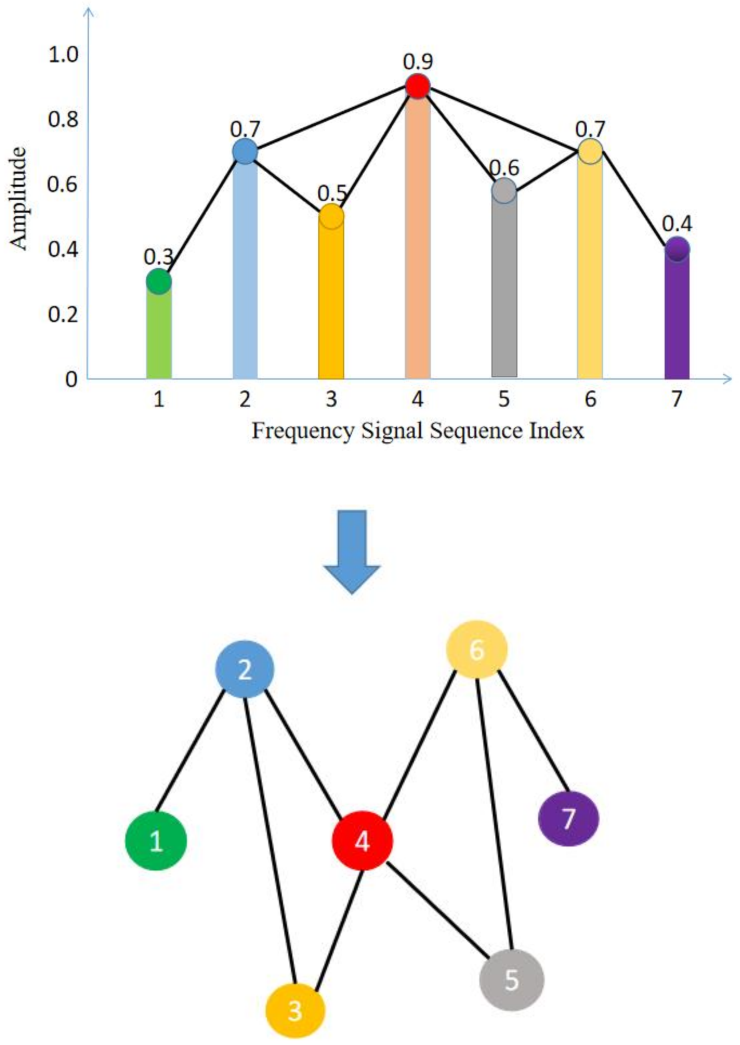


Figure 3.2. Horizontal visibility graph method.

According to the HVG algorithm, each node can at least connect the adjacent nodes on both sides, so HVG is a connected graph. However, according to the graph generated by the HVG algorithm, the nodes in the graph will only be connected to nearby nodes in the time series data. Because for each node, the continuous rise of the latest signal fluctuation before it and the continuous decline of the latest signal fluctuation after it constitute the range of nodes that this node can connect to. As shown in Fig. 3.2, for node 4, the continuous rise of signal fluctuations at 1, 2 and the continuous fall at 6, 7 constitute the range of nodes that node 4 can be connected to in the timing sequence. One of the advantages of using the HVG algorithm is that it can remain unchanged under affine transformation, so different sampling frequencies and normalization methods will not affect the topology of the HVG graph. The basic implementation of the HVG algorithm is shown in Algorithm 4

Algorithm 5 Horizontal Visibility Graph

```
1: FUNCTION createHVG(data):
2:     SET vertices to an empty list
3:     SET edges to an empty list
4:     FOR i = 1 to length(data) - 1:
5:         FOR j = i + 1 to length(data):
6:             IF (data[j] >= data[i]):
7:                 SET intervening to False
8:                 FOR k = i + 1 to j - 1:
9:                     IF (data[k] >= data[i] and data[k] >= data[j]):
10:                        SET intervening to True
11:                        BREAK LOOP
12:                IF (not intervening):
13:                    APPEND tuple (i, j) to edges
14:    FOR i = 1 to length(data):
15:        APPEND tuple (i, data[i]) to vertices
16: RETURN tuple (vertices, edges)
17: END FUNCTION
```

3.2.2 Sparse Graph Connection

Considering the possible association between the three sparsely decomposed frequency bands (band 1, band 2, band 3) for bearing fault detection, We propose a sparse graph connection technique to connect the graphs of the three frequency bands generated using the HVG algorithm.

Figure 3.3 indicates the method we use to connect the three graphs from three frequency bands. Nodes I, II, III represent the three core nodes that are fully connected to the graph band 1, band 2, and band 3, respectively. At the same time, core nodes 1, 2, and 3 are connected to each other to form a fully connected graph with only three nodes (as shown in the core cluster part in Figure 3.3) as the information exchange center between graph band 1, band 2 and band 3.

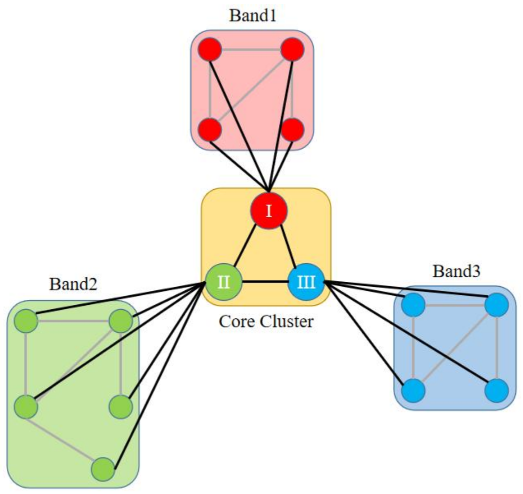


Figure 3.3. Illustration of sparse graph connection.

Comparing with the full connection of nodes between graphs, using core cluster as a transit point between sparse graphs of different frequency bands can greatly reduce the number of connections in the combined graph while maintaining connectivity. Figure 3.4(a) indicates the adjacency matrix using the fully connected method. In the Figure 3.4(a), the red matrix represents the adjacency matrix of the graph band1, band2 and band3 generated using the HVG algorithm; the yellow and blue matrices represent the adjacency matrix formed by the full connection of nodes between graphs. Figure 3.4(b) indicates the adjacency matrix using connection method with core cluster as interaction center. In the Figure 3.4(b), the red part still represents the adjacency matrix of graph band 1, band 2, and band 3; the green part represents the edge where band1 connects to core-I; the purple part represents the edge where band 2 connects to core-II; the gray part represents the edge where band3 connects to core-III; The yellow matrix in the lower right corner represents the fully connected adjacency matrix between core nodes. By comparing Figure 3.4(a) and Figure 3.4(b), it is not difficult to find that compared with full connection, the use of core nodes for graph connection can greatly reduce the number of edges used for graph connection. In comparison with the full connection method, the advantages of using core nodes for graph connection can be summarized as the following two points:

(1) *Reduces the number of irrelevant node connections between graphs:* There is no way to know the correlation of nodes between graphs, and blindly adopting full connections will lead to a large number of useless connections being introduced into the graph. Valuable connectivity

information that contributes to classification results will be diluted. Using core nodes as connection centers can minimize the number of useless connections while ensuring the connectivity of the entire graph.

(2) *Reduce the amount of calculation and improve the calculation speed:* Comparing Figure 3.4(a). with Figure 3.4(b), it can be seen that using the core node-centered connection method can greatly reduce the number of edges used for graph connection, thereby greatly reducing the scale of the graph and improving the calculation speed.

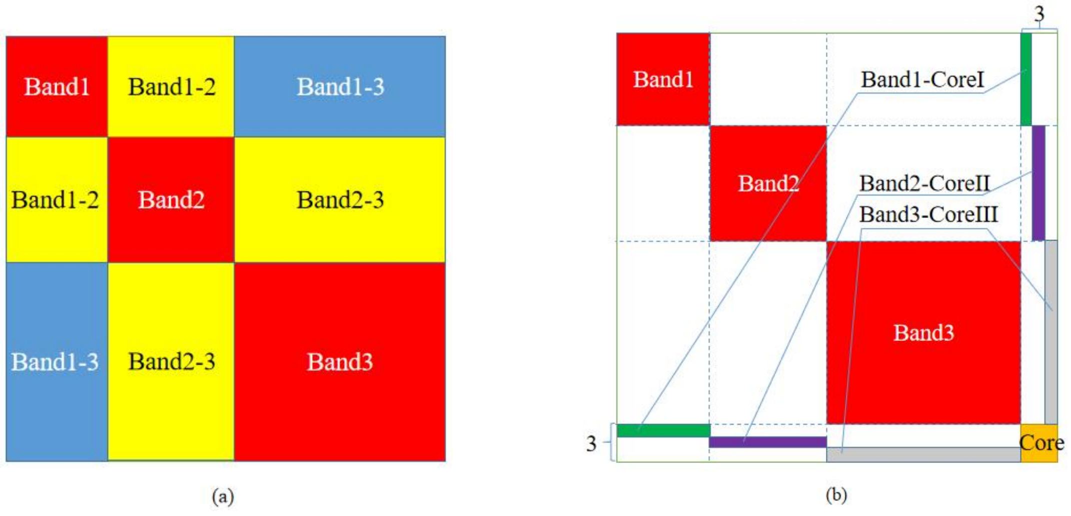


Figure 3.4. Illustration of sparse graph connection. (a) The adjacency matrix using the fully connected method; (b) The adjacency matrix using connection method with core cluster as interaction center.

3.2.3 Graph Convolutional Network

Graph convolutional network (GCN) is a neural network designed to operate on graph-structured data in order to extract features from nodes in the graph. GCNs work by exploiting the structure of a graph or network to perform convolution-like operations. Unlike traditional convolutional neural networks that operate on grid-like data such as images, the GCN operates on irregular graph-structured data, such as social networks, knowledge graphs, etc. [33].

The main idea of GCN is to split the graph structure into a tree structure, that is, taking the extracted node as the root node, and taking the N-hop neighbor nodes of the node in the graph structure as the leaf nodes of different layers. Then, the features of the N-hop adjacent nodes of the extracted node are aggregated layer by layer from bottom to top to obtain the feature vector of the extracted node.

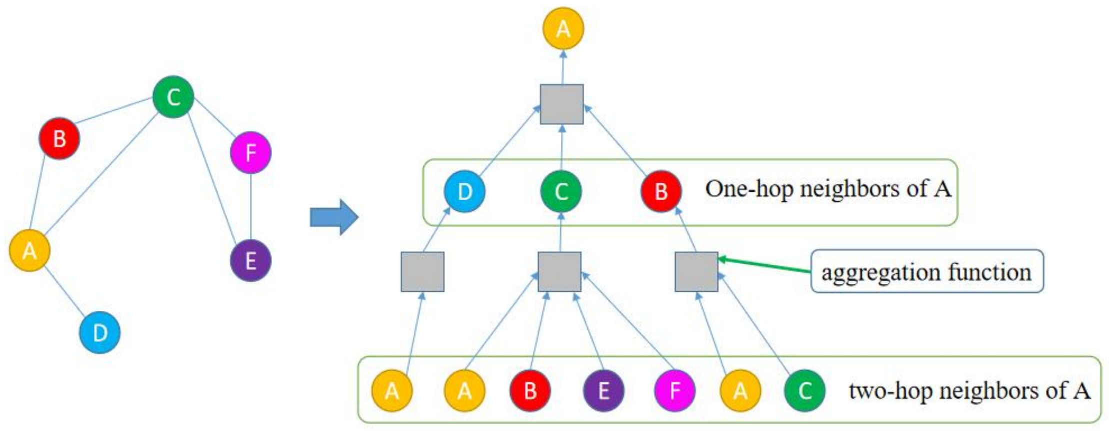


Figure 3.5. The basic operating structure of GCN.

Take a graph with six nodes as an example, as shown on the left side of Figure 3.5. When extracting the feature vector of point A, the tree structure formed with A as the root node and A's one-hop and two-hop neighbor nodes as leaf nodes is shown on the right side of Figure 3.5. [D, C, B] as a one-hop neighbor node of node A is represented in the first level of the tree as a child node of A. [A, A, B, E, F, A, C] as a two-hop neighbor node of node A is represented as a child node of [D, C, B] in the second level of the tree. The gray square in Figure 3.5 represents the aggregation function, which is used to aggregate the feature information of the lower nodes in the tree and pass it to the upper nodes. The aggregation formula is given by Equation (23) below:

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{\|N(v)\|} + B_k h_v^{k-1}\right), \forall k \in \{1, \dots, K\} \tag{23}$$

In Equation (23), *u* is the set of child nodes of node *v* in the tree structure. h_v^k is the feature vector of the node *v* to be calculated. $N(v)$ is the number of child nodes of node *v*. h_v^{k-1} is the previous layer embedding of *v*. σ is a nonlinear function, such as ReLU. It is worth noting that there are two weight parameters W_k and B_k that need to be learned. As shown in Figure 3.6, W_k represents the weight leading to node *v*. B_k represents the autocorrelation weight of node *v*, which is used to consider the current own feature vector. From this, it is not difficult to see that the aggregation function of GCN is mainly composed of two components, that is, one is used to consider the feature information from other nodes, and the other is used to consider its own feature information.

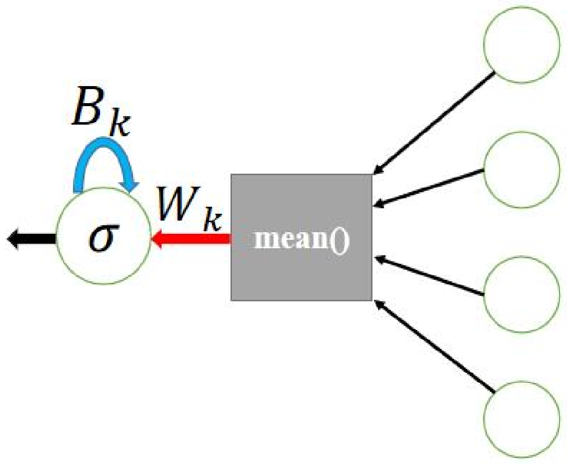


Figure 3.6. Aggregation calculation of GCN.

Our purpose is to use a graph neural network to classify the graph formed by each piece of signal information, so the problem is essentially a graph classification problem rather than classifying a single node in each graph. Therefore, we need a feature vector to describe a graph. The operation of converting multiple vectors of graph nodes into graph representation vectors is called READOUT. The way of READOUT is not unique. The simplest way is to average the output of the last layer of GCN, that is, the vector representation of the node. There is also a weighted summation after averaging the outputs of each GCN layer (including the initial input), as shown in Figure 3.7.

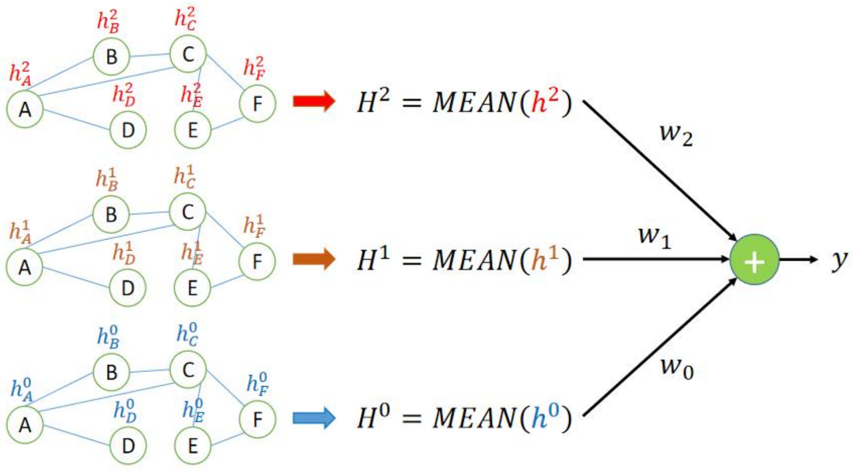


Figure 3.7. Graph feature vectors.

3.2.4 GraphSAGE

In addition to the GCN, another graph embedding algorithm, Graph neural network with self-attention is adopted (GraphSAGE). This method was proposed by Hamilton in 2017 [29]. The main idea behind GraphSAGE is to aggregate local neighbor nodes from nodes in the graph to get feature vectors of corresponding nodes.

GraphSAGE uses a novel sampling strategy that enables it to scale to large graphs. Instead of processing the entire graph at once, GraphSAGE samples a fixed number of neighbors for each node in each layer of the neural network. This enables GraphSAGE to compute embedding of very large graphs without running out of memory.

As shown in Figure 3.8. Similarly, we use a simple graph with six nodes as an example. Taking the extracted feature vector of node A as an example, each node is limited to only consider its three neighbor nodes. Still take the construction of the two-hop adjacent node tree of node A as an example. Node C has 4 one-hop nodes in total. Under the limitation of only considering three neighbor nodes, one neighbor node should be randomly discarded. In Figure 3.8, the neighbor node randomly discarded by node C is B. On the contrary, both node D and node B have less than 3 one-hop nodes, so a certain number of neighbor nodes need to be copied randomly to make up to 3. For node D, it has only one neighbor node A, so it needs to copy A twice to make up the number of neighbor nodes to three. For node B, it has two neighbor nodes A and C, so it is necessary to randomly copy a node in node A and C to complement B's neighbor nodes to three. In Figure 3.8, the neighbor node randomly copied by node B is C. This complementary operation enables the GraphSAGE algorithm to predict future nodes when applied to an evolving graph, which makes GraphSAGE have a higher generalization ability than GCN.

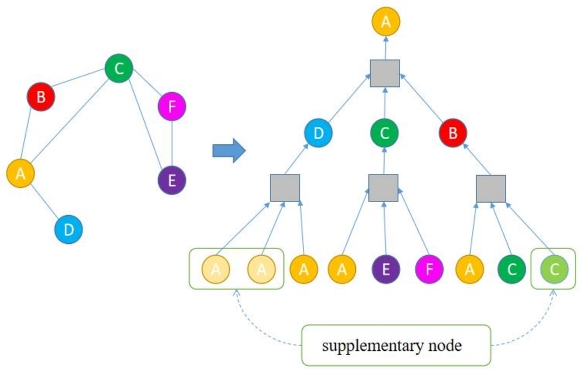


Figure 3.8. The basic operating structure of GraphSAGE.

In addition, the GraphSAGE also uses two new aggregation methods comparing to GCN.

(1) Aggregation operations by Max-pooling.

Similar to the pooling operation in a convolutional neural network (CNN), the GraphSAGE uses the Max-pooling method to aggregate feature information of neighbor nodes, as shown in Equation (24). In other words, the GraphSAGE uses the method of taking the maximum value by dimension instead of the method of taking the average value of an dimension in the original GCN to aggregate the feature vector of the node.

$$h_v^k = \sigma\left(W_k max(h_u^{k-1}) + B_k h_v^{k-1}\right), \forall k \in \{1, \ldots, K\} \tag{24}$$

(2) Aggregation operations by language processing models.

Another newer aggregation method proposed for GraphSAGE than GCN is to use natural language processing models, such as LSTM, to aggregate feature information of nodes. The specific method is shown in Equation (25). We need to build a natural language model with multiple inputs and single output. The number of inputs is equal to the number of feature vectors to be aggregated, and the number of outputs is 1. The feature vectors to be aggregated are input into the LSTM as a sequence, and the output feature vector is the feature vector of the selected node. Figure 3.8 shows the process of using LSTM to gather the feature information of nodes D, C, and B in Figure 3.9 to A.

$$h_v^k = \sigma\left(W_k LSTM(h_u^{k-1}) + B_k h_v^{k-1}\right), \forall k \in \{1, \ldots, K\} \tag{25}$$

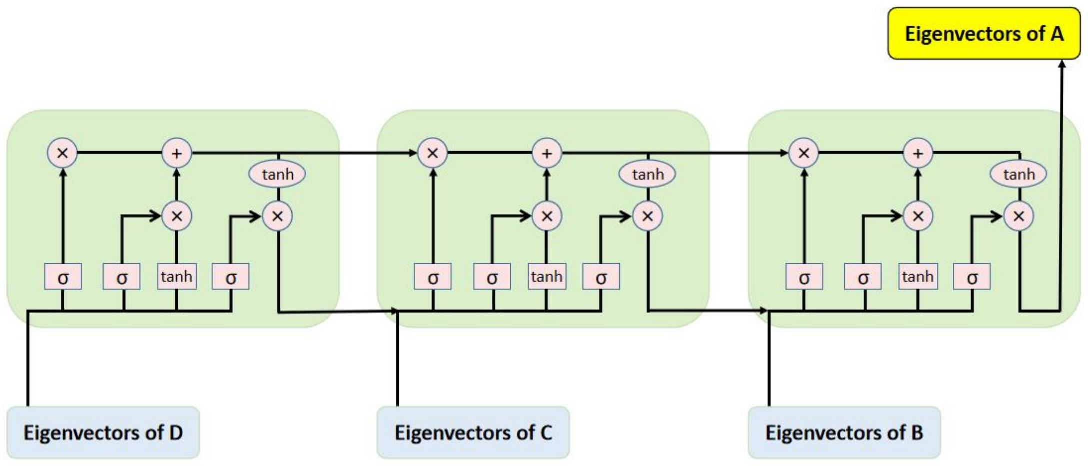


Figure 3.9. Aggregation operations by LSTM.

4. RESULTS AND ANALYSIS

One of the contributions of this thesis research is to design a whole set of architecture from signal acquisition and processing to graph generation and classification, as shown in Figure 4.1.

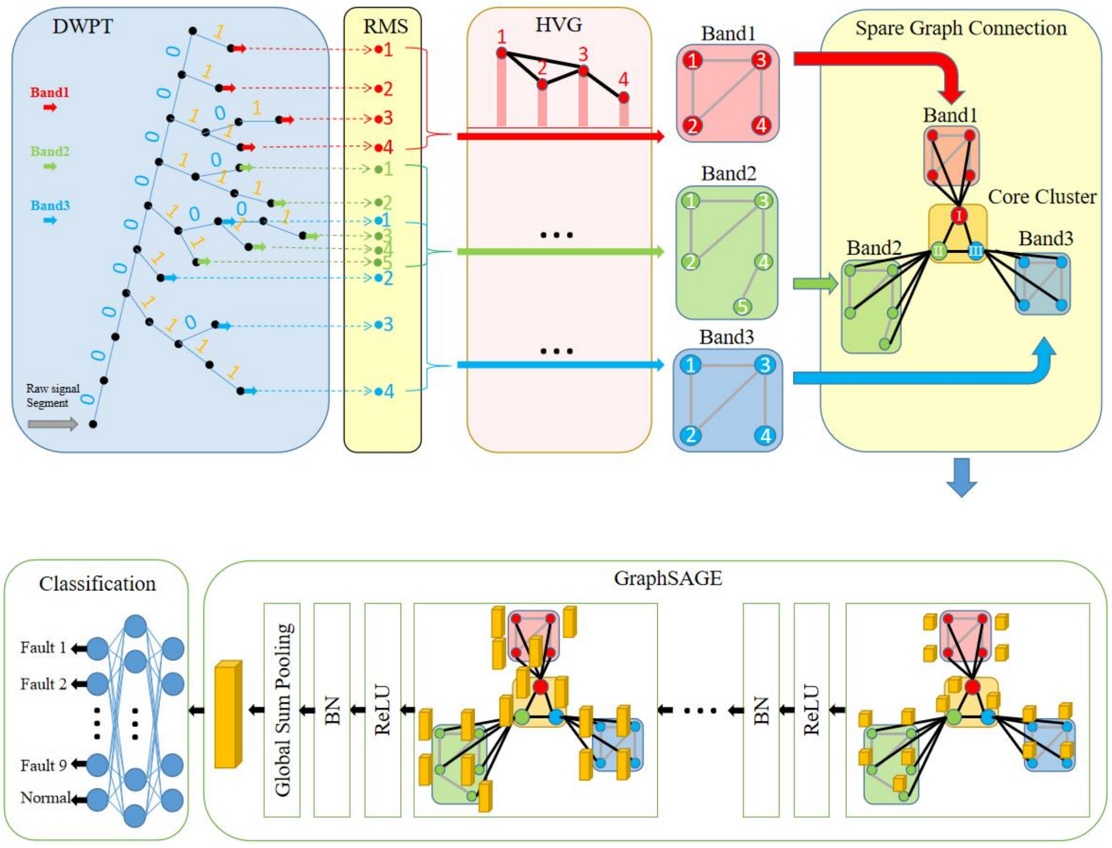


Figure 4.1. Bearing fault diagnosis based on sparse wavelet decomposition and sparse graph connection using GraphSAGE.

Firstly, the collected original signal is split into low, medium and high frequency band data through sparse wavelet decomposition algorithm (DWPT), namely band 1, band 2 and band 3, as shown in the DWPT box in Figure 4.1. In order to reduce the number of nodes in each graph in order to reduce the amount of calculation load, we use the root mean square (RMS) value of each segment of data output from DWPT to replace the data values of the entire segment for subsequent operations. As shown in the RMS block in Figure 4.1, each arrow represents a channel output by the DWPT algorithm, and we will use the root mean square of all values output by each channel as the feature value of the subsequent graph nodes, in other words, the number of nodes in the graph generated by the HVG algorithm is equal to the number of output channels of the DWPT algorithm. Through the HVG algorithm, we can convert the data obtained through the DWPT and RMS into a graph with a specific topology according to the energy level. Because we divide the original signal into band 1, band 2 and band 3 data, each sample will produce three graph structures after the HVG algorithm, as shown in the HVG box in Figure 4.1. Furthermore, we use three additional core nodes to sparsely connect the three graphs as shown in the sparse graph connection block in Figure 4.1.

Finally, the GraphSAGE algorithm is used to extract the feature vectors of the graph nodes for the final classification, as shown in the GraphSAGE box in Figure 4.1

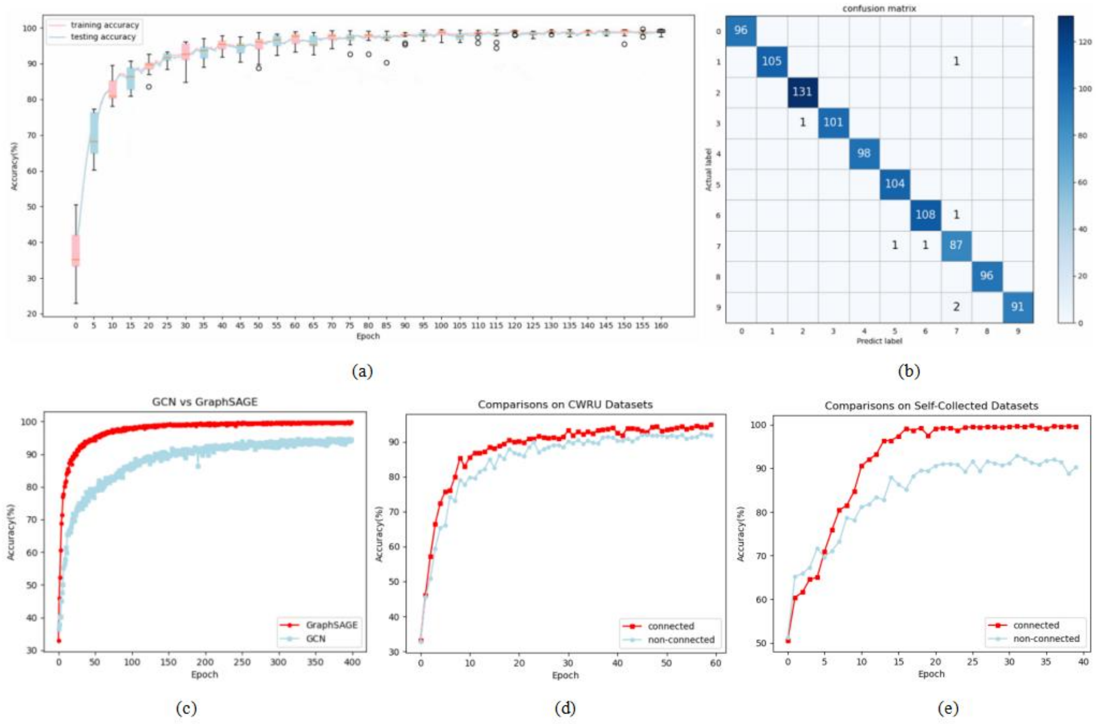


Figure 4.2. Experimental results. (a) Convergence figure for 10 independent trainings; (b) Confusion matrix; (c) Comparison of correctness using GCN and using GraphSAGE; (d) The effect of using the graph connection method on the accuracy rate (CWRU data set); (e) Effect of using graph connection method on accuracy rate (self-collected dataset).

In order to verify the average performance of our designed fault detection model, we conducted ten experiments. For the case of using the Case Western Reserve University (CWRU) bearing dataset, the experimental results are shown in Figure 4.2(a). The average correct rate of ten times of independent runs is 99.73%. Comparing the performance of our proposed method with the other published methods, as shown in Table 4.1, our method shows the most accurate fault classification results. Figure 4.2(b) shows the confusion matrix of the validation set.

Table 4-1 Comparison of Classification Accuracy on CRWU Bearing Dataset

Method	Accuracy (%)
ANN [25]	95.00
APF-KNN [11]	96.67
SVD [12]	85.00
DRSN-CW [13]	89.57
MSCNN [26]	98.53
Multi-scale [17]	93.30
WHVG [24]	99.60
Proposed Method	**99.73**

The effectiveness of the DWPT algorithm has been validated in our previous study [28], so here we only explain why we use the GraphSAGE algorithm as a method to extract graph feature vectors and the effectiveness of graph connections.

Because each node of the graph generated by the HVG algorithm is connected to its nearby nodes in the time series, the graphs generated by the HVG are all locally connected graphs. The GraphSAGE algorithm extracts the feature vector of the graph by aggregating the information of the adjacent nodes of the nodes in the graph. We think this makes the GraphSAGE algorithm very suitable for extracting the feature vector of the graph generated by the HVG algorithm. We compared the results from using GraphSAGE and using traditional GCN (based on the CWRU dataset). As shown in Figure 4.2(c), the results show that the convergence speed and correct rate of training using the GraphSAGE algorithm are significantly improved when compared with the GCN.

Besides, we also verify the effectiveness of the spasre graph connection method. Because one of the advantages of the sparse graph connection method proposed in this research instead of the simple and crude full connection method is that it can greatly reduce a large number of useless edges that may even become interference information. Therefore, when we try to use this method to make the relationship between subgraphs of a graph accounted for, the larger the graph, the more significant the effect will be. In order to increase the scale of the graph, we removed the RMS part and directly used the output data of the DWPT algorithm to generate graphs for training. We conducted the experiments on the CWRU dataset and our own four-category bearing fault dataset, and the results are shown in Figure 4.2 (d) and (e). On the CWRU dataset, using the sparse graph connections increases the accuracy rate by 4%, and on our self-collected dataset, the correct rate increases by 9%. This fully demonstrates that there is a relationship between the three bands. Considering the interaction between different frequency bands is valuable for fault detection and identification of bearings.

To verify the effectiveness of our proposed method, we validated it on CWRU and self-collected datasets, respectively. Table 4.2 and Table 4.3 show the average classification accuracy, standard deviation, F1-score and Recall values after ten runs of independent testing, respectively.

Table 4.2 Accuracy and STD of Different Methods

ACC / **STD**	**GCN**	**GraphSAGE**
Self-collected Dataset	0.9357	0.9958
	0.9372	0.9971
CWRU Dataset	0.9104	0.9973
	0.9106	0.9876

Table 4.3 F1-Score and Recall of Different Methods

F1-score / **Recall**	**GCN**	**GraphSAGE**
Self-collected Dataset	0.9372	0.9971
	0.9372	0.9971
CWRU Dataset	0.9106	0.9876
	0.9106	0.9876

In addition to the methods mentioned above, this thesis also deeply studies and proposes a technique that can copy with the problem of bearing fault diagnosis at different rotating speeds. As shown in Figure 4.3, the original time-domain signal is first transformed into the frequency spectrum using the short-time Fourier transform. After that, the centripetal Centmull-Rom spline method is applied to resample the original frequency spectrum. Then we use the dynamic time warping method to calculate the warping degree between the resampled spectrum and the original frequency spectrum. The result is shown in Figure 4.4(a). The abscissa represents the accuracy, and the ordinate represents the warping degree. Precision 4 is selected corresponding to the turning point of the warping curve as the final resampling frequency. Finally, the resampled frequency spectrum is applied as the input for a one-dimensional CNN for classification. In comparison with the correct rate of classification directly using the original spectrum, the correct rate of classification using the resampled spectrum is significantly improved. Figure 4.4(b) compares the correct rates of using the original frequency spectrum and using the resampled spectrum when the resampling accuracy factor is 4. Figure 4.5 shows the confusion matrix of the classification results using speed 1 as the training set, and then speed 2, speed 3, and speed 4 as the verification sets. In order to verify whether the frequency resampling method is effective, we collected signal datasets of the same type of bearing at four different speeds, and the information of the datasets is listed in Table 4.4. To verify the effectiveness of our proposed frequency resampling method, we compared the results using the original data and using the resampled data based on our self-collected datasets. Tables 4.5 and 4.6 show the average classification accuracy, standard deviation, F1-score and recall values after ten independent testing runs, respectively.

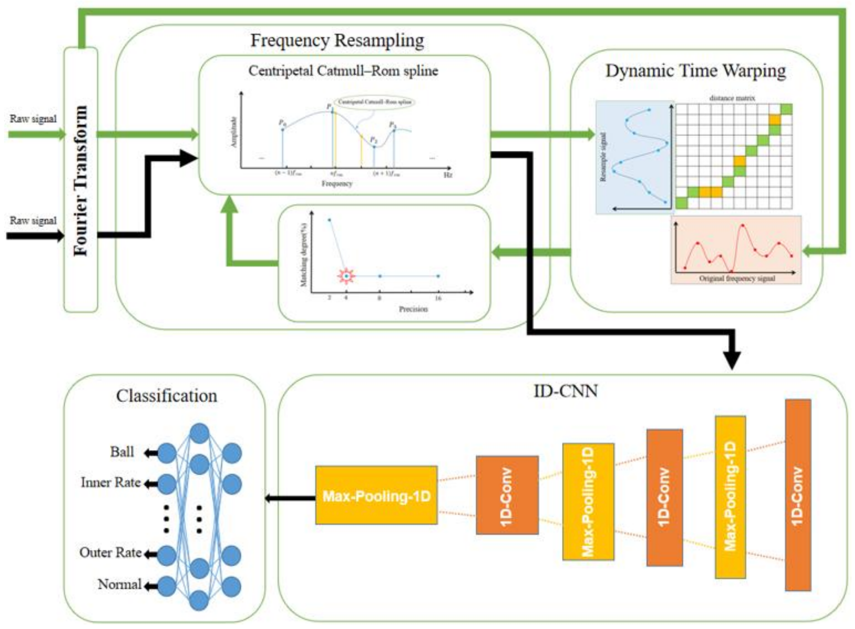


Figure 4.3. Bearing fault diagnosis based on frequency resampling and 1D-CNN.

Table 4.4 Bearing Signal Datasets at Four Rotational Speeds

Dataset	Speed1	Speed2	Speed3	Speed4
RPM	630	930	1575	2200
f_{rm}(Hz)	10.5	15.5	26.25	36.67

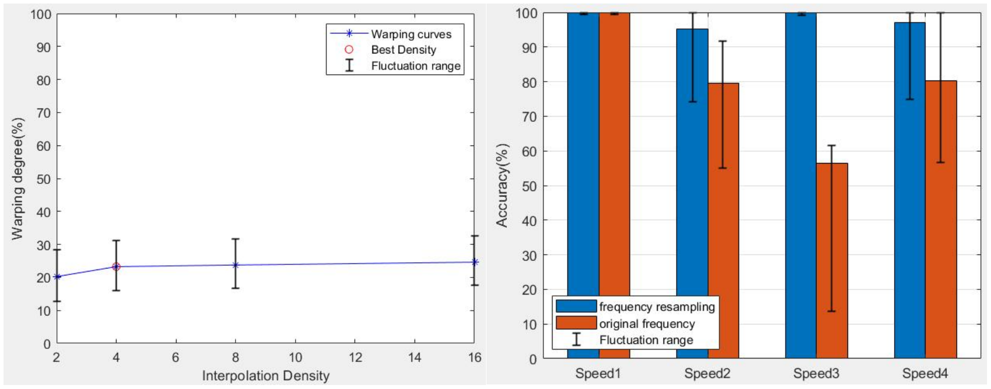


Figure 4.4. (a) Transformation curves of warping degree when the resampling accuracy is set to 2, 4, 8, 16. (b) Comparison of accuracy using original data and using resampled data.

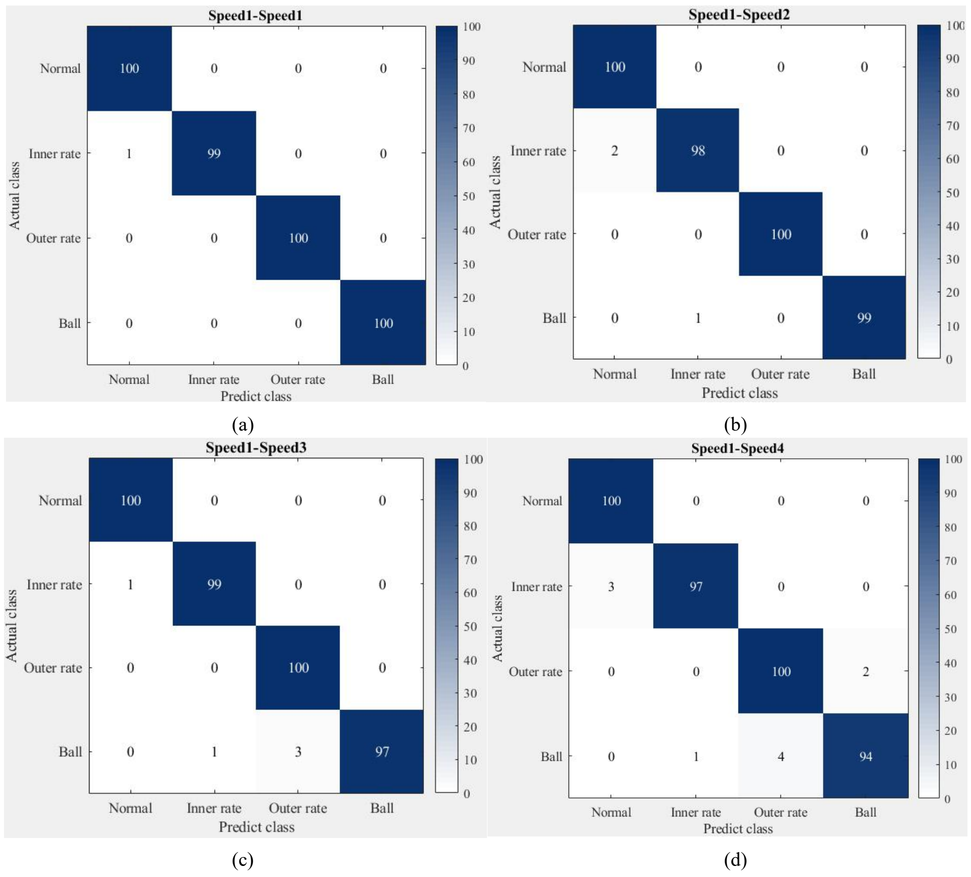


Figure 4.5 Confusion matrix.

Table 4.5 Accuracy and STD of Different Methods

ACC / STD	1D-CNN	GCN	GraphSAGE
Speed1	0.9933	0.9083	0.9069
	0.9778	0.8167	0.8139
FR Speed1	0.9938	0.9431	0.9361
	0.9875	0.8861	0.8722
Speed2	0.7938	0.8833	0.9194
	0.7775	0.7867	0.8389
FR Speed2	0.9675	0.9778	0.9486
	0.9450	0.9500	0.8972
Speed3	0.5564	0.7622	0.7500
	0.5160	0.5644	0.5100
FR Speed3	0.9976	0.8764	0.7792
	0.9633	0.7528	0.5583
Speed4	0.8125	0.6656	0.5722
	0.7912	0.3333	0.1444
FR Speed4	0.9731	0.6153	0.6444
	0.9642	0.2310	0.2889

Table 4.6 F1-Score and Recall of Different Methods

F1-score / Recall	1D-CNN	GCN	GraphSAGE
Speed1	0.9778	0.8167	0.8139
	0.9778	0.8167	0.8139
FR Speed1	0.9875	0.8861	0.8722
	0.9875	0.8861	0.8722
Speed2	0.7775	0.7867	0.8389
	0.7775	0.7867	0.8389
FR Speed2	0.9450	0.9500	0.8972
	0.9450	0.9500	0.8972
Speed3	0.5160	0.5644	0.5100
	0.5160	0.5644	0.5100
FR Speed3	0.9633	0.7528	0.5583
	0.9633	0.7528	0.5583
Speed4	0.7912	0.3333	0.1444
	0.7912	0.3333	0.1444
FR Speed4	0.9642	0.2310	0.2889
	0.9642	0.2310	0.2889

5. CONCLUSION AND FUTURE WORK

In this thesis, we investigate and implement a GNN-based method for bearing fault diagnosis. We propose a sparse graph connection that enables machine learning algorithms to take into account the interrelationships of the effects of different types of faults in different frequency bands for learning the characteristics of bearing fault signals. The method yields 99.73% accuracy on the CWRU dataset, and outperforms other bearing fault detection methods. In addition, we also propose a frequency spectral resampling method, which tackles characteristic frequency shift of the bearing due to the shaft speed changes, making it possible to diagnose bearing faults under different speed conditions. Based on our self-collected four bearing signal datasets at different speeds, using one of the rotational speed datasets as the training set to train the model, it can achieve about 99% classification results on the other rotational speed datasets.

Our future work will use more types of bearings and faults, and different datasets obtained under different speed conditions to conduct comprehensive verification.

REFERENCES

[1] Robert Bond Randall, *Vibration-based Condition Monitoring: Industrial, Automotive and Aerospace Applications.* Second Edition, Wiley, 2021.

[2] Y. Lei, *Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery.* Butterworth-Heinemann, 2017.

[3] M. Benbouzid, M. Vieira, and C. Theys, *" Induction motors' faults detection and localization using stator current advanced signal processing techniques,"* *IEEE Trans. Power Electron*, vol. 14, pp. 14-22, Jan. 1999.

[4] M. El Hachemi Benbouzid, *"A review of induction motors signature analysis as a medium for faults detection,"* *IEEE Trans. Ind. Electron*, vol. 47, no. 5, pp. 984-993, Oct. 2000.

[5] P. Zhang, Y. Du, T. G. Habetler, and B. Lu, *" A survey of condition monitoring and protection methods for medium-voltage induction motors,"* *IEEE Trans*. Ind. Appl., vol. 47, no. 1, pp. 34-46, Jan./Feb. 2011.

[6] M. Cocconcelli, R. Zimroz, R. Rubini, and W. Bartelmus, *"STFT based approach for ball bearing fault detection in a varying speed motor,"* in Proc. Cond. Monit. Mach. Non-Stationary Oper., pp. 41- 50, Oct. 2012.

[7] Z. Peng, F. Chu, and Y. He, *" Vibration signal analysis and feature extraction based on reassigned wavelet scalogram,"* J. Sound Vibrat., vol. 253, no. 5, pp. 1087 – 1100, Jun. 2002.

[8] C. Castejón, M. Jesás, J. J. Gómez, J. Carlos García-Prada, A. J. O. Nez, and H. Rubio, *"Automatic selection of the WPT decomposition level for condition monitoring of rotor elements based on the sensitivity analysis of the wavelet energy,"* Int. J. Acoust. Vib., vol. 20, no. 2, pp. 95 – 100, June 2015.

[9] A. Malhi and R. X. Gao, *" PCA-based feature selection scheme for machine defect classification,"* *IEEE Trans.* Instrum. Meas., vol. 53, no. 6, pp. 1517 – 1525, Dec. 2004.

[10] Y. S. Wang, Q. H. Ma, Q. Zhu, X. T. Liu, and L. H. Zhao, *"An intelligent approach for engine fault diagnosis based on Hilbert – Huang transform and support vector machine,"* Appl. Acoust., vol. 75, pp. 1 – 9, Jan. 2014.

[11] D. H. Pandya, S. H. Upadhyay, and S. P. Harsha, *"Fault diagnosis of rolling element bearing with intrinsic mode function of acoustic emission data using APF-KNN,"* Expert Syst. Appl., vol. 40, no. 10, pp. 4137 – 4145, Aug. 2013.

[12] F. Shen, C. Chen, R. Yan, and R. X. Gao, *"Bearing fault diagnosis based on SVD feature extraction and transfer learning classification,"* in Proc. Prognostics Syst. Health Manage. Conf. (PHM), pp. 1-6, Oct. 2015.

[13] M. Zhao, S. Zhong, X. Fu, B. Tang, and M. Pecht, *"Deep residual shrinkage networks for fault diagnosis,"* IEEE Transactions on Industrial Informatics, vol. 16, no. 7, pp. 4681-4690, July 2020.

[14] J. Chen, J. Jiang, X. Guo, and L. Tan, *"A self-adaptive CNN with PSO for bearing fault diagnosis,"* Systems Science & Control Engineering, vol. 9, no. 1, pp. 11-22, Dec. 2020.

[15] J. Chen, J. Jiang, X. Guo, and 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, Jan. 2021.

[16] W. Chen and K. Shi, *"Multi-scale attention convolutional neural network for time series classification,"* Neural Networks, vol. 136, pp. 126-140, April 2021.

[17] Y. Yao, S. Zhang, S. Yang, and G. Gui, *"Learning attention representation with a multi-scale CNN for gear fault diagnosis under different working conditions,"* Sensors, vol. 20, no. 4, pp. 1233-1253, Feb. 2020.

[18] L. Tan, J. Jiang, Digital Signal Processing: Fundamentals and Applications, New York: Elsevier, 2018.

[19] X. Liu, Y. Cai, Y. Song, and L. Tan, *"Bearing fault diagnosis based on multi-scale neural networks,"* presented at the 2022 IEEE International Conference on Electro Information Technology (EIT), pp. 80-85, May 2022.

[20] X. Liu, J. Centeno, J. Alvarado, and L. Tan, "*One dimensional convolutional neural networks using sparse wavelet decomposition for bearing fault diagnosis,*" IEEE Access, vol. 10, pp. 86998-87007, Aug. 2022.

[21] N. Chimitt, W. Misch, L. Tan, A. Togbe, and J. Jiang, "*Comparative study of simple feature extraction for single-channel EEG based classification,*" in 2017 IEEE International Conference on Electro Information Technology (EIT), pp. 166-170, May 2017.

[22] J. Dai, V. Vijayarajan, X. Peng, L. Tan, and J. Jiang, "*Speech recognition using sparse discrete wavelet decomposition feature extraction,*" in 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 0812-0816, May 2018.

[23] J. Dai, Y. Zhang, J. Hou, X. Wang, L. Tan, and J. Jiang, "*Sparse wavelet decomposition and filter banks with CNN deep learning for speech recognition,*" in 2019 IEEE International Conference on Electro Information Technology (EIT), pp. 098-103, May 2019.

[24] C. Li, L. Mo and R. Yan, "*Fault diagnosis of rolling bearing based on WHVG and GCN,*" IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-11, Jun. 2021.

[25] X. Zhao, M. Jia, J. Bin, T. Wang and Z. Liu, "*Multiple-order graphical deep extreme learning machine for unsupervised fault diagnosis of rolling bearing,*" IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-12, Dec. 2021.

[26] M.-M. Chow, P. M. Mangum, and S. O. Yee, "*A neural network approach to real-time condition monitoring of induction motors,*" IEEE Trans. Ind. Electron., vol. 38, no. 6, pp. 448-453, Dec. 1991.

[27] G. Jiang, H. He, J. Yan, and P. Xie, "*Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox,*" IEEE Trans. Ind. Electron., vol. 66, no. 4, pp. 3196-3207, Apr. 2019.

[28] G. Zhu, X. Liu and L. Tan, "Bearing fault diagnosis based on sparse wavelet decomposition and sparse graph connection using GraphSAGE," 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, pp. 1-6, February 2023.

[29] Hamilton, Will, Zhitao Ying, and Jure Leskovec, “*Inductive representation learning on large graphs,*” Advances in neural information processing systems 30 (2017).

[30] D. Zhang, E. Stewart, M. Entezami, C. Roberts and D. Yu, “*Intelligent acoustic-based fault diagnosis of roller bearings using a deep graph convolutional network,*” Measurement, vol. 156, pp. 107585, May 2020.

[31] Yuksel, Cem, Scott Schaefer, and John Keyser. “*On the parameterization of Catmull-Rom curves.*” In 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, pp. 47-53, October 2009.

[32] Müller, Meinard. “*Dynamic time warping.*” Information retrieval for music and motion, pp. 69-84, 2007.

[33] Kipf, Thomas N., and Max Welling. “*Semi-supervised classification with graph convolutional networks.*” International Conference on Learning Representations. 2017.

PUBLICATIONS

G. Zhu, X. Liu, and L. Tan, “Bearing fault diagnosis based on sparse wavelet decomposition and sparse graph connection using GraphSAGE,” 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, pp. 1-6, February 2023.