CHAPTER 1

INTRODUCTION

Electric machines are used widely in many commercial and industrial applications. The total number of operating machines in the world increased by about 50% in the last five years reaching a number around 16.1 billion in 2011.Bearings are one of the most widely used elements in machines and their failure is the single biggest cause for machine breakdowns.Therefore, there is a great amount of research effort directed towards monitoring of bearing health. Motor vibration analysis and motor current signature analysis are two commonly used noninvasive methods in bearing condition monitoring.

The feature extraction and feature-based classification are the two unique and distinct phases of decision support systems. The proposed method does not require any form of transformation, feature extraction, and postprocessing. The feature extraction and feature based classification phases of the bearing fault detection could be combined into a single learning body with CNN. It can directly work over the raw data, that is, the motor current signal, to detect the anomalies. Here, the time-domain vibration data is fed directly into the CNN which is previously trained with test data using back-propagation (BP). The only preprocessing involved in the process is resampling to get raw data in the desired input format for the CNN.

The fault diagnosis of rotating machinery has been of great practical significance in industrial applications to avoid economic losses and enhance machine availability. 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.

1.1 Research and Motivation

The fault diagnosis of rotating machinery has been of great practical significance in industrial applications to avoid economic losses and enhance machine availability. The vibration signal preprocess is essential 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.

1.2 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 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.

CHAPTER 2

LITERATURE REVIEW

The Literature Review is the part of the dissertation where there is extensive reference to related research and theory in the field Overall, the function of a literature review is to show how related work in the field has shaped and influenced your research.

2.1 Literature review

Tobias Wagner and Sara Sommer[1] Has been Conducted experiment on Bearing fault detection using deep neural network and weighted ensemble learning for multiple motor phase current sources and concluded that developed method is not only applicable to motor currents, but can also be extended to other (internal) motor signals. They evaluated the proposed method experimentally on a public available dataset. As the experimental results showed, there exist a need for applying transfer learning techniques to reduce the loss of accuracy for cross working condition transfer.

Peter W. Tse,Y. H. Peng , Richard Yam [2] They have been conducted experiment on Wavelet Analysis and Envelope Detection For Rolling Element Bearing Fault Diagnosis and concluded that both the WA and the FFT with ED methods are effective in finding the bearing outer-race fault. However, to diagnose the faults of inner-race and roller, the tool of WA is found to be easier for the machine operator to interpret the analyzed results. Both the methods of wavelet analysis and FFT with ED are effective in diagnosing the outer-race defect However, when using FFT with ED, the faults caused by defective inner-race and roller are difficult to identify using visual inspection.

K. Baiche, M. Zelmat, A. Lachouri [3] Done experiment on Application of Multi-Scale PCA and Energy Spectrum to Bearing Fault Analysis and Detection in Rotating Machinery and concluded that the multi-scale principal component and spectral energy methods have been presented. To detect the presence of faults, the squared Prediction Error (SPE) and the spectrum energy of the detail coefficients are used. The SPE is compared with the limit corresponding to 95% of maximum amplitude of the healthy signal and the spectrum energy calculated from the detail coefficients.

M. Bilginer Gu¨lmezog˘lu*,y and Semih Ergin [4] Has been conducted experiment on An approach for bearing fault detection in electrical motors and concluded that the advantage of the proposed approach is that calculation of the common vector and indifference subspace for each class in the training process is simple. This also takes a very short time, approximately equals to 6.9 seconds per class. Very high recognition rates obtained for four- and 10-class problems indicate that CVA is very effective for fault detection and diagnosis of electrical machines.

Yu Guo , Ting-WeiLiu , JingNa , Rong-FongFung [5] They have been conducted experiment on Envelope order tracking for fault detection in rolling element bearings and concluded that this approach takes the advantages of envelope analysis, order tracking and the SK scheme. In the approach, the SK is utilized to realize the self-adaptive envelope extraction and determine the demodulation frequency band at first. the envelope order spectrum is employed to indicate the characteristic orders related with the REB faults clearly.

Jafar Zarei , Mohammad Amin Tajeddini , Hamid Reza Karimi [6] Has been conducted experiment on Vibration analysis for bearing fault detection and classification using an intelligent filter a new algorithm for fault detection and classification, in two steps, is proposed. In the first step, a filter is designed by neural network to omit the non-bearing fault component of the sampled signal. This filter called removing non-bearing fault component (RNFC) filter. In the next step, time-domain features are extracted from RNFC filter output and applied to another neural network in the form of prototype features to detect faults using a pattern recognition technique.

J P dron L rasolofondraibe and C couet, A. pavan [7] They have been conducted experiment on Fault detection and monitoring of a ball bearing benchtest and a production machine via autoregressive spectrum analysis and concluded that This study has established that parametric techniques along with conventional methods bring extra information since they display all of the signals occurring in a vibratory signature This enables the detection of a fault at an early stage and the monitoring of its evolution on a time scale Therefore it is a worthwhile technique at the conditional method level For machine accessibility purposes the

measuring campaign was limited to nine months This time period is not long enough to detect the evolution of the typical frequency amplitudes of identified faults on the machine.

S. A. McInerny and Y. Dai [8] has been conducted experiment on Basic Vibration Signal Processing for Bearing Fault Detection An instructional module on fault detection in rolling element bearings has been described. This module was developed for an existing course in applied spectral analysis, but could be used as a stand-alone tutorial or incorporated into a course on machine condition monitoring. After reviewing the basic operation of rolling element bearings and the characteristics of idealized bearing fault vibration signatures, the shortcomings of conventional spectral analysis were illustrated with a synthetic signal generated. a set of graphically driven procedures developed to illustrate bearing fault detection techniques was presented.

A. Rezig , A. N'diye , A. Djerdir and M.R. Mekideche [9] Done Experimental investigation of vibration monitoring technique for online detection of bearing fault in induction motors The results of the theoretical model show that fault in bearing elements can be traduced by a specific frequency harmonics in the power spectrum of the bearing vibration. These results were confirmed experimentally for the case of outer race defect. The vibration analysis monitoring technique is implemented also for bearing fault detection. This permits us to direct the online detection of these faults. They have noticed that in using vibration monitoring technique, the frequency defect can be shown directly on the spectrum with any intelligent artificial technique. This is a very important advantage of the diagnosis of motor faults by the vibration technique.

Chao Yang, Wentao Mao*, Yamin Liu, Siyu Tian [10] They have been conducted experiment on Incremental Weighted Support Vector Data Description Method for Incipient Fault Detection of Rolling Bearing and concluded that This method integrates incremental learning strategy and weighted learning strategy into SVDD to improve its robust performance for online detection. The most vital innovation is proposing a sample state determination strategy to calculate effective weights for online data. The experimental results show the proposed method can keep the detection results unchanged or even slightly better, with much less false alarm rate.

Aleksandra Ziaja, Ifigeneia Antoniadou, Tomasz Barszcz, Wieslaw J Staszewski and Keith Worden [11] Has been conducted experiment on Fault detection in rolling element bearings using wavelet-based variance analysis and novelty detection and concluded that signal processing tool for automatic fault detection in rolling element bearings. The proposed methodology originates from fractal theory and a novelty detection approach. The wavelet-based variance originally developed for self-similarity analysis and neural networks are used for fault detection. The former is used as a pre-processor for a neural-network-based novelty detector.

Shen zhang , (Student Member, IEEE), Shibo Zhang , (Student Member, IEEE), Bingnan Wang ,(Senior Member, IEEE), and Thomas G. Habetler [12] They have been conducted experiment on Deep Learning Algorithms for Bearing Fault Diagnostics and concluded that Special emphasis is placed on deep learning based approaches that has spurred the interest of the research community over the past five years. It is demonstrated that, despite the fact that deep learning algorithms require a large dataset to train, they can automatically perform adaptive feature extractions on the bearing data without any prior expertise on fault characteristic frequencies or operating conditions, making them promising candidates to perform real-time bearing fault diagnostics. A comparative study is also conducted comparing the performance of many DL algorithm variants using the common CWRU bearing dataset.

Wentao Sui and Dan Zhang [13] Has been conducted Research on Envelope Analysis For Bearings Fault Detection and concluded that The traditional method has some disadvantages. On the one hand, FFT method is widely used in the spectrum analysis of envelope signals. However, it could only give the global energy-frequency distributions and fail to reflect the details of a signal. So it is hard to analyze a signal effectively when the fault signal is weaker than the interfering signal. The methods based on the wavelet transform and morphological filters are introduced in this paper to extract features of defects in bearings. This approach is demonstrated by experiments to be sensitive and reliable in detecting the defects on the outer race, inner race.

Jin Woo Oh, Dogun Park and Jongpil Jeong [14] They have been conducted experiment Fault Detection for Lubricant Bearing with CNN and concluded that a CNN model for the diagnosis of lubricant bearing defects is proposed. Unlike other bearing failures, lubricated bearings are very difficult to detect. Therefore, additional sensors such as thermal cameras should be considered, but using the one of the popular models of the deep neural networks, we propose effective error detection without additional sensors. Except when considering using the CNN model only, it is necessary to consider using a combination of existing CNN and other traditional models. For example, it can be possible to combine CNN with SVM or CNN and RF models to get better performance.

Anand S. Reddy*,Praveen Kumar Agarwal,Satish Chand [15] Has been conducted experiment on Application of artificial neural networks for the fault detection and diagnosis of active magnetic bearings and concluded that An ANN based methodology with statistical analysis is proposed for the detection and diagnosis of three types of faults (bias, multiplicative and noise addition). Nine back propagation neural networks (BPNNs) are used where eight BPNNs are designed to predict the coil currents and one BPNN is designed for predicting the rotor position. Then by statistical analysis of the residuals, faulty sensor or actuator along with the type of the fault is identifiedwith statistical analysis can detect single/multiple faults and can diagnose the type of the faults in sensors and actuators satisfactorily.

Sanjay Kumar, Deepam Goyal, Rajeev K. Dang*,Sukhdeep S. Dhami, B.S. Pabla [16] They have been done experiment on Condition based maintenance of bearings and gears for fault detection the condition based maintenance of gears and bearings for fault detection has been introduced. With advancements to late developments in sensors, signal processing and computing methods, condition based maintenance need turned into a critical field about exploration. Vibration measurement in the frequency domain has the advantage that it can detect the location of the defect. some special cases, the twoconventional maintenance policies, namely the run-to-failure policy and the time-based preventive maintenance can be applied with satisfactory results. However, in many situations, especially when both maintenance and failure are very costly, CBM is absolutely a better choice than the conventional ones.

Michael J. Devaney and Levent Eren [17] conducted an experiment on Monitoring an induction motor's current and detecting bearing failure and concluded that Uncertainty is a pervasive and persistent quality of real-time systems. The total elimination of uncertainty is often impossible, however, because of the complex nature of the systems under control. But rather than admit defeat, a proactive approach to mitigating uncertainty is needed. This approach starts with acknowledging uncertainty's existence and then identifying its possible causes so that the mitigation strategy can be designed.

Jong-Hyo Ahn, Dae-Ho Kwak and Bong-Hwan Koh * [18] They have been conducted experiment on Fault Detection of a Roller-Bearing System through the EMD of a Wavelet Denoised Signal and concluded that In this paper, we perform an experimental validation of a condition monitoring system for a roller-bearing system. Here, we assumed the bearing fault was a scratch-mark on the surface of the inner race. Before performing fault detection, we carried out wavelet-based denoising through a soft-thresholding scheme. Unlike the conventional low-pass filter, the wavelet-based denoising methoddoes not distort the elastic waves due to collision of the defect in bearing. We will extend this study in the near future to more challenging cases, in which multiple sources of faults co-exit, such as a bearing dismantled due to fatigue, and cracks in a gear tooth of a gearbox system.

N. HarishChandra,A.S.Sekhar [19] Has been conducted experiment on Fault detection in rotor bearing systems using time frequency techniques and concluded that The comparative study of the methods is focused towards detecting the least possible level of the fault induced and the computational time consumed. Thestart up vibrations for diagnosis of rotor systems equipped with journal bearings.The use of run- up responses along with new time–frequency techniques provides more information about the existing faults.Faults such as oil whirl and oil whip show different harmonic signatures which can be detected using the techniques presented in this paper.

Based on the above literature, it is ovserved that all researchers used 1D CNN,KNN,SVM as a Deep Learning and Machine Learning models for faults detection. These models have better accuracy in the field of faults ditection. Our Present study focusses on 2D CNN as well as 1D CNN and SVM.

CHAPTER 3

PROBLEM DEFINITION AND OBJECTIVE

The current issue or problem that requires timely action to improve the situation. The problem statement, in addition to defining a pressing issue, is a lead-in to a proposal of a timely, effective solution.

3.1 Problem definition

Rolling-element bearings are one of the most common components in industrial machinery. They are used in almost every rotating mechanism (from electric motors to gearboxes to conveyor belt systems) to allow the transmission of power along load bearing axles with a minimum of losses due to friction.

Rotating machines are widely used in manufacturing industry, operating usually for long time under harsh conditions. Sudden failures occurring on key machine component like bearings may lead to unexpected breakdown of machines and cause economic loss, environmental pollution and human casualties.

Early and accurate detection of defects and failures of such components of rotating machinery is critical to ensure operational reliability and avoid catastrophic accidents in industrial applications.

In order to solve above problems, this paper proposes a fault diagnosis method based on CNN structure.

3.2 Objectives of the Problem

Conventional machine learning (ML) method, support vector machines have been successfully applied to the detection and categorization of bearing faults for decades, recent developments in DL algorithms in the last five years have sparked renewed interest in both industry and academia for intelligent machine health monitoring.

- The goal of this project is to detect and classify the different bearing faults based on the raw vibration data.
- To detect the small cracks or defects within the mechanical components of the bearing, leading to increased the vibration using deep learning.
- To test the accuracy of CNNs as classifiers on bearing fault data, by varying the configurations of the CNN from one-layer up to a deep three-layer model
- To compare Machine Learning algorithm that is Support vector Machine and Deep Learning algorithm that is Convolutional neural network.
- To show that CNN can efficiently extract the feature information contained in massive data, which is very suitable for processing large quantities of data.

CHAPTER 4

METHODOLOGY AND EXPERIMENTATION

The flow chart for methodology This tell how the process carried out step by step.

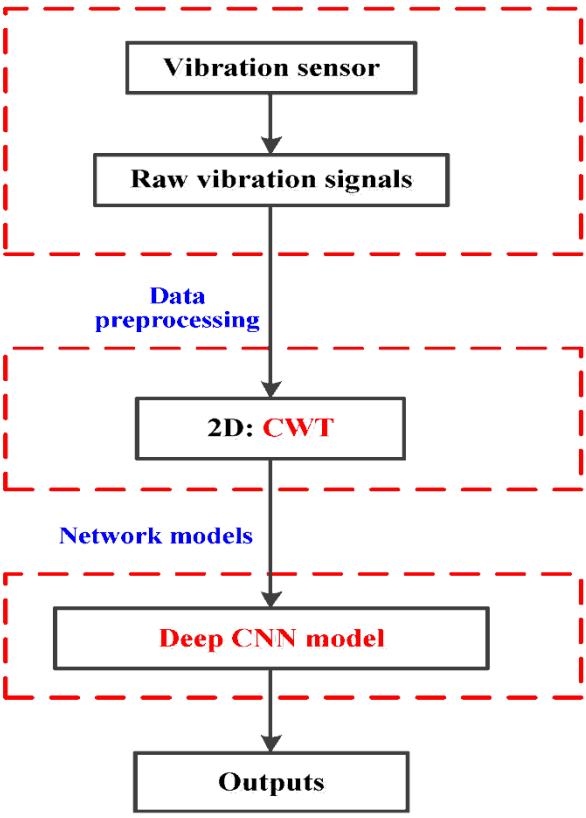

Figure 4.1 Methodology flow chart

4.1 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 Case Western University Dataset often used for bearing remaining useful life predictions. Paderborn dataset and Case Western University Dataset especially

suitable for bearing fault detection. We used Case Western University Dataset for the experiment.

4.1.1 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. Figure 4.2 describes the experimental platform.

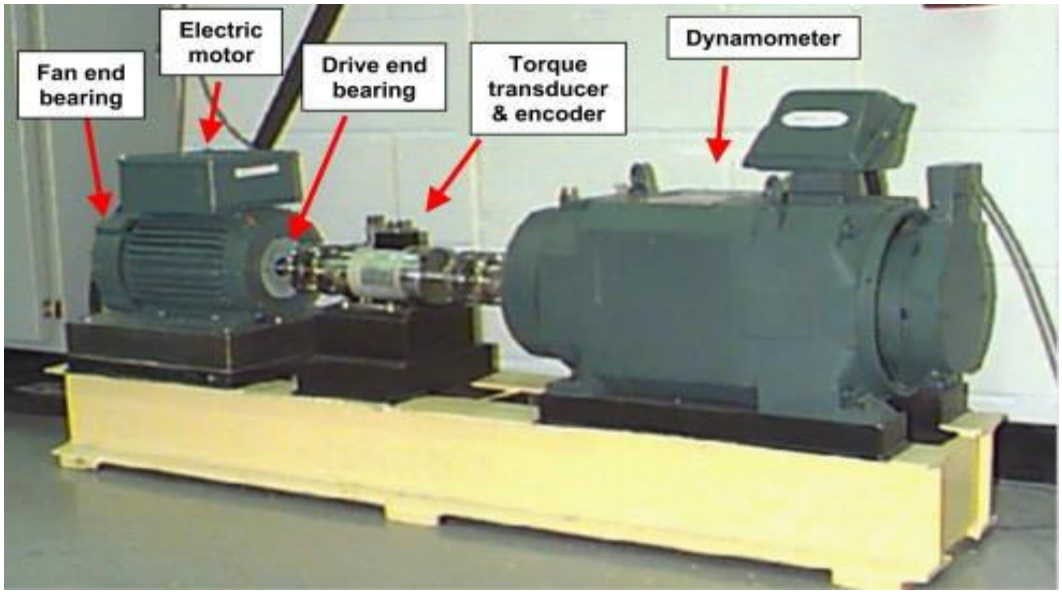

Figure 4.2 Experimental setup for CWRU bearing dataset

The bearings support the motor shaft. The single-point artificial faults were seeded in the bearing's inner raceway (IR), and outer raceway (OR) with an electrodischarge machining (EDM) operation. Figure 4.2 depicts the structure of bearing.

There is a time series for each defect located in 1 of 3 parts of the bearing: ball, inner race, outer race. Telemetry measurements are come from 3 acceleroments installed on 3 positions in the system:

- Drive end (DE)

- Fan end (FE)
- Base (BA)

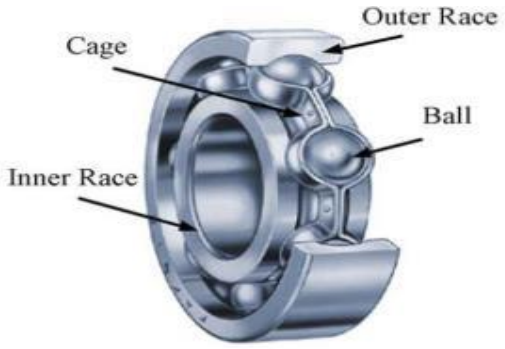

Figure 4.3 Structure of a rolling element bearing

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 4.1.

Fault depth	Fault name	Class Name Used
0.028	0.028	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)

Table 4.1 Classes in CWRU Bearing Dataset

4.1.2 Experiment Data Setup

As shown in Table 4.1, 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.

4.2 Sample data sets.

The available data as provided by CWRU is in .mat files so we will converted these files into digital format.

```
{'X098_DE_time': array([[ 0.046104 ],
```

[-0.03713354],

[-0.089496],

...,

[-0.09909231],

[-0.10827138],

[-0.07092923]]), 'X098_FE_time': array([[0.02321636],

[0.08115455],

[0.09533091],

...,

[-0.00760182],

[0.04026909],

[0.06102]]), 'X099_DE_time': array([[0.06425354],

[0.06300185],

[-0.00438092],

...,

[0.00521538],

[-0.06571385],

[-0.12266585]]), 'X099_FE_time': array([[0.03862545],

[0.09676909],

[0.12738182],

...,

[0.03164],

[0.113],

[0.16292545]]), '__globals__': [], '__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Fri Jan 28 11:19:25 2000', '__version__': '1.0', 'ans': array([[1]], dtype=uint8)}

4.3 Signal processing methods

Signal processing involves convertingor transforming data in a way allows us to see things in it that are not possible via direct observation. Signal processing allows engineers and scientists to analyze, optimize, and signals, including scientific data, audio streams, images, and video.

4.3.1 Cepstrum

Cepstrum analysis 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.

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

……(4.1)

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.

4.3.2 Wavelet Packet Transform

Wavelet decomposition (wavelet transform) 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 4.4, 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.

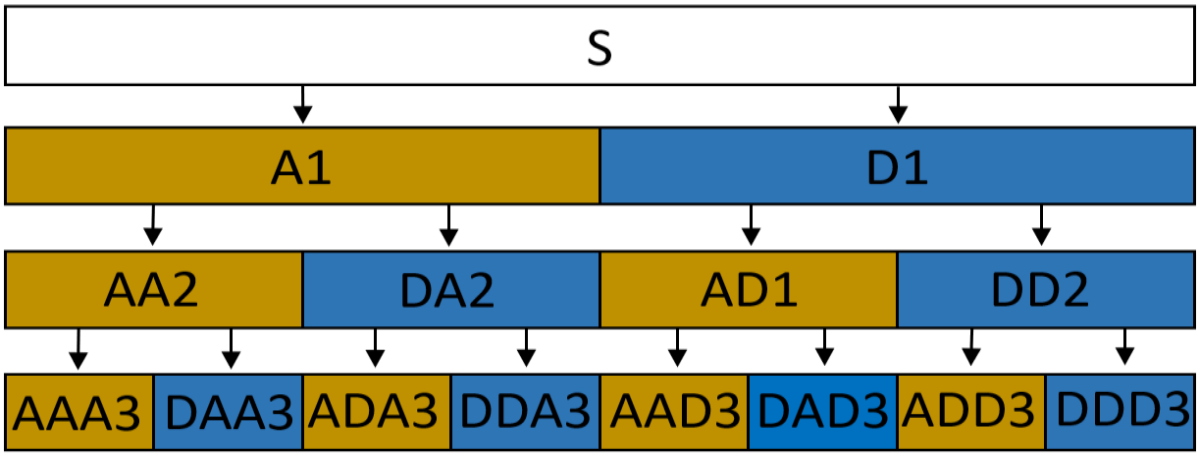

Figure 4.4 WPT framework

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

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

……..(4.2)

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

……..(4.3)

where m is designated as the decomposition level. Equation (4.2) and (4.3) represent the wavelet packet of low-frequency and high-frequency components, respectively. Note that $W_m(n), m = 0,1,2,\ldots\ldots$ and $W_0 = x(n)$ For our study, the wavelet packet transform (WPT) is used to transform each data block. Figure 4.5 displays a dyadic decomposition used.

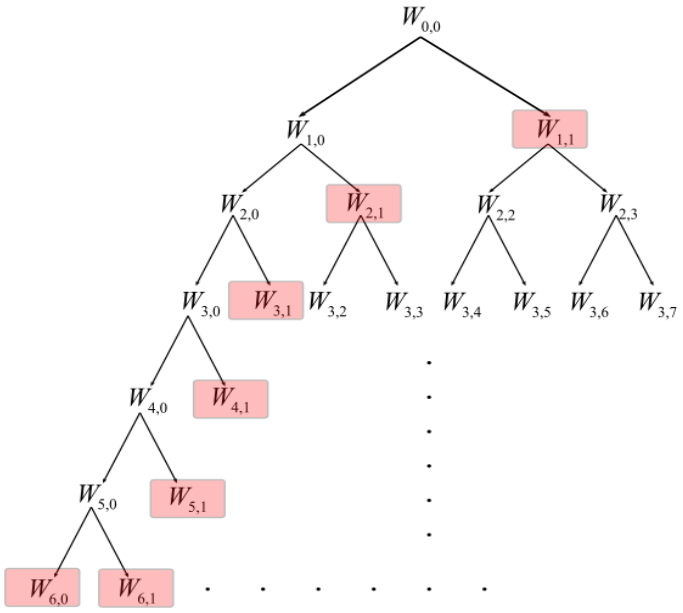

Figure 4.5 Wavelet packet selection

Compared with Figure 4.5, the WPT continues to decompose the lower frequency band and keep the higher frequency band at each level. In our study, Figure 4.5 shows the decomposition at level 6. The red boxes indicate the wavelet packet we need. Table 4.2 further lists the details of a WPT transformed data block with a size of 64 (wavelet coefficients).

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

Table 4.2 Number of Coefficients in Each Wavelet Packet

4.3.3 Empirical Mode Decomposition

The empirical mode decomposition (EMD) is an adaptive signal stabilization processing method proposed by Huang N E 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 . There are two conditions that an IMF must satisfy:

- 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.
- 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 4.5 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.

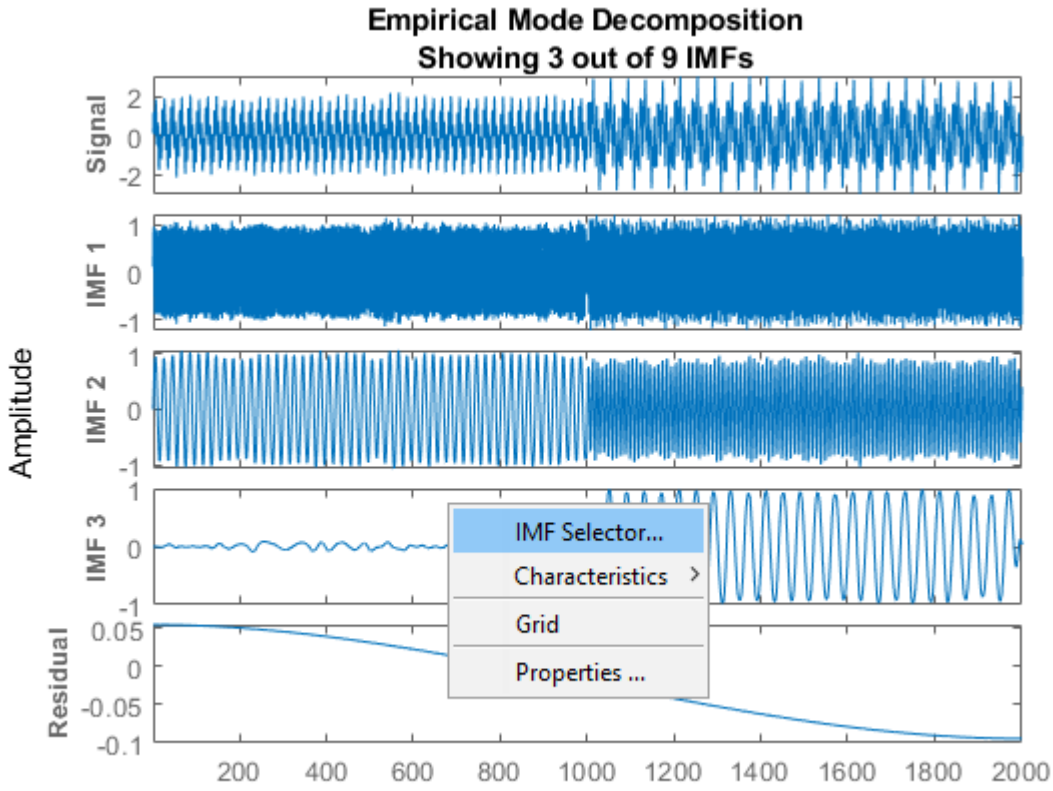

Figure 4.6 Composition of a signal after EMD

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

$$y = (1 - \lambda) * x + \lambda * IMF \quad \text{.....(4.4)}$$

where λ is chosen to be 0.3 in this study.

EMD generates an interactive plot with the original signal, the first 3 IMFs, and the residual. The table generated in the command window indicates the number of sift iterations, the relative tolerance, and the sift stop criterion for each generated IMF. You can hide the table by removing the 'Display' name-value pair or specifying it as 0.

4.3.4 Short-time Fourier Transform

Short-time Fourier transform (STFT) is a sequence of Fourier transforms of a windowed signal. STFT provides the time-localized frequency information for situations in which frequency components of a signal vary over time, whereas the standard Fourier transform provides the frequency information averaged over the entire signal time interval.

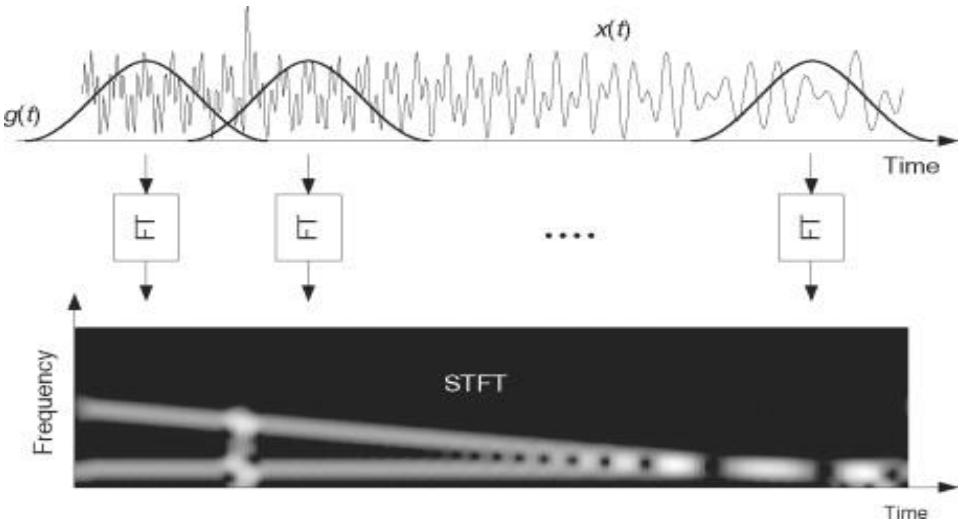

Figure 4.7 Short-time Fourier Transform

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. 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.

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

……(4.5)

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

There exists a trade-off between time and frequency resolution in STFT. In other words, although a narrow-width window results in a better resolution in the time domain, it generates a poor resolution in the frequency domain, and vice versa. Visualization of STFT is often realized via its spectrogram, which is an intensity plot of STFT magnitude over time. Three spectrograms illustrating different time-frequency resolutions.

The short-time Fourier transform (STFT) allows us to perform time-frequency analysis. It is used to generate representations that capture both the local time and frequency content in the signal.

However, the STFT has better temporal and frequency localization properties compared with the Fourier transform. However, since the product of temporal and frequency resolution is constant (because of the classical Heisenberg's uncertainty principle), the generated features cannot achieve instantaneous localization of both time and frequency. In addition, due to using a fixed window length and fixed basis functions, the STFT still cannot capture events with different durations or when the signal contains fast (sharp) events.

4.4 Performance of machine learning technique

Before applying the deep learning network, we tested common machine learning algorithm and compared their performance in terms of classification accuracy.

4.4.1 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 5.2 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.

The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.

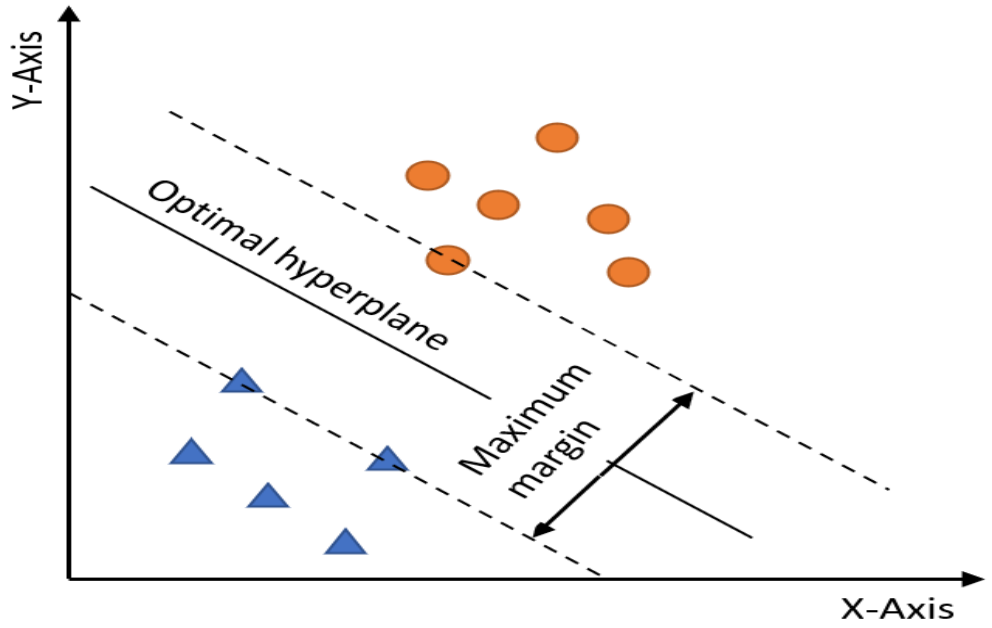

Figure 4.8 Optimal hyperplane

Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning.

4.4.2 Maximal-Margin Classifier

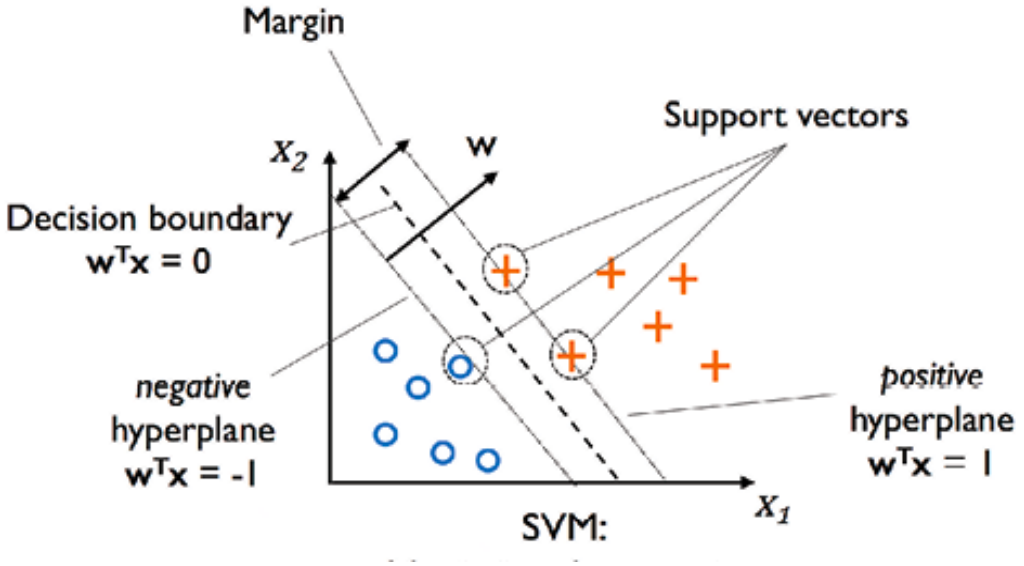

Figure 4.9 Maximal-Margin Classifier

The Maximal-Margin Classifier is a hypothetical classifier that best explains how SVM works in practice. The numeric input variables (x) in your data (the columns) form an n-dimensional space. For example, if you had two input variables, this would form a two-dimensional space. A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two-dimensions you can visualize this as a line and let's assume that all of our input points can be completely separated by this line. For example:

$$B0 + (B1 \times X1) + (B2 \times X2) = 0$$

……..(4.6)

Where the coeficients (B1 and B2) that determine the slope of the line and the intercept (B0) are found by the learning algorithm, and X1 and X2 are the two input variables. You can make classifications using this line. By plugging in input values into the line equation, you can calculate whether a new point is above or below the line.

- Above the line, the equation returns a value greater than 0 and the point belongs to the first class (class 0).
- Below the line, the equation returns a value less than 0 and the point belongs to the second class (class 1).
- A value close to the line returns a value close to zero and the point may be dificult to classify.
- If the magnitude of the value is large, the model may have more confidence in the prediction.

The distance between the line and the closest data points is referred to as the margin. Thebest or optimal line that can separate the two classes is the line that as the largest margin.

This is called the Maximal-Margin hyperplane. The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane. The hyperplane is learned from training data using an optimization procedure that maximizes the margin. Only these points are relevant in defining the line and in the construction of the classifier.

4.4.3 Soft Margin Classifier

In practice, real data is messy and cannot be separated perfectly with a hyperplane. The constraint of maximizing the margin of the line that separates the classes must be relaxed. This is often called the soft margin classifier. This change allows some points in the training data to violate the separating line. An additional set of coeficients are introduced that give the margin wiggle room in each dimension. These coeficients are sometimes called slack variables. This increases the complexity of the model as there are more parameters for the model to fit to the data to provide this complexity.

A tuning parameter is introduced called simply C that defines the magnitude of the wiggle allowed across all dimensions. The C parameters defines the amount of violation of the margin allowed. A C = 0 is no violation and we are back to the inexible Maximal-Margin Classifier described above. The larger the value of C the more violations of the hyperplane are permitted. During the learning of the hyperplane from data, all training instances that lie within the distance of the margin will affect the placement of the hyperplane and are referred to as support vectors. And as C affects the number of instances that are allowed to fall within the margin, C inuences the number of support vectors used by the model.

- The smaller the value of C, the more sensitive the algorithm is to the training data (higher variance and lower bias).
- The larger the value of C, the less sensitive the algorithm is to the training data (lower variance and higher bias).

The constraint of maximizing the margin of the line that separates the classes must be relaxed. This is often called the soft margin classifier. This change allows some points in the training data to violate the separating line. Support Vector Classifier is an extension of the Maximal Margin Classifier. It is less sensitive to individual data. Since it allows certain data to be misclassified, it's also known as the Soft Margin Classifier. it is the distance between the hyperplane and the observations closest to the hyperplane (support vectors). In SVM large margin is considered a good margin. SVMs avoid overfitting by choosing a specific hyperplane among the many that can separate the data in the feature space.

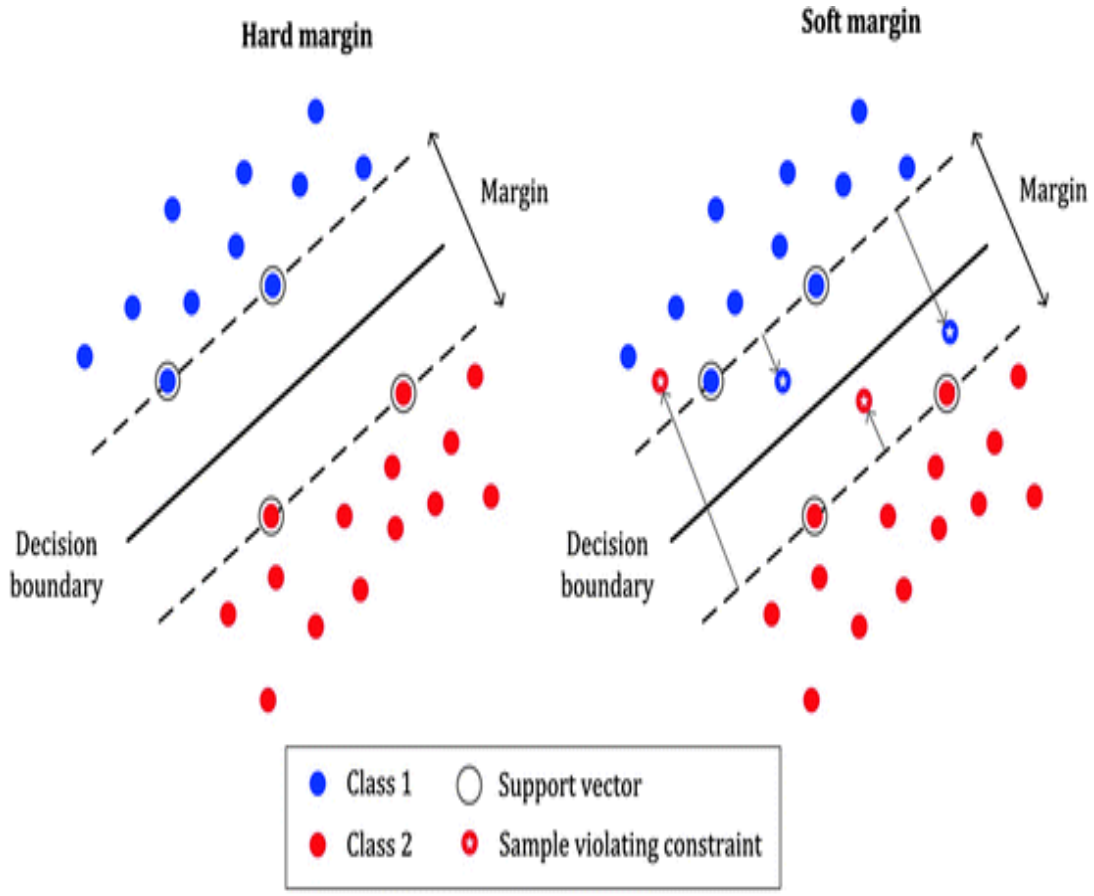

Figure 4.10 Soft Margin and Hard Margin Classifier

4.4.4 Support Vector Machines (Kernels)

The SVM algorithm is implemented in practice using a kernel. The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra, which is out of the scope of this introduction to SVM. A powerful insight is that the linear SVM can be rephrased using the inner product of any two given observations, rather than the observations themselves. The inner product between two vectors is the sum of the multiplication of each pair of input values. For example, the inner product of the vectors [2,3] and [5, 6] is $2 \times 5 + 3 \times 6$ or 28. The equation for making a prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B0 + \sum_{i=1}^{n} (a_i \times (x \times x_i))$$

……(4.7)

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coeficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

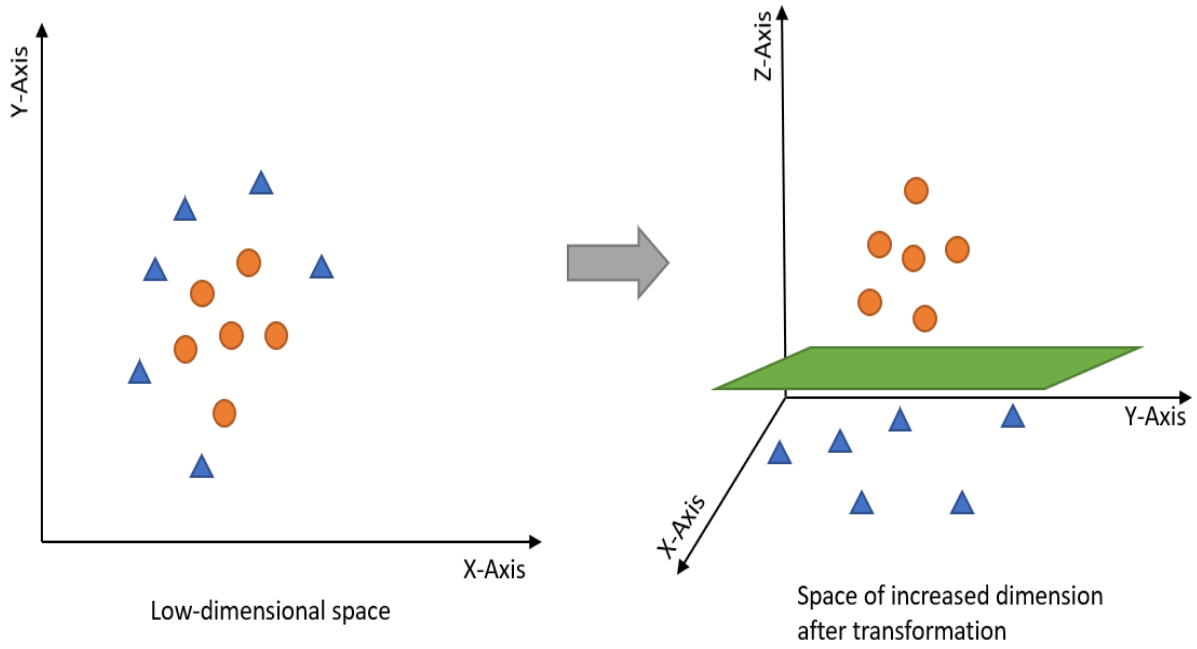

Figure 4.11 Importance of kernel trick

4.4.5 Linear Kernel SVM

The dot-product is called the kernel and can be re-written as:

$$K(x, x_i) = \sum (x \times x_i)$$

……..(4.8)

The kernel defines the similarity or a distance measure between new data and the support vectors. The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs. Other kernels can be used that transform the input space into higher dimensions such as a Polynomial Kernel and a Radial

Kernel. This is called the Kernel Trick. It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. This in turn can lead to more accurate classifiers.

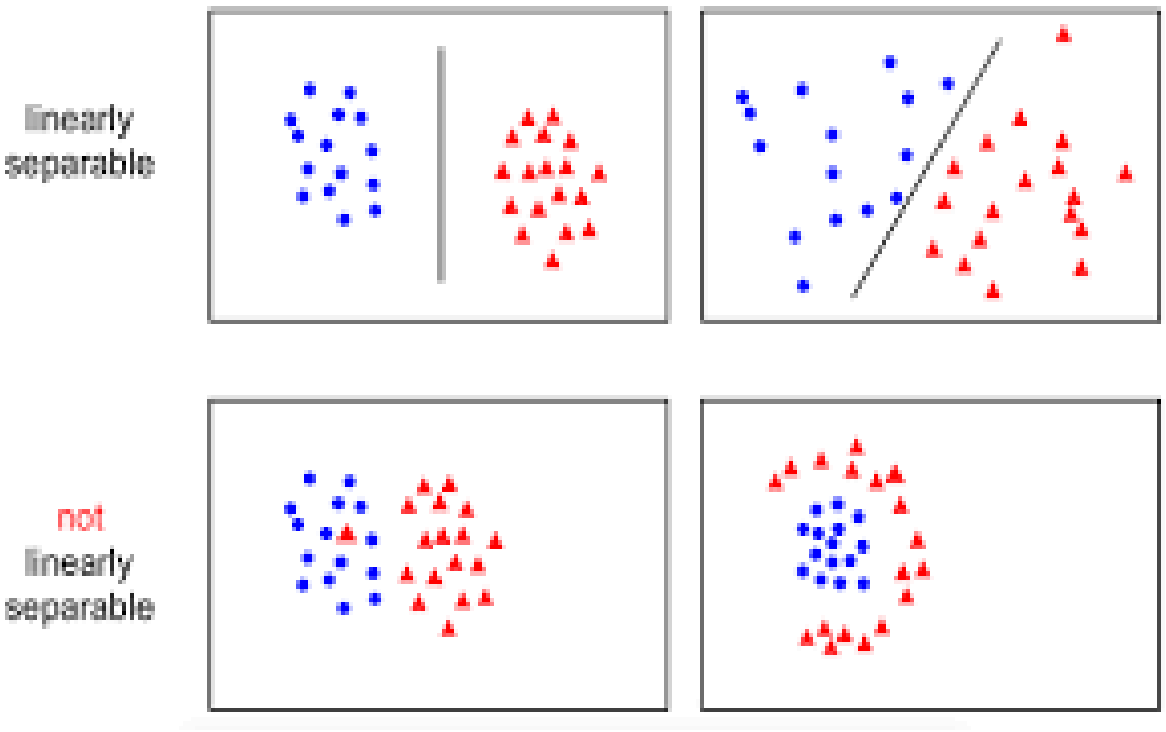

Figure 4.12 Linear Kernel SVM

4.4.6 Polynomial Kernel SVM

Instead of the dot-product, we can use a polynomial kernel, for example:

$$K(x, x_i) = 1 + \sum (x \times x_i)^d$$

……..(4.9)

Where the degree of the polynomial must be speci_ed by hand to the learning algorithm. When d = 1 this is the same as the linear kernel. The polynomial kernel allows for curved lines in the input space.

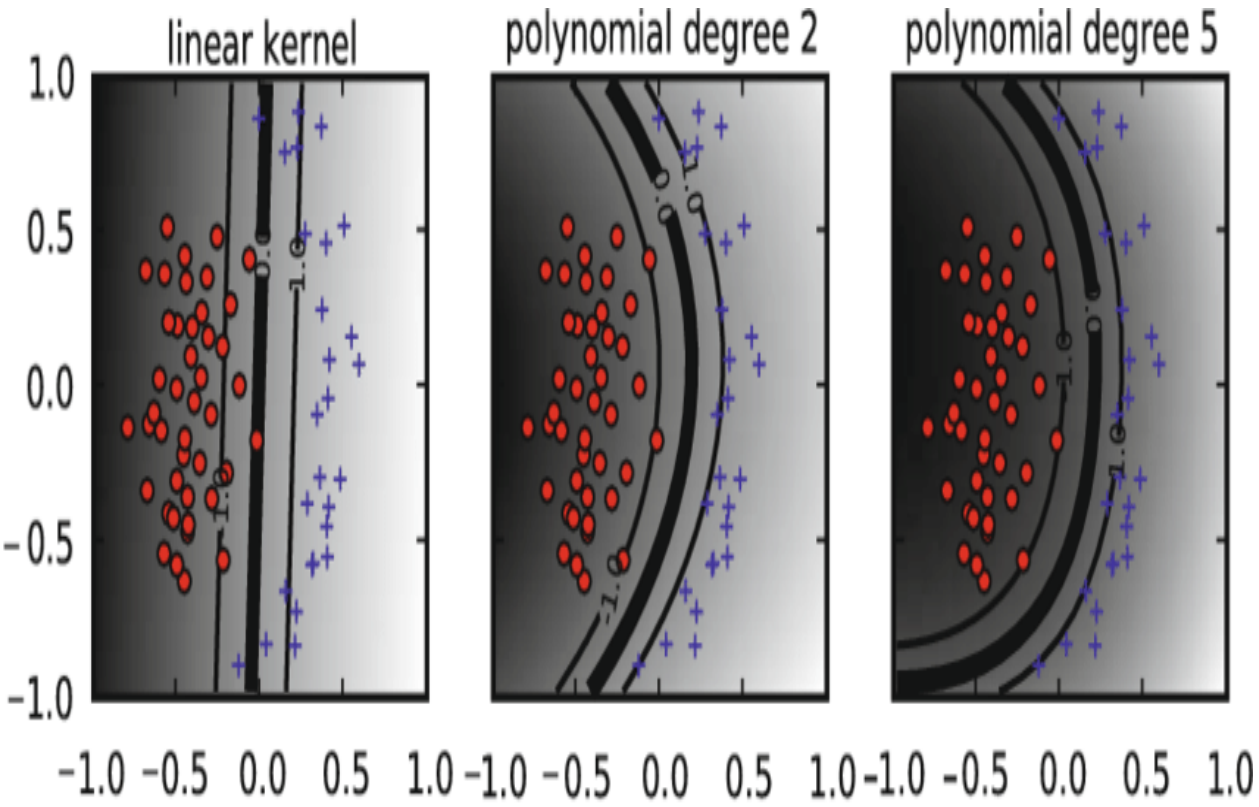

Figure 4.13 Polynomial Kernel SVM

4.4.7 Radial Kernel SVM

Finally, we can also have a more complex radial kernel. For example:

$$K(x, x_i) = e^{-\gamma \times \sum((x - x_i)^2)}$$

…….(4.10)

Where gamma is a parameter that must be speci_ed to the learning algorithm. A good default value for gamma is 0.1, where gamma is often 0 < gamma < 1. The radial kernel is very local and can create complex regions within the feature space, like closed polygons in a two-dimensional space.

In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

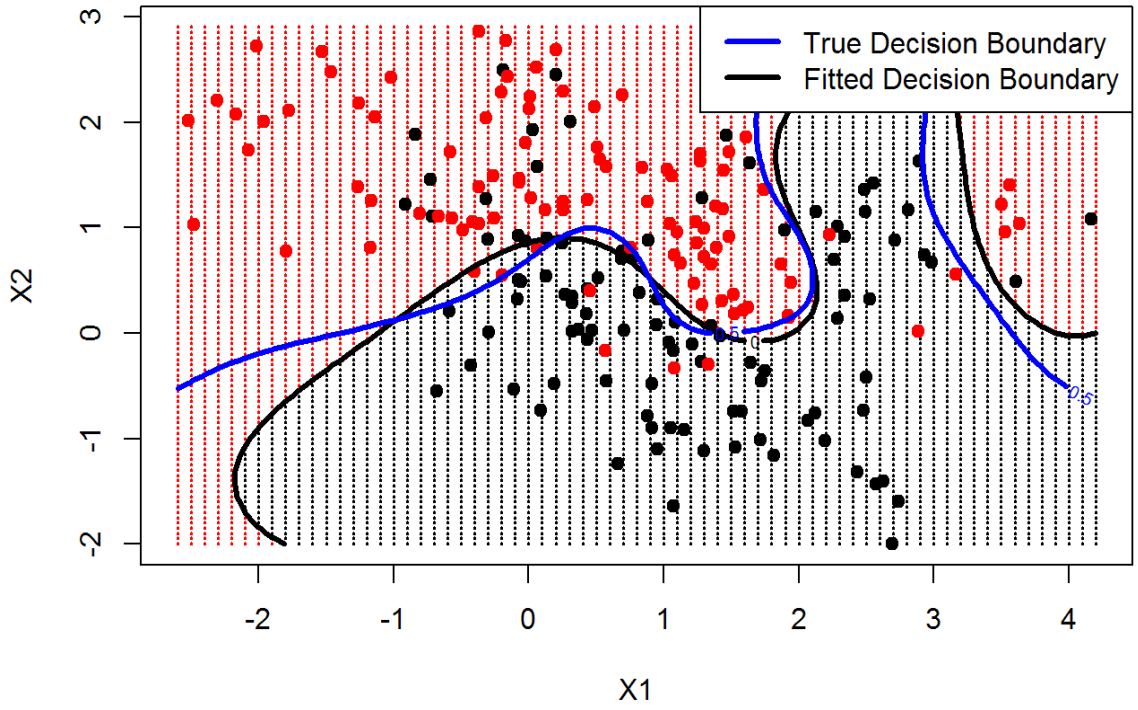

Figure 4.14 Radial Kernel SVM

4.4.8 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.

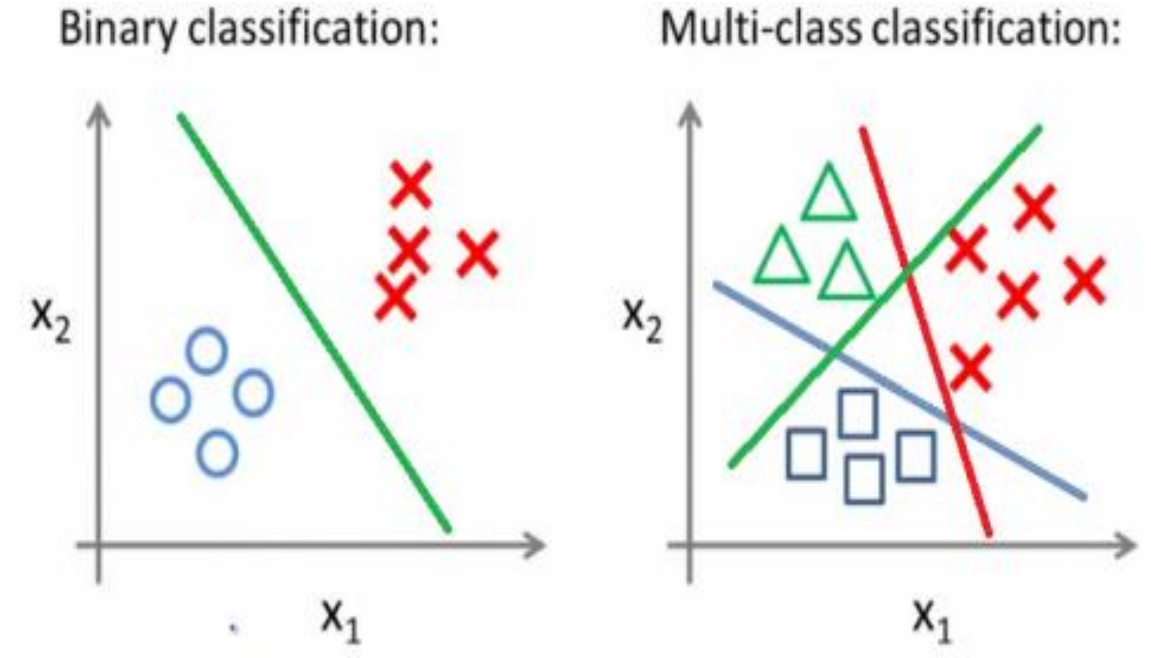

Figure 4.15 Multi-class SVM

4.4.9 How to Learn a SVM Model

The SVM model needs to be solved using an optimization procedure. You can use a numerical optimization procedure to search for the coeficients of the hyperplane. This is ineficient and is not the approach used in widely used SVM implementations like LIBSVM. If implementing the algorithm as an exercise, you could use a variation of gradient descent called sub-gradient descent.

There are specialized optimization procedures that re-formulate the optimization problem to be a Quadratic Programming problem. The most popular method for fitting SVM is the Sequential Minimal Optimization (SMO) method that is very eficient. It breaks the problem down into sub-problems that can be solved analytically (by calculating) rather than numerically (by searching or optimizing).

4.4.10 Preparing Data For SVM

This section lists some suggestions for how to best prepare your training data when learning an SVM model.

- Numerical Inputs: SVM assumes that your inputs are numeric. If you have categorical inputs you may need to covert them to binary dummy variables (one variable for each category).
- Binary Classification: Basic SVM as described in this chapter is intended for binary (two-class) classification problems. Although, extensions have been developed for regression and multiclass classification.

4.5 Deep learning neural networks performance

4.5.1 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 onedimensional 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.

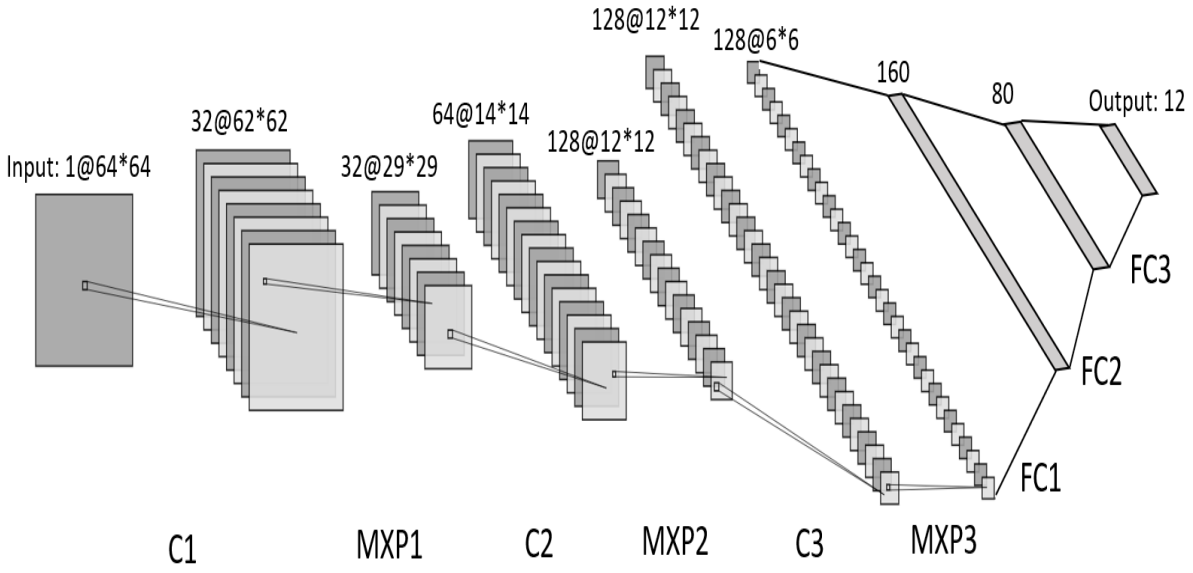

Figure 4.16 Structure of CNN

4.6 Experimantation

4.6.1 Source Code

This is the source code we implemented for bearing fault detection in the Python language .

We used both Machine Learning and Deep Learning Models for faults detection.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import scipy.io
import numpy as np
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
#Importing data
def ImportData():
  X99_normal = scipy.io.loadmat('./Classification_of_bearing_faults_usi
ng_ML-
main/BearingData_CaseWestern/99.mat')['X099_DE_time']
  X111_InnerRace_007 = scipy.io.loadmat('./Classification_of_bearing_fa
ults_using_ML-main/BearingData_CaseWestern/111.mat')['X111_DE_time']
  X124_Ball_007 = scipy.io.loadmat('./Classification_of_bearing_faults_
using_ML-main/BearingData_CaseWestern/124.mat')['X124_DE_time']
  X137_Outer_007 = scipy.io.loadmat('./Classification_of_bearing_faults
_using_ML-main/BearingData_CaseWestern/137.mat')['X137_DE_time']
  X176_InnerRace_014 = scipy.io.loadmat('./Classification_of_bearing_fa
ults_using_ML-main/BearingData_CaseWestern/176.mat')['X176_DE_time']
  X191_Ball_014 = scipy.io.loadmat('./Classification_of_bearing_faults_
using_ML-main/BearingData_CaseWestern/191.mat')['X191_DE_time']
  X203_Outer_014 = scipy.io.loadmat('./Classification_of_bearing_faults
_using_ML-main/BearingData_CaseWestern/203.mat')['X203_DE_time']
  X215_InnerRace_021 = scipy.io.loadmat('./Classification_of_bearing_fa
ults_using_ML-main/BearingData_CaseWestern/215.mat')['X215_DE_time']
  X228_Ball_021 = scipy.io.loadmat('./Classification_of_bearing_faults_
using_ML-main/BearingData_CaseWestern/228.mat')['X228_DE_time']
  X240_Outer_021 = scipy.io.loadmat('./Classification_of_bearing_faults
_using_ML-main/BearingData_CaseWestern/240.mat')['X240_DE_time']
```

```
  return [X99_normal,X111_InnerRace_007,X124_Ball_007,X137_Outer_007,X1
76_InnerRace_014,X191_Ball_014,X203_Outer_014,X215_InnerRace_021,X228_B
all_021,X240_Outer_021]


def Sampling(Data, interval_length, samples_per_block):
  # Calculate the number of blocks that can be sampled based on the int
erval length
  No_of_blocks = (round(len(Data)/interval_length) -
 round(samples_per_block/interval_length)-1)
  SplitData = np.zeros([No_of_blocks, samples_per_block])
  for i in range(No_of_blocks):
    SplitData[i,:] = (Data[i*interval_length:(i*interval_length)+sample
s_per_block]).T
  return SplitData


def DataPreparation(Data, interval_length, samples_per_block):
  for count,i in enumerate(Data):
    SplitData = Sampling(i, interval_length, samples_per_block)
    y = np.zeros([len(SplitData),10])
    y[:,count] = 1
    y1 = np.zeros([len(SplitData),1])
    y1[:,0] = count
    # Stack up and label the data
    if count==0:
      X = SplitData
      LabelPositional = y
      Label = y1
    else:
      X = np.append(X, SplitData, axis=0)
      LabelPositional = np.append(LabelPositional,y,axis=0)
      Label = np.append(Label,y1,axis=0)
  return X, LabelPositional, Label


Data = ImportData()
interval_length = 200
samples_per_block = 1681
```

Y_CNN is of shape (n, 10) representing 10 classes as 10 columns. In each sample, for the c lass to which it belongs,

the corresponding column value is marked 1 and the rest as 0, facilitating Softmax impleme ntation in CNN

Y is of shape (m, 1) where column values are between 0 and 9 representing the classes dire ctly. - 1-hot encoding

```
X, Y_CNN, Y = DataPreparation(Data, interval_length, samples_per_block)
```

```
print('Shape of Input Data =', X.shape)
print('Shape of Label Y_CNN =', Y_CNN.shape)
print('Shape of Label Y =', Y.shape)
```

```
Shape of Input Data = (24276, 1681)
Shape of Label Y_CNN = (24276, 10)
Shape of Label Y = (24276, 1)
```

Export Input Data to MATLAB

- For feature Extraction - time and frequency domain analysis
- For implementing Self Organizing Feature Maps using MATLAB 'nctool' feature

```
XX = {'X':X}
scipy.io.savemat('Data.mat', XX)
```

K-Fold Cross Validation

```
# k-fold cross validation
kSplits = 5
kfold = KFold(n_splits=kSplits, random_state=32, shuffle=True)
```

4.6.2 Neural Network Models

// 1-Dimensional Convolutional Neural Network Classification.

1D Convolutional Neural Networks are similar to well known and more established 2D Convolutional Neural Networks. 1D Convolutional Neural Networks are used mainly used on text and 1D signals

```
# Reshape the data - 1 dimensional feed
Input_1D = X.reshape([-1,1681,1])
```

```
# Test-Train Split
```

```
X_1D_train, X_1D_test, y_1D_train, y_1D_test = train_test_split(Input_1
D, Y_CNN, train_size=0.75,test_size=0.25, random_state=101)

# Define the CNN Classification model
class CNN_1D():
  def __init__(self):
    self.model = self.CreateModel()

  def CreateModel(self):
    model = models.Sequential([
        layers.Conv1D(filters=16, kernel_size=3, strides=2, activation=
'relu'),
        layers.MaxPool1D(pool_size=2),
        layers.Conv1D(filters=32, kernel_size=3, strides=2, activation=
'relu'),
        layers.MaxPool1D(pool_size=2),
        layers.Conv1D(filters=64, kernel_size=3, strides=2, activation=
'relu'),
        layers.MaxPool1D(pool_size=2),
        layers.Conv1D(filters=128, kernel_size=3, strides=2, activation
='relu'),
        layers.MaxPool1D(pool_size=2),
        layers.Flatten(),
        layers.InputLayer(),
        layers.Dense(100,activation='relu'),
        layers.Dense(50,activation='relu'),
        layers.Dense(10),
        layers.Softmax()
        ])
    model.compile(optimizer='adam',
                  loss=tf.keras.losses.CategoricalCrossentropy(),
                  metrics=['accuracy'])
    return model


accuracy_1D = []

# Train the model
for train, test in kfold.split(X_1D_train,y_1D_train):
  Classification_1D = CNN_1D()
  history = Classification_1D.model.fit(X_1D_train[train], y_1D_train[t
rain], verbose=1, epochs=12)

  # Evaluate the accuracy of the model on the training set
```

```
  kf_loss, kf_accuracy = Classification_1D.model.evaluate(X_1D_train[te
st], y_1D_train[test])
  accuracy_1D.append(kf_accuracy)

CNN_1D_train_accuracy = np.average(accuracy_1D)*100
print('CNN 1D train accuracy =', CNN_1D_train_accuracy)


# Evaluate the accuracy of the model on the test set
CNN_1D_test_loss, CNN_1D_test_accuracy = Classification_1D.model.evalua
te(X_1D_test, y_1D_test)
CNN_1D_test_accuracy*=100
print('CNN 1D test accuracy =', CNN_1D_test_accuracy)
```

// Confusion Matrix Calculation.

```
def ConfusionMatrix(Model, X, y):
  y_pred = np.argmax(Model.model.predict(X), axis=1)
  ConfusionMat = confusion_matrix(np.argmax(y, axis=1), y_pred)
  return ConfusionMat
```

// Plot results - CNN 1D

```
plt.figure(1)
plt.title('Confusion Matrix - CNN 1D Train')
sns.heatmap(ConfusionMatrix(Classification_1D, X_1D_train, y_1D_train)
, annot=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()

plt.figure(2)
plt.title('Confusion Matrix - CNN 1D Test')
sns.heatmap(ConfusionMatrix(Classification_1D, X_1D_test, y_1D_test) ,
annot=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()

plt.figure(3)
plt.title('Train - Accuracy - CNN 1D')
plt.bar(np.arange(1,kSplits+1),[i*100 for i in accuracy_1D])
plt.ylabel('accuracy')
plt.xlabel('folds')
plt.ylim([70,100])
plt.show()

plt.figure(4)
plt.title('Train vs Test Accuracy - CNN 1D')
plt.bar([1,2],[CNN_1D_train_accuracy,CNN_1D_test_accuracy])
plt.ylabel('accuracy')
```

```
plt.xlabel('folds')
plt.xticks([1,2],['Train', 'Test'])
plt.ylim([70,100])
plt.show()
```

// 2-Dimensional Convolutional Neural Network Classification.

It is called 2 dimensional CNN because the kernel slides along 2 dimensions on the data In 2D CNN, kernel moves in 2 directions. Input and output data of 2D CNN The convolutional layer is composed of multiple convolutional neurons.

```
# Reshape the data - 2 dimensional feed
Input_2D = X.reshape([-1,41,41,1])

# Test-Train Split
X_2D_train, X_2D_test, y_2D_train, y_2D_test = train_test_split(Input_2
D, Y_CNN, train_size=0.75,test_size=0.25, random_state=101)

# Define the CNN Classification model
class CNN_2D():
  def __init__(self):
    self.model = self.CreateModel()

  def CreateModel(self):
    model = models.Sequential([
        layers.Conv2D(filters=16, kernel_size=(3,3), strides=(2,2), pad
ding ='same',activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=32, kernel_size=(3,3),strides=(2,2), padd
ing ='same',activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=64, kernel_size=(3,3),strides=(2,2),paddi
ng ='same', activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=128, kernel_size=(3,3),strides=(2,2),padd
ing ='same', activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Flatten(),
        layers.InputLayer(),
        layers.Dense(100,activation='relu'),
        layers.Dense(50,activation='relu'),
        layers.Dense(10),
        layers.Softmax()
        ])
```

```
    model.compile(optimizer='adam',
                  loss=tf.keras.losses.CategoricalCrossentropy(),
                  metrics=['accuracy'])
    return model


accuracy_2D = []

# Train the model
for train, test in kfold.split(X_2D_train,y_2D_train):
  Classification_2D = CNN_2D()
  history = Classification_2D.model.fit(X_2D_train[train], y_2D_train[t
rain], verbose=1, epochs=12)

  # Evaluate the accuracy of the model on the training set
  kf_loss, kf_accuracy = Classification_2D.model.evaluate(X_2D_train[te
st], y_2D_train[test])
  accuracy_2D.append(kf_accuracy)

CNN_2D_train_accuracy = np.average(accuracy_2D)*100
print('CNN 2D train accuracy =', CNN_2D_train_accuracy)


# Evaluate the accuracy of the model on the test set
CNN_2D_test_loss, CNN_2D_test_accuracy = Classification_2D.model.evalua
te(X_2D_test, y_2D_test)
CNN_2D_test_accuracy*=100
print('CNN 2D test accuracy =', CNN_2D_test_accuracy)
```

// Plot results - CNN 2D

```
plt.figure(5)
plt.title('Confusion Matrix - CNN 2D Train')
sns.heatmap(ConfusionMatrix(Classification_2D, X_2D_train, y_2D_train)
, annot=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()

plt.figure(6)
plt.title('Confusion Matrix - CNN 2D Test')
sns.heatmap(ConfusionMatrix(Classification_2D, X_2D_test, y_2D_test) ,
annot=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()

plt.figure(7)
plt.title('Train - Accuracy - CNN 2D')
```

```
plt.bar(np.arange(1,kSplits+1),[i*100 for i in accuracy_2D])
plt.ylabel('accuracy')
plt.xlabel('folds')
plt.ylim([70,100])
plt.show()

plt.figure(8)
plt.title('Train vs Test Accuracy - CNN 2D')
plt.bar([1,2],[CNN_2D_train_accuracy,CNN_2D_test_accuracy])
plt.ylabel('accuracy')
plt.xlabel('folds')
plt.xticks([1,2],['Train', 'Test'])
plt.ylim([70,100])
plt.show()
```

4.6.3 Support Vector Machine Model

Import Feature Data

- The input data has been pre processed in MATLAB to extract the features like max, min, peak-to-peak, mean, variance, standard deviation, root-mean-square, skewness, crest factor, kurtosis in time domain, and amplitude and value of the maximum frequency in the frequency domains.

```
X_Features = scipy.io.loadmat('./Classification_of_bearing_faults_using_ML-main/BearingData_CaseWestern/X_Features.mat')['Feature_Data']
# Feature data shape (no. of samples, no. of features)
X_Features.shape
```

Import packages and dependencies

```
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from tqdm import tqdm_notebook as tqdm
import warnings
warnings.filterwarnings('ignore')
```

Normalize the Input Data

```
X_Norm = StandardScaler().fit_transform(X_Features)
```

Dimensionality Reduction - Principal Component Analysis.

```
pca = PCA(n_components=5)
Input_SVM_np = pca.fit_transform(X_Norm)
Input_SVM = pd.DataFrame(data = Input_SVM_np)
Label_SVM = pd.DataFrame(Y, columns = ['target'])
```

Define Parameters

```
parameters = {'kernel':('rbf','poly','sigmoid'),
              'C': [0.01, 1],
              'gamma' : [0.01, 1],
              'decision_function_shape' : ['ovo']}

# Support vector Machine
svm = SVC()
```

Train the Model

```
# Test-Train Split
X_train_SVM, X_test_SVM, y_train_SVM, y_test_SVM = train_test_split(Inp
ut_SVM_np, Y, train_size=0.75,test_size=0.25, random_state=101)

# Train the model to obtain the best parameters
svm_cv = GridSearchCV(svm, parameters, cv=5)
svm_cv.fit(X_train_SVM, y_train_SVM)

print("Best parameters = ",svm_cv.best_params_)

SVM_train_accuracy = svm_cv.best_score_*100
print('SVM train accuracy =', SVM_train_accuracy)

# Evaluate the accuracy of the model on the test set
SVM_test_accuracy = svm_cv.score(X_test_SVM, y_test_SVM)
SVM_test_accuracy*=100
print('SVM test accuracy =', SVM_test_accuracy)
def ConfusionMatrix_SVM(Model, X, y):
  y_pred = Model.predict(X)
  ConfusionMat = confusion_matrix(y, y_pred)
  return ConfusionMat

print(svm_cv.score(X_train_SVM, y_train_SVM))

plt.figure(13)
plt.title('Confusion Matrix - SVM Train')
sns.heatmap(ConfusionMatrix_SVM(svm_cv, X_train_SVM, y_train_SVM) , ann
ot=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()
```

```
plt.figure(14)
plt.title('Confusion Matrix - SVM Test')
sns.heatmap(ConfusionMatrix_SVM(svm_cv, X_test_SVM, y_test_SVM) , annot
=True, fmt='d',annot_kws={"fontsize":8},cmap="YlGnBu")
plt.show()

plt.figure(16)
plt.title('Train vs Test Accuracy - SVM')
plt.bar([1,2],[SVM_train_accuracy,SVM_test_accuracy])
plt.ylabel('accuracy')
plt.xlabel('folds')
plt.xticks([1,2],['Train', 'Test'])
plt.ylim([70,100])
plt.show()
```

Plot Decision boundaries in 2D

```
from mlxtend.plotting import plot_decision_regions
value = 0
width = 1
plt.figure(17)
plt.figure(figsize=(10.4,8.8))
plt.title('Decision Boundary - SVM')
plot_decision_regions(X_test_SVM, (y_test_SVM.astype(np.integer)).flatt
en(), clf=svm_cv, legend=2,
                  feature_index=[0,1],
                  filler_feature_values={2:value, 3:value, 4:value},
                  filler_feature_ranges={2:width, 3:width, 4:width},)
plt.show()
```

4.6.4 Comparison Of Models

In this we are going to compare the accuracy of all the three models that we used previously that is 1D CNN, 2D CNN and SVM.

```
plt.figure(18)
plt.title('Accuracy in Training data')
plt.bar([1,2,3],[CNN_1D_train_accuracy, CNN_2D_train_accuracy, SVM_trai
n_accuracy])
plt.ylabel('accuracy')
```

```
plt.xlabel('folds')
plt.xticks([1,2,3],['CNN-1D', 'CNN-2D' ,  'SVM'])
plt.ylim([70,100])
plt.show()


plt.figure(19)
plt.title('Accuracy in Test data')
plt.bar([1,2,3],[CNN_1D_test_accuracy, CNN_2D_test_accuracy, SVM_test_a
ccuracy])
plt.ylabel('accuracy')
plt.xlabel('folds')
plt.xticks([1,2,3],['CNN-1D', 'CNN-2D' , 'SVM'])
plt.ylim([70,100])
plt.show()
```

CHAPTER 5

RESULTS AND DISCUSSION

The results and accuracy that obtained as a output after implementing the code. Results are obtained in the form of statistical graph and confusion matrix to evaluate the output.

5.1 Results and accuracy of 1D CNN

The average accuracy achieved for Train and Test by 1D CNN is 98.769% and 98.912% Respectively.

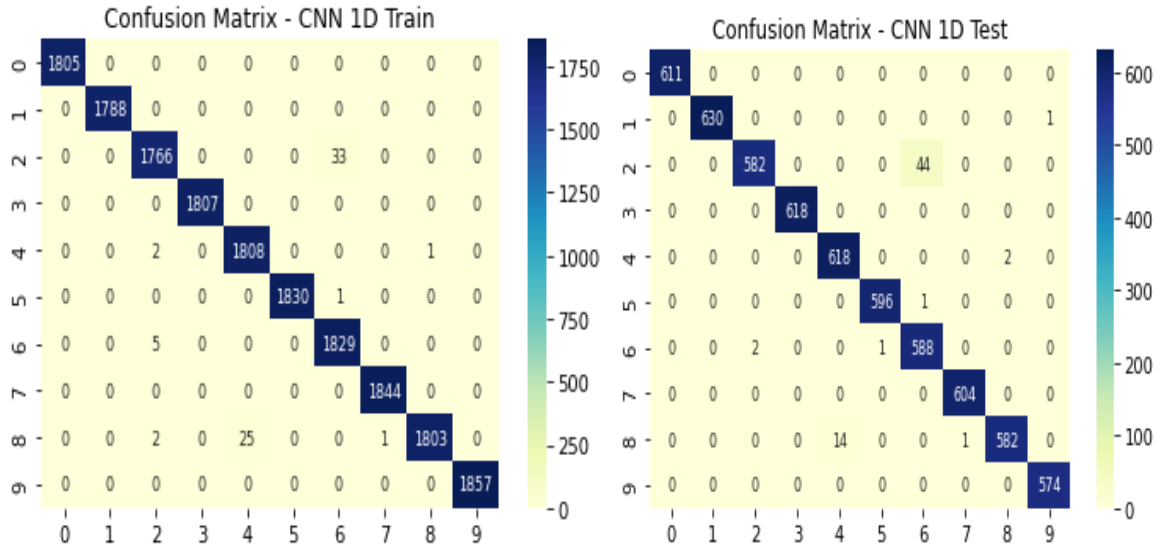

Figure 5.1 1D CNN Train and Test Result

The statistical graph is shown in the Figure 5.2 this is the comparision between the Train and Train vs Test accuracy for 1D CNN .

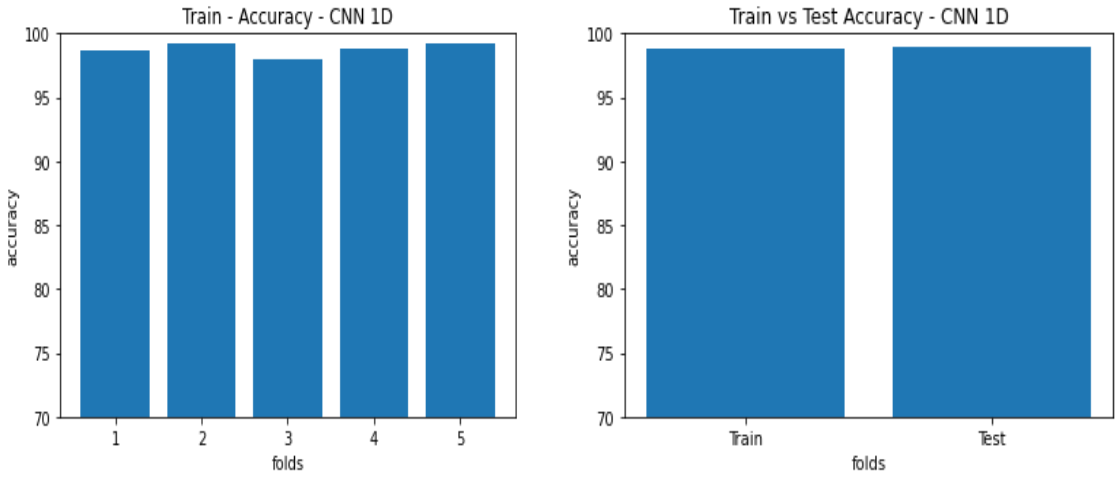

Figure 5.2 1D CNN Train and Train vs Test Accuracy

5.2 Results and accuracy of 2D CNN

The average accuracy achieved for Train and Test by 2D CNN is 95.48% and 95.32% Respectively.

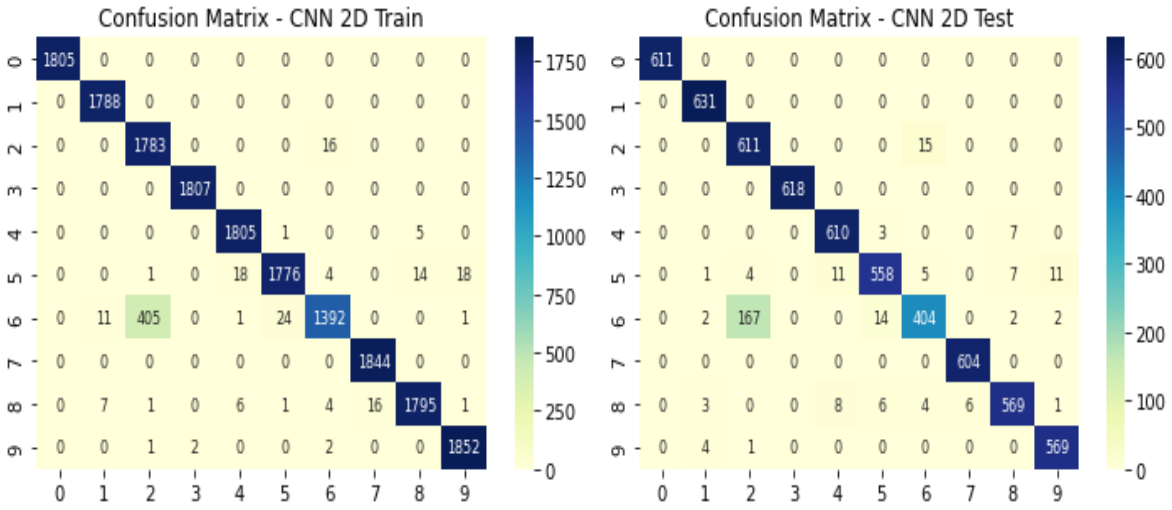

Figure 5.3 2D CNN Train and Test Result

The statistical graph is shown in the Figure 5.4 this is the comparision between the Train and Train vs Test accuracy for 2D CNN .

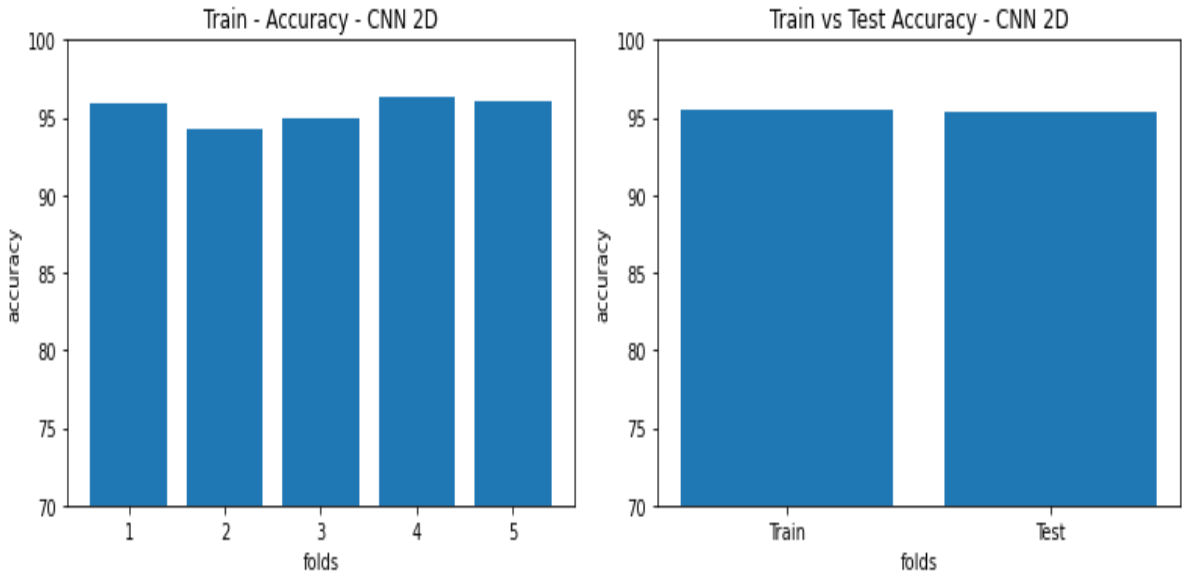

Figure 5.4 2D CNN Train and Train vs Test Accuracy

5.3 Results and accuracy of SVM

The average accuracy achieved for Train and Test by SVM is 92.81% and 92.55% Respectively.

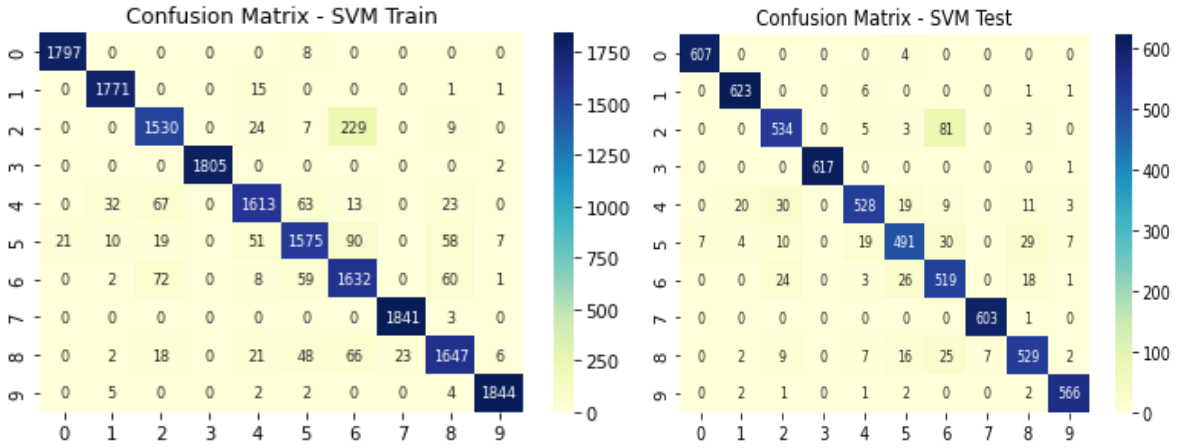

Figure 5.5 SVM Train and Test Result

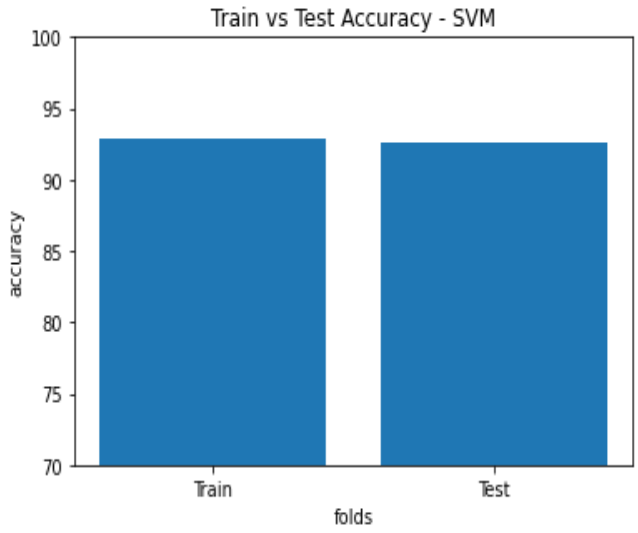

Figure 5.6 Train vs Test Accuracy in SVM

The statistical graph is shown in the Figure 5.6 this is the comparision between the Train vs Test accuracy for SVM.

5.4 Comparision between Models

Figure 5.7 Comparision between Models

The statistical graph is shown in the Figure 5.7 is the comparision of all three models. After Evaluating Results the Train Accuracy of 1D CNN is 98.769% and Test Accuracy of 1D CNN is 98.912%.Train Accuracy of 2D CNN is 95.48% and Test Accuracy of 2D CNN is 95.32%. Train Accuracy of SVM is 92.81% and Test Accuracy of SVM is 92.55% After Comparing all models 1D CNN gives good Accuracy of 98% in both Train and Test.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

After implementing the obtained results of the Machine Learning Algorithm and Deep Learning Algorithm in that Deep Learning Models works more accurately than the Machine Learning (SVM). In Deep Learning 1-Dimentional CNN works more Accurately than the 2-dimentional.

It is demonstrated that, despite the fact that deep learning algorithms require a large dataset to train, they can automatically perform adaptive feature extractions on the bearing data without any prior expertise on fault characteristic frequencies or operating conditions, making them promising candidates to perform realtime bearing fault diagnostics. A comparative study is also conducted comparing the performance of many DL algorithm variants using the common CWRU bearing dataset. Finally, detailed recommendations and suggestions are provided in regards to choosing the most appropriate type of DL algorithm for specific application scenarios. Future research directions are also discussed to better facilitate the transition of DL algorithms from laboratory tests to real-world applications. We evaluated the proposed method experimentally on a public available dataset. As it was shown in the experiments part the proposed method improved the reachable maximum accuracies.

6.2 Future Scope

In future work, researchers should combine theory and practice to increase the size of the available data set as the data basis for future research, improve the accuracy and speed of the diagnosis algorithm as much as possible, and combine this with the equipment in actual working conditions to improve the diagnosis and the usability of the algorithm in the industrial field

To alleviate the problem of “limited labels”, semi-supervised learning can be utilized to make full use of the limited labeled data and the massive unlabeled data. One potential

routine is to employ the variational encoder based deep generative model perform variational inference on data with limited labels.

Sensor fusion: To solve the potential problem of“noisy data”, it might be worthwhile to deploy other types of sensors, such as the load cell, the current sensor, and the acoustic emission sensor, etc., and apply sensor fusion techniques to synthesize these data and improve the robustness of bearing fault diagnosis.

REFERENCES

[1] "Bearing fault detection using deep neural network and weighted ensemble learning for multiple motor phase current sources" by **Tobia's Wagner and Sara Sommer.**

[2] **Peter W. Tse,Y. H. Peng , Richard Yam** "Wavelet Analysis and Envelope Detection For Rolling Element Bearing Fault Diagnosis."

[3] **K. Baiche, M. Zelmat, A. Lachouri** " Application of Multi-Scale PCA and Energy Spectrum to Bearing Fault Analysis and Detection in Rotating Machinery."

[4] **M. Bilginer Gu¨lmezog˘lu*,y and Semih Ergin** " experiment on An approach for bearing fault detection in electrical motors."

[5] "Envelope order tracking for fault detection in rolling element bearings." By **Yu Guo , Ting-WeiLiu , JingNa , Rong-FongFung.**

[6] **Jafar Zarei , Mohammad Amin Tajeddini , Hamid Reza Karimi** "Vibration analysis for bearing fault detection and classification using an intelligent filter a new algorithm for fault detection and classification."

[7] **J P dron L rasolofondraibe and C couet, A. pavan** "Fault detection and monitoring of a ball bearing benchtest and a production machine via autoregressive spectrum analysis."

[8] **S. A. McInerny and Y. Dai "**Basic Vibration Signal Processing for Bearing Fault Detection."

[9] **A. Rezig , A. N'diye, A. Djerdir and M.R. Mekideche** "Experimental investigation of vibration monitoring technique for online detection of bearing fault in induction motors."

[10] **Chao Yang, Wentao Mao*, Yamin Liu, Siyu Tian** "Incremental Weighted Support Vector Data Description Method for Incipient Fault Detection of Rolling Bearing"

[11] **Aleksandra Ziaja, Ifigeneia Antoniadou, Tomasz Barszcz, Wieslaw J Staszewski and Keith Worden** "Fault detection in rolling element bearings using wavelet-based variance analysis and novelty detection."

[12] **Shen zhang , (Student Member, IEEE), Shibo Zhang , (Student Member, IEEE), Bingnan Wang ,(Senior Member, IEEE), and Thomas G. Habetler** "Deep Learning Algorithms for Bearing Fault Diagnostics."

[13] **Wentao Sui and Dan Zhang** " Research on Envelope Analysis For Bearings Fault Detection."

[14] **Jin Woo Oh, Dogun Park and Jongpil Jeong** " Fault Detection for Lubricant Bearing with CNN."

[15] **Anand S. Reddy*,Praveen Kumar Agarwal,Satish Chand** " Application of artificial neural networks for the fault detection and diagnosis of active magnetic bearings."

[16] **Sanjay Kumar, Deepam Goyal, Rajeev K. Dang*,Sukhdeep S. Dhami, B.S. Pabla** "Condition based maintenance of bearings and gears for fault detection the condition based maintenance of gears and bearings for fault detection has been introduced."

[17] **Michael J. Devaney and Levent Eren** "Monitoring an induction motor's current and detecting bearing failure."

[18] **Jong-Hyo Ahn, Dae-Ho Kwak and Bong-Hwan Koh *** "Fault Detection of a Roller-Bearing System through the EMD of a Wavelet Denoised Signal."

[19] **N. HarishChandra,A.S.Sekhar** "Fault detection in rotor bearing systems using time frequency techniques."