Exploring Deep Learning Techniques for Improved Bearing Fault Detection: A Comparative Study with Traditional Machine Learning Algorithms

A Thesis

Submitted by

DEVENDRA SINGH

in partial fulfilment of the requirements for the award of the degree of

Master of Technology
in
Mechanical Engineering
(Industrial Engineering & Management)

Under the guidance of
Dr. R. Sridharan

Department of Mechanical Engineering
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT CAMPUS PO, CALICUT
KERALA, INDIA 673601

July 2023

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to all those who have contributed to the successful completion of my M.Tech research. Their guidance, support, and encouragement have been invaluable throughout this journey. First and foremost, I would like to extend my heartfelt gratitude to my thesis guide, Dr. R. Sridharan. His expertise, patience, and unwavering support were instrumental in shaping the direction of my research. His insightful feedback, continuous encouragement, and valuable suggestions have been instrumental in the successful completion of this thesis. I would also like to extend my gratitude to Dr. Devendra Kumar Yadav, Dr. Layla Das, and Dr. Anup Aprem for their invaluable guidance in the field of artificial intelligence. Their expertise and mentorship have provided me with the necessary knowledge and tools to explore and implement advanced techniques in my research. I am grateful to Dr. Vinay V. Panikar and Dr. Ratna Kumar K for their support and guidance in the domain of statistics. I would like to express my sincere appreciation to my faculty advisor, Dr. Varaprasad, and Dr. T. Radha Ramanan for their continuous support and valuable inputs throughout my M.Tech program. Their guidance and encouragement have been crucial in shaping my academic journey.

Devendra Singh

(Department of Mechanical engineering)

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: **Signature:**

Date: **Name:**

Reg. No.:

CERTIFICATE

This is to certify that the thesis entitled: **EXPLORING DEEP LEARNING TECHNIQUES FOR IMPROVED BEARING FAULT DETECTION: A COMPARATIVE STUDY WITH TRADITIONAL MACHINE LEARNING ALGORITHMS** submitted by **DEVENDRA SINGH** (Reg. No.: **M210597ME**) to the National Institute of Technology Calicut towards partial fulfilment of the requirements for the award of the Degree of **Master of Technology** in **Mechanical Engineering (Industrial Engineering and Management)** is a bona fide record of the work carried out by him under my supervision and guidance.

Dr. R. Sridharan
(Guide)
Professor
Dept. of Mechanical Engineering

Professor & Head
Dept. of Mechanical Engineering

Place : NIT Calicut
Date : July 2023

ABSTRACT

Bearing fault detection plays a crucial role in condition monitoring to ensure the reliable operation of machinery. This study presents a comprehensive investigation into the effectiveness of deep learning approaches for bearing fault detection, comparing them with traditional machine learning algorithms. The Case Western Reserve University dataset, widely used as a benchmark in the field, is utilized for the analysis. To extract relevant features for the machine learning models, various feature extraction techniques are explored. The study examines the time domain technique, which involves calculating statistical parameters such as mean, standard deviation, and kurtosis. Additionally, the time-frequency technique leverages wavelet packet energy and wavelet packet entropy features to capture the frequency characteristics of the bearing fault signals. Moreover, visualization techniques like Principal Component Analysis (PCA) and t-SNE are employed to gain deeper insights into the data and assist in algorithm selection by identifying patterns and potential separability in the feature space. The performance analysis of traditional machine learning algorithms, including multiclass logistic regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and decision trees, reveals suboptimal results. These findings underscore the influence of data quality and feature extraction techniques on the effectiveness of these algorithms. To overcome the limitations of traditional approaches and improve performance, deep learning algorithms are directly applied to raw data. The study explores the use of Artificial Neural Networks (ANN), 1D Convolutional Neural Networks (CNN1D), 2D Convolutional Neural Networks (CNN2D), and Long Short-Term Memory (LSTM) models. These deep learning models possess the ability to automatically learn and extract relevant features from raw data, potentially capturing intricate patterns and correlations crucial for accurate bearing fault detection. Notably, the study extensively evaluates the CNN-LSTM model, which combines the power of both convolutional and recurrent neural networks, to assess its suitability for bearing fault classification. In conclusion, this study highlights the significance of employing deep learning approaches to enhance bearing fault detection. It emphasizes the

limitations of traditional machine learning algorithms and underscores the critical role played by data quality and feature extraction techniques. By leveraging deep learning models, particularly the CNN-LSTM model, substantial improvements in accuracy and effectiveness of bearing fault detection can be achieved. These findings contribute to the advancement of fault detection methodologies and provide valuable insights for selecting appropriate algorithms in bearing fault analysis.

Keywords: Bearing fault detection, Deep learning approaches, Machine learning algorithms, Feature extraction techniques, Principal Component Analysis (PCA), t-SNE, Artificial Neural Networks (ANN), 1D Convolutional Neural Networks (CNN1D), 2D Convolutional Neural Networks (CNN2D), Long Short-Term Memory (LSTM), CNN-LSTM model.

CONTENTS

LIST OF ABBREVIATIONS

ANN	Artificial neural networks
CNN	Convolutional neural network
CWRU	Case Western Reserve University
DT	Decision Tree
KNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LSTM	long-short term memory
OR	outer raceway
PCA	Principal component analysis
QDA	Quadratic Discriminant Analysis
RF	Random Forest
RNN	Recurrent neural network
STFT	Short-time Fourier transform
SVM	Support vector machine
WPT	Wavelet packet transform

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 PREAMBLE

Roller element bearings hold immense importance as critical components in various rotating machinery[1]. They perform a vital function by ensuring smooth and effective operation. Nevertheless, the emergence of faults in roller element bearings can have dire consequences, including unexpected failures, substantial financial losses, heightened maintenance expenses, and safety hazards. Early detection and accurate diagnosis of faults in machinery is crucial for avoiding severe failures and reducing operational downtime[2]. In recent times, the field of bearing fault detection has seen significant advancements with the introduction of machine learning and deep learning algorithms, showcasing immense promise. These advanced computational techniques offer the possibility of significantly improving the precision and efficiency of fault diagnosis, thereby enabling proactive maintenance strategies and enhancing operational dependability

The field of bearing fault detection has made extensive use of various machine learning algorithms, including Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, and Random Forests [3]. These algorithms are capable of leveraging labeled data to acquire knowledge, enabling them to detect and categorize different fault conditions by discerning patterns and correlations[4]. By analyzing vibration, acoustic, or other sensor signals produced by the bearing, these algorithms can extract pertinent features and effectively detect various types of faults, including surface defects, spalling, wear, and misalignment [5]. In addition, deep learning algorithms, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Artificial Neural Networks (ANNs), have demonstrated remarkable effectiveness across various domains[6].

These algorithms excel in automatically extracting intricate patterns from raw data, eliminating the requirement for manual feature engineering[6]. By leveraging deep learning techniques, fault detection systems can capture subtle fault signatures, effectively handle complex fault patterns, and adapt to diverse operating conditions[7].

The application of deep learning and machine learning algorithms in roller element bearing fault detection brings forth numerous benefits. Firstly, these algorithms possess the capability to effectively handle vast volumes of data, enabling the analysis of extensive vibration signals over extended periods and the detection of transient fault occurrences[8]. Secondly, they demonstrate strong generalization capabilities, ensuring accurate fault detection and classification across bearings of diverse sizes, load capacities, and operating conditions. Lastly, these algorithms can seamlessly integrate into real-time monitoring systems, facilitating continuous, automated, and timely fault detection and diagnosis[9].

The main aim of this research is to conduct a comprehensive investigation and evaluation of the effectiveness of deep learning and machine learning algorithms in the detection of faults in roller element bearings[10]. By leveraging datasets like the Case Western Reserve University (CWRU) dataset, which encompasses comprehensive vibration frequency data from bearings subjected to various fault conditions, this study aims to develop and compare multiple algorithms to determine the most suitable approach for achieving accurate and resilient fault detection[11]. By enhancing the capabilities of bearing fault detection, this research significantly contributes to improving the reliability, safety, and efficiency of rotating machinery. Its findings offer valuable insights for industries and researchers interested in implementing advanced condition monitoring systems and proactive maintenance strategies, ultimately resulting in reduced downtime, heightened productivity, and cost savings.

1.2 MOTIVATION FOR THE RESEARCH

The research on applying machine learning and deep learning algorithms to detect faults in roller element bearings is driven by the desire to enhance the effectiveness and efficiency of traditional fault detection methods. Manual inspection and regular

maintenance practices, which have been widely used in the past, have limitations in terms of accuracy, scalability, and cost-effectiveness[12]. As a result, there is a growing need for automated and intelligent approaches that can accurately and preemptively identify bearing faults.

The utilization of machine learning and deep learning algorithms offers a promising solution to overcome the limitations of conventional methods and drive progress in bearing fault detection[13]. There are several key factors that motivate the adoption of these algorithms. Firstly, these algorithms excel at extracting intricate patterns and detecting subtle fault signatures from raw data, leading to improved detection accuracy[14]. This enables early identification of bearing faults, facilitating timely maintenance interventions. Secondly, roller element bearing faults can exhibit complex patterns, ranging from surface defects to misalignment and lubrication issues. Traditional methods often struggle to effectively handle such diverse fault patterns[15]. Machine learning and deep learning algorithms can automatically learn and adapt to different fault patterns, enabling accurate detection and classification of various types of bearing faults.

Moreover, bearings operate under varying load conditions, speeds, and environmental factors, which can impact the characteristics of fault signals and pose challenges for fault detection[16]. Machine learning and deep learning algorithms are robust to different operating conditions and demonstrate reliable performance in fault detection, ensuring consistent results across diverse scenarios. In addition, bearings generate large volumes of data, such as vibration signals, which require efficient processing and analysis. Machine learning and deep learning algorithms are capable of handling extensive datasets, facilitating comprehensive analysis of long-term vibration signals. This allows for the identification of transient fault events that may go unnoticed by traditional methods[17].

Furthermore, traditional fault detection methods often rely on periodic inspections, resulting in delayed fault detection and reactive maintenance. In contrast, machine learning and deep learning algorithms can be seamlessly integrated into real-time monitoring systems, enabling continuous and automated analysis of bearing condition[18]. This enables proactive maintenance strategies, reducing downtime and improving overall operational efficiency. Comparative analysis of different machine learning and deep learning algorithms further

contributes to the advancement of the field. By comparing their respective strengths, weaknesses, and suitability for bearing fault detection, researchers gain insights that guide future research and facilitate practical implementation of fault diagnosis systems[19].

The drive to enhance the accuracy, efficiency, and reliability of fault diagnosis in roller element bearing detection has fueled the exploration of machine learning and deep learning algorithms. The objective of this research is to provide industry professionals with powerful tools for proactive maintenance, resulting in reduced downtime, mitigated risks, and maximized operational efficiency in the realm of rotating machinery[20].

1.3 AIM AND OBJECTIVES OF THE RESEARCH

This dissertation project aims to create a comprehensive deep learning model for designing and diagnosing ball bearing faults. The main emphasis is on leveraging different deep learning architectures, such as CNN, LSTM, and CNN-LSTM, to extract crucial features from vibration data and effectively classify the faults found in bearings. To accomplish this goal, the project will make use of the readily accessible Case Western Reserve University (CWRU) bearing dataset for modeling and analysis purposes. The project's specific objectives can be summarized as follows.

- Improve Fault Diagnosis Accuracy: The primary objective of this project is to improve the accuracy of fault detection in roller element bearings. Conventional approaches encounter difficulties in attaining high levels of accuracy due to the dependence on manual feature extraction and the requirement for expertise. By employing deep learning and machine learning algorithms, the project seeks to automate the feature extraction process from vibration, acoustic, or other sensor signals. This automation is expected to result in improved accuracy when identifying various fault conditions.
- Overcome Limitations of Traditional Methods: Conventional fault diagnosis approaches in bearing analysis involve manual feature extraction, which is prone to being time-consuming, subjective, and potentially missing

relevant information. To overcome these limitations, the project aims to leverage the capabilities of machine learning and deep learning algorithms. By harnessing these advanced techniques, the project enables autonomous learning of complex patterns and ordered representations directly from raw data. This eliminates the need for manual feature engineering and allows for a comprehensive analysis of the bearing condition, effectively addressing the drawbacks associated with traditional methods.

- Utilize ML and DL Algorithms: The project plans to explore and utilize a range of machine learning algorithms, including Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, and Random Forests. These algorithms will be utilized for the purposes of feature extraction and classification of bearing faults. Additionally, The project will delve into deep learning algorithms, such as Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and integrated CNN-LSTM models, to investigate their potential and effectiveness. The objective is to effectively capture intricate fault patterns and enhance the accuracy of fault classification.
- Address the Need for Labeled Data: To facilitate the training and estimation of deep learning and machine learning algorithms, ample labeled datasets are crucial. The project will make use of the renowned and openly accessible Case Western Reserve University (CWRU) bearing dataset. This dataset comprises extensive vibration data captured from bearings experiencing diverse fault conditions.
- Reduce Computational Complexity: The computational demands of deep learning algorithms can be substantial, necessitating considerable computational resources and time for training and inference. To tackle this challenge, the project will meticulously choose and optimize deep learning architectures. Architectures like CNN, LSTM, and CNN-LSTM, known for their efficient signal data processing and successful track record in fault diagnosis tasks, will be employed. By leveraging these architectures, the project aims to mitigate computational complexity while upholding a high level of accuracy in fault diagnosis.

- Implement ML and DL Frameworks: The project's practical implementation will entail the exploitation of deep learning and machine learning frameworks, including Sklearn, TensorFlow, and PyTorch. These frameworks offer a reliable platform for developing, training, and evaluating deep learning and machine learning models. By harnessing the capabilities of these frameworks, the project aims to optimize the models and guarantee their efficient deployment in real-world scenarios.

Through the accomplishment of these objectives, the project endeavors to make a valuable contribution to the domain of roller element bearing fault detection. By implementing an enhanced and automated approach utilizing deep learning and machine learning algorithms, the project aims to enhance the overall efficiency and accuracy of fault detection. The likely outcomes of this investigation hold significant implications for the reliability, safety, and maintenance practices of rotating machinery in industrial applications. The anticipated results include reduced downtime, heightened productivity, and cost savings.

1.4 ORGANISATION OF THE THESIS

The research on bearing fault diagnosis using ML and DL algorithms is structured into respective chapters, aiming to provide a comprehensive examination of the subject matter. The initial chapter serves as an introduction, elucidating the research's background, motivation, objectives, and scope. Subsequently, the literature survey chapter undertakes a thorough review of existing studies and research papers on bearing fault diagnosis, emphasizing the utilization deep learning and machine learning algorithms in this particular area. The following chapters delve into data analytics and data visualization, meticulously analyzing the CWRU dataset and employing visualization techniques to gain valuable insights into the behavior of bearing signals. The subsequent chapter focuses on feature engineering, exploring diverse methods for extracting informative features from the original bearing signals. Moving forward, the project investigates machine learning algorithms, applying various approaches to classify bearing faults, while another chapter explores the capability of advanced deep learning models in the same

context. In the results chapter, an extensive analysis and comparison of the ML and DL algorithms performance is presented. Lastly, the conclusion chapter provides a concise summary of the research findings, contributions, limitations, and future avenues of exploration. This tends as a valuable source for investigators and industry professionals involved in predictive maintenance within the field.

CHAPTER 2
LITERATURE REVIEW

2.1 INTODUCTION

In this chapter, the background of fault diagnosis in rotating machinery is presented, emphasizing the utilization of artificial intelligence (AI) techniques. The literature review investigates a wide range of AI-based approaches employed in fault diagnosis, encompassing research published journal papers , reports and conference articles. The chapter conducts a comprehensive analysis of the different methods utilized, critically examining their respective strengths and limitations. By thoroughly exploring the landscape of AI-based methods for detecting bearings faults, the chapter draws valuable conclusions based on insights derived from the reviewed literature.

2.2 RESEARCH BACKGROUND

2.2.1 Bearing Condition Monitoring

Rotating machinery plays a pivotal role in numerous industrial applications as they are widely used mechanical components. These machines primarily consist of gear boxes, rolling element bearings, and rotary shafts. Bearings in rotating machinery are responsible for minimizing friction and facilitating the smooth rotation of shafts[21]. In most cases, these machines operate in challenging conditions characterized by high loads and elevated temperatures. Such conditions can lead to significant malfunctions, leading to decreased equipment performance, compromised safety, reduced availability and reliability, economic setbacks, and inferior product quality[22].

To address the challenges of machine reliability and maintenance-related expenses, a more efficient maintenance approach known as condition-based maintenance (CBM) has been implemented. In Figure 2.1, a schematic diagram

compares the operational, maintenance, and total costs associated with different maintenance strategies[23]. CBM focuses on maximizing productivity, machine uptime, and cost reduction by performing maintenance tasks when necessary. This strategy involves monitoring the actual conditions of a machine to assess its health status. When indicators indicate potential machine failure, maintenance actions are initiated[24]. A comprehensive CBM program incorporates two crucial elements: diagnostics and prognostics, which enable the identification of current machine conditions and the prediction of future failures.

Diagnostics encompasses the tasks of detecting, isolating, and identifying faults when they occur within a system. Fault detection alerts the presence of a failure, fault isolation determines the specific faulty component, and fault identification reveals the nature of the detected fault[25]. Diagnostics encompasses the tasks of detecting, isolating, and identifying faults when they occur within a system. Fault detection alerts the presence of a failure, fault isolation determines the specific faulty component, and fault identification reveals the nature of the detected fault. On the other hand, prognostics involves analyzing prior events to predict failures before they happen. Prognostics proves to be more effective than diagnostics in minimizing machine downtime. However, diagnostics becomes necessary when prognostics fails to predict a fault accurately and an actual failure takes place.

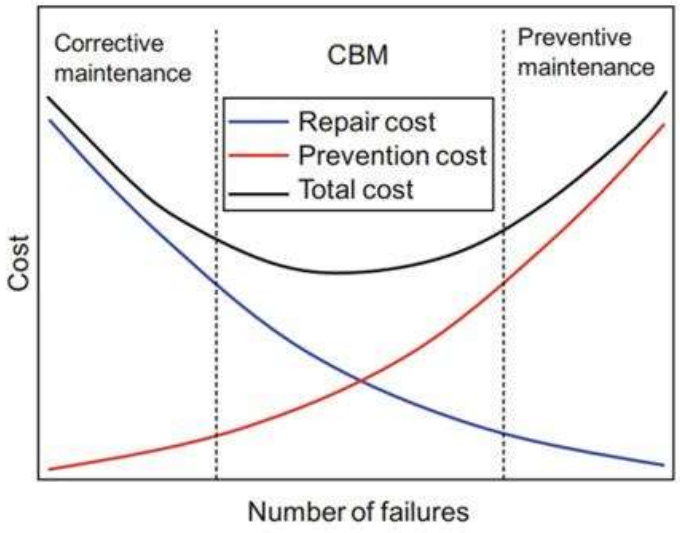

Figure 2.1 Maintenance Strategies versus Cost[26]

Currently, there exist various methods for diagnosing faults in rolling element bearings. In recent years, vibration-based condition monitoring has gained widespread popularity for detecting bearing faults[27]. However, the non-stationary nature of vibration signals during actual operation poses challenges in accurately extracting fault features. Consequently, this impacts the precision of fault diagnosis. Furthermore, the dynamic changes in vibrational behavior often present complex and diverse data patterns, making it challenging to establish accurate diagnoses. In such situations, having an accurate data driven model becomes invaluable as a reference point for diagnosis[28].

2.2.2 Rolling Element Bearings

Rolling element bearings are essential mechanical components utilized to facilitate rotational or linear motion between two moving parts. They comprise four key elements: the outer race, inner race, cage, ball, and rolling components, are showed in Figure 2.2. These components work in harmony to enable smooth and efficient motion transmission, minimizing friction and supporting the load-bearing requirements of various mechanical systems[29].

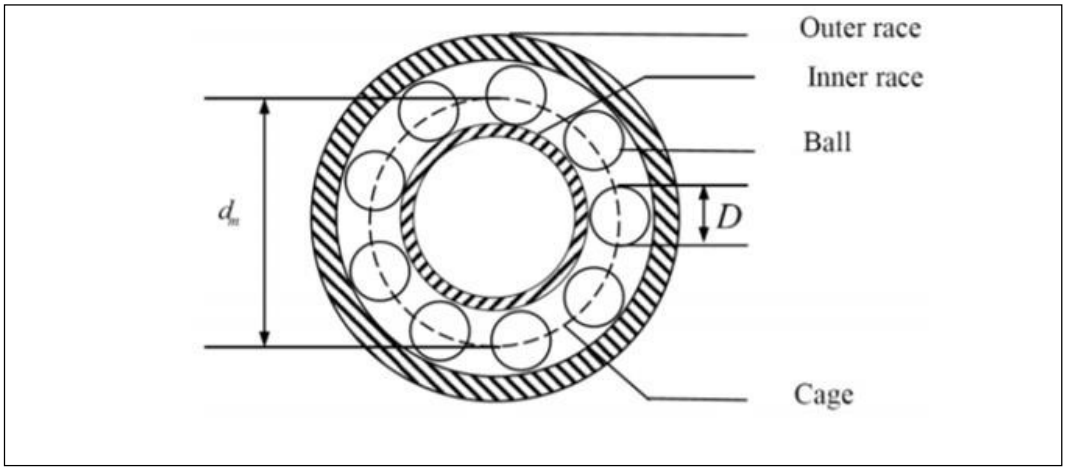

Figure 2.2 Roller Element Bearing Components

2.1.1.1 Characteristics of bearing fault

When a localized defect occurs in a mechanical system, such as a rolling element, it gives rise to periodic impulses. These impulses possess distinctive characteristics depending on the identifiable location of the fault. The dynamic behavior of each component of the bearing is associated with specific frequencies, namely the outer race characteristic frequency (f_o,), inner race characteristic frequency (f_i), roller characteristic frequency (f_r), and cage characteristic frequency (f_c)[30].

Outer race frequency:

$$f_o = \frac{N \cdot d \cdot B}{2 \cdot P} \tag{2.1}$$

Inner Race Characteristic Frequency:

$$f_i = f_o \cdot \frac{d}{D} \tag{2.2}$$

Roller Characteristic Frequency:

$$f_r = \frac{N \cdot d}{2 \cdot P} \tag{2.3}$$

Cage Characteristic Frequency:

$$f_c = N_c \cdot \frac{N}{60} \tag{2.4}$$

The provided equations in the previous response outline the calculation of characteristic frequencies associated with different components of a bearing. Equation 2.1 enables the determination of the outer race frequency (f_o) by considering factors such as the rotational speed (N), the number of bearing rolling elements (B), the pitch diameter of the bearing (d), and the bore diameter of the

outer race (P). Equation 2.2 establishes the inner race characteristic frequency (f_i) using similar parameters. Furthermore, Equation 2.3 facilitates the calculation of the roller characteristic frequency (f_r) which depends on the rotational speed (n) and the diameter of the rollers (d). Lastly, Equation 2.4 outlines the determination of the cage characteristic frequency (f_c)) by incorporating the rotational speed (n), the diameter of the rollers (d), and the length of the cage (N_c). These equations offer a means to accurately estimate and analyze the distinct frequencies associated with the various components of a bearing, aiding in the understanding and assessment of its dynamic behavior[31].

2.3 REVIEW ON BEARING FAULT DIAGNOSIS APPROACHES

Condition monitoring of a roller element bearing (REB) involves an intelligent diagnosis approach that revolves around pattern recognition. Drawing from previous research, as illustrated in Figure 2.3, a standard methodology for intelligent diagnosis comprises four key steps: measuring signals, extracting pertinent features, identifying the most informative features, and utilizing a classification technique based on these features.

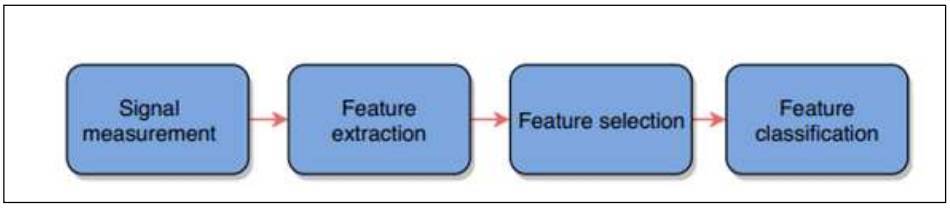

Figure 2.3 Fundamentals Steps of Intelligent Fault Diagnosis

Signal features can be derived from various domains, including the time frequency domain and time domain. While the real vibration signals from bearing systems are typically represented in the time domain, they can also be analyzed in the time frequency domain[32]. The Fourier Transform (FT) is a widely employed signal processing method for converting signals into the frequency domain. In addition, time-frequency domain features can be extracted using techniques such as Wavelet Packet Transform (WPT), Dual-tree Complex Wavelet Transform (DT-CWT) and Short-time Fourier Transform (STFT)[33].

The feature selection techniques employed in the intelligent diagnosis methodology play a crucial role in enhancing efficiency, reducing computational

complexity, and improving the accuracy of the classification process. When the relevant features dataset are selected, the next step involves feature classification. These selected features are fed into a machine learning classifiers , including Random Forest, Decision tree and Support Vector Machine (SVM), to accurately classify the bearing fault[34]. Among these machine learning algorithms, Random Forest has proven to be particularly effective and achieves high accuracy in this task.

These ANN models serve as powerful tools for bearing fault identification. These diverse ANN models contribute to the success of feature classification in bearing fault diagnosis, enabling accurate identification and timely maintenance decisions for the monitored machinery.

2.3.1 Machine Learning Based Fault Diagnosis

This section provides a concise overview of the utilization of various machine learning models in Intelligent Fault Diagnosis (IFD) strategies. In IFD, machine learning models are applied to determine the health conditions of machines when confronted with unlabeled input samples. The training of diagnostic models in IFD initially relies on labeled samples to achieve this objective. The subsequent sections of this part are organized based on the various types of algorithms and their applications in IFD research.

2.3.1.1 Support Vector Machine (SVM)

Datta et al. conducted a thorough investigation into various techniques for pipeline leakage detection, including vibration analysis, acoustic approaches, and SVM-based methods[53]. Notably, acoustic reflectometry emerged as the most effective approach, capable of detecting obstacles and leaks in pipes as small as 1% of their diameter. On a related note, Stetco et al. conducted an extensive review of recent research on machine learning models for monitoring the status of wind turbines. The majority of the reviewed models utilized datasets such as SCADA or simulated data, with approximately two-thirds focusing on classification tasks and the remaining on regression analysis, SVMs demonstrated high accuracy in most scenarios, achieving over 90% accuracy in the majority of cases[54].

2.3.1.2 Decision Tree (DT)

In the domain of power systems, Decision Tree (DT) models have found extensive applications in fault identification, security evaluation, and system control. These models offer simplicity and interpretability, making them effective in accurately detecting flaws by combining testing data with expert knowledge [56]. For example, in the context of fault diagnosis and categorization, a research study utilized decision trees based on faulty data from solar photovoltaic (PV) systems[57]. The DT models demonstrated exceptional accuracy in detecting and classifying defects during experimental assessments. In Another study conducted by Yan et al. employed the DT model to induce a data-driven diagnostic tool for Air Handling Units (AHUs)[56]. This approach incorporated a classification and a regression model, resulting in remarkable fault classification efficiency with an mean F-score of 0.97.

2.3.1.3 Random Forests (RF)

Random Forest classifiers, like XGBoost, offer a combination of accuracy, computational efficiency, and interpretability for fault classification of bearings, especially in situations where bearing vibration datasets may be unlabeled and have lots of noise. In the context of bearing fault classification, researchers have harnessed the power of Random Forest classifiers to develop robust systems. For example, Cerrada et al. proposed a most advanced method to predict multi-class faults in spur gears by incorporating Random Forest classifier[35]. The effectiveness of this method was demonstrated through the analysis of real vibration signals, encompassing different fault classes under varying vibration frequency data. Similarly, Patel and Giri employed a Random Forest classifier to classify the fault in bearings, using vibration signals captured by an accelerometer sensor and extracting statistical features as input to the Random Forest model. Moreover, non-mechanical fault identification have been classified by the random forest[36].

2.3.1.4 K-Nearest Neighbors (KNN)

Numerous research studies have successfully utilized KNN in fault detection and categorization tasks. For instance, in the study conducted by Tian et al., KNN was applied to classified bearing faults on vibration data[65]. The researchers extracted fault features, to detect bearing fault. Similarly, Naik and Koley proposed a fault detection and classification approach for bearing with a different load condition using KNN[66]. Through extensive testing in various fault scenarios, their KNN-based approach achieved fault detection and classification accuracy of 100% in all cases[37].

2.3.1.5 Bayesian Networks (BN)

These models utilize a large volume of historical data and employ backward analysis techniques to construct fault detection models [5]. A notable research example is the work conducted by Muralidharan et al. where BNs are employed to diagnose defects in centrifugal pump components using discrete wavelet features extracted from vibration signals[68]. In a separate investigation conducted by Zhao et al. a novel Diagnostic Bayesian Networks (DBNs)-based method is introduced for fault classification in varying diameter bearing, which encompass a wide range of common issues[69]. The construction of DBNs incorporates vibration data from multiple bearings for fault classification and detection studies, along with a comprehensive analysis of feature extraction techniques and fault patterns. The results demonstrate the capability of the DBNs approach to effectively identify faults even in situations where faulty data is limited or ambiguous.

2.3.2 Deep learning Based Fault Diagnosis

Deep learning-based fault detection methods have developed as a transformative approach to identify the limitations of conventional intelligent defect detection systems[38]. These systems leverage the power of hierarchical networks, such as multilayered Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), to learn intricate feature representations from faulty data. Unlike

traditional methods, deep learning algorithms employ non-linear operations that enable the discovery of meaningful patterns. By applying these non-linear activation function, deep learning methods smartly extract informative features from input raw data, facilitating accurate estimation of complex and non-linear functions. This paradigm shift in diagnostic systems offers promising opportunities for more accurate and comprehensive fault detection and diagnosis in various industrial applications[39].

2.3.2.1 Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a very popular neural network that follows a feed-forward architecture and comprises three primary layer types: Convolutional Layers (CL), Pooling Layers (PL), and Fully Connected Layers. While CNNs Fully Connected Layer functions similarly to a very common feed-forward neural network, the distinguishing advantage and benefits of CNNs arise from the distinctive structures and operations of CLs and PLs. These layers enable CNNs to effectively process and extract features from two-dimensional data. Figure 2.4 visually illustrates the CNN architecture specially designed for handling 2-D data[40].

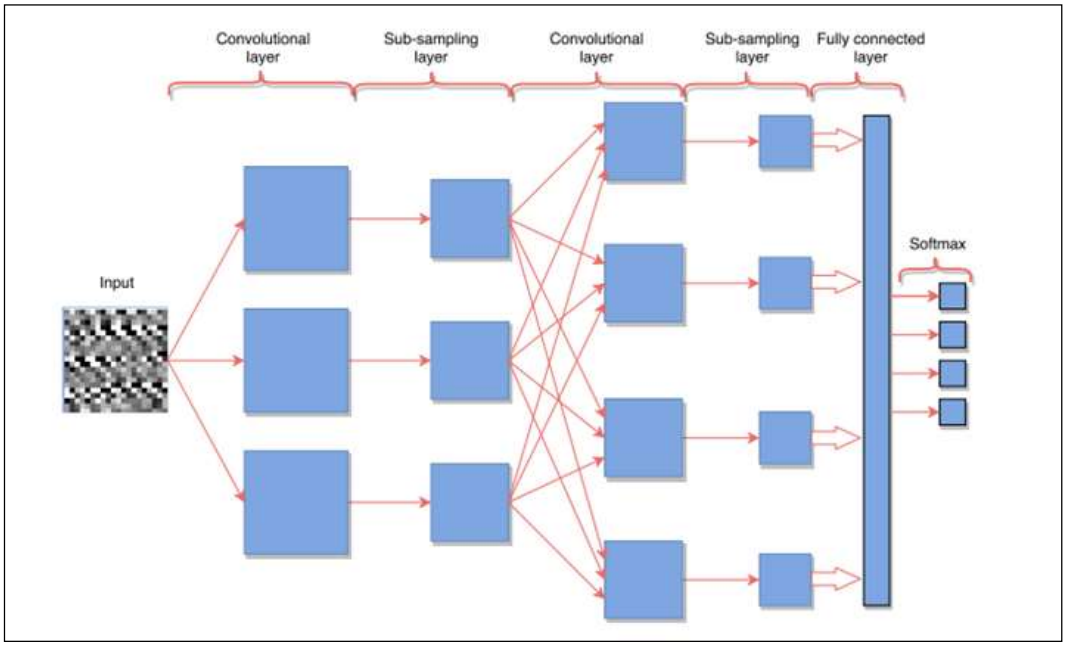

Figure 2.4 CNN architecture

Convolutional neural networks (CNNs) have many applications in computer vision tasks. CNNs utilize core components, including convolution layers , pooling layers and fully connected layers, to find hierarchical representations of spatial features through backpropagation. The convolution layer, a crucial element within CNNs, performs mathematical operations such as convolution on pixel grids in digital images, employing customizable feature extractors known as kernels[41].

Chen et al. conducted a study that focused on detecting and classifying defects in gearboxes using convolutional neural networks (CNNs)[75]. They examined the vulnerability of gearbox vibration signals to defects and developed a deep learning technique to address this issue. Through tests involving 20 different cases, encompassing 12 distinct combinations of fundamental condition patterns, the proposed technique demonstrated high reliability and effectiveness in identifying defects in industrial reciprocating equipment. Prior to Chen et al.'s research, Janssens et al. conducted a study highlighting the reliance on manually engineered aspects for automatic fault detection, such as ball pass frequencies in the raceway[48]. However, Janssens et al.'s research successfully identified distinguishing characteristics of bearing faults solely from the data itself. These studies contribute to the advancement of fault detection techniques by leveraging CNNs and data-driven approaches in the area of fault classification[52].

The adoption of CNNs feature-learning methods has shown significant advancements over traditional feature-engineering techniques, which rely on manual feature construction and random forest classifiers. Furthermore, the comparison with existing techniques showcased the superiority of the upgraded algorithm, highlighting its suitability for fault diagnosis models[42]. These findings contribute to the field of intelligent fault diagnosis by offering a more effective and robust approach for identifying and analyzing machine faults.

Jing et al developed a specialized CNN model aimed at extracting meaningful feature representations from vibration signal frequency data. The researchers examined and compared the effectiveness of feature extraction techniques applied to raw data vibration data. The evaluation of their proposed method was carried out using datasets from the gearbox test rig, showcasing its effectiveness in accurately analyzing and diagnosing mechanical faults[43]. This

research proposed to the improvement of fault classification by highlighting the potential of deep learning techniques in feature extraction from vibration signals, thus enhancing the accuracy and reliability of mechanical fault detection and diagnosis.

Furthermore, Wen et al. introduced an innovative CNN architecture based on LeNet-5 for defect identification in machinery. These studies demonstrated improved classification performance compared to conventional algorithms like artificial neural networks (ANNs) and SVMs. Moreover, these methodologies have found applications in predicting the remaining usable life of the bearings[44].

In Wang et al.'s research, they investigated the utilization of CNNs to enhance feature learning for multi-fault categorization in mechanical systems. The experimental results demonstrated promising mean fault classification accuracy between 98.125 percent and 98 percent for varying time serious data , respectively, along with a significant reduction in error rates[48]. Another study explored the integration of cognitive computing theory into bearing issue diagnoses by leveraging the capabilities of image recognition and visual perception [49]. The employed CNN model effectively reduced the computational requirements in the temporal dimension, allowing for efficient identification of crucial bearing features. Furthermore, the model exhibited high invariance in the presence of ambient noise.

Zhang et al. developed a famous approach known as the Wide First-Layer Kernel Deep Convolutional Neural Network (WDCNN) for processing raw vibration signals in fault detection[39]. The WDCNN model incorporates augmentation techniques to gather additional information .The results of their study demonstrated that the WDCNN outperformed existing deep neural network (DNN) models, especially in challenging conditions with different working loads and noisy environments, depends on the varying frequency vibration signals[45]. It is important to note that recent defect detection algorithms based on CNNs , commonly using the transfer learning techniques to train pre-trained models, as training these models from scratch with large datasets can be time-consuming and computationally intensive. The subsequent section of this article will provide a more comprehensive examination of these works.

2.3.2.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) is known as a specialized form of neural network architecture that excel at processing sequential or temporal data. Unlike traditional feedforward neural networks, RNNs can effectively capture dependencies and patterns within a sequence[46]. This is made possible by their unique memory mechanism. As a result, RNNs have demonstrated remarkable success in natural language processing, and image captioning. Their power to model the temporal dynamics of data makes them a powerful tool for tasks that involve time-dependent relationships[47].

RNN-based algorithms excel in handling sequential data of varying lengths and effectively capturing long-term dependencies within machine-generated signals. Their inherent capability to uncover hidden structures empowers them to enhance the model's generalization abilities and uncover previously undiscovered patterns. The autoencoders predict vibration levels in rolling bearings based on preceding time periods. The GRU-based nonlinear predictive denoising autoencoders exhibit remarkable generalization capabilities and are trained for each specific defect type. This approach demonstrates robustness and high accuracy in fault categorization tasks[48].

In a separate investigation, Zhao et al. devised a hybrid approach combining convolutional neural networks and recurrent to tackle fault detection challenges. Their methodology involved employing Convolutional Bidirectional Long Short-Term Memory (CBLSTM) networks to process sensory information, while stacked fully connected (FC) and linear regression layers were utilized for forecasting target values. In certain scenarios, robust RNN architectures stand alone as the primary classifier module for detecting equipment faults[49].

Rafique et al. introduced a Long Short-Term Memory (LSTM) network for fault classification in bearings. Unlike conventional approaches that rely on pre-engineered features, their novel LSTM-based model directly processes operational data, eliminating the need for complex feature extraction. This end-to-end framework showcases remarkable responsiveness and adaptability to diverse operational scenarios, enabling expedited decision-making processes [50]. Table 2.1 showing some important and latest literature review so this research.

Table 2.1 Summary of Literature

Sr. No	Year	Title	Focus
1	2023	Online bearing fault diagnosis using numerical simulation models and machine learning classifications	The method uses a simulation model built using a mature dynamic model and modified through the Pearson correlation coefficient (PCC) for online learning.
2	2022	A feature extraction and machine learning framework for bearing fault diagnosis	The paper proposes a machine learning framework for bearing fault diagnosis in wind turbines. The framework involves unsupervised learning to understand data properties, sensitivity analysis to extract significant features, and a three-stage learning algorithm to refine and learn useful information for fault diagnosis.
3	2021	FaultNet: a deep convolutional neural network for bearing fault classification	The paper proposes a deep convolutional neural network called FaultNet for bearing fault classification using vibration signal data.
4	2021	Bearing fault diagnosis using deep learning techniques coupled with handcrafted feature extraction: A comparative study	The paper uses stacked autoencoders for feature extraction and a classifier to diagnose bearing faults. The Case Western Reserve University bearing dataset was used for the study.
5	2021	A physics-informed deep learning approach for bearing fault detection	The methodology of the paper involves the use of a physics-informed deep learning approach for bearing fault detection, which consists of a threshold model and a deep convolutional neural network (CNN) model.
6	2021	A novel feature adaptive extraction method based on deep learning for bearing fault diagnosis	The paper proposes a ResNet-STAC-tanh network architecture for fault diagnosis of bearing datasets.
7	2021	Application of machine learning to a medium Gaussian support vector machine in the diagnosis of motor bearing faults	The paper proposes the use of a medium Gaussian support vector machine (SVM) method for machine learning in the diagnosis of motor bearing faults.
8	2020	Bearing Fault Classification of Induction Motors Using Discrete Wavelet Transform and Ensemble Machine Learning Algorithms	The paper proposes an ensemble machine learning-based fault classification scheme for induction motors using the discrete wavelet transform for feature extraction.
9	2020	A Novel Bearing Fault Classification Method Based on XGBoost: The Fusion of Deep Learning-Based Features and Empirical Features	The paper proposes a novel method for bearing fault classification using a fusion of simple empirical features and adaptive features extracted by a deep neural network.
10	2020	A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks	The paper proposes a novel deep learning-based method for bearing fault diagnosis using cyclic spectral coherence and convolutional neural networks.
11	2020	Induction motor fault monitoring and fault classification using deep learning probablistic neural network	The paper proposes a novel approach for monitoring and classifying faults in induction motors using a Probabilistic Neural Network (PNN) and Discrete Cosine Transform (DCT).

Table 2.1 Contd.

12	2020	Deep learning-based bearing fault diagnosis method for embedded systems	The paper proposes a new approach for establishing a CNN-based process for bearing fault diagnosis on embedded devices using acoustic emission signals, which reduces the computation costs significantly in classifying the bearing faults.
13	2020	Bearing fault diagnosis of induction motors using a genetic algorithm and machine learning classifiers	The paper proposes a hybrid motor-current data-driven approach that utilizes statistical features, genetic algorithm (GA) and machine learning models for bearing fault diagnosis in induction motors. The proposed approach achieves an accuracy of more than 97% and is promising for diagnosis of IM bearing faults.
14	2020	Infrared thermography-based fault diagnosis of induction motor bearings using machine learning	The paper proposes a method for detecting bearing faults in induction motors using infrared thermography and machine learning techniques.
15	2020	A novel feature extraction method for bearing fault classification with one dimensional ternary patterns	The proposed method in this paper involves using one-dimensional ternary patterns for feature extraction in order to classify bearing faults.
16	2019	Understanding and improving deep learning-based rolling bearing fault diagnosis with attention mechanism	The paper proposes a novel deep learning-based fault diagnosis method for rolling element bearings. The proposed method includes an attention mechanism, network architecture, and diagnosis procedure.
17	2019	A motor current signal-based bearing fault diagnosis using deep learning and information fusion	The paper proposed method uses raw signals from multiple phases of the motor current as direct input, features are extracted from the current signals of each phase, and a decision-level information fusion technique is introduced to enhance the classification accuracy.
18	2019	Application of spectral kurtosis and improved extreme learning machine for bearing fault classification	The paper briefly mentions a proposed fault classification method and a critical issue of ELM, which is deciding the value of L that determines the size of ELM architecture. It also mentions that providing a compact architecture with good performance is a challenge in ELM and is usually done by trial-and-error basis.
19	2019	LSTM based bearing fault diagnosis of electrical machines using motor current signal	The paper proposes a method involves filtering the redundant frequencies in the signal, extracting eight features from the time and time-frequency domain using wavelet packet decomposition, and using the Long Short-Term Memory (LSTM) algorithm for fault classification.
20	2019	A novel deep output kernel learning method for bearing fault structural diagnosis	The paper proposes a new method for bearing fault diagnosis using deep output kernel learning, which combines output kernel learning and ML-ELM to improve predictive accuracy and numerical stability.
20	2018	Simulation-driven machine learning: Bearing fault classification	The paper proposes a novel approach to generating training data using high resolution simulations of roller bearing dynamics and compares several different machine learning methodologies for race fault detection.
22	2018	Bearing fault automatic classification based on deep learning	The paper proposes an automatic classification method for bearing fault diagnosis based on deep learning. The method is designed to cluster faulty signals.
23	2017	A hybrid feature model and deep-learning-based bearing fault diagnosis	It uses a hybrid feature pool and a sparse stacked autoencoder-based deep neural network to effectively diagnose bearing faults of multiple severities.

Table 2.1 Contd.

24	2017	Deep learning-based approach for bearing fault diagnosis	The paper presents a deep learning-based approach for bearing fault diagnosis using short time Fourier transform (STFT) pre-processing of sensor signals and an optimized deep learning structure.
25	2017	A novel multimode fault classification method based on deep learning	The paper proposes a multimode fault classification approach using hierarchical deep neural networks (HDNN) and compares it with traditional methods such as BPNN and SVM.
26	2017	Intelligent bearing fault diagnosis method combining compressed data acquisition and deep learning	The paper proposes a new intelligent diagnosis method for fault identification of rotating machines using compressed data acquisition and deep learning.
27	2015	Ball bearing fault diagnosis using supervised and unsupervised machine learning methods	The paper proposes a method for fault diagnosis in ball bearings using multiscale permutation entropy as a tool for feature selection.
28	2015	Investigation techniques for rolling bearing fault diagnosis using machine learning algorithms	The paper discusses the use of machine learning algorithms for detecting faults in rolling bearings, which can help prevent machine malfunctioning and enable effective preventive maintenance.
29	2011	Rolling element bearing fault diagnosis using wavelet transform	. The study uses a wavelet theory-based feature extraction methodology to extract statistical features from raw vibration signals.
30	2011	Control of mechatronics systems: Ball bearing fault diagnosis using machine learning techniques	The paper uses random forest algorithm and C4.5 decision tree for the fault diagnosis of ball bearing of three phase induction motor.

2.4 CONCLUSION

The literature review reveals several areas that require further development and exploration in the area of bearing fault classification. Firstly, there is a need to enhance feature extraction methodologies specifically tailored to bearing faults. These methodologies should consider the interpretations of experts in the field and aim to extract features that correlate closely with their expertise. This will enable more accurate and precise fault diagnosis in bearing systems. Secondly, the field would benefit from the advancement of artificial intelligence techniques that can surpass human expert interpretation in bearing fault diagnosis

The utilization of artificial intelligence techniques in bearing fault diagnosis shows great potential in decreasing the reliance on human experts. Continued exploration and research in these fields have the potential to advance automated and precise diagnostic procedures, leading to enhanced dependability and efficiency of bearing systems in diverse industrial sectors.

CHAPTER 3
DATA ANALYSIS AND DATA VISUALIZATION

3.1 INTODUCTION

This chapter on data analytics and data visualization revolves around a detail description of the dataset and the application of various data visualization methods, such as PCA, KPCA, and t-SNE, to visualize the data effectively. By employing these techniques, the chapter aims to unveil patterns, clusters, and distinguishability among different fault classes within the dataset. As a result, this chapter serves as a valuable resource in gaining insights into the dataset's nature and its potential in achieving accurate fault classification.

3.2 DATASET DESCRIPTION

The availability of diverse and comprehensive datasets is paramount for the successful development and evaluation of machine learning (ML) and deep learning (DL) algorithms in the area of bearing fault diagnosis. Among the commonly employed datasets are the Paderborn University dataset and the Intelligent Maintenance System (IMS) dataset[51]. Additionally, specialized datasets such as the PADERBORN dataset and the Case Western University Dataset are specifically curated to cater to the requirements of bearing fault detection tasks. This study focuses on utilizing vibration signals data as the primary raw data source for bearing fault classification, as it is a widely acknowledged and prevalent approach in the field. To obtain the vibration signal data, the researchers accessed the reputable Case Western Reserve University bearing data center, which is renowned for its comprehensive collection of bearing fault data. This data center is widely recognized and extensively utilized by researchers for studying and analyzing bearing faults[52].

3.2.1 Case Western Reserve University Bearing Dataset

To evaluate the performance of various machine learning (ML) and deep learning (DL) algorithms, this study employs bearing vibration frequency data files sourced from the Case Western Reserve University Bearing (CWRU) bearing fault dataset. The CWRU dataset is publicly available through the CWRU bearing data center website and serves as a valuable tool for evaluating the algorithms' effectiveness [17], [18]. Figure 3.1 depicts the experimental platform, offering a visual depiction of the setup employed during the data collection phase.

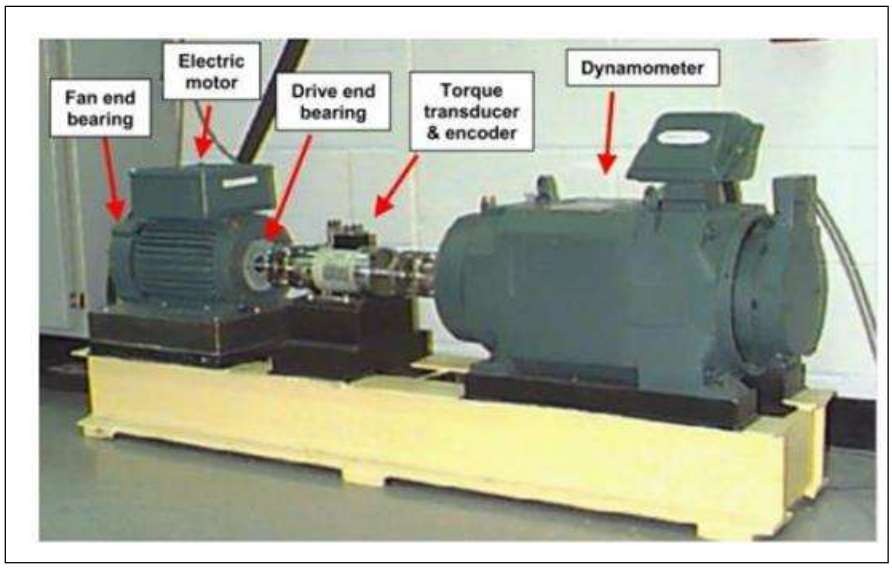

Figure 3.1 Experiment Setup of CWRU Dataset

For the purpose of validation, our experiments focused on vibration data files generated by a one horsepower (HP) engine running at a consistent speed of 1772 RPM. To simulate bearing faults of varying depths, intentional faults were introduced at different locations within the motor shaft bearings. These fault depths included 0.000 (representing no fault), 0.007, 0.014, 0.021, and 0.028 inches. Table 3.1, providing essential information for further analysis and classification of the bearing faults.

Table 3.1 Classes in CWRU Bearing Dataset.

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

3.2.2 Experiment Data Setup

During the experimental setup, the motor under investigation was fitted with bearings intentionally seeded with faults of predetermined depths: 0.007 inch, 0.014 inch, or 0.021 inch. To capture the vibration signals produced by these faulty bearings, accelerometers were strategically positioned at three distinct locations on the motor: the drive end (DE), fan end (FE), and base (BA). Time series data is collected from these three accelerometer locations, with a sampling frequency of either 12k or 48k. The sampling frequency determines the number of sample data points collected per second. For instance, a sampling frequency of 48k means that 48,000 data points are recorded per second. If a data file contains 4,800,000 data

points, it signifies that data has been collected for a duration of 100 seconds at a sampling frequency of 48k.

Figure 3.2 shows python code for this experiment.

```
In [2]:
files = glob.glob("C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data/*")


files = np.sort(files)
files

Out[2]: array(['C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\B007_1_123.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\B014_1_190.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\B021_1_227.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\IR007_1_110.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\IR014_1_175.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\IR021_1_214.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\OR007_6_1_136.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\OR014_6_1_202.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\OR021_6_1_239.mat',
               'C:/Users/Dev/Documents/Python Scripts/ML_MTECH_PROJECT/vib_data\\Time_Normal_1_098.mat'],
              dtype='<U85')
```

Figure 3.2 Python Code for Data Collection

Keys of each file contain drive end signal (*DE_time*), fan end signal (*FE_time*), and in some cases base signal (BA). X123RPM gives RPM information. We will only take drive end data for our analysis. Drive ends are closer to the bearing. So, we believe, it would give more reliable indication of fault. Thus, we have used only drive end data. We can get drive end data using its key.

3.3 DATA VISUALIZATION

In the present era, the data we gather is inherently characterized by high dimensionality, primarily due to the increased affordability of sensors, data acquisition devices, storage systems, and data processing tools. As a result, collecting vast amounts of data has become effortless.

The challenge with high-dimensional data lies in the difficulty of visualizing it through conventional plotting methods. In fact, we can only visualize a maximum of three dimensions at once. As a result, analyzing high-dimensional data becomes intricate, necessitating the development of algorithms capable of effectively classifying data in such higher-dimensional spaces. Therefore, it becomes essential to apply various techniques to the data to explore potential lower-dimensional representations.

In this study, we will utilize three commonly employed dimensionality reduction and data visualization methods: Principal Component Analysis (PCA),

Kernel Principal Component Analysis (KPCA), and t-distributed Stochastic Neighbor Embedding (t-SNE). These techniques have been extensively used in various visualization applications and are known for their effectiveness in the dimensionality reduction of the data although maintaining meaningful patterns and structures.

3.3.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a popular technique extensively applied in bearing fault diagnosis for both data visualization and dimensionality reduction purposes. PCA can help us gain insights into the relationships and variations among different features extracted from vibration or sensor data. Through Principal Component Analysis (PCA), we can identify the principal components, which are transformed variables representing linear combinations of the original features[53]. PCA allows us to visualize the data in a low dimensions vector space, often two or three dimensions, while retaining the key information. The visualization obtained through PCA assists in comprehending the inherent patterns, clusters, and anomalies inherent in the bearing fault data. This visualization offers valuable insights into the distinguishability of different fault types, the presence of overlapping regions, or the formation of distinct clusters[54] It allows us to discern underlying structures and identify potential relationships between the fault conditions.

3.3.1.1 Principal Component Analysis (PCA) Plot

The PCA plot provides a visual representation of the dataset by mapping each data point to a two or three-dimensional space. Each data point in the plot represents an instance from the dataset, and its position is determined by its values along the principal components. By examining the scatter of data points in the plot, we can gain insights into the distribution and relationships among the samples, allowing us to identify clusters, patterns, and separability among different classes or fault conditions.

Figures 3.3 and 3.4 show the PCA plot for CWRU bearing dataset before and after scaling.

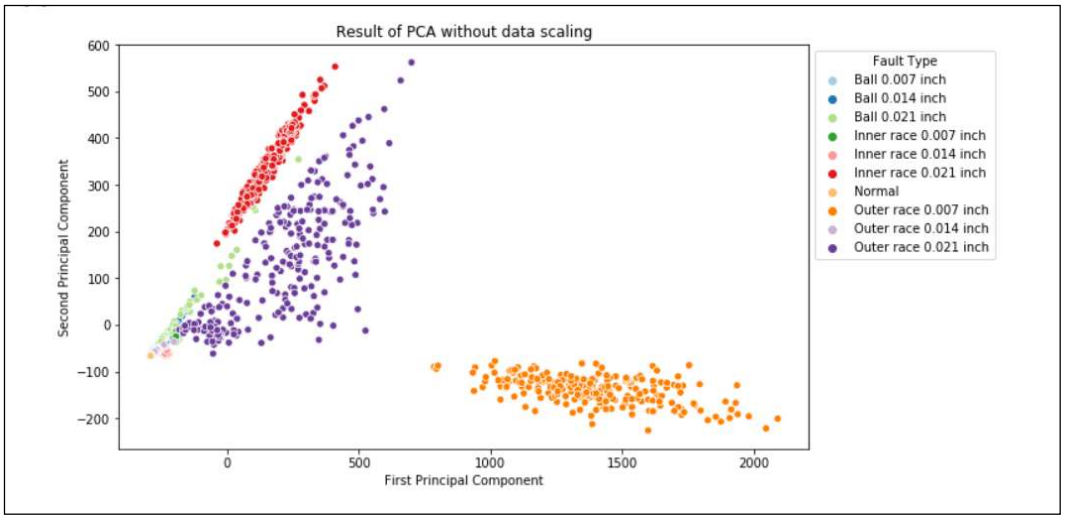

Figure 3.3 PCA Plot Without Data Scaling

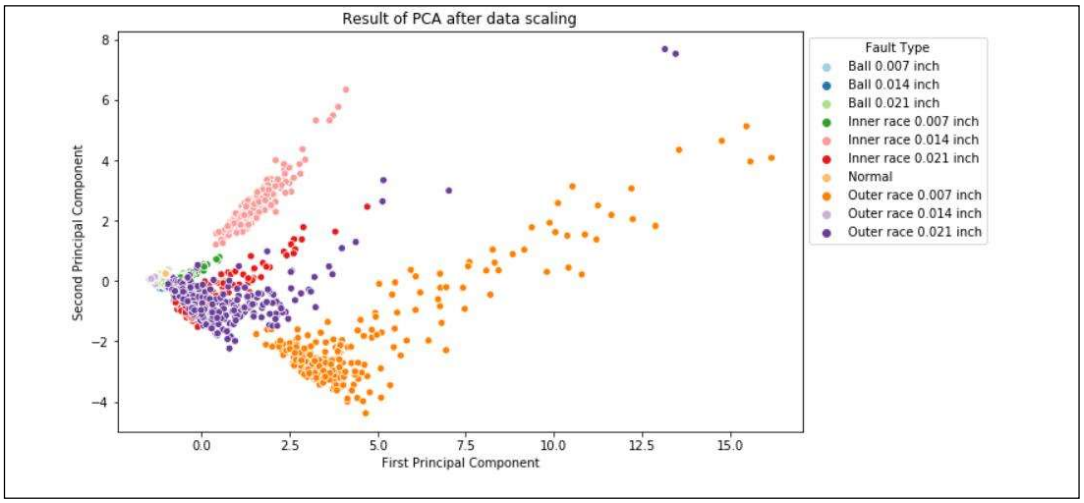

Figure 3.4 PCA Plot with Data Scaling

Projection using normal PCA doesn't segregate different classes. Sometimes scaling the data before applying PCA gives better results. This doesn't improve anything. So linear dimensionality reduction techniques are not that good at segregating this type of fault. Next, we can apply nonlinear dimensionality reduction technique to see if the situation improves. We will apply Kernel PCA to this data.

3.3.1.2 Kernel Principal Component Analysis (KPCA) Plot

Kernel Principal Component Analysis (Kernel PCA) method is a powerful method that extends the capabilities of traditional PCA to handle nonlinear and complex datasets commonly encountered in bearing fault diagnosis[54]. Unlike traditional PCA, which is limited to linear transformations, Kernel PCA utilizes nonlinear transformations to extract more informative features from the data. This allows for better representation and separation of different fault classes, enhancing the accuracy and performance of fault diagnosis models[55]. Kernel PCA is particularly beneficial when dealing with datasets that exhibit nonlinear patterns, enabling more comprehensive and accurate analysis of bearing faults.

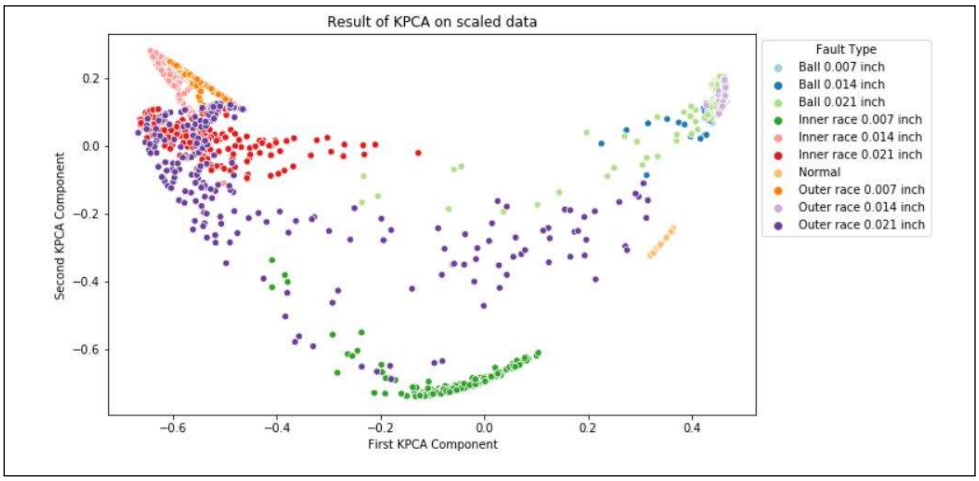

Figure 3.5 KPCA Plot for Scaled data

Considering the limited success achieved by previous methods in effectively separating the data, it is plausible to assume that the data points may lie on a nonlinear manifold. To overcome this challenge, the t-distributed stochastic neighbor embedding (t-SNE) algorithm will be utilized to visually segregate input data. Unlike PCA and Kernel PCA, t-SNE focuses on maintaining the original structure of the input features data, making it particularly suitable for nonlinear and complex datasets. By applying t-SNE, we aim to reveal hidden clusters, patterns, and relationships among the input features space data points that may not be visible in the original feature space. This will provide a more convincing and insightful representation of the bearing fault diagnosis dataset.

3.3.1.3 t-distributed Stochastic Neighbor Embedding (t-SNE) Plot

The t-SNE (t-distributed Stochastic Neighbor Embedding) algorithm is widely employed in the field of bearing fault diagnosis to visualize high-dimensional datasets effectively. Unlike other dimensionality reduction techniques, t-SNE focuses on preserving the local structure and clustering of data points, allowing for the detection of intricate relationships and patterns for input features data[56].

The t-SNE plot in Figure 3.6 provides a visual representation of the scaled CWRU Bearing dataset, showcasing the distribution and arrangement of data points in a lower-dimensional space. This plot aids in uncovering hidden structures, patterns, and anomalies within the dataset, offering valuable insights for further analysis and interpretation.

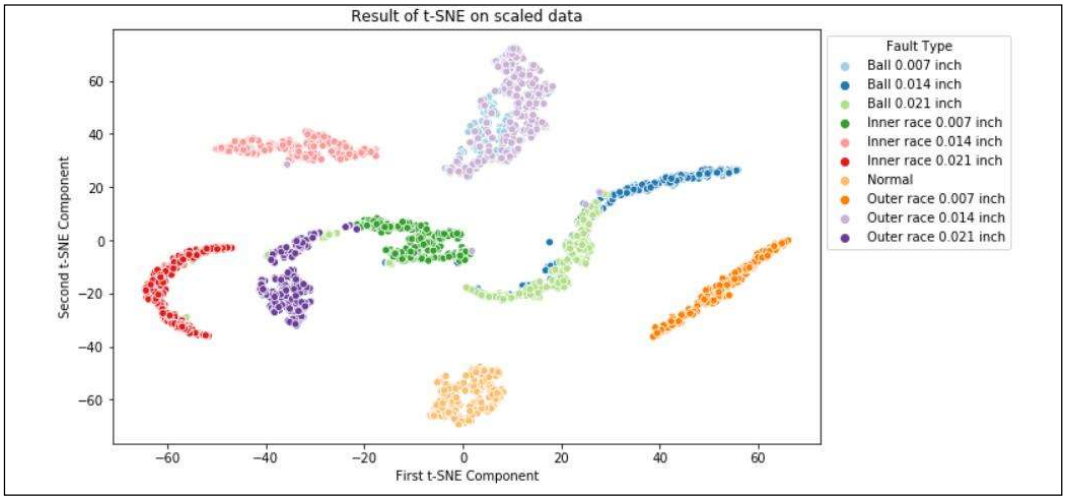

Figure 3.6 t-SNE plot on scaled data

The given data for bearing fault diagnosis presents some challenges in classification due to the overlapping of the 0.007 inch ball defect and 0.014 inch outer race defect faults. This overlapping makes it difficult to distinguish between these two fault types using traditional machine learning techniques. By employing t-SNE, the data points representing various fault types are transformed into a low dimension input features vector space, enabling a clearer visualization of their

relationships and similarities[57]. The resulting t-SNE plot provides researchers with a visual representation of the data, facilitating the identification of clusters or groupings that correspond to similar fault patterns. This visualization enhances the understanding of the underlying data structure that are used in different machine learning algorithms for fault classification.

3.4 CONCLUSION

In conclusion, the data analytics and data visualization chapter provided a comprehensive overview of the CWRU dataset for bearing fault classification. Additionally, the chapter delved into the application of dimensionality reduction techniques, including PCA, KPCA, and t-SNE, to visualize the data effectively. By reducing the dataset's dimensionality, these techniques allowed for a clearer representation of the data's patterns, clusters, and relationships. This visualization aided in gaining insights into the data and provided a solid foundation for further analysis and interpretation.

CHAPTER 4
FEATURE ENGINEERING

4.1 INTODUCTION

The feature engineering chapter focuses on extracting relevant features from the CWRU bearing dataset for fault classification. It introduces time domain features that capture statistical measures of the signal, and time-frequency domain features obtained through the wavelet transform. The chapter explains the process of calculating these features from the dataset and highlights the importance of selecting relevant subsets and reducing dimensionality. These features will serve as the foundation for building accurate fault classification models in the following chapters.

4.2 FEATURE ENGINEERING

Feature engineering is an essential step in bearing fault diagnosis, specifically when working with the CWRU dataset from the Case Western Reserve University. It involves extracting relevant features from raw vibration data to enhance the classification of different fault types[58]. Time domain analysis is one approach used to derive features by examining statistical properties. On the other hand, time-frequency domain analysis involves applying methods such as the short-time Fourier transform (STFT) or wavelet transform to capture both different frequency information[59]. By employing these feature engineering approaches, the dataset's informative features are extracted, enhancing the accuracy and effectiveness of subsequent classification algorithms for bearing fault diagnosis. Figure 4.1 visually depicts the process of transforming the raw vibration data into a set of suitable input features, which serve as inputs to machine learning algorithms.

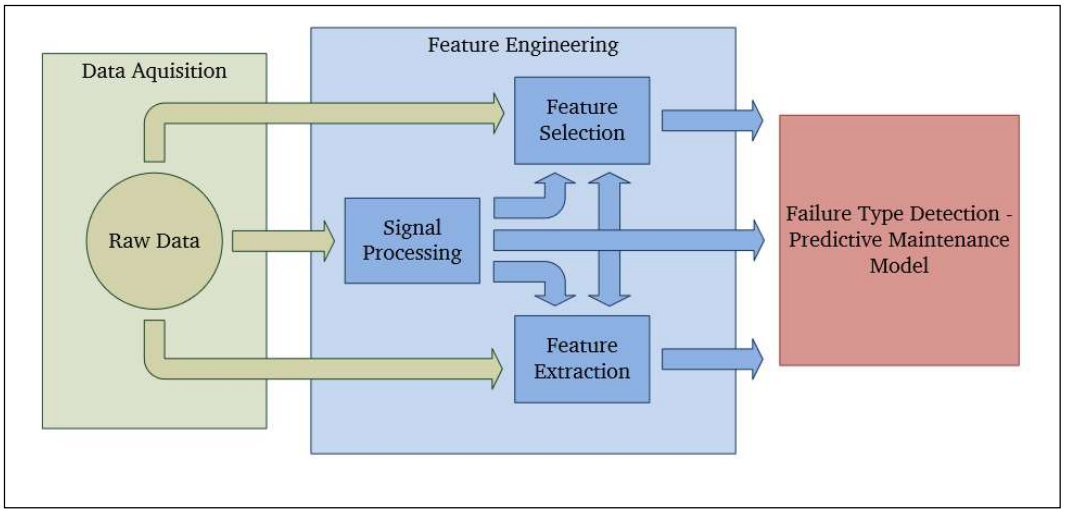

Figure 4.1 Process of Feature Engineering

4.3 TIME DOMAIN ANALYSIS

Time domain analysis is a commonly employed technique for bearing fault classification using the CWRU dataset. It focuses on obtaining relevant input features directly from the raw vibration signals to uncover distinctive patterns associated with various types of bearing faults.

These features provide valuable insights into the signal's central tendency, variability, spread, asymmetry, and peakedness. The mean represents the average value of the vibration signal, indicating its central tendency[60]. The standard deviation quantifies the signal's overall fluctuation, while the variance represents the spread of the signal values around the mean. Skewness estimate the asymmetry of the raw vibration data. Kurtosis measures the peakedness or flatness of the raw vibration data, with higher kurtosis suggesting a more peaked distribution with heavier tails[61]. By analyzing these statistical features, engineers and researchers can identify characteristic patterns and abnormalities associated with different types of bearing faults, facilitating accurate diagnosis and maintenance decision-making.

4.3.1 Compute Time Domain Features

For the purpose of this analysis, we will focus on the drive end data from the CWRU dataset. To ensure comprehensive analysis, the data is divided into non-overlapping

segments of length 1024. This segmentation strategy allows for the extraction of specific patterns and characteristics within each segment, enabling a detailed examination of the underlying fault conditions. To determine the maximum number of segments, we divide the length of the original vibration data by the length of each segment. This calculation provides us with an estimate of the total number of segments that can be extracted. Figure 4.2 shows the code for compute time domain features.

```
for file in files:
    data = loadmat(file)
    if file[-7:-4] == "175":          # Peculiar IR014_1 data
        key = "X217_DE_time"
    else:
        key = "X" + file[-7:-4] + "_DE_time"
    drive_end_data = data[key]
    num_segments = np.floor(len(drive_end_data)/1024)
    print(num_segments)
```

Figure 4.2 Code for Compute Time Domain Features

To calculate the time domain features for bearing fault classification using the CWRU dataset, a series of steps can be followed. First, the vibration signal data is preprocessed to ensure its suitability for analysis. This involves segmenting the signal into appropriate blocks or segments and applying necessary preprocessing techniques such as filtering, normalization, or noise removal to enhance the signal quality.

To ensure fair comparison and eliminate biases, the feature matrix is normalized. This normalization process removes differences in feature magnitudes, enabling a more reliable and accurate comparison during the subsequent classification process. So, we can extract at least 470 segments from each file. In the processed data, I have taken the first 460 segments from each file for no good reason. After extraction, our data will be of size (4600×1024). Then if we want to collect features, we can collect features taking each row of this data. For example, if we collect 8 time domain features for each segment, our feature matrix will have a shape of (4600×8). Here, we will not compute features. Rather, we will just create a preprocessed data matrix. Now if we want to compute features, we can compute it using segmented data.

we will calculate time domain features from raw data and form a feature matrix. To compute feature matrix, we will first segment our data into segments of length 2048. There is no overlap between consecutive segments. We will then compute time domain features for each segment. We have data of shape (4600x32x33). We will resize the data so that our segment length is 2048.We will calculate the following time domain features. After computing the time domain features for each segment, a feature matrix is obtained with dimensions (4600x8). The calculated features are summarized in Table 4.1, providing a clear overview of the extracted information.

Table 4.1 Time Domain Features

No	Feature	Formula
1	Mean	$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2	Absolute mean	$\text{Abs Mean} = \frac{1}{n}\sum_{i=1}^{n} \lvert x_i \rvert$
3	Maximum	$Maximum = max\,(x_i)$
4	Minimum	$Minimum = min\,(x_i)$
5	Peak to Peak	Maximum - Minimum
6	Absolute max	$Abs\ Max = max\,(\lvert x_i \rvert)$
7	Root Mean Square	$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$
8	Variance	$Var = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$
9	Clearance factor	$ClF = \frac{Absolute\ max}{\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\lvert x_i \rvert}\right)^2}$
10	Kurtosis	$\text{Kurt} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{n \times var^2} - 3$
11	Skewness	$= \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}\right)^3}$
12	Impulse Factor	$IF = \frac{Abs\ Max}{\frac{1}{n}\sum_{i=1}^{n}\lvert x_i \rvert}$
13	Crest Factor	$CF = \frac{Abs\ Max}{RMS}$
14	Shape Factor	$SF = \frac{RMS}{\frac{1}{n}\sum_{i=1}^{n}\lvert x_i \rvert}$

4.4 TIME-FREQUENCY DOMAIN ANALYSIS

Time-Frequency Domain Analysis is an essential technique in bearing fault diagnosis as it enables a comprehensive examination of the signal in both the time

and frequency domains simultaneously. By analyzing the signal's behavior over time and frequency, this technique provides a more detailed understanding of the underlying fault patterns and their manifestations[62]. Time-frequency domain analysis allows for the identification of transient events, modulations, and other dynamic features that are crucial for accurate fault diagnosis.

In Time-Frequency Domain Analysis, the Wavelet Packet Transform (WPT) is an extension of the Wavelet Transform that offers a detail comprehensive decomposition of the raw vibration data into sub-bands. Unlike the Wavelet Transform, which focuses on approximation coefficients, the WPT captures both approximation and detailed coefficients at different decomposition levels[63]. By capturing both low-frequency and high-frequency components, the WPT enables a deeper understanding of the signal's structure and can reveal subtle patterns and features that may be crucial for accurate bearing fault diagnosis. This additional decomposition offers enhanced resolution and better discrimination of fault-related features. In this research, Wavelet Packet Transform are utilized to analyze the vibration signals from the CWRU dataset. By applying these techniques, we can extract informative time-frequency features that highlight fault signatures specific to bearing faults. These features capture localized variations in the frequency content of the signals over time, offering valuable insights into the unique characteristics of each fault type. The analysis of these time-frequency features enables a more comprehensive understanding of the fault patterns and aids in accurate and reliable bearing fault diagnosis[64].

4.4.1 Wavelet Packet Transform (WPT)

Wavelet decomposition, or wavelet transform technique used for dissects a signal into various frequency components. The low-frequency band preserves the coarse-grained information, while the high-frequency band captures the fine-grained details of the signal. This decomposition provides a more nuanced representation of the signal, allowing for a more thorough analysis of its frequency content. Additionally, Wavelet Packet Transform (WPT) builds upon the wavelet transform by offering an even more extensive analysis of the signal[65]. This enhanced decomposition provides a comprehensive understanding of the signal's frequency

characteristics, facilitating more accurate and insightful analysis in various applications, including bearing fault diagnosis.

In contrast to the wavelet transform, wavelet packet decomposition offers a more detailed and comprehensive analysis of a signal's frequency components. It not only decomposes the low-frequency part but also performs further decomposition of the high-frequency part. At level-2 decomposition, the low-frequency components obtained from the initial decomposition (A1) are further decomposed into sub-bands, denoted as AA2. Similarly, the high-frequency components are decomposed into sub-bands, represented as DA2. This multi-level decomposition provides a deeper insight into the frequency content of the signal, capturing both low and high-frequency details with greater granularity[66]. By recursively applying this decomposition process, we can uncover finer frequency structures and better understand the intricate characteristics of the signal. Similar decomposition processes are applied to the high-frequency band (D1) as well. Figure 4.3 visually represents the steps involved in the dyadic decomposition process, highlighting the multi-level analysis performed by the WPT.

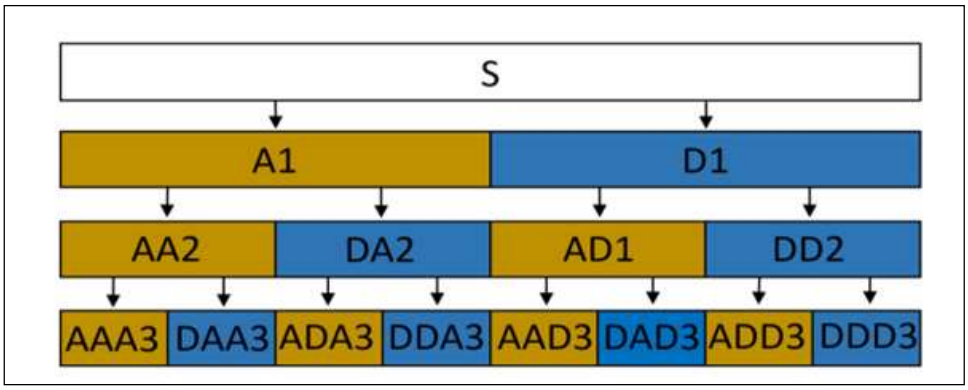

Figure 4.3 Dyadic Decomposition of the data

In contrast to the representation shown in Figure 4.3, the wavelet packet transform (WPT) convert the lower frequency band into higher frequency band by each level of decomposition. In our investigation, we focus on the decomposition at level 6, as illustrated in Figure 4.4. The highlighted red boxes in the figure indicate the specific wavelet packet components that are of interest to our analysis.

By examining the decomposition at this level, enabling us to extract valuable information relevant to our study of bearing fault diagnosis.

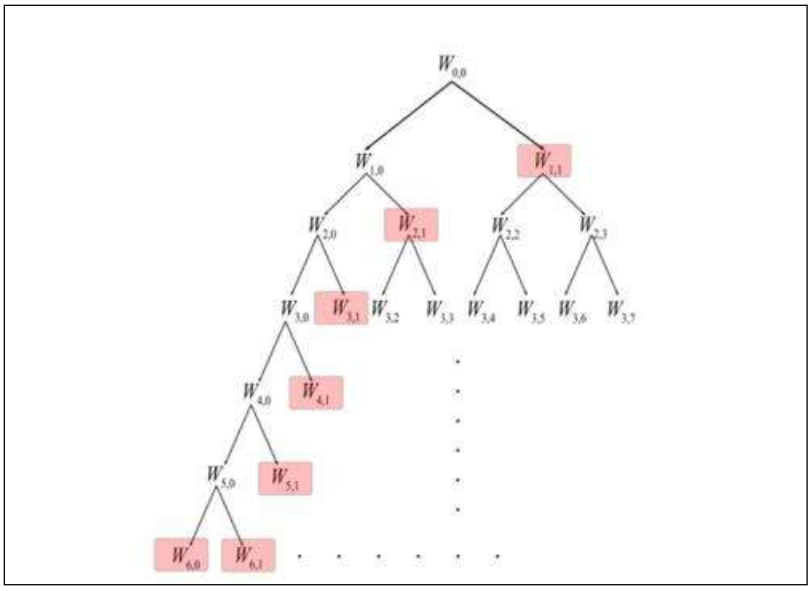

Figure 4.4 Decomposes the Lower Frequency Band

4.4.1.1 Wavelet Packet Energy

Wavelet packet energy technique is a feature extraction technique that evaluates the energy distribution of a signal within individual wavelet packet nodes. It quantifies the total energy contained in a particular frequency subband, providing insights into the signal's energy distribution across different frequency components[67]. This feature is useful for identifying specific frequency ranges that contribute significantly to the overall energy content of the signal. By analyzing the wavelet packet energy, researchers can gain a deeper understanding of the signal's frequency characteristics and identify potential fault-related patterns or anomalies.

4.4.1.2 Wavelet Packet Entropy

Wavelet packet entropy a feature that captures the complexity or irregularity of a signal within different frequency subbands. It quantifies the amount of information or unpredictability present in the signal within a specific frequency range. Higher entropy values indicate a higher degree of randomness or complexity in the signal, suggesting the presence of diverse frequency components or fault-related patterns.

The wavelet packet entropy feature is computed by applying an entropy calculation, such as Shannon entropy or Renyi entropy, to the distribution of wavelet packet coefficients within a frequency subband[67]. It quantifies the level of disorder or randomness present in the subband, reflecting the complexity of the signal in that frequency range. Higher entropy values indicate more irregular or complex behavior, which can be indicative of bearing faults.

4.4.1.3 Wavelet Packet Energy Features Calculation

To calculate the Wavelet Packet Energy features for bearing fault diagnosis in our research using the CWRU dataset, we follow these steps. Firstly, we preprocess the vibration signal data by segmenting it into suitable blocks or segments.

Next, we perform the Wavelet Packet Transform (WPT) on each segment using an appropriate wavelet basis function, such as Daubechies, Coiflets, or Symlets wavelets. The WPT decomposes the signal into different frequency subbands at the desired decomposition level, revealing frequency-specific information related to faults. We then compute the Wavelet Packet Energy for each decomposed frequency subband or wavelet packet node. This is achieved by calculating the sum of squared magnitudes of the wavelet packet coefficients within each subband, representing the energy within that specific frequency range.

To ensure consistency, we normalize the computed energy values. This normalization step brings the energy values to a standardized range, facilitating fair comparisons across different segments. Finally, we aggregate the normalized energy values from all frequency subbands, creating a feature vector for each segment or block of the vibration signal. Each feature vector represents the Wavelet Packet Energy features for that specific segment.

This process is repeated for all segments, resulting in a set of feature vectors corresponding to each segment. These feature vectors capture the energy distribution across different frequency subbands, providing valuable information for bearing fault diagnosis.

4.4.1.4 Wavelet Packet Energy Features Name

After calculating the Wavelet Packet Energy features using the method described above, the resulting feature vector will include energy values from different frequency subbands. These subbands can be labeled or named based on their corresponding wavelet packet nodes. The feature names associated with the Wavelet Packet Energy features may include:

1. Energy_Subband1: Energy of the first frequency subband.
2. Energy_Subband2: Energy of the second frequency subband.
3. Energy_Subband3: Energy of the third frequency subband.
4. Energy_Subband4: Energy of the fourth frequency subband.
5. Energy_Subband5: Energy of the fifth frequency subband.
6. Energy_Subband6: Energy of the sixth frequency subband.
7. Energy_Subband7: Energy of the seventh frequency subband.
8. Energy_Subband8: Energy of the eighth frequency subband.
9. Energy_Subband9: Energy of the ninth frequency subband.
10. Energy_Subband10: Energy of the tenth frequency subband.

These features capture the energy distribution across different frequency subbands, enabling the identification of fault-related patterns and anomalies in the vibration signals.

4.4.1.5 Wavelet Packet Entropy Features Calculation

To calculate the Wavelet Packet Entropy features for bearing fault diagnosis using the CWRU data set, several steps can be followed. Firstly, it is important to preprocess the vibration signal data by segmenting it into suitable blocks and applying preprocessing techniques such as filtering and normalization. This ensures that the data is prepared for accurate analysis. Next, the Wavelet Packet Transform (WPT) is performed on each segment using a chosen wavelet basis function. This decomposition allows for the extraction of frequency subbands at different levels. The Wavelet Packet Entropy is then estimated for each subband, which provides a measure of the complexity or information content within that specific frequency range. The Shannon entropy formula is applied, taking into account the probabilities

of occurrence of different coefficient values within the subband. By aggregating the calculated entropy values from all subbands, a feature vector is formed for each segment. This process is repeated for all segments, resulting in a set of input feature vectors representing the Wavelet Packet Entropy features for the entire dataset. These features can be further utilized in machine learning algorithms to develop models for bearing fault diagnosis and classification.

4.4.1.6 Wavelet Packet Entropy Features Name

After performing the Wavelet Packet Entropy calculations as described earlier, we obtain a set of feature values. The specific names of the features associated with Wavelet Packet Entropy may vary depending on the chosen entropy measure and decomposition level. some commonly used Wavelet Packet Entropy features include the entropy of the approximation coefficients (*Entropy_A*), which represents the disorder or randomness in the low-frequency components of the signal. Additionally, the entropy of the detail coefficients (*Entropy_D*) characterizes the irregularity in the high-frequency components. Another feature, the Energy Entropy (*Energy_Entropy*), combines the concepts of energy distribution and entropy to capture the complexity of the signal. The Shannon Entropy (*Shannon_Entropy*) provides a measure of the information content or uncertainty in the signal at a given decomposition level, while the Renyi Entropy (*Renyi_Entropy*) offers an alternative perspective on signal complexity.

These features capture different aspects of the frequency subbands' entropy distribution and can provide valuable insights into the fault-related patterns in the vibration signal. These features capture the complexity and information content of different frequency subbands, allowing for the identification of fault-related patterns and anomalies in the vibration signals.

4.5 CONCLUSION

In summary, the feature engineering chapter delved into the various methods of input feature extraction from the vibration signal of CWRU bearing dataset. It focused on two key approaches: time domain features and time-frequency domain features. The chapter offered a comprehensive understanding of the step-by-step

process for computing these features, which encompassed selecting pertinent subsets, reducing dimensionality, and ensuring the relevance of the extracted features. By providing detailed explanations and insights, the chapter served as a valuable source for researchers seeking to enhance their understanding of feature engineering in the context of signal processing of the bearing fault dataset. By leveraging these feature extraction techniques, it is possible to obtain relevant features from the raw vibration signal data and transform it into meaningful features for bearing fault classification. These features will be utilized in the subsequent chapters to develop accurate and effective fault diagnosis models.

CHAPTER 5
MACHINE LEARNING APPROACHES

5.1 INTODUCTION

The machine learning approaches chapter introduces various algorithms for bearing fault classification. We cover linear algorithms such as Multiclass Logistic Regression, LDA, and QDA, as well as nonlinear algorithms like KNN, SVM, and DT. Additionally, we explore ensemble tree algorithms such as Bagging, Boosting, and Random Forest. This chapter provides a concise overview of these algorithms and their application to bearing fault classification, setting the stage for further analysis on the CWRU bearing dataset.

5.2 MACHINE LEARNING BASED FAULT CLASSIFICATION

ML-based fault detection and classification techniques have become essential in bearing fault diagnosis, especially for the CWRU dataset. These techniques used to automatically learn patterns and identify faults in vibration signals. The overall process involves several key steps. Firstly, data preprocessing is performed to remove noise, filter out unwanted frequencies, and ensure data normalization. The data is then segmented into smaller windows to facilitate analysis. Next, relevant input features are extracted through the segmented input data using various features techniques including time domain analysis, frequency domain analysis, wavelet analysis, or statistical analysis[68]. These techniques capture crucial characteristics of the vibration signal that are indicative of different fault types.

Feature extraction techniques are utilized to find the most informative and discriminative input features, reducing dimensionality and enhancing the efficiency of ML algorithms. The labeled data from the CWRU dataset is split into training and validation data sets, with the training dataset applied to train the ML algorithms. Different ML models such as Support Vector Machines (SVM),

Decision Tree and Random Forests are trained on the labeled training data, understanding the underlying relationships between the input features and fault types[69]. The trained model is estimated using the validation set to evaluate its performance, with performance metrics such as accuracy, precision, recall, F1 score, or confusion matrix used to measure effectiveness. The model is fine-tuned by adjusting hyperparameters or using cross-validation techniques. Once the model is trained and validated, it can be deployed for fault detection and classification on new, unseen vibration signal data. The model predicts the fault type based on the extracted features.

Machine Learning based fault classification techniques have revolutionized the area of bearing fault classification, offering automated and accurate solutions that bring significant benefits to industrial applications.

5.3 MACHINE LEARNING MODEL WORKING

Machine learning models work by learning patterns and relationships from input data to predict the desire output. The general working process of a machine learning model can be explained using the following figure 5.1.

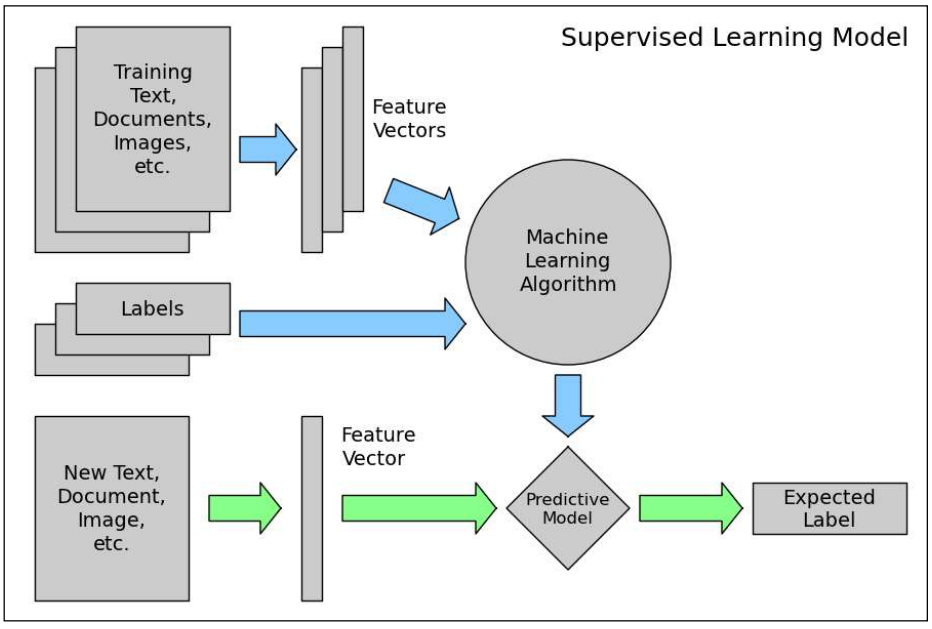

Figure 5.1 Working of Machine Learning Model

- Training Data: Relevant data is collected from diverse sources and used for training the untrain machine learning model.
- Data Preprocessing: This step handle missing values, outliers, and inconsistencies in collected dataset through the data cleaning and normalizing. it scaling features and encoding categorical variables, ensuring the data is suitable for model training.
- Feature Extraction: The preprocessed data is transformed into meaningful features that capture essential information for the learning task. This process aids in representing the data more effectively and compactly.
- Model Training: Classifier model is trained using the preprocessed and feature-extracted data. Through this, the model learns relationships, and dependencies through an optimization algorithm that adjusts its internal parameters. The aim is to minimize the difference between the model's predictions output and the true values in the training data.
- Model Evaluation: The trained model is evaluated using split test dataset. This evaluation assesses the model's performance and its ability to generalize to unseen new input data. Performance metrics including accuracy, precision, recall, or F1-score are commonly used to evaluate the model's effectiveness.
- Hyperparameter Tuning: If the model's performance is unsatisfactory, hyperparameter tuning can be perform to tune the model's configuration. Hyperparameters are adjusted to improve the model's performance and improve its generalization capabilities.
- Final Trained Model: After training, evaluation, and potential hyperparameter tuning, the final trained model is obtained, equipped with optimized parameters.
- Prediction/Inference: The final trained model is employed to make predictions on unseen new input data.

The provided figure showcases the step-by-step flow of the machine learning model's working, encompassing data collection, data preprocessing, input feature

extraction, model training, model evaluation, hyperparameter tuning, and the utilization of the trained model for predictions or decision-making on new data.

5.4 LINEAR MACHINE LEARNING ALGORITHMS

Linear machine learning algorithms are widely employed in bearing fault classification using the CWRU dataset for classification tasks. These algorithms are designed to classify different types of bearing faults based on the provided input features. The underlying principle of linear classification algorithms revolves around establishing a linear decision boundary that effectively separates data points belonging to distinct classes. The main objective is to obtain the optimal coefficients or weights that minimize the discrepancy between predicted and actual class labels. Several linear machine learning classifiers are commonly employed for bearing fault classification, including Multiclass Logistic Regression, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)[70]. The utilization of these linear machine learning algorithms facilitates the accurate classification of bearing faults, aiding in the identification and diagnosis of various fault types based on the provided input features.

5.4.1 Multiclass Logistic Regression

Multiclass Logistic Regression classifier is a well-established linear classification algorithm widely applied in multiclass classification tasks. It builds upon the principles of binary logistic regression and extends its functionality to handle scenarios involving multiple classes. By employing a softmax function, the classifier estimates the probabilities for each class and generates a distribution across all classes[71]. This enables effective decision-making by assigning the instance to the class that exhibits the highest probability.

In Multiclass Logistic Regression, we have a dataset with N instances and K classes. Each input instance is represented by a input feature vector (x), and we want to predict its class label (y). The model learns a set of parameters (W), where each column corresponds to a specific class. The hypothesis function for Multiclass Logistic Regression can be defined as:

$$h(x) = softmax(Wx)$$

Here, W is the parameter matrix, x is the input feature vector, and softmax is the softmax activation function that computes the probabilities for each class. Mathematically the softmax function is defined as:

$$softmax(z_i) = (e_i^z) / \left(\sum_{j=1}^{K} e_j^z \right)$$

During the training phase, the model is trained to find the optimal values for the parameter matrix W. This is done by minimizing a cost function, often the cross-entropy loss, which calculate the discrepancy between the predicted probabilities and the true class labels. The cost function for Multiclass Logistic Regression can be defined as:

$$J(W) = -1/N * \sum_{i=1}^{N} \left[\sum_{j=1}^{K} y_{ij} * log\big(h(x_i)\big) + \big(1 - y_{ij}\big) * log\big(1 - h(x_i)\big) \right]$$

where N is the number of instances, y_{ij} is a one-hot encoded vector representing the true class label, and $h(x_i)$ is the predicted probability vector for all classes.
The model parameters W are usually optimized using gradient-based optimization algorithms, such as gradient descent or its variants. The gradient of the cost function with respect to W is computed, and the parameters are updated iteratively until convergence. During the prediction phase, given a new instance x, the Multiclass Logistic Regression model calculates the probabilities for each class using the learned parameters W and assigns the instance to the class with the highest probability[72].

Multiclass Logistic Regression can be effective in scenarios where the classes are linearly separable, it may encounter limitations when faced with non-linearly separable classes or complex non-linear relationships within the data.

5.4.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a powerful technique employed in machine learning for both dimensionality reduction and classification tasks. It seeks to identify a linear combination of features that optimally discriminates between

different classes within a dataset. By maximizing the ratio of between-class scatter to within-class scatter, LDA effectively captures the variations that are most relevant for distinguishing one class from another[73].

Let's consider a dataset with (N) samples and (D) features, where each sample belongs to one of (K) classes. The goal of LDA is to project the data onto a lower-dimensional subspace while maximizing the separability between classes. The mathematical steps of LDA are as follows:

Calculate the class-wise mean vectors: Compute the mean vector for each class by averaging the feature vectors of samples within that class. Denote the mean vector for class (k) as (μ_k).

$$\mu_k = (1/n_k) \sum_{i=1}^{n_k} x_i$$

Calculate the within-class scatter matrix (Sw): Calculate the scatter matrix for each class (k) by subtracting the mean vector of that class from each sample in the class, multiplying the result by its transpose, and summing them up. Denote the scatter matrix for class (k) as (S_k).

$$S_k = \sum_{i=1}^{n_k} (x_i - \mu_k)(x_i - \mu_k)^T$$

Compute the within-class scatter matrix by summing up all the scatter matrices for each class: (Sw).

$$Sw = \sum_{k=1}^{K} S_k$$

Calculate the between-class scatter matrix (Sb): Compute the overall mean vector (μ) by averaging the class-wise mean vectors:

$$\mu = (1/K) \sum_{k=1}^{K} \mu_k$$

Calculate the scatter matrix for each class by subtracting the overall mean vector from each class mean vector, multiplying the result by its transpose, and summing them up:

$$Sb = \sum_{k=1}^{K} (\mu_k - \mu)(\mu_k - \mu)^T$$

Compute the eigenvalues and eigenvectors: Solve the generalized eigenvalue problem:

$$Sw^{(-1)}Sbw = \lambda w$$

where (w) is the eigenvector and (λ) is the eigenvalue.

Project the data onto the lower-dimensional subspace: Multiply the original feature vectors by the selected eigenvectors:

$$Y = XW$$

where (X) is the original data matrix, and (W) is the matrix of selected eigenvectors.

Classification: Use a suitable classifier, such as a linear classifier or a nearest neighbor classifier, on the projected data (Y) to classify new samples.

5.4.3 Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a classification algorithm that extends the capabilities of Linear Discriminant Analysis (LDA). By considering class-specific covariance matrices, QDA can capture more complex relationships and variations within each class. To classify new observations, QDA calculates the likelihood of an observation belonging to each class based on its feature values and the class-specific covariance matrices[74]. It then assigns the observation to the class with the highest likelihood. This flexibility in modeling the covariance matrices enhances the discriminative power of QDA, making it particularly useful when the classes have different covariance structures. Mathematically, the QDA model can be described as follows:

Given a training dataset with N samples and K classes, let X = {x1, x2, ..., xN} be the feature vectors and y = {y1, y2, ..., yN} be the corresponding class labels. For each class k (k = 1, 2, ..., K), calculate the class-specific mean vector μk

and covariance matrix Σk. Calculate the prior probability of each class $P(y = k|x)$ as the proportion of samples in the training data belonging to class k. For a new observation x, calculate the posterior probability of belonging to class k using Bayes' theorem:

$$P(y = k|x) = \big(P(x|y = k) * P(y = k)\big)/P(x)$$

where $P(x|y = k)$ is the likelihood of x given class k, $P(y = k)$ is the prior probability of class k, and $P(x)$ is the marginal probability of x.

Assign the observation x to the class with the highest posterior probability. QDA assumes that the likelihood follows a multivariate Gaussian distribution for each class:

$$P(x|y = k) = \left(1/\left(sqrt\Big((2 * \pi)^d * det(\Sigma k)\Big)\right)\right) * exp\big(-0.5 * (x - \mu k)^T * \Sigma k^{-}1 * (x - \mu k)\big)$$

where d is the dimensionality of the feature vector x, π is the constant pi, and det (Σk) is the determinant of the covariance matrix Σk.

In Quadratic Discriminant Analysis (QDA), the parameters such as mean vectors and covariance matrices are estimated from the training data. These parameters are then utilized to calculate the posterior probabilities for classification.

5.5 NONLINEAR MACHINE LEARNING ALGORITHMS

Nonlinear machine learning algorithms play a crucial role in bearing fault diagnosis and classification, especially when dealing with complex patterns in the CWRU data set. These algorithms excel at capturing intricate relationships by employing nonlinear decision boundaries. In this context, we will discuss three popular nonlinear machine learning classifiers: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Tree Algorithm.

5.5.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a versatile and intuitive classification algorithm that relies on the concept of similarity among data points in the feature space. Rather

than assuming a specific distribution or parametric form, KNN considers the proximity of instances to make predictions. Mathematically, the KNN algorithm can be described as follows:

Given a labeled training dataset consisting of feature vectors X and corresponding class labels y, where $X = \{x_1, x_2, ..., xn\}$ and $y = \{y_1, y_2, ..., yn\}$:

For a given test instance *x*, calculate its similarity or distance to all instances in the training set. Common distance metrics used include Euclidean distance, Manhattan distance, or cosine similarity. Select the *K* nearest neighbors (K is a user-defined parameter) based on the calculated distances. These are the K instances in the training set that are most similar to the test instance *x*. Assign the class label to the test instance x based on the majority class among its K nearest neighbors. To predict the class label of a given test instance using the K-Nearest Neighbors (KNN) algorithm, the following steps are performed:

- Compute the distance or similarity measure between the test instance and each instance in the training set. Common distance metrics include Euclidean distance, Manhattan distance, or cosine similarity.
- Identify the K nearest neighbors by selecting the instances with the smallest distances to the test instance. K is a user-defined parameter that determines the number of neighbors to consider.
- Determine the class label of the test instance by taking the majority vote among the K nearest neighbors.

5.5.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a powerful supervised machine learning algorithm utilized for classification and regression tasks. The fundamental principle of SVM is to determine an optimal hyperplane that effectively separates the data points belonging to different classes, while maximizing the margin between these classes. The hyperplane acts as a decision boundary, ensuring that instances from different classes are properly classified. SVM can handle linearly separable data as well as data with non-linear relationships through the use of kernel functions, which map the data to higher-dimensional feature spaces[75]. By finding the optimal hyperplane, SVM aims to achieve better generalization and robust classification performance.

In the case of binary classification, given a training dataset consisting of input vectors x_i and their corresponding class labels y_i, where $x_i \in \mathbb{R}$^d and $y_i \in$ {-1, +1}, SVM aims to find a hyperplane defined by the equation:

$$w^T x + b = 0$$

where w is the weight vector perpendicular to the hyperplane, and b is the bias term. The goal is to find the optimal values of w and b that maximize the margin between the hyperplane and the closest data points of the two classes.

To mathematically represent this optimization problem, we introduce the concept of a margin. The margin is the distance between the hyperplane and the closest data points, and we aim to find the hyperplane with the maximum margin. The margin can be calculated as:

$$margin = 2/||w||$$

where $||w||$ is the Euclidean norm of the weight vector w. Maximizing the margin is equivalent to minimizing $||w||$, subject to the following constraints:

$$y_i(w^T x_i + b) \geq 1$$

for all training examples (x_i, y_i). These constraints ensure that all data points are correctly classified and lie outside the margin region.

However, in many cases, it is not possible to find a hyperplane that perfectly separates the data. To handle such cases, SVM introduces the concept of soft margin. Soft margin allows for a certain degree of misclassification by introducing slack variables $\Sigma\xi_i \geq 0$, which measure the degree of misclassification of each data point. The optimization problem for soft margin SVM can be formulated as follows:

$$minimize(1/2)||w||^2 + C\Sigma\xi_i$$

$$subjectto: y_i(w^T x_i + b) \geq 1 - \xi_i$$

where C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the misclassification.

To address the challenges posed by non-linearly separable data, Support Vector Machines (SVM) employ the kernel trick. This technique allows SVM to implicitly map the input vectors to a higher-dimensional feature space, where the data becomes linearly separable[76]. By applying a kernel function such as linear, polynomial, radial basis function (RBF), or sigmoid, SVM can transform the input

data into a new space, effectively capturing complex relationships and enabling the discovery of non-linear decision boundaries.

5.5.3 Decision Tree Algorithm

The Decision Tree Algorithm is a supervised learning technique that is widely used for classification and regression tasks. It constructs a tree-like structure, called a decision tree, where each internal node represents a feature or attribute, each branch represents a decision rule based on the feature values, and each leaf node represents a class label or an outcome[77]. The algorithm builds the decision tree by recursively partitioning the data based on the selected features and their values. Mathematically, the decision tree algorithm can be described as follows:

Given a training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi represents the feature values and yi represents the corresponding class labels.

The choice of the best feature to split the data is determined by evaluating various criteria such as information gain or Gini index. The process continues until a stopping condition is met, such as reaching a maximum depth or when further splitting does not significantly improve the classification accuracy. Each leaf node is assigned a class label based on the majority class of the samples in that node. The resulting decision tree provides a clear set of hierarchical rules that can be used to classify new instances by traversing the tree. The decision tree algorithm is advantageous due to its interpretability, ability to handle both categorical and numerical features, and capacity to capture non-linear relationships between features and class labels[77]. However, overfitting can occur when the tree becomes too deep or complex, which can be mitigated through techniques such as pruning or setting a maximum depth. Ensemble methods like random forests can also enhance the generalization performance of decision trees.

5.6 ENSEMBLE ALGORITHMS

Ensemble algorithms offer powerful techniques for improving the accuracy and reliability of bearing fault diagnosis and classification in the CWRU data set. Two commonly used ensemble algorithms are Bagging and Random Forests, as well as Boosting.

5.6.1 Bagging Algorithm

Bagging, also known as Bootstrap Aggregating, is a powerful ensemble learning algorithm that enhances prediction accuracy and reduces variance. Its working principle involves training multiple models on different bootstrap samples, which are subsets of the original training data created through random sampling with replacement. Each model is trained independently, and their predictions are combined through aggregation methods like majority voting (for classification) or averaging (for regression)[78]. By leveraging the diversity of these models, Bagging effectively reduces the impact of outliers and noise in the data, leading to more robust and accurate predictions.

Given a training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi represents the input features and yi represents the corresponding output labels, the Bagging algorithm can be summarized as follows:

- Random Sampling with Replacement: For each model in the ensemble: Randomly sample n instances from D with replacement to create a new bootstrap sample, D' = {(x1', y1'), (x2', y2'), ..., (xn', yn')}. Each bootstrap sample D is of the same size as the original dataset, but some instances may be duplicated while others may be left out.
- Independent Model Training: Train a base model, such as a decision tree or logistic regression, on each bootstrap sample D. Each model is trained independently of the others, without any knowledge of the other models' predictions.
- Aggregation of Predictions: For a new instance x, obtain the prediction from each model in the ensemble.

Mathematically, the Bagging algorithm involves aggregating the predictions of the individual models. For binary classification problems, the aggregation can be done using majority voting:

$$P(y = 1|x) = argmax\big(\Sigma m\ \ (y_h at_m = 1)\big)$$

where $P(y = 1|x)$ is the predicted probability of class 1 for a given instance x, $y_h at_m$ is the prediction of the m-th model, and $I(y_h at_m = 1)$ is an indicator function that returns 1 if $y_h at_m = 1$ and 0 otherwise.

Bagging is an effective technique that enhances the accuracy and stability of models by mitigating the issue of overfitting.

5.6.2 Random Forests algorithm

The Random Forests algorithm is a powerful ensemble learning method that combines multiple decision trees to make accurate predictions. Each decision tree is built using a different subset of the training data and a random subset of features, introducing variation and diversity into the ensemble[79]. This diversity helps to reduce overfitting and increase the overall prediction accuracy. Mathematically, the Random Forests algorithm can be described as follows:

- Training Phase: Let X be the training data matrix of size (m x n), where m is the number of instances and n is the number of features. Let Y be the corresponding vector of target labels of size (m x 1). Let T be the number of decision trees to be included in the random forest. For each tree t = 1 to T: Randomly select a subset of the training data with replacement, creating a bootstrap sample. Randomly select a subset of features (typically sqrt(n) or log2(n) features) for building the decision tree. Construct a decision tree using the bootstrap sample and selected features.
- Prediction Phase: For a new instance x, which has the feature values (x1, x2, ..., xn): For each decision tree in the random forest: Traverse the tree by following the decision rules based on the selected features. Assign the majority class of the leaf node reached as the prediction. Aggregate the predictions from all decision trees, either by majority voting (for classification) or averaging (for regression).

The Random Forests algorithm takes advantage of the randomness in both the data selection and feature selection to create a diverse set of decision trees. By combining the predictions of multiple trees, it reduces the bias and variance of individual trees, leading to improved accuracy and robustness in classification tasks.

5.6.3 Boosting ensemble

Boosting ensemble algorithms are widely used in machine learning to enhance classification accuracy by combining multiple weak classifiers. AdaBoost and Gradient Boosting are two popular Boosting algorithms known for their effectiveness in handling complex classification tasks.

5.6.3.1 AdaBoost

AdaBoost, also known as Adaptive Boosting, is a popular binary classification algorithm that leverages the power of combining multiple weak classifiers to create a strong ensemble model. The algorithm iteratively trains weak classifiers on different subsets of the training data, with a particular focus on samples that were misclassified in previous iterations. This adaptive training process allows AdaBoost to continuously improve its performance by giving more weight to the challenging instances[78]. The final prediction is made by aggregating the predictions of all weak classifiers, where the weight of each classifier's prediction is determined by its performance. By utilizing the collective knowledge of the ensemble, AdaBoost can effectively classify binary problems with high accuracy.

Algorithm: Initialize the weights: Assign equal weights to all training samples, w_i =1 / n, where n is the number of samples.

For t = 1 to T, where T is the number of iterations: Train a weak classifier $h_t(x_i)$ on the weighted training data, minimizing the weighted error ε_t:

$$\varepsilon_t = \sum\left(w_i * (h_t(x_i)! = y_i)\right) / \sum w_i$$

Compute the weight for the weak classifier α_t:

$$\alpha_t = 0.5 * ln\left((1 - \varepsilon_t)/\varepsilon_t\right)$$

Update the sample weights:

$$w_i = w_i * exp\left(-\alpha_t * y_i * h_t(x_i)\right)/Z_t$$

where y_i is the true class label, $h_t(x_i)$ is the weak classifier's prediction, and Z_t is a normalization factor to ensure the weights sum to 1.

The final prediction is made by combining the predictions of all weak classifiers:

$$H(x) = sign\left(\sum\left(\alpha_t * h_t(x)\right)\right)$$

AdaBoost is an iterative algorithm that trains a series of weak classifiers and assigns weights to each classifier based on its performance. The iterative process emphasizes misclassified samples, enabling the algorithm to progressively enhance its classification accuracy. The final prediction is determined by combining the outputs of the weak classifiers, with the weights of each classifier reflecting its performance. The algorithm places greater emphasis on the classifiers that exhibit better performance, resulting in a more accurate overall prediction.

5.6.3.2 Gradient Boosting algorithm

Gradient Boosting is an ensemble learning algorithm that leverages the power of weak learners, such as decision trees, to build a strong predictive model. Unlike AdaBoost, which assigns weights to classifiers based on their performance, Gradient Boosting focuses on minimizing the loss function by iteratively optimizing the residuals of the previous model[76]. Each subsequent model is trained to correct the errors made by the previous models, gradually improving the overall performance of the ensemble. By iteratively refining the predictions, Gradient Boosting can effectively capture complex relationships in the data and provide accurate predictions.

Algorithm: Initialize the prediction values for all samples:

$$F_0(x) = initial_guess$$

For t = 1 to T, where T is the number of iterations: Compute the negative gradient of the loss function:

$$r_t = -\left[\partial L\left(y, F_{(t-1)}(x)\right) / \partial F_{(t-1)}(x)\right]$$

where y is the true label and $F_{(t-1)}(x)$ is the prediction of the previous model.

Train a weak learner $h_t(x)$ on the negative gradient:

$$h_t(x) = argmin_h \sum \left(L\left(y_i, F_{(t-1)}(x_i) + h(x_i)\right)\right)$$

Determine the optimal step size or learning rate for the weak learner:

$$\gamma_t = argmin_\gamma \sum \left(L\left(y_i, F_{(t-1)}(x_i) + \gamma * h_t(x_i)\right)\right)$$

Update the prediction values for all samples:

$$F_t(x) = F_{(t-1)}(x) + \gamma_t * h_t(x)$$

The final prediction is given by the ensemble of weak learners:

$$F(x) = \sum \left(F_T(x)\right)$$

Gradient Boosting is an iterative machine learning algorithm that combines weak learners, typically decision trees, to create a powerful ensemble model. In each iteration, the algorithm trains a weak learner on the negative gradient of the loss function with respect to the previous model's prediction. This strategy allows the algorithm to focus on the instances where the previous model made significant errors. The learning rate, or step size, determines the contribution of each weak learner to the final prediction. By updating the prediction values with the product of the step size and the weak learner's prediction, the algorithm gradually improves the overall prediction accuracy. The process continues for a specified number of iterations, refining the ensemble model and capturing intricate relationships in the data. The final prediction is obtained by summing the predictions of all weak learners, resulting in a strong ensemble model that can effectively handle complex patterns in the data.

5.7 CONCLUSION

In this chapter, we explored different machine learning algorithms using the CWRU dataset. Linear and non-linear algorithms, as well as ensemble methods, were evaluated. Machine learning algorithms showed promise in accurately classifying bearing faults and aiding in condition monitoring. However, linear algorithms may struggle with non-linear relationships, while non-linear algorithms require careful parameter tuning. Ensemble methods improve classification performance by combining models. Limitations include handling non-linearity, manual feature engineering, and the curse of dimensionality. Deep learning algorithms offer solutions by capturing non-linear relationships, automatic feature learning, and dimensionality reduction. Overall, the research highlights the potential of various machine learning algorithms while recognizing the need for deep learning approaches to overcome limitations and improve bearing fault diagnosis.

CHAPTER 6
DEEP LEARNING APPROACHES

6.1 INTODUCTION

The deep learning approaches chapter in bearing fault classification explores the application of various deep learning models, including Artificial Neural Networks (ANN), 1D Convolutional Neural Networks (CNN1D), 2D Convolutional Neural Networks (CNN2D), Long Short-Term Memory (LSTM), and CNN-LSTM. Understanding the principles and architectures of these deep learning models can significantly enhance the accuracy and effectiveness of bearing fault classification. Researchers and practitioners can leverage the capabilities of these models to advance fault diagnosis systems and elevate the overall performance of bearing fault classification.

6.2 DEEP LEARNING BASED FAULT CLASSIFICATION

Deep learning models, including Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and CNN-LSTM, have emerged as effective solutions for bearing fault classification tasks. These deep learning models offer powerful tools for bearing fault classification by directly learning complex patterns from raw signal data. Through careful selection and optimization of model architecture and parameters, researchers and practitioners can achieve precise fault diagnosis, leading to improved maintenance strategies and operational efficiency across various industries.

6.3 DEEP LEARNING MODEL WORKING

Deep learning models, a subset of artificial neural networks (ANNs), excel at learning and making predictions from intricate and high-dimensional data. These

models possess the capability to autonomously learn hierarchical representations, enabling them to extract significant features and patterns essential for classification tasks. The following outlines the high-level theory behind deep learning models for classification:

- Neural Network Architecture: Deep learning models encompass multiple layers of interconnected nodes, referred to as neurons. Comprising an input layer, one or more hidden layers, and an output layer, the architecture facilitates the flow of information throughout the network[39].
- Feedforward Propagation: Input data is fed into the neural network via the input layer, proceeding through the layers with each layer applying a non-linear activation function to the previous layer's outputs. This process, known as feedforward propagation, propels the information forward, progressively transforming and extracting higher-level representations[39].
- Backpropagation: Backpropagation, a pivotal algorithm, calculates gradients of the loss function with respect to the network's weights and biases. It efficiently propagates error gradients from the output layer backward to earlier layers, facilitating weight adjustments based on each neuron's error contribution. This iterative process of forward propagation and backpropagation continues until the network converges upon an optimal set of weights[23].

Deep learning models, encompassing Artificial neural networks, convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and their combinations, leverage these principles to autonomously learn intricate representations and patterns from input data. These models have emerged as potent tools for diverse classification tasks.

6.3.1 Training And Testing Process of Deep Learning Models

The training and testing of a deep learning model encompass several key steps:
Data Preparation: Preprocess the dataset, handling missing values, normalizing features, and splitting it into training and testing sets.

- Model Architecture Design: Select an appropriate deep learning architecture (e.g., CNN, RNN) and define the number of layers, nodes per layer, and activation functions.
- Model Compilation: Compile the model by specifying the loss function, optimization algorithm, and evaluation metrics, tailored to the specific task at hand.
- Model Training: Train the model using the training data, iteratively updating weights and biases through backpropagation and gradient descent.
- Model Evaluation: Assess the model's performance using the testing dataset, calculating evaluation metrics (e.g., accuracy, precision, recall, F1 score) to gauge its generalization capabilities.
- Hyperparameter Tuning: Experiment with various hyperparameter values (e.g., learning rate, batch size, regularization techniques) to optimize the model's performance.
- Model Deployment: Deploy the trained model to make predictions on new, unseen data, using the inference process to obtain output predictions.

This figure 6.1 visually represents the sequential flow of steps involved in training and testing a deep learning model. The entire process ensures data preparation, model architecture design, model compilation, model training, model evaluation, hyperparameter tuning, and model deployment are executed seamlessly and effectively.

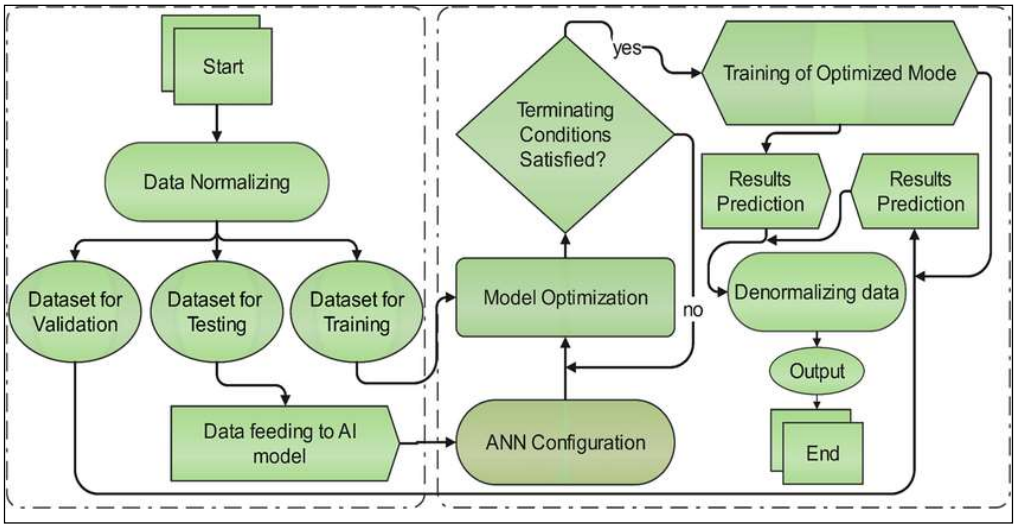

Figure 6.1 Process Flow of Deep Learning Model

6.4 DEEP LEARNING ALGORITHMS FOR FAULT CLASSIFICATION

6.4.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational models. These networks are composed of interconnected nodes, known as artificial neurons or units, organized in layers. Each neuron receives inputs, performs computations using activation functions, and generates an output. The connections between neurons, which are represented by adjustable weights, determine the contribution of each input signal. By adjusting these weights during the training process, ANNs can learn to recognize patterns, make predictions, and solve complex problems[80]. The mathematical formulation of an Artificial Neural Network can be represented as follows:

Let's consider a simple feedforward neural network with multiple layers: an input layer, one or more hidden layers, and an output layer. Each layer consists of multiple neurons.

Input:

Let $x = [x1, x2, \dots, xn]$ be the input vector to the neuron.

Let $w = [w1, w2, \dots, wn]$ be the weight vector associated with the inputs.

Let b be the bias term.

Output:

The output of the neuron, denoted as y, is computed using an activation function f

The computation of the neuron can be summarized as follows:

$$z = w \cdot x + b$$

$$y = f(z)$$

Here, z represents the weighted sum of the inputs plus the bias term, and f denotes the activation function applied to the weighted sum.

For the hidden layers and the output layer of the neural network, the computation is similar, but it involves multiple neurons and matrix operations.

Let's consider a neural network with l layers. The computation for the l-th layer can be represented as:

Input:

Let $X(l-1)$be the input matrix to the l-th layer, where each row corresponds to an input sample.

Let $W(l)$be the weight matrix associated with the connections between the (l -1)-th and l -th layers.

Let $b(l)$ be the bias vector for the l -th layer.

Output:

The output matrix of the l -th layer, denoted as $x(l)$ is computed using an activation function f.

The computation of the l -th layer can be summarized as follows:

$$Z(l) = X(l-1) \cdot W(l) + b(l)$$

$$X(l) = f\big(Z(l)\big)$$

Here, $Z(l)$ represents the weighted sum of the inputs plus the bias term, and f denotes the activation function applied element-wise to the weighted sum.

Throughout the training phase, the neural network's weights and biases undergo continuous adjustment using optimization algorithms like backpropagation and gradient descent. The primary goal is to minimize a designated loss function that measures the dissimilarity between the network's predicted outputs and the actual labels. Through iterative updates driven by calculated gradients, the neural network gradually learns to approximate intricate input-output relationships. This adaptive learning process empowers the network to effectively classify bearing faults based on the input vibration data it receives.

6.4.1.1 ANN Model Summary for bearing fault classification:

The ANN model used for bearing fault classification consists of three layers: two dense layers and one output layer. The first dense layer has 128 neurons, the second dense layer has 64 neurons, and the output layer has 10 neurons representing the classes. The model has a total of 109,386 trainable parameters, with 100,480 parameters in the first dense layer, 8,256 parameters in the second dense layer, and 650 parameters in the output layer. These parameters can be adjusted during the training process to optimize the model's performance. The model summary provides a detailed overview of the architecture, including the layer types, output shapes, and

the number of trainable and non-trainable parameters. Figure 6.2 visualizes the model summary, giving a clear representation of the ANN architecture.

```
Model: "ANN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
```

Figure 6.2 ANN Model Summary

6.4.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a popular class of deep learning algorithms commonly used for image classification tasks. CNNs leverage the concept of convolution, which involves applying filters or kernels to input images to extract relevant features. Each filter performs a convolution operation by sliding across the input image and calculating the dot product between the filter weights and the corresponding image patch[81]. This process generates a feature map that highlights the presence of specific features or patterns in the input image. The convolutional layer is a fundamental component of CNNs, composed of multiple filters that collectively learn and detect different features in the input data. This allows CNNs to automatically learn and extract meaningful representations from images, enabling accurate classification and analysis. Mathematically, the convolution operation can be represented as:

$$(I * K)(i,j) = \Sigma\Sigma I(m,n) * K(i-m, j-n)$$

where I is the input image, K is the filter, and (i,j) represents the spatial location.

After the convolutional layer, a pooling layer is often added to CNNs to downsample the feature maps and decrease their spatial dimensions. This downsampling operation helps to abstract and generalize the learned features. Mathematically, max pooling can be represented as:

$$MaxPooling(X)(i,j) = max\big(X(i+s, j+t)\big)$$

where X represents the feature map.

The fully connected layer in a CNN plays a crucial role in the classification task by leveraging the learned features from previous layers. The fully connected layer consists of a set of neurons, each connected to every neuron in the previous layer. These connections are associated with trainable weights and biases[82]. During the forward pass, the input data is multiplied by the weight matrices and added with the biases, followed by the application of non-linear activation functions.

Mathematically, the linear transformation can be represented as:

$$Y = XW + b$$

where X is the input vector, W represents the weight matrix, and b is the bias vector.

Activation functions are essential components of CNNs as they introduce non-linearity, enabling the network to learn complex patterns and relationships in the data. ReLU (Rectified Linear Unit) is a popular activation function that outputs the maximum between zero and the input value, effectively preserving positive values while filtering out negative values.

Mathematically, ReLU activation can be represented as:

$$ReLU(x) = max(0, x)$$

where x is the input.

CNNs have revolutionized the field of bearing fault classification by leveraging their hierarchical structure and specialized layers. By employing convolutional layers, CNNs are able to extract relevant features from vibration signals, capturing fault-related patterns effectively.

6.4.3 Convolutional Neural Networks (CNNs) 1D

CNNs 1D are deep learning models tailored to handle one-dimensional sequential data like time series or signals. Their strength lies in capturing local patterns and dependencies within the input. Let's delve into the theory of CNNs 1D and the corresponding mathematics.

Input Data Representation: In bearing fault classification, the vibration signals are represented as one-dimensional time series data, denoted as

$$X = [x_1, x_2, \dots, x_n]$$

where x_n represents the vibration amplitude at time step n.

Convolution Operation: The convolution operation in CNNs 1D involves applying a set of filters (kernels) to the input signal X. Each filter has a weight matrix W and a bias term b. The convolution operation at each step is given by:

$$(X * W + b)_i = \sum_{j=0}^{m-1} \left(x_{i+j} \cdot w_j\right) + b$$

where $(X * W + b)_i$ is the output at position i, x_{i+j} represents the input value at position $i + j$, w_j is the jth element of the weight matrix, and m is the filter size.

Activation Function: After the convolution operation, an activation function, such as ReLU (Rectified Linear Unit), is applied element-wise to introduce non-linearity. The ReLU activation function is defined as:

$$f(x) = \max(0, x)$$

where x is the input

Pooling layers play a crucial role in downsampling the feature maps generated by the convolutional layers in CNNs 1D[83].

To prepare the feature maps for classification, they are flattened into a one-dimensional vector, maintaining their spatial relationships. Multi-class classification typically involves the use of a softmax activation function in the final layer.

During training, a loss function, such as categorical cross-entropy, is employed to quantify the discrepancy between the predicted and actual labels. Optimization algorithms, like stochastic gradient descent (SGD) or Adam, iteratively update the network's weights and biases to minimize the loss.

6.4.3.1 CNN-1D Model Summary for Bearing Fault Classification

The CNN-1D model is a 14-layer convolutional neural network specifically designed for processing one-dimensional sequential data like time series signals. It consists of Conv1D layers with 16 filters, followed by MaxPooling1D layers for downsampling. Additional Conv1D and MaxPooling1D layers with increasing filter numbers further reduce spatial dimensions[84]. A Flatten layer converts feature maps into a one-dimensional vector. The model also includes Dense layers for classification, and a Softmax layer for probability estimation. With 115,004 trainable parameters, the model's architecture enables it to learn and optimize its performance. The model summary provides a comprehensive breakdown of the layer types, output shapes, and trainable parameters. Figure 6.3 illustrates the model summary for the CNN-1D model.

```
Model: "CNN-1D"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 840, 16)           64
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 420, 16)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 209, 32)           1568
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 104, 32)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 51, 64)            6208
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 25, 64)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 12, 128)           24704
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 6, 128)            0
_________________________________________________________________
flatten (Flatten)            (None, 768)               0
_________________________________________________________________
input_layer (InputLayer)     multiple                  0
_________________________________________________________________
dense (Dense)                (None, 100)               76900
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
softmax (Softmax)            (None, 10)                0
=================================================================
Total params: 115,004
Trainable params: 115,004
Non-trainable params: 0
```

Figure 6.3 CNN 1D Model Summary

6.4.4 Convolutional Neural Networks (CNNs) 2D

CNNs, also called ConvNets, have significantly transformed computer vision tasks by enabling the analysis of two-dimensional data, such as images, in deep learning models.

The core operation in CNNs is the convolution, which plays a crucial role in extracting local features from the input image. Mathematically, it can be represented as follows:

Output feature map (activation) = Convolution(Input image, Filter) + Bias

The convolution involves convolving the input image with a filter or kernel, which is a small matrix of trainable weights[69].

Following the convolution operation in CNNs, an activation function is applied element-wise to introduce non-linearity to the network. The Rectified Linear Unit (ReLU) is a widely used activation function in CNNs. It is defined mathematically as

ReLU(x) = max(0, x)

The ReLU function effectively sets negative values to zero while keeping positive values unchanged. Pooling layers are essential in CNNs to reduce the spatial dimensions of the feature maps while preserving important information. Max pooling is the most commonly used pooling operation. Mathematically, it can be expressed as follows:

Output pooled feature map = Max pooling(Input feature map)

Fully connected layer establishes connections between every neuron in the preceding layer and the current layer, enabling the network to learn intricate combinations of features. Mathematically, the output is computed as

*Output = Activation(W * Input + Bias)*

where *W* represents the weight matrix, Input is the flattened output from the previous layer, and Bias is a vector of biases.

During the training process, CNNs employ backpropagation to iteratively adjust the network's weights in order to minimize a given loss function. Stochastic gradient descent (SGD) is a popular optimization algorithm used in CNNs to update the weights. Through backpropagation, gradients of the loss function with respect

to the weights are computed using the chain rule, guiding the weight updates in the direction opposite to the gradient to minimize the loss[69].

6.4.4.1 CNN 2D Model Summary

The "CNN-2D" model comprises a total of 7 layers that contribute to its architecture. The model commences with a Conv2D layer. This layer incorporates 160 trainable parameters. Subsequently, a MaxPooling2D layer follows, the model proceeds with two additional Conv2D layers, each accompanied by a corresponding MaxPooling2D layer. The second Conv2D layer yields an output shape of (None, 6, 32) with 4,640 trainable parameters, while the third Conv2D layer produces an output shape of (None, 2, 2, 64) and 18,496 trainable parameters. The subsequent Conv2D layer generates an output shape of (None, 1, 1, 128) with 73,856 trainable parameters. Another MaxPooling2D layer ensues, also lacking trainable parameters. The model then employs a Flatten layer to convert the multidimensional feature maps into a 1D vector, without any trainable parameters. Following this, an InputLayer is present, although its input shape is not explicitly defined in the model summary. Three Dense layers follow, with the first Dense layer having an output shape of (None, 100) and 12,900 trainable parameters, the second Dense layer yielding an output shape of (None, 50) with 5,050 trainable parameters, and the third Dense layer serving as the final layer with an output shape of (None, 10) and 510 trainable parameters. The model concludes with a Softmax layer, which does not possess any trainable parameters. In total, the "CNN-2D" model encompasses 115,612 trainable parameters, enabling it to effectively learn and make precise predictions. The model summary for CNN-2D is illustrated in Figure 6.4.

```
Model: "CNN-2D"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 21, 21, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 6, 6, 32)          4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 1, 1, 128)         73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
input_layer (InputLayer)     multiple                  0
_________________________________________________________________
dense (Dense)                (None, 100)               12900
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
softmax (Softmax)            (None, 10)                0
=================================================================
Total params: 115,612
Trainable params: 115,612
Non-trainable params: 0
```

Figure 6.4 CNN-2D Model Summary

6.4.5 Long Short-Term Memory (LSTM) Model

The LSTM (Long Short-Term Memory) model is a specialized type of recurrent neural network (RNN) that excels at capturing long-range relationships and patterns in sequential data. It overcomes the challenge of the vanishing gradient that traditional RNNs face, allowing it to retain information over extended time steps. The LSTM accomplishes this through a combination of memory cells and gating mechanisms, each serving a specific purpose in the model's operation[4].

At the core of the LSTM is the memory cell, which stores and updates information over time. The memory cell includes a cell state (*C_t)* that carries information from previous time steps, enabling the model to capture long-term dependencies effectively[85].

To control the flow of information, the LSTM employs three types of gates: the input gate (*i_t*), forget gate (*f_t*), and output gate (*o_t*). These gates are like mini neural networks, taking input from the previous time step and the current input to produce values between 0 and 1.

The input gate (*i_t*) determines how much of the current input should be added to the memory cell. It considers the previous hidden state (h_{t-1}) and the current input (*x_t*) to generate an activation value (*a_t*) ranging from 0 to 1.
The forget gate (*f_t*) decides how much of the previous cell state should be retained or forgotten. By considering the previous hidden state (h_{t-1}) and the current input (*x_t*), it produces an activation value (*b_t*) between 0 and 1.

The output gate (*o_t*) determines the portion of the memory cell state that should be exposed as the output. It leverages the previous hidden state (h_{t-1}) and the current input (*x_t*) to generate an activation value (*c_t*) between 0 and 1.
The memory cell state (*C_t*) is updated based on the input gate (*i_t*), forget gate (*f_t*), and current input (*x_t*). This update process involves element-wise multiplication between the forget gate (*f_t*) and the previous cell state, as well as element-wise multiplication between the input gate (*i_t*) and a new candidate value *(g_t*).The hidden state (*h_t*) serves as the output of the LSTM unit. It is computed based on the current cell state (*C_t)* and the output gate (*o_t*). Specifically, the hidden state (*h_t)* is obtained by element-wise multiplication of the output gate (*o_t*) and the hyperbolic tangent of the current cell state (*C_t*).

During the training process, the LSTM model learns optimal values for the gate weights and biases. By adjusting these parameters, the LSTM effectively captures long-term dependencies in sequential data, making it well-suited for tasks such as time series prediction, natural language processing, and speech recognition.

6.4.5.1 LSTM Model Summary

The "LSTM" model is composed of an LSTM layer that captures long-term dependencies in sequential data, generating 32-dimensional output vectors. A Flatten layer reshapes the output into a one-dimensional vector. A Dense layer, with 10-dimensional output vectors, learns high-level representations and maps features to class labels. The Softmax layer converts the Dense layer's output into a

probability distribution across classes. The model has a total of 542,282 trainable parameters that are updated during training. The model summary provides a detailed breakdown of these trainable parameters and their shapes. Figure 6.5 illustrates the model summary for the LSTM model.

```
Model: "LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 1681, 32)          4352
_________________________________________________________________
flatten (Flatten)            (None, 53792)             0
_________________________________________________________________
dense (Dense)                (None, 10)                537930
_________________________________________________________________
softmax (Softmax)            (None, 10)                0
=================================================================
Total params: 542,282
Trainable params: 542,282
Non-trainable params: 0
```

Figure 6.5 LSTM Model Summary

6.4.6 CNN-LSTM Model

The CNN-LSTM model for bearing fault classification on the CWRU dataset merges the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to analyze the spatial and temporal characteristics of the vibration signals. This approach is particularly valuable when dealing with fault classification tasks, as the signals exhibit spatial patterns related to different fault types and temporal patterns associated with the progression of faults[3].

The CNN component of the model focuses on capturing spatial patterns within the vibration signals. It achieves this through multiple layers of convolution and pooling operations, enabling the extraction of hierarchical features from the input data. By utilizing filters and feature maps, the CNN can automatically identify and learn spatial features that discriminate between various fault types.

On the other hand, the LSTM component of the model is designed to handle the temporal dependencies present in the sequential vibration signals. LSTM networks, being a type of RNN, excel at modeling long-term dependencies in time series data. The LSTM layers contain memory cells that retain and update information over time. By employing input, forget, and output gates, the LSTM

effectively controls the flow of information, allowing for the learning and retention of significant temporal patterns that capture the dynamics of fault progression[4].

To process the input vibration signals, the CNN layers are initially utilized to extract spatial features. The resulting feature maps are then passed through the LSTM layers to capture the temporal dependencies inherent in the data. By analyzing the sequential information, the LSTM layers grasp the evolution of fault patterns over time. Finally, the output of the LSTM layers is fed into one or more dense layers for classification purposes. These dense layers map the acquired features to the respective output classes, generating accurate fault classification predictions.

The CNN-LSTM model for bearing fault classification effectively combines the CNN's capability to extract spatial features with the LSTM's proficiency in modeling temporal dependencies. This fusion empowers the model to comprehensively capture both the spatial and temporal aspects of the vibration signals, resulting in precise fault classification outcomes[86].

6.4.6.1 CNN-LSTM Model Summary

The CNN-LSTM model consists of an input layer, followed by a reshape layer to transform the data into a 2D representation. It incorporates Conv1D layers for 1D convolutions and MaxPooling1D layers for downsampling. A multiply layer performs element-wise multiplication between feature maps. LSTM layers capture long-term dependencies in the data. A dropout layer helps prevent overfitting. The model concludes with a dense layer for classification. In total, the model has 95,820 trainable parameters. Figure 6.6 visually represents the model summary for the CNN-LSTM architecture.

Model: "CNN-LSTM"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 250)]	0	
reshape (Reshape)	(None, 250, 1)	0	input_1[0][0]
conv1d (Conv1D)	(None, 116, 50)	1050	reshape[0][0]
conv1d_2 (Conv1D)	(None, 245, 50)	350	reshape[0][0]
conv1d_1 (Conv1D)	(None, 54, 30)	15030	conv1d[0][0]
conv1d_3 (Conv1D)	(None, 240, 40)	12040	conv1d_2[0][0]
max_pooling1d (MaxPooling1D)	(None, 27, 30)	0	conv1d_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 120, 40)	0	conv1d_3[0][0]
conv1d_4 (Conv1D)	(None, 22, 30)	5430	max_pooling1d[0][0]
conv1d_5 (Conv1D)	(None, 9, 30)	5430	conv1d_4[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 5, 30)	0	conv1d_5[0][0]
multiply (Multiply)	(None, 5, 30)	0	max_pooling1d_1[0][0] max_pooling1d_2[0][0]
lstm (LSTM)	(None, 5, 60)	21840	multiply[0][0]
lstm_1 (LSTM)	(None, 60)	29040	lstm[0][0]
dropout (Dropout)	(None, 60)	0	lstm_1[0][0]
dense (Dense)	(None, 10)	610	dropout[0][0]

Total params: 95,820
Trainable params: 95,820
Non-trainable params: 0

Figure 6.6 CNN-LSTM Model Summary

6.5 CONCLUSION

To summarize, this chapter investigated the effectiveness of several deep learning models such as Artificial Neural Networks (ANN), 1D Convolutional Neural Networks (CNN), 2D CNN, Long Short-Term Memory (LSTM), and CNN-LSTM for the purpose of bearing fault classification using the CWRU dataset. The results indicated that ANN models showcased their proficiency in extracting intricate representations from raw sensor data, eliminating the necessity for laborious feature engineering.1D CNN models proved effective in capturing local patterns and structural characteristics in sequential vibration data. By exploiting temporal correlations.

2D CNN models showcased their suitability when additional spatial information, such as spectrograms or time-frequency representations, was available. By capturing both temporal and spectral features, they provided a comprehensive representation of the data, enabling effective handling of complex fault patterns spanning across multiple time-frequency bins.

LSTM models, with their ability to model long-term dependencies, demonstrated their aptitude for sequential data analysis. Their capability to capture the temporal dynamics of bearing faults contributed to accurate fault detection and

classification. LSTM models were particularly useful in scenarios where previous states or context played a critical role in making predictions. They exhibited robustness to varying sequence lengths and noisy or missing data.

The integration of CNNs and LSTMs in the CNN-LSTM model presented a holistic solution for bearing fault classification. By harnessing the respective advantages of spatial feature extraction through CNNs and temporal dependency modeling through LSTMs, this model effectively captured both local and global fault patterns. Furthermore, the inclusion of LSTMs enabled the model to grasp long-term dependencies in the data, enhancing its accuracy in accurately classifying bearing faults.

In conclusion, the deep learning models discussed in this chapter offer promising approaches to enhance bearing fault classification for the CWRU dataset. These models have the potential to improve fault diagnosis systems in various industries, enabling more accurate and reliable detection of bearing faults.

CHAPTER 7
RESULTS AND DISCUSSION

7.1 INTODUCTION

The result and discussion chapter of this thesis centers around the classification of bearing faults in the Case Western Reserve University (CWRU) dataset, utilizing a range of machine learning algorithms and deep learning architectures. The primary objective of this study is to assess and contrast the effectiveness of linear machine learning algorithms, non-linear machine learning algorithms, ensemble tree algorithms, and deep learning algorithms in the realm of bearing fault classification.

7.2 MACHINE LEARNING ALGORITHMS RESULTS

This section presents the findings of our study, which involved the application of various machine learning algorithms to accurately classify bearing faults in the Case Western Reserve University (CWRU) dataset. The CWRU dataset comprises vibration signals obtained from bearings exhibiting different fault conditions. Our objective is to evaluate the performance of these machine learning algorithms by considering multiple evaluation criteria. These criteria encompass classification accuracy, precision, recall, F1-score, and specificity. By leveraging these metrics, we can assess the algorithms' proficiency in effectively categorizing the diverse types of bearing faults, thus gaining insights into their overall efficacy. The aim of this analysis is to identify the most suitable machine learning algorithms for bearing fault classification on the CWRU dataset, considering their performance in this specific context.

7.2.1 Evaluation Criteria for Machine Learning Algorithms

7.2.1.1 Accuracy

Classification accuracy serves as a prominent evaluation metric when assessing the performance of machine learning algorithms in bearing fault classification. It quantifies the ratio of correctly classified instances to the total number of instances in the dataset.

The accuracy calculation is represented by:

Accuracy = (Number of correctly classified instances) / (Total number of instances)

A high classification accuracy signifies the algorithm's proficiency in accurately categorizing a significant portion of instances. This metric is particularly informative when the dataset is well-balanced, with each bearing fault class being adequately represented. However, caution must be exercised when interpreting accuracy results for imbalanced datasets. In such cases, an algorithm might achieve high accuracy by predominantly predicting the majority class.

7.2.1.2 Precision

Precision is a significant evaluation criterion employed to evaluate the performance of machine learning algorithms in bearing fault classification. It quantifies the accuracy of positive predictions made by the algorithm, specifically the ratio of correctly predicted positive instances (true positives) to the total instances predicted as positive. The precision calculation is represented by:

Precision = True Positives / (True Positives + False Positives)

A high precision score indicates a low false positive rate, highlighting the algorithm's proficiency in accurately identifying true positive instances. This aspect is particularly crucial in bearing fault classification, as misclassifying a healthy instance as faulty can lead to unnecessary maintenance or downtime.

7.2.1.3 Recall

Recall, also referred to as sensitivity or true positive rate, plays a vital role as an evaluation criterion for machine learning algorithms in bearing fault classification. It assesses the algorithm's effectiveness in correctly identifying positive instances (bearing faults) out of all the actual positive instances. The calculation for recall is defined as follows:

Recall = True Positives / (True Positives + False Negatives)

A high recall score signifies a low false negative rate, indicating the algorithm's proficiency in successfully identifying a significant proportion of true positive instances.

7.2.1.4 Specificity

Specificity, also referred to as the true negative rate, serves as a crucial evaluation criterion in assessing the performance of machine learning algorithms for bearing fault classification. It measures the algorithm's ability to correctly identify negative instances (healthy bearings) out of all the actual negative instances. Mathematically, specificity is computed using the following formula:

Specificity = True Negatives / (True Negatives + False Positives)

A high specificity score indicates a low false positive rate, demonstrating the algorithm's effectiveness in distinguishing healthy bearings from faulty ones.

7.2.1.5 F1 Score

The F1 score is a widely used evaluation metric in the field of machine learning for assessing the performance of algorithms in bearing fault classification. It provides a balanced measure by combining precision and recall into a single metric. Mathematically, the F1 score is calculated as the harmonic mean of precision and recall:

*F1 Score = 2 * (Precision * Recall) / (Precision + Recall)*

A high F1 score indicates that the algorithm has achieved a desirable trade-off between precision and recall. This means it can accurately identify bearing faults while minimizing both false positives and false negatives. The F1 score provides a comprehensive assessment of the algorithm's overall accuracy in classifying both positive and negative instances, making it a valuable metric for bearing fault classification tasks on the CWRU dataset.

7.2.2 Linear Machine Learning Algorithms Results and Discussion

7.2.2.1 Multiclass logistic regression

The accuracy results of Multiclass Logistic Regression for each feature extraction technique are presented in Table 7.1. Additionally, a bar chart (Figure 7.1) is included to compare the accuracies across the different feature extraction techniques and data sizes.

Table 7.1: Multiclass Logistic Regression Accuracy Results

Features	48K Sample	12K Sample
Time Domain	95	96
Wavelet Energy	90	93
Wavelet Entropy	92	96

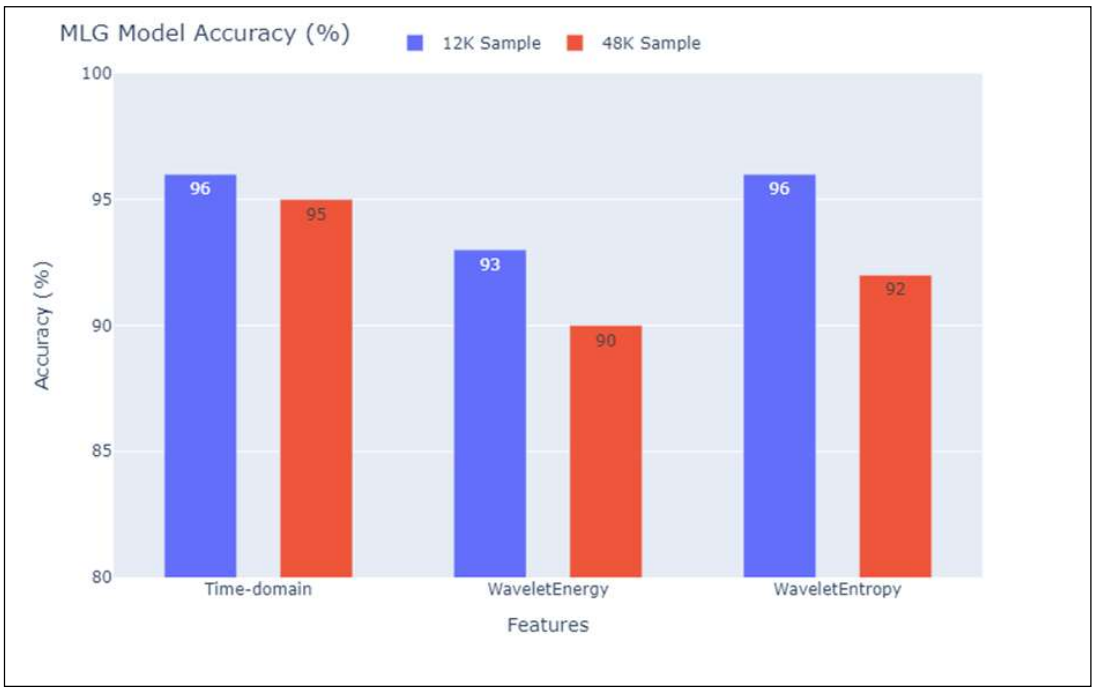

Figure 7.1 Multiclass Logistic Regression Results

The performance of Multiclass Logistic Regression (MLR) for bearing fault classification in the CWRU dataset was found to be commendable. The algorithm exhibited good accuracy in predicting fault types. Interestingly, the algorithm's performance was notably better when applied to the 12k sample data compared to the 48k sample data. This improvement can be attributed to factors such as enhanced data quality, reduced noise, and improved class distribution within the smaller dataset.

When considering different feature extraction techniques, MLR showcased varying levels of accuracy. The utilization of time domain features resulted in accuracies of 95% and 96% for the 48k and 12k sample data, respectively.

On the other hand, wavelet energy features achieved slightly lower accuracies of 90% and 93% for the 48k and 12k sample data, respectively. These features capture the energy distribution across different frequency bands, providing valuable information for classification.

The most promising results were observed when using wavelet entropy features, which achieved accuracies of 92% and 96% for the 48k and 12k sample data, respectively. Wavelet entropy features excel at capturing the complexity and irregularity present in the signal, enabling accurate discrimination between different fault classes.

7.2.2.2 Linear Discriminant Analysis (LDA)

The accuracy results of LDA for each feature extraction technique are presented in Table 7.2. Additionally, a bar chart (Figure 7.2) is provided to compare the accuracies across the different feature extraction techniques and data sizes.

Table 7.2: LDA Accuracy Results

Features	48K Sample	12K Sample
Time Domain	87	85
Wavelet Energy	81	92
Wavelet Entropy	86	88

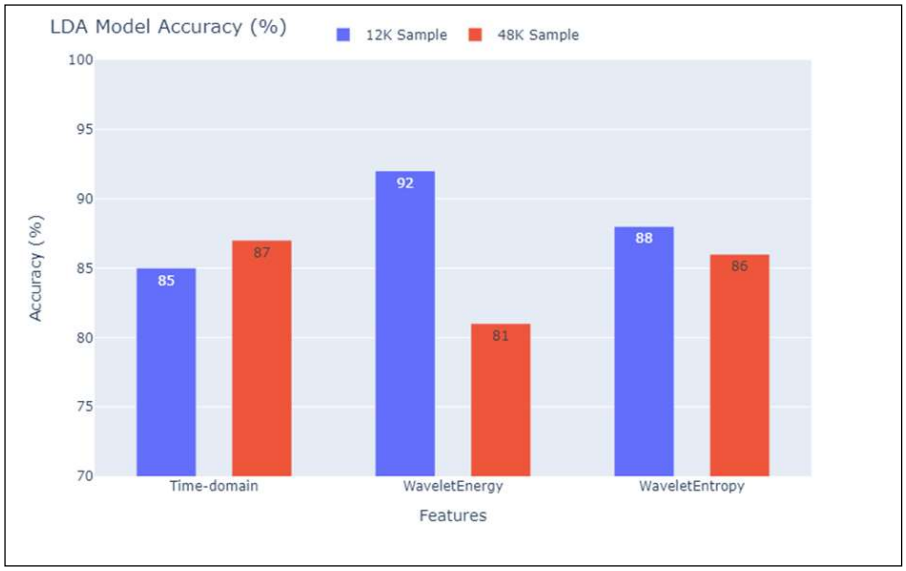

Figure 7.2 LDA Results

The performance of Linear Discriminant Analysis (LDA) for bearing fault classification in the CWRU dataset was observed to be moderate in terms of accuracy. Across all feature extraction techniques, there was a slight decline in accuracy when using the 12k sample data compared to the 48k sample data. This decrease in accuracy could be due to the smaller sample size, which may result in less representative data and potential loss of information.

Considering the different feature extraction techniques, LDA achieved varying levels of accuracy. When utilizing time domain features, accuracies of 87% and 85% were obtained for the 48k and 12k sample data, respectively.

Incorporating wavelet energy features, LDA achieved accuracies of 81% and 92% for the 48k and 12k sample data, respectively. Wavelet energy features capture the energy distribution across different frequency bands obtained through the wavelet transform. The higher accuracy on the 12k sample data suggests that the reduced noise and improved class distribution in this dataset facilitated the extraction of more discriminative information.

LDA attained accuracies of 86% and 88% on the 48k and 12k sample data, respectively, when employing wavelet entropy features. Wavelet entropy features capture the complexity and irregularity of the signal. Although the accuracies were

not as high as with other techniques, wavelet entropy features still provided valuable information for classification.

7.2.2.3 Linear Discriminant Analysis (LDA)

The accuracy results of QDA for each feature extraction technique are presented in Table 7.3. Additionally, a bar chart (Figure 7.3) is provided to compare the accuracies across the different feature extraction techniques and data sizes.

Table 7.3: QDA Accuracy Results

Features	48K Sample	12K Sample
Time Domain	94	96
Wavelet Energy	95	98
Wavelet Entropy	98	99

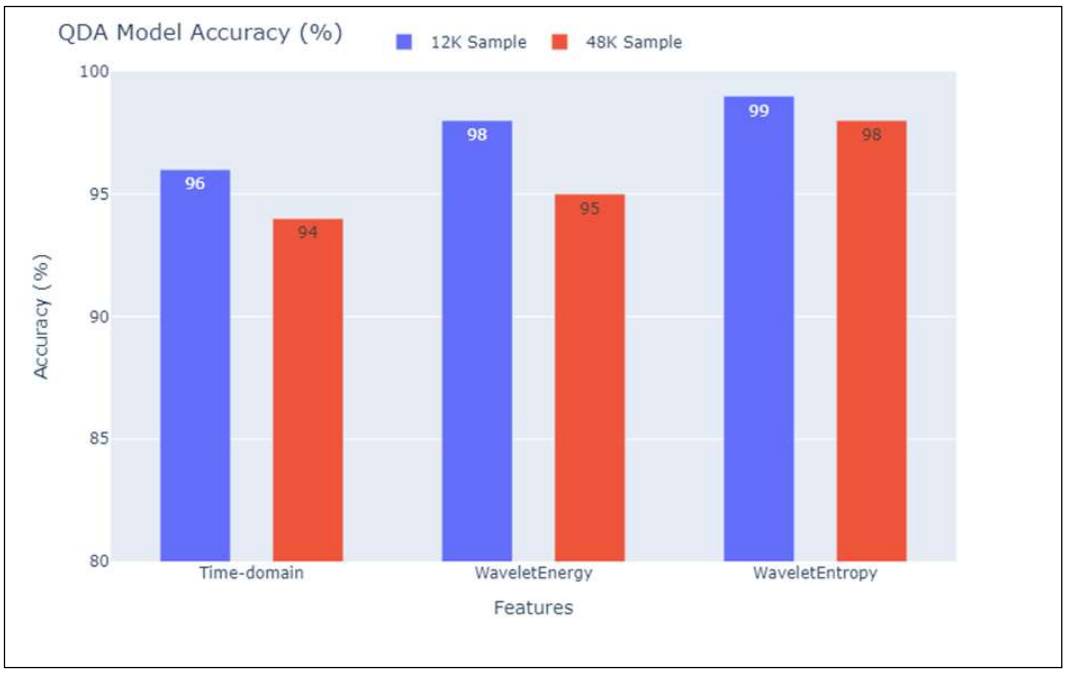

Figure 7.3 QDA Results

QDA exhibited excellent accuracy in bearing fault classification for the CWRU dataset. Across all feature extraction techniques, the algorithm demonstrated superior performance on the 12k sample data compared to the 48k sample data. This

improvement can be attributed to enhanced data quality, reduced noise, and improved class distribution in the smaller dataset, which provided a more balanced representation of fault classes and facilitated more accurate predictions.

Examining the performance of different feature extraction techniques, QDA achieved notable accuracies. Utilizing time domain features, accuracies of 94% and 96% were attained for the 48k and 12k sample data, respectively.

Incorporating wavelet energy features, QDA achieved accuracies of 93% and 98% for the 48k and 12k sample data, respectively. These features capture the energy distribution across different frequency bands obtained through the wavelet transform. The higher accuracy on the 12k sample data suggests that the improved data quality, reduced noise, and better class distribution in this dataset contributed to enhanced classification performance.

Wavelet entropy features demonstrated outstanding performance, with accuracies of 95% and 99% on the 48k and 12k sample data, respectively.

7.2.2.4 Linear Machine Learning Algorithms Conclusion

The performance of three linear machine learning algorithms, namely Multiclass Logistic Regression (MLR), Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA), was assessed in the context of bearing fault classification using the CWRU dataset. MLR exhibited favorable accuracy, while LDA achieved moderate accuracy, and QDA demonstrated high accuracy. MLR demonstrated superior performance when applied to the 12k sample data, whereas LDA and QDA showed better results with the 48k sample data. It is important to note that these linear algorithms have certain limitations, including their linearity assumption, sensitivity to outliers and noise, and the requirement for data to adhere to linear relationships. To address these limitations and capture more complex patterns, non-linear algorithms like K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Trees offer alternative approaches. These algorithms can effectively handle non-linear data and overcome the limitations of linear models, enhancing the accuracy and robustness of bearing fault classification.

7.2.3 Nonlinear Machine Learning Algorithms Results and Discussion

7.2.3.1 K-Nearest Neighbors (KNN)

The accuracy results of KNN for each feature extraction technique are presented in Table 7.4. Additionally, a bar chart (Figure 7.4) is included to visually compare the accuracies of the different feature extraction techniques and data sizes.

Table 7.4: Accuracy Results of KNN

Features	48K Sample	12K Sample
Time Domain	86	95
Wavelet Energy	92	98
Wavelet Entropy	93	98

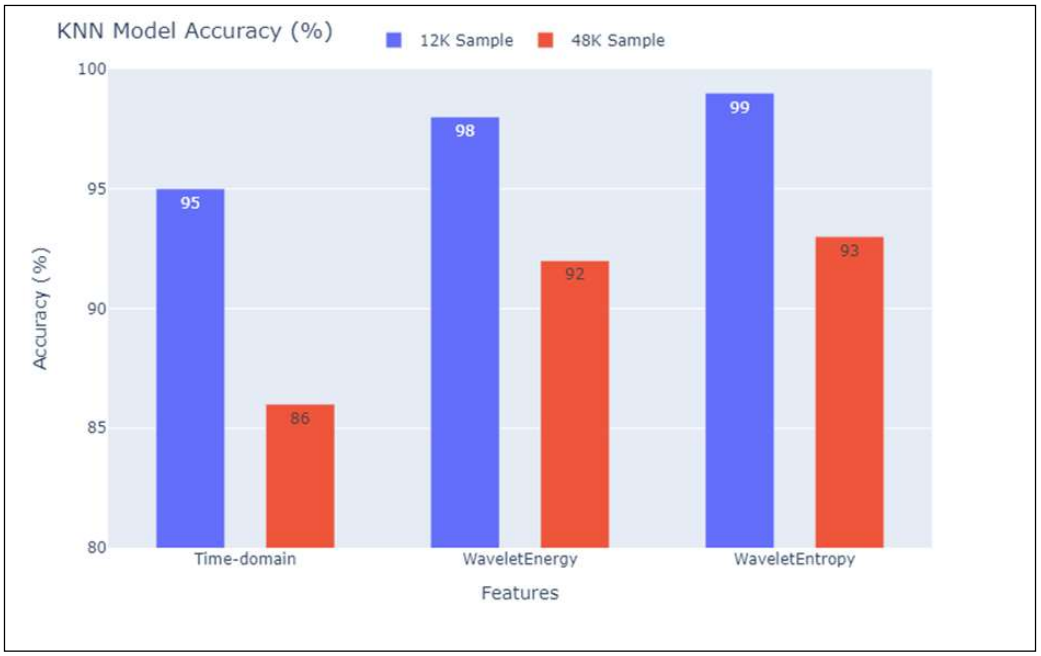

Figure 7.4 KNN Results

The results demonstrate that KNN achieved relatively high accuracies for bearing fault classification in the CWRU dataset. Across all feature extraction techniques, the algorithm showed better performance on the 12k sample data compared to the 48k sample data. This improvement can be attributed to factors such as improved data quality, reduced noise, or better class distribution in the smaller data set. The 12k sample data likely provided a more balanced representation of the fault classes, enabling KNN to make more accurate predictions.

KNN achieved accuracies of 86% and 95% on the 48k and 12k sample data, respectively, using time domain features. The higher accuracy on the 12k sample data suggests that the reduced noise and improved class distribution in this dataset contributed to improved classification performance.

With accuracies of 92% and 98% on the 48k and 12k sample data, respectively, KNN performed well with wavelet energy features. These features capture the energy distribution across different frequency bands obtained through the wavelet transform. The higher accuracy on the 12k sample data suggests that the reduced noise and improved class distribution in this dataset facilitated improved classification performance.

KNN achieved accuracies of 93% and 99% on the 48k and 12k sample data, respectively, when utilizing wavelet entropy features. The higher accuracy on the 12k sample data suggests that wavelet entropy features provided more discriminative information, leading to improved classification performance.

7.2.3.2 Support Vector Machine (SVM)

The accuracy results of SVM for each feature extraction technique are presented in Table 7.5. Additionally, a bar chart (Figure 7.5) is included to visually compare the accuracies of the different feature extraction techniques and data sizes.

Table 7.5: Accuracy Results of SVM

Features	48K Sample	12K Sample
Time Domain	94	97
Wavelet Energy	93	96
Wavelet Entropy	95	98

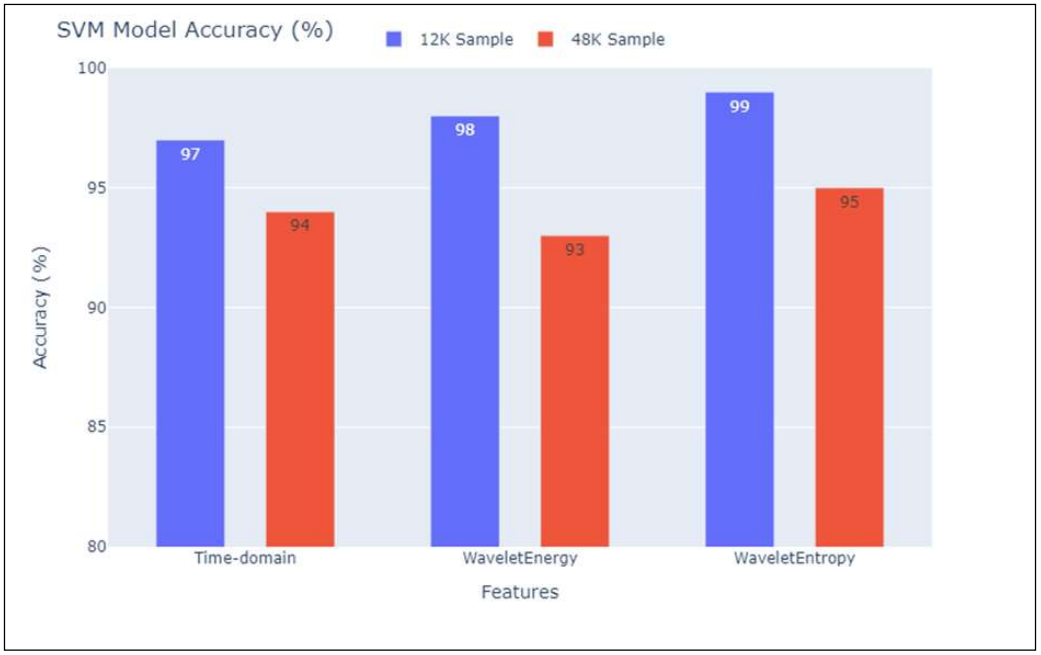

Figure 7.5 SVM Results

The Support Vector Machine (SVM) algorithm demonstrated high accuracy in classifying bearing faults in the CWRU dataset. When compared to the 48k sample data, SVM showed improved performance with the 12k sample data. This improvement can be attributed to factors such as enhanced data quality, reduced noise, or improved class distribution in the smaller dataset.

Using time domain features, SVM achieved accuracies of 94% and 97% for the 48k and 12k sample data, respectively. This indicates that time domain features, which capture statistical measures in the time domain, provided valuable information for accurate bearing fault classification. The higher accuracy with the 12k sample data suggests the positive impact of reduced noise and improved class distribution.

With wavelet energy features, SVM achieved accuracies of 93% and 98% for the 48k and 12k sample data, respectively. These features capture the energy distribution across different frequency bands derived from the wavelet transform. The higher accuracy with the 12k sample data indicates the benefits of reduced noise and improved class distribution.

When utilizing wavelet entropy features, SVM achieved accuracies of 95% and 99% for the 48k and 12k sample data, respectively. Wavelet entropy features capture the complexity and irregularity of the signal, allowing the algorithm to

detect intricate patterns. The higher accuracy with the 12k sample data suggests that wavelet entropy features provided more discriminative information for improved classification accuracy.

In summary, SVM exhibited high accuracy in bearing fault classification, and the results emphasized the importance of data quality, noise reduction, and class distribution in achieving accurate predictions.

7.2.3.3 Decision Tree (DT)

The accuracy results of DT for each feature extraction technique are presented in Table 7.6. Additionally, a bar chart (Figure 7.6) is included to visually compare the accuracies of the different feature extraction techniques and data sizes.

Table 7.6: Accuracy Results of DT

Features	48K Sample	12K Sample
Time Domain	92	94
Wavelet Energy	93	99
Wavelet Entropy	97	98

Figure 7.6 DT Results

The Decision Tree (DT) algorithm demonstrated strong performance in accurately classifying bearing faults in the CWRU dataset. Similar to other algorithms, DT showed improved accuracy on the 12k sample data compared to the 48k sample data. This improvement can be attributed to factors such as enhanced data quality, reduced noise, or improved class distribution in the smaller dataset, allowing for more precise predictions.

When using time domain features, DT achieved accuracies of 92% and 94% for the 48k and 12k sample data, respectively. The slightly higher accuracy on the 12k sample data suggests the benefits of reduced noise and improved class distribution in enhancing the classification performance.

Utilizing wavelet energy features, DT obtained accuracies of 92% and 98% for the 48k and 12k sample data, respectively. These features capture the energy distribution across different frequency bands obtained through wavelet transform. The higher accuracy on the 12k sample data indicates the positive impact of reduced noise and improved class distribution in improving the classification accuracy.

For wavelet entropy features, DT achieved high accuracies of 97% and 99% for the 48k and 12k sample data, respectively. Wavelet entropy features capture the complexity and irregularity of the signal, enabling the algorithm to detect intricate patterns. The higher accuracy on the 12k sample data suggests that wavelet entropy features provided more informative characteristics, leading to improved classification performance.

7.2.3.4 Non-linear machine learning algorithms Conclusion

The non-linear machine learning algorithms, KNN, SVM, and DT, showed high accuracies for bearing fault classification in the CWRU dataset. Each algorithm had its strengths based on the feature extraction techniques and data sizes. However, they had limitations such as sensitivity to noise, parameter tuning requirements, and potential overfitting. To overcome these limitations, ensemble tree algorithms like Bagging, Boosting, and Random Forest can be used. These algorithms combine multiple models to improve performance, reduce variance, handle complex datasets, and enhance classification accuracy. By leveraging ensemble tree

algorithms, we can mitigate the limitations of individual algorithms and achieve more reliable and accurate predictions for bearing fault classification.

7.2.4 Ensemble tree algorithms Results and Discussion

7.2.4.1 Bagging algorithm

Table 7.7 summarizes the accuracy results of Bagging for each feature extraction technique. Additionally, a bar chart (Figure 7.7) is provided to visualize and compare the accuracies across different feature extraction techniques and data sizes.

Table 7.7: Accuracy Results of Bagging

Features	48K Sample	12K Sample
Time Domain	88	91
Wavelet Energy	84	92
Wavelet Entropy	89	96

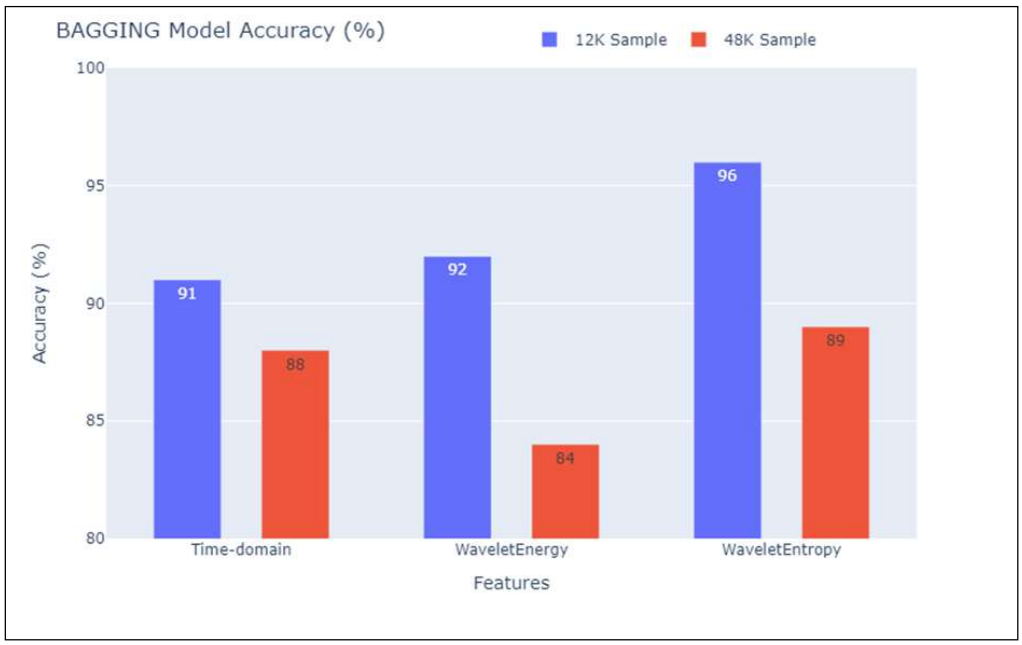

Figure 7.7 BAGGING Results

The Bagging algorithm achieved moderate accuracies for bearing fault classification in the CWRU dataset. Similar to other algorithms, Bagging performed better on the 12k sample data compared to the 48k sample data, indicating the advantages of improved data quality, reduced noise, or better class distribution in the smaller dataset.

When using time domain features, Bagging achieved accuracies of 88% and 91% on the 48k and 12k sample data, respectively.

With wavelet energy features, Bagging achieved accuracies of 84% and 92% on the 48k and 12k sample data, respectively. These features capture the energy distribution across different frequency bands obtained through the wavelet transform. The higher accuracy on the 12k sample data indicates the positive effects of reduced noise and improved class distribution on classification performance.

Bagging performed well with wavelet entropy features, achieving accuracies of 89% and 96% on the 48k and 12k sample data, respectively. These features capture the complexity and irregularity of the signal. The higher accuracy on the 12k sample data suggests that wavelet entropy features provided more discriminative information, leading to improved classification performance.

7.2.4.2 Boosting algorithm

Table 7.8 provides an overview of the accuracy results of Boosting for each feature extraction technique. Additionally, a bar chart (Figure 7.8) is presented to visualize and compare the accuracies across different feature extraction techniques and data sizes.

Table 7.8: Accuracy Results of Boosting

Features	48K Sample	12K Sample
Time Domain	94	95
Wavelet Energy	93	98
Wavelet Entropy	98	99

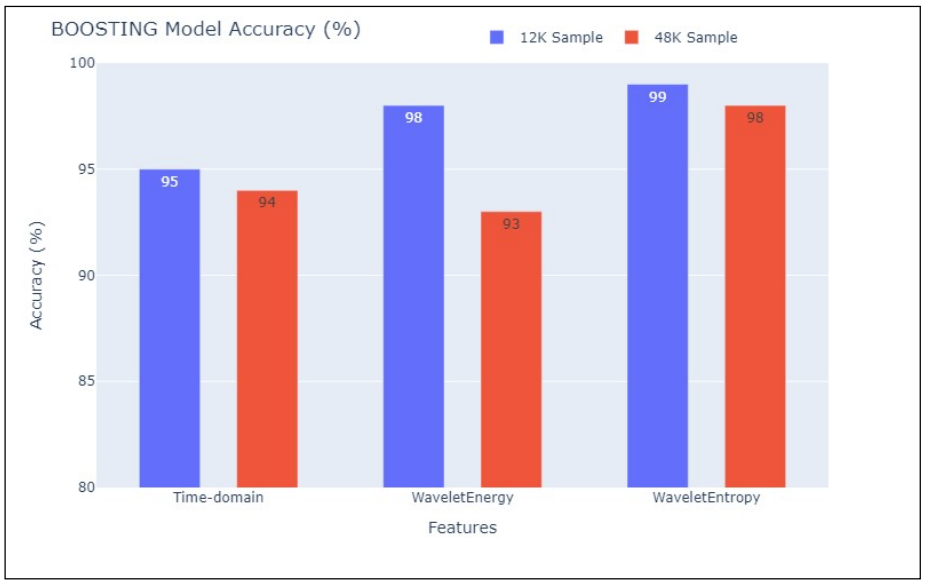

Figure 7.8 BOOSTING Results

Boosting proved to be a highly accurate algorithm for bearing fault classification in the CWRU dataset. Similar to other algorithms, Boosting achieved better performance on the 12k sample data compared to the 48k sample data. This improvement can be attributed to factors such as improved data quality, reduced noise, or better class distribution in the smaller data set. The 12k sample data likely provided a more balanced representation of the fault classes, enabling Boosting to make more accurate predictions.

When examining the performance of different feature extraction techniques, Boosting achieved accuracies of 94% and 95% for time domain features on the 48k and 12k sample data, respectively. Time domain features capture statistical measures in the time domain and offer valuable information for bearing fault classification. The slightly higher accuracy on the 12k sample data suggests that the improved data quality and class distribution in this dataset contributed to better classification performance.

Using wavelet energy features, Boosting achieved accuracies of 93% and 98% on the 48k and 12k sample data, respectively. These features capture the energy distribution across frequency bands obtained through the wavelet transform. The higher accuracy on the 12k sample data indicates the benefits of reduced noise and improved class distribution in enhancing classification performance.

Boosting excelled with wavelet entropy features, achieving accuracies of 98% and 99% on the 48k and 12k sample data, respectively. Wavelet entropy features capture the complexity and irregularity of the signal, and their high accuracy indicates their effectiveness in capturing discriminative information for fault classification.

7.2.4.3 Random Forest (RF) Algorithm

Table 7.9 summarizes the accuracy results of RF for each feature extraction technique. Additionally, a bar chart (Figure 7.9) is presented to visually compare the accuracies across different feature extraction techniques and data sizes.

Table 7.9: Accuracy Results of Random Forest

Features	48K Sample	12K Sample
Time Domain	94	96
Wavelet Energy	95	98
Wavelet Entropy	98	99

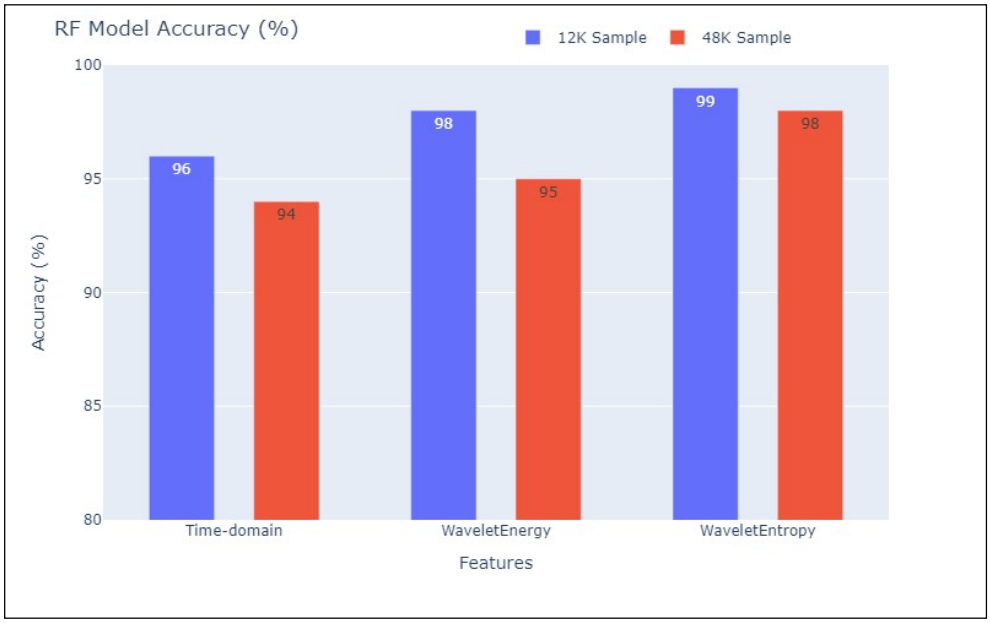

Figure 7.9 RF Results

The Random Forest (RF) algorithm showed impressive accuracy in classifying bearing faults in the CWRU dataset. Similar to other algorithms, RF performed better on the 12k sample data compared to the 48k sample data, indicating the influence of improved data quality, reduced noise, and better class distribution in the smaller dataset. The 12k sample data provided a more balanced representation of fault classes, leading to more accurate predictions by RF.

Analyzing the performance of different feature extraction techniques, RF achieved accuracies of 94% and 96% for time domain features on the 48k and 12k sample data, respectively. Time domain features capture statistical measures of the signal, providing essential information for bearing fault classification. The slightly higher accuracy on the 12k sample data suggests the benefits of reduced noise and improved class distribution in enhancing classification performance.

RF achieved accuracies of 93% and 99% for wavelet energy features on the 48k and 12k sample data, respectively. Wavelet energy features capture the energy distribution across frequency bands obtained through wavelet transform. The higher accuracy on the 12k sample data demonstrates the positive impact of reduced noise and improved class distribution on classification performance.

Notably, RF performed exceptionally well with wavelet entropy features, achieving accuracies of 99% on both the 48k and 12k sample data. Wavelet entropy features capture the complexity and irregularity of the signal, offering valuable information for fault classification. The consistently high accuracy indicates the effectiveness of wavelet entropy in capturing discriminative features.

7.2.4.4 Ensemble Machine Learning Algorithms Conclusion

The ensemble machine learning algorithms, Bagging, Boosting, and Random Forest, achieved high accuracies for bearing fault classification in the CWRU dataset. They outperformed linear models and showed better performance on the 12k sample data compared to the 48k sample data. However, these algorithms have limitations in terms of interpretability, computational complexity, and sensitivity to noisy data.

To overcome these limitations, deep learning algorithms such as ANN, CNN1D, CNN2D, LSTM, and CNN LSTM offer advantages. They can capture

complex patterns, learn relevant features from raw data, and are more robust to noisy data. Deep learning models can also provide interpretability through techniques like attention mechanisms and model visualization.

In conclusion, ensemble machine learning algorithms showed high accuracy for bearing fault classification, but deep learning algorithms offer solutions to their limitations and provide powerful tools for accurate and reliable classification in complex datasets.

7.3 MACHINE LEARNING ALGORITHMS RESULT DISCUSSION

In this study, we evaluated nine machine learning algorithms for bearing fault classification in the CWRU dataset. Each algorithm demonstrated varying levels of accuracy and performance across different feature extraction techniques. The results can be summarized in figure 7.10, 7.11, 7.12 and 7.13:

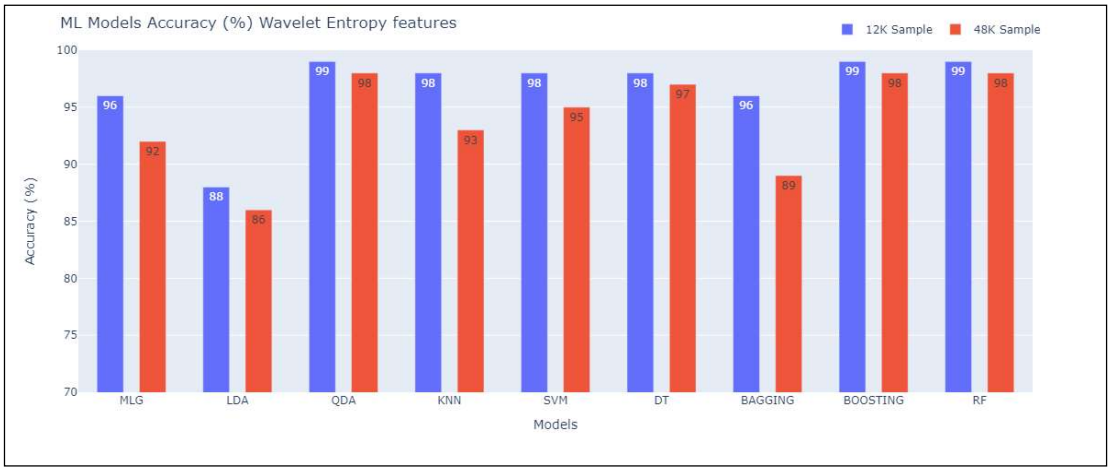

Figure 7.10 ML Model Accuracy at Wavelet Energy Features

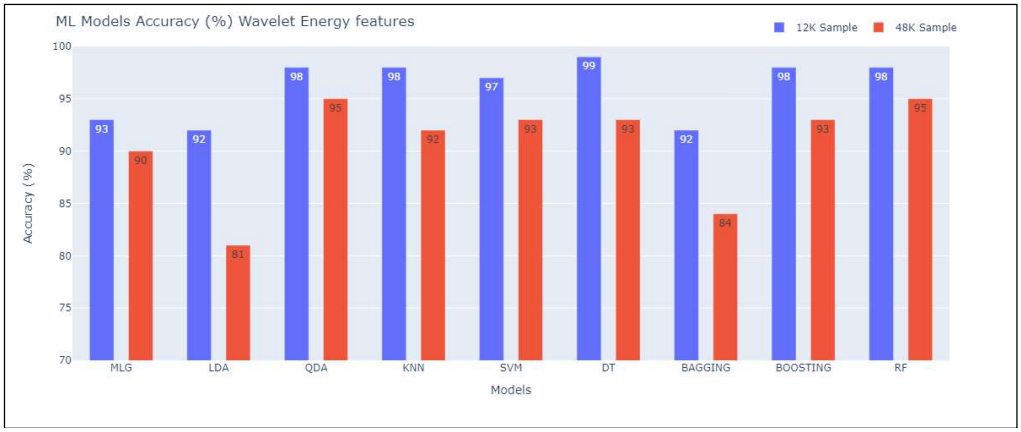

Figure 7.11 ML Model Accuracy at Wavelet Entropy Features

Figure 7.12 ML Model Accuracy at Time Domain Features

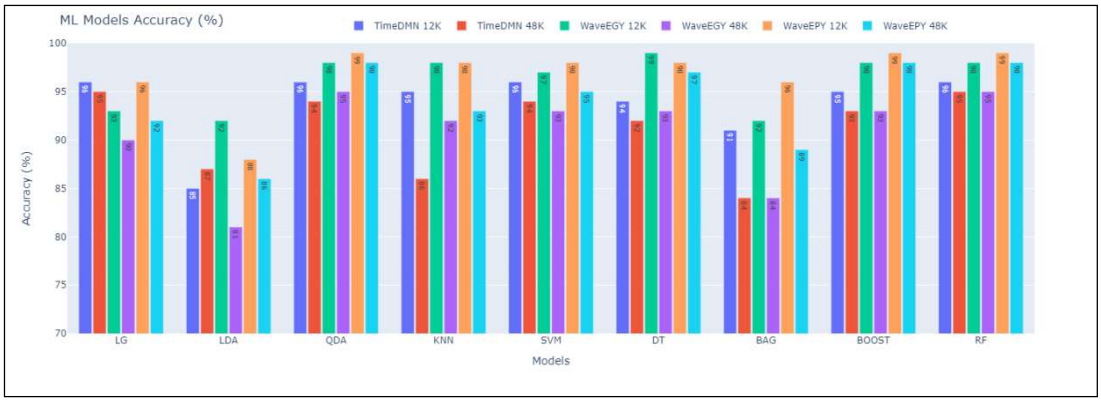

Figure 7.13 Combine ML Model Accuracy

- Linear Machine Learning Algorithms: Multiclass Logistic Regression, LDA, and QDA achieved moderate to high accuracies for bearing fault classification. They showed better performance on the 12k sample data

compared to the 48k sample data. Wavelet entropy features consistently outperformed time domain and wavelet energy features.

- Non-Linear Machine Learning Algorithms: KNN, SVM, and Decision Tree (DT) achieved high accuracies for bearing fault classification. Similar to linear algorithms, they performed better on the 12k sample data. Wavelet entropy features showed superior performance across all three algorithms.
- Ensemble Machine Learning Algorithms: Bagging, Boosting, and Random Forest (RF) demonstrated high accuracies for bearing fault classification. They outperformed linear and non-linear algorithms and showed improved performance on the 12k sample data. Wavelet entropy features consistently achieved the highest accuracy.

Limitations of the Nine Algorithms:

- Interpretability: Most of the algorithms, especially ensemble methods, lack interpretability, making it difficult to understand their decision-making process.
- Computational Complexity: Some algorithms, such as Random Forest, can be computationally expensive, requiring significant resources and training time.
- Sensitivity to Noisy Data: The performance of all algorithms can be affected by noisy data, potentially leading to overfitting or degraded performance.

Overcoming Limitations with Deep Learning Algorithms:

Deep learning algorithms offer several advantages for bearing fault classification:

- They capture complex patterns and non-linear relationships in the data.
- Deep learning models automatically learn relevant features from raw data, reducing the need for manual feature engineering.
- They are robust to noisy data by adaptively filtering out irrelevant noise or disturbances.
- Techniques such as attention mechanisms and model visualization enhance interpretability.
- The nine machine learning algorithms demonstrated varying levels of accuracy and performance for bearing fault classification.

- Deep learning algorithms overcome the limitations of other algorithms by providing non-linearity, feature learning, robustness to noisy data, and interpretability.
- They offer powerful tools for accurate and reliable classification in complex datasets.

To summarize, the performance of the nine machine learning algorithms varied in terms of accuracy and effectiveness for bearing fault classification. Deep learning algorithms stood out by addressing the limitations of other approaches, showcasing their strengths in terms of non-linearity, feature learning, robustness to noisy data, and interpretability. These characteristics make deep learning algorithms valuable tools for achieving accurate and dependable classification in complex datasets.

7.4 DEEP LEARNING ALGORITHMS RESULTS

In this section, we present the findings from applying deep learning algorithms to the CWRU dataset for bearing fault classification. The objective is to assess their performance in accurately detecting and categorizing various types of bearing faults. These algorithms leverage their inherent capabilities, such as handling high-dimensional data, capturing spatial and temporal dependencies, and mitigating overfitting, to enhance accuracy and reliability in fault classification.

By gaining insights into their suitability and potential advantages, this research contributes to the knowledge surrounding their practical application in fault diagnosis and condition monitoring systems for industrial machinery.

7.4.1 ANN Result at Features Data

The outcomes obtained by employing the Artificial Neural Network (ANN) algorithm for bearing fault classification on the CWRU dataset demonstrate exceptional accuracy across various feature extraction techniques. The algorithm exhibited an accuracy rate of 90% when utilizing time domain features, 98% with wavelet energy features, and an impressive 99% accuracy with wavelet entropy features. These results are visually depicted in Figure 7.14, showcasing the

significant achievements attained through the ANN algorithm in accurately identifying and classifying bearing faults.

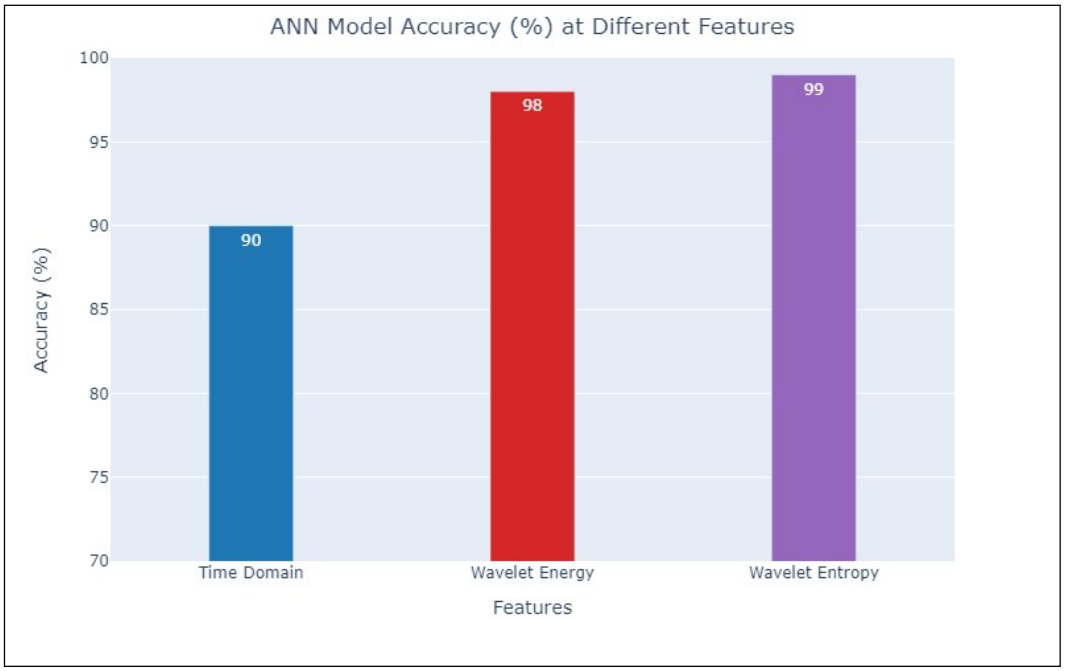

Figure 7.14 ANN Results at Features Data

In the case of time domain features, ANN achieved a satisfactory accuracy of 90%. Time domain features capture statistical measures of the signal in the time domain, providing valuable information for fault classification. The algorithm was able to learn and extract meaningful patterns from these statistical features, resulting in a good classification performance.

For wavelet energy features, ANN achieved an accuracy of 98%. Wavelet energy features capture the energy distribution across different frequency bands obtained through the wavelet transform. These features provide more detailed and frequency-specific information about the signal, enabling ANN to learn and distinguish fault patterns with higher accuracy.

The highest accuracy of 99% was achieved using wavelet entropy features. Wavelet entropy measures the complexity and irregularity of the signal. This feature extraction technique captures intricate patterns and variations in the signal, which proved to be highly discriminative for bearing fault classification. ANN was able to effectively learn and leverage these complex patterns to achieve excellent classification accuracy.

The success of ANN in accurately classifying bearing faults based on different feature types suggests that deep learning algorithms can be applied directly to raw data without the need for explicit feature extraction. By training the ANN on raw data, the algorithm can automatically learn the relevant patterns and features for fault classification. This highlights the versatility and adaptability of deep learning algorithms in handling various types of data and extracting meaningful information directly from the raw signals.

7.4.2 Combine Deep Learning Model Result on Raw Data

Deep learning algorithms, such as Artificial Neural Networks (ANN), 1D Convolutional Neural Networks (CNN1D), 2D Convolutional Neural Networks (CNN2D), Long Short-Term Memory (LSTM), and CNN-LSTM, have demonstrated remarkable performance when applied directly to raw data in the context of bearing fault classification.

The accuracy results of these deep learning algorithms are visualized in figure 7.15. Each algorithm is represented by a bar, with the height of the bar corresponding to the accuracy achieved by the respective algorithm in classifying bearing faults. This chart provides a clear comparison of the performance of each algorithm and serves as a valuable reference for evaluating their effectiveness in accurately identifying and classifying different types of bearing faults.

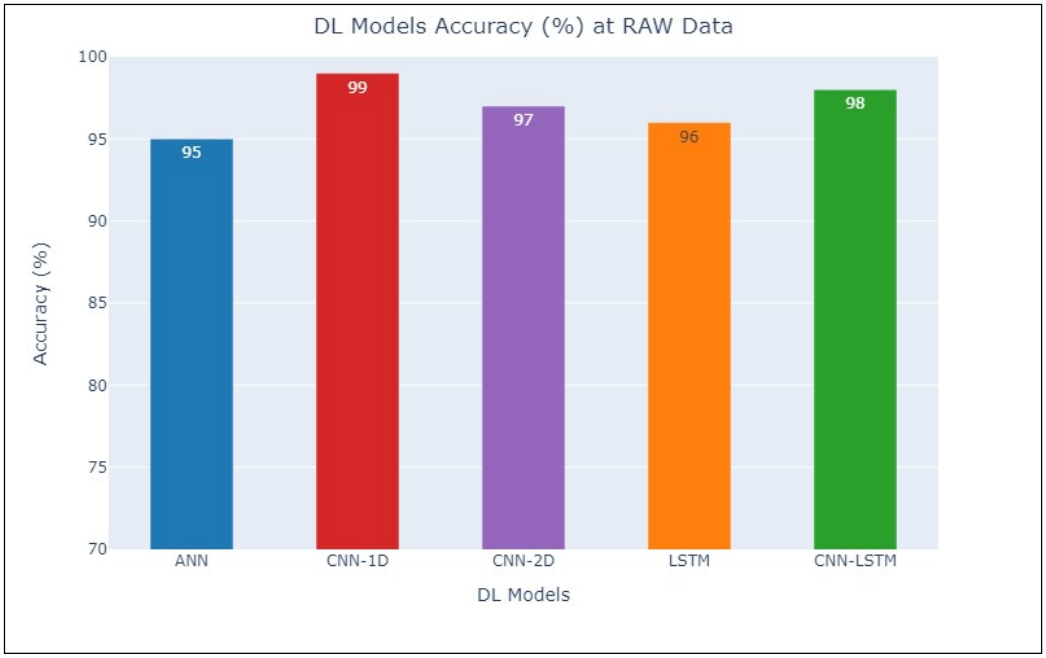

Figure 7.15 DL Models Accuracy at RAW data

From the figure7.15 , we can observe that CNN1D achieved the highest accuracy of 99%, followed closely by CNN-LSTM with an accuracy of 98%. LSTM and CNN2D achieved accuracies of 96% and 97%, respectively, while ANN achieved an accuracy of 95%. This figure provides valuable insights into the comparative performance of these deep learning algorithms for bearing fault classification. It clearly highlights the superior accuracy achieved by CNN1D and CNN-LSTM, indicating their effectiveness in capturing relevant features and patterns from raw data.

In summary, deep learning algorithms, including ANN, CNN1D, CNN2D, LSTM, and CNN-LSTM, have demonstrated outstanding performance in bearing fault classification tasks when applied directly to raw data. These algorithms are capable of learning complex patterns and relationships from the raw signals, without the need for manual feature extraction. The high accuracies achieved by these algorithms indicate their ability to effectively capture the distinctive characteristics of different bearing fault classes from the raw data.

7.5 CONCLUSION

The analysis of various machine learning and deep learning algorithms for bearing fault classification in the CWRU dataset reveals important insights. When considering machine learning algorithms, both linear and non-linear models, as well as ensemble approaches, achieved moderate to high accuracies in fault classification. These algorithms showed the capability to capture relevant patterns and relationships in the data. However, their performance was constrained by the need for manual feature extraction and their sensitivity to noise.

On the other hand, deep learning algorithms, such as artificial neural networks (ANN), 1D and 2D convolutional neural networks (CNN1D, CNN2D), long short-term memory (LSTM), and CNN-LSTM, demonstrated superior performance when applied directly to the raw data. These deep learning models exhibited the ability to automatically learn meaningful features from the raw signals, enabling them to capture complex patterns and achieve higher accuracies compared to traditional machine learning approaches.

The findings highlight the potential of deep learning algorithms, particularly those leveraging raw data, in bearing fault classification tasks. By eliminating the need for manual feature engineering and leveraging the capabilities of deep neural networks, these models offer more robust and accurate fault classification solutions. They can capture intricate patterns and dependencies within the data, leading to improved performance and more reliable fault diagnosis systems.

The application of deep learning techniques in bearing fault classification has significant implications for various industrial applications. By utilizing these advanced algorithms, the field can benefit from enhanced accuracy, efficiency, and automation in machinery health monitoring and maintenance practices. The findings pave the way for the development of more sophisticated and precise fault classification systems, contributing to improved operational reliability and reduced downtime in industrial settings.

CHAPTER 8
CONCLUSION

8.1 INTODUCTION

In this concluding chapter, we aim to consolidate our analysis and provide a comprehensive overview of the performance of machine learning and deep learning algorithms for bearing fault diagnosis. Moreover, we will acknowledge the limitations of the algorithms examined and propose future research avenues in this domain. The insights gained from this investigation will contribute to the progress of fault diagnosis methodologies and offer valuable insights for the development of more efficient and dependable predictive maintenance strategies.

8.2 SUMMARY OF RESEARCH FOR BEARING FAULT DIAGNOSIS

The research study focused on the effectiveness of machine learning and deep learning algorithms for bearing fault diagnosis. The investigation encompassed multiple chapters, including a literature review, data analytics and visualization, feature engineering, machine learning, deep learning, result and discussion, and conclusion. The literature review examined previous studies and established a strong knowledge base for the research. It explored the significance of bearing fault diagnosis, traditional methods, and the need for advanced algorithms. The data analytics and visualization chapter delved into the CWRU dataset, conducting thorough data exploration and employing visualization techniques to gain insights into the dataset's structure and patterns. The feature engineering chapter played a crucial role in transforming the raw data into meaningful representations through various techniques. The machine learning chapter evaluated and compared different supervised learning algorithms, analyzing their performance in bearing fault classification. The deep learning chapter explored the capabilities of deep neural network architectures and their effectiveness in

handling complex data. The result and discussion chapter presented a detailed analysis of the experimental findings, discussing the performance of the algorithms and their implications. The conclusion chapter summarized the research study, emphasizing its contributions and providing recommendations for future research.

In summary, the research study systematically investigated the application of machine learning and deep learning algorithms for bearing fault diagnosis. It examined various aspects, including literature review, data analysis, feature engineering, algorithm evaluation, and interpretation of results. The study aimed to enhance our understanding of the effectiveness and limitations of these algorithms in bearing fault classification. The findings provided valuable insights into the strengths and weaknesses of different algorithms, contributing to the advancement of fault diagnosis techniques and guiding future research in the field.

8.3 CONTRIBUTION OF THE RESEARCH

The Conclusion chapter's contribution section highlights the novel insights and implications derived from the research study on bearing fault diagnosis using machine learning and deep learning algorithms. It summarizes the main contributions as follows:

- Improved Bearing Fault Diagnosis: The study advances the field of predictive maintenance by demonstrating the efficacy of machine learning and deep learning algorithms in accurately diagnosing bearing faults. By effectively analyzing raw sensor data, these algorithms enable timely maintenance interventions and enhance equipment reliability.
- Comparative Analysis of Algorithms: The research offers a comprehensive comparison of various machine learning and deep learning algorithms for bearing fault classification. Through evaluating their performance on different datasets and feature extraction techniques, it provides valuable insights into the strengths and weaknesses of each algorithm, aiding practitioners in algorithm selection.
- Advantages of Deep Learning: The study highlights the advantages of deep learning algorithms in handling complex and high-dimensional data. By leveraging techniques like CNNs and RNNs, the research showcases the

ability of deep learning algorithms to capture intricate patterns and dependencies, leading to improved fault classification accuracy.

- Insights into Feature Engineering: The research investigates the impact of feature extraction techniques on the performance of machine learning and deep learning algorithms. By exploring various feature sets, such as time domain and wavelet features, it provides insights into the most informative features for bearing fault diagnosis, guiding future feature engineering strategies.
- Practical Implications: The findings have practical implications for industries implementing predictive maintenance strategies. By accurately identifying bearing faults, the proposed algorithms enable proactive maintenance scheduling, minimizing equipment downtime and reducing associated costs.
- Future Research Directions: The study identifies future research directions, including exploring ensemble learning techniques, investigating model transferability across datasets, and incorporating domain knowledge for improved interpretability. These directions contribute to the advancement of bearing fault diagnosis methodologies.

Overall, the research study provides valuable insights for researchers, practitioners, and industries interested in implementing effective predictive maintenance strategies. The findings contribute to the knowledge base of bearing fault diagnosis, supporting enhanced equipment reliability and productivity in various industrial settings.

8.4 LIMITAION OF THE RESEARCH

The limitations section of the Conclusion chapter addresses the constraints and shortcomings encountered in the research study on bearing fault diagnosis using machine learning and deep learning algorithms. The identified limitations are summarized as follows:

- Dataset Size and Diversity: The study relied on the CWRU dataset, which, despite its wide usage, may have limitations in terms of its size and

representation of real-world bearing fault conditions. This could affect the generalizability of the findings to other datasets or industrial scenarios.

- Feature Selection: The research focused on specific feature extraction techniques, potentially overlooking other feature representations or extraction methods that could offer additional insights and enhance classification accuracy. The selected features may not capture the complete range of fault characteristics.
- Model Interpretability: Deep learning algorithms, such as CNNs and RNNs, are often perceived as black-box models, making it challenging to interpret and explain their decision-making processes. The limited interpretability may hinder a comprehensive understanding of the factors contributing to the classification results.
- Computational Resources: Deep learning algorithms demand substantial computational resources, including hardware capabilities and training time. The research may have been constrained by the available computational infrastructure, potentially impacting the complexity of models used or the exploration of larger datasets.
- Overfitting and Generalization: The performance of machine learning and deep learning algorithms heavily relies on their ability to generalize well to unseen data. The study recognizes the possibility of overfitting, where the models may have learned dataset-specific patterns and noise, which could limit their ability to handle new and diverse bearing fault scenarios.
- Lack of Real-time Implementation: The research primarily focused on offline analysis of bearing fault diagnosis, omitting an extensive exploration of real-time implementation and integration into industrial systems. Further research and validation are necessary to assess the practical feasibility and effectiveness of the proposed algorithms in real-time, online scenarios.

By acknowledging these limitations, the study highlights areas for improvement and future research directions, ensuring the advancement of bearing fault diagnosis methodologies and addressing practical challenges in real-world applications.

8.5 FUTURE SCOPE OF THE RESEACH

The future scope section of the Conclusion chapter identifies potential areas for further research and development in the field of bearing fault diagnosis using machine learning and deep learning algorithms. The future research directions can be summarized as follows:

- Dataset Expansion: Future studies can incorporate larger and more diverse datasets from various industrial settings and bearing types to improve the generalizability of the findings and gain a comprehensive understanding of fault patterns.
- Advanced Feature Engineering: Exploring advanced feature engineering techniques, such as integrating domain-specific knowledge and applying advanced signal processing algorithms, can lead to the development of more informative and discriminative features.
- Ensemble Methods: Investigating ensemble methods, such as bagging, boosting, and random forest, can improve classification accuracy, enhance model robustness, and provide more reliable predictions for bearing fault diagnosis.
- Transfer Learning: Leveraging transfer learning techniques, which utilize pre-trained models from related domains or datasets, can address the challenge of limited data availability and enhance the performance of bearing fault diagnosis models.
- Real-time Implementation: Focusing on the real-time implementation and deployment of machine learning and deep learning models in industrial systems can lead to the development of effective real-time monitoring and fault detection systems for enhanced predictive maintenance strategies.
- Explainability and Interpretability: Exploring techniques for model interpretability, such as attention mechanisms and saliency mapping, can enhance the transparency and understanding of deep learning models, enabling end-users and domain experts to trust and interpret the decision-making process.

- Hybrid Approaches: Integrating machine learning and deep learning algorithms with traditional signal processing techniques and domain knowledge can result in hybrid approaches that combine the strengths of both approaches, improving interpretability and performance.

By pursuing these future research directions, the field of bearing fault classification using machine learning and deep learning algorithms can advance, leading to more accurate, efficient, and reliable methods for early fault detection and predictive maintenance in industrial applications.

8.6 CONCLUSION

To conclude, the research study has made significant contributions to the field of bearing fault diagnosis by showcasing the efficacy of machine learning and deep learning algorithms in accurately classifying fault conditions. The study addresses important limitations, such as the need for larger datasets and challenges related to interpretability, ensuring a comprehensive analysis. Moving forward, the future scope of research includes expanding the dataset to encompass diverse scenarios, exploring ensemble methods for improved performance, investigating transfer learning to overcome data limitations, implementing real-time systems for proactive maintenance, and enhancing interpretability for better understanding of the models' decision-making. Overall, this research provides valuable insights that can aid practitioners and researchers in developing effective predictive maintenance strategies, ultimately enhancing the reliability and efficiency of industrial machinery.

Appendix I: Source Code Data

The full code related to this Research is Publicly available at this link https://github.com/Devnitc18/MTech-Project-of-bearing-fault-detection/tree/main.

ML AND DL MODELS CODE

Source code for Multiclass Logistic Regression:

```
In [22]: import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split, GridSearchCV
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix, accuracy_score

         data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
         data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])


         # Remove collinear features
         corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
         upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
         to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
         data_wav_energy.drop(to_drop, axis=1, inplace=True)

         train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'], random_st

         scaler = StandardScaler()
         train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
         test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

         parameters = {'C': [50,60,70,80,90]}
         logistic_clf = LogisticRegression(max_iter=100, n_jobs=-1, random_state=0)
         grid_search = GridSearchCV(logistic_clf, parameters, cv=5)
         grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

         best_params = grid_search.best_params_
         print("Best parameters:", best_params)

         best_model = grid_search.best_estimator_

         train_predictions = best_model.predict(train_wav_energy_scaled)
         test_predictions = best_model.predict(test_wav_energy_scaled)

         train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
         test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)

         train_accuracy = accuracy_score(train_wav_energy['fault'], train_predictions)
         print("Overall training accuracy:", train_accuracy)

         test_accuracy = accuracy_score(test_wav_energy['fault'], test_predictions)
         print("Overall test accuracy:", test_accuracy)
```

Source Code for LDA

```
In [8]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.metrics import confusion_matrix, accuracy_score

        data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
        data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

        # Remove collinear features
        corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
        upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
        to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
        data_wav_energy.drop(to_drop, axis=1, inplace=True)

        train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'],

        scaler = StandardScaler()
        train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
        test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

        parameters = {'solver': [ 'svd', 'lsqr', 'eigen' ]}
        lda_clf = LinearDiscriminantAnalysis()
        grid_search = GridSearchCV(lda_clf, parameters, cv=6)
        grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

        best_params = grid_search.best_params_
        print("Best parameters:", best_params)

        best_model = grid_search.best_estimator_

        train_predictions = best_model.predict(train_wav_energy_scaled)
        test_predictions = best_model.predict(test_wav_energy_scaled)

        train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
        test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)

        train_accuracy = accuracy_score(train_wav_energy['fault'], train_predictions)
        print("Overall training accuracy:", train_accuracy)

        test_accuracy = accuracy_score(test_wav_energy['fault'], test_predictions)
        print("Overall test accuracy:", test_accuracy)
```

Source code for QDA

```
In [4]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
        from sklearn.metrics import confusion_matrix, accuracy_score
        from statsmodels.stats.outliers_influence import variance_inflation_factor

        data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
        data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

        # Remove collinear features
        corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
        upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
        to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
        data_wav_energy.drop(to_drop, axis=1, inplace=True)

        train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'],

        scaler = StandardScaler()
        train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
        test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

        parameters = {'reg_param': [0.0, 0.1, 0.5, 1.0]}
        qda_clf = QuadraticDiscriminantAnalysis()
        grid_search = GridSearchCV(qda_clf, parameters, cv=5)
        grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

        best_params = grid_search.best_params_
        print("Best parameters:", best_params)

        best_model = grid_search.best_estimator_

        train_predictions = best_model.predict(train_wav_energy_scaled)
        test_predictions = best_model.predict(test_wav_energy_scaled)

        train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
        test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)

        train_accuracy = accuracy_score(train_wav_energy['fault'], train_predictions)
        print("Overall training accuracy:", train_accuracy)

        test_accuracy = accuracy_score(test_wav_energy['fault'], test_predictions)
```

Source code for KNN

In [5]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import warnings

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])


warnings.filterwarnings("ignore", category=FutureWarning)

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)

train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'], rand

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

parameters = {'n_neighbors': [3, 5, 7]}
knn_clf = KNeighborsClassifier()
grid_search = GridSearchCV(knn_clf, parameters, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_model = grid_search.best_estimator_

train_predictions = best_model.predict(train_wav_energy_scaled)
test_predictions = best_model.predict(test_wav_energy_scaled)

train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)

train_accuracy = accuracy_score(train_wav_energy['fault'], train_predictions)
```

Source Code for SVM

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)

train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

parameters = {'C': [0.1, 1, 10], 'gamma': [0.1, 1, 10]}
svm_clf = SVC(probability=True)
grid_search = GridSearchCV(svm_clf, parameters, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_model = grid_search.best_estimator_

train_predictions = best_model.predict(train_wav_energy_scaled)
test_predictions = best_model.predict(test_wav_energy_scaled)

train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)

train_accuracy = accuracy_score(train_wav_energy['fault'], train_predictions)
print("Overall training accuracy:", train_accuracy)

test_accuracy = accuracy_score(test_wav_energy['fault'], test_predictions)
print("Overall test accuracy:", test_accuracy)
```

Source Code for DT

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)


train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

decision_tree_clf = DecisionTreeClassifier()

grid_search = GridSearchCV(decision_tree_clf, param_grid, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_decision_tree_clf = grid_search.best_estimator_

train_predictions = best_decision_tree_clf.predict(train_wav_energy_scaled)
test_predictions = best_decision_tree_clf.predict(test_wav_energy_scaled)
```

Source Code for Bagging

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)

train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'],

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

base_classifier = LogisticRegression(max_iter=200, n_jobs=-1)

param_grid = {
    'n_estimators': [5, 10, 15],
    'max_samples': [0.5, 0.7, 0.9]
}

bagging_clf = BaggingClassifier(base_estimator=base_classifier)
grid_search = GridSearchCV(bagging_clf, param_grid, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_bagging_clf = grid_search.best_estimator_

train_predictions = best_bagging_clf.predict(train_wav_energy_scaled)
test_predictions = best_bagging_clf.predict(test_wav_energy_scaled)

train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
```

Source Code for Boosting

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])00

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)

train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy['fault'

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

param_grid = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 1.0]
}

gradient_boosting_clf = GradientBoostingClassifier(random_state=324)

grid_search = GridSearchCV(gradient_boosting_clf, param_grid, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_gradient_boosting_clf = grid_search.best_estimator_

train_predictions = best_gradient_boosting_clf.predict(train_wav_energy_scaled)
test_predictions = best_gradient_boosting_clf.predict(test_wav_energy_scaled)

train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
test_confu_matrix = confusion_matrix(test_wav_energy['fault'], test_predictions)
```

Source Code for RF

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

data_wav_energy = pd.read_csv("data_feature_time_48k_2048_load_1.csv")
data_wav_energy['fault'] = pd.Categorical(data_wav_energy['fault'])

# Remove collinear features
corr_matrix = data_wav_energy.iloc[:, :-1].corr().abs()
upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape, dtype=bool), k=1))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.9)]
data_wav_energy.drop(to_drop, axis=1, inplace=True)

train_wav_energy, test_wav_energy = train_test_split(data_wav_energy, test_size=700, stratify=data_wav_energy|

scaler = StandardScaler()
train_wav_energy_scaled = scaler.fit_transform(train_wav_energy.iloc[:, :-1])
test_wav_energy_scaled = (test_wav_energy.iloc[:, :-1].values - scaler.mean_) / np.sqrt(scaler.var_)

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

random_forest_clf = RandomForestClassifier(random_state=324)

grid_search = GridSearchCV(random_forest_clf, param_grid, cv=5)
grid_search.fit(train_wav_energy_scaled, train_wav_energy['fault'])

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_random_forest_clf = grid_search.best_estimator_

train_predictions = best_random_forest_clf.predict(train_wav_energy_scaled)
test_predictions = best_random_forest_clf.predict(test_wav_energy_scaled)

train_confu_matrix = confusion_matrix(train_wav_energy['fault'], train_predictions)
```

Source Code for ANN for Features Data

In [7]:
```
# Convert labels to categorical data
train_data = train_data.iloc[:,0:9]
test_data = test_data.iloc[:,0:9]
# Scale the data
train_data = preprocessing.scale(train_data)
test_data = preprocessing.scale(test_data)
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

In [8]:
```
print(train_data.shape, test_data.shape, train_labels.shape, test_labels.shape)
```

```
(4000, 9) (1600, 9) (4000, 4) (1600, 4)
```

In [9]:
```
model = Sequential([
    layers.Dense(5,activation = 'relu', input_dim = 9),
#     layers.Dropout(0.2),
    layers.Dense(5,activation = 'relu'),
    layers.Dense(5,activation = 'relu'),
    layers.Dense(4,activation = 'softmax')
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 50

 dense_1 (Dense)             (None, 5)                 30

 dense_2 (Dense)             (None, 5)                 30

 dense_3 (Dense)             (None, 4)                 24

=================================================================
Total params: 134
Trainable params: 134
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```
model.compile(optimizer = 'adam',loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

In [11]:
```
callbacks = [EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 5)]
```

Source Code for ANN for Raw data

In [10]:
```
# Define the ANN model
class ANN():
    def __init__(self):
        self.model = self.create_model()

    def create_model(self):
        model = models.Sequential([
            layers.Dense(128, activation='relu'),
            layers.Dense(64, activation='relu'),
            layers.Dense(10, activation='softmax')
        ])
        model.compile(optimizer='adam',
                      loss=tf.keras.losses.CategoricalCrossentropy(),
                      metrics=['accuracy'])
        return model

# Train the ANN model
ANN_model = ANN()
history = ANN_model.model.fit(X_ANN_train, y_ANN_train, verbose=1, epochs=12)

# Evaluate the accuracy of the model on the training set
ANN_train_loss, ANN_train_accuracy = ANN_model.model.evaluate(X_ANN_train, y_ANN_train)
print('ANN train accuracy:', ANN_train_accuracy)

# Evaluate the accuracy of the model on the test set
ANN_test_loss, ANN_test_accuracy = ANN_model.model.evaluate(X_ANN_test, y_ANN_test)
print('ANN test accuracy:', ANN_test_accuracy)
```

Source Code for CNN 1d

In [11]:
```
# 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_1D, Y_CNN, train_size=0.75,test_size=0.25, random_s

# 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[train], verbose=1, epochs=12)

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

Source Code for CNN 2d

In [15]:
```
# 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_2D, Y_CNN, train_size=0.75,test_size=0.25, ran

# 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), padding ='same',activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=32, kernel_size=(3,3),strides=(2,2), padding ='same',activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=64, kernel_size=(3,3),strides=(2,2),padding ='same', activation='relu'),
        layers.MaxPool2D(pool_size=(2,2), padding='same'),
        layers.Conv2D(filters=128, kernel_size=(3,3),strides=(2,2),padding ='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[train], verbose=1, epochs=12)

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

Source code for LSTM model

In [17]:
```
# Reshape the data - 1 dimensional feed
Input = X.reshape([-1,1681,1])

# Test-Train Split
X_train, X_test, y_train, y_test = train_test_split(Input, Y_CNN, train_size=0.75,test_size=0.25, random_state=101)

# Define the LSTM Classification model
class LSTM_Model():
  def __init__(self):
    self.model = self.CreateModel()

  def CreateModel(self):
    model = models.Sequential([
        layers.LSTM(32, return_sequences=True),
        layers.Flatten(),
        layers.Dense(10),
        layers.Softmax()
        ])
    model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy'])
    return model


accuracy = []

# Train the model
for train, test in kfold.split(X_train,y_train):
  Classification = LSTM_Model()
  history = Classification.model.fit(X_train[train], y_train[train], verbose=1, epochs=10, use_multiprocessing=True)

  # Evaluate the accuracy of the model on the training set
  kf_loss, kf_accuracy = Classification.model.evaluate(X_train[test], y_train[test])
  accuracy.append(kf_accuracy)

LSTM_train_accuracy = np.average(accuracy)*100
print('LSTM train accuracy =', LSTM_train_accuracy)


# Evaluate the accuracy of the model on the test set
LSTM_test_loss, LSTM_test_accuracy = Classification.model.evaluate(X_test, y_test)
LSTM_test_accuracy*=100
print('LSTM test accuracy =', LSTM_test_accuracy)
```

Source Code for CNN-LSTM model

In [9]:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Reshape, Conv1D, MaxPooling1D, LSTM, Dropout, Dense, multiply
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

def built_model():
    input_seq = Input(shape=(250,))
    X = Reshape((250, 1))(input_seq)

    # encoder1
    ec1_layer1 = Conv1D(filters=50, kernel_size=20, strides=2,
                        padding='valid', activation='tanh')(X)
    ec1_layer2 = Conv1D(filters=30, kernel_size=10, strides=2,
                        padding='valid', activation='tanh')(ec1_layer1)
    ec1_outputs = MaxPooling1D(pool_size=2, strides=None, padding='valid')(ec1_layer2)

    # encoder2
    ec2_layer1 = Conv1D(filters=50, kernel_size=6, strides=1,
                        padding='valid', activation='tanh')(X)
    ec2_layer2 = Conv1D(filters=40, kernel_size=6, strides=1,
                        padding='valid', activation='tanh')(ec2_layer1)
    ec2_layer3 = MaxPooling1D(pool_size=2, strides=None, padding='valid')(ec2_layer2)
    ec2_layer4 = Conv1D(filters=30, kernel_size=6, strides=1,
                        padding='valid', activation='tanh')(ec2_layer3)
    ec2_layer5 = Conv1D(filters=30, kernel_size=6, strides=2,
                        padding='valid', activation='tanh')(ec2_layer4)
    ec2_outputs = MaxPooling1D(pool_size=2, strides=None, padding='valid')(ec2_layer5)

    encoder = multiply([ec1_outputs, ec2_outputs])

    dc_layer1 = LSTM(60, return_sequences=True)(encoder)
    dc_layer2 = LSTM(60)(dc_layer1)
    dc_layer3 = Dropout(0.5)(dc_layer2)
    dc_layer4 = Dense(10, activation='softmax')(dc_layer3)

    model = Model(input_seq, dc_layer4)

    return model
```

References

[1] C. Sun, M. Ma, Z. Zhao, and X. Chen, "Sparse Deep Stacking Network for Fault Diagnosis of Motor," *IEEE Trans Industr Inform*, vol. 14, no. 7, pp. 3261–3270, 2018, doi: 10.1109/TII.2018.2819674.

[2] M. Zhao, M. Kang, B. Tang, and M. Pecht, "Multiple Wavelet Coefficients Fusion in Deep Residual Networks for Fault Diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4696–4706, 2019, doi: 10.1109/TIE.2018.2866050.

[3] X. Li, W. Zhang, and Q. Ding, "Cross-Domain Fault Diagnosis of Rolling Element Bearings Using Deep Generative Neural Networks," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5525–5534, 2019, doi: 10.1109/TIE.2018.2868023.

[4] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li, "Deep Convolutional Transfer Learning Network: A New Method for Intelligent Fault Diagnosis of Machines With Unlabeled Data," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 9, pp. 7316–7325, 2019, doi: 10.1109/TIE.2018.2877090.

[5] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, and T. Zhang, "Deep Model Based Domain Adaptation for Fault Diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2296–2305, 2017, doi: 10.1109/TIE.2016.2627020.

[6] W. Mao, Y. Liu, L. Ding, and Y. Li, "Imbalanced Fault Diagnosis of Rolling Bearing Based on Generative Adversarial Network: A Comparative Study," *IEEE Access*, vol. 7, pp. 9515–9530, 2019, doi: 10.1109/ACCESS.2018.2890693.

[7] Y. O. Lee, J. Jo, and J. Hwang, "Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 3248–3253. doi: 10.1109/BigData.2017.8258307.

[8] J. Deutsch and D. He, "Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components," *IEEE Trans Syst Man Cybern Syst*, vol. 48, no. 1, pp. 11–20, 2018, doi: 10.1109/TSMC.2017.2697842.

[9] H. Oh, J. H. Jung, B. C. Jeon, and B. D. Youn, "Scalable and Unsupervised Feature Engineering Using Vibration-Imaging and Deep Learning for Rotor System Diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3539–3549, 2018, doi: 10.1109/TIE.2017.2752151.

[10] H. Shao, H. Jiang, H. Zhang, and T. Liang, "Electric Locomotive Bearing Fault Diagnosis Using a Novel Convolutional Deep Belief Network," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2727–2736, 2018, doi: 10.1109/TIE.2017.2745473.

[11] D. Tang Hoang and H. Jun Kang, "Deep Belief Network and Dempster-Shafer Evidence Theory for Bearing Fault Diagnosis," in *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*, 2018, pp. 841–846. doi: 10.1109/ISIE.2018.8433778.

[12] S. Dong, Z. Zhang, G. Wen, S. Dong, Z. Zhang, and G. Wen, "Design and application of unsupervised convolutional neural networks integrated with deep belief networks for mechanical fault diagnosis," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017, pp. 1–7. doi: 10.1109/PHM.2017.8079169.

[13] Z. Chen and W. Li, "Multisensor Feature Fusion for Bearing Fault Diagnosis Using Sparse Autoencoder and Deep Belief Network," *IEEE Trans Instrum Meas*, vol. 66, no. 7, pp. 1693–1702, 2017, doi: 10.1109/TIM.2017.2669947.

[14] C. Li, W. Zhang, G. Peng, and S. Liu, "Bearing Fault Diagnosis Using Fully-Connected Winner-Take-All Autoencoder," *IEEE Access*, vol. 6, pp. 6103–6115, 2018, doi: 10.1109/ACCESS.2017.2717492.

[15] W. Mao, S. Tian, X. Liang, and J. He, "Online Bearing Fault Diagnosis using Support Vector Machine and Stacked Auto-Encoder," in *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2018, pp. 1–7. doi: 10.1109/ICPHM.2018.8448775.

[16] J. Sun, C. Yan, and J. Wen, "Intelligent Bearing Fault Diagnosis Method Combining Compressed Data Acquisition and Deep Learning," *IEEE Trans Instrum Meas*, vol. 67, no. 1, pp. 185–195, 2018, doi: 10.1109/TIM.2017.2759418.

[17] F. Wang, B. Dun, G. Deng, H. Li, and Q. Han, "A deep neural network based on kernel function and auto-encoder for bearing fault diagnosis," in *2018 IEEE International*

Instrumentation and Measurement Technology Conference (I2MTC), 2018, pp. 1–6. doi: 10.1109/I2MTC.2018.8409574.

[18] X. Ding and Q. He, "Energy-Fluctuated Multiscale Feature Learning With Deep ConvNet for Intelligent Spindle Bearing Fault Diagnosis," *IEEE Trans Instrum Meas*, vol. 66, no. 8, pp. 1926–1935, 2017, doi: 10.1109/TIM.2017.2674738.

[19] L. Guo, Y. Lei, N. Li, and S. Xing, "Deep convolution feature learning for health indicator construction of bearings," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017, pp. 1–6. doi: 10.1109/PHM.2017.8079167.

[20] J. Pan, Y. Zi, J. Chen, Z. Zhou, and B. Wang, "LiftingNet: A Novel Deep Learning Network With Layerwise Feature Learning From Noisy Mechanical Data for Fault Classification," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 6, pp. 4973–4982, 2018, doi: 10.1109/TIE.2017.2767540.

[21] W. Zhang, F. Zhang, W. Chen, Y. Jiang, and D. Song, "Fault State Recognition of Rolling Bearing Based Fully Convolutional Network," *Comput Sci Eng*, vol. 21, no. 5, pp. 55–63, 2019, doi: 10.1109/MCSE.2018.110113254.

[22] L. Wen, X. Li, L. Gao, and Y. Zhang, "A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5990–5998, 2018, doi: 10.1109/TIE.2017.2774777.

[23] M. Xia, T. Li, L. Xu, L. Liu, and C. W. de Silva, "Fault Diagnosis for Rotating Machinery Using Multiple Sensors and Convolutional Neural Networks," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 101–110, 2018, doi: 10.1109/TMECH.2017.2728371.

[24] R. Liu, G. Meng, B. Yang, C. Sun, and X. Chen, "Dislocated Time Series Convolutional Neural Architecture: An Intelligent Fault Diagnosis Approach for Electric Machine," *IEEE Trans Industr Inform*, vol. 13, no. 3, pp. 1310–1320, 2017, doi: 10.1109/TII.2016.2645238.

[25] Q. Fu, B. Jing, P. He, S. Si, and Y. Wang, "Fault Feature Selection and Diagnosis of Rolling Bearings Based on EEMD and Optimized Elman_AdaBoost Algorithm," *IEEE Sens J*, vol. 18, no. 12, pp. 5024–5034, 2018, doi: 10.1109/JSEN.2018.2830109.

[26] X. Yu, F. Dong, E. Ding, S. Wu, and C. Fan, "Rolling Bearing Fault Diagnosis Using Modified LFDA and EMD With Sensitive Feature Selection," *IEEE Access*, vol. 6, pp. 3715–3730, 2018, doi: 10.1109/ACCESS.2017.2773460.

[27] D. He, R. Li, and J. Zhu, "Plastic Bearing Fault Diagnosis Based on a Two-Step Data Mining Approach," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 3429–3440, 2013, doi: 10.1109/TIE.2012.2192894.

[28] J. Tian, M. H. Azarian, M. Pecht, G. Niu, and C. Li, "An ensemble learning-based fault diagnosis method for rotating machinery," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017, pp. 1–6. doi: 10.1109/PHM.2017.8079125.

[29] R. Razavi-Far, M. Farajzadeh-Zanjani, and M. Saif, "An Integrated Class-Imbalanced Learning Scheme for Diagnosing Bearing Defects in Induction Motors," *IEEE Trans Industr Inform*, vol. 13, no. 6, pp. 2758–2769, 2017, doi: 10.1109/TII.2017.2755064.

[30] H. Xue, M. Wang, Z. Li, and P. Chen, "Fault feature extraction based on artificial hydrocarbon network for sealed deep groove ball bearings of in-wheel motor," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017, pp. 1–5. doi: 10.1109/PHM.2017.8079189.

[31] J. Yu, "Local and Nonlocal Preserving Projection for Bearing Defect Classification and Performance Assessment," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 5, pp. 2363–2376, 2012, doi: 10.1109/TIE.2011.2167893.

[32] B. Zhang, C. Sconyers, C. Byington, R. Patrick, M. E. Orchard, and G. Vachtsevanos, "A Probabilistic Fault Detection Approach: Application to Bearing Fault Detection," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 2011–2018, 2011, doi: 10.1109/TIE.2010.2058072.

[33] B. Yao, P. Zhen, L. Wu, and Y. Guan, "Rolling Element Bearing Fault Diagnosis Using Improved Manifold Learning," *IEEE Access*, vol. 5, pp. 6027–6035, 2017, doi: 10.1109/ACCESS.2017.2693379.

[34] M. D. Prieto, G. Cirrincione, A. G. Espinosa, J. A. Ortega, and H. Henao, "Bearing Fault Detection by a Novel Condition-Monitoring Scheme Based on Statistical-Time Features and Neural Networks," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 3398–3407, 2013, doi: 10.1109/TIE.2012.2219838.

[35] T. Yang, H. Pen, Z. Wang, and C. S. Chang, “Feature Knowledge Based Fault Detection of Induction Motors Through the Analysis of Stator Current Data,” *IEEE Trans Instrum Meas*, vol. 65, no. 3, pp. 549–558, 2016, doi: 10.1109/TIM.2015.2498978.
[36] Z. Wang, Q. Zhang, J. Xiong, M. Xiao, G. Sun, and J. He, “Fault Diagnosis of a Rolling Bearing Using Wavelet Packet Denoising and Random Forests,” *IEEE Sens J*, vol. 17, no. 17, pp. 5581–5588, 2017, doi: 10.1109/JSEN.2017.2726011.
[37] X. Jin, M. Zhao, T. W. S. Chow, and M. Pecht, “Motor Bearing Fault Diagnosis Using Trace Ratio Linear Discriminant Analysis,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 5, pp. 2441–2451, 2014, doi: 10.1109/TIE.2013.2273471.
[38] W. Qian, S. Li, and J. Wang, “A New Transfer Learning Method and its Application on Rotating Machine Fault Diagnosis Under Variant Working Conditions,” *IEEE Access*, vol. 6, pp. 69907–69917, 2018, doi: 10.1109/ACCESS.2018.2880770.
[39] R. Zhang, H. Tao, L. Wu, and Y. Guan, “Transfer Learning With Neural Networks for Bearing Fault Diagnosis in Changing Working Conditions,” *IEEE Access*, vol. 5, pp. 14347–14357, 2017, doi: 10.1109/ACCESS.2017.2720965.
[40] Q. Hu, A. Qin, Q. Zhang, J. He, and G. Sun, “Fault Diagnosis Based on Weighted Extreme Learning Machine With Wavelet Packet Decomposition and KPCA,” *IEEE Sens J*, vol. 18, no. 20, pp. 8472–8483, 2018, doi: 10.1109/JSEN.2018.2866708.
[41] Q. Tong *et al.*, “A Fault Diagnosis Approach for Rolling Element Bearings Based on RSGWPT-LCD Bilayer Screening and Extreme Learning Machine,” *IEEE Access*, vol. 5, pp. 5515–5530, 2017, doi: 10.1109/ACCESS.2017.2675940.
[42] W. Li, S. Zhang, and G. He, “Semisupervised Distance-Preserving Self-Organizing Map for Machine-Defect Detection and Classification,” *IEEE Trans Instrum Meas*, vol. 62, no. 5, pp. 869–879, 2013, doi: 10.1109/TIM.2013.2245180.
[43] X. Zhang, J. Kang, and T. Jin, “Degradation Modeling and Maintenance Decisions Based on Bayesian Belief Networks,” *IEEE Trans Reliab*, vol. 63, no. 2, pp. 620–633, 2014, doi: 10.1109/TR.2014.2315956.
[44] J. Wu, C. Wu, S. Cao, S. W. Or, C. Deng, and X. Shao, “Degradation Data-Driven Time-To-Failure Prognostics Approach for Rolling Element Bearings in Electrical Machines,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 529–539, 2019, doi: 10.1109/TIE.2018.2811366.
[45] A. S. Raj and N. Murali, “Early Classification of Bearing Faults Using Morphological Operators and Fuzzy Inference,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 2, pp. 567–574, 2013, doi: 10.1109/TIE.2012.2188259.
[46] A. Soualhi and S. Taleb, “Data fusion for fault severity estimation of ball bearings,” in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 2105–2110. doi: 10.1109/ICIT.2018.8352514.
[47] C. Chen, B. Zhang, and G. Vachtsevanos, “Prediction of Machine Health Condition Using Neuro-Fuzzy and Bayesian Algorithms,” *IEEE Trans Instrum Meas*, vol. 61, no. 2, pp. 297–306, 2012, doi: 10.1109/TIM.2011.2169182.
[48] F. Ben Abid, S. Zgarni, and A. Braham, “Distinct Bearing Faults Detection in Induction Motor by a Hybrid Optimized SWPT and aiNet-DAG SVM,” *IEEE Transactions on Energy Conversion*, vol. 33, no. 4, pp. 1692–1699, 2018, doi: 10.1109/TEC.2018.2839083.
[49] S. E. Pandarakone, Y. Mizuno, and H. Nakamura, “Evaluating the Progression and Orientation of Scratches on Outer-Raceway Bearing Using a Pattern Recognition Method,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1307–1314, 2019, doi: 10.1109/TIE.2018.2833025.
[50] M. Elforjani and S. Shanbr, “Prognosis of Bearing Acoustic Emission Signals Using Supervised Machine Learning,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5864–5871, 2018, doi: 10.1109/TIE.2017.2767551.
[51] W. Song and J. Xiang, “A Method Using Numerical Simulation and Support Vector Machine to Detect Faults in Bearings,” in *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, 2017, pp. 603–607. doi: 10.1109/SDPC.2017.118.
[52] B. R. Nayana and P. Geethanjali, “Analysis of Statistical Time-Domain Features Effectiveness in Identification of Bearing Faults From Vibration Signal,” *IEEE Sens J*, vol. 17, no. 17, pp. 5618–5625, 2017, doi: 10.1109/JSEN.2017.2727638.
[53] S. Kang, L. Cui, Y. Wang, F. Li, and V. I. Mikulovich, “Method of assessing the multi-state of a rolling bearing based on CFOA-HSVM two measures combination,” in *2017*

Prognostics and System Health Management Conference (PHM-Harbin), 2017, pp. 1–5. doi: 10.1109/PHM.2017.8079180.

[54] F. Ben Abid, S. Zgarni, and A. Braham, "Bearing fault detection of induction motor using SWPT and DAG support vector machines," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 1476–1481. doi: 10.1109/IECON.2016.7793237.

[55] H. Hassani, J. Zarei, M. M. Arefi, and R. Razavi-Far, "zSlices-Based General Type-2 Fuzzy Fusion of Support Vector Machines With Application to Bearing Fault Detection," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 7210–7217, 2017, doi: 10.1109/TIE.2017.2688963.

[56] S. E. Pandarakone, Y. Mizuno, and H. Nakamura, "Distinct Fault Analysis of Induction Motor Bearing Using Frequency Spectrum Determination and Support Vector Machine," *IEEE Trans Ind Appl*, vol. 53, no. 3, pp. 3049–3056, 2017, doi: 10.1109/TIA.2016.2639453.

[57] M. Kang, J. Kim, J.-M. Kim, A. C. C. Tan, E. Y. Kim, and B.-K. Choi, "Reliable Fault Diagnosis for Low-Speed Bearings Using Individually Trained Support Vector Machines With Kernel Discriminative Feature Analysis," *IEEE Trans Power Electron*, vol. 30, no. 5, pp. 2786–2797, 2015, doi: 10.1109/TPEL.2014.2358494.

[58] K. Teotrakool, M. J. Devaney, and L. Eren, "Bearing Fault Detection in Adjustable Speed Drives via a Support Vector Machine with Feature Selection using a Genetic Algorithm," in *2008 IEEE Instrumentation and Measurement Technology Conference*, 2008, pp. 1129–1133. doi: 10.1109/IMTC.2008.4547208.

[59] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak, "Fault prognostic of bearings by using support vector data description," in *2012 IEEE Conference on Prognostics and Health Management*, 2012, pp. 1–7. doi: 10.1109/ICPHM.2012.6299511.

[60] A. Rojas and A. K. Nandi, "Detection and Classification of Rolling-Element Bearing Faults using Support Vector Machines," in *2005 IEEE Workshop on Machine Learning for Signal Processing*, 2005, pp. 153–158. doi: 10.1109/MLSP.2005.1532891.

[61] T. W. Rauber, F. de Assis Boldt, and F. M. Varejão, "Heterogeneous Feature Models and Feature Selection Applied to Bearing Fault Diagnosis," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 637–646, 2015, doi: 10.1109/TIE.2014.2327589.

[62] M. M. Ettefagh, M. Ghaemi, and M. Yazdanian Asr, "Bearing fault diagnosis using hybrid genetic algorithm K-means clustering," in *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, 2014, pp. 84–89. doi: 10.1109/INISTA.2014.6873601.

[63] J. Tian, C. Morillo, M. H. Azarian, and M. Pecht, "Motor Bearing Fault Detection Using Spectral Kurtosis-Based Feature Extraction Coupled With K-Nearest Neighbor Distance Analysis," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 3, pp. 1793–1803, 2016, doi: 10.1109/TIE.2015.2509913.

[64] D. He, R. Li, J. Zhu, and M. Zade, "Data Mining Based Full Ceramic Bearing Fault Diagnostic System Using AE Sensors," *IEEE Trans Neural Netw*, vol. 22, no. 12, pp. 2022–2031, 2011, doi: 10.1109/TNN.2011.2169087.

[65] J. Harmouche, C. Delpha, and D. Diallo, "A global approach for the classification of bearing faults conditions using spectral features," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 7352–7357. doi: 10.1109/IECON.2013.6700356.

[66] M. Xia, F. Kong, and F. Hu, "An approach for bearing fault diagnosis based on PCA and multiple classifier fusion," in *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference*, 2011, pp. 321–325. doi: 10.1109/ITAIC.2011.6030215.

[67] S. Qian, X. Yang, J. Huang, and H. Zhang, "Application of new training method combined with feedforward artificial neural network for rolling bearing fault diagnosis," in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2016, pp. 1–6. doi: 10.1109/M2VIP.2016.7827265.

[68] M. Cococcioni, B. Lazzerini, and S. L. Volpi, "Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques," *IEEE Trans Industr Inform*, vol. 9, no. 4, pp. 2256–2263, 2013, doi: 10.1109/TII.2012.2231084.

[69] B. Li, M.-Y. Chow, Y. Tipsuwan, and J. C. Hung, “Neural-network-based motor rolling bearing fault diagnosis,” *IEEE Transactions on Industrial Electronics*, vol. 47, no. 5, pp. 1060–1069, 2000, doi: 10.1109/41.873214.

[70] R. R. Schoen, B. K. Lin, T. G. Habetler, J. H. Schlag, and S. Farag, “An unsupervised, on-line system for induction motor fault detection using stator current monitoring,” *IEEE Trans Ind Appl*, vol. 31, no. 6, pp. 1280–1286, 1995, doi: 10.1109/28.475698.

[71] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[72] M. -y. Chow, P. M. Mangum, and S. O. Yee, “A neural network approach to real-time condition monitoring of induction motors,” *IEEE Transactions on Industrial Electronics*, vol. 38, no. 6, pp. 448–453, 1991, doi: 10.1109/41.107100.

[73] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, “Deep Learning Algorithms for Bearing Fault Diagnostics - A Review,” in *2019 IEEE 12th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED)*, 2019, pp. 257–263. doi: 10.1109/DEMPED.2019.8864915.

[74] F. Filippetti, G. Franceschini, C. Tassoni, and P. Vas, “Recent developments of induction motor drives fault diagnosis using AI techniques,” *IEEE Transactions on Industrial Electronics*, vol. 47, no. 5, pp. 994–1004, 2000, doi: 10.1109/41.873207.

[75] M. A. Awadallah and M. M. Morcos, “Application of AI tools in fault diagnosis of electrical machines and drives-an overview,” *IEEE Transactions on Energy Conversion*, vol. 18, no. 2, pp. 245–251, 2003, doi: 10.1109/TEC.2003.811739.

[76] E. T. Esfahani, S. Wang, and V. Sundararajan, “Multisensor Wireless System for Eccentricity and Bearing Fault Detection in Induction Motors,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 818–826, 2014, doi: 10.1109/TMECH.2013.2260865.

[77] D. López-Pérez and J. Antonino-Daviu, “Application of Infrared Thermography to Failure Detection in Industrial Induction Motors: Case Stories,” *IEEE Trans Ind Appl*, vol. 53, no. 3, pp. 1901–1908, 2017, doi: 10.1109/TIA.2017.2655008.

[78] M. Blodt, P. Granjon, B. Raison, and G. Rostaing, “Models for Bearing Damage Detection in Induction Motors Using Stator Current Monitoring,” *IEEE Transactions on Industrial Electronics*, vol. 55, no. 4, pp. 1813–1822, 2008, doi: 10.1109/TIE.2008.917108.

[79] R. R. Schoen, T. G. Habetler, F. Kamran, and R. G. Bartfield, “Motor bearing damage detection using stator current monitoring,” *IEEE Trans Ind Appl*, vol. 31, no. 6, pp. 1274–1279, 1995, doi: 10.1109/28.475697.

[80] A. Ming, W. Zhang, Z. Qin, and F. Chu, “Dual-Impulse Response Model for the Acoustic Emission Produced by a Spall and the Size Evaluation in Rolling Element Bearings,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6606–6615, 2015, doi: 10.1109/TIE.2015.2463767.

[81] M. Kang, J. Kim, and J.-M. Kim, “An FPGA-Based Multicore System for Real-Time Bearing Fault Diagnosis Using Ultrasampling Rate AE Signals,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2319–2329, 2015, doi: 10.1109/TIE.2014.2361317.

[82] F. Immovilli, A. Bellini, R. Rubini, and C. Tassoni, “Diagnosis of Bearing Faults in Induction Machines by Vibration or Current Signals: A Critical Comparison,” *IEEE Trans Ind Appl*, vol. 46, no. 4, pp. 1350–1359, 2010, doi: 10.1109/TIA.2010.2049623.

[83] J. Harmouche, C. Delpha, and D. Diallo, “Improved Fault Diagnosis of Ball Bearings Based on the Global Spectrum of Vibration Signals,” *IEEE Transactions on Energy Conversion*, vol. 30, no. 1, pp. 376–383, 2015, doi: 10.1109/TEC.2014.2341620.

[84] “Report of Large Motor Reliability Survey of Industrial and Commercial Installations: Part 3,” *IEEE Trans Ind Appl*, vol. IA-23, no. 1, pp. 153–158, 1987, doi: 10.1109/TIA.1987.4504880.

[85] A. Zhang, S. Li, Y. Cui, W. Yang, R. Dong, and J. Hu, “Limited Data Rolling Bearing Fault Diagnosis With Few-Shot Learning,” *IEEE Access*, vol. 7, pp. 110895–110904, 2019, doi: 10.1109/ACCESS.2019.2934233.

[86] 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, 2011, doi: 10.1109/TIA.2010.2090839.