Automatic Data Processing Framework Based Deep Neural Network for Condition Monitoring

Ibrahim Alqatawneh

School of Computing and Engineering

This thesis is submitted to the School of Computing and Engineering, University of Huddersfield, in partial fulfilment of the requirements for the degree of Doctor of Philosophy

April 2021

Copyright Statement

- ✓ The author of this thesis owns any copyright in it (the "Copyright") and s/he has given The University of Huddersfield the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

- ✓ Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

- ✓ The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the "Intellectual Property Rights") and any reproductions of copyright works, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

Abstract

With the rapid growth of data volumes and complexity in the field of condition monitoring (CM) of machinery, the need to automate tasks such as information extraction and classification has become more important than ever. Artificial intelligence (AI) remains a promising solution to such challenging tasks. From a learning perspective, the majority of AI based shallow learning methods for CM have been applied for classification, whereas feature extraction task is still manually processed, which requires hand-crafted features based on expertise knowledge. Hence, the classification accuracy of the shallow learning method relies entirely on the quality of the extracted features. Contrariwise, AI based deep learning methods, and in particular the convolutional neural network (CNN) for CM have the capability to address the shortcomings of shallow learning methods by integrating both feature extraction and classification tasks into a single model. With deep CNN architecture, severe problems can appear caused by the activation function layer, which significantly affect the network performance, these include the vanishing gradient and dying ReLU problems.

To automate the task of data processing and achieve high classification accuracy for CM, an improved AI framework has been developed in this research: Firstly, a CNN architecture has been designed for automatic feature extraction and classification. CNN was chosen as it has been found to offer several benefits; it can be trained in a supervised learning manner, representative features are extracted directly from the raw data, data dimensionality can be reduced, and it can automatically identify different classes for a given data set. Secondly, to addresses the shortcoming of the existing activation functions, and enhance the learning ability of the network, a hybrid activation function has been developed called the Improved Rectified Linear Unit and Hyperbolic Tangent function (IReLU-Tanh). The developed framework has been implemented and evaluated using both simulated and experimental vibration data. The results shown that the developed CNN architecture with the proposed IReLU-Tanh yields robust classification with high diagnostic accuracy and outperforms the commonly used activation functions Tanh, ReLU, LReLU, and ELU.

Keywords: *Artificial intelligence, Deep learning, Convolutional neural network, Activation function, Condition monitoring.*

Declaration

I am the author of this thesis and to the best of my knowledge it contains no materials previously published or written by another person, I also declare that no portion of the presented work in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Acknowledgements

First and foremost, all praises and thanks to ALLAH for the strength and blessing in completing this work on time. ‘ALHAMDULILLAH’.

My special word of thanks to my mother, my father, my wife, brothers, and sisters for all their prayers, support, and encouragement.

I would like to express my sincere appreciation to my supervisor Professor Fengshou Gu and Professor Andrew Ball for their excellent guidance, and support during my studies.

Many thanks to all my friends who have supported and encouraged me all the time.

I am humbled and grateful to all who helped and supported me in my PhD journey that is coming to an end, and this will be the beginning of a new journey in my life.

LIST OF CONTENTS

List of Figures

List of Tables

List of Symbols

x_i	Time series signal
μ	Mean of the signal
$H[x(i)]$	Hilbert transform
$g(i)$	Averaged signal
c_n	Intrinsic mode function
$\vec{t}$	Target vector
w	Weight vector
f	Activation Function
b	Bias vector
$\tilde{x}_i$	Reconstructed signal
h	Hidden encoder
θ	Neural network parameters
E	Reconstruction error
η	Learning rate
$\hat{x}_i$	Normalised value
γ	Scaling parameter
β	Shifting parameter
q	Softmax function
f_p	Fault frequency
σ_B^2	Variance of the mini-batch training data
μ_B	Mean of the mini-batch training data
B	Mini-batch size of the training data
θ_{old}^l	Current neural network parameter at l^{th} layer
θ_{new}^l	Updated neural network parameter at l^{th} layer
α	Activation function parameter

List of Abbreviations

ANN	Artificial Neural Network
AI	Artificial Intelligence
CM	Condition Monitoring
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DBN	Deep Belief Network
DT	Decision tree
DAQ	Data Acquisition System
DCNN	Deep Convolutional Neural Network
ELU	Exponential Linear Unit
EMD	Empirical Model Decomposition
ELMD	Ensemble Local Mean Decomposition
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GD	Gradient Decent
IReLU-Tanh	Improved Rectified Linear Unit and Hyperbolic Tangent Function
KNN	K-Nearest Neighbour
LReLU	Leaky Rectified Linear Unit
LSTM	Long Short Term Memory
MED	Minimum Entropy Deconvolution
MOMEDA	Multipoint Optimal Minimum Entropy Deconvolution Adjusted
MKurt	Multipoint Kurtosis
MSE	Mean Square Error
1D-DCNN	One-Dimensional Deep Convolutional Neural Network
OVO	One-Versus-One
OVA	One-Versus-All
PG	Planetary Gearbox

PCA	Princple Component Analysis
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
RMS	Root Mean Square
SAE	Stacked auto-encoder
SVM	Support Vector Machine
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
Tanh	Hyperbolic Tangent Function
TSA	Time Synchronous Averaging
TP	True Positive
TN	True Negative
WVD	Wigner-Ville Distribution

List of Publications

- **Alqatawneh, I.,** Deng, R., Rabeyee, K., Chao, Z., Gu, F., & Ball, A. D. (2021, September). A Developed Convolutional Neural Network Architecture for Condition Monitoring. In 2021 26th International Conference on Automation and Computing (ICAC). (Accepted).
- **Alqatawneh, I.,** Rabeyee, K., Zhang, C., Feng, G., Gu, F., & Ball, A. D. (2020, April). A Modified Activation Function for Deep Convolutional Neural Network and Its Application to Condition Monitoring. In International Conference on Maintenance Engineering. Springer, Cham.
- **Alqatawneh, I.**, Kuosheng, J., Mones, Z., Zeng, Q., Gu, F., & Ball, A. D. (2018, September). Condition Monitoring and Fault Diagnosis Based on Multipoint Optimal Minimum Entropy Deconvolution Adjusted Technique. In 2018 24th International Conference on Automation and Computing (ICAC).
- Mones, Z., **Alqatawneh, I.**, Zhen, D., Gu, F., & Ball, A. D. (2019, September). Fault Diagnosis for Planetary Gearbox Using On-Rotor MEMS Sensor and EMD Analysis. In 2019 25th International Conference on Automation and Computing (ICAC).
- Alabied, S., Daraz, A., Rabeyee, K., **Alqatawneh, I.**, Gu, F., & Ball, A. D. (2019, September). Motor Current Signal Analysis Based on Machine Learning for Centrifugal Pump Fault Diagnosis. In 2019 25th International Conference on Automation and Computing (ICAC).
- Mones, Z., Zhen, D., **Alqatawneh, I.**, Zeng, Q., Gu, F., & Ball, A. D. (2018, September). Fault Diagnosis of Planetary Gearboxes via Processing the On-Rotor MEMS Accelerometer Signals. In 2018 24th International Conference on Automation and Computing (ICAC).

Chapter One: Introduction

This chapter presents the importance of condition monitoring for machinery. It starts with the different approaches to condition monitoring, including application based, data gathering based, and data processing based. Then, an overview is given of the data processing based approach including signal processing, data enhancement and artificial intelligence methods. The motivation for the research is explained, after which the research aim, objectives, and research methodology are presented. Finally, this chapter ends with an outline of the contents of the thesis.

1.1. Background

Data processing in the field of industrial applications has become very important due to the enormous amount of data generated by modern measurement systems that can be used for the condition monitoring (CM) of machines and instruments. CM is the process of monitoring and analysing the gathered data in a way that can be used to assess a system's health condition [1]. CM has gained increased importance in industrial applications to ensure reliability and system health condition. CM can be described as a data processing approach, which can be carried out through signal processing and artificial intelligence (AI) methods.

With the increased amount of data and its complexity, extracting meaningful information from a given dataset is becoming a challenging task. Hence, feature extraction and selection have attracted significant attention in recent years [2, 3]. Feature extraction and selection are generally employed in ways that depend on the application's goal, e.g., for dimensionality reduction, extracting representative features from the data, or selecting features that are expected to be the most relevant to the problem. However, traditional manual methods rely greatly on prior knowledge and expertise to extract and select meaningful information from a given dataset [4, 5]. Today, AI can be utilised to automate the task of extracting and selecting meaningful and informative information.

A branch of AI called deep learning has become a popular method as it integrates both feature extraction and classification tasks into a single model [6]. Deep learning yields automatic feature extraction directly from the raw data through a multi-layered structure. It is commonly referred to as a deep neural network (DNN), where multiple levels of hidden layers are stacked in the network architecture [7]. Several deep learning methods have been developed, such as the deep belief network (DBN), the convolutional neural network (CNN), the recurrent neural network (RNN), and the stacked auto-encoder (SAE) [8]. Amongst these, CNN is the most widely used as it achieves the state-of-the-art performance in various applications, including speech recognition, image processing, object detection, robotics, and automated fault diagnosis [9, 10]. CNN was originally developed for handwritten digit classification [11]. It integrates several features into a single deep hierarchical model including; feature extraction, dimensionality reduction, and classification. Integrating these features through a multi-layered configuration allows

the CNN to perform feature extraction and classification for CM [12]. However, with the deep CNN architecture, severe problems caused by the activation function layer begin to appear, which affects the training of the network to extract representative features from the raw data, and hence affects the classification accuracy of the model [13], as a results of using traditional activation functions such as the hyperbolic tangent function (Tanh), the rectified linear unit (ReLU), the leaky rectified linear unit (LReLU), and the exponential linear unit (ELU) [14]. Therefore, in this research, an automated approach based on the deep CNN is investigated for feature extraction and classification. In addition, a hybrid activation function is proposed for the deep CNN to address the shortcomings in the existing activation functions, enhance the learning ability of the network during the training process, and hence improve the overall performance of the network.

1.2. Condition Monitoring

CM generally comprises of three main steps as shown in Figure 1-1; data acquisition, data processing, and decision-making. Recently, a considerable number of researchers have extensively studied each step of the CM [15]. Consequently, a wide range of techniques and algorithms have been developed. This thesis will focus on the data processing task to automate the tasks of feature extraction and classification by adopting deep learning methods.

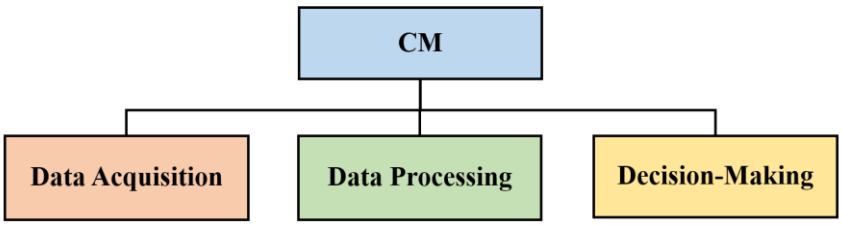

Figure 1-1: CM main steps

1.2.1. Data Acquisition

The data acquisition step is the process of collecting the raw data from sensors mounted on the monitored system [16]. A number of CM techniques have been developed for the data acquisition, including vibration monitoring, oil monitoring, The acquired data can be classified into two main categories [15, 17]:

- **Value type:** data collected and stored at a particulate time such as temperature, pressure, and oil analysis.
- **Waveform**: data collected and stored in a timely manner, as a time-series, such as acoustic and vibration data.

1.2.2. Data Processing

The data processing step involves analysis, interpretation, and extracting useful information from the measured data. Several types of data processing techniques have been developed over past decades for analysing the measured data [18]. The data processing step aims to provide an accurate description of the health of the system being monitored.

1.2.3. Decision-Making

The decision-making step will recommend an effective maintenance policy. Over recent years, several techniques have been introduced for decision-making. These techniques can be classified into the following groups; detection, diagnosis, and prognosis [19]. Detection refers to the process of detecting abnormalities in the collected data. Diagnosis refers to the process of identifying the location and the severity of the fault. Prognosis is the process of predicting the remaining useful life of the monitored system [20].

1.3. Condition Monitoring Approaches

Studying CM is generally carried out through the following three main approaches [21]; application based, data gathering based, and data processing based, as shown in Figure 1-2.

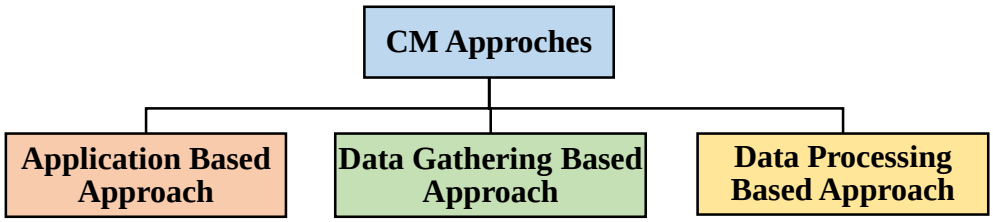

Figure 1-2: CM approaches

1.3.1. Application Based Approach

Application based approach focuses on different types of rotating machinery such as planetary gearboxes (PG), induction motors, bearings, etc. In this research, a PG is considered as a case study of rotating machinery.

1.3.2. Data Gathering Based Approach

The data gathering based approach focuses on developing instruments that are required for data sensing and storing, such as accelerometers, microphones, and wireless transmitters.

1.3.3. Data Processing Based Approach

The data processing based approach plays an important role in a CM study, and it can be classified into three main categories as shown in Figure 1-3, signal processing, data enhancement, and AI methods [22-24].

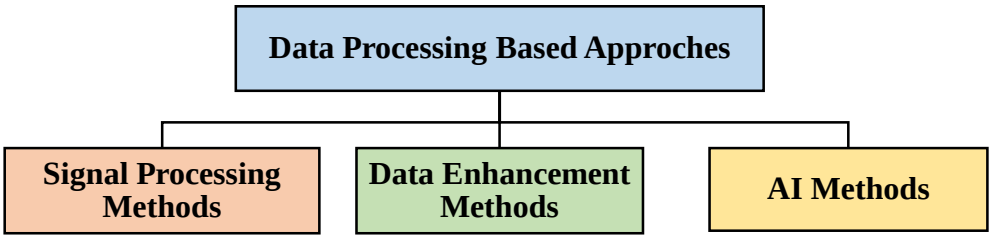

Figure 1-3: Data processing based approaches

Signal processing methods are used to process and analyse the monitored data. Data enhancement methods enhance the important features in the data and suppress noise. AI methods implement AI algorithms to automate the diagnostic procedure and minimize human interference. AI methods generally can be classified into two main groups, shallow and deep learning methods.

In this research, AI based deep learning is used for CM. The main purpose of adopting the deep learning method is to extract the representative features directly from the raw data and automatically determine the system's health condition.

1.4. Signal Processing Methods

In CM, a wide range of signal processing methods are based on the analysis of the time domain and spectrum of the monitored data. The signal processing methods applied can be categorised into the following three main groups [25]; time domain, frequency domain, and time-frequency domain analyses, as seen in Figure 1-4. Vibration signals generally contain massive amounts of information [26] and therefore a variety of signal processing techniques have been developed over past decades to highlight features of interest in the monitored signal.

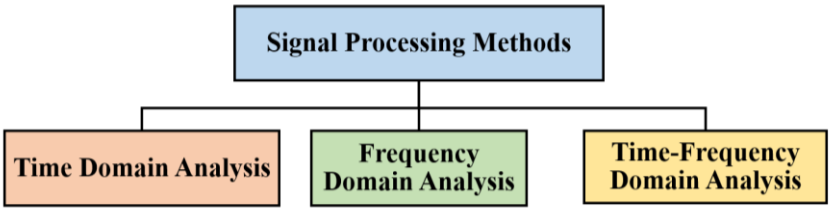

Figure 1-4: Signal processing methods

1.4.1. Time Domain Analysis

Time domain analysis is the process of analysing the vibration data as a function of time. It is employed to explore statistical features of the monitored vibration data [27]. The most common statistical features used for vibration data analysis are: peak value, kurtosis, skewness, crest factor, and root mean square (RMS). [28]. These statistical features are commonly referred to as time domain features. However, it has been reported that time domain analysis shows poor performance in fault diagnosis of rotating machinery [29].

Peak value is the maximum amplitude of the vibration data. The peak value can be expressed as [28]:

$$Peak\ Value = \max(x_i) \tag{1.1}$$

RMS is used to measure the signal energy of the vibration data, and it can be computed as [30]:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2} \tag{1.2}$$

where x_i is the acquired time-series signal and N is the total number of data points.

Kurtosis is employed to measure the impulsiveness of the vibration data, and it can be calculated as [30]:

$$Kurtosis = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^4}{[\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2]^2} \quad (1.3)$$

where μ is the mean of the signal.

1.4.2. Frequency Domain Analysis

Frequency domain analysis is the process of analysing the monitored data based on its frequency content, and is a widely used technique for vibration data analysis [31]. The primary technique of performing the frequency domain analysis is the Fast Fourier Transform (FFT). Using this technique, the raw vibration signal generated by rotating components is transformed from the time domain to the frequency domain [32]. The time domain signal is transformed into a spectrum which represents the frequency of each component [33]. In the frequency domain, the spectral components of the signal exhibits much more useful information regarding the details of rotating parts, which can be more beneficial for determining the system condition. However, rotating machinery such as gears generally produces complex vibration signals. Therefore, identifying gear defects by observing only the spectrum signals is difficult. Moreover, it has been reported that the FFT technique is not suitable for analysing a non-stationary signal, because it is based on the assumption of a stationary signal [25, 34].

- **Envelope Analysis**

Envelope analysis, also called demodulation analysis, can be used for fault diagnosis in rotating machinery where the fault exhibits an amplitude modulation (AM) that effects the characteristic frequencies of the machinery. In a PG system, multiple gears revolve around their own centre and rotate around a fixed sensor mounted on the PG housing. In this case, when the gear defect is rotating and passing through the fixed sensor, the maximum amplitude of the vibration is recorded, and when the gear defect moves away from the fixed sensor, the amplitude of the vibration gradually decreases, resulting in AM [35]. Envelope analysis can obtain the diagnostic fault features from the data. The envelope of the signal can be calculated as [36]:

$$a(t) = \sqrt{x^2(i) + H^2[x(i)]} \quad (1.4)$$

Where $H[x(i)]$ denotes the Hilbert transform of signal $x(i)$.

1.4.3. Time-Frequency Domain Analysis

The non-stationary characteristic of vibration data makes extracting a representative feature demanding. To overcome this drawback, several time-frequency domain analysis methods have been developed for processing non-stationary signal [37]. The Short Time Fourier Transform (STFT) is one of the well-known time-frequency domain analysis techniques. The STFT is based on the idea that the entire vibration data is divided into short time signals by using a window function and then each short time signal can be processed by applying a FFT [38]. However, it has a disadvantage regarding the time-frequency resolution, this is due to using a window function of fixed length [39]. The Wigner-Ville Distribution (WVD) is another time-frequency domain analysis technique developed to overcome the issue of time-frequency resolution. Other time-frequency domain analysis methods include wavelet transforms [39, 40].

1.5. Data Enhancement Methods

Data enhancement methods aim to enhance the important features in the vibration data and suppress the noise. Recently, it has been widely applied to CM such as time synchronous averaging (TSA), empirical mode decomposition (EMD), minimum entropy deconvolution (MED), and multipoint optimal minimum entropy deconvolution adjusted (MOMEDA).

1.6. Artificial Intelligence Methods

The concept of AI was firstly proposed by Turing [41] in 1950 and since its creation, it has played a major role in the creation of computers that function as close as possible to human brains. AI is a branch of computer science that is concerned with the automation of intelligent behaviour [42]. AI has made significant contributions to a variety of applications and it has led to innovations in robotics, computer vision, natural language processing, and speech recognition. Motivated by the outstanding performance of AI in the above domains, it has been widely used in the field of CM [43]

Automating the diagnostic procedure is gaining importance in industrial applications due to machinery systems are more complicated than ever before, and because the volume of data produced by modern machines has increased massively [44]. Hence, using AI for automatic CM can provide an accurate assessment of the system's health [45] and minimizes the negative effects of human interpretation such as it being time-consuming and dependent on well-trained labours [46]. Several AI methods have been applied for automatic fault diagnosis including: artificial neural network (ANN), support vector machine (SVM), decision tree (DT), k-nearest neighbour (KNN), CNN, DBN, and SAE [47, 48]. As shown in Figure 1-5, these methods can be divided into two main groups, shallow and deep learning methods [49].

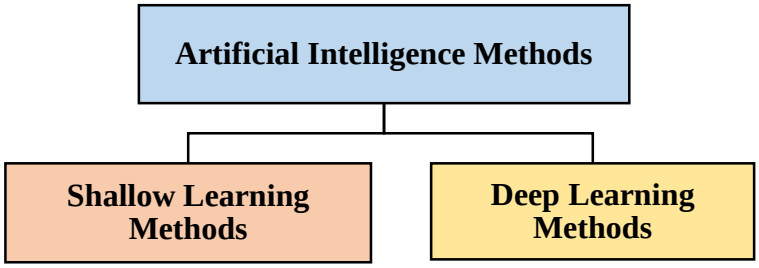

Figure 1-5: AI methods

AI methods generally involve three main steps: (1) Data acquisition is the process of collecting vibration data from the monitored system; (2) Feature extraction is the task of extracting representative features from the monitored vibration data; and (3) classification, where the extracted features are used to train AI algorithms [44, 50]. By adopting these procedures, the condition of the monitored system can be determined. Therefore, discovering valuable features in the observed data plays a vital role in training the AI approach [51].

1.6.1. Learning Methods

Training AI methods can be classified into three main learning paradigms: supervised learning, semi-supervised learning, and unsupervised learning [52], as shown in Figure 1-6. Each learning method is used for different tasks such as classification, regression, clustering.

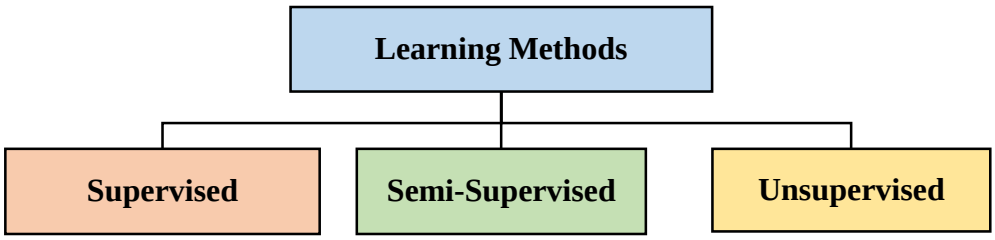

Figure 1-6: AI learning paradigms

1.6.1.1. Supervised Learning

The supervised learning method is a learning paradigm that uses labelled data, and the goal is to determine a good approximation between a set of input data and desired outputs. In supervised learning, the algorithm is trained with a set of training data and their corresponding labels, and during the training process the algorithm learns how to produce the desired output based on the given examples of input-output (labelled data) [52]. Supervised learning is used for classification tasks, where the training data belong to one of two possible classes (binary classification), or one of many possible classes (multi-class classification), or regression tasks, where the output consists of a continuous output variable [53].

1.6.1.2. Semi-Supervised Learning

Semi-supervised learning is another learning paradigm where partially labelled or unlabelled training data are available. In semi-supervised learning, the algorithm is trained with a few labelled data. It then uses the trained model to predict the remaining portion of unlabelled training data. After labelling the unlabelled training data, the algorithm is trained with the complete training data set, which comprises the few labelled data together with unlabelled data that has been labelled by the algorithm. Semi-supervised learning method is a combination of supervised and unsupervised learning methods [54].

1.6.1.3. Unsupervised Learning

Unsupervised learning draws inferences from the data itself without using any corresponding label (unlabelled data) [55]. Unsupervised learning is commonly used for clustering tasks, where the aim is to cluster or group similar data together; or for anomaly

detection, where the aim is to identify the abnormality in the data that is significantly different from the rest of the data; or dimensionality reduction, where the aim is to reduce the dimensionality of the data whilst preserving the most important features [51].

1.6.2. Classification

Classification is one of the widely studied tasks in AI. It refers to the process of building a model from historical labelled data to predict a target class for unseen data [56]. It is a supervised learning algorithm because it is driven by means of labelled data, where the input data are provided along with their corresponding labels. The classification model learns how to determine a good approximation function that maps the input training data to an output of two, or more than two classes [57]. Classification techniques have been widely used in various applications including; image classification, speech classification, and text classification. Classification generally can be categorized into two main types: binary and multi-class classifications [58].

1.6.2.1. Binary Classification

Binary classification is a type of classification technique that is used to map the input training data to a unique target class. In other words, binary classification involves classifying the training data into either one of two target classes [58], as shown in Figure 1-7. For example, binary classification can be used to predict the target class of the input training data x_1 whether it belongs to class 1 (Y_1) or class 2 (Y_2). Binary classification has been adopted for intelligent fault diagnosis, for instance Nguyen, Prosvirin [59] adopted a binary classification technique based on a SVM to classify the input training data to one of two target classes. Yuan, Ma [60] implement a binary classification technique to classify two types of conditions (baseline and faulty conditions) using a CNN.

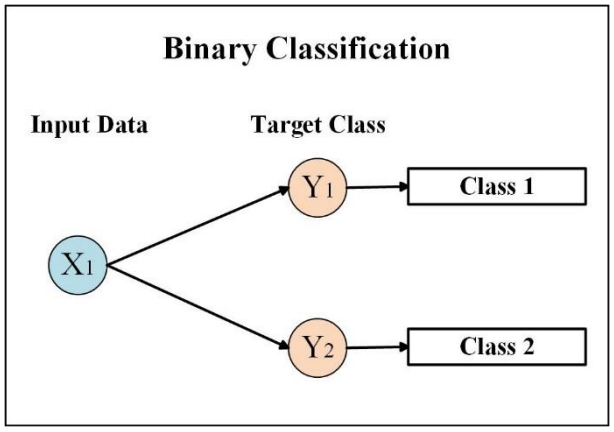

Figure 1-7: Example of binary classification

1.6.2.2. Multi-Class Classification

Multi-class classification is a way of classifying a set of input training data $[x_1, x_2, \dots x_n]$ to a set of target classes $[y_1, y_2, \dots y_n]$. Multi-class classification involves assigning each input of the training data to one of several classes [61], as shown in Figure 1-8. The multi-class classification has been widely implemented in the field of intelligent fault diagnosis, for instance, Tiwari, Bordoloi [62] implemented a multi-class classification technique to classify four types of vibration data under different conditions (baseline and three different types of fault). Sun, Yao [63] adopted a multi-class classification based on a deep CNN for intelligent fault diagnosis. In this study, the deep CNN is trained with four types of vibration data collected from the gearbox under different conditions. Then, the trained model is tested with unseen data to predict the target class for each data set. Chen, Liu [64] adopted a multi-class classification technique based on a CNN and SVM for intelligent fault diagnosis. In that study, the SVM was combined with the CNN, and placed at the end to classify the fault. The CNN model was trained with seven types of vibration data, where each data set represents different gearbox conditions. Then, the trained CNN model is evaluated to verify its efficiency in predicting the target class of unseen data.

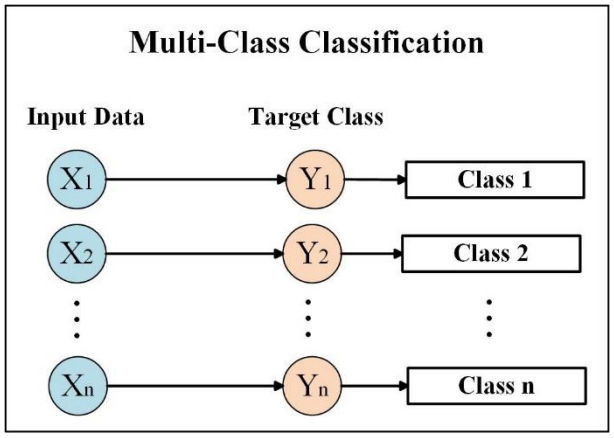

Figure 1-8: Example of multi-class classification

1.7. Research Motivation

Analysing experimental vibration signals collected from machinery system normally is not an easy task, because the measured signals are often non-stationary. Moreover, machinery systems generally produce complex vibration signals that contain strong background noise, making it difficult to extract the representative feature from the data. Several data processing techniques have been developed over the past decades to analyse non-stationary signals, however, these methods rely on having prior knowledge of the machinery system and require expertise in signal processing techniques to apply them successfully.

AI based shallow learning methods have attracted the attention of many researchers in the field of CM. However, the vast majority of AI based shallow learning methods are applied for classification, while the feature extraction task is still manually processed. Thus, the classification accuracy of shallow learning method relies entirely on the quality of the extracted features.

To address the above-mentioned shortcomings, developing an automated approach for CM based on deep CNN architecture is the main motivation of this research. The developed approach aims to automate the extraction of representative features directly from the raw data and then automatically identify different classes for the given set of data. Deep CNN integrates several features into a single model including; feature extraction, dimensionality reduction, and classification. Integrating these features into a deep model would enable CNN to automate feature extraction and classification, and provide a promising way of overcoming the above-mentioned obstacles.

Although, reasonable results have been obtained from the application of CNN to raw vibration data for CM, however, several issues have been reported such as the vanishing gradient problem and dying ReLU [13, 14, 65]. These issues are due to the influence of the activation function layer, which can affect the overall performance of CNN, and lead to a low classification accuracy. Therefore, in this study, a hybrid activation function will be proposed to address the shortcomings in the existing activation functions and to enhance the overall performance of the network. The automated approach based on the developed CNN architecture with the proposed activation function will be evaluated using both simulated and real experimental data.

1.8. Research Aim and Objectives

The main aim of this research is to develop an automated approach based deep CNN for machinery CM that will automate the tasks of feature extraction and classification. The second aim of this research, an activation function called improved rectified linear unit and hyperbolic tangent function (IReLU-Tanh) is proposed to enhance the learning ability of deep CNN architecture and improve the overall performance of the network.

To achieve the aims, the research will be carried out according to the following prioritized objectives:

Objective One: To review the existing data enhancement methods and automated approaches based on AI such as shallow and deep learning methods used for vibration data.

Objective Two: To investigate and evaluate the existing activation functions used for deep CNN.

Objective Three: To develop and implement an optimal CNN architecture for feature extraction and classification based on typical CM vibration data.

Objective Four: To develop an IReLU-Tanh function for deep CNN architecture.

Objective Five: To evaluate the performance of the developed CNN architecture with IReLU-Tanh function using simulated and experimental vibration data. This will include a systematic comparison of the performance against the existing activation functions.

1.9. Research Methodology

The research methodology identified for this thesis is typical computer science research methods, developed through simulation and real experimentation. Then, comparing the outcomes with existing solutions. The proposed approach consists of five work packages. The first work package is focusing on a literature review. The next three work packages are scientific research packages, which include algorithm development, data collection, implementation and evaluation. The last work package is focusing on the writing up of the thesis.

- **Work Package 1: Literature Review**

  This work package concentrates on reviewing the data enhancement and AI methods used for vibration data. It also reviews the existing activation functions used for deep CNNs. This work package will include the following milestones:

  (1) Review the existing data enhancement methods used for CM, and highlight the limitations of these methods (see Chapter Two).

  (2) Review the application of AI based shallow learning methods for intelligent fault diagnosis, including; ANN, SVM, KNN, and DT, and then highlight the shortcomings of the shallow learning methods (see Chapter Two).

  (3) Review the application of deep learning methods for automated fault diagnosis, including; deep SAE, RNN, and CNN for automatic feature extraction and classification (see Chapter Two).

  (4) Review the existing activation functions such as Tanh, ReLU, LReLU, ELU, and highlight their drawbacks when applied to deep CNN architecture (see Chapter Five).

- **Work Package 2: Algorithm Development (Framework)**

  This work package consists of two main tasks:

  (1) The first task is to develop an automated approach-based deep CNN for machinery CM. The automated approach aims to automate the tasks of feature extraction and classification. This task will include describing the fundamental concepts of the developed algorithm, including CNN architecture, learning algorithm and the implementation steps of CNN for CM (see Chapter Four).

  (2) The second task is to develop the IReLU-Tanh function for the deep CNN architecture, where the proposed function will address the limitations of existing

activation functions, and enhance the learning ability of deep CNN architecture. In this task, the advantages and disadvantages of each activation function are discussed in detail. It will then combine the advantages of two different functions (ReLU and Tanh) to create a hybrid IReLU-Tanh function (see Chapter Five).

- **Work Package 3: Data Collection**

  This work package is focused on the data collection process to be used to evaluate the performance of the developed automated approach. Experimental vibration data will be collected from the PG test rig for five different gear conditions (baseline, two levels of fault severity on the planet gear, and two levels of fault severity on the sun gear) under five different load conditions. In total 25 data sets will be collected for analysis (see Chapter Six).

- **Work Package 4: Implementation and Evaluation**

  This work package illustrates the practical phase of algorithm development. In this work package the architectural design of the deep CNN will be investigated, including number of layers, number of convolutional filters, etc., in order to select the optimal architecture and parameters for the CNN. The effectiveness of the developed CNN architecture will be evaluated and compared with three recent CNN architectures using simulated data. Moreover, the performance of the developed CNN architecture with the proposed IReLU-Tanh function will be evaluated and compared against the existing activation functions including Tanh, ReLU, LReLU, and ELU. Finally, this work package presents a conclusion obtained from the simulated and experimental results (see Chapter Eight and Chapter Nine).

- **Work Package 5: Writing up**

  This work package is focused on writing up the thesis, which is based on the results obtained from all previous packages.

1.10. Thesis Outline

This thesis is organised into ten chapters as follows:

Chapter Two outlines the relevant literature reviews, it starts with reviewing data enhancement methods including TSA, EMD, MED and MOMEDA. This chapter then sheds light on the application of AI methods including shallow and deep learning for CM.

Finally, the key findings obtained from the literature review are discussed at the end of the chapter.

Chapter Three presents the PG case study used in this research including PG components and characteristic frequencies. Then, fault frequencies including sun gear, planet gear and ring gear are illustrated. Finally, gear failure modes and their potential causes are described.

Chapter Four presents the theoretical background of deep CNN and describes the fundamental concepts associated with its architecture, including convolution, activation function, pooling, and fully connected layers. Followed by, the learning algorithm, illustrating its theoretical bases. Finally, the implementation procedure of the deep CNN is discussed in detail.

Chapter Five presents the theoretical background of the activation function. It starts with reviewing the most widely used activation functions including Tanh, ReLU, IReLU and ELU used for deep CNN. Followed by, highlighting the main shortcomings of these activation functions when applied to deep CNN. Then, the chapter presents the proposed IReLU-Tanh function, which will be developed to address the limitations of existing activation functions.

Chapter Six presents the experimental facilities used in this study, including test rig development, instruments used to carry out the vibration measurement, fault simulation, and finally the experimental procedure.

Chapter Seven presents the results obtained from applying the conventional signal processing and MOMEDA methods to the experimental vibration data. It starts with analysis of the time and frequency domains of the vibration signals. Finally, it presents the results obtained from applying the MOMEDA method to the vibration data.

Chapter Eight presents the implementation of the CNN architecture with the proposed IReLU-Tanh function. The details of the creation of the CNN architecture including design, training and parameters tuning, validation and testing are presented. The effectiveness of the developed CNN architecture will be evaluated and compared with three recent CNN architectures using simulated data. Moreover, the performance of the

developed CNN architecture with the proposed IReLU-Tanh function will be evaluated and compared against the existing activation functions including Tanh, ReLU, LReLU, and ELU.

Chapter Nine presents the evaluation performance of the developed CNN architecture with the proposed IReLU-Tanh function when applied to vibration data. The results obtained are presented and compared against the existing activation functions.

Chapter Ten presents the conclusions with a summary of the thesis achievements, contribution to the knowledge, and recommendations for future work.

Chapter Two: Literature Review

This chapter presents a relevant literature review, starting with reviewing data enhancement methods including time synchronous averaging, empirical mode decomposition, minimum entropy deconvolution, and multipoint optimal minimum entropy deconvolution adjusted. Then, the application of AI methods including shallow learning and deep learning for CM are reviewed. Finally, the key findings obtained from the reviews are presented.

2.1. Introduction

A number of data processing techniques have been developed in the field of CM and applied to analyse the monitored data. These techniques are classified into three main types, as discussed in Chapter One, namely: signal processing, data enhancement, and AI [22-24]. Signal processing techniques are used to process and analyse the monitored data. Data enhancement techniques are employed to enhance the important features in the monitored data, and have been widely implemented for CM. In this chapter, the data enhancement techniques used for CM are presented in section 2.2. AI techniques are used to automate the diagnostic procedures and minimize human involvement in CM. AI techniques can be classified into two main groups, shallow learning and deep learning [49] and these are reviewed with their theoretical bases in section 2.3.

2.2. Data Enhancement Methods for Condition Monitoring

A number of studies have applied data enhancement to CM and the most common methods are discussed in the next section.

2.2.1. Time Synchronous Averaging

TSA or synchronous averaging is a well-known time domain technique developed by McFadden in 1991 [66]. It is based on the idea of removing random background and non-synchronous features from the periodic signal by averaging the obtained signal over many rotations. TSA can be computed as follows [67]:

$$g(i) = \frac{1}{N}\sum_{n=0}^{N-1} x(i + nT) \tag{2.1}$$

where: $g(i)$ is the averaged signal, $x(i)$ is the measured signal, T is the averaging period, and N is the number of averaged segments.

TSA has been widely applied to enhance the vibration signal, for instance, Fan, Zhou [68] implemented TSA to enhance the periodic vibration signal related to a gear of interest, then the enhanced signal was fed to train a SVM to classify the gear's condition. Zuber, Bajrić [69] combined TSA with an ANN for analysis of vibration signals. In this study, the TSA was used for suppressing background noise and non-synchronous features from

the periodic vibration signal, then the enhanced signal was fed to train the ANN to classify different types of faults.

2.2.2. Empirical Mode Decomposition

EMD was developed by Huang [70] for analysing non-linear and non-stationary data [71]. The basic idea of the EMD method is that the input data can be decomposed into a set of Intrinsic Mode Functions (IMFs) based on local characteristic time scales [72]. According to the EMD method, the input data $x(t)$ can be decomposed into a set of IMFs as follows [73]:

$$x(i) = \sum_{n=1}^{N} c_n + r_N \tag{2.2}$$

Where c_n represent the intrinsic mode function (IMF) and r_N is the residue of data $x(i)$.

2.2.3. Minimum Entropy Deconvolution

The MED technique was developed by Wiggins in 1978 [74] and has proved its efficiency in enhancing fault impulse features [75]. The basic idea of the MED is to find an inverse filter that counteracts the effect of the transmission path by assuming the original input signal was impulsive and hence having a high kurtosis value [75]. Kurtosis is an indicator that reflects the "peakiness" of a signal, and therefore its impulsive features [76]. The maximum kurtosis value is used as the termination condition for the iteration. The MED technique was successfully applied by Endo and Randall [77] for removing the effect of the transmission path and to enhance the clarity of the fault impulse features. Li, Ji [78] combined MED and variational mode decomposition for gear fault diagnosis and claimed that the developed method enhanced fault impulse features. MED has also been widely applied to enhance the fault impulse features for gear fault detection and diagnosis [79-81]. Despite its successes in enhancing fault impulse features, it has been reported that the MED is only able to enhance a single fault impulse feature as opposed to periodic fault impulses. In addition, MED is an algorithm that finds a good filter solution iteratively, as a result of adopting the maximum kurtosis value as the objective function for the iteration termination process [82, 83].

2.2.4. Multipoint Optimal Minimum Entropy Deconvolution Adjusted

MOMEDA is an improved method of the MED proposed by McDonald and Zhao [82], to overcome the shortcomings of MED method [84]. MOMEDA has the capability to extract periodic impulse features instead of single fault impulse feature. Moreover, it is a non-iterative solution for the filter selection, which means that the MOMEDA method has the capability to reach the optimal filter solution without the iterative process [85].

In general, the vibration signal generated from the machinery system is composed of several parts and it can be expressed as in equation (2.3) [86]:

$$x(i) = h(i) * v(i) + e(i) \tag{2.3}$$

where $x(i)$ is the measured signal, $*$ represents the convolution operation, $h(i)$ is the transfer function, $v(i)$ is the periodic impulse feature, and $e(i)$ is the noise added into the signal.

The process of recovering the properties of the input signal $v(i)$ is referred to as deconvolution. Deconvolution can be carried out by applying an inverse filter $f(i)$ to the measured signal $x(i)$ that aims to restore the properties of the input signal $v(i)$. The deconvolution can be expressed as [84, 87]:

$$v(i) = f(i) * x(i) = \sum_{l=1}^{L} f_l x_{i-l} \tag{2.4}$$

Where L is the length of the deconvolution filter of $f(i)$, $x(i)$ is the measured signal, and $v(i)$ is the recovered signal.

To obtain the periodic impulse features, McDonald and Zhao introduced a target vector $\vec{t}$ that represents the location of the impulses to be deconvolved, and which can be determined by the period of the fault [82], for example:

$$\vec{t} = [0000100001000]^T \tag{2.5}$$

The target vector $\vec{t}$ aims to deconvolve two impulses in the signal. The first impulse is located at $n = 5$ and the second impulse at location $n = 10$. The normalised level is between 0 and 1, where a value of 1 indicates that the optimal target solution was reached, and the fault impulse feature has been extracted. Therefore, the target vector $\vec{t}$ can be used

to separate and determine the location of the fault impulse feature to be deconvolved [82]. In order to measure the extracted fault feature, multipoint kurtosis (MKurt) was introduced as a measure of the fault at multiple impulses [84]:

$$MKurt = \frac{(\sum_{n=1}^{N-L} t_n^2)^2 \sum_{n=1}^{N-L}(t_n v_n)^4}{\sum_{n=1}^{N-L} t_n^8 (\sum_{n=1}^{N-L} v_n^2)^2} \quad (2.6)$$

where N is the length of the input signal and L is the length of the filter.

The definition is based on kurtosis and has been extended to multiple impulses at the controlled locations as determined by the target vector. Wang, Wang [85] applied MOMEDA to a vibration signal in order to de-convolve the effect of the transmission path and enhance the fault impulse features in the signal. Cai and Wang [88] recently used a combined ensemble local mean decomposition (ELMD) and MOMEDA to extract a series of fault impulses features from a vibration signal. ELMD was used to decompose the measured signal into several product functions, MOMEDA was then used to extract the periodic fault impulse features from the decomposed signal. Cai, Yang [89] applied MOMEDA to extract a series of multiple fault features generated by a gearbox defect.

2.3. Artificial Intelligence Methods for Condition Monitoring

2.3.1. Shallow Learning Methods

Shallow learning is a branch of AI that is concerned with processing and analysing the manually extracted features. Shallow learning can be used for solving different tasks including regression, classification, and anomaly detection [54, 90]. Learning methods, including unsupervised and supervised learning, can be implemented based on shallow learning methods. Supervised based shallow learning can be employed to build a classifier that learns how to produce the desired output based on the given examples of input-output (labelled data). Unsupervised based shallow learning draws inferences from the data itself without using any corresponding label (unlabelled data) [91]. Methods such as ANN, SVM, KNN and DT, belong to shallow learning, have attracted significant attention in a variety of applications, and since then it has been widely applied for intelligent fault diagnosis [53]. The construction of shallow learning methods generally comprises three main steps; data acquisition, feature extraction or selection, and classification [92], as shown in Figure 2-1.

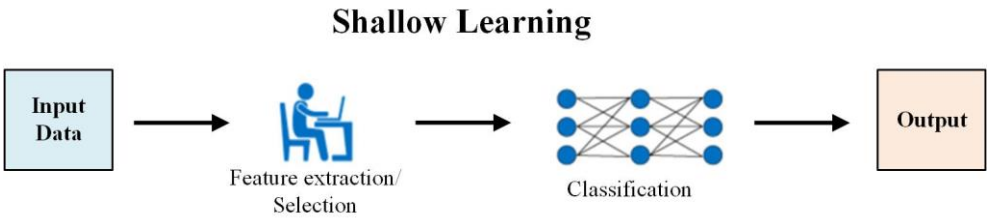

Figure 2-1: Application of shallow learning

Feature extraction is the process of extracting the representative features from the data [93]. Feature selection aims to reduce the dimensionality of the data by selecting only discriminative features that are expected to be the most relevant to the problem [94]. For intelligent fault diagnosis, the feature extraction and selection have attracted most attention by many researchers in recent years [95]. In the classification step, the extracted features are used to train the shallow learning methods. A number of studies have applied shallow learning methods for classification, these have included ANN, SVM, KNN, and DT [96]. However, the features are still extracted using conventional signal processing techniques and the classification performance of shallow learning methods relies entirely on the quality of the extracted and selected features [93].

2.3.1.1. Artificial Neural Network

ANN is an information processing paradigm inspired by the nervous systems of the human brain [97]. It has been widely applied in several applications including image processing and speech recognition, etc. ANNs can be trained based on supervised or unsupervised learning. As shown in Figure 2-2, ANN consists of three different kinds of layers: input layer, hidden layer, and output layer. Each ANN layer is composed of a number of interconnected processing units called neurons, where each neuron has connections to neurons in the preceding and succeeding layers via weighted connections. These interconnected processing neurons work together to solve a specific task [98]. During training of ANNs, the network updates the interconnection weights of the model until the error value between the desired target output and the network output predictions is minimised as close as possible to the minimum error value [99].

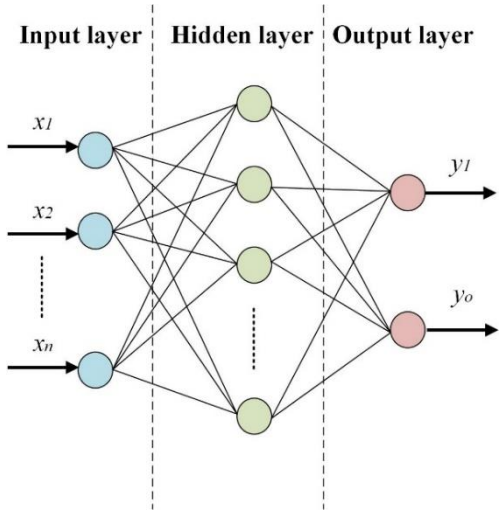

Figure 2-2: Typical ANN structure

To better understand the ANN structure, Figure 2-3 shows an example of a simple ANN, which consists of a single neuron.

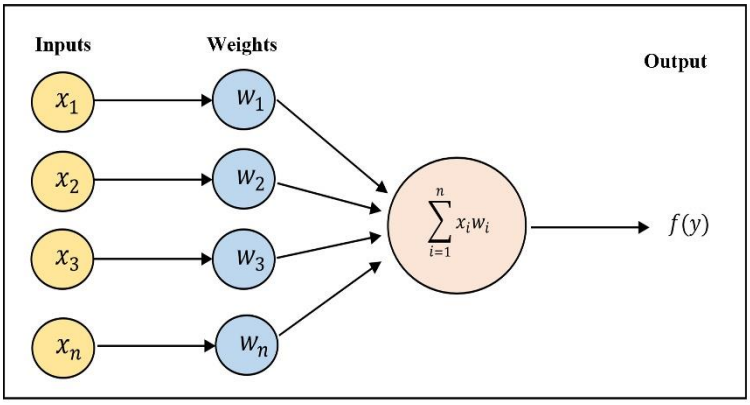

Figure 2-3: Simple neuron structure

The neuron in an ANN is a computational unit that multiplies the input data x_1, x_2, x_3, x_n with individual weights $w_1, w_2, w_3, w_\mathrm{n}$, see equation (2.7) [100]:

$$y = \sum_{i=1}^{n} x_i w_i \tag{2.7}$$

where y is the neuron output, x is the input data and w is the weights.

The sum of the products of multiplication between the input data and weights is fed into an activation function to introduce a non-linear characteristic to the ANN model. A common activation function is Tanh, and it can be computed as in equation (2.8) [101]:

$$f_{Tanh}(y) = \frac{(exp^{y} - exp^{-y})}{(exp^{y} + exp^{-y})} \quad (2.8)$$

where f is the activation function.

The Tanh function output ranges between -1 and 1. However, a number of activation functions have been developed in recent years, such as ReLU, LReLU, and ELU. These activation functions will be discussed in more detail in Chapter Five.

2.3.1.2. Support Vector Machine

SVM is a supervised learning method proposed by Vapnik [102] for solving a variety of tasks such as classification and regression. It was originally designed for solving binary classification problems [103]. The basic idea of the SVM is to separate the data into two-classes by finding an optimal hyper-plane between the classes so that class A and class B are separated by a clear gap that is as wide as possible, i.e. the margin is maximized as shown in Figure 2-4 [104].

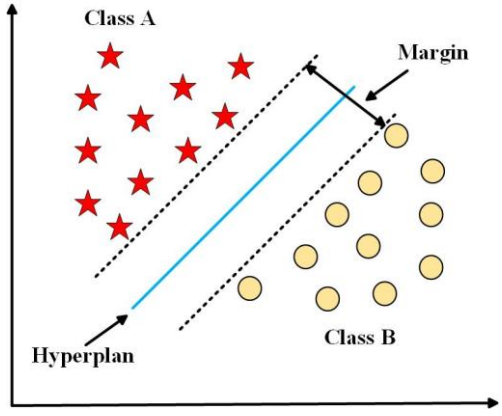

Figure 2-4: SVM for binary class classification

SVM can be applied to solve multi-class classification problems by using a combination of several binary SVM classifiers, such as One-Versus-One (OVO) and One-Versus-All (OVA). For a multi-class classification of N classes, the OVO method constructs $\frac{N(N-1)}{2}$ classifiers, where each classifier is trained on data having two different classes [105]. In the case of OVA, it constructs N multi-class SVM models where each model is trained using all the training data in which the data belonging to class A has a positive label and the rest (i.e., class B, C) have a negative label [106], see Figure 2-5.

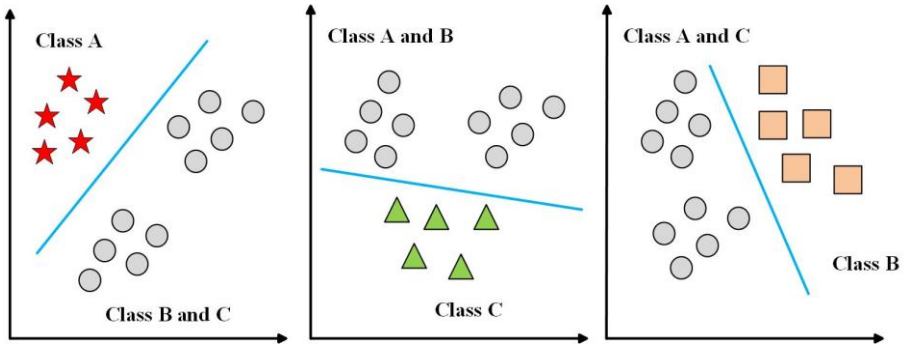

Figure 2-5: Examples of using One-Versus-All for multi-class classification SVM

SVM has been successfully applied in many real world problems such as image classification, handwritten character recognition, and speech recognition. For rotating machinery, it has been widely used to classify the machine condition, for instance Zhang, Peng [107] employed a multi-class classification SVM to classify three types of gear condition. Chao, Lu [108] applied the SVM method to spectrum data to classify different types of gearbox condition. Kang, Zhang [109] presented intelligent fault diagnosis for a gearbox based on wavelet packet analysis and SVM. In this study, the standard deviations of the wavelet packet coefficients were selected to identify the faults as the feature vector. Then, the selected features were used to train the SVM method to diagnose several types of faults.

2.3.1.3. K-Nearest Neighbour

KNN is a supervised learning method developed by Cover and Hart [110] and is commonly used for solving classification and regression tasks. The objective of KNN is to assign data samples into one of the predefined classes based on k nearest (closest) neighbour in the training data. A sample is assigned by the majority vote of its neighbours, with the sample being assigned to the most common class amongst its k nearest neighbours [111]. Figure 2-6 shows an example of KNN used to classify a sample into one of two predefined classes, A and B.

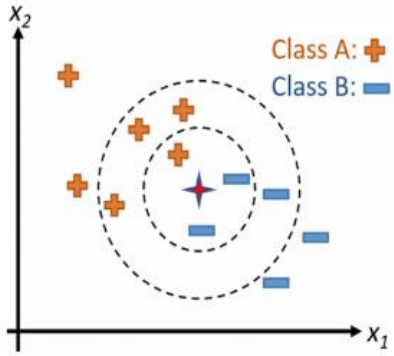

Figure 2-6: A simple example of KNN model for different values of k

For example, in Figure 2-6, assume that there are two classes (A and B), and a data sample (star point) needs to be classified. i.e., If k is set to be 3, (samples in the smaller circle), then the test data sample (star point) will be classified as class B; this is because there are two samples belonging to class B and only one sample belonging to class A. Whereas, if k is set to be 7 (samples in the larger circle), then the test data sample will be classified as class A because the majority (four) of its seven neighbours belong to class A and only three data samples belong to class B.

2.3.1.4. Decision Tree

DT is a well-known classification method developed by Quinlan [112]. The training of the DT is based on a supervised learning paradigm in which a set of decision rules are created to classify unseen data. A DT is structured in a way that progressively breaks down the data set into smaller sub-divisions, starting from the root node to the final leaf node. Figure 2-7 shows an example of a DT, samples at the root node will be divided into left and right nodes according to the inferred features from the data set [113]. Each branch represents one of the possible outcomes that lead to the target class, and each leaf node represents the class label. This process is applied iteratively for each node of the tree until the leaf node is reached, where the data cannot be divided to any further nodes. The final leaf node of the tree returns the final classification for all the samples that end up in that node [114]

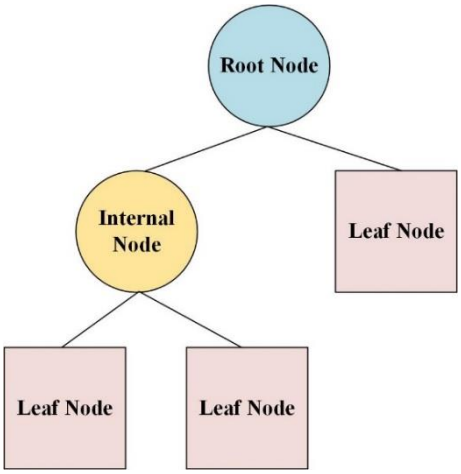

Figure 2-7: Example of DT

2.3.1.5. Application of Shallow Learning Methods to CM

Several studies have taken advantage of shallow learning methods to analyse extracted features and classify a system's health condition. For instance, Guolian, Pan [115] in 2010 implemented an intelligent fault diagnosis based ANN for identifying different types of fault in a wind turbine control system. The results confirmed the ANN as an effective and accurate fault diagnosis method. Yang, Hoi [116] in 2011 proposed intelligent fault diagnosis based on the combination of two shallow learning methods ANN and genetic algorithm (GA). In this study, GA was used for feature selection and then the selected features were fed to the ANN classifier to detect nine types of faults in a gearbox. Khazaee, Ahmadi [117] in 2012 applied the Least-Squares SVM (LS-SVM) to detect and classify three PG conditions: healthy, and two different gears, each with a worn gear tooth. Statistical features were extracted from the frequency domain signals. The extracted features were fed to the SVM classifier for fault classification. The results demonstrated that the LS-SVM classifier was able to detect and classify the PG conditions. Praveenkumar, Saimurugan [118] in 2014 implemented a SVM to diagnose the condition of a gear using statistical features (mean, median, RMS, and kurtosis) extracted from the time domain signal. The extracted features were fed to a SVM classifier which successfully identified several faults, and it was claimed that the SVM could be used for intelligent fault diagnosis. Yang, Liu [119] in 2015 used ensemble EMD for feature extraction and the extracted features were fed into a SVM classifier to detect different types of gear defects. Heidari, Homaei [120] in 2016 compared two intelligent SVM models: wavelet-SVM and LS-SVM to detect several types of faults in a gearbox.

Features were extracted using a wavelet packet transform and then the extracted features were fed into the two SVM models. The results of the classification accuracy showed that the wavelet-SVM had better diagnostic performance and was more accurate than the LS-SVM. Montaña and Sanz-Bobi [121] in 2018 extracted statistical parameters from the time and frequency domain signals obtained from a wind turbine gearbox, and then used princple component analysis (PCA) to reduce the dimensionality of the data and select representative features. The selected features were fed to a self-organizing map and Gaussian mixture model to detect anomalies in the gearbox. Stetco, Dinmohammadi [53] in 2019 reviewed the recent literature on shallow learning methods that have been used for machine CM.

2.3.1.6. Limitations of Existing Shallow Learning Methods

Based on the recent studies outlined in Section 2.3.1.5, it can be said that the vast majority of the AI based shallow learning methods for CM are applied for the classification task, while the features are still manually extracted [122]. Such an approach has the following deficiencies:

- Feature extraction and classification tasks are two independent processes and as a result of that, the feature extraction task must be redesigned for every diagnostic task [123, 124].
- The classification accuracy of the shallow learning method relies entirely on the quality of the extracted features [125, 126].

These inherent shortcomings in shallow learning methods presently used for CM need to be overcome. Therefore, it is important to employ an automated technique based on deep learning that is able to extract representative features directly from the raw data and automatically classify the machine condition [127, 128].

2.3.2. Deep Learning Methods

With the continuous development of AI, a branch called deep learning was proposed by Hinton, Osindero [129] for the automatic learning of features and has become a popular method for a variety of applications [49, 130]. The key difference between shallow learning and deep learning methods is in how the features are extracted. As discussed in

section 2.3.1, shallow learning methods require hand-crafted features based on expertise knowledge. In the case of deep learning, the features are learned automatically from the raw input data by means of a multi-layered structure [131], as shown in Figure 2-8. The deep learning method integrates both feature extraction and classification tasks into a single hierarchical model and by doing so, it provides an effective solution to the current problems that occur in AI-based shallow learning methods for CM [132].

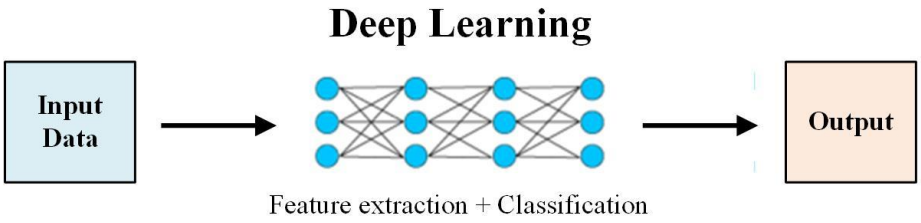

Figure 2-8: Application of deep learning method

Deep learning is commonly referred to as DNN, where multiple hidden layers are stacked in the network architecture to perform feature extraction and then automatically identify different classes for a given set of data. The term "*deep*" refers to the number of hidden layers present in the network architecture, which can range from tens to thousands of hidden layers [133]. A number of deep learning methods have been developed in recent years, such as CNN, SAE, and RNN [134, 135]. These methods will be discussed in more detail in section 2.3.2.3.

2.3.2.1. Fundamentals of Deep Learning

As with the ANN, deep learning also refers to a neural network model inspired by the nervous system of the human brain. The difference is, as the name suggests, deep learning is based on a deep architecture obtained by stacking multiple hidden layers between the input and output, hence giving the name Deep Neural Network (DNN), as shown in Figure 2-9. Each hidden layer is made up of a number of neurons to perform an operation on their inputs, with each neuron connected to a neuron in the next layer using weighted connections. The deeper the neural network, the more hidden layers and neurons the architecture will contain. With multiple levels of hidden layers, the DNN architecture can automatically learn more complex representations than a shallow architecture [136].

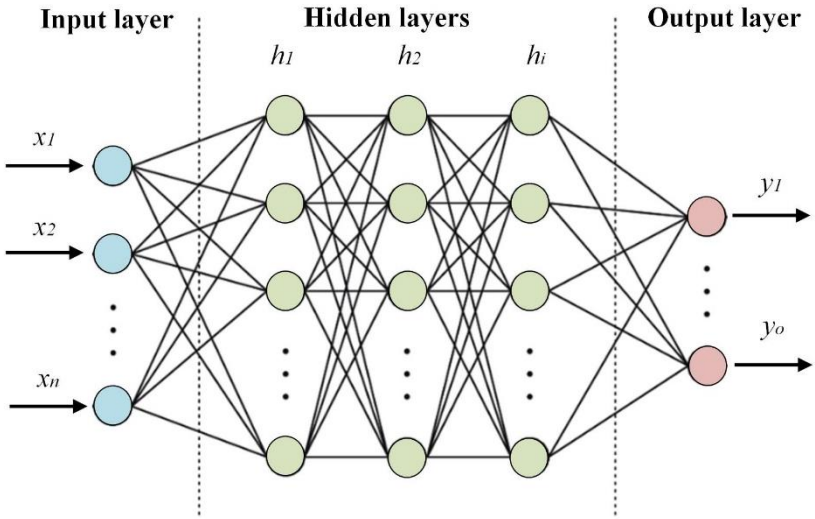

Figure 2-9: Deep neural network architecture

The implementation of DNN is generally consists of three main steps: training, validation, and testing. In the training step, the DNN is fed with training data to learn the representative features directly based on labelled or unlabelled data depending on the type of learning algorithm. The ***training step***, with DNN mainly involves forward and backward propagation. In forward propagation (from the input layer to the output layer), the training data is passed through the DNN architecture to produce an initial network output. Then, the network calculates the error value based on the current network parameters. In the backward propagation (from the output layer towards the input layer), the network iteratively minimises the error value by updating the network parameters to minimise the error value. The ***validation step*** is carried out by feeding the DNN with a set of unseen data to be used for producing the final model and to avoid overfitting problem through the early stopping technique. The ***testing step*** is the final step of applying the DNN and is conducted by feeding the trained DNN model with unseen data called testing data, to measure the classification accuracy of the trained model [137, 138].

2.3.2.2. Training Deep Neural Network

DNNs can be used to solve a variety of tasks such as classification, anomaly detection and regression. Several studies have adopted deep learning methods for automated fault diagnosis in order to take advantage of its automatic learning of features, its high classification accuracy, and to avoid hand-crafted features [139-142]. Training DNN generally is carried out using one of the three main learning paradigms:

- **Supervised learning,** the DNN is trained using a set of labelled data, each input data has a target output, and during the training process the network learns how to produce the desired output based on examples of input-output (labelled data). Common supervised learning methods for deep learning include CNN, RNN, long short term memory (LSTM), and gated recurrent units (GRU) [52]. See also Section 1.6.1.1.
- **Semi-supervised learning,** is a mix between supervised and unsupervised learning, in which the DNN is trained with partially labelled data and then uses the trained DNN model to predict the remaining portion of unlabelled training data. Typical semi-supervised learning methods for deep learning are the generative adversarial network (GAN), LSTM and GRU [143]. See also Section 1.6.1.2.
- **Unsupervised learning,** the DNN is trained with a set of unlabelled data without using any corresponding target output. Common unsupervised learning methods for deep learning include auto-encoders (stacked, de-noising, and contractive), the restricted boltzmann machine [136]. See also Section 1.6.1.3.

2.3.2.3. Types of Deep Neural Network Algorithms

2.3.2.3.1. Auto-Encoder

Auto-encoder (AE), is a type of unsupervised neural network proposed by Rumelhart, Hinton [144], to learn feature representations from unlabelled data. An AE with deep architecture will be stacked with multiple levels of hidden layers between the input and output layers, as shown in Figure 2-10. The key difference between the deep AE and the ANN is the number of neurons in the output layer is equal to a number of neurons in the input layer [145].

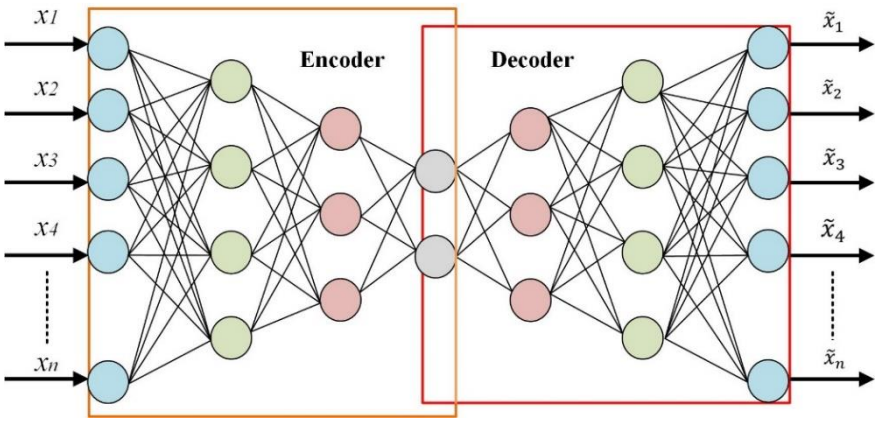

Figure 2-10: Deep auto-encoder

Training deep AEs generally involves two main stages, encoder and decoder. The encoder is used for mapping the input data into a hidden layer with a smaller number of neurons to force the network to learn representative features from the input data. The decoder refers to the process of reconstructing the input data from the hidden layer representation [146]. It is generally based on the idea that the AE will try to make the outputs as equal as possible to the inputs. In other words, AE approximates the identity function to obtain an output of AE $\tilde{x}_i$ that is similar to the input x_i, by minimizing the reconstruction error, possibly in the form of the mean square error (MSE). The encoder and decoder steps of AE can be computed as shown in equations (2.9) and (2.10) [147]:

$$h_i = f(w_i^e x_i + b_i^e) \tag{2.9}$$

$$\tilde{x}_i = f(w_i^d h_i + b_i^d) \tag{2.10}$$

where h_i is the hidden encoder obtained from input data x_i, f is the activation function, w_i^e is the encoder weight matrix, b_i^e is the encoder bias vector, $\tilde{x}_i$ is the reconstructed data, w_i^d is the decoder weight matrix, and b_i^d is the decoder bias vector.

During training of the AE, model parameters including $\theta = [w_i^e, w_i^d, b_i^e, b_i^d]$ are optimized to minimize the reconstruction error between input data x_i and the reconstructed data $\tilde{x}_i$. The reconstruction error of AE can be computed using equation (2.11) [148]:

$$E = \frac{1}{2}\sum_{i=1}^{N}(x_i - \tilde{x}_i)^2 \tag{2.11}$$

where E is the reconstruction error between the input data x_i and the reconstructed data $\tilde{x}_i$.

The deep AE is trained to minimize the MSE by optimizing the AE parameters to reconstruct the output $\tilde{x}_i$ from the input data x_i so that the reconstruction error $\frac{1}{2}\sum_{i=1}^{N}(x_i - \tilde{x}_i)^2$ is minimised [144]. Once the output is reconstructed from the hidden layer representation, it can be said that the deep AE has learned the important features contained in the input data and mapped those features into the hidden layers [145].

2.3.2.3.2. Recurrent Neural Network

RNN is a type of DNN architecture developed by Williams and Zipser [149] for processing sequential data such as text, video and time series data. The key idea of the RNN architecture is that it processes the input data in a recurrent manner. With a flow path going from the input layer to the hidden layer, where the hidden layer contains cyclic connections, and finally to the output layer [150].

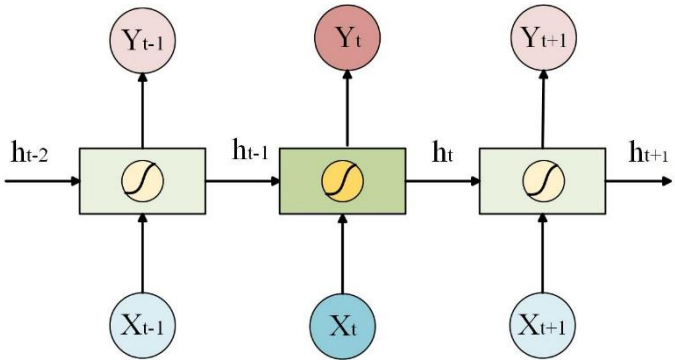

Figure 2-11: A simple recurrent neural network architecture

As shown in Figure 2-11, the RNN consists of multiple neurons, each neuron has a function for generating the current hidden state h_t, and the output y_t, using the current input x_t, and the previous hidden state h_{t-1}, and therefore h_t and $\mathrm{y_t}$ can be computed according to equations (2.12) and (2.13) [52, 151]:

$$h_t = f(w_h h_{t-1} + w_x x_t + b_h) \tag{2.12}$$

$$y_t = f(w_y h_t + b_y) \tag{2.13}$$

where w_h, w_x, and w_y are the weights for the recurrent connections of hidden-to-hidden, input-to-hidden, and hidden-to-output, respectively, b is the bias vector, and f denotes the activation function.

2.3.2.3.3. Convolutional Neural Network

CNN is a type of deep learning method proposed by LeCun, Bottou [152]. It is a supervised DNN and based on the idea that the training data needs to be labelled and then the network is allowed to model a relationship between the input data and the corresponding target output [153]. CNN integrates several features into a single deep

hierarchical model including; feature extraction, dimensionality reduction and classification [154]. The feature extraction task consists of several hidden layers, including convolution, activation function, and pooling layers. The convolution layer is intended to extract the representative features from the input data. The activation function is used to introduce non-linear characteristics to the model. The pooling layer is employed to reduce the dimensionality of the data whilst preserving the most important features. Stacking several convolution, activation and pooling layers allows the CNN to learn the representative features directly from the raw data. Finally, fully connected and softmax layers are applied at the end of the architecture for performing classification [155]. CNN has gained the special attention of many researchers and has been widely applied in several applications, including speech recognition, image classification, and object detection [154]. It has been reported that CNN can cope with different types of data, including three-dimensional (3D) data for videos, two-dimensional (2D) data for images, and one-dimensional (1D) data for signals [156-158]. It has recently been applied in the field of CM [63, 159, 160]. This research focuses on the development of CNN method for CM, which will be discussed in more detail in Chapter Four.

2.3.2.4. Application of Deep Learning Methods to CM

Several studies have attempted to use deep learning methods for automated fault diagnosis. For instance, Chen, Li [161] in 2015 implemented a deep CNN for automated diagnosis of a gearbox fault. Statistical parameters were extracted from the signal in the time and frequency domains and the extracted features were used to train a deep CNN using a supervised learning paradigm. Experimental vibration data was used to evaluate the developed deep CNN method, and the classification accuracy obtained was compared with that of a shallow learning model (SVM). It was claimed that the deep CNN outperformed the SVM model. Jing, Zhao [92] in 2017 developed a CNN to learn fault features directly from the frequency domain of the vibration data. A supervised learning paradigm was adopted to train the CNN method and classify the vibration data into six predetermined classes. Vibration data was collected experimentally from the gearbox and PG. The developed CNN method was compared with shallow learning models such as SVM, fully connected neural network and random forest. It was reported that the developed CNN method achieved higher classification accuracy and outperformed the shallow learning models.

Liu, Cheng [162] in 2018 proposed feature extraction and fault classification based on variational mode decomposition and CNN for automated fault diagnosis. Experimental vibration data was collected for different PG conditions, and training was by a supervised learning paradigm. It was claimed that the proposed method could successfully extract the fault features and accurately identify different PG fault conditions with high classification accuracy. Liu, Bao [147] also in 2018 presented automated fault diagnosis based on a deep SAE method that learned fault features directly from the frequency domain signals. The presented method was applied to vibration data collected from a gearbox with different health conditions. An unsupervised learning paradigm was used to train the deep SAE method. The study claimed that the deep SAE method was more effective and outperformed conventional intelligent diagnostic methods.

Park, Marco [163] in 2019 proposed an integrated deep learning approach based on auto-encoder and LSTM for automated fault diagnosis. In this study, the auto-encoder was used to detect the anomalies in the input data in an unsupervised learning manner, and then the predicted anomalies were fed into LSTM network to classify different types of faults using a supervised learning paradigm. It was claimed that the proposed auto-encoder and LSTM method achieved an outstanding performance in detecting and classifying different types of fault. Zhang, Lin [164] and Zhao, Yan [165] both in 2019 reviewed the application of deep learning methods including SAE, CNN, RNN, and the restricted Boltzmann machine for automated fault diagnosis.

Yin, Yan [141] in 2020 developed LSTM for automated fault diagnosis. Experimental vibration data was collected from a wind turbine gearbox to evaluate the effectiveness of the developed LSTM method. A supervised learning paradigm was used to train the developed method and classify the input data into ten predetermined class conditions. It was reported that the developed method gave superior classification performance compared to so-called “classical methods” including SVM and KNN. Lin, Han [166] in 2020 developed a feature learning method based on CNN. The developed method was applied to learn features directly from the spectrum data of a vibration signal. A comparison of various input data types was carried out using both time domain and frequency domain data. A supervised learning paradigm was used to train the developed CNN method. The study confirmed that learning features from the spectrum data provide better fault classification performance than the time domain features.

2.3.2.5. Challenges of Existing Deep Learning Methods

Based on the studies listed in Section 2.3.2.4, deep architectures are observed to learn better feature representations from the data through a multi-layered structure. However, it has been reported that severe problems can appear within the deep architecture that is caused by the activation function. This affects the training of the network to extract representative features from the raw data, and hence the classification accuracy of the model [13, 14, 65]. Common problems include the vanishing gradient problem during the backward propagation through a multi-layered neural network, this is due to the value of the gradient decreasing exponentially to zero as it is propagated backwards [167]. This problem is exacerbated when an activation function such as the Tanh function is used. This is due to the fact that the Tanh function saturates at -1 and $+1$ for large negative and positive inputs, with the value of the gradient close to zero [168]. To overcome the vanishing gradient problem, a number of activation functions such as ReLU, LReLU, and ELU have been developed in recent years [169-171]. However, these activation functions still have some drawbacks such as the dying ReLU problem, non-zero fixed value, and adding a hyper-parameter to the network architecture. These shortcomings will be discussed in more detail in Chapter Five.

2.4. Key Findings

- A number of data enhancement methods have been developed for CM as discussed in Section 2.2. However, these methods rely on prior knowledge of the machinery system and well-trained technical to apply them successfully.
- Most of the AI based shallow learning methods for CM are applied for the classification task, while the features are still manually extracted using conventional signal processing techniques. Thus, the classification accuracy of shallow learning method relies entirely on the quality of the extracted features.
- Despite the wide implementation of automated fault diagnosis based deep learning methods, several studies have claimed that the activation function can also influence the overall performance of the deep architecture model and hence can lead to low classification accuracy of the model.
- As discussed in Section 2.3.2.5, the vanishing gradient caused by use of the Tanh activation function is the most common problem in deep architecture. This has a

significant effect on the network training process and hence on the overall performance of the model.

To address the obstacles mentioned above, an automated approach based on a deep CNN architecture with a proposed activation function is investigated for CM. The developed approach aims to automate the tasks of feature extraction and classification. In addition, an activation function will be developed to address the shortcomings in the existing activation functions, enhance the learning ability of deep CNN architecture during the training process, and consequently improve the overall performance of the model. The effectiveness of the developed deep CNN architecture with the proposed activation function will be evaluated and compared against existing activation functions, named Tanh, ReLU, LReLU, and ELU using simulated and experimental vibration data.

Chapter Three: Planetary Gearbox Failure Modes and Vibration Responses (Case Study)

This chapter presents a brief introduction to PG case study used in this research, beginning with PG components and their characteristic frequencies. Then, it introduces fault frequencies for the sun, planet and ring gears. Gear failure modes and their potential causes are presented, followed by a list of key findings.

3.1. Introduction

PGs are widely utilized in many industrial applications such as wind turbines and helicopters, owing to their high transmission efficiency [172]. The PG can offer different speed ratios, which means that different speeds can be achieved from the same planetary set, depending on which component remains stationary, which component is the input, and which component is the output. This makes PG an ideal option to be used in several industrial applications [173]. The reliability of the gears in a PG depends on its smooth operation inside the machine. However, most gears are subject to failure after long-term operation. A failure of the PG could cause a shutdown of key components or the entire system and may lead to significant economic losses and catastrophic accidents [174].

Generally, a PG is a compound gear system, see Figure 3-1, and consists of an externally toothed sun gear located at the centre, several externally toothed planet gears inserted between the sun gear and an internally toothed ring gear, and a carrier that holds the planet gears. The planet gears not only rotate around their own unfixed centres but also revolve around the centre of the sun gear [175], thus, the planet gear meshes simultaneously with both the ring gear and sun gear. With this structure, the PGs have several unique behavioural characteristics that pose great challenges for fault detection and diagnosis [176].

- According to the planet gear's structure there are multiple time-varying vibration transmission paths from the meshing points to the sensor [175, 177].
- The fault characteristics are easily affected by heavy noise in the low-frequency range, because at least one component in the PG rotates at low speed to achieve a large transmission ratio [25].
- Multiple planet gears meshing simultaneously (planet-ring and planet-sun) will generate similar vibrations with different meshing phases [177].

According to the unique behaviours described above, the vibration data acquired by a fixed accelerometer will be complicated. Therefore, CM of PG has attracted considerable attention aimed at solving the challenge of detection and diagnosis of PG [178].

3.2. Planetary Gearbox Components

A PG consists of four basic parts: sun gear, several planet gears, ring gear, and carrier, as seen in Figure 3-1.

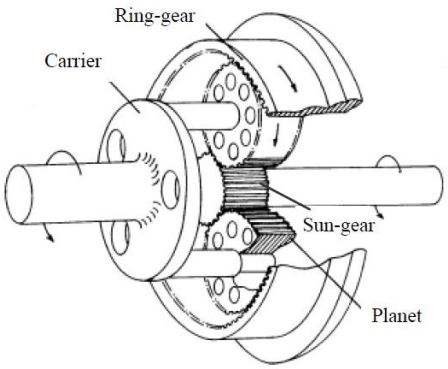

Figure 3-1: Planetary gearbox [173]

- **Sun Gear**

The sun gear rotates around its own centre and it is mounted can be either the input or output rotating shaft, depending on the planetary gear configuration.

- **Planet Gear**

Several planet gears are inserted between the sun and ring gears. These planet gears are held by the carrier, and they not only rotate around their own unfixed centres but also revolve around the centre of the sun gear.

- **Ring Gear**

The ring gear is mounted in the gearbox housing and it is usually the fixed component.

- **Carrier**

The carrier holds the planet gears and is mounted on the input or output rotating shaft, depending on the planetary gear configuration.

3.3. Characteristic Frequencies of the PG

The characteristic frequencies of the PG can be defined in terms of meshing frequency, rotational frequencies of the carrier, sun, and planet gears. For the PG used in this study, $Z_{ri} = 62$ is the number of teeth on the ring gear, $Z_{pi} = 26$ is the number of teeth on each planet gear, $Z_{si} = 10$ is the number of teeth on the sun gear, and $i = 7.2$ is the gear

transmission ratio. The characteristic frequencies calculated based on this PG structure are given below [25]:

The PG transmission ratio can be expressed as:

$$i = 1 + \frac{Z_{ri}}{Z_{si}} \tag{3.1}$$

The rotational frequencies of the carrier (f_{rci}), planet gears (f_{rpi}) and sun gear (f_{rsi}) can be computed using the following equations, respectively:

$$f_{rci} = \frac{Z_{si}}{(Z_{ri}+Z_{si})} * f_{rsi} \tag{3.2}$$

$$f_{rpi} = \frac{(Z_{ri}-Z_{pi})*Z_{si}}{(Z_{ri}+Z_{si})*Z_{pi}} * f_{rsi} \tag{3.3}$$

$$f_{rsi} = \left(\frac{Z_{ri}+Z_{si}}{Z_{si}}\right) * f_{rci} \tag{3.4}$$

The meshing frequency (f_{pmi}) can be calculated as:

$$f_{pmi} = \frac{(Z_{ri}*Z_{si})}{(Z_{si}+Z_{ri})} * f_{rsi} \tag{3.5}$$

3.4. Characteristic Frequencies of Faulty Gears in a PG

Defects in PGs can be classified into three types: sun gear fault, planet gear fault, and ring gear fault. The characteristic frequencies of each of these faulty gears are calculated in this section [179]:

- **Characteristic Frequency of Faulty Sun Gear**

The characteristic frequency of the sun gear with fault (f_{sf}) can be expressed as:

$$f_{sf} = K\frac{f_{pmi}}{Z_{si}} \tag{3.6}$$

Where K denotes the number of planet gears.

- **Characteristic Frequency of Faulty Ring Gear**

The characteristic frequency of the ring gear with fault (f_{rf}) can be computed as:

$$f_{rf} = K\frac{f_{pmi}}{Z_{ri}} \tag{3.7}$$

- **Characteristic Frequency of Faulty Planet Gear**

The characteristic frequency of the planet gear with fault (f_{pf}) can be calculated as:

$$f_{pf} = 2\frac{f_{pmi}}{z_{pi}} = 2(f_{rpi} + f_{rci}) \quad (3.8)$$

3.5. Gear Failure Modes

Components such as gears in machinery systems are all subject to failure after long-term operation [180]. Gear defects can be classified into two groups: distributed defects and local defects. The main difference between distributed and local defects is that a distributed defect affects a number of gears, while a local defect affects a localised area of a gear and tends to increase rapidly once initiated. Thus, it is important to monitor the gear condition and detect any gear defect at an early stage as possible to avoid machine breakdown and catastrophic accidents [181].

- ***Pitting***: is a surface fatigue failure of the gear tooth. It occurs in a localised area on the gear tooth surface, and tends to extend rapidly to adjacent regions until the whole surface is covered [182].
- ***Scoring***: this type of failure occurs as a result of inappropriate lubrication between the gear teeth. It takes place in a localised area along the contacting surfaces of the gear teeth. The result can be a high temperature with weld spots appearing on the contacting surfaces of the two-mating gears. This can in turn, result in the gear teeth becoming distorted or even to break away as the gear rotates [183].
- ***Tooth Breakage***: this type of failure is considered as the most dangerous, and can cause serious damage to other rotating parts or the entire system. Tooth breakage generally occurs due to excessive load on the gear teeth, it starts with small cracks in a single tooth and then it increases in size until the entire tooth breaks off [184].
- ***Wear***: wear failure can be considered as removal of the surface material of the gear tooth due to repetitive contact on its surfaces, and it can take three forms: ***Adhesive wear*:** occurs when the force between two-mated gears teeth is sufficient to produce a weld spot over the area of contact. As a result, the weld spot breaks away as the gears rotate, and metal particles are released from the gear surfaces and contaminate the system. ***Abrasive wear*:** the occurrence of this type of wear is usually due to contamination of the gear lubricant by internal sources such as metal particles, dust,

and sand, and takes place on the meshing area between gear teeth [185]. ***Corrosive wear*****:** this type of wear is caused by chemical action, which is often caused by active ingredients in the lubricating oil, such as acid, water, or moisture [186].

3.6. Key Findings

To summarize, the key finding of this chapter are as follows:

- Multiple vibration transmission paths from the gear meshing points to the sensor are time-varying.
- The fault characteristics are easily affected by high noise levels in the low-frequency range, because at least one component in a PG rotates at low speed to achieve a large transmission ratio.
- Multiple planet gears meshing simultaneously (planet-ring and planet-sun) will generate similar vibrations with different meshing phases.
- Defects in PGs can be classified into three types: sun gear fault, planet gear fault, and ring gear fault. The characteristic frequencies of these faulty gears are given in Section 3.4.
- Tooth breakage failure is considered as the most dangerous, and can cause a serious risk to other rotating components or the entire system.

Thus, the vibration signal acquired by a fixed accelerometer mounted on the PG housing is complicated in nature. CM of a PG based on conventional signal processing techniques requires skilled technicians and will be costly and time-consuming. Thus, an automated approach based deep CNN could be a promising way to automate the diagnostic procedure and provide an accurate diagnosis of the system's health.

Chapter Four: Convolutional Neural Network

This chapter presents the background of deep CNNs and their architecture layers, including convolutional, batch normalization, activation function, and pooling. Followed by learning algorithms including loss function, backpropagation, gradient decent and learning rate, with their theoretical bases illustrated. Finally, the implementation of the CNN for CM is discussed in detail.

4.1. Introduction

One of the successful deep learning methods is the CNN. This method has achieved remarkable success in a variety of applications including image processing, speech recognition, natural language processing, and has been recently used in the field of CM [49, 130].

4.2. Convolutional Neural Network

CNN is a deep learning method proposed by LeCun, Bottou [152] for handwritten digit classification. It is a special type of neural network that employs a deep architecture with multiple hidden layers. CNN is a supervised DNN algorithm and based on the idea that the training data needs to be labelled, and then allow the CNN to model a relationship between the input training data and the corresponding target classes [143]. CNN integrates several features into a single deep hierarchical model, including: feature extraction, dimensionality reduction, and classification [128]. As shown in Figure 4-1, the feature extraction task comprises several of the hidden layers, including convolutional, batch normalization, activation function, and pooling, followed by a fully connected and softmax for performing classification. Integrating these hidden layers through a multi-layered configuration, allows the CNN to learn representative features directly from the raw data.

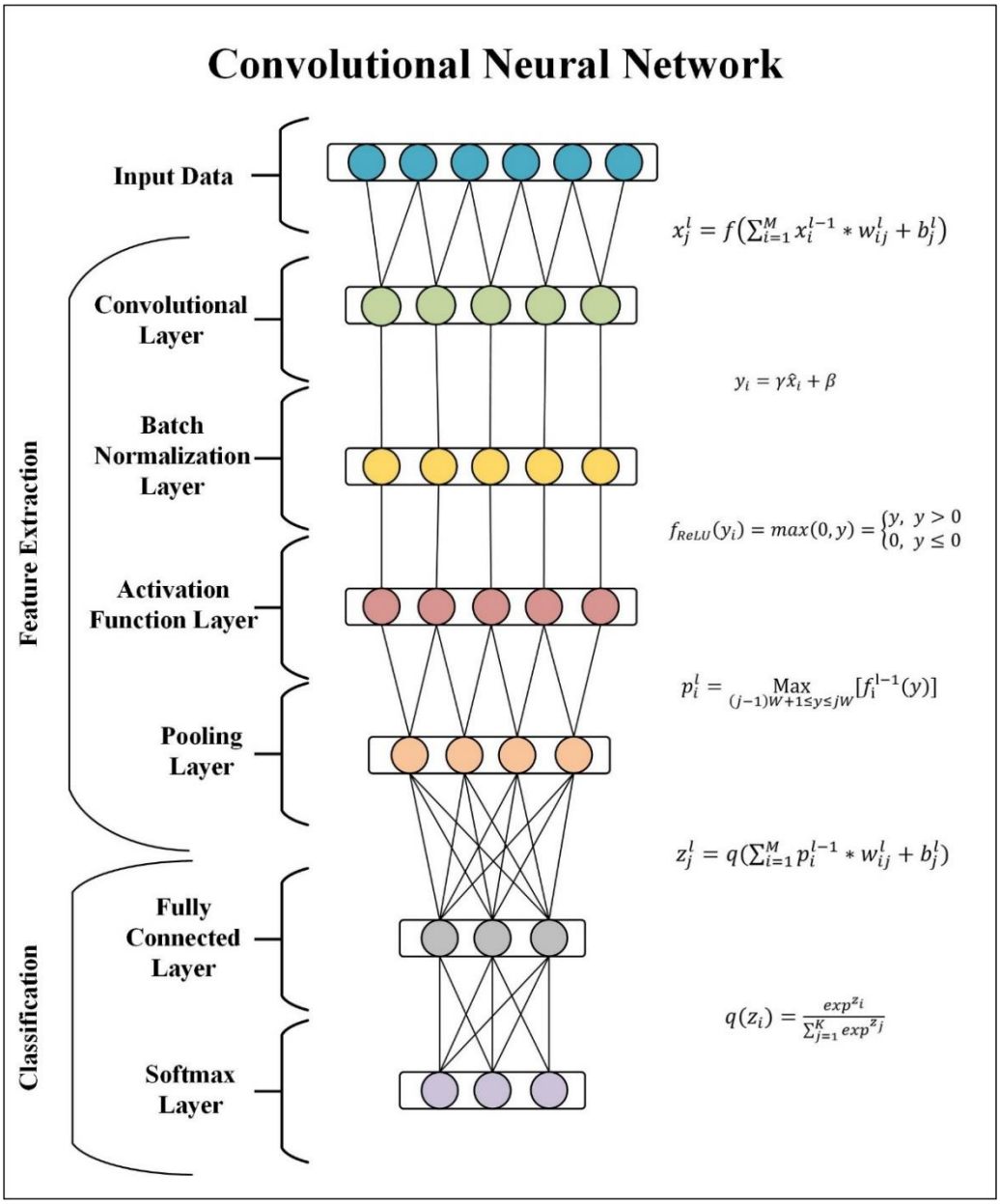

Figure 4-1: Schematic of a convolutional neural network

CNN is composed of three architectural ideas that play a significant role in its learning process: comprising local receptive field, shared weights, and pooling layer [156]. In the case of the traditional ANN, each neuron in the hidden layer is connected to all the neurons in the previous layer and each connection between neurons has its own weight, as discussed in Section 2.3.1.1, but in the case of CNN, each neuron in the hidden layer is only connected to a local region of neurons in the previous layer, referred to as the local receptive field [152, 187]. Another architectural idea of the CNN is that it shares the same weights in a particular layer. By sharing the same weights, CNN is forced to detect the same feature but at different locations of the data [157]. Figure 4-2 illustrates the idea of

the local receptive field and shared weights in a CNN. Typically, the final layer of feature extraction is the pooling layer. The basic idea of the pooling layer is that it reduces the dimension of the data while preserving the most important features [128], this leads to a reduction of connections or parameters to the next layers. Based on the above three architectural features, the CNN has several benefits including: (1) reducing the number of learnable weights to train the CNN, (2) reducing the network complexity and the dimensionality of the data, (3) reducing the potential risk of overfitting, (4) learning robust features from the data, and (5) achieving better learning and classification [143, 152, 157, 158].

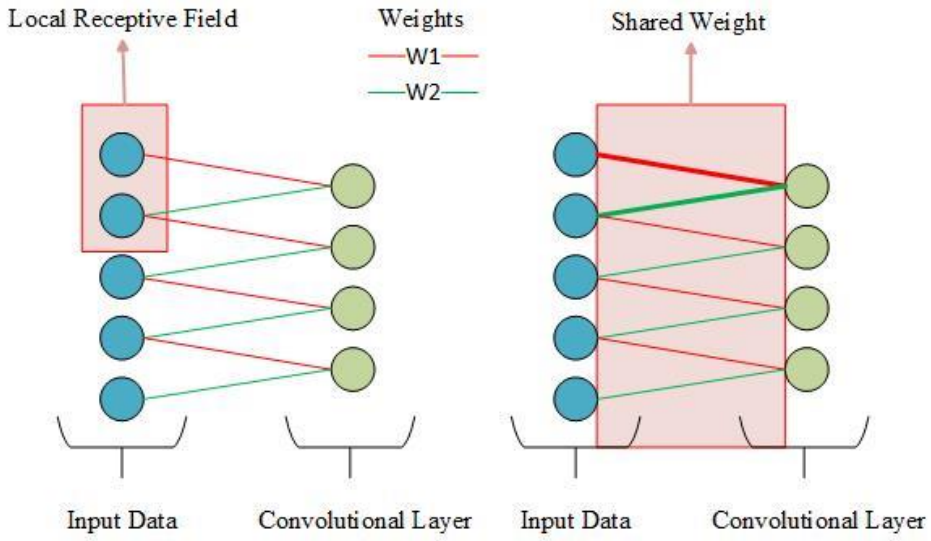

Figure 4-2: The idea of the local receptive field and shared weight in the CNN

Due to its unique architecture, CNN has gained the special attention of many researchers as a promising form of deep learning, and has been successfully applied in several applications. It has been reported that CNN can cope with different types of data, such as three-dimensional (3D) data for videos, two-dimensional (2D) data for images, and one-dimensional (1D) data for signals [156-158], and has been recently applied to CM [63, 159, 160].

4.3. CNN Architecture

As discussed in the previous section, the typical CNN architecture consists of several hidden layers, see Figure 4.1, with each layer having a different role and operation. The arrangement of these layers play a crucial role in designing the CNN architecture to

achieve better feature extraction and classification. In this section, the role of these layers is discussed in detail.

4.3.1. Convolutional Layer

The convolutional layer is the core building block of the CNN. It is employed as a feature extractor to extract representative features from the data. It uses the strategies of shared weights and local receptive field by means of the convolution operation instead of matrix multiplication as in the traditional ANN [152, 157]. In the convolutional layer, the input data is convolved with one or multiple convolutional filters or kernels and is applied systematically by moving (also known as striding) over the input data, creating an output feature map for each convolutional filter [12]. All convolutional filters have the same size as the local receptive field, which is connected to local region of neurons via a set of learnable weights. Each convolutional filter shares the same weights within the same convolutional layer [188]. The output of the convolutional layer consists of multiple output feature maps; each created using different convolutional filters with different weights [189]. The output feature map of the convolutional layer can be computed as [137]:

$$x_j^l = f\left(\sum_{i=1}^{N} x_i^{l-1} * w_{ij}^l + b_j^l\right) \quad (4.1)$$

Where x_j^l is the output value of j^{th} neuron in the feature map at the l^{th} layer, f denotes the activation function, x_i^{l-1} is the input data at $(l-1)$, w_{ij}^l is the convolutional filter connecting the i^{th} input feature map at $(l-1)$ with j^{th} feature map at l, and b_j^l denotes the bias.

The size of the output feature map generated by the convolutional layer is determined by several hyper-parameters, which play a key role in extracting the representative features from the data. These hyper-parameters are as follows; number of convolutional filters, size of convolutional filters, and stride [190]. The number of convolutional filters refers to the number of features to be learned within a single convolutional layer. The size of the convolutional filter refers to the size of the local receptive field that is convolved with the input data. The stride refers to the step size of the convolutional filter being slid over the input data [191].

4.3.2. Batch Normalization Layer

Training CNNs can be complicated because for each input layer the distribution of the input data varies during the training process as the network parameters are updated. The change in the input data distribution of each layer in the network is referred to as the "*internal covariate shift"* [137, 192]. It leads to slow down the convergence of CNN by requiring lower learning rates and appropriate initialization of network parameters. In order to address this issue, Ioffe and Szegedy [193] proposed a batch normalisation layer to reduce the shift of internal covariate. In doing so, it significantly accelerates the training process of CNNs. It draws its power from making the normalisation as a part of the network architecture itself [143]. Batch normalisation works by normalising the output feature map of the convolutional layer by first subtracting the mini-batch mean and dividing it by the mini-batch standard deviation. Then, the normalised values are shifted and scaled using two learnable parameters (beta β and gamma γ), whose values are learned during the training process [194]. The mathematical expressions for the batch normalisation are described in the following steps [193]:

A. Equations 4.2 and 4.3 are used to calculate the mean (μ_B) and the variance (σ_B^2) of the mini-batch training data:

$$\mu_B = \frac{1}{m}\sum_{j=1}^{m} x_j \tag{4.2}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{j=1}^{m}(x_j - \mu_B)^2 \tag{4.3}$$

B. Equation 4.4 is used to calculate the normalised value ($\hat{x}_i$) by subtracting the mini-batch mean (μ_B) from (x_j) and dividing by the mini-batch standard deviation (σ), a small number (ϵ) is added to avoid division by zero ($\epsilon = 1e-5$)):

$$\hat{x}_i = \frac{x_j - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{4.4}$$

C. Equation 4.5 is used to calculate the batch normalisation output (y_i) by multiplying the normalised value ($\hat{x}_i$) with a learnable scale (γ) and adding a learnable shift (β):

$$y_i = \gamma\hat{x}_i + \beta \tag{4.5}$$

Where B denotes a mini-batch of size m; x_j is the output feature map of the previous layer; γ, and β are learnable parameters that are updated during training.

4.3.3. Activation Function layer

The activation function is applied to the output of the convolutional layer, as shown in Figure 4-1 and according to Equation (4.1). It is used to determine whether a neuron should be activated or deactivated. The main role of the activation function is to introduce nonlinearity characteristics into the model [195, 196]. The activation function generally transforms the feature map of the convolutional layer to produce a non-linear output in the range of $[-1, +1]$, or $[0, \infty]$, depending on the choice of activation function [197]. Hence, the activation function allows the CNN to observe non-linear expressions of the data so that many complex problems can be resolved [198]. A number of non-linear activation functions have been developed in recent years, a common activation function used for CNN is the ReLU function, which can be computed as:

$$f_{ReLU}(y_i) = max(0, y) = \begin{cases} y, & y > 0 \\ 0, & y \leq 0 \end{cases} \tag{4.6}$$

The ReLU function output ranges between $[0, \infty]$, however, other types of activation functions such as Tanh, LReLU, and ELU [169-171] exist and will be discussed in more detail in Chapter Five.

4.3.4. Pooling Layer

Another important architectural feature in the CNN model is the pooling layer. It is employed to reduce the dimension of the feature map while preserving the most important features [128]. The pooling layer partitions each feature map into a set of non-overlapping pooling regions, and then extracts the maximum value (max-pooling) or computes the average value (average pooling) for each pooling region [188]. The max-pooling operation refers to extracting the maximum value for each non-overlapping pooling region in the feature map. Average pooling computes the average values for each non-overlapping pooling region [143]. Thus, it reduces the size of each feature map and the amount of computation needed in the network [137, 198]. Figure 4-3 illustrates the max-pooling and average pooling operations.

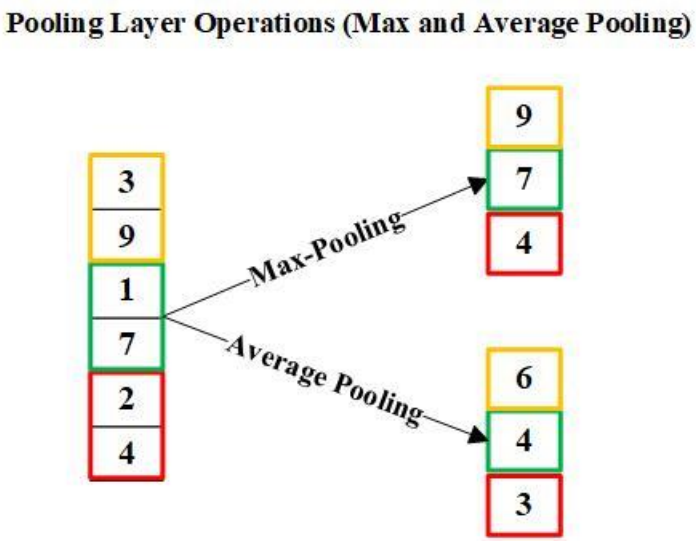

Figure 4-3: Max pooling and average pooling operation

The max-pooling and average pooling operations can be calculated as follows [198, 199]:

$$p_i^l = \underset{(j-1)W+1 \le y \le jW}{\text{Max}} [f_{\text{i}}^{l-1}(y)] \tag{4.7}$$

$$p_i^l = \underset{(j-1)W+1 \le y \le jW}{Avg} [f_i^{l-1}(y)] \tag{4.8}$$

Where $f_{\text{i}}^{l-1}(y)$ denotes the value of y^{th} neuron in the i^{th} feature map at layer $l-1$, W is the width of the pooling region, j denotes the j^{th} moving step of the pooling operation, and p_i^l denotes the output value of i^{th} neuron in the feature map at l^{th} layer of the pooling operation.

Previous studies [200-202] have shown that max-pooling significantly outperforms the average pooling operation for image classification, and it also leads to a faster convergence rate and reducing the amount of computation, resulting in better efficiency during the training process. The max-pooling operation is the most widely used in modern CNN architecture and has been successfully implemented in CNN architecture for CM [198, 203, 204].

4.3.5. Fully Connected Layer

A fully connected layer is commonly used in the classification task for CNN architecture, see Figure 4-1. The main role of this layer is to combine all of the features learned by the previous layers (convolutional and pooling layers) to classify the representative features

[205]. During training of the CNN, the output feature map of the previous layer (e.g., pooling layer) is typically flattened, e.g., transformed into a one-dimensional (1D) vector and as the name 'fully connected' suggests, all neurons in the fully connected layer are connected to all the neurons in the previous layer by a learnable weight [206]. For the classification task, the fully connected layer has the same number of K target classes. The output of the fully connected layer is then forwarded to the softmax layer in order to calculate the probabilities for the predictions of each target class in a supervised learning manner [207]. The fully connected layer can be computed as:

$$z_j^l = q(\sum_{i=1}^{M} p_i^{l-1} * w_{ij}^l + b_j^l) \tag{4.9}$$

Where z_j^l is the output of j^{th} neuron in the fully connected at layer l, q is the softmax function, p_i^{l-1} is the input feature map at $(l-1)$ generated from the previous layer, w_{ij}^l is the weight connecting the i^{th} feature map at $(l-1)$ with j^{th} feature map at l, and b_j^l donates the bias.

4.3.6. Softmax Layer

A softmax layer is employed at the end of the CNN architecture for a multi-class classification task, see Figure 4-1. It normalises the output of the fully connected layer of K values into a probability distribution, creating an output consisting of K probabilities [208]. The output probability of each value ranges between [0 and 1], and the total output probabilities sum to 1. The softmax function returns the output probabilities of each class and the highest output probability among K probabilities will be taken as the predicted class [209]. The softmax function can be computed as [204]:

$$q(z_i) = \frac{exp^{z_i}}{\sum_{j=1}^{K} exp^{z_j}} \tag{4.10}$$

Where K is the number of classes and $q(z_i)$ is the estimated probability value of an observation data z that belongs to i^{th} class.

4.4. Learning Algorithm

Learning algorithms can be achieved in different ways as explained in Chapter Two, these include supervised learning, unsupervised learning, or semi-supervised learning. In this

thesis, supervised learning is used to train the CNN with a labelled-data set. The main goal of the learning algorithm is to find the optimal model parameters by iteratively minimising the error value between the desired target output and the network output prediction. To find such optimal parameters, the gradient descent (GD) optimisation algorithm was used during the training of the CNN. It is an iterative process, where the model parameters are repeatedly being optimised during the training process to produce better results.

4.4.1. Loss Function

Cross entropy is employed as a loss function for multi-class classification. It is used to measure the error between the desired target output and the network output prediction [210, 211]. During training, the GD algorithm updates the model parameters to obtain the best relationship between the training data and its target class. This is done by iteratively updating the parameters until the lowest level or minimum error value is reached, where the cross-entropy loss function cannot be further minimized. At this stage, it can be said that the training process has achieved it's optimum value, where the model is best matched to the training data and its target class [212]. The mathematical expression of the cross-entropy loss function can be computed as [157]:

$$Loss = -\sum_{i=1}^{K} t(z_i) log(q(z_i)) \tag{4.11}$$

Where K is the number of classes, $t(z_i)$ is the target class, and $q(z_i)$ is the estimated probability value of an observation data z that belongs to i^{th} class.

4.4.2. Backpropagation

Backpropagation (backward propagation) is a well-known algorithm proposed by Rumelhart, Hinton [144] for training neural networks. It is used in conjunction with the GD algorithm to calculate the gradient of the loss function, e.g., the rate at which the loss function is minimised with respect to the model parameters [213]. Training a CNN generally involves two main steps, forward and backward propagations.

Forward Propagation (from the input layer to output layer) - passes the training data through the network to produce initial network predictions based on the current model

parameters. Then, the network output predictions are fed into the cross-entropy loss function to calculate the error between network output predictions and the desired target output [214].

Backward Propagation (from the output layer towards the input layer), the model calculates the gradient of the loss function with respect to the current model parameters. Then, the GD algorithm updates the model parameters by taking a step size of η in the direction of minimising the loss function.
The process of forward and backward propagations are repeated during training the CNN until the loss function is minimised [137].

4.4.3. Gradient Descent Algorithm

GD is an optimisation algorithm used in training CNNs to minimise the loss function by iteratively taking a step size of η which can be in the opposite direction of the gradient in order to reach the lowest or minimum error value [156]. There are three variant algorithms of GD [215-217]:

- **Stochastic GD:** calculates the gradient of the loss function for each training data and then performs a single update for the model parameters for each training data, one by one. The advantages of stochastic GD are that the frequent updates to the model parameters give an insight into the performance of the model and the rate of improvement. However, the two main disadvantages of stochastic GD are that; (1) it is more computationally expensive than other GD methods as a result of the frequent updates to the model parameters for each training data, (2) frequent updates to the model parameters can cause the gradient of the loss function to fluctuate severely. The mathematical expression for updating the model parameters using stochastic GD can be written as:

$$\theta_{new}^{l} = \theta_{old}^{l} - \eta \frac{\partial}{\partial \theta_{old}^{l}} J(\theta; x_i, t_i) \tag{4.12}$$

Where θ_{new}^{l} is the updated model parameter value at l^{th} layer, θ_{old}^{l} is the current or old model parameters at l^{th} layer that needs to be updated, η is the learning rate and it is referred to as the step size of the model parameters being updated during the training process, $\frac{\partial}{\partial \theta_{old}^{l}} J(\theta; x_i, t_i)$ is the derivative of the loss function with respect to the

current model parameter θ_{old}^{l}, x_i is the training data and its corresponding target class is t_i.

- **Batch GD:** calculates the gradient of the loss function for the entire training data and then performs only one update for the model parameters. This process is called a cycle and it is referred to as a training epoch. The advantages of batch GD are that; (1) it is more computationally efficient than stochastic GD, as it performs only one update of the model parameters for every training epoch. (2) it produces a stable gradient and more rapid convergence due to fewer updates to the model parameters. However, the disadvantages of batch GD are that; (1) stable gradient can sometimes result in a state of convergence that is not optimal, (2) it requires a huge consumption of computational resources as the entire training data needs to remain in the memory and be available to the algorithm. The mathematical expression for updating the model parameters using batch GD can be formulated as:

$$\theta_{new}^{l} = \theta_{old}^{l} - \eta \frac{\partial}{\partial \theta_{old}^{l}} J(\theta) \tag{4.13}$$

- **Mini-Batch GD:** is the most commonly used algorithm for training the CNN [92, 128, 204]. It is a combination of stochastic and batch GD. Mini-batch GD splits the training data into mini-batches and updates the model parameters for every mini-batch of training data x_B, and its corresponding class y_B. The hyper-parameter B represents the batch size, e.g., how many training data are processed simultaneously to perform an update to the model parameters. According to [215, 217] the most common mini-batch sizes range between 50 and 256. However, the mini-batch size B should not be very large and its value depends on the size of the training data. The advantages of mini-batch GD are; (1) it produces more stable convergence than batch and stochastic GD, (2) the mini-batch algorithm is a more computationally efficient process, especially for extremely large sets of training data. The mathematical expression for updating the model parameters using mini-batch GD can be written as:

$$\theta_{new}^{l} = \theta_{old}^{l} - \eta \frac{\partial}{\partial \theta_{old}^{l}} J(\theta; x_{(i:i+B)}, t_{(i:i+B)}) \tag{4.14}$$

4.4.4. Learning Rate

The learning rate is a hyper-parameter that needs to be tuned. It refers to the step size of updating the model parameters during the training process [218]. The learning rate parameter plays an important part in training the CNN, due to two reasons. Firstly, if the learning rate is set too high, it is likely to overshoot or fluctuate around the minimum error value, secondly, if the learning rate is set too small, it may lead to slow convergence. As shown in Figure 4-4, the idea is to have a smooth minimisation process in order to have the loss function as close as possible to the optimal minimum value [219]. It is usually recommended to use a lower learning rate when training the neural network as it improves the network training performance compared to a higher learning rate [220-222].

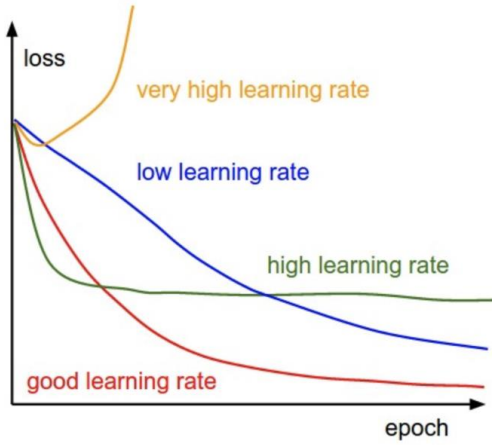

Figure 4-4: Effects of different learning rates [223]

4.5. The Implementation of a CNN

The application of the CNN to CM involves building a model to discover a relationship between the training data and its target class using supervised learning. Then, the performance of the trained model is evaluated by predicting the target class for new data (called testing data) based on the model relationships learned during the training process. The implementation step of the CNN begins by dividing the measured vibration data into N segments. Each data segment length being set to cover more than one period of the expected fault feature. Next, divide the entire set of data segments into three data groups called training data, validation data, and testing data. Following previous studies [224-226], the entire data segments will be divided randomly into 60% for training, 20% for

validation, and 20% for testing. Each data set serves a different purpose as discussed below.

4.5.1. Training Stage

The training stage is the first process of using CNN for CM. In this stage, the training data is fed to the CNN to find the optimal model parameters that best explain the model relationships between the training data and its target class. As discussed in Section 4.4.2, training CNN involves two main steps; forward and backward propagations. These two processes aim to find optimal model parameters by iteratively minimising the error between the desired target class and the network output prediction. The CNN training process involves the following steps:

1. Divide the measured vibration data into N segments. Each data segment length being set to cover more than one period of the expected fault feature
2. All data segments are divided randomly into 60% for training, 20% for validation, and 20% for testing.
3. Initialise the model parameters with a random value during the forward propagation process.
4. Starting with the forward propagation process, passing mini-batch training data and its target classes into several hidden layers of the CNN model, as discussed in Section 4.4.2 and their calculations from Equation 4.1 to Equation 4.9.
5. Implement the softmax function to generate the network output prediction, as seen in Equation 4.10.
6. Calculate the error between the desired target output and the network output prediction by implementing the cross-entropy loss function, see Equation 4.11.
7. Perform backward propagation by calculating the gradient of the loss function with respect to the current model parameters.
8. Update the model parameters for every mini-batch of training data and its corresponding target classes by iteratively taking step size of η in the direction of minimising the loss function, as seen in Equation 4.14.
9. Repeat steps 4 to 8 until the loss function is minimised.

4.5.2. Validation Stage

This is done by feeding the CNN with a set of unseen data (called validation data) that were not used in the training but follow the same distribution and model relationship as the training data. The validation step is commonly used for producing the final model and to avoid overfitting problem through early stopping.

Overfitting is a common problem that mostly occurs during the training process. It refers to a model that performs very well on the training data but poorly on new data which is not a part of the training data. This is because the model learned features that were specific to the training data and not present in other data, and consequently impacts negatively on the performance of the model on new data [227]. Therefore, early stopping is being adopted during training CNN to reduce the effect of overfitting and to improve the generality of the model. Early stopping defines the number of times that the loss on the validation data can be equal or larger to the previously smallest loss value before the training process is terminated [137]. Put simply, the early stopping process is the continuous monitoring the loss on the validation data after each training epoch. When the loss on the validation data improves (e.g., loss decreases to a lower error value), that would mean the generalisation ability of the model is improved, thus the model continues the training process. However, when the loss on the validation data starts to degrade (e.g., loss begins to increase to a higher error value), that would mean the model at this stage had reached the stage of beginning to over-fit the training data. Consequently, the training process is terminated (early stopping) to avoid overfitting.

4.5.3. Testing Stage

The testing stage is the final process of applying the CNN and is carried out to test the overall performance of the trained model on unseen data (called testing data) that were not used in the training and validation stages. The testing stage is conducted by feeding the trained CNN model with unseen data to evaluate its performance. The accuracy of any supervised classification model can be calculated from four possible classification outcomes (true positive, true negative, false positive and false negative) derived from the confusion matrix. These four possible classification outcomes will be discussed in more detail in Chapter Eight.

4.6. Key Findings

To summarize, the main key findings of this chapter are as follows:

- CNN is a type of deep learning method proposed by LeCun, Bottou [152] for handwritten digit classification.
- CNN has gained special attention as a promising form of deep learning. It has been successfully applied in several applications such as object detection, natural language processing and automated fault diagnosis.
- CNN is a supervised DNN algorithm and based on the idea that the training data needs to be labelled, and then allow the CNN to model a relationship between the input training data and the corresponding target classes.
- CNN integrates several features into a single deep hierarchical model, including: feature extraction, dimensionality reduction and classification. By integrating these features via a multi-layered configuration, it allows the CNN to learn representative features directly from the raw data and automatically identify different classes for a given set of data.
- The implementation procedure of the CNN for CM consists of three main steps, training, validation, and testing. In the training step, the input training data is fed to CNN to find the optimal model parameters that best model the relationships between the training data and its target class. In the validation step, a set of unseen validation data are fed into the CNN model to produce a final model and to avoid overfitting problem through early stopping. Testing is the final step of applying a CNN and is conducted by feeding the trained model with new data to measure the classification accuracy of the trained model.

Based on the key findings listed above, an automated approach based on a deep CNN is investigated for machinery CM. This automated approach has the capability to address the shortcomings in conventional manual methods by integrating both feature extraction and classification into a single model. Deep CNN applied for CM can be used to extract representative features directly from the raw vibration data and automatically determine the system's health condition with high diagnostic accuracy.

Chapter Five: Activation Functions for the Convolutional Neural Network

This chapter aims to look at the theoretical background of the activation functions. It starts with reviewing the most commonly activation functions used for deep CNN: Tanh, ReLU, LReLU and ELU. Then, the limitations of these activation functions are presented and discussed. Followed by, the hybrid IReLU-Tanh function proposed as a means of addressing the shortcomings, enhance the learning ability of the network, and improve the classification accuracy of the model. Finally, this chapter ends with a review of key findings.

5.1. Introduction

Existing automated approach based on deep CNN for CM focus on the design of the network architecture and the hyper-parameters tuning, including the number of layers, the use of batch normalisation, size of the convolutional filter etc [199]. However, several studies have claimed that the commonly used activation functions have critical drawbacks such as vanishing gradient problem and dying ReLU, which can affect the overall performance of deep architecture and result in low classification accuracy of the model [228, 229]. This chapter reviews existing activation functions used for deep CNNs, highlights their main shortcomings, and then proposes an IReUL-Tanh function to overcome the drawbacks and enhance the overall performance of deep CNN architecture.

5.2. Activation Function for a CNN

The activation function is a critical layer within the CNN architecture that has a crucial impact on the training process and hence on the overall performance of the network, see Section 4.3.3. It is used in CNNs to compute the weighted sum of the input, as shown in Figure 5-1, which is used to determine whether a neuron should be activated or not [197]. The main role of the activation function is to introduce nonlinear characteristics into the model [195, 196]. Generally, it transforms the feature map of the convolutional layer to generate a non-linear output in the range of $[0, \infty]$ or $[-1, +1]$ depending on the type of activation function. The activation functions allow the CNN to observe non-linear expressions of the data so that many complex problems can be resolved [198].

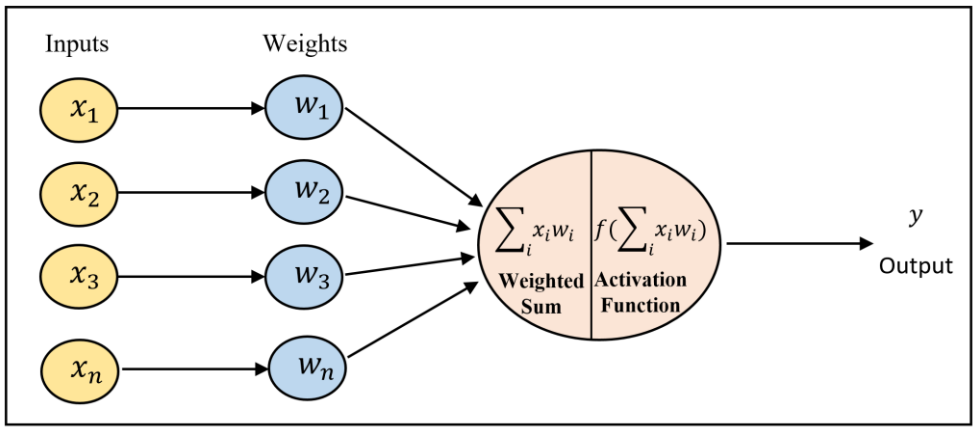

Figure 5-1: Typical neuron structure

Nowadays, deep CNN architecture can consist of many hidden layers. The activation function is applied to the output of the convolutional layer to introduce a nonlinear characteristic into the network and then pass its output to the next layer. As discussed in Chapter Four, during training of the CNN, forward propagation passes the training data from the input layer to the output layer to produce initial network output predictions [214]. The backward propagation step updates and adjusts the network parameters until the loss function is minimised. This update greatly depends on the behaviour of the activation function, and hence the choice of the activation function has a significant effect on the network training process. The mathematical expression for updating the network parameters during backward propagation can be written as [230]:

$$\theta_{new}^{l} = \theta_{old}^{l} - \eta \frac{\partial L}{\partial \theta_{old}^{l}} \tag{5.1}$$

where θ_{new}^{l} is the new or the updated model parameter, θ_{old}^{l} is the current or old model parameter at l^{th} layer that needs to be updated, η is the learning rate, and $\frac{\partial L}{\partial \theta_{old}^{l}}$ is the derivative of the loss function with respect to the current model parameters at l^{th} layer. and $\frac{\partial L}{\partial \theta_{old}^{l}}$ can be computed as:

$$\frac{\partial L}{\partial \theta_{old}^{l}} = \frac{\partial L}{\partial Z^{l}} * \frac{\partial Z^{l}}{\partial f^{l}} * \frac{\partial f^{l}}{\partial \theta_{old}^{l}} \tag{5.2}$$

where $\frac{\partial L}{\partial Z^{l}}$ is the derivative of the loss function with respect to the neuron output at the l layer, $\frac{\partial Z^{l}}{\partial f^{l}}$ is the derivative of the activation function after applying the function to the neuron output, and $\frac{\partial f^{l}}{\partial \theta_{old}^{l}}$ is the neuron output with respect to the current model parameters at the $(l-1)$ layer.

Deep CNNs architecture are observed to learn better representations of the features in the data sets, layer by layer. However, it has been reported that severe problems can appear within the deep CNN architecture caused by the activation function [14]. Common problem include vanishing gradient problems during the backward propagation through a multi-layered neural network, this is due to the value of the gradient decreasing exponentially to zero as it is propagated backwards [167]. This problem is exacerbated

when an activation function such as the Tanh function is used. This is due to the fact that the Tanh function saturates at -1 and $+1$ for large negative and positive inputs, with the value of the gradient close to zero [168].

During back propagation, the gradients of the loss function with respect to the current network parameters are updated (i.e., re-calculated), as shown in Equation 5.2. If a value close to zero is multiplied several times by other values close to zero, the output becomes very close to zero. In this way, the value of the gradient can decreases exponentially to zero as the backward propagation process goes deeper into the network [231]. This means the network parameters are not updated properly, and the loss function stops approaching the lowest error value [232]. At this stage, it can be said that the network is not being properly trained to produce the best model relationship between the training data and its target class. Thus, in the backward propagation process, the choice of activation function has a significant impact on the performance of the network training task of the CNN model [233]. To overcome the vanishing gradient problem, a number of activation functions have been developed in recent years, including ReLU, LReLU, and ELU [169-171], which are discussed in following subsections.

5.2.1. Hyperbolic Tangent Function

Tanh is a type of activation function used in neural network architecture [197]. It is a smooth zero-centred function whose output ranges between $[-1, +1]$ with an S-shape, see Figure 5-2. It can be considered a saturating activation function because it squashes large positive values to $+1$, and large negative values to -1, with the value of the gradient close to zero as shown in Figure 5-2. The Tanh function and its gradient can be expressed as [195, 234]:

$$f_{Tanh}(x) = tanh(x) \tag{5.3}$$

$$f_{Tanh}'(x) = 1 - tanh^2(x) \tag{5.4}$$

where $f_{Tanh}(x)$ is the output of the tanh function, x is the input data, exp is the exponential, and $f_{Tanh}'(x)$ is the gradient of the tanh function.

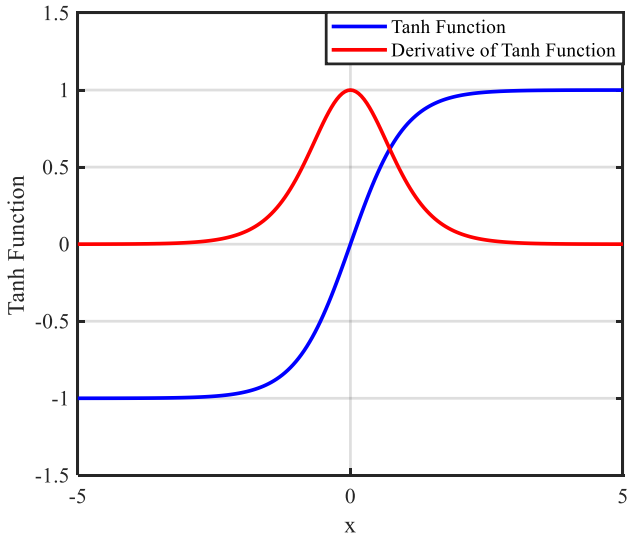

Figure 5-2: Tanh function and its derivative

In the past few decades, the Tanh function has been used as a standard activation function for neural network architecture [143]. However, with the introduction of deep CNN architecture, it was found that severe problem called the vanishing gradient problem caused by the form of the Tanh function. This begins to appear during the back propagation process as a result of the saturation of the Tanh function, which occurs at +1 and -1 for large positive and negative inputs, see Equation 5.3 [235]. This results in the value of the gradient of the neuron approaching to zero [171], as shown in Figure 5-2 and Equation 5.4. Thus, the Tanh function has shortcomings when used for local optimisation, this is because the value of the gradient for large positive and negative inputs become close to zero as the backward propagation process goes deeper into the network [236]. This process leads to the network parameters not being optimised and failing to provide the best possible relationship between the training data and its target class, resulting in poor training performance and low classification accuracy [65].

5.2.2. Rectified Linear Units

The ReLU function is one of the most widely used activation function in CNN architecture. It was proposed by Nair and Hinton [170] to address the shortcoming of the vanishing gradient problem that occurs with the Tanh function [237]. As seen in equation 5.5, the ReLU function performs a threshold operation, when the input value is less than or equal to zero ($x \leq 0$), the function output will be equal to zero, and the gradient will also be zero. A non-saturation feature for the positive value, when the input value is

greater than zero ($x > 0$) the ReLU function output is equal to the input and the gradient will be equal to +1 [212]. Thus, the ReLU function addresses the vanishing gradient problem associated with the Tanh function and improves the overall performance of the network [169]. The ReLU function and its gradient can be written as: [238, 239]:

$$f_{ReLU}(x) = max(0, x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{5.5}$$

$$f_{ReLU}'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{5.6}$$

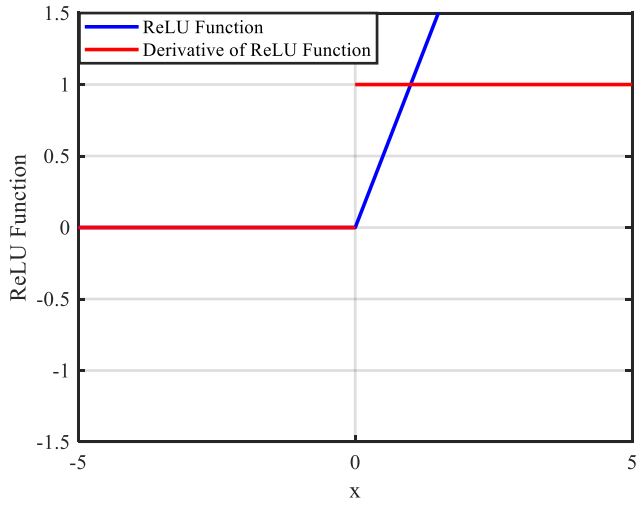

Figure 5-3: ReLU function and its derivative

The ReLU function is one of the most widely used activation function for training deep CNNs [240]. However a major drawback of the ReLU function is that it produces a value of zero for the gradient ($f_{ReLU}'(x) = 0$) when ($x \leq 0$) for all negative values, hence it discards important information contained in those values and generates the so-called "dead neuron" or "dying ReLU" problem [239]. A dead neuron once generated will output zero value and remain deactivated. This can lead to those dead neurons (deactivated) never be updated, and not participate in the training process. If a large number of neurons remain deactivated the overall performance of the model will be badly affected [232].

5.2.3. Leaky Rectified Linear Units

The LReLU is an improved version of ReLU function proposed by Maas, Hannun [171], to address the dying ReLU problem. The LReLU addresses the problem of deactivated

neurons in the ReLU function by giving negative inputs a non-zero fixed value, as seen in Figure 5-4 so that the information associated with the negative values is not lost [241]. In this way, the LReLU function enables neurons that would otherwise be deactivated, to remain active, preserving and updating them, and having them continue to participate in the training process. The non-zero fixed value for the negative input is determined by the hyper-parameter alpha (α) which is normally set as $\alpha = 0.01$, prior to the commencement of training [242].

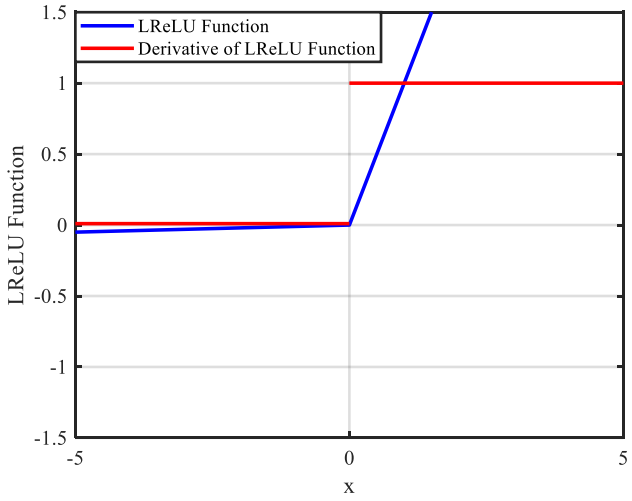

Figure 5-4: LReLU function and its derivative ($\alpha = 0.01$)

The LReLU function is the same as the ReLU function in terms of its non-saturation feature for the positive values. The output of the LReLU function for positive input ($x > 0$) is equal to the input itself, and the gradient is equal to +1, as shown in Figure 5-4, Equations 5.7 and 5.8. For negative input, the LReLU function output will be ($f_{LReLU}(x) = \alpha * x$), and the gradient is equal to α, as seen in Equations 5.7 and 5.8 [137] and Figure 5-4. The LReLU function and its gradient are computed as [171]:

$$f_{LReLU}(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{5.7}$$

$$f_{LReLU}'(x) = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases} \tag{5.8}$$

where α is a small real number.

Xu et al., have reported that the LReLU function performs slightly better than the ReLU function for image [238] and time series classification tasks [243]. However, the

downside of the LReLU function is that it adds a hyper-parameter (α) to the CNN architecture, which means that the hyper-parameter needs to be specified based on a trial and error, limiting the enhancement provided by LReLU over ReLU [244]. Additionally, a non-zero fixed value ($\alpha = 0.01$) for negative inputs ($x \leq 0$), may still lead to a risk of the occurrence of the vanishing gradient, and consequently the network parameters may not be adequately updated during the training process [232, 245].

5.2.4. Exponential Linear Unit

The ELU is another type of activation function used in CNN architecture developed by Clevert, Unterthiner [169] to enhance the training process and hence improve the overall performance of the model. As shown in Figure 5-5, the ELU function avoids the occurrence of a non-zero fixed value and dying neurons during training by using the exponential function for negative inputs. However, it saturates for large negative inputs with the saturation level controlled by the hyper-parameter (α), which is normally set to $\alpha = 1.0$ [239, 246].

As with the ReLU and LReLU, the output of the ELU function for positive inputs is equal to the input, and its gradient value is equal to +1, as shown in Equations 5.9 and 5.10, and Figure 5-5. Thus, the ELU function is the same as the ReLU and LReLU functions in terms of its non-saturation feature for positive values. For negative input values, the ELU function is similar to the Tanh function as shown in Figure 5-5. The ELU function smoothly approaches a value equal to ($-\alpha$) for negative inputs ($x \leq 0$) [244] as seen in Figure 5-5. The ELU function output ranges between $[-\alpha, \infty]$. The ELU function and its gradient can be expressed as [137, 242]:

$$f_{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha * (exp(x) - 1), & x \leq 0 \end{cases} \tag{5.9}$$

$$f_{ELU}{}'(x) = \begin{cases} 1, & x > 0 \\ f(x) + \alpha, & x \leq 0 \end{cases} \tag{5.10}$$

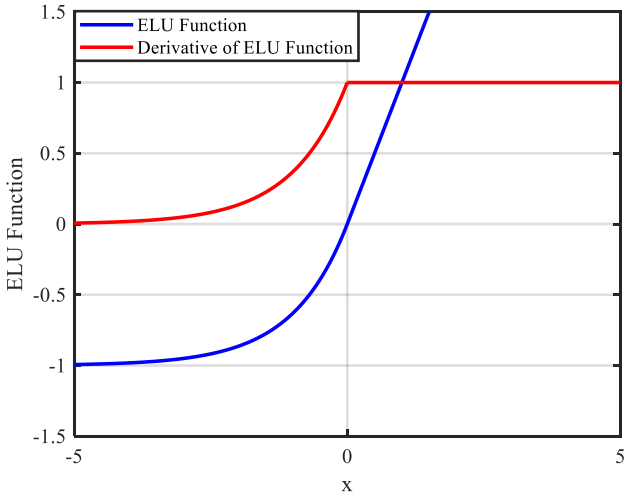

Figure 5-5: ELU function and its derivative ($\alpha = 1.0$)

It has been shown that the ELU function outperforms both ReLU and LReLU for both image classification [169] and time-series classification [247]. However, the ELU function is similar to the LReLU function in terms of adding a hyper-parameter (α) to the CNN architecture. The saturation level for the negative values is controlled by the hyper-parameter, the optimum value of the (α) parameter needs to be determined by a trial and error process. However, if the hyper-parameter (α) is set to a velue less than one (e.g., $a = 0.5$), the ELU function will saturate at (-0.5) and the value of the gradient becomes close to zero which means the model parameters not being sufficiently optimised during the training process [244, 245, 248].

5.3. Recent Reviews of Activation Functions

Several studies have applied the above activation functions to automated fault diagnosis using CNNs, recent works are as follows:

- Jiang, He [158] in 2018 proposed automated fault diagnosis for a wind turbine gearbox based on multi-scale CNN (MSCNN). The proposed MSCNN architecture consisted of three different multi-scales, and each scale comprised of a convolutional layer, ReLU function and pooling layer followed by a concatenation layer, fully connected, and finally a softmax layer. In this study, the author stated that the ReLU function was selected for the proposed MSCNN architecture to prevent vanishing gradient problems and accelerate the convergence of the CNN. Experimental vibration data was used to evaluate the proposed MSCNN architecture, and it was

claimed that an outstanding classification accuracy was obtained using the proposed architecture compared to a traditional CNN with a single scale.

- Gong, Chen [128] in 2019 proposed the CNN-SVM method for automated fault diagnosis of rotating machinery. The proposed method consisted of convolutional layer, ReLU function, pooling, global average pooling, softmax layer, and finally a SVM as a classifier. In this study, it was reported that the ReLU function is the most widely used than the Tanh function and was employed in the proposed CNN-SVM method to avoid the vanishing gradient problem. A comparison was carried out between the Tanh and ReLU functions and it was claimed that the proposed CNN-SVM method with the ReLU function outperformed the Tanh function giving more accurate diagnoses.
- Wu, Jiang [249] in 2019 developed a one-dimensional (1-D) CNN method to learn features directly from the raw vibration data collected from a PG. The developed 1-D CNN method was constructed with convolutional layer, ReLU function and pooling layer, followed by a fully connected layer and finally a softmax layer. In this study, it was reported that the ReLU is the most widely used activation function for CNN, and that was the reason it was employed in the 1-D CNN method. Experimental vibration data was used to evaluate the effectiveness of the developed 1-D CNN method, and it was reported that the developed method had a higher classification accuracy compared to shallow learning method.
- Han, Tang [250] in 2019 presented an enhanced CNN (ECNN) with enlarged receptive fields to improve the feature learning ability of a CNN for automated fault diagnosis of a PG. In this study, it was again reported that the ReLU function is the most widely used activation function in CNN methods and it was selected for the ECNN method to alleviate the risk of a vanishing gradient and to accelerate the convergence of the network. The proposed ECNN method was evaluated through the use of experimental vibration data and it was claimed that superior classification performance was achieved.
- Chen, Hu [203] in 2019 proposed a deep CNN (DCNN) based multi-sensor data fusion technique for identifying the health of a PG. The proposed method was constructed with convolutional, batch normalisation, ReLU function and pooling layers, followed by a fully connected, dropout layer, and finally a softmax layer. In this study, the ReLU function was adopted to prevent vanishing gradient problem and

its effectiveness was evaluated using experimental vibration data. The DCNN method was compared with SVM and BPNN methods and it was claimed that the proposed DCNN method gave better identification.

- Hsiao, Shivam [243] in 2020 proposed automated fault diagnosis using a 1-D CNN with ensemble learning. The proposed method consisted of convolutional, batch normalisation, LReLU function, pooling, fully connected, and softmax layers. In this study the performance obtained using the LReLU function was compared with ReLU function. The two methods were evaluated using experimental vibration data and it was claimed that a higher classification accuracy was achieved using LReLU compared to the traditional CNN with the ReLU function.
- He, Shao [251] in 2020 proposed an ensemble transfer CNN for automated fault diagnosis of a gearbox. The proposed method was constructed with convolutional, LReLU function, pooling, fully connected, and finally softmax layer. In this study it was reported that the LReLU successfully addressed the problem of dying ReLUs by giving the negative inputs a non-zero fixed value. The proposed method was evaluated using experimental vibration data and a comparison was carried out with other deep learning methods, including LSTM, SAE, and DBN. It was claimed that superior classification accuracy was achieved using the presented method compared to the others.
- Li, Li [247] in 2020 developed automated fault diagnosis based on a DNN for gear fault diagnosis. In this study, the ELU function was employed in the proposed DNN method to avoid the vanishing gradient problem, and to overcome the limitation of the ReLU function due to dead neurons. The developed DNN method was evaluated using experimental vibration data. The classification accuracy achieved from the developed DNN method was compared with the results obtained from the DNN with the ReLU function. It was reported a better classification accuracy was achieved using the DNN method with the ELU function compared to the ReLU function.
- Cao, He [252] in 2020 presented automated fault diagnosis for rotating machinery based on a multi-scale 1-D CNN. The presented method consisted of three different multi-scales, and each scale had a different convolutional filter size, followed by batch normalisation layer, ELU function, pooling layer, fully connected, and finally softmax layers. In this study, the ELU function was employed to avoid the vanishing gradient problem and address the shortcoming of the ReLU function due to its setting negative

input values to zero. Experimental vibration data was used to evaluate the multi-scale 1-D CNN method. In addition, a comparison was carried out between using ELU and ReLU functions. It was claimed that the presented method with ELU function outperformed the ReLU function with a better diagnostic performance.

5.4. Limitations of Existing Activation Functions

This section summarises the shortcomings of the existing activation functions, including Tanh, ReLU, LReLU, and ELU as follows:

- **Vanishing gradient:** The Tanh activation function saturates at -1 and $+1$ for large positive and negative inputs, and the gradient value of the Tanh function approaches zero, as shown in Figure 5-2. This means that when the number of CNN layers increases, the gradient value decreases exponentially towards zero as it is back propagated through a multi-layered neural network. Consequently, the network parameter updates become very small which is a particular problem for deep CNN architecture [65, 171, 235, 236].
- **Dying ReLU:** An activation function such as ReLU can generate a dead neuron as output and be deactivated for the duration of the training process. This occurs because the gradient value of the ReLU function is zero when the input is equal to or less than zero, as shown in Figure 5-3. Hence, it discards important information contained in these values. Consequently, the dead neurons (deactivated) are never updated or contribute to the final output [232, 239].
- **Hyper-Parameter alpha (α):** Activation functions such as LReLU and ELU add a hyper-parameter (α) to the CNN architecture. This hyper-parameter determines the gradient value for the negative input, as seen in Equations 5.8 and 5.10. However, the hyper-parameter needs to be manually specified based on a trial and error process, limiting the enhancement provided. As discussed in Section 5.2.3, the gradient value of the LReLU function is a non-zero fixed value for all negative inputs. Consequently, it may lead to a risk of occurrence of the vanishing gradient problem and consequently the network parameters may not be adequately updated during the training process. On the other hand, as illustrated in Section 5.2.4, the gradient value of the ELU function is controlled by the hyper-parameter that determines the saturation level for negative inputs. This means that if the hyper-parameter is set to a value less than one

(e.g., $a = 0.5$), the ELU function will saturate at (-0.5), and hence the gradient value of the ELU function become close to zero [232, 244, 245, 248].

5.5. Proposed Activation Function

Based on the challenges and limitations of the existing activation functions described above, this research proposes an improved activation function called IReLU-Tanh function for deep CNN architectures. The proposed IReLU-Tanh function is inspired by the shared feature of several activation functions including ELU, LReLU and ReLU for covering the positive region, and the properties of the Tanh function for covering the negative region. The IReLU-Tanh function is developed to enhance the network training in both the forward and backward propagation processes, and hence improve the overall performance of the network to obtain the best model relationship between the training data and its target class with minimum error value. The mathematical expression of the proposed IReLU-Tanh function and its gradient are computed as:

$$f_{IReLU-Tanh}(x) = \begin{cases} x, & x > 0 \\ tanh(x), & x \leq 0 \end{cases} \tag{5.11}$$

$$f_{IReLU-Tanh}'(x) = \begin{cases} 1, & x > 0 \\ 1 - tanh^2(x), & x \leq 0 \end{cases} \tag{5.12}$$

The proposed IReLU-Tanh function is the same as the ELU, LReLU and ReLU functions in terms of its non-saturation feature for positive values. During the forward propagation process, when the input value is greater than zero $(x > 0)$, the proposed IReLU-Tanh function generates an output that is equal to the input as seen in Equation 5.11. In the case of the backward propagation process, the value of the gradient is equal to +1, as shown in Figure 5-6 and Equation 5.12. By adopting the non-saturation feature for covering the positive region, the proposed IReLU-Tanh function addresses the vanishing gradient problem in the same manner as the ELU, LReLU, and ReLU functions.

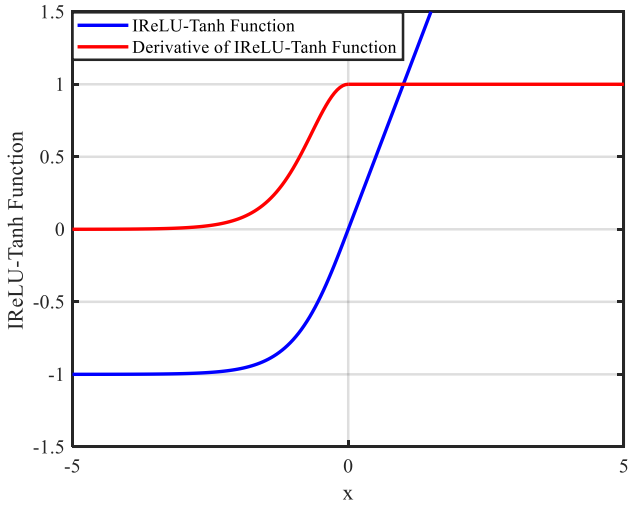

Figure 5-6: The proposed IReLU-Tanh function and its derivative

For negative input values, the IReLU-Tanh function is inspired by the Tanh function for covering the negative region. The IReLU-Tanh function smoothly approaches a value equal to -1, as shown in Figure 5-6, and Equation 5.11, but like the Tanh function, saturates for large negative inputs. By adopting the feature of covering the negative region from the Tanh function, the IReLU-Tanh function addresses the shortcomings of dying neurons in the ReLU function and a non-zero fixed value in LReLU function. This is because the IReLU-Tanh function generates an output for all negative input values during both the forward and backward propagation processes. Consequently, the IReLU-Tanh function allows the network to preserve neurons with negative values being optimised during the backward propagation process. Hence, the IReLU-Tanh mitigates the vanishing gradient problem for negative inputs, to some extent, and improves the overall performance of the network during the training process.

Besides the above advantages, the IReLU-Tanh function does not require adding a hyper-parameter to the architecture. This means that the saturation level for the negative input is determined by the Tanh function, instead of adding a hyper-parameter that needs to be tuned based on a trial and error process, as required by the LReLU and ELU functions. Another reason for adopting the Tanh function for covering the negative region instead of an exponential function as in the ELU, is because the Tanh function saturates earlier than the ELU function for negative inputs, consequently, it enhances the stability of the network during forward propagation. In addition, as seen in Figure 5-7, the Tanh function

has a higher gradient value for small negative inputs compared to the ELU function, and hence it enables a higher update for the model parameters in the earlier layers.

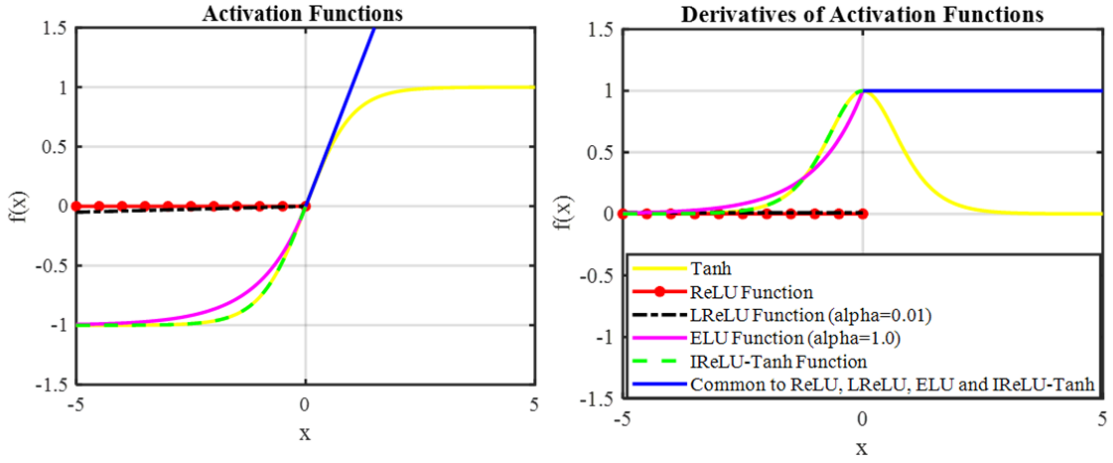

Figure 5-7: Comparison between (a) the proposed IReLU-Tanh function and the existing functions, (b) corresponding derivatives

The flowchart shown in Figure 5-8 demonstrates the proposed IReLU-Tanh function for CNN. The implementation steps of the CNN architecture with the proposed IReLU-Tanh function start with feeding the training data into the deep CNN to obtain the best model relationship between the training data and its target class with the minimum error value. Then, the trained model will be tested with unseen data to evaluate the overall performance of the trained CNN model in the classification task.

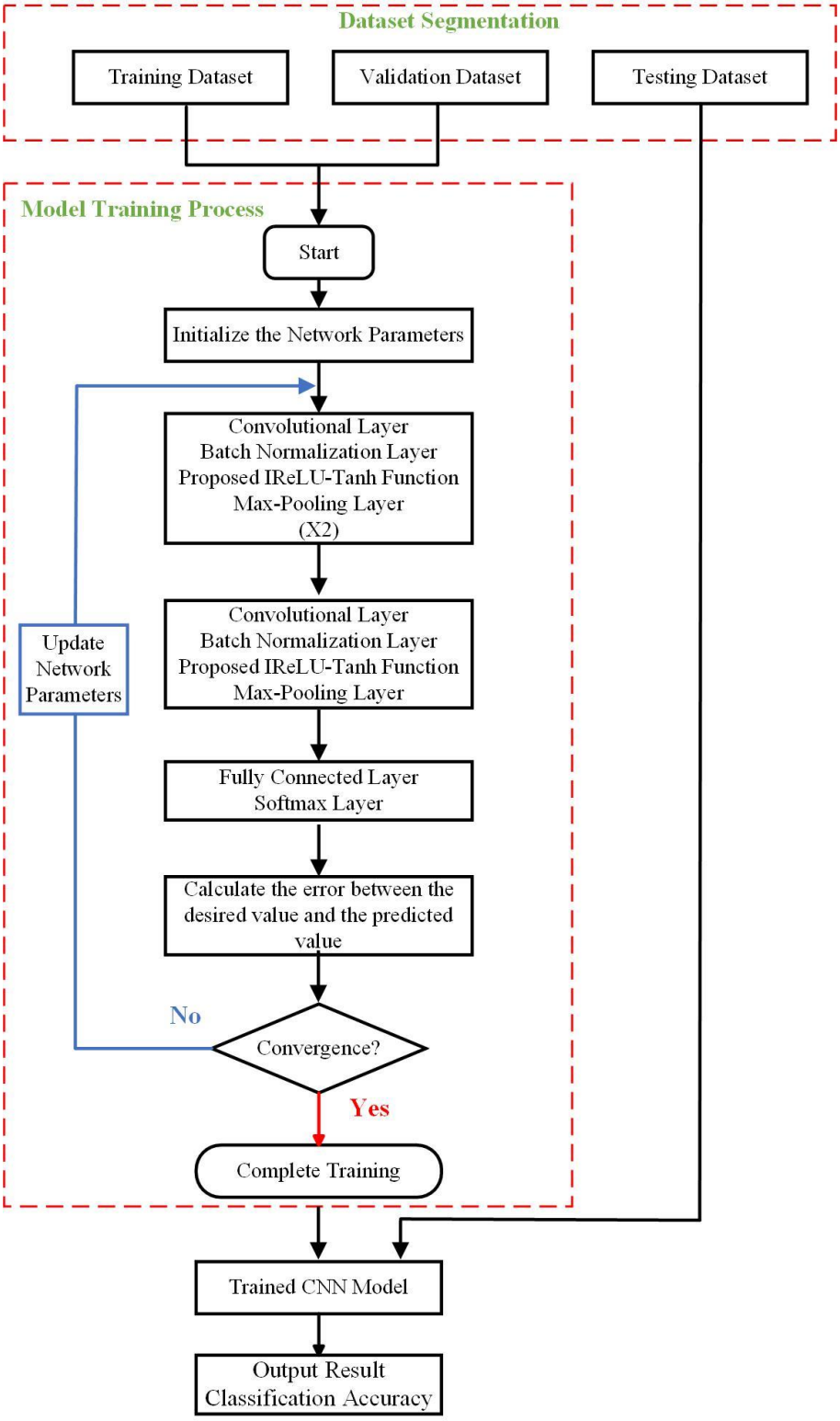

Figure 5-8: Flowchart of the proposed IReLU-Tanh function for CNN

5.6. Key Findings

To conclude, the following key findings are as follows:

- The activation function is a critical layer within the CNN architecture that has a crucial impact on the training process and hence on the overall performance of the network.
- Several studies have demonstrated that existing activation functions have severe drawbacks, which can affect the overall performance of CNNs and lead to a lower classification accuracy.
- It has been reported that the vanishing gradient is the most common problem in deep architecture caused by the Tanh activation function. This leads to a significant effect on the network training process and hence on the overall performance of the model.
- A number of activation functions have been developed in recent years to overcome the vanishing gradient problem, such as ReLU, LReLU, and ELU functions. However, these activation functions still have some critical drawbacks such as the dying ReLU problem, non-zero fixed value, and adding a hyper-parameter to the network architecture.

As a result of the above key findings, a hybrid activation function (IReLU-Tanh) has been developed to address the shortcomings in the existing activation functions. The advantages of the proposed IRaLU-Tanh function are: (i) it shares the non-saturation feature with several activation functions such as ReLU, LReLU and ELU for covering the positive region and so addresses the vanishing gradient problem of the Tanh function; (ii) for the negative region, it adopts the advantage of covering the negative region from the Tanh function, and so addresses the shortcomings of dying neurons in the ReLU function and non-zero fixed value; and (iii) unlike the LReLU and ELU functions, the proposed IReLU-Tanh function does not require adding a hyper-parameter to the network architecture. Therefore, the proposed IReLU-Tanh function can enhance the learning ability of the network during the training process, and hence improve overall performance of the network to obtain the best model relationship between the training data and its target class with minimum error value.

Chapter Six: Experimental Facilities and Data Acquisitions

This chapter describes the test system used in this study. It starts with the test rig components and then the instruments used to carry out the measurements including the accelerometer, thermocouple, encoder and data acquisition system. Finally, the fault simulation and the experimental procedure are presented in detail.

6.1. Introduction

An important element of CM research is the introduction of defects in a controlled manner into a practical system to provide real data similar to that found in industrial applications. A test rig was developed in the Centre for Efficiency and Performance Engineering Laboratory (CEPE) at the University of Huddersfield. This laboratory provides easy access to all necessary experimental facilities and required instruments.

In order to evaluate the method proposed in Chapters Four and Five for experimental data analysis, an experiment was carried out on the PG. The test rig is carried out in a way that is similar to real industrial applications where the faults can be simulated in a controlled manner, and the sensor data can be measured accurately. The reason for choosing a PG is its wide use in industrial applications such as wind turbines and helicopters. The test rig and experimental procedure will be discussed in this chapter.

6.2. Test Rig Development

The experimental work was performed on the PG test rig, as shown in Figure 6-1 and Figure 6-2. It is consisted of two parts: the first part consists of a drive motor, two types of PGs and a DC generator to act as load. The second part consists of the instrumentation required to acquire the vibration data, temperature, and shaft speed.

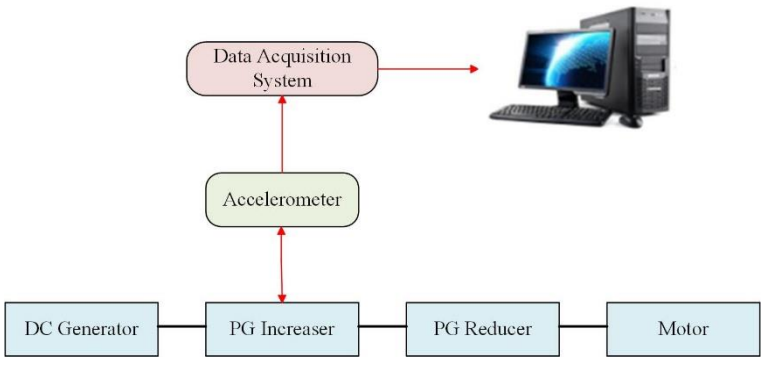

Figure 6-1: Schematic diagram of the test rig

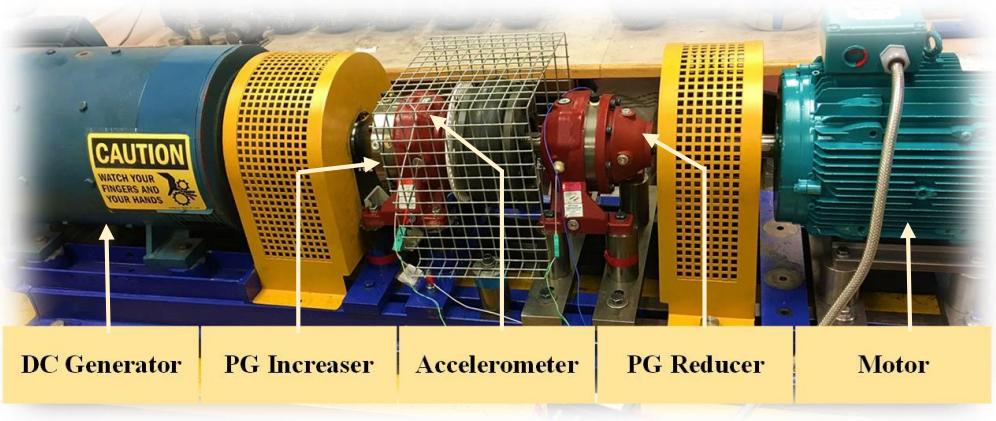

Figure 6-2: Test rig

6.2.1. Motor

A Brook Crompton induction motor is employed in the test rig to drive the system, see Figure 6-2. Motor specifications are shown in Table 6-1. A flexible coupling is used to connect the motor with the PG reducer.

Table 6-1: Motor specifications Brook Crompton

Motor Specifications	
Number of Phases	3
Number of Poles	4
Rated Voltage	415 V
Current	22 A
Shaft Speed	1490 RPM

6.2.2. Planetary Gearbox

Two types of PG manufactured by STM Power Transmission Ltd were used in this research, as shown in Figure 6-2. The first PG with transmission ratio of 5.76 was connected with the input shaft of the motor to reduce the rotational speed. It consists of the sun gear, four planet gears, fixed ring gear, and the carrier connected to the output shaft. The specifications of the PG reducer are shown in Table 6-2.

Table 6-2: PG reducer specifications

PG Reducer Specifications	
Number of Planet Gears	4
Ring Gear Teeth	62
Planet Gear Teeth	24
Sun Gear Teeth	13
Transmission Ratio	5.76

The second PG with transmission ratio of 7.2 was connected with the PG reducer and acts to increase the rotational speed. As shown in Figure 6-2, the carrier of the PG increaser was connected to the input shaft of the PG reducer and the sun gear was connected to the output shaft. The PG increaser consists of carrier, three planet gears, sun gear and fixed ring gear. In this study, the simulated defects were in one of the planet gears and sun gear of the PG increaser. The specifications of the PG increaser are listed in Table 6-3.

Table 6-3: PG increaser specifications

PG Increaser Specifications	
Number of Planet Gears	3
Ring Gear Teeth	62
Planet Gear Teeth	26
Sun Gear Teeth	10
Transmission Ratio	7.2

6.2.3. DC Generator

As seen in Figure 6-2, the DC generator is employed in the test rig to provide different loads to the system. The DC generator converts mechanical energy of rotation to electricity, which is dissipated as heat in a resistor bank. The specifications of the DC generator are shown in Table 6-4.

Table 6-4: DC generator specifications

DC Generator Specifications	
No	G6380/N
Size	SD 200XLC
Power	85 kW
Speed	1750 RPM
Mass	48.2 Kg

6.2.4. Data Acquisition System

A typical way of transforming a physical phenomenon into a digital format is to use a data acquisition system (DAQ). It is used to collect different types of data such as vibration, temperature and current. It converts the signal to a series of numeric (digital) values which can then be stored and manipulated by a computer. For example, the accelerometer converts movement due to vibration to an electrical signal which is then amplified to increase the magnitude of the signal to be input to the DAQ for recording. The DAQ converts the signal from analogue to a digital format and then sends the acquired signal to the computer for analysis task. The specifications of the DAQ are presented in Table 6-5.

As seen in Table 6-5, the vibration data was collected and measured at a sampling rate of 100 kHz. The reasons for choosing a high sampling rate are as follows: (i) according to the Nyquist theorem, the sampling frequency must be at least twice the maximum frequency of interest to avoid aliasing, and since the accelerometer has an upper limit of 12 kHz, a sampling rate of 100 kHz is more than adequate. (ii) A high sampling rate provides better time resolution of the waveform.

Table 6-5: DAQ PD2-MF-16-500/16L PCI specifications

DAQ Specifications	
Number of Channels	12 differential, 16 single ended
Data Resolution	16 bits
Sampling Rate	100 kHz
Max Working Voltage	12 V

6.2.5. Accelerometer

The accelerometer is employed to measure the vibration signal. In this test rig, the vibration signal is collected using a piezo-electric accelerometer, PCB model 338C04, see Figure 6-2, mounted vertically on the housing of the PG increaser. This type of accelerometer is the most commonly used for vibration measurement. The piezo-electric accelerometer has a wide frequency range to capture the vibration signals, both low and high frequency. The accelerometer is connected to a charge amplifier, and then the amplified output signal is passed to the DAQ for recording and then to the computer for analysis. The specifications of the accelerometer are listed in Table 6-6.

Figure 6-3: Accelerometer PCB model 338C04

Table 6-6: Accelerometer specifications

Accelerometer Specifications	
Sensitivity	100 mv/g
Sensitivity Tolerance	±10%
Frequency Range (±5%)	0.3 Hz to 12 kHz
Resonant Frequency	≥35 kHz
Transverse Sensitivity	≤5%
Temperature Range (Operation)	-53 to +93 ℃
Output Voltage	8 to 12 VDC
Measurement Range	±50g pk

6.2.6. Encoder

A typical way of measuring instantaneous angular speed is via an encoder installed at the end of the motor shaft. The incremental encoder used in the test rig is a Hengstler type R132, as shown in Figure 6-4. The specifications of the encoder used in the test rig are presented in Table 6-7.

Figure 6-4: Hengstler encoder

Table 6-7: Encoder specifications

Encoder Specifications	
Product Series	R132
Supply Voltage	10 to 30 V Dc
Weight	50 g
Max. pulse frequency	5V=300 kHz, 10-30V=200kHz
Maximum Speed	6000 RPM
Minimum Operating Temperature	-10℃
Maximum Operating Temperature	+60℃

6.2.7. Thermocouples

A typical way of measuring temperature is to use a thermocouple. It has been fitted on the PG housing and connected to the DAQ system. The specifications of the thermocouple are shown in Table 6-8.

Table 6-8: Thermocouple specifications

Thermocouple Specifications	
Type	K
Probe Length	2 m
Temperature Range	-60 ℃ to +350 ℃
Termination Type	Miniature Plug
Standards Met	IEC
Response Time	Fast

6.2.8. Fault Simulation and Seeding

In this research, five gear conditions were used to study the effects of tooth breakage faults with different levels of severity. The first condition was for healthy condition (no fault exist in the system). Fault simulation was performed sequentially, with one fault present at a time. Two types of fault severities were seeded into the sun and planet gear, as shown in Figure 6-5. The small fault was seeded by the removal of 30% of the length

of one tooth on the gear. The larger fault was seeded by extending the first fault to remove 60% of the length of one tooth of the gear. For ease of reference, the faults are labelled as follows: Sun-F1 and Planet-F1 refer to the 30% defects in the sun and planet gears respectively. Sun-F2 and Planet-F2 refer to the 60% defects in the sun and planet gears respectively, see Table 6-9.

Table 6-9: Seeded defect size and label for sun and planet gears

Gear No.	Condition	Percentage Tooth Removed	Defect Label
1	"Healthy"	---	Baseline
2	Sun gear with small fault	30%	Sun-F1
3	Sun gear with large fault	60%	Sun-F2
4	Planet gear with small fault	30%	Planet-F1
5	Planet gear with large fault	60%	Planet-F2

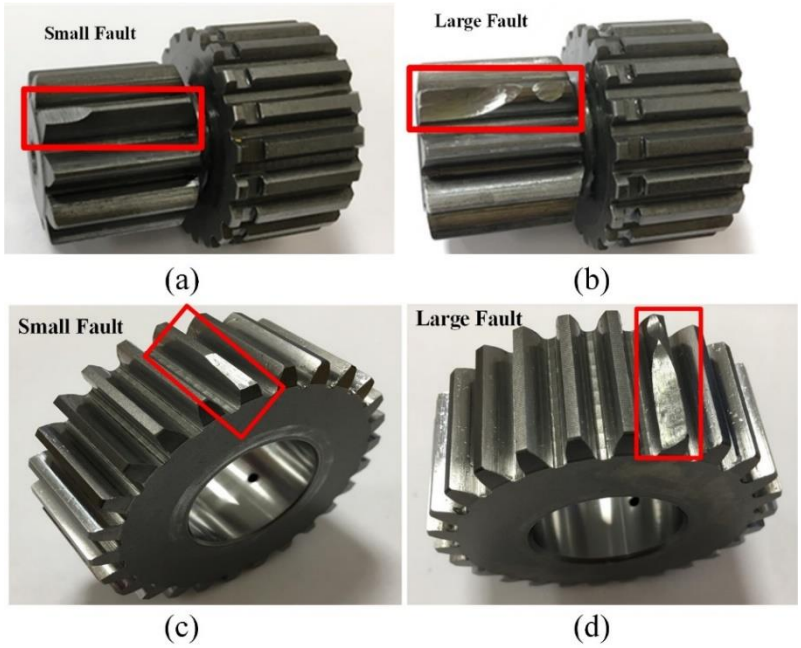

Figure 6-5: Seeded defects (a) 30% (b) 60% on sun gear, and (c) 30% (d) 60% on planet gear

6.3. Experimental Procedure

The vibration data was collected using the following experimental procedure:

1. The PG test rig was allowed to run for 30 minutes to warm up before collecting the vibration data.
2. All vibration data was collected under $60\% \approx 894\ rpm$ of the full motor speed and under different load conditions: zero, 25%, 50%, 75%, and 90% of full load.
3. The first PG reduces the shaft speed from 894 to 155.2 rpm. Then, the second PG increases the speed from 155.2 to 1117 rpm.
4. The vibration data was collected and measured at a sampling rate of 100 kHz for a period of 30 seconds. The length of the recorded vibration data for each data was $100000*30=3x10^6$ data points.
5. The collected vibration data was for five different gear conditions, baseline and two types of fault severities on one of the planet gears and the sun gear.
6. The vibration data was collected using the accelerometer mounted vertically on the PG increaser housing.
7. The baseline data was measured first.
8. The second set of measurements was for the planet gear fault, first with the small fault and then the large fault.
9. The third set of measurements was for the sun gear fault, first with the small fault and then the large fault.
10. For each measurement, three sets of vibration data were recorded to ensure that the obtained data are consistent.

Chapter Seven: Vibration Analysis Using Conventional and Enhancement Methods

In this chapter, the collected vibration signals are analysed using conventional signal processing methods and Multipoint Optimal Minimum Entropy Deconvolution Adjusted (MOMEDA). The chapter starts with the analysis of time and frequency domains. Then it presents the results obtained from applying the MOMEDA method to the collected vibration signals. Finally, the results obtained from these methods are discussed.

7.1. Introduction

The collected vibration data from the baseline and four defective gears will be analysed in this chapter, beginning with the time domain signal. Statistical parameters such as RMS and kurtosis are calculated to explore trends in the time domain data. This is followed by frequency domain analysis to investigate vibration features caused by rotating components. Next, the MOMEDA technique will be applied to the collected vibration data to enhance the periodic fault impulse features, and then envelope spectrum analysis is applied. Finally, the results are discussed and summarized at the end of this chapter.

7.2. Experimental Results Obtained Using Conventional Signal Processing Techniques

The analysis of the collected vibration data begins with time domain analysis Section 7.2.1, and frequency domain analysis Section 7.2.2.

7.2.1. Time Domain Analysis

The analysis of the vibration data begins with exploring the time domain of different cases; baseline, Sun-F1, Sun-F2, Planet-F1, and Planet-F2, under loads of zero, 25%, 50%, 75% and 90% of full load, as shown in Figure 7-1 to Figure *7-4*. In all four figures the blue trace is for the baseline. In Figures 7-1 and 7-3 the black trace is for the Sun-F1 and Planet-F1 faults, respectively. In Figures 7-2 and 7-4 the red trace is for the Sun-F2 and Planet-F2 faults, respectively.

Figure 7-1 and Figure *7-2* show the time domain vibration signals for baseline, Sun-F1 and Sun-F2 under different loads. It can be seen that, generally, increasing the load results in an increased in the amplitude of the vibration signal. For both fault cases, under loads of 50%, 75% and 90% of full load, the signals show peaks produced by both the Sun-F1 and Sun-F2. However, for both fault cases under low load conditions, the time domain signals do not show any clear differences.

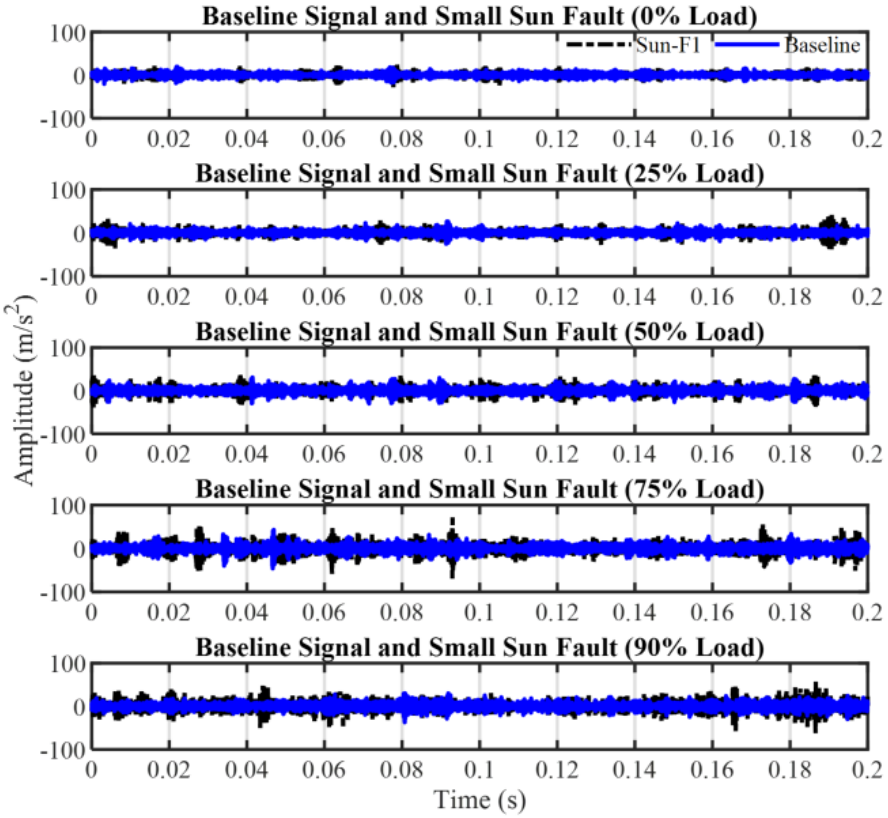

Figure 7-1: Time domain signal for baseline (blue) and Sun-F1 (black)

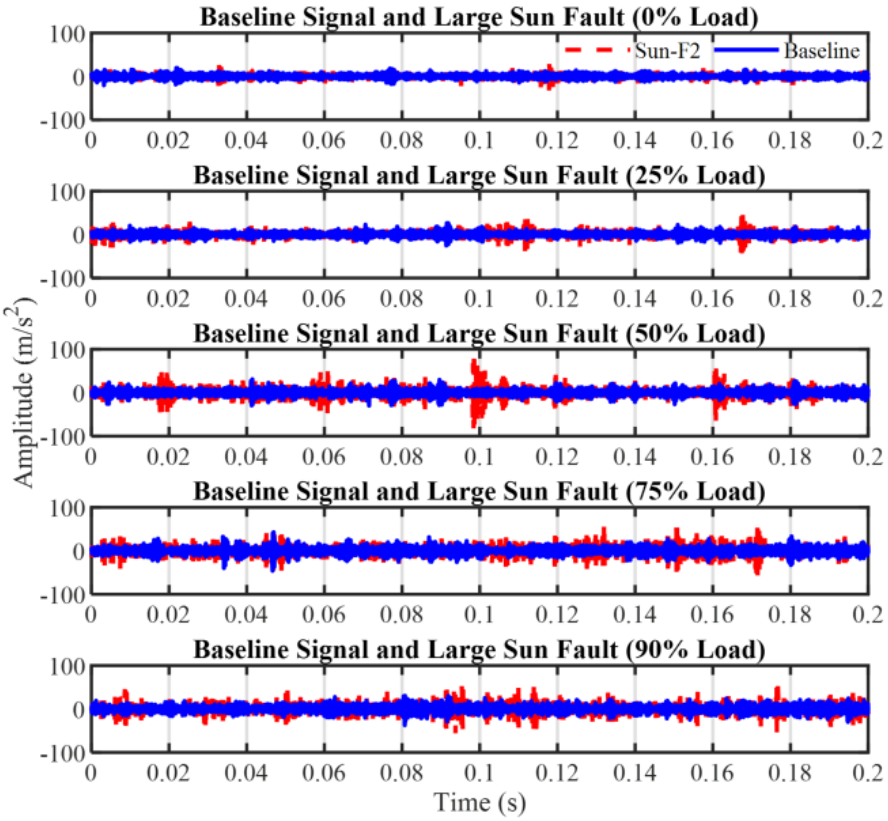

Figure 7-2: Time domain signal for baseline (blue) and Sun-F2 (red)

Figure 7-3 and Figure *7-4* show the time domain signals for the baseline (blue trace), Planet-F1 (black trace), and Planet-F2 (red trace), respectively. It can be observed that the signal amplitude measured with the Planet-F2 fault at 90% of full load demonstrates the presence of peaks generated by the defect. However, for both fault cases Planet-F1 and Planet-F2, under low loads, there are no significant changes in signal amplitude due to the presence of either fault.

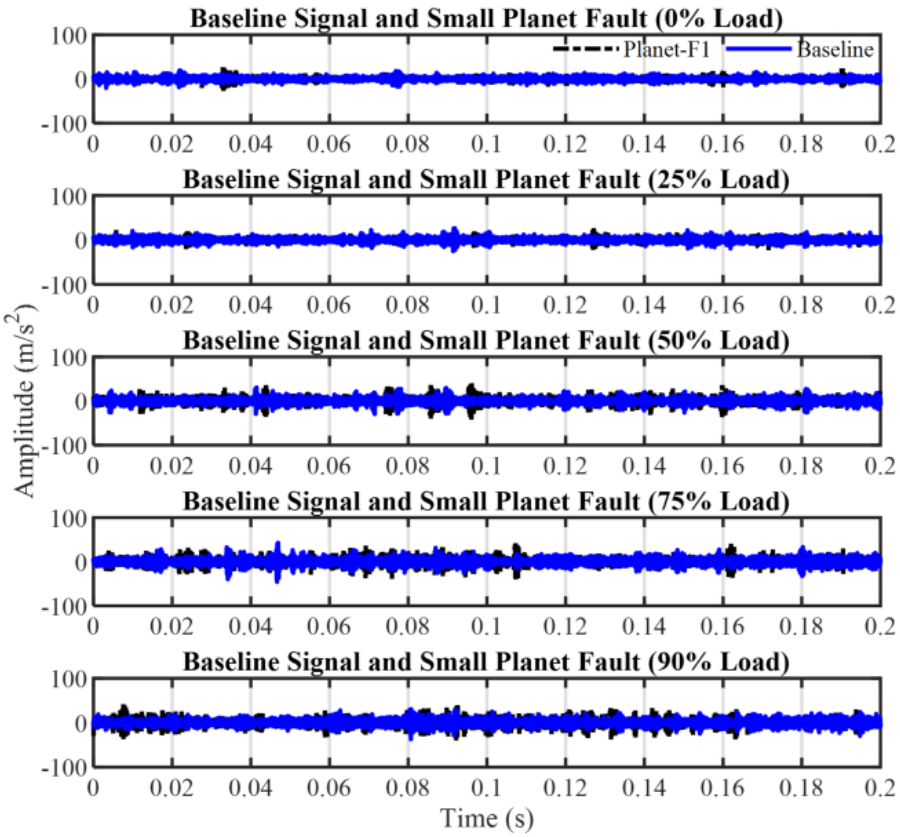

Figure 7-3: Time domain signal for baseline (blue) and Planet-F1 (black)

As shown in Figure 7-1 to Figure *7-4*, it can be said that the amplitude of some peaks that appeared in the vibration signals of the faulty sun gear are higher than the peaks that appear in the signals for the faulty planet gear. This is because the PG consists of three planet gears that not only rotate around their own centres but also revolve around the centre of the sun gear. This leads to the fault impulse generated by the planet defect being masked by other vibration components.

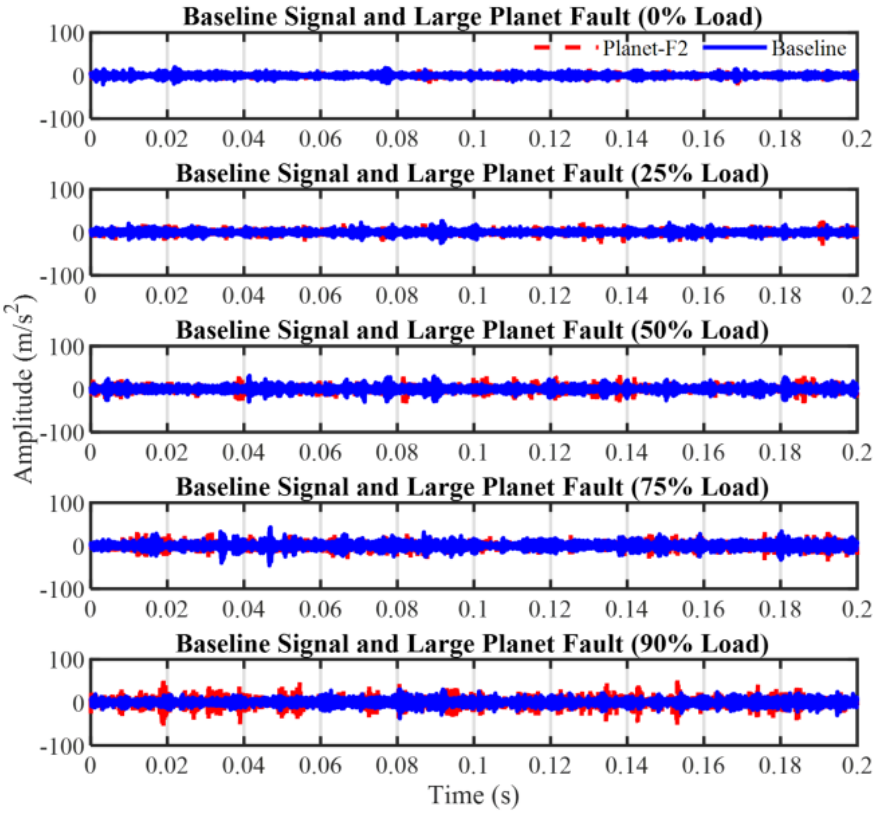

Figure 7-4: Time domain signal for baseline (blue) and Planet-F2 (red)

Based on analysing the vibration data using the time domain analysis for baseline, Sun-F1, Sun-F2, Planet-F1 and Planet-F2, it can be concluded that the time domain signal alone is not adequate to detect the defects occurred in the fault cases. This is because the raw vibration data will be contaminated by many rotating components and masked by the background noise of the entire system. Therefore, the fault cases and their severities cannot be detected from the raw time domain signal alone.

Statistical parameters such as RMS and kurtosis will be used to explore trends in the vibration data. Figure 7-5 and Figure *7-6* display the RMS and kurtosis values obtained for each case. In Figure 7-5 (a), it can be seen that the RMS value of vibration signal for the Sun-F1 fault has a higher amplitude than the baseline data, with slight further increase in amplitude in the case of Sun-F2. Figure 7-6 (a) shows the RMS value of the vibration signal for baseline and faulty planet gears. It can be seen that both Planet-F1 and Planet-F2 show a slight increase in the RMS amplitude compared to the baseline, but the RMS amplitude for Planet-F2 does not provide any clear difference from that for Planet-F1.

Figure 7-5 (b) and Figure *7-6* (b) show the results obtained from calculating the kurtosis value for each case, it can be observed that the kurtosis is unable to detect the presence of the faults for either sun or planet gears, or in the case of different severities.

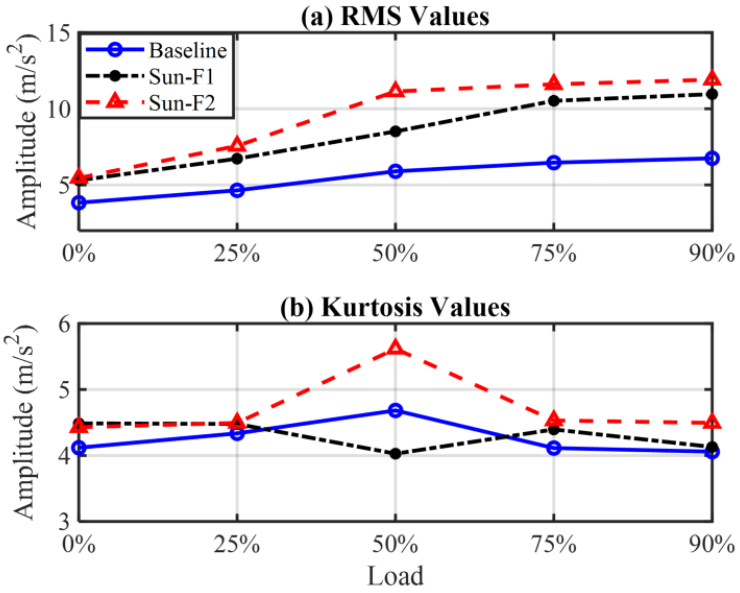

Figure 7-5: RMS and kurtosis for time domain (baseline, Sun-F1 and Sun F2)

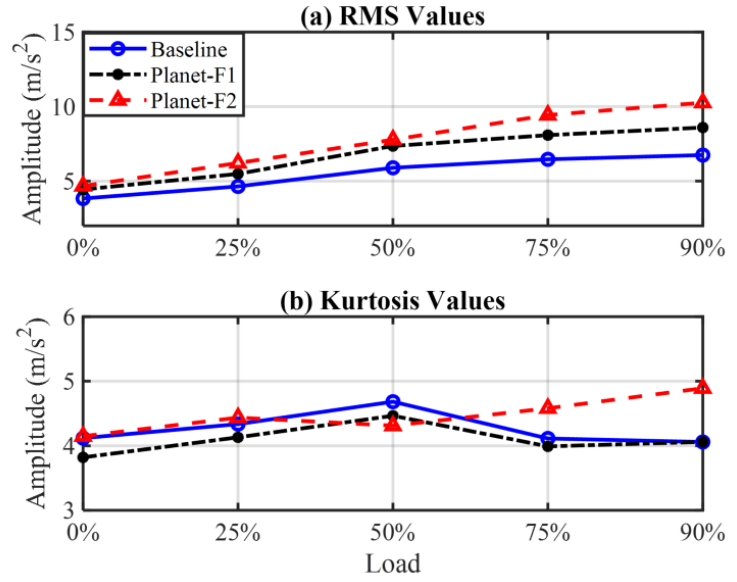

Figure 7-6: RMS and kurtosis for time domain (baseline, Planet-F1 and Planet F2)

7.2.2. Spectrum Analysis

Spectrum analysis begins by determining the characteristic frequencies of the PG, in terms of meshing frequency and the rotational frequencies associated with the sun, carrier, and planet gears. These characteristic frequencies are crucial in the detection of gear faults

and can be calculated from Equation 3.2 to Equation 3.8, as discussed in Chapter Three. The calculated characteristic frequencies of the PG used in this research are listed in Table 7-1.

Table 7-1: Characteristic frequencies of PG at 1117 rpm

f_{rsi}	Sun gear rotational frequency	18.60 Hz
f_{rci}	Carrier rotational frequency	2.58 Hz
f_{rpi}	Planet gear rotational frequency	3.58 Hz
f_{pmi}	Meshing frequency	160.22 Hz
f_{sf}	Sun gear fault frequency	48.12 Hz
f_{pf}	Planet gear fault frequency	12.34 Hz

Figure 7-7 to Figure *7-9* show the vibration spectrums for the baseline, Sun-F1 and Sun-F2 up to the third meshing frequency. From these figures, it can be seen that the amplitudes of the first three harmonics of the meshing frequency ($f_{pmi} = 160.2\,Hz$, $2f_{pmi} = 320.4\,Hz$ and $3f_{pmi} = 480.7\,Hz$) under zero load increase noticeably with increasing load. As shown in Figure 7-7 to Figure *7-9*, the signal amplitudes at zero load condition, are of relatively small amplitude, while at higher load condition the amplitudes of the peaks increase, which confirms that the load has an effect on the magnitude of the vibration signal obtained.

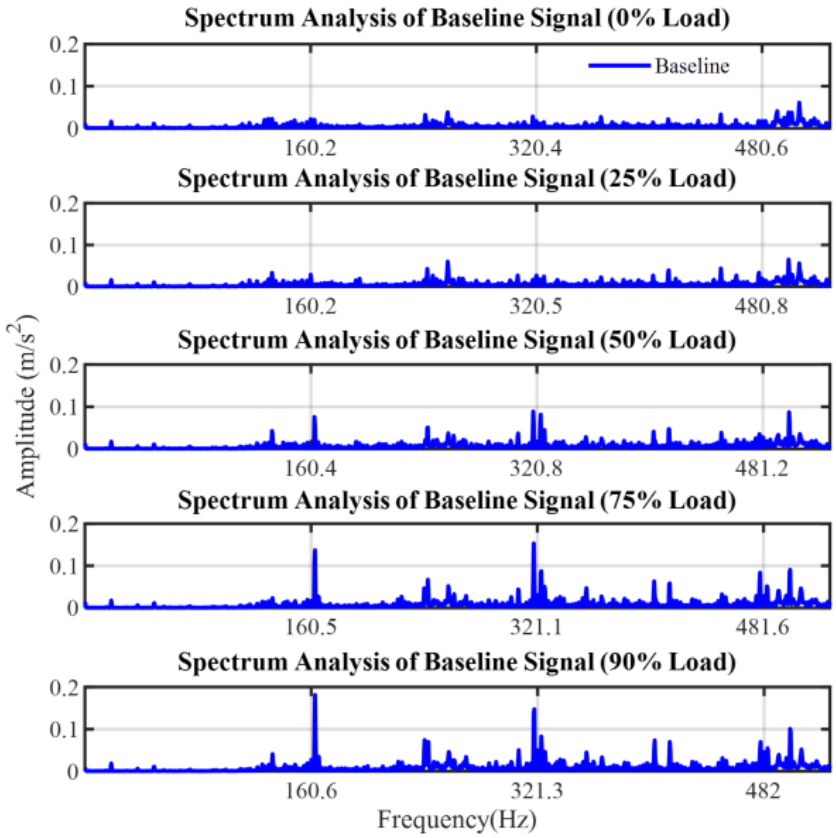

Figure 7-7: Spectrum analysis for baseline (blue)

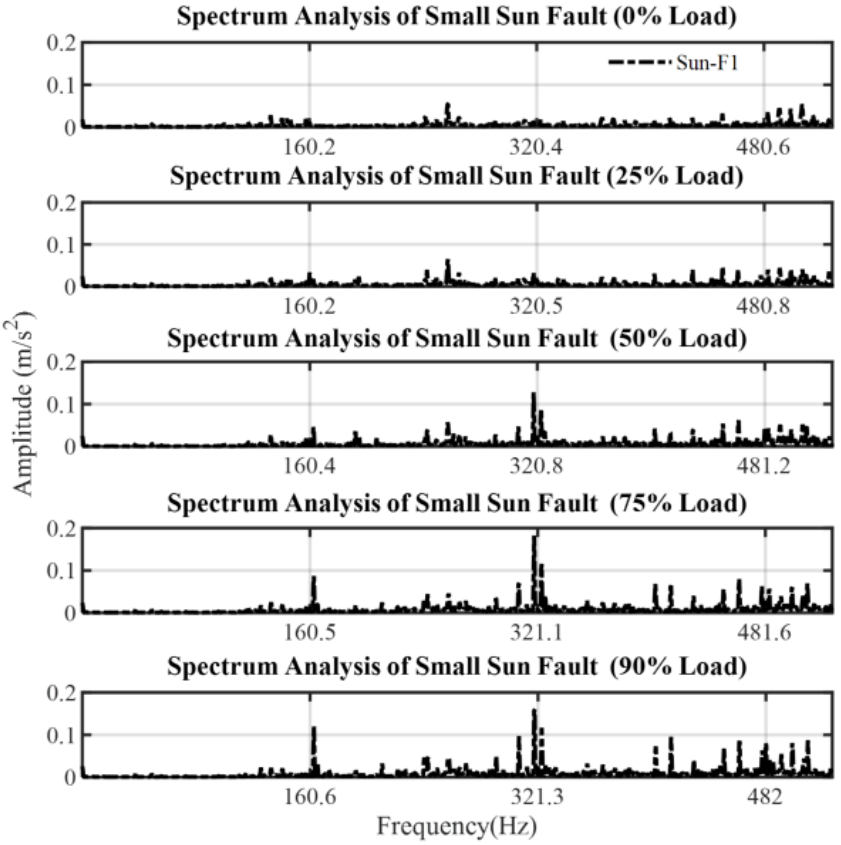

Figure 7-8: Spectrum analysis for Sun-F1 (black)

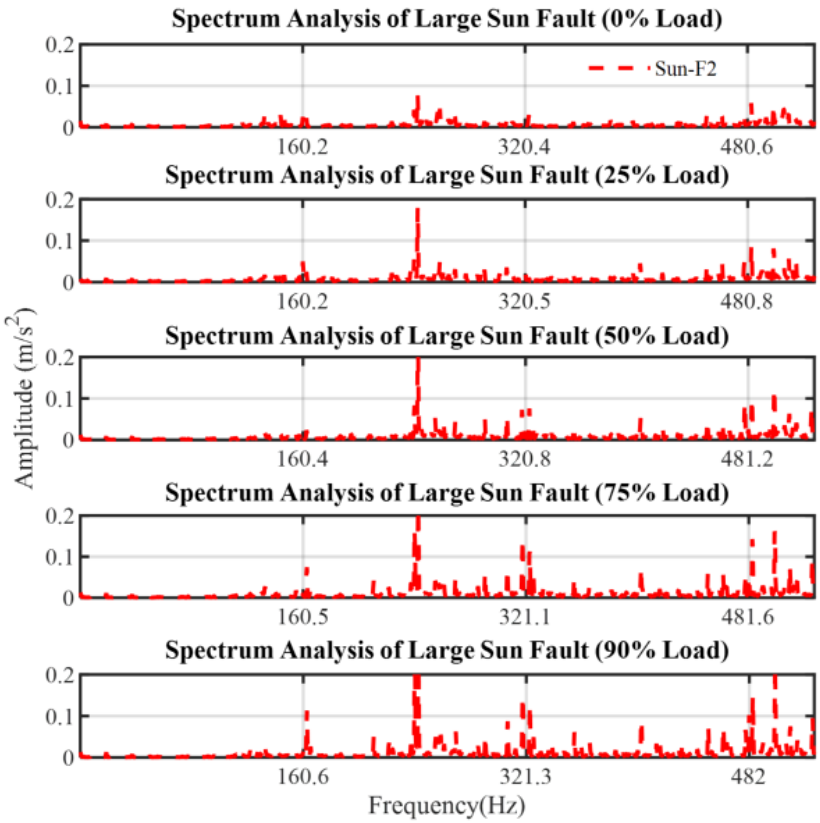

Figure 7-9: Spectrum analysis for Sun-F2 (red)

Figure 7-10 shows the spectrum of the vibration data for baseline (blue), Sun-F1 (black) and Sun-F2 (red), under 50% load. The amplitudes peaks produced by the faults can be used to diagnose the presence of the faults. As shown in Table 7-1, the sun gear fault frequency (f_{sf}) is around 48.12 Hz. However, it can be seen that the vibration spectrum does not show any dominant peaks at this frequency for either Sun-F1 or Sun-F2. However, Figure 7-10, presents a clear peak belonging to Sun-F2 close to 240 Hz, which happens to be between the first and second harmonics of the meshing frequency. This peak has the highest amplitude of any in the spectrum and relates to the fifth harmonics of the sun gear fault frequency (5 x 48.12 = 240.60 Hz). It has sidebands spaced at 2.58 Hz due to the carrier rotational frequency. For the Sun-F1, the spectrum does not show any significant increase in the amplitude compared with the baseline data and it was concluded that it is not possible to identify the presence of the Sun-F1 fault from this spectrum.

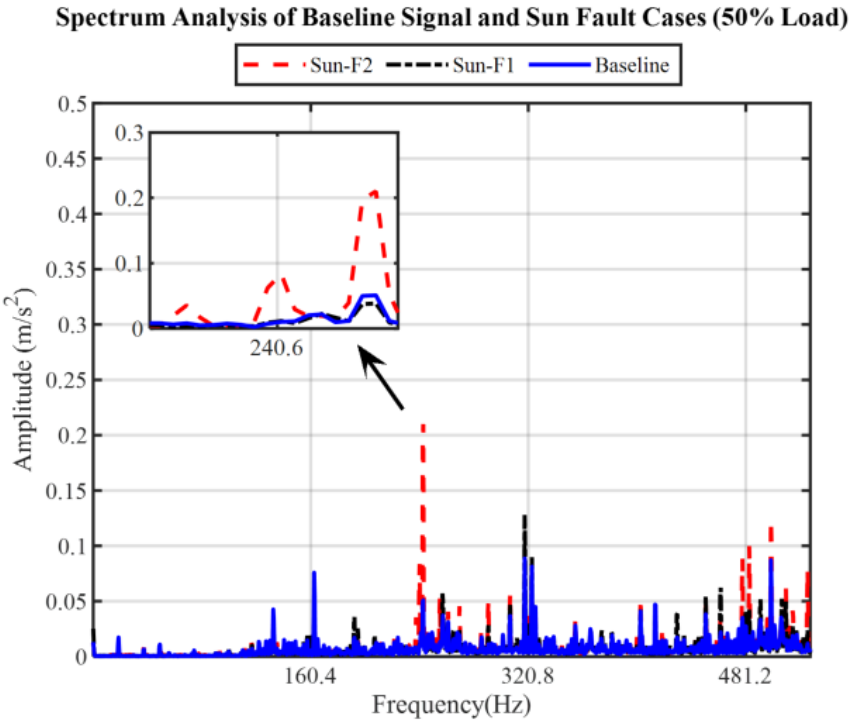

Figure 7-10: Spectrum analysis for baseline (blue), Sun-F1(black) and Sun-F2 (red) under 50% load

Figure 7-11 and Figure *7-12* show the vibration spectrums for Planet-F1 and Planet-F2 under zero, 25%, 50%, 75% and 90% of full load, up to the third meshing frequency.

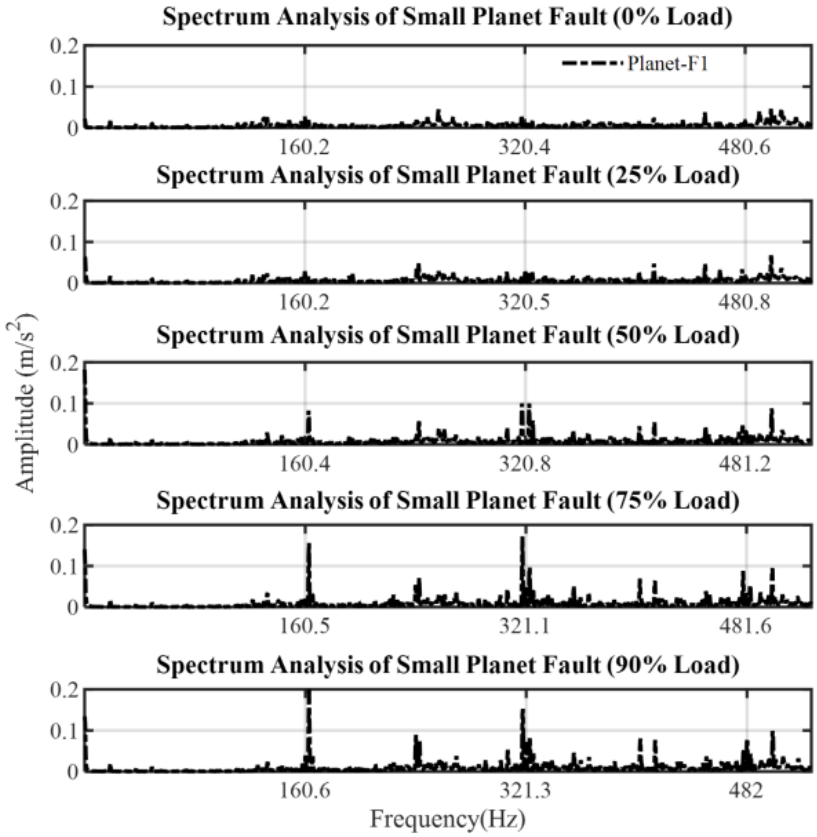

Figure 7-11: Spectrum analysis for Planet-F1 (black)

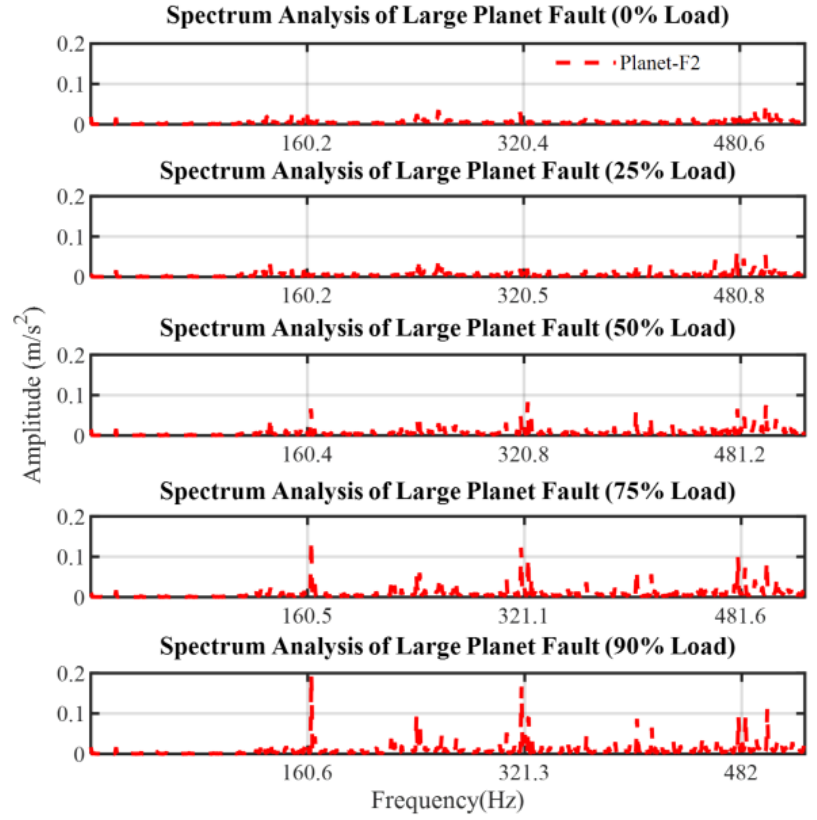

Figure 7-12: Spectrum analysis for Planet-F2 (red)

Figure 7-13 shows the spectrum of the vibration data for baseline (blue), Planet-F1 (black) and Planet-F2 (red) under 50% full load. The planet gear fault (f_{pf}) was calculated as 12.34 Hz, see Table 7-1. However, unlike the case of the sun gear, no clearly discernible peaks belonging to Planet-F1 and Planet-F2 can be observed to identify faulty planet gears even under the more severe fault condition.

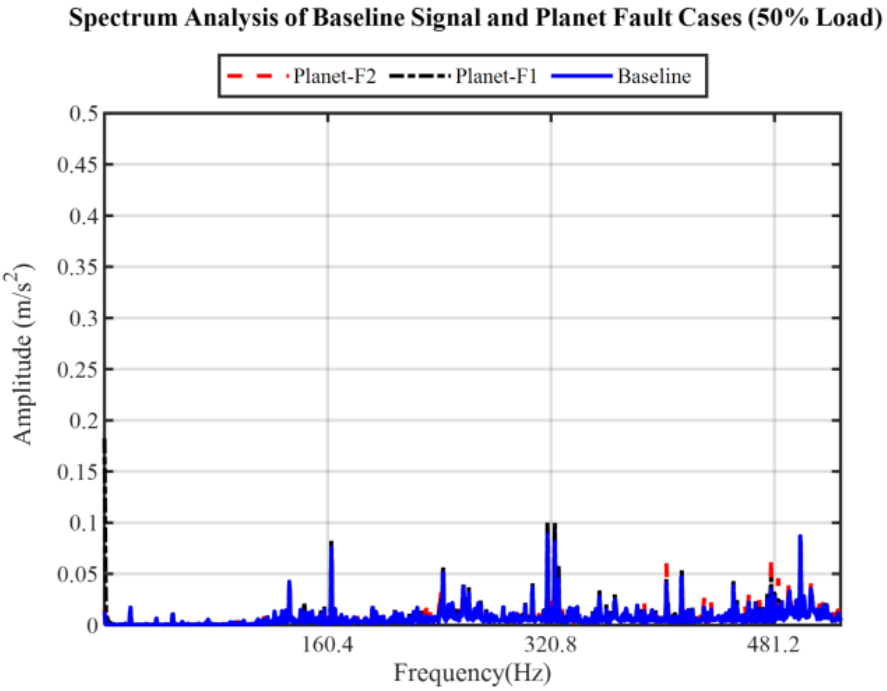

Figure 7-13: Spectrum analysis for baseline (blue), Planet-F1(black) and Planet-F2 (red) under 50% load

7.3. Experimental Results Obtained Using Multipoint Optimal Minimum Entropy Deconvolution Adjusted

This section presents the results obtained from applying MOMEDA to the collected vibration data for baseline, Sun-F1, Sun-F2, Planet-F1, and Planet-F2.

7.3.1. Time Domain Analysis

The application of the MOMEDA method to the collected vibration data begins by investigating the detection performance of the MOMEDA technique under different filter lengths from 600 to 1200. Figure 7-14 shows the MKurt values under different filter lengths for baseline, Sun-F1, and Sun-F2 under 90% load. The figure shows that under all filter lengths, the MKurt value for the baseline is smaller than the value for the Sun-

F1 and Sun-F2 faults. A filter length of L=1000 is selected for this experiment as a balance between a lower value and a higher value of MKurt. A filter length above 1000 needs a huge computer memory to calculate it.

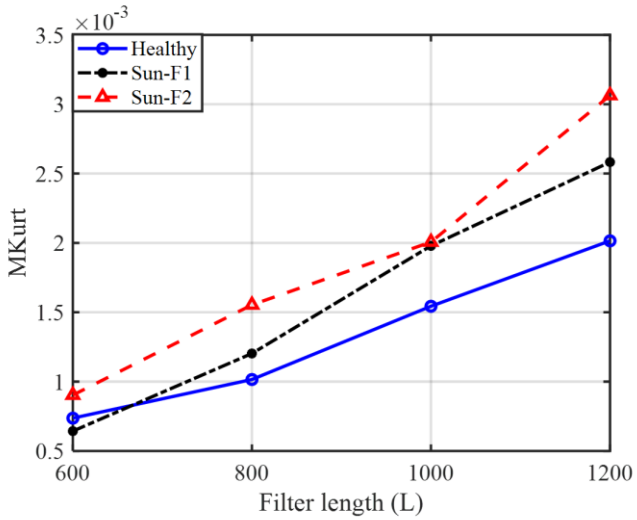

Figure 7-14: MKurt values for baseline, Sun-F1 and Sun-F2 with different filter lengths

In order to extract periodic fault features generated by the sun gear defect, the MOMEDA requires a fault location period that represents the fault features in data points. The fault features generated by the sun gear defect can be calculated as Fs/f_{sf}, where Fs is the sampling rate 100 kHz, and f_{sf} is the sun gear fault frequency 48.12 Hz (Fs/f_{sf}= 100000/48.12 = 2078), hence the period corresponding to this peak is used as the range of the sun gear fault feature.

Figure 7-15 toFigure *7-17* show the results obtained from the MOMEDA technique for baseline, Sun-F1, and Sun-F2 under five different load conditions (zero, 25%, 50%, 75% and 90% of full load). From the results obtained, it can be seen that the MOMEDA is able to extract periodic fault features in the signal generated by the faults (Sun-F1 and Sun-F2). However, neither Sun-F1 and Sun-F2 show any clear change in signal amplitude compared with the baseline. In addition, it can be seen there is an unexpected peak in the baseline data at multiples of about 2080 Hz, depending on load. This could be explained as due to a non-uniform force between gear components may cause a feature to be extracted, as stated in a previous study by McDonald and Zhao [82].

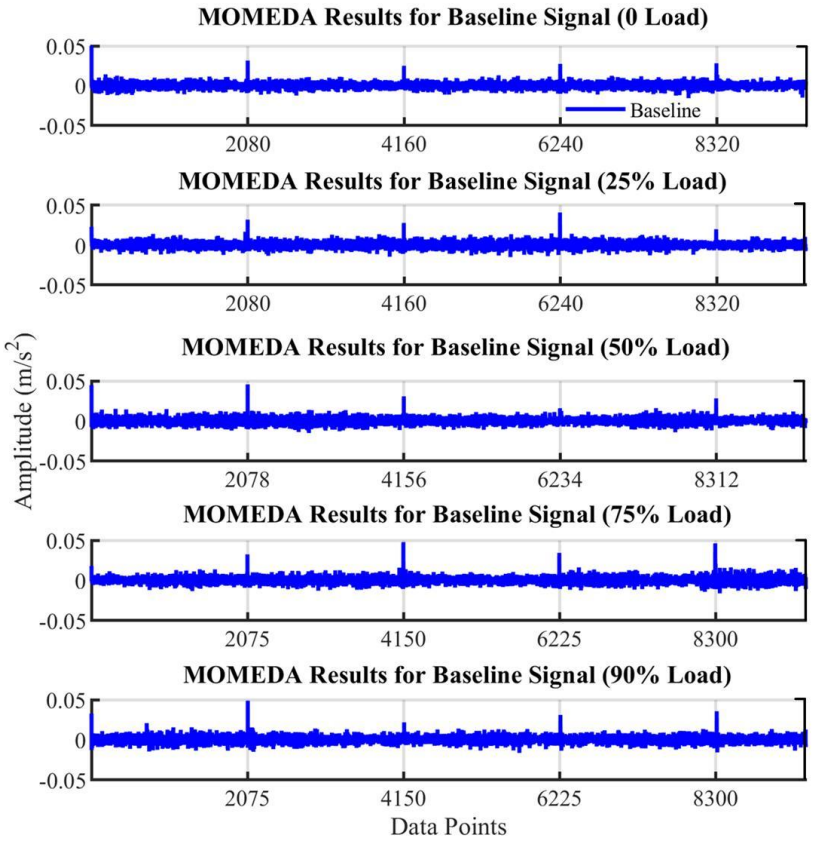

Figure 7-15: MOMEDA results for baseline signal

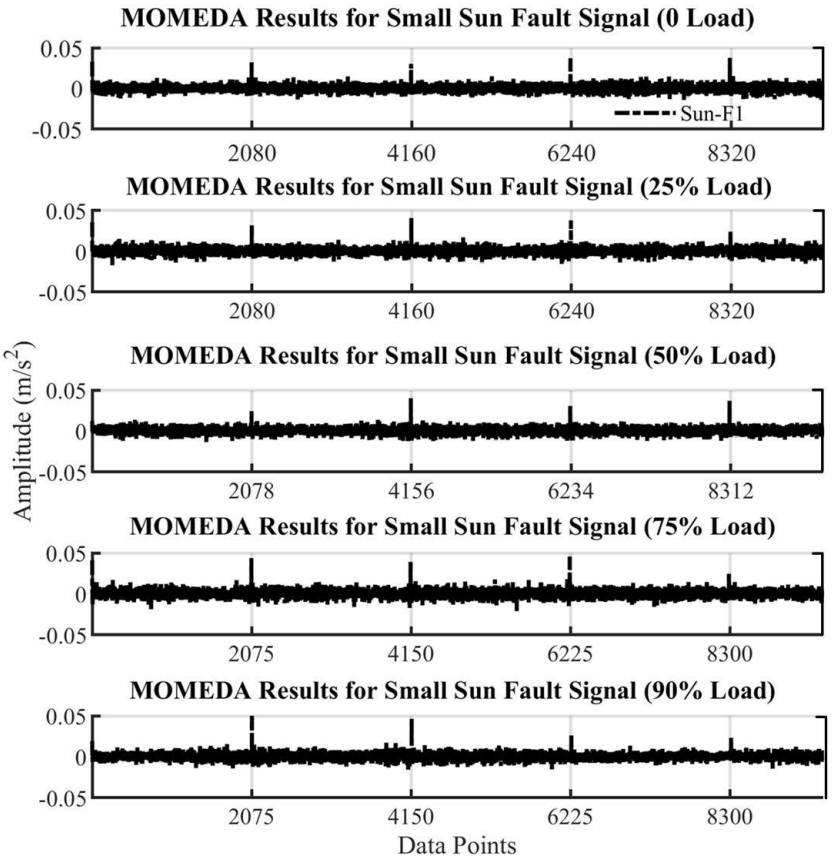

Figure 7-16: MOMEDA results for Sun-F1

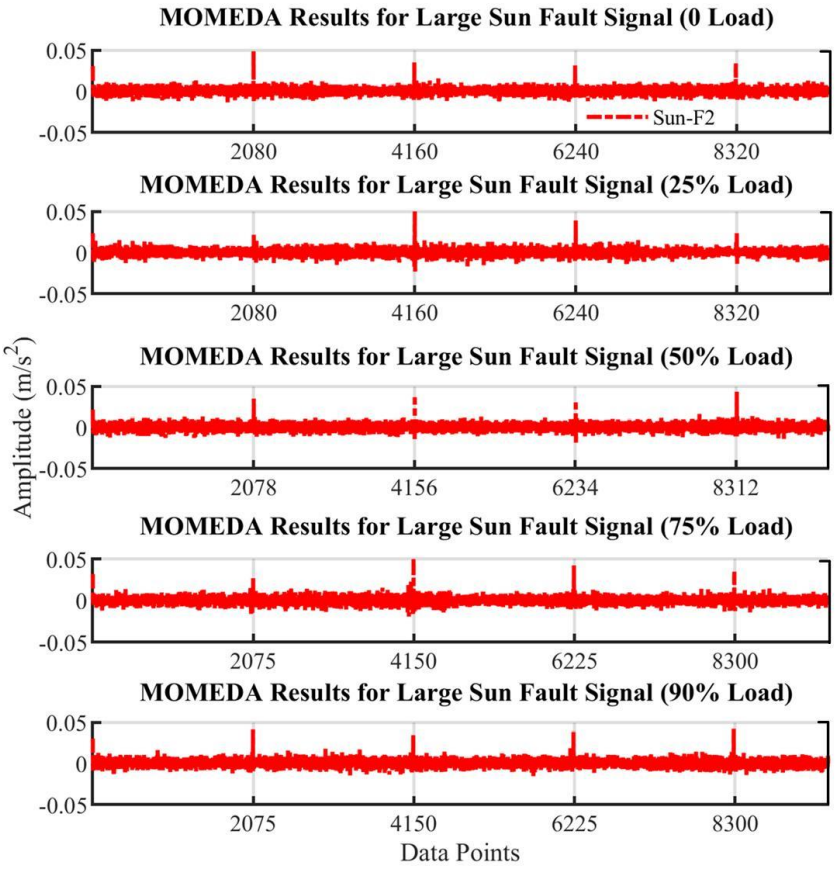

Figure 7-17: MOMEDA results for Sun-F2

Following the same procedure as adopted for the sun gear, Figure 7-18 shows the MKurt values under different filter lengths for baseline, Planet-F1, and Planet-F2. It can be seen that for all filter lengths, the MKurt value for both Planet-F1 and Planet-F2 are higher than the value of the baseline. A filter length of L=1000 is selected in this experiment.

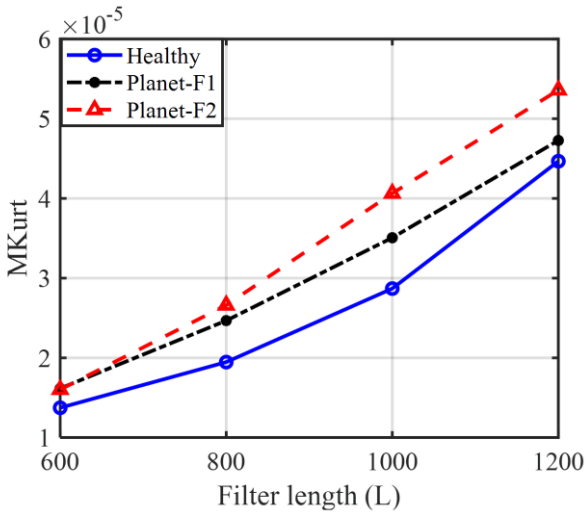

Figure 7-18: MKurt values for baseline, Planet-F1 and Planet-F2 with different filter lengths

As for the sun gear, when extracting periodic fault features generated by the planet gear defect, the fault feature can be calculated as Fs/f_{pf}, where f_{pf} is the planet gear fault frequency 12.3 Hz (Fs/f_{pf}= 100000/12.34 = 8103), and hence the period corresponding to this peak is used as the range for the planet gear fault feature.

The results obtained from the MOMEDA technique for baseline, Planet-F1 and Planet-F2 under five different loads are shown in Figure 7-19 toFigure *7-21*. It can be observed that MOMEDA is unable to extract periodic fault features generated by Planet-F1 or Planet-F2. This can be explained by the PG consisting of three planet gears that mesh simultaneously with both the sun and ring gears, which could affect the fault features generated by a single planet gear defect. Moreover, it can be observed that there are peaks that appeared in the baseline data, which were not expected, and which could lead to misdiagnosing the planet gear condition.

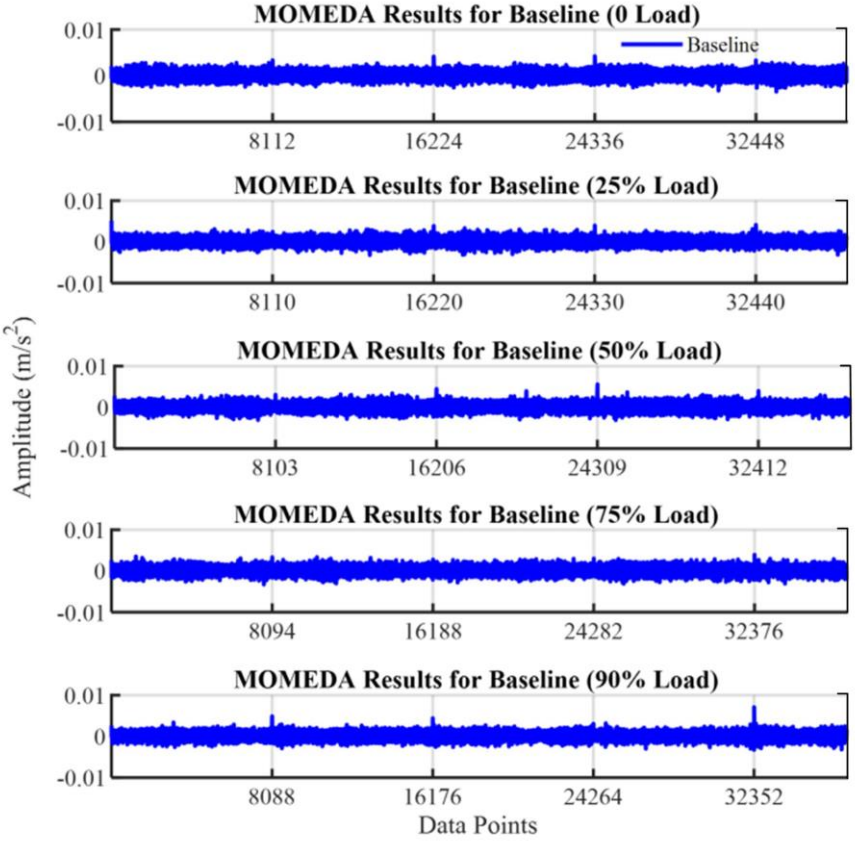

Figure 7-19: MOMEDA results for baseline signal

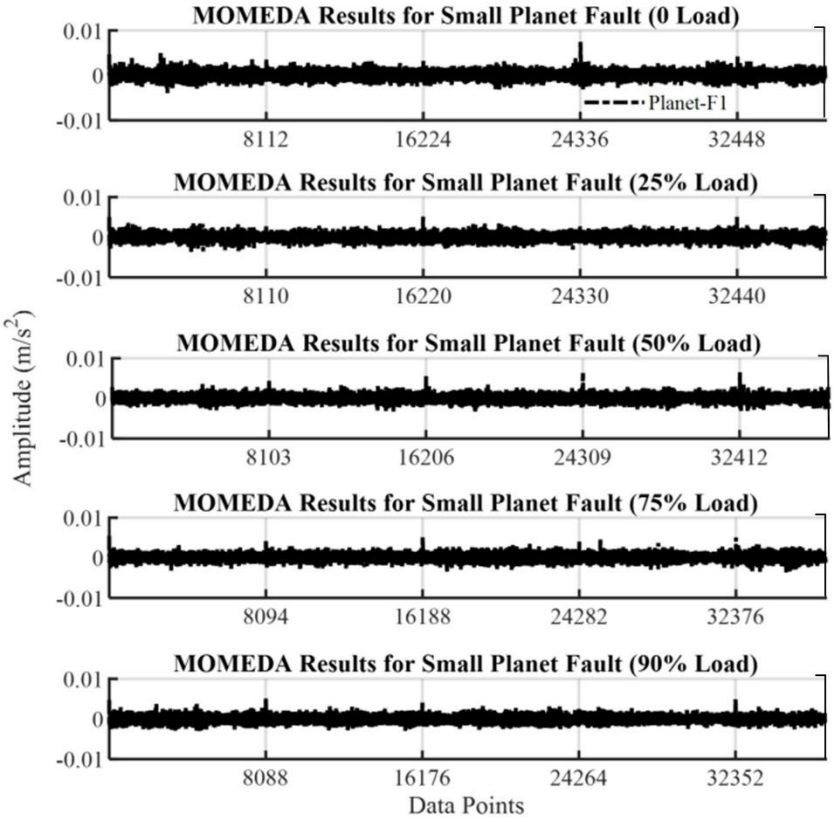

Figure 7-20: MOMEDA results for Planet-F1

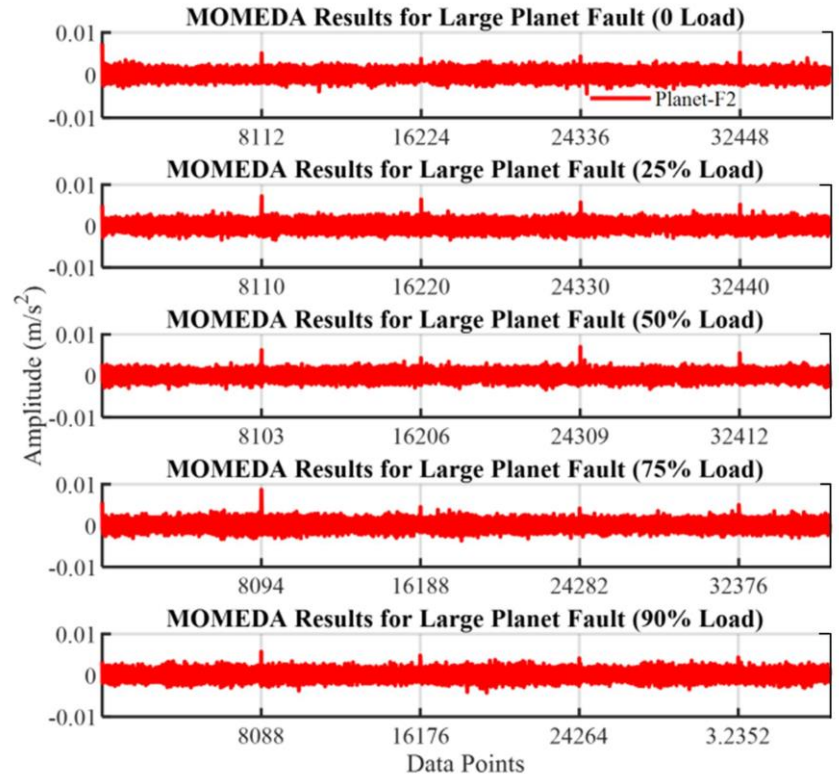

Figure 7-21: MOMEDA results for Planet-F2

From Figure 7-15 toFigure *7-21*, it can be concluded that the presence of the sun and planet gear faults and the level of their severities cannot be detected using MOMEDA. Therefore, a further investigation is carried out using spectrum analysis, see the next section.

7.3.2. Envelope Spectrum Analysis of the Filtered Data

For further investigation, the filtered signal obtained in the previous section was demodulated using envelope analysis. Figure 7-22 and Figure 7-23 show the envelope spectrum for baseline, Sun-F1 and Sun-F2 under five different load conditions. From Figure 7-22, it can be observed that the amplitude of the Sun-F1 spectrum at the sun gear fault frequency is slightly greater than the baseline. However, the diagnostic features cannot be readily detected from the envelope spectrum for the Sun-F1.

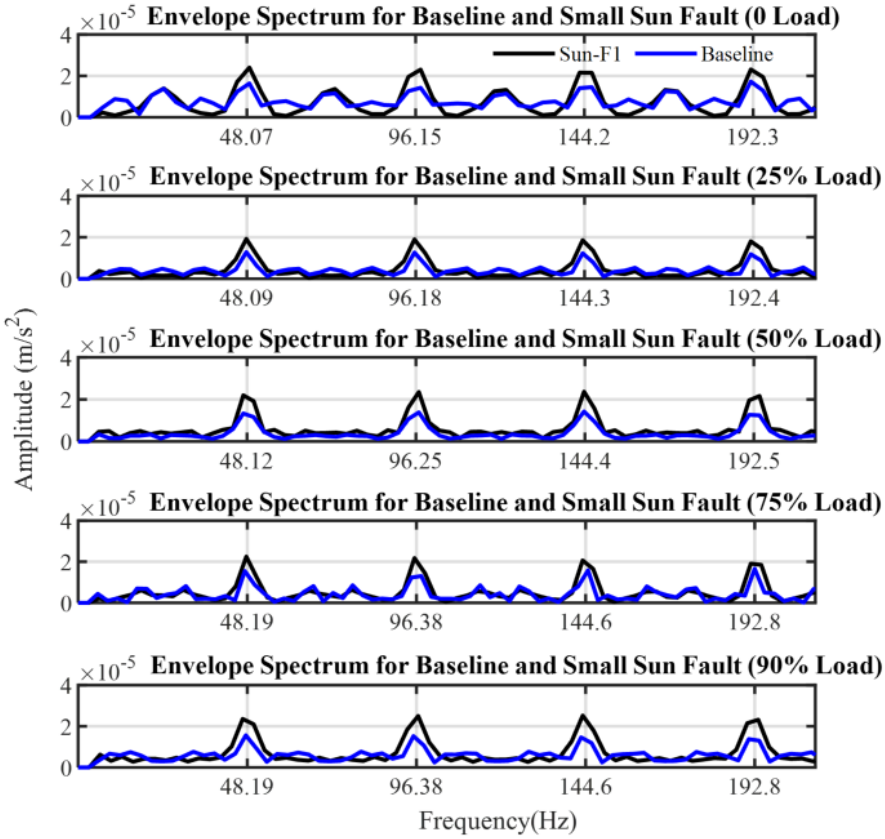

Figure 7-22: Envelope spectrum analysis for the baseline and Sun-F1

It can be seen in Figure 7-23 that the amplitude of the Sun-F2 spectrum is clearly higher than that of the baseline particularly at the sun gear fault frequency and its harmonics. From these results, it can be observed that the more severe fault (Sun-F2) generates a

higher amplitude signal than both Sun-F1 and the baseline, and the diagnostic features for the Sun-F2 are readily identified using the envelope spectrum.

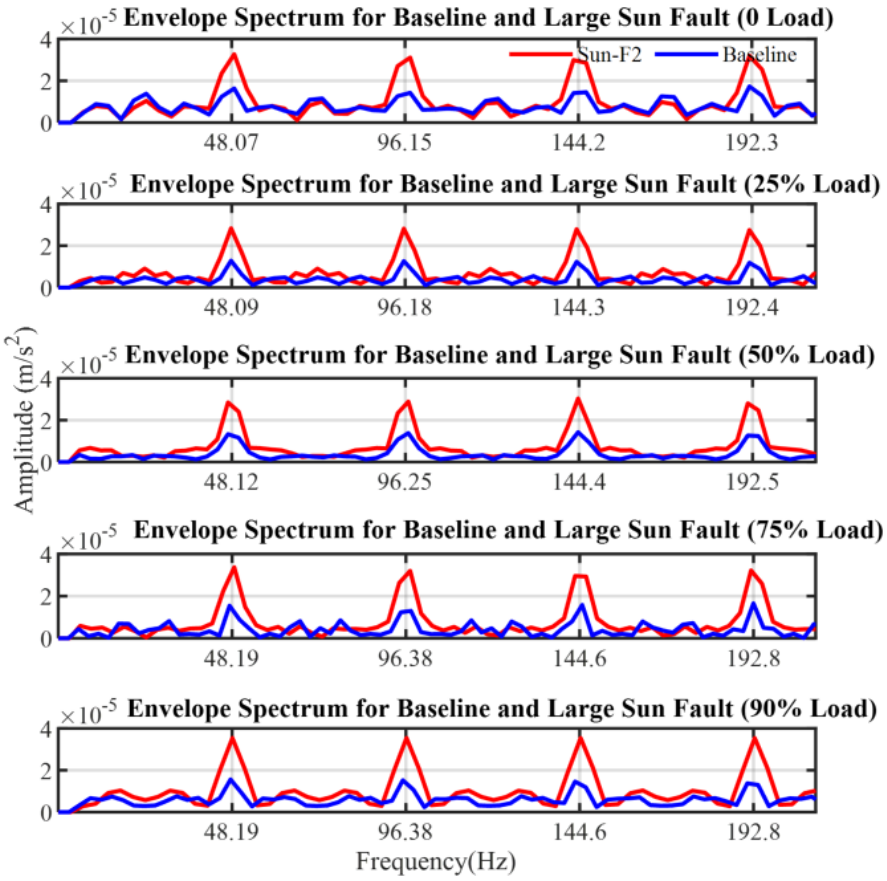

Figure 7-23: Envelope spectrum analysis for the baseline and Sun-F2

Figure 7-24 and Figure *7-25* show the envelope spectrum for baseline, Planet-F1 and Planet-F2. Similar to the Sun-F1 case, the results in Figure 7-24 shows that the amplitude of the Planet-F1 is slightly higher than the baseline at the planet gear fault frequency and its harmonics, especially at low loads (zero and 25% of full load). However, at medium loads (50% and 75% of full load), the amplitudes do not provide any clear differences. Therefore, the diagnostic features cannot be readily detected from the envelope spectrum for the Planet-F1.

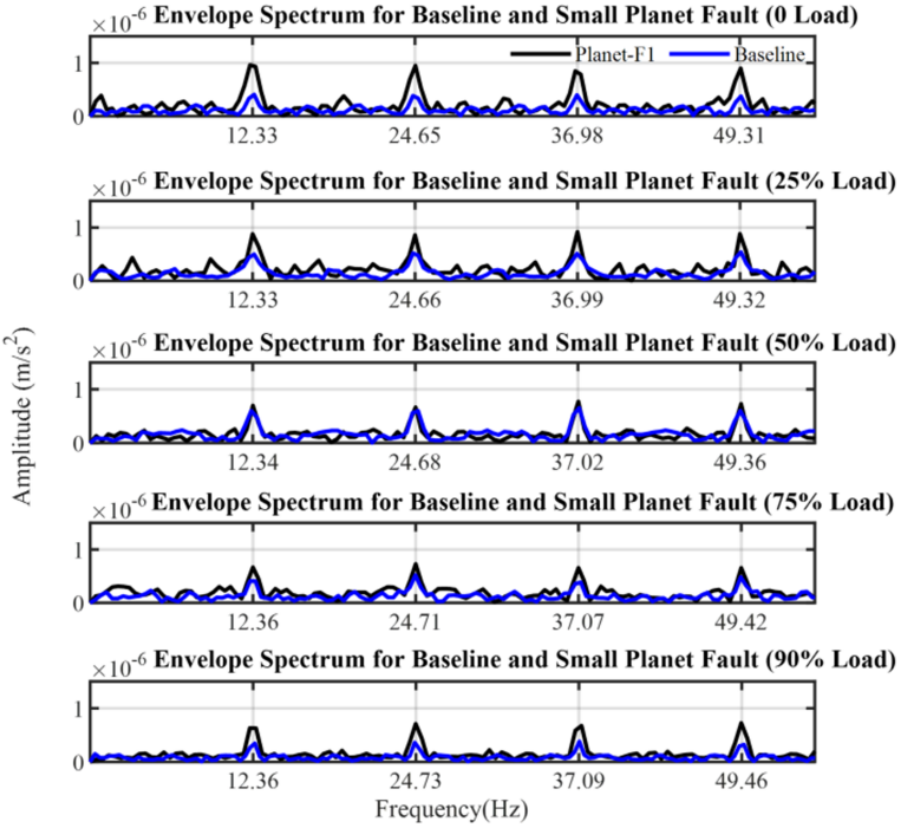

Figure 7-24: Envelope spectrum analysis for the baseline and Planet-F1

Figure 7-25 shows the envelope spectrum for baseline and Planet-F2 under five different load conditions. From the figure below, it is clear that the peak amplitude at the planet gear fault frequency and its harmonics are higher than that of the baseline, indicating the more severe fault (Planet-F2) generates a higher amplitude signal. Thus, the diagnostic features for the Planet-F2 can be observed and identified using the envelope spectrum.

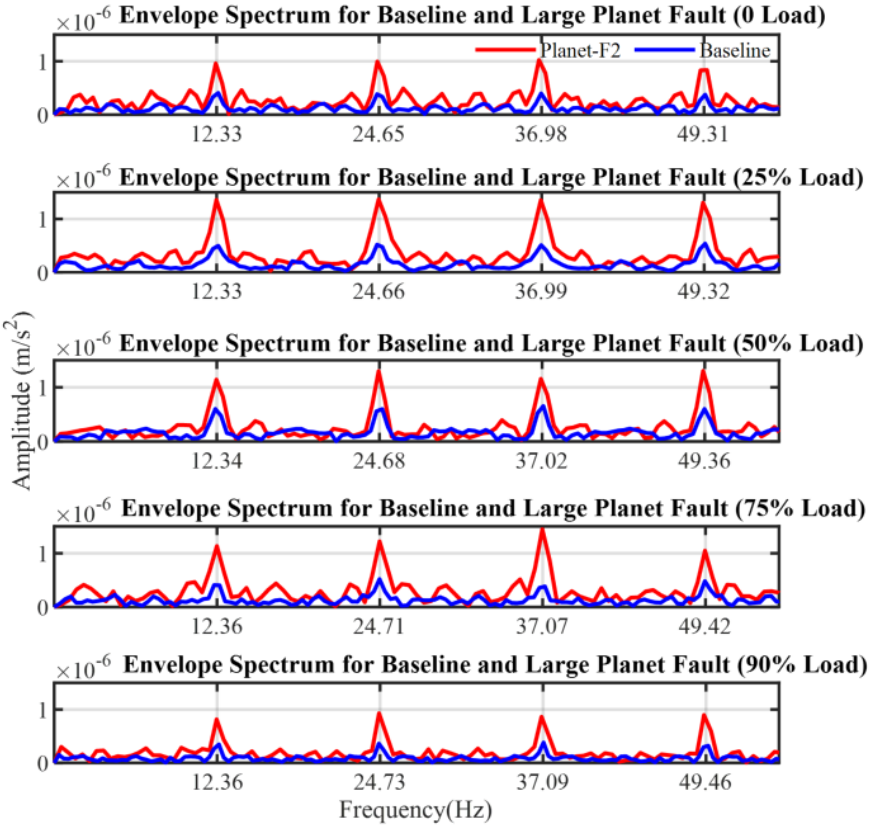

Figure 7-25: Envelope spectrum analysis for the baseline and Planet-F2

7.4. Results and Discussion

Conventional signal processing methods (time and frequency domains) have been widely used for fault detection and diagnosis. However, these methods are limited to diagnose and identify the defect that has occurred in a given case. As observed in Section 7.2.1, it can be said that the time domain analysis is not adequate to detect the defects occurred in the faulty cases. Section 7.2.2 showed that frequency domain analysis was unable to detect the peaks at 48.12 Hz and 12.3 Hz generated by the sun and planet gear faults, even under different fault severities.

From the results obtained using the MOMEDA method, it can be seen that the MOMEDA is able to extract periodic fault features generated by the defect for both Sun-F1 and Sun-F2 faults. While for Planet-F1 and Planet-F2, it has difficulties in extracting periodic fault features generated by the planet gear defect. Moreover, the results presented in Figure 7-15 toFigure *7-21* show that there is a presence of peaks in the baseline data, which were not expected and which could lead to misdiagnosing the gear condition.

The filtered signal obtained in Section 7.3.1 was then demodulated using envelope analysis. From the results presented in Figure 7-23 andFigure *7-25*, it can be seen that the amplitude of the peaks at the corresponding fault frequencies and their harmonics increase with the magnitude of the faults for both the sun and planet gears. Hence, it can be demonstrated that the larger amplitude peaks for the Sun-F2 and Planet-F2 are an indication of the presence of a fault. However, for Sun-F1 and Planet-F1, the diagnostic features cannot be readily detected using the envelope spectrum. Hence, in the next chapter, further investigation will be carried out using an automated approach based on a deep CNN method.

Chapter Eight: Automated Data Processing Using Convolutional Neural Network for Simulated Data

To evaluate the performance of the developed CNN-Three architecture with IReLU-Tanh function, this chapter begins by presenting the implementation steps of the CNN-Three model. It starts with the details of the CNN-Three architecture including architecture design, training and parameters tuning, validation, and testing. Then, it applies the CNN-Three architecture to simulated data to evaluate its performance in classifying different levels of fault. The effectiveness of the developed CNN-Three architecture is evaluated and compared with three commonly used CNN architectures. In addition, the performance of the proposed CNN-Three architecture with IReLU-Tanh function is evaluated and compared against the most widely used activation functions, Tanh, ReLU, LReLU, and ELU. Finally, the results obtained are discussed.

8.1. Introduction

This chapter presents an evaluation of the performance of the developed CNN-Three architecture with the proposed IReLU-Tanh function using simulated data. It starts with the details of the CNN-Three architecture, including architecture design, training and parameters tuning, validation, and testing. Each step has a different purpose in the development of the CNN-Three architecture. The effectiveness of the developed CNN-Three architecture will be evaluated and compared with three recent CNN architectures. In addition, the performance of the CNN-Three architecture with the proposed IReLU-Tanh function will be evaluated and compared against the most widely used activation functions, Tanh, ReLU, LReLU, and ELU. Finally, the results obtained are discussed at the end of this chapter.

8.2. CNN Architecture Creation

The developed CNN-Three architecture consisted of four main steps: architecture design, training and parameter tuning, validation, and testing. These four main steps are discussed in detail below.

8.2.1. CNN Architecture Design

This section focuses on designing a CNN architecture appropriate for feature extraction and classification. Figure 8-1 shows the developed CNN architecture and it consists of three CNN feature extraction groups, followed by the fully connected and softmax layers for performing multi-class classification. Each CNN feature extraction group contains four layers including a convolutional layer employed to extract the features directly from the data, a batch normalisation layer to reduce the internal covariate shift and significantly accelerate the training process of the CNN model, the proposed IReLU-Tanh function was used to introduce nonlinear characteristics into the model by transforming the input feature map of the previous layer to a non-linear output in the range of $[-1, \infty]$, and a max-pooling layer was applied at the end of each CNN feature extraction group to reduce the dimension of the output feature map whilst preserving the most important features.

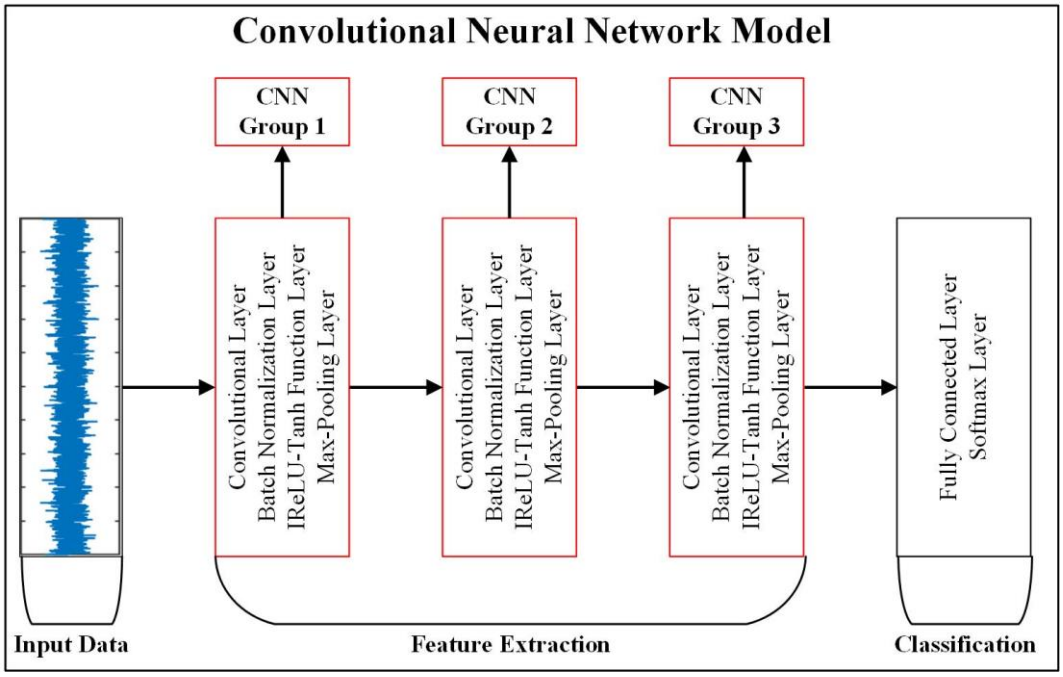

Figure 8-1: The developed CNN architecture

As discussed in Chapter Four, the CNN architecture generally includes several hyper-parameters that need to be tuned including, the number of CNN layers, number of convolutional filters, convolutional filter size, learning rate, mini-batch size, etc. Tuning these hyper-parameters will be discussed in detail in the next section.

8.2.2. Training and Parameter Tuning

As discussed in Chapter Four (see Section 4.5.1), for both simulated and experimental data, the training of the network starts by dividing the vibration data into N segments. Each data segment length was set to cover more than one period of the expected fault feature. Then, the entire set of data segments are divided randomly into three data groups: training, validation, and testing. Following previous studies [224-226], 60% of the data segments will be for training, 20% for validation, and 20% for testing. The implementation of the developed CNN architecture starts with model training using the labelled-data set so as to let the model learn representative features directly from the raw data. In this step, the CNN model tries to map the input training data to a unique target class.

The hyper-parameters of the developed CNN model were optimised using trial and error strategies to find the optimal parameters for the network. Three sets of signals with three

different fault severities were generated and used during the optimisation process. In this study, the simulated signal model was adopted from a previous study by Feng and Zuo [253], which modelled gear vibrations including two common frequencies: gear meshing frequency and gear fault frequency. Additive random noise was included in the signal, see Equation 8.1:

$$x(t) = A(t)\,cos(2\pi f_{mesh}\,t) + e(t) \tag{8.1}$$

and $A(t)$ can be written as:

$$A(t) = A\,cos(2\pi f_p\,t) \tag{8.2}$$

where A is the amplitude modulation, f_{mesh} is the meshing frequency, f_p is the fault frequency, t is the time, and $e(t)$ is the added noise.

The network is trained with the three sets of simulated data with three different fault severities (small, medium, large) using Equation (8.1), and a small amount of noise with SNR of (4 dB) was added to each simulated signal, as shown in Figure 8-2. To simulate a signal similar to the experimental vibration data, the following parameters were adopted; gear meshing frequency f_{mesh} =160 Hz, gear fault frequency f_p= 50 Hz, sampling rate F_s = 100 kHz, and time t set to 15 seconds.

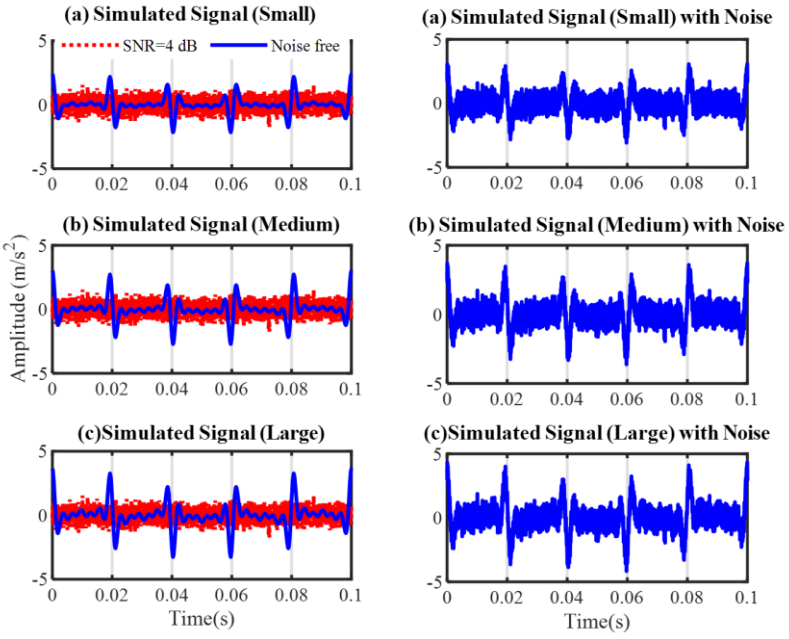

Figure 8-2: Training data - simulated signals with SNR=4 dB

After training the network, the trained model was tested using the same simulated signal but with different levels of SNR ranging from -1 dB to -5 dB, as seen in Figure 8-3. The different levels of noise were used to evaluate the effect of noise strength on the performance of the CNN architecture.

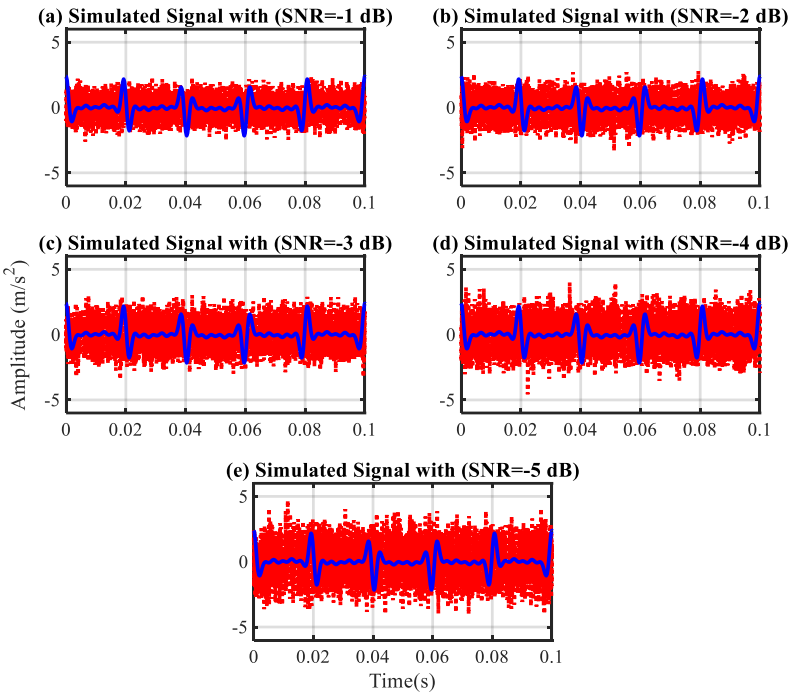

Figure 8-3: Testing data - simulated signals with different levels of SNR ranging from (-1 dB) to (-5 dB)

After testing the CNN architecture with the different SNR levels, several parameters including, number of CNN layers, number of convolutional filters, convolutional filter size, etc., were tested using Equation (8.1) and the parameters giving the best classification performance were selected, see Table 8-1.

Table 8-1: Optimal network parameters of the developed CNN-Three architecture

No.	Layer Type	Filter Size	Number of Filters	Stride
1	Convolution	$128 * 1$	16	$6 * 1$
2	Max-Pooling	$2 * 1$	16	$2 * 1$
3	Convolution	$4 * 1$	24	$1 * 1$
4	Max-Pooling	$2 * 1$	24	$2 * 1$
5	Convolution	$4 * 1$	32	$1 * 1$
6	Max-Pooling	$2 * 1$	32	$2 * 1$
Training Parameters			**Value**	
Mini-batch size			90	
Learning rate			0.04	
Epoch			50	

The process of selecting the optimal parameters is discussed in the following subsections:

8.2.2.1. Number of CNN Groups

The number of CNN groups determines the depth of the network, i.e., the deep structure of the network refers to the number of hidden layers present in the architecture, and these can range from tens to thousands. It has been claimed that the network depth is a high priority for improving the classification performance of the network [254]. Generally, the deeper the network, the better its ability to learn representative features from the raw data. However, while the depth of the network is important, the classification accuracy becomes degraded if the number of layers (depth) is excessive [255]. Hence, adding more layers to the network means a large number of model parameters to train, which makes the deeper networks are more prone to overfitting [203]. Thus, it is important to evaluate the effect of network depth on the performance of the CNN architecture.

In this study, three configurations of CNN architecture were investigated: CNN with two feature extraction groups (CNN-Two), CNN with three feature extraction groups (CNN-Three), and CNN with four feature extraction groups (CNN-Four). Each feature extraction group consists of four layers: convolutional, batch normalisation, activation

function layer and pooling. Table 8-2 shows the parameters of the CNN architecture using various configurations (feature extraction groups). Here, FS1, FS2, FS3 and FS4 denote the convolutional filter size and NF1, NF2, NF3, and NF4 refer to the number of convolutional filters, in the first, second, third and fourth convolutional layers of the feature extraction group.

Table 8-2: Parameters of the CNN architecture using various configurations (feature extraction groups)

Model	**FS1**	**NF1**	**FS2**	**NF2**	**FS3**	**NF3**	**FS4**	**NF4**
CNN-Two	128x1	16	4x1	24	-	-	-	-
CNN-Three	**128x1**	**16**	**4x1**	**24**	**4x1**	**32**	**-**	**-**
CNN-Four	128x1	16	4x1	24	4x1	32	4x1	32

Ten trials were carried out to ensure the reliability of the result. The average classification accuracies obtained using various configurations with different SNR levels are shown in Table 8-3.

Table 8-3: Comparison of average classification accuracies obtained using various configurations of CNN feature extraction groups with SNR levels ranging from (–1 dB) to (–5 dB)

Model	**SNR**				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
CNN-Two	99.80%	98.97%	97.66%	96.16%	93.47%
CNN-Three	**99.91%**	**99.25%**	**98.11%**	**96.72%**	**94.22%**
CNN-Four	99.91%	99.22%	97.94%	96.36%	94.11%

Figure 8-4 shows in graphical form the classification accuracies presented in Table 8-3. The classification accuracy obtained is shown using three different bar colours: orange for CNN-Two, blue for CNN-Three, and yellow for CNN-Four.

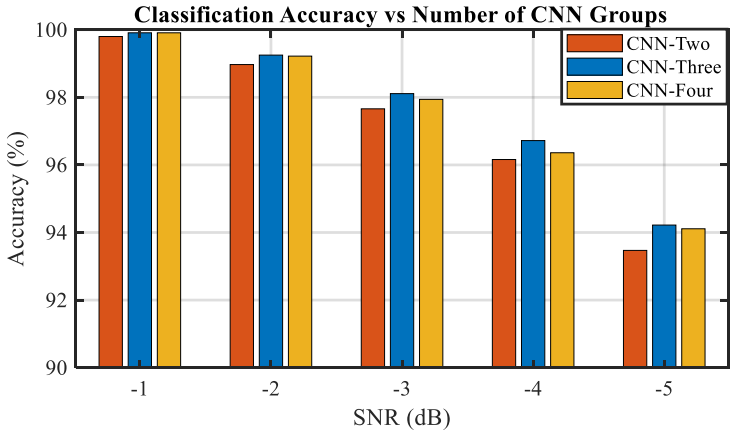

Figure 8-4: Classification accuracy using various configurations of CNN feature extraction groups

As shown in Table 8-3 and Figure 8-4, CNN-Three achieved the highest classification accuracy compared to CNN-Two and CNN-Four. Whereas, the CNN-Four is lower than CNN-Three for all SNR levels confirming that when adding more hidden layers into the network, it indicates a large number of model parameters to train, and it can lead to overfitting problem, resulting in a decrease in the classification accuracy of the model.

However, a lower classification accuracy was obtained when using CNN model with two feature extraction groups, showing that the CNN-Two is not deep enough and too simple, to learn the representative features effectively from the raw data, which resulted in lower classification accuracy of the model on the testing data.

Based on the classification accuracy obtained in Table 8-3 and Figure 8-4,, it can be said that CNN-Three achieved the best classification accuracy for the three sets of simulated data with SNR levels ranging from (-1 dB) to (-5 dB). CNN-Three was considered the optimal structure for developing the CNN architecture.

8.2.2.2. Convolutional Filter Size

Convolutional filter size refers to the size of the receptive field that is convolved with the input data. The size of the convolutional filter plays a vital role in extracting the representative features from the input raw data. It has been reported that a large convolutional filter has a larger receptive field than a small filter, and hence more information can be obtained with a wide filter, while small filters are expected to extract more detailed features [198, 203, 204, 211, 256].

In recent years, a number of studies have used a large receptive field in the first convolutional filter and then employed smaller receptive fields in later layers, which makes the network deeper to extract more detailed features from the early layer. Some recent work, for instance, Zhang, Peng [198] employed a large receptive field in the first convolutional layer with filter size of (64x1) and small receptive field in a later layer with size of (3x1). Jiao, Zhao [204] used a wide filter size of (51x1) in the first convolutional layer and a narrow filter size of (5x1) in the second convolutional layer. Sadoughi, Downey [257] employed a large filter size of (48x1) in the first layer and then a small filter size in the later layers.

Based on recent studies, it is important to evaluate the effect of using a large receptive field in the first convolutional layer on the performance of the developed CNN architecture. In this study, several experiments were conducted, setting different receptive field sizes in the first convolutional layer varying from 32 to 160 using the optimal CNN-Three architecture. Small receptive fields with filter size of (4x1) were used in the second and third CNN groups to make the network deeper and acquire better feature representation. Average classification accuracies obtained using various configurations of convolutional filter size are shown in Table 8-4.

Table 8-4: Comparison of classification accuracy obtained using different convolutional filter sizes with SNR levels ranging from (-1 dB) to (-5 dB)

Filter Size	**SNR**				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
32	98.94%	97.16%	94.88%	92.69%	89.22%
48	99.25%	98.16%	95.58%	93.91%	91.50%
64	99.44%	98.19%	96.55%	95.22%	92.36%
80	99.52%	98.69%	96.69%	95.33%	92.13%
96	99.50%	98.72%	96.77%	95.97%	93.33%
112	99.66%	98.86%	97.16%	95.88%	93.83%
128	**99.91%**	**99.25%**	**98.11%**	**96.72%**	**94.22%**
144	99.88%	99.19%	97.91%	96.66%	94.05%
160	99.91%	99.13%	98.02%	96.83%	94.11%

In Table 8-4, it can be observed that the average classification accuracy increases as the filter size in the first convolutional layer increases, especially at SNR= -5 dB, e.g., the average classification accuracy was only 89.22% when the filter size was set to 32, and it reaches 94.22% when the filter size increased to 128. It can also be seen that, the best classification accuracy occurs at filter sizes 128, 144, and 160, which confirms that a larger filter size can obtain better fault information features with a longer convolutional filter. Based on the average classification accuracy obtained in Table 8-4, a filter size of (128x1) was selected as optimal filter size for the first convolutional layer for the CNN-Three architecture, as it achieved the highest classification accuracy compared to other filter sizes with an accuracy value of 94.22%.

8.2.2.3. Number of Convolutional Filters

The number of convolutional filters plays an important role in the feature extraction step of deep architecture models. It refers to the number of features to be learned during the

training process. It has been reported that the number of convolutional filters has great impact on the training efficiency and the overall performance of the model [258], the more convolutional filters used, the more features are extracted in each convolutional layer. However, using too many convolutional filters in the architecture means increasing the number of model parameters, which results in increasing the computational complexity [259]. On the other hand, using too few convolutional filters means the feature extraction will be insufficient and unable to extract the representative features from the raw data, leading to a decrease in the classification accuracy of the model [123]. Thus, it is necessary to evaluate the effect of the number of convolutional filters on the performance of the CNN architecture.

Recently, a number of studies have employed different numbers of convolutional filters, for instance, Jian, Li [211] employed three different convolutional filters (16/32/64). Sadoughi, Downey [257] used 8, 16, 32 and 32 convolutional filters for the first, second, third, and fourth layers. Zhang, Li [256] used four convolutional layers with different number of filters 8, 16, 32 and 64. Chen, Hu [203] applied six different convolutional filters, 16, 32, 64, 64, 64, and 64. Based on these studies, four types of convolutional filter configurations were investigated using the optimal CNN-Three architecture, see Table 8-5.

Table 8-5: Parameters of the CNN-Three architecture using various configurations of filter size (FS) and number of filters (NF)

No. Filters	FS1	NF1	FS2	NF2	FS3	NF3
8/16/24	128x1	8	4x1	16	4x1	24
8/16/32	128x1	8	4x1	16	4x1	32
16/24/32	**128x1**	**16**	**4x1**	**24**	**4x1**	**32**
16/32/64	128x1	16	4x1	32	4x1	64

The average classification accuracy obtained using four types of convolutional filter configurations with five SNR levels are shown in Table 8-6.

Table 8-6: Comparison of classification accuracy using four different configurations of convolutional filters with five SNR levels

No. Filters	**SNR**				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
8/16/24	99.88%	99.16%	97.75%	96.13%	93.44%
8/16/32	99.75%	99.22%	97.41%	96.55%	94.02%
16/24/32	**99.91%**	**99.25%**	**98.11%**	**96.72%**	**94.22%**
16/32/64	99.83%	99.02%	98.02%	96.25%	93.44%

Figure 8-5 shows the average classification accuracy using different numbers of convolutional filters under five different noise levels, the green bar is for the 8/16/24, the orange for 8/16/32, the blue bar for 16/24/32, and the yellow bar for 16/32/64.

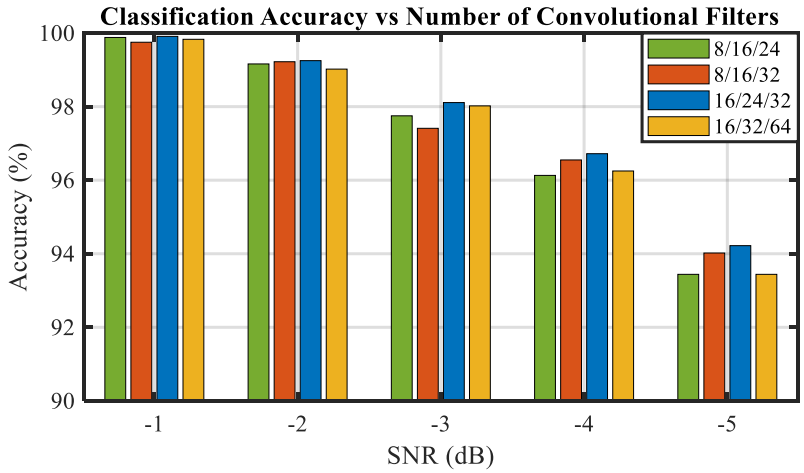

Figure 8-5: Classification accuracy using four different number of convolutional filters

In Figure 8-5 it can be observed that the CNN-Three with 16, 24 and 32 convolutional filters (blue bar) achieved the highest classification accuracy for all five SNR levels. As demonstrated in Figure 8-5, it can be confirmed that using too few convolutional filters (e.g., 8, 16 and 24), the feature extraction will be insufficient or unable to extract the representative hidden features from the raw data, leading to decreases the classification accuracy of the model. On the other hand, increasing the number of convolutional filters for the first, second and third convolutional layers, increases average classification accuracy, but only up to a point. Too many convolution filters (e.g., 16, 32 and 64)

reduced the classification accuracy, especially at SNR= -5 dB. This decrease in the classification accuracy is indicative of the architecture containing too many model parameters to train, leading to an overfitting problem.

Based on the classification accuracy presented in Table 8-6, it can be seen that the CNN-Three architecture with 16, 24 and 32 convolutional filters achieved the best classification accuracy when classifying the three sets of simulated data. Therefore, CNN-Three with 16, 24 and 32 convolutional filters were selected as optimal parameters for the CNN-Three architecture.

8.2.2.4. Stride

The number of strides determined the step size of the convolutional filter as it is slid over the input data. For example, if the number of the stride was set to 1, then the convolutional filter moved one step at a time over the input data. As discussed in Section 8.2.2.2, the earlier convolutional layer is generally responsible for extracting representative features from the raw data, whereas the later layer is to acquire better feature representation. Thus, it is helpful to use a larger stride in the earlier convolutional layer to speed up the calculations, reduce the amount of overlapping and produce an output feature map with lower dimension, then use small strides in the later convolutional layers for extracting more detailed features. In this study, a large convolutional stride in the earlier layer was set to 6x1, and then a small stride of 1x1 was used in the second and third convolutional layers.

8.2.2.5. Max-Pooling

Max-pooling is employed to reduce the dimension of the output feature map, while preserving the most important features. The max-pooling operation is used to extract the maximum value for each non-overlapping pooling size (when the stride is equal to pooling size), see Section 4.3.4. In most CNN architectures, the max-pooling layer is commonly applied after the convolutional layer with a pooling window of size 2x1 and stride of 2x1. In this study, the max-pooling layer was employed at the end of each CNN feature extraction group, as seen in Figure 8-1 and Table 8-1.

8.2.2.6. Mini-Batch Size

Mini-batch size is one of the parameters that needs to be tuned. It represents the number of input training data in a batch to be fed into the network to make one update for the model parameters. As discussed in Section 4.4.3, the mini-batch GD splits the training data into N mini-batches, each mini-batch is fed into the network to compute the gradient of the loss function and then updates the model parameters. Passing the entire N mini-batch of the training data into the network corresponds to one complete cycle of the training epoch. It has been reported that the number of input training data per mini-batch has a significant impact on the training performance [128]. Selecting a small value will result in each mini-batch comprising of few training data, leading to faster convergence than a large mini-batch, but a large mini-batch can reach a local optimum with minimum error. Thus, it is important to evaluate the effect of the input training data per mini-batch on the performance of the CNN architecture.

As discussed in Section 8.2.2, training of the network starts by segmenting the input data into N segments. For the generated simulated signal in Section 8.2.2, a sampling rate of $Fs =$ 100 kHz, and fault frequency f_p = 50 Hz, then each data segment length was set to 2500 (Fs/f_p=100000/50=2000) to cover more than one period of the expected fault feature. For 15 seconds, a total number of 600 data segments were obtained for each simulated data (small, medium and large). In this way 1800 data segments were obtained for the three levels of fault severity. In this study, 60% of the 600 data segments for each fault severity (360 data segments) were selected randomly as the training data, 20% (120 data segments) were selected randomly as the validation data, and the remaining 20% (120 data segments) were used for testing data. Table 8-7 shows the data segmentation used in this study.

Table 8-7: Details of data segmentations used for simulated data

Severity	Total Data Segments	Dataset	Percentage	Data Segments
Small	600	Training	60%	360
		Validation	20%	120
		Testing	20%	120
Medium	600	Training	60%	360
		Validation	20%	120
		Testing	20%	120
Large	600	Training	60%	360
		Validation	20%	120
		Testing	20%	120

As a total number of 1080, 360 and 360 data segments were selected for the training, validation and testing data respectively, hence the mini-batch size should be factor of 360, such as 120, 90, 60 and 30; otherwise, the network will discard the remaining part of any training data that does not equal to the size of previous mini-batch of training data. In order to study the effect of input training data per mini-batch on the performance of the CNN-Three architecture, four experiments were conducted: setting different sizes for the mini-batch including 30, 60, 90 and 120. The average classification accuracy obtained using these mini-batch sizes with different SNR levels are shown in Table 8-8.

Table 8-8: Comparison of classification accuracy using different mini-batch size for five SNR levels

Mini-Batch Size	SNR				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
30	99.80%	98.44%	97.27	96.22%	93.05%
60	99.88%	99.11%	98.08%	96.58%	93.72%
90	**99.91%**	**99.25%**	**98.11%**	**96.72%**	**94.22%**
120	99.86%	98.97%	97.52%	96.63%	94.02%

Figure 8-6 presents the results shown in Table 8-8 graphically, the green bar represents a mini-batch size of 30, orange is for 60, blue is for 90, and yellow is for 120. It can be seen from Figure 8-6 that for a given level of the SNR the best classification accuracy was achieved when the mini-batch was set to 90. It is noticeable in Figure 8-6 that mini-batch size of 120 (yellow bar) gave a lower classification accuracy than achieved by the 90 mini-batch, for all five SNR levels, confirming that increasing the mini-batch size will result in each mini-batch comprise of many training data and hence fewer updates per training epoch, this can lead to in a state of convergence that is not the local optimum.

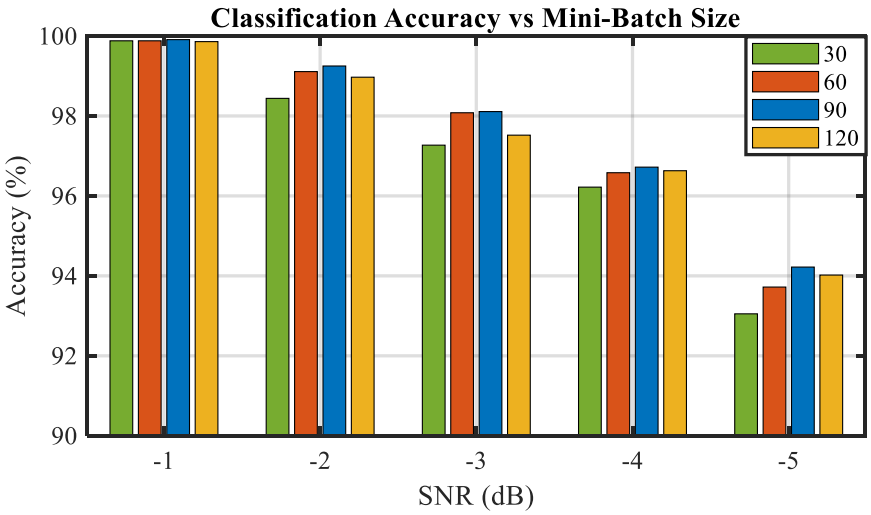

Figure 8-6: Classification accuracy using different mini-batch sizes for five levels of SNR

A lower classification accuracy was obtained when using a small mini-batch size (30 and 60) as a small mini-batch size will result in each mini-batch containing relatively few training data, and the frequent updates can cause unstable gradients and may overshoot or fluctuate around the minimum error value.

Based on the classification accuracy shown in Table 8-8 and Figure 8-6, it can be seen that the CNN-Three architecture with mini-batch size of 90 achieved the best classification accuracy compared to the others mini-batch sizes (30, 60 and 120). Therefore, a mini-batch size of 90 was used as the optimal parameter for the developed CNN-Three architecture.

8.2.2.7. Learning Rate

As discussed in Section 4.4.4, the learning rate refers to the step size taken when updating the model parameters during the training process. As studied previously in [220, 221], it

is usually recommended to use a lower learning rate when training the neural network as it improves network training performance compared to a higher learning rate [222]. In this study, the learning rate was selected to be as small as possible to obtain smooth error minimisation during the training process with a learning rate value of 0.04.

8.2.2.8. Epoch

The training epoch refers to the number of iterations of the entire N mini-batches training data into the model required to complete one cycle training epoch. The epoch parameter is dependent on the selection of other parameter to complete the specified epoch number, e.g., using the early stopping technique during the training process. If the validation loss decreases to a lower error value, this means that the generalisation ability of the model is improved and the model continues the training process until it reaches the specified value of the training epoch. However, if the validation loss increases to a higher error value, this means that the model at this stage has begun to over-fit the training data. Consequently, the training process will be terminated (early stopping) before completing the specified epoch number to avoid overfitting.

8.2.3. Validation

The validation step is carried out by feeding the developed CNN-Three architecture with a set of unseen data (validation data) that were not used in training but follow the same distribution and model relationship. This step is commonly used for producing the final model and to avoid overfitting via early stopping technique. Validation patience is the number of times that the loss on the validation data can be equal or larger to the previously smallest loss value before the training process is terminated. It involves stopping the training process if the validation loss does not improve (start to increase to a higher error value) with a validation patience number of 5 epochs.

8.2.4. Testing

Testing is the final step of applying the developed CNN-Three architecture to simulated data. It is carried out by feeding the trained model with a set of unseen data (testing data) to evaluate the overall performance of the model. In any supervised classification task, the results obtained from the classifier are built from an NxN dimensional confusion matrix, where N is the number of target classes. This matrix is commonly used to evaluate

the performance of classification models. It compares the predicted classes against the target/actual classes [260]. Each class in the confusion matrix has one row and one column ($N_{i,j}$), where i^{th} denotes the predicted class, and j^{th} represents the target/actual class. Ideally, a high classification accuracy will be obtained by having large numbers of the predicted samples on the main diagonal, where $i = j$ [261]. In general, the confusion matrix provides four possible classification outcomes with respect to one target class. Figure 8-7 shows the confusion matrix for a multi-class classification model with 3 classes.

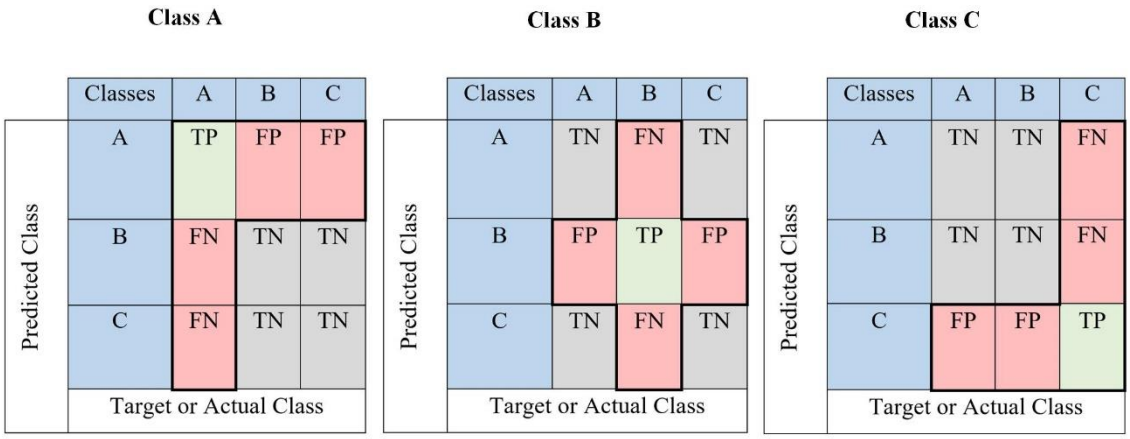

Figure 8-7: Examples of how to present TP, TN, FN, and FP in confusion matrix for each class in multi-class classification

In Figure 8-7 True Positive (TP) refers to positive samples (baseline data) correctly classified as positive (baseline data), False Positive (FP) is when negative samples (fault data) are incorrectly classified as positive (baseline data), False Negative (FN) is when positive samples (baseline data) are incorrectly classified as negative (fault data), and True Negative (TN) is when negative samples (fault data) are correctly classified as negative (fault data).

In order to evaluate the performance of any supervised classification model, the classification accuracy can be calculated from the above four possible outcomes (TP, TN, FP, and FN). The classification accuracy is defined as the percentage of correctly classified samples (TP) in relation to the total number of samples (TP, TN, FP, and FN) [262]. From the confusion matrix in Figure 8-8, the correct classifications are the sum of the TP samples for each class divided by the total number of samples. The classification accuracy of the model is calculated as in Equation 8.3 [263]:

$$Classification\ Accuracy = \frac{\sum_{i=1}^{N} TP_i}{Total\ samples} = \frac{\sum_{i=1}^{N} TP_i}{TP+TN+FP+FN} * 100 \quad (8.3)$$

Where i is the number of classes.

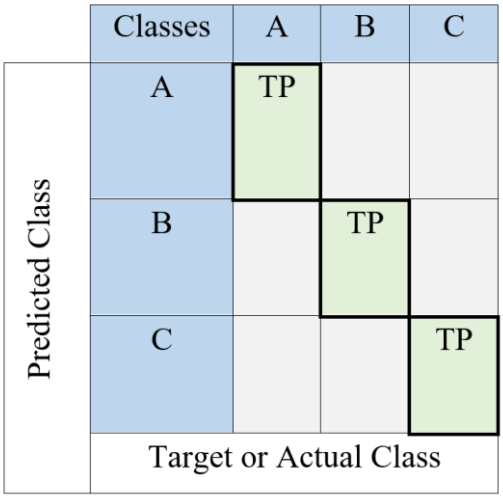

Figure 8-8: Determination of classification accuracy of the model using a confusion matrix for multi-class classification

8.3. Evaluation of the Developed CNN-Three Architecture Using Simulated Data

This section presents the evaluation of the performance of the developed CNN-Three architecture using the simulated data with five SNR levels from (-1 dB) to (-5 dB). The effectiveness of the developed CNN-Three architecture will be evaluated and compared with three recent CNN architectures. To ensure consistency in the comparison, the default activation function (ReLU) is employed in all the architectures. The results obtained from applying the developed CNN-Three and the three recent CNN architectures to the simulated data are presented and discussed in Section 8.5.

8.3.1. Training Step

The developed CNN-Three architecture was trained using three sets of simulated data with different fault severities, and a small amount of noise was added to the data with SNR of (4 dB), as seen in Figure 8-2. The training step starts with segmenting the simulated data into N segments, each of length 2500 sufficient to cover more than one period of the expected fault feature. For the simulated data of duration 15 seconds, a total

number of 600 data segments were obtained for each level of fault severity (small, medium and large). As discussed in Section 8.2.2.6, 60% of the 600 data segments for each severity (360 data segments) were selected randomly as the training data, 20% (120 data segments) were selected randomly as the validation data, and the remaining 20% (120 data segments) were selected for the testing data. The developed CNN-Three model parameters listed in Table 8-1, were set to the optimal values as found in the network optimisation described in Section 8.2.2. In this study, the developed CNN-Three model was trained using the simulated data shown in Figure 8-2, and then the trained CNN-Three model was tested with the same simulated data but with five SNR levels ranging from (-1 dB) to (-5 dB), see Figure 8-3.

8.3.2. Testing Step

The trained CNN-Three model was tested with the 120 data segments for each severity (in total 360 segments for the three levels of fault severity) of simulated data with five SNR levels. Figure 8-9 shows the classification accuracy obtained from using the CNN-Three model with ReLU function to classify three sets of simulated data with increasing SNR levels. It can be seen that the classification accuracy achieved by the developed CNN-Three architecture was 99.31% accuracy for the simulated datasets with SNR of (-1 dB) and 85.11% accuracy with SNR level (-5 dB). The classification accuracy obtained in Figure 8-9 shows that the developed CNN-Three architecture is an effective and reliable model.

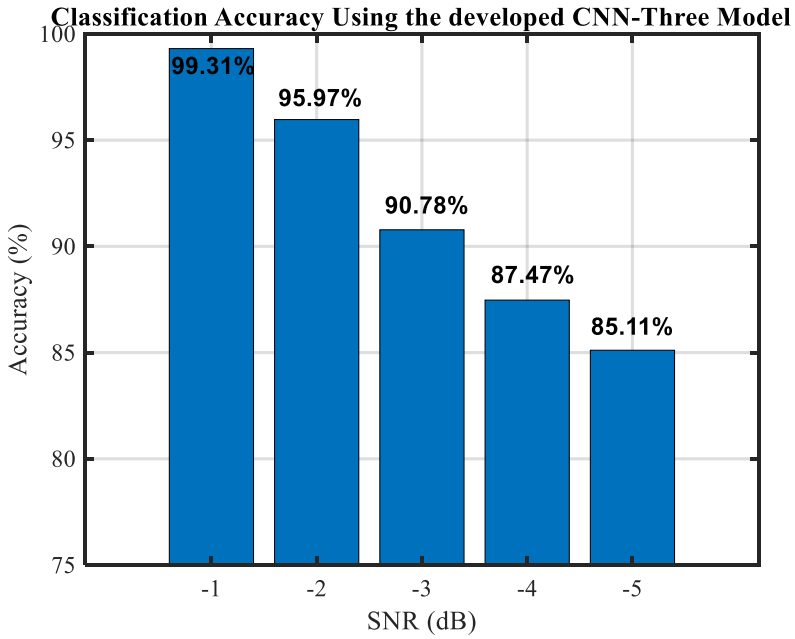

Figure 8-9: Classification accuracy for the developed CNN-Three architecture using simulated data with five SNR levels

8.3.2.1. Comparison of the Developed CNN-Three Architecture Against Existing CNN Architectures

To evaluate the effectiveness and performance of the developed CNN-Three architecture, the following three recent CNN architectures were used as comparisons.

- **CNN** [92]**:** this study proposed CNN architecture to learn features directly from raw data and then evaluated using experimental vibration data collected from a PG. In this study, the proposed CNN architecture consisted of five layers: convolutional, ReLU function, max pooling, fully connected, and finally softmax layers. More details of the CNN architecture can be found in [92]. Details of the network parameters used for the CNN are shown in Table 8-9.

Table 8-9: Network parameters for the CNN architecture

No.	Layer Type	Filter Size	Number of Filters	Stride
1	Convolution	$32 * 1$	10	$32 * 1$
2	Max-Pooling	$2 * 1$	10	$2 * 1$

- **1D-DCNN** [123]**:** this study used a one-dimensional deep convolutional neural network (1D-DCNN) architecture for machinery CM, which was evaluated using experimental vibration data collected from a gearbox. The proposed 1D-DCNN architecture comprises of two CNN feature extraction groups, each group consisted of convolutional, ReLU function, and max pooling layers, followed by fully connected, dropout and softmax layers. More details of the 1D-DCNN architecture can be found in [123]. The network parameters used for the 1D-DCNN are shown in Table 8-10.

Table 8-10: Network parameters for the 1D-DCNN architecture

No.	Layer Type	Filter Size	Number of Filters	Stride
1	Convolution	$257 * 1$	24	$2 * 1$
2	Max-Pooling	$2 * 1$	24	$1 * 1$
3	Convolution	$127 * 1$	48	$2 * 1$
4	Max-Pooling	$2 * 1$	48	$1 * 1$

- **DCNN** [203]**:** this study also used a deep convolutional neural network (DCNN) architecture for data fusion of vibration measurements made in two-directions to identify the health condition of a PG. The proposed DCNN architecture consisted of six CNN feature extraction groups, each group comprised of convolutional, batch normalisation, ReLU function, and max pooling layers, followed by fully connected, dropout and softmax layers at the end of the model. The proposed DCNN architecture was evaluated using experimental vibration data collected from the PG. More details of the DCNN architecture can be found in [203]. Details of the network parameters used for the DCNN are shown in Table 8-11.

Table 8-11: Network parameters for the DCNN architecture

No.	Layer Type	Filter Size	Number of Filters	Stride
1	Convolution	$512 * 1$	16	$2 * 1$
2	Max-Pooling	$2 * 1$	16	$2 * 1$
3	Convolution	$65 * 1$	32	$1 * 1$
4	Max-Pooling	$2 * 1$	32	$1 * 1$
5	Convolution	$3 * 1$	64	$1 * 1$
6	Max-Pooling	$2 * 1$	64	$1 * 1$
7	Convolution	$3 * 1$	64	$1 * 1$
8	Max-Pooling	$2 * 1$	64	$1 * 1$
9	Convolution	$3 * 1$	64	$1 * 1$
10	Max-Pooling	$2 * 1$	64	$1 * 1$
11	Convolution	$3 * 1$	64	$1 * 1$
12	Max-Pooling	$2 * 1$	64	$1 * 1$

The above-mentioned CNN, 1D-DCNN and DCNN architectures were applied to the simulated data shown in Figure 8-2, and then the trained CNN, 1D-DCNN and DCNN models were tested using the same simulated data but with five SNR levels ranging from (-1 dB) to (-5 dB), as shown in Figure 8-3. The classification accuracies obtained from CNN, 1D-DCNN and DCNN were used for comparison with the developed CNN-Three architecture.

To evaluate the effect of different CNN architecture configurations on the learning performance of the network, unseen validation data was fed into the trained CNN-Three, DCNN, 1D-DCNN and CNN models using the same simulated data but with different SNR levels. Figure 8-10 (a) and (b) show the average validation accuracy and loss curves for the developed CNN-Three, DCNN, 1D-DCNN and CNN architectures with SNR of (-3 dB).

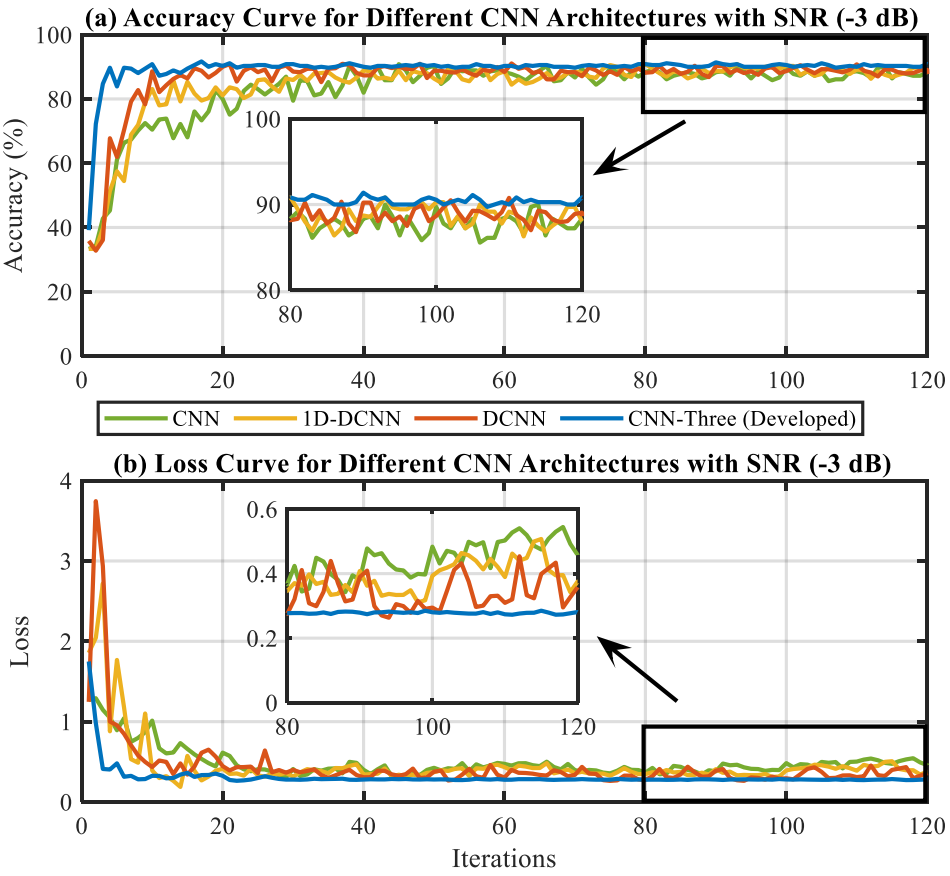

Figure 8-10: Validation accuracy and loss curves for different CNN architectures with SNR of (-3 dB)

Figure 8-10 (a) and (b) show the accuracy and loss curves after 120 iterations (10 epochs). It can be observed in Figure 8-10 (a) that the accuracy curve for the developed CNN-Three architecture is slightly higher than DCNN, 1D-DCNN and CNN architectures. It reaches a maximum classification accuracy of 90% with an SNR level of (-3 dB). A lower classification accuracy was achieved using the DCNN, 1D-DCNN and CNN architectures, with accuracy values of 88%, 88% and 87%, respectively.

It can also be seen in Figure 8-10 (b) that the loss curve for the developed CNN-Three architecture decreased to a lower error rate. Therefore, it can be said that the developed CNN-Three architecture learned better feature representation from the simulated datasets and the model parameters reach its optimum values with a lower error rate compared to the other CNN architectures. It can also be observed in Figure 8-10 (b) that the loss curve reached its minimum error value after 24 iterations (2 epochs) and remained stable until the training process was terminated (early stopping) at iteration number 120 (10 epochs) to avoid over-training and over-fitting the data. A higher error rate was obtained using

DCNN, 1D-DCNN and CNN architectures, as shown in Figure 8-10 (b). Therefore, it can be said that the network depth, filter size, number of filters, etc., have a significant impact on the network training and hence on the overall performance of the network on unseen data.

Table 8-12 presents the multi-class confusion matrix obtained from ten trials for the developed CNN-Three, DCNN, 1D-DCNN and CNN architectures with SNR (-3 dB). As seen in Table 8-12, the vertical axis of the confusion matrix represents the predicted class of the unseen testing data under three fault severities (small, medium, large), and the horizontal axis represents the actual class under the three fault severities. Appendix A presents the other multi-class confusion matrix results obtained for the four architectures using simulated data with SNR levels of (-1 dB), (-2 dB), (-4 dB) and (-5 dB).

As shown in Table 8-12 for the small severity class, the DCNN achieved the highest classification, correctly classifying 1179 out of 1200 samples, but incorrectly classifying 21 samples as medium severity class. The second and third highest predictions were obtained when using 1D-DCNN and CNN-Three architectures, they correctly predicted 1104 and 1102 samples out of 1200 samples, respectively, as belonging to the small severity class. The lowest classification was achieved when using CNN architecture, only 1005 samples were correctly classified as belonging to the small severity class and 195 samples incorrectly classified as medium severity class.

Table 8-12: Multi-class confusion matrix for CNN-Three, DCNN, 1D-DCNN and CNN using simulated data with SNR (-3 dB)

			Actual Class		
	Class		Small	Medium	Large
Predicted Class	Small	CNN	1005	23	0
		1D-DCNN	1104	46	0
		DCNN	1179	18	0
		CNN-Three	1102	0	0
	Medium	CNN	195	982	46
		1D-DCNN	96	875	3
		DCNN	21	802	0
		CNN-Three	98	966	0
	Large	CNN	0	195	1154
		1D-DCNN	0	279	1197
		DCNN	0	380	1200
		CNN-Three	0	234	1200
Ground Truth			1200	1200	1200

As shown in Table 8-12, for the medium severity class, the highest classification was achieved using the CNN architecture, 982 out of 1200 samples were correctly predicted as medium severity class, 23 and 195 samples incorrectly predicted as belonging to the small and large severity classes, respectively. The developed CNN-Three architecture achieved the second highest prediction, correctly classifying 966 samples out of 1200 samples, and 234 samples incorrectly predicted as large severity class. The lowest classification was obtained using the DCNN architecture, only 802 samples out of 1200 samples were correctly classified, 380 samples incorrectly classified as belonging to the large severity class, and the remaining 18 samples incorrectly classified as small severity class.

For the large severity class, it can be seen in Table 8-12 that the highest classification was obtained by both the developed CNN-Three and DCNN architectures. Both models correctly classified all 1200 samples as belonging to the large severity class. The 1D-DCNN correctly classified 1197 samples out of 1200 samples as belonging to the large severity class, and only 3 samples incorrectly classified as belonging to the medium severity class. The lowest classification was obtained using the CNN architecture, 1154

samples out of 1200 samples were correctly classified, and 46 samples incorrectly classified as belonging to the medium severity class.

Based on the classification presented in the multi-class confusion matrix in Table 8-12 and Appendix A, the average classification accuracies for the ten trials for CNN, 1D-DCNN, and DCNN architectures using simulated data with five SNR levels are shown in Figure 8-11 to Figure 8-13.

Figure 8-11 shows the classification accuracy obtained from applying the CNN architecture to classify three sets of simulated fault data (small, medium, and large) with five SNR levels. The CNN architecture achieved a lowest classification accuracy compared to 1D-DCNN, DCNN, and the developed CNN-Three architectures. Table 8-12 shows that the CNN architecture correctly classified 3141 samples out of a total of 3600 samples, achieving 87.25% accuracy with a SNR level of (-3 dB). For the other SNR levels, the CNN architecture achieved 96.19%, 92.27%, 83.91%, and 77.13% classification accuracies for SNR levels of (-1 dB), (-2 dB), (-4 dB), and (-5 dB), respectively.

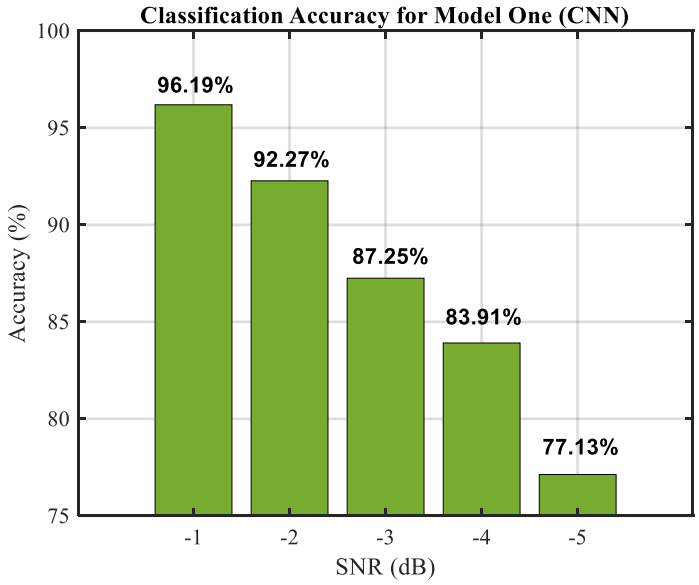

Figure 8-11: Classification accuracy for CNN architecture using simulated data with five SNR levels

Using the 1D-DCNN architecture gave an improvement when classifying the three sets of simulated data with five different SNR levels, see Figure 8-12. The average classification accuracy for 1D-DCNN was higher than the CNN architecture. The 1D-DCNN architecture achieved 97.77%, 92.72%, 88.22%, 85.86%, and 82.61%

classification accuracy for five SNR levels (-1 dB), (-2 dB), (-3 dB), (-4 dB), and (-5 dB), respectively.

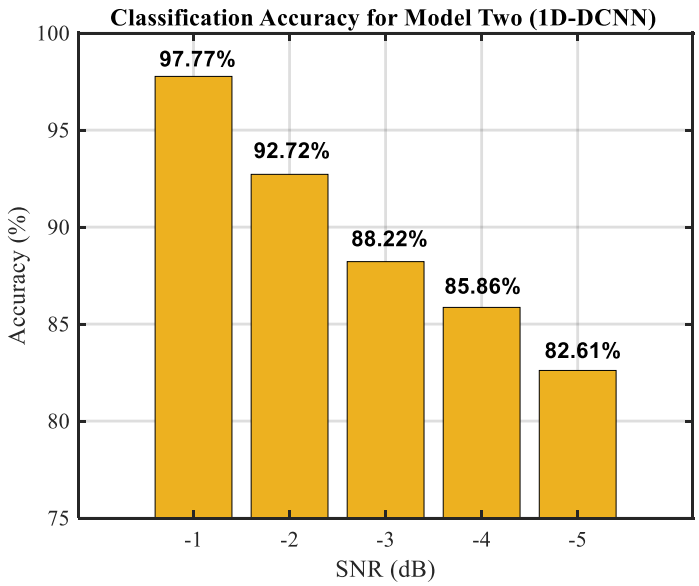

Figure 8-12: Classification accuracy for 1D-DCNN architecture using simulated data with five SNR levels

A further slight improvement in the classification accuracy was achieved by using the DCNN architecture, see Figure 8-13, with an accuracy values of 98.44%, 93.05%, 88.36%, 86.08%, and 84.30% for SNR levels of (-1 dB), (-2 dB), (-3 dB), (-4 dB), and (-5 dB), respectively.

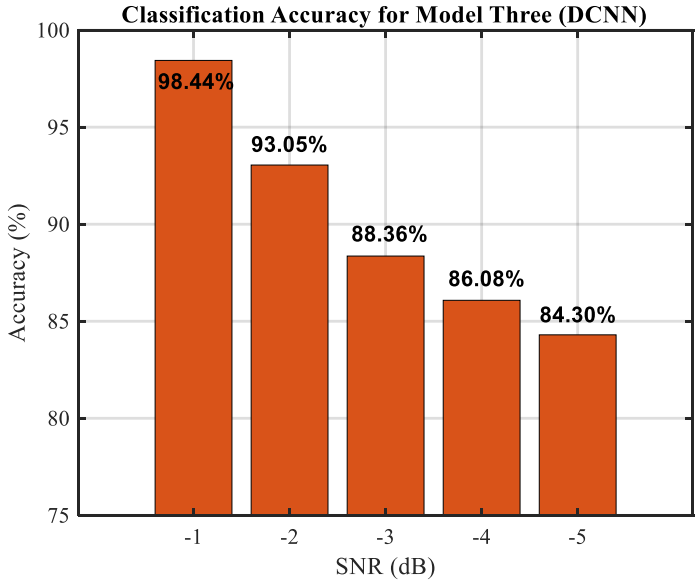

Figure 8-13: Classification accuracy for DCNN architecture using simulated data with five SNR levels

Based on the results obtained from Figure 8-9 to Figure 8-13, it can be said that the developed CNN-Three with the default ReLU activation function achieved the highest classification accuracy for the three sets of simulated data with five SNR levels. Table 8-12 shows that the developed CNN-Three model correctly classified 3268 samples out of a total of 3600 samples, achieving 90.78% classification accuracy for the simulated datasets with SNR level of (-3 dB). The developed CNN-Three architecture confirmed its improvement in classifying three sets of simulated fault data (small, medium, and large) with a classification accuracy improvement of 1% compared to the DCNN model, 2% improvement compared to the 1D-DCNN model, and 4% improvement compared to the CNN model. Therefore, it can be said that applying the developed CNN-Three architecture to simulated data yields a robust classification with high diagnostic accuracy and outperforms the CNN, 1D-DCNN, and DCNN.

8.4. Evaluation of the Proposed IReLU-Tanh Function Using Simulated Data

In this section, the developed CNN-Three architecture with the proposed IReLU-Tanh function will be applied to the simulated data generated using Equation 8.1, in a manner similar to that used in Section 8.3. The effectiveness of the developed CNN-Three architecture with the proposed IReLU-Tanh function will be evaluated and compared with the most widely used activation functions, Tanh, ReLU, LReLU, and ELU. The results obtained are presented and discussed in Section 8.5.

Figure 8-14 shows the classification accuracy obtained from applying the developed CNN-Three with the proposed IReLU-Tanh function for the simulated datasets with five SNR levels. It can be seen that the classification accuracy achieved by the developed CNN-Three architecture with the proposed IReLU-Tanh function was 99.91% accuracy for the simulated datasets with SNR of (-1 dB) and 94.22% accuracy with SNR level of (-5 dB).

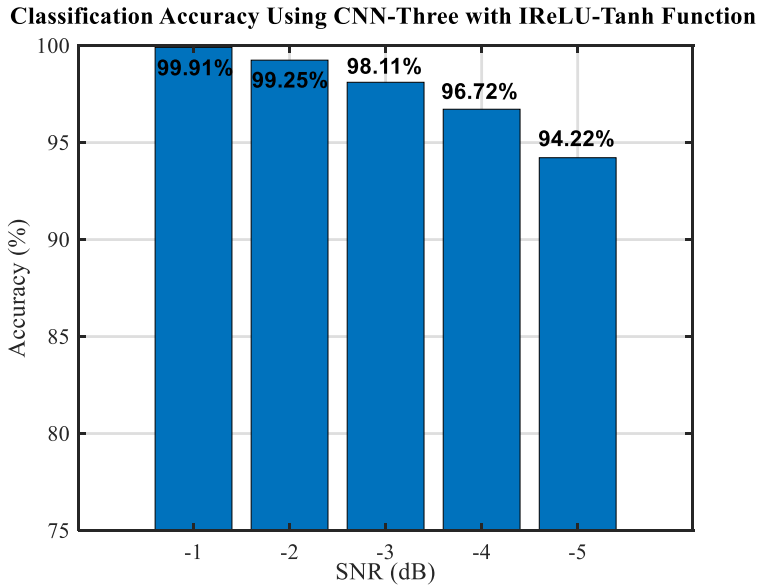

Figure 8-14: Classification accuracy for CNN-Three with IReLU-Tanh function for simulated data with five SNR levels

8.4.1. Comparison of the Proposed IReLU-Tanh Function Against Existing Activation Functions

To evaluate the effect of the activation functions, Tanh, ReLU, LReLU, ELU and IReLU-Tanh on the learning performance of the developed CNN-Three architecture, unseen validation data was fed into the trained CNN-Three model using the same simulated data but with different SNR levels. Figure 8-15 (a) and (b) show the average validation accuracy and loss curves for the different activation functions: the proposed IReLU-Tanh, ELU, LReLU, ReLU and Tanh with SNR of (-3 dB).

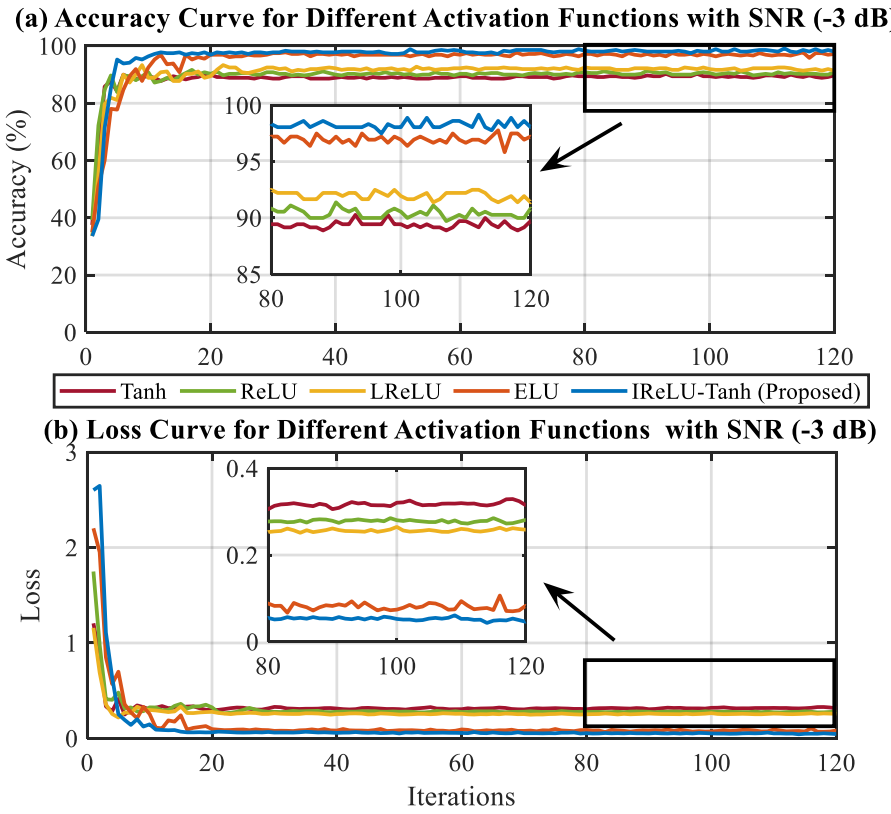

Figure 8-15: Validation accuracy and loss curves for different activation functions with SNR (-3 dB)

Figure 8-15 (a) and (b) show the accuracy and loss curves after 120 iterations (10 epochs). It can be seen in Figure 8-15 (a) that the accuracy curve for CNN-Three with IReLU-Tanh function is higher than the existing activation functions (ELU, LReLU, ReLU and Tanh). With an SNR of (-3 dB), the developed CNN-Three with the proposed IReLU-Tanh achieved the highest classification accuracy of 98%. A lower classification accuracy was achieved by the other activation functions, with an accuracy values of 96%, 92%, 90% and 89% for ELU, LReLU, ReLU and Tanh functions, respectively.

It can be seen in Figure 8-15 (b) that the loss curve of the proposed IReLU-Tanh function decreased to a lower error rate compared to the other activation functions. Therefore, it can be said that the developed CNN-Three architecture with the proposed IReLU-Tanh function learned better feature representation from the raw data and yields a higher classification accuracy curve with a lower error rate compared to the existing activation functions. It can also be seen in Figure 8-15 (b) that the loss curve has reached its minimum error value after 24 iterations (2 epochs) and remained stable until the training process was terminated (early stopping) at iteration number 120 (10 epochs) to avoid

over-training and over-fitting the data. A higher error rate was obtained when using the CNN-Three with the other activation functions, ELU, LReLU, ReLU and Tanh, as illustrated in Figure 8-15 (b). Therefore, it can be said that the drawbacks reported in Chapter Five, including vanishing gradient, dead neuron, and fixed gradient value have a significant impact on the network training and hence on the overall performance of the model, and consequently lead to decreased the classification accuracy of the network on unseen data.

The results obtained from applying CNN-Three with different activation functions to simulated data is evaluated by constructing a multi-class confusion matrix and calculating the classification accuracy of the model using Equation 8.3. Table 8-13 presents the multi-class confusion matrix from ten trials under different activation functions (the proposed IReLU-Tanh, ELU, LReLU, ReLU and Tanh) with SNR of (-3 dB). Appendix B presents the other multi-class confusion matrix results obtained for the given activation functions using simulated data with SNR levels of (-1 dB), (-2 dB), (-4 dB) and (-5 dB).

As shown in Table 8-13 for the small severity class, the highest classification was achieved when using the developed CNN-Three with the proposed IReLU-Tanh function, as it correctly classified 1180 samples out of 1200 samples belonging to the small severity class, and 20 samples incorrectly classified as medium severity class. The second highest classification was obtained using the CNN-Three with ELU function, it correctly classified 1174 samples as belonging to the small severity class, and 26 samples incorrectly classified as belonging to the medium severity class. The lowest classification was obtained when applying the CNN-Three with the ReLU function, only 1102 samples were correctly classified and 98 sample incorrectly classified as belonging to the medium severity class.

Table 8-13: Multi-class confusion matrix for different activation functions using simulated data with SNR (-3 dB)

			Actual Class		
	Class		Small	Medium	Large
Predicted Class	Small	Tanh	1171	87	0
		ReLU	1102	0	0
		LReLU	1123	0	0
		ELU	1174	22	0
		IReLU-Tanh	1180	10	0
	Medium	Tanh	29	1061	204
		ReLU	98	966	0
		LReLU	77	1015	0
		ELU	26	1137	21
		IReLU-Tanh	20	1163	11
	Large	Tanh	0	52	996
		ReLU	0	234	1200
		LReLU	0	185	1200
		ELU	0	41	1179
		IReLU-Tanh	0	27	1189
Ground Truth			1200	1200	1200

For the medium severity class, it can be seen in Table 8-13 that the proposed IReLU-Tanh function achieved the highest correct classification, 1163 samples correctly classified as medium severity class, 10 and 27 samples incorrectly classified as belonging to the small and large severity classes, respectively. The developed CNN-Three with ELU function achieved the second highest correct classification, it classified 1137 samples out of 1200 samples as correctly belonging to the medium severity class, 22 samples incorrectly classified as small severity class, and the remaining 41 samples misclassified as large severity class.

As seen in Table 8-13 for the large severity class, it can be seen that a highest correct classification was obtained using CNN-Three with the ReLU and LReLU functions. In both scenarios, all 1200 samples were correctly classified as belonging to the large severity class. The developed CNN-Three with the proposed IReLU-Tanh function correctly classified 1189 samples as belonging to large severity class, and 11 samples incorrectly classified as belonging to medium severity class. The lowest classification

was obtained when using the developed CNN-Three with Tanh function, only 996 samples out of 1200 samples were correctly classified as belonging to the large severity class, and 204 samples incorrectly classified as belonging to the medium severity class.

Based on the classification obtained from the multi-class confusion matrix in Table 8-13 and Appendix B, the average classification accuracy for the ten trials for Tanh, ReLU, LReLU and ELU functions using simulated data with five SNR levels are shown in Figure 8-16 to Figure 8-19.

Figure 8-16 shows the classification accuracy obtained from applying the CNN-Three with Tanh function to classify three sets of simulated data (small, medium and large) with five different SNR levels. It can be seen that the CNN-Three with Tanh function achieved the lowest classification accuracy compared to ReLU, LReLU, ELU and the proposed IReLU-Tanh functions. For example, Table 8-13 shows that CNN-Three with Tanh function correctly classified 3228 out of the total number of 3600 samples, achieving 89.67% accuracy when classifying the three sets of simulated data with SNR level of (-3 dB). For the SNR levels of (-1 dB), (-2 dB), (-4 dB), and (-5 dB), it achieved 98.14%, 94.64%, 85.56%, and 83.81% classification accuracies, respectively.

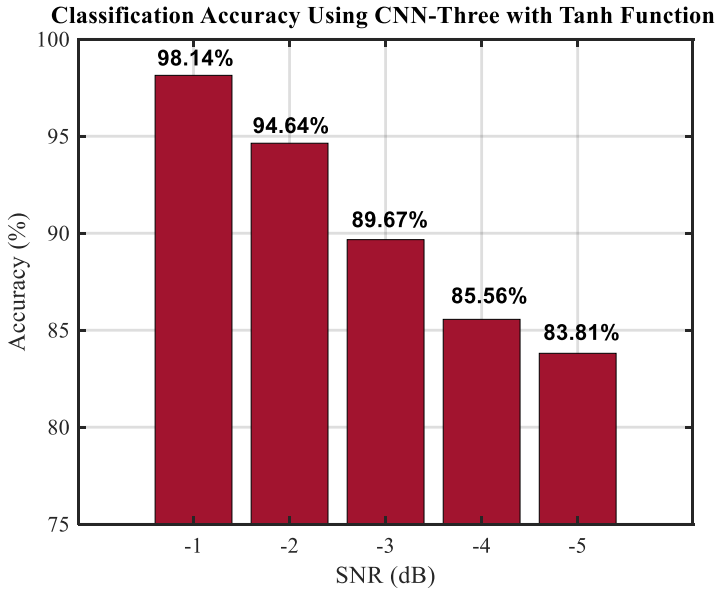

Figure 8-16: Classification accuracy for CNN-Three with Tanh function using simulated data with five SNR levels

Figure 8-17 shows the classification accuracy achieved when using the CNN-Three with ReLU function for the three simulated datasets with five different SNR levels. It can be

seen that the average classification accuracy obtained for CNN-Three with ReLU function is higher than Tanh function, it achieved 99.31%, 95.97%, 90.78%, 87.47%, and 85.11% classification accuracies for SNR levels of (-1 dB), (-2 dB), (-3 dB), (-4 dB), and (-5 dB), respectively.

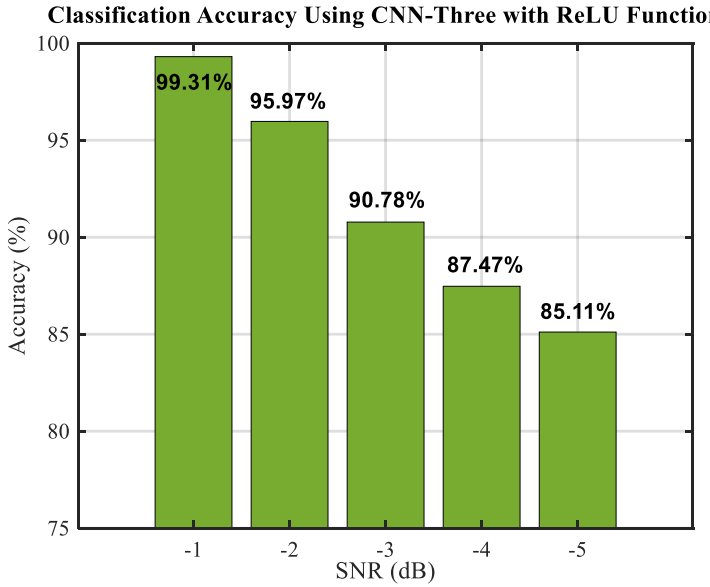

Figure 8-17: Classification accuracy for CNN-Three with ReLU function using simulated data with five SNR levels

Figure 8-18 shows that a slight improvement of around 1% was achieved when using the developed CNN-Three with LReLU function compared to the ReLU function. The average classification accuracy decrease from 99.42% accuracy with SNR level of (-1 dB) to 86.36% accuracy with SNR level of (-5 dB).

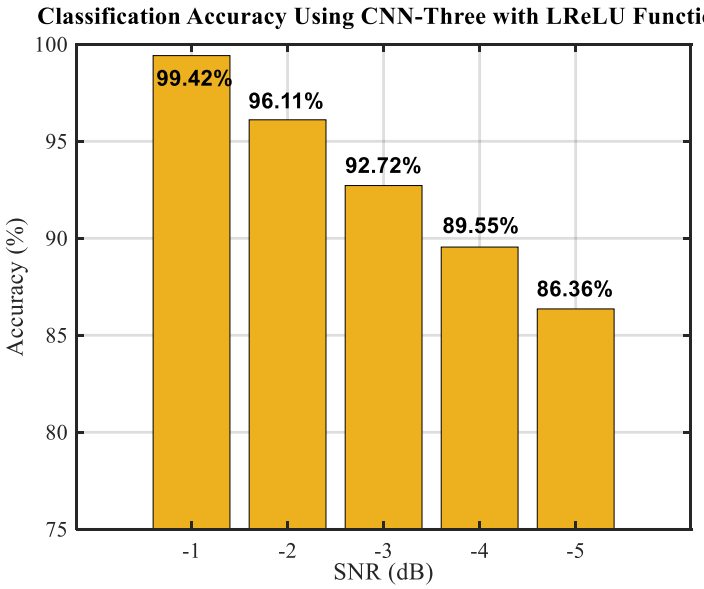

Figure 8-18: Classification accuracy for CNN-Three with LReLU function using simulated data with five SNR levels

Figure 8-19 shows the classification accuracy achieved when using the CNN-Three with ELU function, there was a further improvement in the classification accuracy compared to LReLU function. The CNN-Three with ELU function achieved classification accuracies of 99.83%, 98.83%, 96.94%, 95.08%, and 92.94% for SNR levels of (-1 dB), (-2 dB), (-3 dB), (-4 dB), and (-5 dB), respectively.

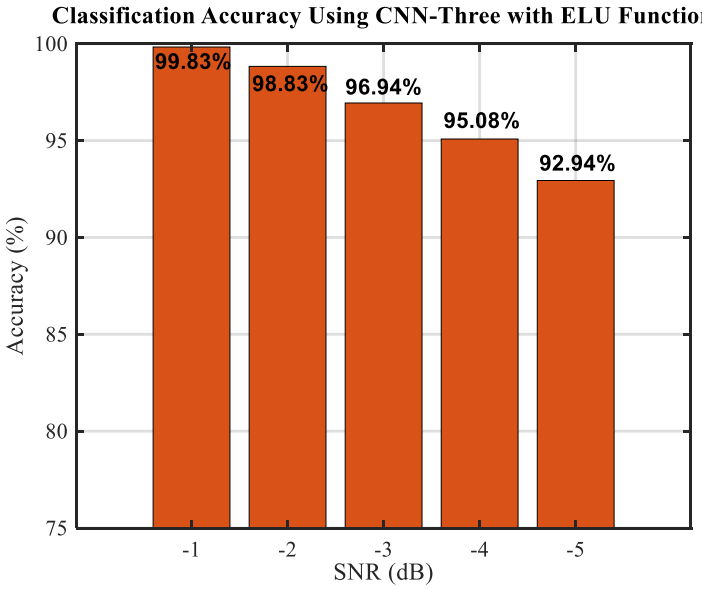

Figure 8-19: Classification accuracy for CNN-Three with ELU function using simulated data with five SNR levels

Based on the results obtained from Figure 8-14 to Figure 8-19, it can be said that the developed CNN-Three with the proposed IReLU-Tanh function achieved the highest classification accuracy for the three sets of simulated data with five SNR levels. Table 8-13 shows that the developed CNN-Three with the proposed IReLU-Tanh function correctly classified 3532 samples out of a total of 3600 samples, achieving 98.11% classification accuracy for the simulated datasets with SNR level of (-3 dB). In addition, the classification accuracies shown in Figure 8-14 are all above 94% when classifying three sets of simulated data with SNR levels from (-1 dB) to (-5 dB). Therefore, it can be said that the developed CNN-Three with the proposed IReLU-Tanh function to simulated data yields a robust classification with high diagnostic accuracy and outperforms the existing activation functions: ELU, LReLU, ReLU and Tanh.

8.5. Results and Discussion

Based on the results obtained in Section 8.3, the developed CNN-Three architecture has shown its effectiveness and ability to learn the representative features directly from the simulated datasets and yield a robust classification with high diagnostic accuracy. The developed CNN-Three model achieved the highest classification accuracy and outperformed the CNN, 1D-DCNN and DCNN architectures. As seen in Table 8-14 that the developed CNN-Three architecture confirmed its improvement in classifying three sets of simulated data (small, medium, and large) with a classification accuracy improvement of 1% compared to the DCNN model, 2% improvement compared to the 1D-DCNN model, and 4% improvement compared to the CNN model.

Table 8-14: Average classification accuracies of ten trials for the developed CNN-Three and three other models with SNR levels from (-1 dB) to (-5 dB)

Model	SNR				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
CNN	96.19%	92.27%	87.25%	83.91%	77.13%
1D-DCNN	97.77%	92.72%	88.22%	85.86%	82.61%
DCNN	98.44%	93.05%	88.36%	86.08%	84.30%
CNN-Three	**99.31%**	**95.97%**	**90.78%**	**87.47%**	**85.11%**

The advantage of the developed CNN-Three architecture is that it consists of three CNN feature extraction groups, each group contains convolutional, batch normalisation, activation function, max pooling layers, followed by fully connected and softmax layers and therefore the CNN-Three is deeper than CNN and 1D-DCNN architectures. Table 8-14 shows that the CNN, 1D-DCNN models achieved lower classification accuracies compared to DCNN and the CNN-Three models, this is because their architectures are not deep enough and too simple to learn the representative hidden features effectively from the raw data. As discussed in Section 8.2.2.1, the network depth is a high priority for improving the classification performance of a network [254]. The deeper the network, the stronger its ability to learn representative features from the data. However, classification accuracy gets degraded if the number of layers is overly increased [255] because adding more layers means a large number of model parameters to train, which makes deeper networks are more prone to overfitting problem [203]. The DCNN architecture achieved the second highest classification accuracy, however, from the results obtained in Table 8-14, it can be confirmed that when adding more layers into the network, it meant a large number of the DCNN parameters to train, hence it led to overfitting problem and resulted in a lower classification accuracy than the developed CNN-Three architecture.

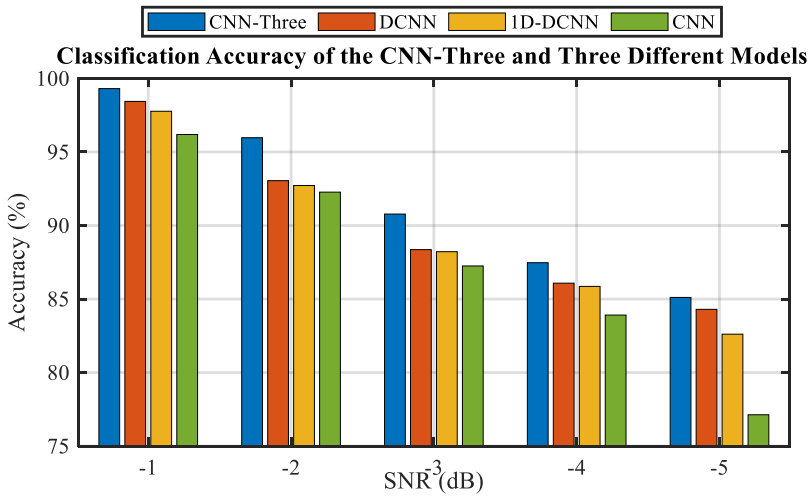

Figure 8-20: Comparison between the developed CNN-Three architecture and three other architectures using simulated data with five SNR levels

Figure 8-20 compares the classification accuracy obtained from the developed CNN-Three architecture and three other architectures (CNN, 1D-DCNN and DCNN). It can be clearly observed that the developed CNN-Three architecture yielded robust classification

with high diagnostic accuracy and outperformed the other architectures. The classification accuracy achieved by the developed CNN-Three architecture was 99.31% for the simulated datasets with SNR of (-1 dB) and 85.11% with SNR level (-5 dB).

Applying the developed CNN-Three with the proposed IReLU-Tanh function to the simulated data demonstrated that the network could reliably learn better feature representation from the raw data. As discussed in Section 8.4.1, the developed CNN-Three with the proposed IReLU-Tanh function reduces the loss curve to the lowest error rate, and consequently achieved the highest classification accuracy with a maximum value of 98% with SNR of (-3 dB). The developed CNN-Three with the proposed IReLU-Tanh function showed its effectiveness in achieving the highest classification accuracy and outperforming the most widely used activation functions, see Table 8-15.

Table 8-15: Average classification accuracies of ten trials for different activation functions with SNR levels from (-1 dB) to (-5 dB)

Function	SNR				
	-1 dB	-2 dB	-3 dB	-4 dB	-5 dB
Tanh	98.14%	94.64%	89.67%	85.56%	83.81%
ReLU	99.31%	95.97%	90.78%	87.47%	85.11%
LReLU	99.42%	96.11%	92.72%	89.55%	86.36%
ELU	99.83%	98.83%	96.94%	95.08%	92.94%
IReLU-Tanh (Proposed)	**99.91%**	**99.25%**	**98.11%**	**96.72%**	**94.22%**

The advantage of the proposed IRaLU-Tanh function is that it addresses the vanishing gradient problem in the Tanh function, by adopting the non-saturation property from the ReLU function for covering the positive region. As discussed in Section 5.5, during the forward propagation process, when the input value is greater than zero $(x > 0)$, the proposed IReLU-Tanh function output is equal to the input itself $(f(x) = x)$. In the case of backward propagation, the gradient value of the proposed IReLU-Tanh function is equal to $(f'(x) = 1)$. Moreover, the proposed IReLU-Tanh function addresses the shortcomings of dying neurons in ReLU function and fixed gradient value in the LReLU

function, by having $(f(x) = tanh(x))$ for $(x \leq 0)$. Also, the proposed IReLU-Tanh function does not require a hyper-parameter (α) to be included in the architecture. This means the saturation level for negative inputs is determined by the Tanh function, instead of requiring the addition of a hyper-parameter as the LReLU and ELU functions that need to be tuned based on a trial and error process. Therefore, the proposed IReLU-Tanh function enhances the network training by learning the representative features directly from simulated datasets, and consequently improves the overall performance of the network to obtain a higher classification accuracy compared to the ELU, LReLU, ReLU, and Tanh functions.

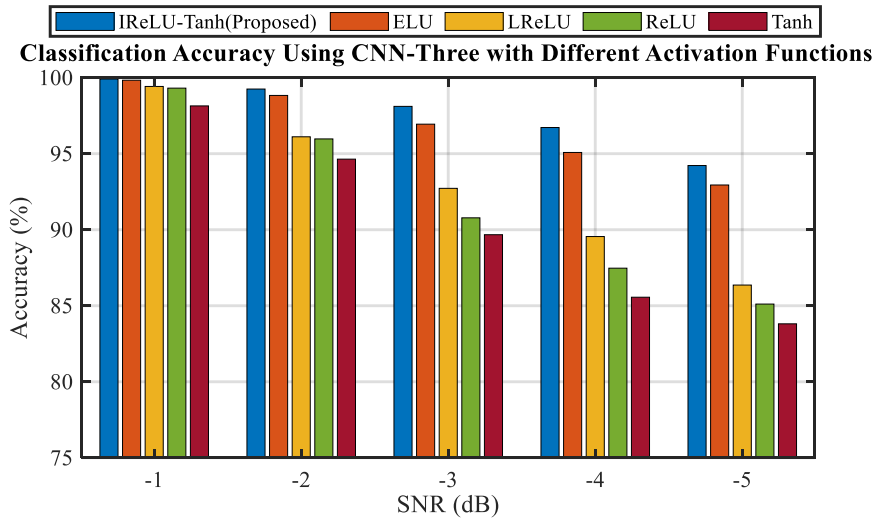

Figure 8-21: Comparison between the proposed IReLU-Tanh function and the existing activation functions using simulated data with five SNR levels

Figure 8-21 compares the classification accuracy obtained from the proposed IReLU-Tanh function with the existing activation functions, Tanh, ReLU, LReLU and ELU. It can be clearly seen that the highest classification accuracy was achieved using the developed CNN-Three with the proposed IReLU-Tanh function. It has been demonstrated that applying the developed CNN-Three with the proposed IReLU-Tanh function to simulated data yields robust classifications with high diagnostic accuracy and outperforms the most widely used activation functions.

Chapter Nine: Automated Data Processing Using Convolutional Neural Network for Real Data

This chapter presents an evaluation of the performance of the developed CNN-Three architecture with the proposed IReLU-Tanh function using experimentally recorded vibration data. The results obtained are presented and compared against the most widely used activation functions, Tanh, ReLU, LReLU, and ELU. Finally, the results obtained are discussed.

9.1. Introduction

This chapter presents the evaluation of the performance of the developed CNN-Three architecture with the proposed IReLU-Tanh function using vibration data obtained experimentally. It starts with the training step using five sets of vibration data (baseline, Sun-F1, Sun-F2, Planet-F1 and Planet-F2) to train the CNN-Three model with the IReLU-Tanh function. Then, the trained model was validated using unseen validation data. This was followed by the testing step where the trained model was tested with unseen testing data. The results obtained from applying the developed CNN-Three with the proposed IReLU-Tanh function to the experimental vibration data is presented and compared against the results obtained using the most widely used activation functions. Finally, the results are discussed at the end of the chapter.

9.2. Evaluation of the Proposed IReLU-Tanh Function Using Real Data

This section presents the evaluation of the performance of the developed CNN-Three with the proposed IReLU-Tanh using experimental vibration data collected from the PG test rig under different load conditions (zero, 25%, 50%, 75% and 90% of full load). The results obtained are compared with those obtained using existing activation functions, and finally the results are discussed in Section 9.3.

9.2.1. Training Step

The CNN-Three architecture was trained using five sets of vibration data under different fault severities on the sun and planet gears. Data set one was collected for the normal (baseline) condition, data sets two and three were collected from the sun gear under two different tooth breakage severities (Sun-F1 and Sun-F2), data sets four and five were collected from the planet gear under two different tooth breakage severities (Planet-F1 and Planet-F2). As previously, each data set was collected and measured at a sampling rate 100 kHz for a period of 30 seconds, hence each data set contained (3×10^6 samples). As shown in Section 7.2.2, the planet gear fault frequency is calculated as 12.34 Hz, the period of the expected fault feature corresponded to 8103 data points ($\frac{Fs}{f_{pf}} = \frac{100000}{12.34} =$

8103). Therefore, each data segment length was set to 8330 data points in order to ensure each data segment covered more than one period of the expected planet fault feature.

For 30 seconds measurement, a total number of 360 data segments were obtained for each of the five data sets (baseline, Sun-F1, Sun-F2, Planet-F1 and Planet-F2). In this way a total of 1800 data segments were obtained. For each data set, 60% of the 360 data segments (216 data segments) were selected randomly as the training data, 20% (72 data segments) were selected randomly as the validation data, and the remaining 20% (72 data segments) were selected as the testing data. Therefore, for all five data sets, a totals of 1080, 360 and 360 data segments were obtained for the training, validation, and testing data respectively, and each load condition had the same number of data segments. Table 9-1 shows the data segmentation used in this study.

Table 9-1: Details of the data segmentations used for experimental vibration data

Severity	Total Data Segments	Dataset	Percentage	Data Segments
Baseline	360	Training	60%	216
		Validation	20%	72
		Testing	20%	72
Sun-F1	360	Training	60%	216
		Validation	20%	72
		Testing	20%	72
Sun-F2	360	Training	60%	216
		Validation	20%	72
		Testing	20%	72
Planet-F1	360	Training	60%	216
		Validation	20%	72
		Testing	20%	72
Planet-F2	360	Training	60%	216
		Validation	20%	72
		Testing	20%	72

For experimental vibration data, the CNN-Three model parameters were set to the same values found in the network optimisation reported in Section 8.2.2, but with a decrease of the learning rate to 0.01, as the collected vibration data is more complicated and contains more components than the simulated data. In this experiment, the developed CNN-Three architecture was trained with five sets of vibration data (baseline, Sun-F1, Sun-F2, Planet-F1 and Planet-F2) under one load, and then the trained model was tested with five sets of vibration data under the same load condition.

9.2.2. Testing Step

Figure 9-1 shows the average classification accuracy of ten trials obtained from applying the developed CNN-Three with the proposed IReLU-Tanh to all five sets of experimental vibration data under five different load conditions. It can be observed from the classification accuracy obtained that the developed CNN-Three with the proposed IReLU-Tanh function is an effective and reliable method, achieving 93.03% 94.14%, 93.39%, 94.00%, and 94.11% for zero, 25%, 50%, 75%, and 90% loads, respectively.

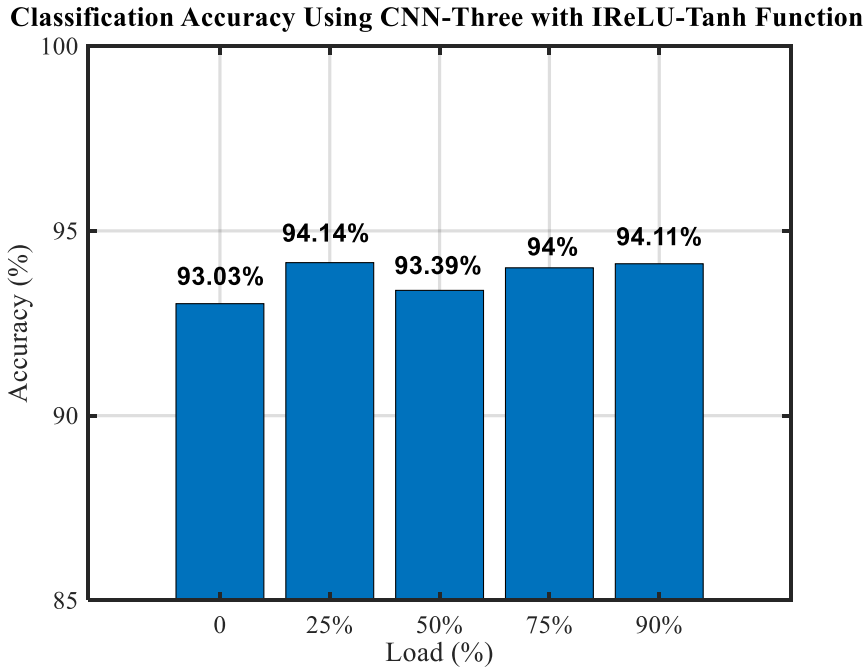

Figure 9-1: Classification accuracy for CNN-Three with IReLU-Tanh function using experimental vibration data under different load conditions

9.2.2.1. Comparison of the Proposed IReLU-Tanh Function Against Existing Activation Functions

In order to evaluate the effect of the activation functions Tanh, ReLU, LReLU, ELU and IReLU-Tanh on the learning performance of the developed CNN-Three model, unseen validation data was fed into the trained CNN-Three model under the same load condition. Figure 9-2 (a) and (b) show the average validation accuracy and loss curves for different activation functions, including the proposed IReLU-Tanh, ELU, LReLU, ReLU and Tanh, under 50% of full load.

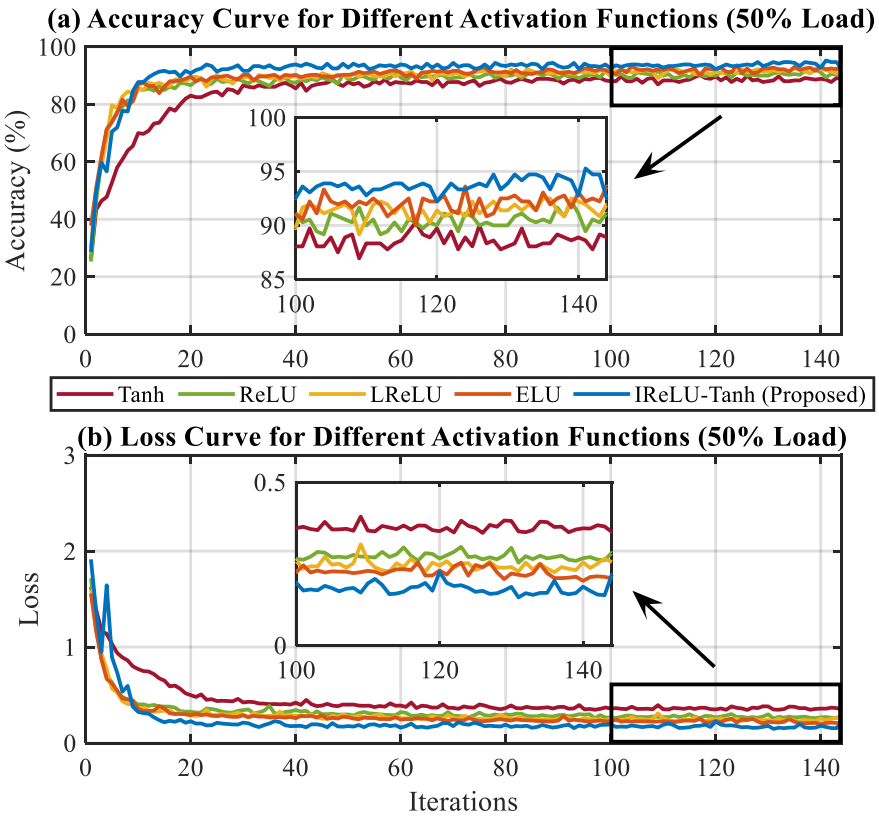

Figure 9-2: Validation accuracy and loss curves for different activation functions (50% Load)

Figure 9-2 (a) and (b) show the accuracy and loss curves after 144 iterations (12 epochs). It can be seen in Figure 9-2 (a) that the accuracy curve for CNN-Three with the proposed IReLU-Tanh function is higher than the existing activation functions, ELU, LReLU, ReLU and Tanh. It reached a maximum classification accuracy of 93% (under 50% load). A lower classification accuracy was obtained for the ELU, LReLU, ReLU and Tanh functions, achieving 91%, 90%, 90% and 87%, respectively as shown in Figure 9-2 (a).

It can also be observed in Figure 9-2 (b) that the loss curve of the proposed IReLU-Tanh function decreased to a lower error rate compared to the other activation functions. Therefore, it can be said that developed CNN-Three with the proposed IReLU-Tanh function enhanced the overall performance of the network in two aspects; firstly, in the training task the model parameters reach the optimum values with lower error rate so the network can learn the hidden features more effectively from the raw data. Secondly, it improves the classification accuracy of the network and yields better diagnostic accuracy compared to the other activation functions. It can also be seen in Figure 9-2 (b) that the loss curve has reached its minimum error value after 36 iterations (3 epochs) and remained stable until the training process was terminated (early stopping) at about iteration number 144 (12 epochs) to avoid over-training and over-fitting the data. A higher error rate was obtained when using the CNN-Three with the other activation functions, ELU, LReLU, ReLU and Tanh, as illustrated in Figure 9-2 (b). Therefore, it can be said that the existing activation functions have a significant impact on the network training and hence on the overall performance of the model, consequently lead to decreased the classification accuracy of the network on unseen data.

Table 9-2 presents the multi-class confusion matrix obtained from ten trials under different activation functions (IReLU-Tanh, ELU, LReLU, ReLU and Tanh) under 50% load. Appendix C presents the other multi-class confusion matrix results obtained for the given activation functions using experimental vibration data with zero, 25%, 75% and 90% loads.

As seen in Table 9-2 the developed CNN-Three with the proposed IReLU-Tanh achieved the highest classification predictions, it correctly classified 3362 samples in relation to the total number of 3600 samples, achieving 93.39% classification accuracy for classifying five sets of vibration data under 50% load condition. For other load conditions, see Appendix C. It can be said that the developed CNN-Three with the proposed IReLU-Tanh function to experimental vibration data yielded a robust classification with high diagnostic accuracy and outperformed the most widely used activation functions, ELU, LReLU, ReLU and Tanh.

As shown in Table 9-2 for baseline and Sun-F2 classes, it can be observed that there are four activation functions (IReLU-Tanh, ELU, LReLU, and ReLU) achieved almost the

same classification predictions. The lowest classification was obtained when using CNN-Three with Tanh function, in both classes it correctly classified 712 samples out of 720 samples belonging to baseline and Sun-F2 classes.

For Sun-F1 and Planet-F2 classes, it can be seen from Table 9-2 that the CNN-Three with the proposed IReLU-Tanh achieved the highest correct classification. It correctly classified 656 samples out of 720 samples as belonging to Sun-F1 class, and 608 samples correctly classified as belonging to Planet-F2 class. The second highest classification was obtained using ELU and ReLU, both functions correctly classified 601 samples out of 720 samples as belonging to Sun-F1 class, while for Planet-F2 class, 583 and 566 samples out of 720 samples were correctly classified using ELU and ReLU functions, respectively. The lowest classification was obtained when using CNN-Three with Tanh function, only 549 and 543 samples were correctly classified as belonging to Sun-F1 and Planet-F2 classes, respectively.

Table 9-2: Multi-class confusion matrix for different activation functions with 50% load

			Actual Class				
		Class	Baseline	Sun-F1	Sun-F2	Planet-F1	Planet-F2
Predicted Class	Baseline	Tanh	712	0	0	26	3
		ReLU	719	0	0	18	0
		LReLU	717	0	0	7	3
		ELU	718	0	0	11	0
		IReLU-Tanh	717	0	0	3	0
	Sun-F1	Tanh	1	549	2	7	145
		ReLU	0	601	2	6	112
		LReLU	0	595	1	5	106
		ELU	0	601	0	6	106
		IReLU-Tanh	0	656	0	4	83
	Sun-F2	Tanh	0	32	712	0	13
		ReLU	0	2	717	0	10
		LReLU	0	8	718	0	9
		ELU	0	2	717	0	4
		IReLU-Tanh	0	0	718	0	7
	Planet-F1	Tanh	0	10	0	616	16
		ReLU	0	14	0	650	32
		LReLU	1	20	0	673	37
		ELU	1	23	0	688	27
		IReLU-Tanh	3	13	0	663	22
	Planet-F2	Tanh	7	129	6	71	543
		ReLU	1	103	1	46	566
		LReLU	2	97	1	35	565
		ELU	1	94	3	15	583
		IReLU-Tanh	0	51	2	50	608
Ground Truth			720	720	720	720	720

As seen in Table 9-2 for Planet-F1 class, a highest classification was achieved using CNN-Three with ELU function, 688 samples were correctly classified as belonging to Planet-F1 class. The second and third highest classification were obtained when using LReLU and IReLU-Tanh functions, as they correctly classified 673 and 663 samples out of 720 samples as belonging to Planet-F1 class. The lowest classification was achieved when using CNN-Three with Tanh function, only 616 samples were correctly classified as belonging to Planet-F1 class, 26, 7 and 71 samples incorrectly classified as belonging to baseline, Sun-F1 and Planet-F2 respectively.

Based on the classification obtained from the multi-class confusion matrix in Table 9-2 and Appendix C, the average classification accuracy for the ten trials for Tanh, ReLU, LReLU and ELU functions under different load conditions are shown in Figure 9-3 to Figure *9-6*.

Figure 9-3 shows the classification accuracy obtained from applying the CNN-Three with Tanh function to classify five sets of vibration data under different load conditions. It can be seen that the CNN-Three with Tanh function achieved the lowest classification accuracy. For example, Table 9-2 shows that the CNN-Three with Tanh function correctly classified 3132 out of the total number of 3600 samples, achieving an overall 87.00% accuracy for classifying the baseline and the four fault datasets Sun-F1, Sun-F2, Planet-F1, and Planet-F2 under 50% load. For the other loads, the CNN-Three with Tanh function achieved 88.67%, 86.75%, 87.33%, and 87.06% classification accuracies for zero, 25%, 75%, and 90% load conditions, respectively.

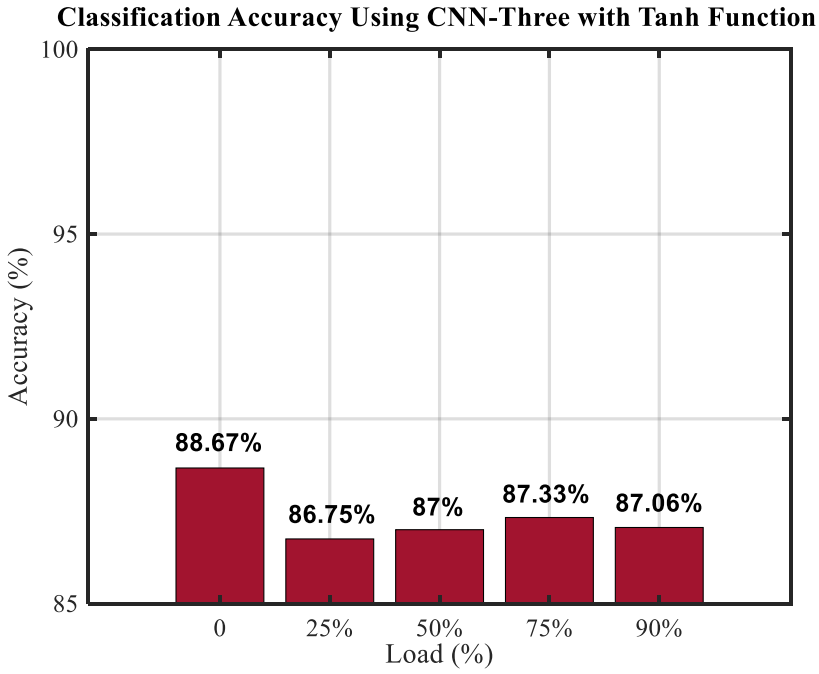

Figure 9-3: Classification accuracy for CNN-Three with Tanh function using experimental vibration data under different load conditions

Figure 9-4 andFigure *9-5* show the improvement obtained when using CNN-Three with ReLU and LReLU functions. Both functions achieved almost the same classification accuracy for classifying the five sets of vibration data under different load conditions. The average classification accuracy for both ReLU and LReLU functions is improved by

around 3% than CNN-Three with Tanh function. The CNN-Three with ReLU function achieved 90.92%, 90.69%, 90.36%, 90.75%, and 90.44% classification accuracies, whereas the LReLU function achieved 90.50%, 90.92%, 90.78%, 90.87%, and 90.94% classification accuracies for zero, 25% 50%, 75%, and 90% load conditions, respectively.

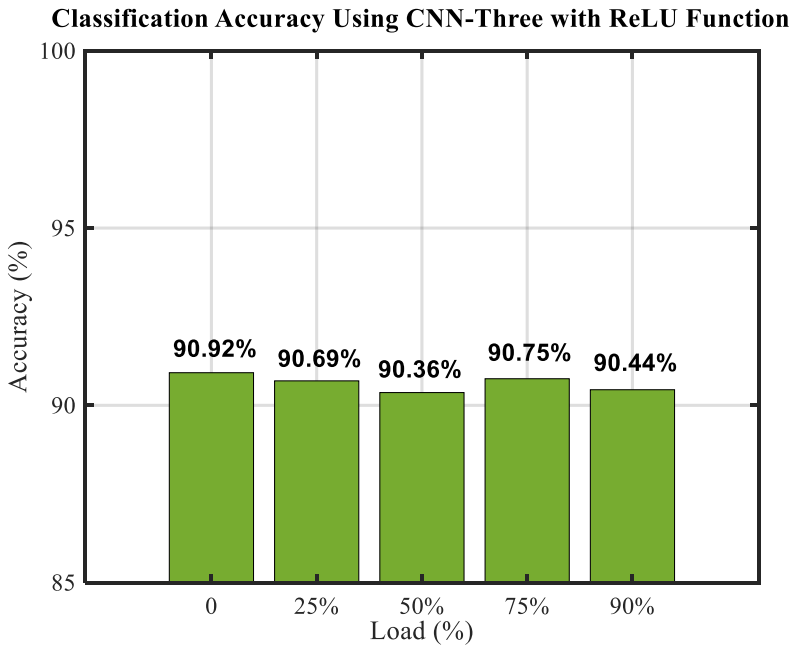

Figure 9-4: Classification accuracy for CNN-Three with the ReLU function using experimental vibration data under different load conditions

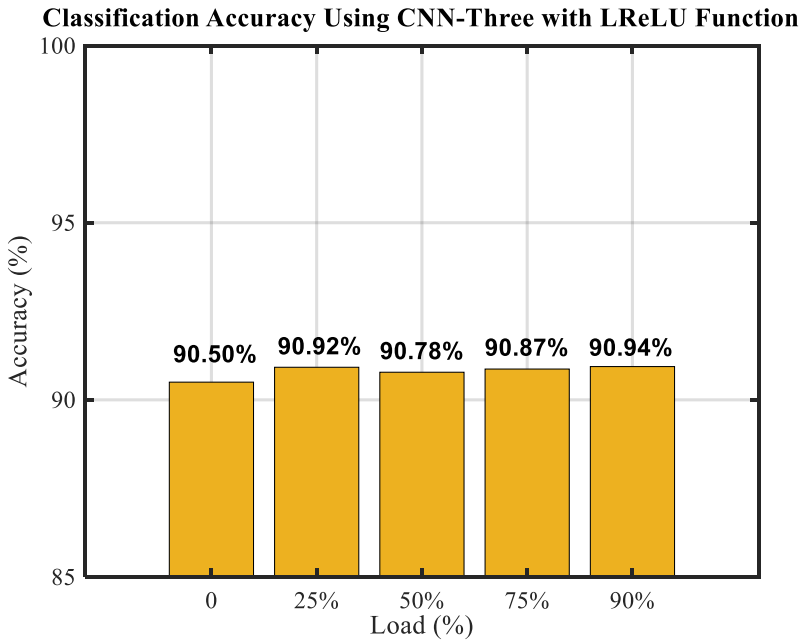

Figure 9-5: Classification accuracy for CNN-Three with the LReLU function using experimental vibration data under different load conditions

It can be seen in Figure 9-6 that a slight improvement of about 1% is achieved when using CNN-Three with the ELU function compared to the ReLU and LReLU functions. The CNN-Three with ELU function achieved the second highest classification, with an accuracy values of 91.78%, 91.42%, 91.86%, 92.33%, and 92.31% for zero, 25%, 50%, 75%, and 90% load conditions, respectively.

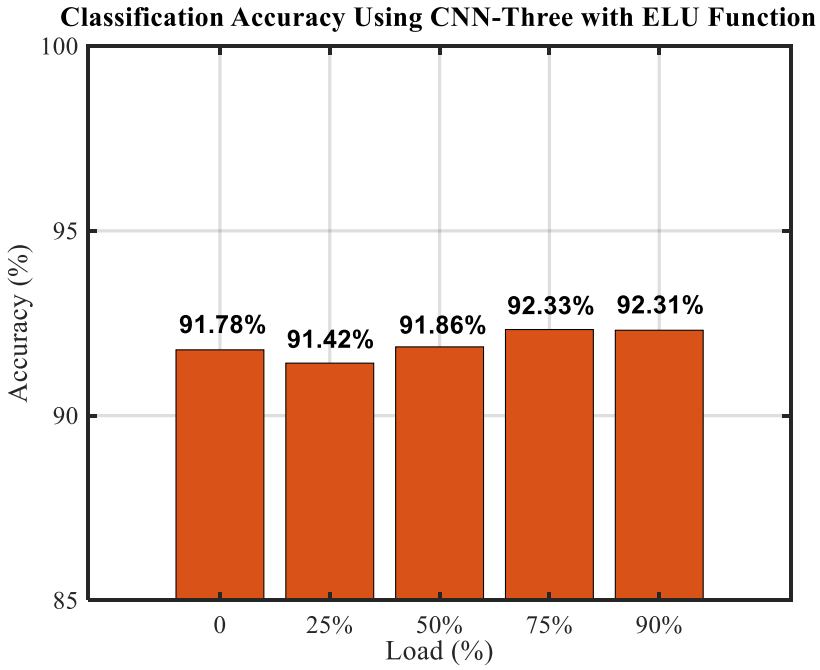

Figure 9-6: Classification accuracy for CNN-Three with ELU function using experimental vibration data under different load conditions

9.3. Results and Discussion

Based on the results reported in Section 9.2, the developed CNN-Three with IReLU-Tanh function has shown its effectiveness in minimising the error rate to the minimum value, and consequently improving the overall performance of the network with an average classification accuracy of more than 93% under five load conditions, as seen in Table 9-3. The developed CNN-Three with the proposed IReLU-Tanh function achieved the highest classification accuracy and outperformed the most widely used activation functions. It can be shown in Table 9-3 that the developed CNN-Three with the proposed IReLU-Tanh function achieved 93.03%, 94.14%, 93.39%, 94.00%, and 94.11% classification accuracies for zero, 25%, 50%, 75% and 90% loads, respectively.

Table 9-3: Average classification accuracies of ten trials for different activation functions under different load conditions

Function	Load				
	0	25%	50%	75%	90%
Tanh	88.67%	86.75%	87.00%	87.33%	87.06%
ReLU	90.92%	90.69%	90.36%	90.75%	90.44%
LReLU	90.50%	90.92%	90.78%	90.87%	90.94%
ELU	91.78%	91.42%	91.86%	92.33%	92.31%
IReLU-Tanh (Proposed)	**93.03%**	**94.14%**	**93.39%**	**94%**	**94.11%**

The proposed IReLU-Tanh function enhanced the ability of the network to learn the hidden features directly from raw vibration data, and consequently improved the overall performance of the developed CNN-Three architecture in classifying five sets of unseen vibration data. Section 8.5 explains the advantage of the proposed IRaLU-Tanh function, it addresses the vanishing gradient problem in the Tanh function, by adopting the non-saturation property from the ReLU function for covering the positive region. Section 8.5 also explains that for the negative region, the proposed IRaLU-Tanh function shares the property of Tanh function ($f(x) = tanh(x)$ for $x \leq 0$), and so addresses the shortcomings of dying neurons in the ReLU function, a fixed gradient value in the LReLU function and the need to add a hyper-parameter to the network architecture as in the LReLU and ELU functions.

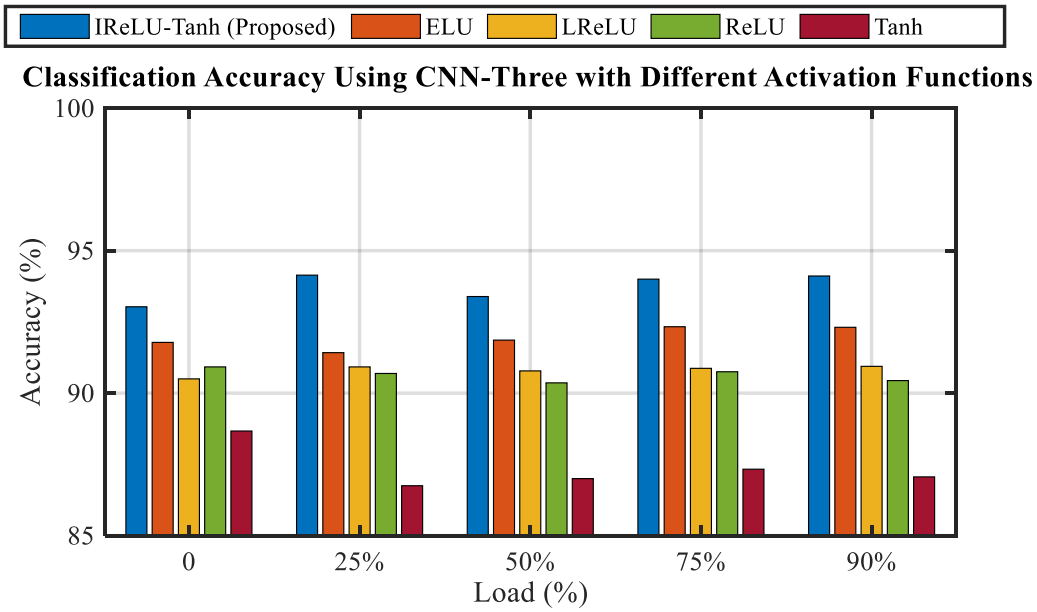

Figure 9-7: Comparison between the proposed IReLU-Tanh function and the existing activation functions using experimental vibration data obtained under different load conditions

Figure 9-7 compares the classification accuracy obtained from the proposed IReLU-Tanh function with the existing activation functions, Tanh, ReLU, LReLU and ELU. It can be clearly observed that a highest classification accuracy was achieved using the developed CNN-Three with the proposed IReLU-Tanh function. It has been demonstrated that when applying the developed CNN-Three architecture with the proposed IReLU-Tanh function to experimental vibration data, it yielded a robust classification with high diagnostic accuracy and outperformed the existing activation functions.

Chapter Ten: Conclusions and Future Work

This chapter summarises the research work and discusses how the aims and objectives of this research as stated in Chapter One were achieved. This is followed by the contributions to knowledge achieved by the research, and finally recommendations for future work are suggested at the end of the chapter.

10.1. Introduction

To conclude, the study of CM is usually conducted through the following three main approaches; application based, data gathering based and data processing based. In this thesis, the data processing based approach, including signal processing, data enhancement and AI methods was considered, and particularly an automated deep CNN approach was the main focus of this research. A hybrid IReLU-Tanh function has been proposed for the deep CNN architecture to address the shortcomings in existing activation functions and enhance the overall performance of the network. The aim of this research was met through achieving of the research aims and objectives as described in the next section.

10.2. Aims, Objectives and Achievements

The first aim of this research, to develop an automated approach based deep CNN for CM that would automate the tasks of feature extraction and classification has been achieved. The second aim of this research, to propose an activation function called improved rectified linear unit and hyperbolic tangent function (IReLU-Tanh) to enhance the learning ability of deep CNN architecture and improve the overall performance of the network has also been achieved. Both were extensively evaluated and successfully demonstrated as described below. The objectives presented in Chapter One and the achievements are detailed as follows:

Objective One: To review the existing data enhancement methods and automated approaches based on AI such as shallow and deep learning methods used for vibration data.

Achievement One: A comprehensive review was carried out covering existing data enhancement methods used for vibration data, in particular TSA, EMD, MED and MOMEDA. In addition, the application of AI based shallow learning methods for CM have been presented. AI based deep learning methods including SAE, RNN and CNN have been reviewed in Chapter Two.

Objective Two: To investigate and evaluate the existing activation functions used for deep CNN.

Achievement Two: The existing activation functions, Tanh, ReLU, LReLU, and ELU used for deep CNN have been reviewed. Moreover, common problems inherent in the use of these activation functions have been discussed individually in Chapter Five.

Objective Three: To develop and implement an optimal CNN architecture for feature extraction and classification based on typical CM vibration data.

Achievement Three: An automated approach based on a deep CNN algorithm was investigated in Chapter Four and successfully implemented to automate feature extraction and classification for CM using both simulated and experimental vibration data was demonstrated in Chapters Eight and Nine.

Objective Four: To develop an IReLU-Tanh function for deep CNN architecture.

Achievement Four: The proposed IReLU-Tanh function was developed in Chapter Five to address the shortcomings in the existing activation functions. The proposed IReLU-Tanh function was successfully implemented to both simulated and experimental vibration data in Chapters Eight and Nine.

Objective Five: To evaluate the performance of the developed CNN architecture with IReLU-Tanh function using simulated and experimental vibration data. This included a systematic comparison of the performance against the most widely used activation functions.

Achievement Five: The developed CNN-Three architecture with IReLU-Tanh function has been successfully applied to both simulated and experimental vibration data. It has been demonstrated that the developed CNN-Three with the proposed IReLU-Tanh function is a reliable and effective method in classifying several unseen datasets with high diagnostic accuracy. The proposed IReLU-Tanh function outperformed the most commonly used activation functions, Tanh, ReLU, LReLU and ELU.

10.3. Conclusions

Based on the theoretical and experimental studies achieved in this thesis, a number of conclusions have been drawn as follows:

Conclusion 1: Conventional signal processing methods, including time domain and frequency domain, have been investigated and implemented for fault detection and diagnosis of a PG. However, it was found from the results presented in Chapter Seven that these methods are limited to diagnose and identify the defect that has occurred in faulty cases.

Conclusion 2: Data enhancement based on the MOMEDA method was investigated and applied in this research for the purpose of enhancing periodic fault features in the vibration data. The results presented in Chapter Seven showed that the MOMEDA method was able to detect the presence of the large defects seeded into the Sun-F2 and Planet-F2. However, a small defects seeded into both the Sun-F1 and Planet-F1 were not readily detected and diagnosed. Also, the implementation of such technique rely entirely on prior knowledge and experience of well-skilled technical staff.

Conclusion 3: Automating the diagnostic process using AI based deep learning, and particularly the CNN method facilitates the tasks of feature extraction and classification for machinery CM. Automated fault diagnosis based on deep CNN method has been successfully applied to extract representative features directly from the raw data and automatically determine the PG health condition with high diagnostic accuracy.

Conclusion 4: An automated approach based on applying the developed CNN-Three architecture to simulated data showed that the developed architecture reliably learns the hidden features from the simulated datasets. The effectiveness of the developed CNN-Three architecture was compared with three recent architectures. The results presented in Chapter Eight showed that the developed CNN-Three architecture confirmed its improvement in classifying three sets of simulated data (small, medium, and large) with a classification accuracy improvement of 1% compared to the DCNN architecture, 2% improvement compared to the 1D-DCNN architecture, and 4% improvement compared to the CNN architecture.

Conclusion 5: The developed CNN-Three architecture with the proposed IReLU-Tanh function to simulated data achieved the highest classification accuracy and outperformed the most widely used activation functions, Tanh, ReLU, LReLU and ELU. As presented in Chapter Eight that the CNN-Three with the proposed IReLU-Tanh function achieved

99.91% classification accuracy for the simulated datasets with an SNR of (-1 dB), and 94.22% classification accuracy with an SNR level of (-5 dB).

Conclusion 6: The proposed IReLU-Tanh function has shown its effectiveness in minimising the error rate to the minimum value, and hence improving the overall performance of the network to obtain a higher classification accuracy, see Figure 8-15 and Figure 9-2.

Conclusion 7: The CNN-Three with the IReLU-Tanh function is an effective and reliable method for vibration data. It was found in the results presented in Chapter Nine that the CNN-Three architecture with the proposed IReLU-Tanh function can achieve more than 93% accuracy for classifying five sets of vibration data with different severities. It has been shown that the developed CNN-Three with the proposed IReLU-Tanh function to experimental vibration data achieved high diagnostic accuracy and outperformed the existing activation functions, Tanh, ReLU, LReLU and ELU.

10.4. Contribution to Knowledge

In this thesis, a number of contributions were obtained as follows:

First Contribution: A CNN architecture called CNN-Three has been developed that can effectively learn hidden features directly from raw vibration data. The developed CNN-Three architecture was shown to be highly effective in automating the feature extraction and classification tasks for machinery CM. It has been demonstrated that the developed CNN-Three architecture was robust in classifying different types of fault severities and outperformed three recent CNN architectures.

Second Contribution: A hybrid IReLU-Tanh function for deep CNN architecture has been proposed to address the shortcomings in existing activation functions and enhance the overall performance of deep CNN architecture. The proposed IReLU-Tanh function has shown an outstanding performance and outperforms the most widely used activation functions.

Third Contribution: The developed framework based CNN-Three architecture combined with the proposed IReLU-Tanh function is a novel and fully automated approach for CM of PGs. This fully automated approach can be applied to extract the

representative features directly from the raw data and automatically identify different classes for a given set of data. It has been shown that the fully automated approach yields robust classification with high diagnostic accuracy. The developed automated approach has been found to be a reliable and effective method for automating the diagnostic procedure and automatically determine the PG condition.

10.5. Recommendations for Future Work

- Deep CNN has a large number of hyper-parameters that need to be tuned, such as the number of layers, convolutional filter size, the number of convolutional filters, and so on. The selection of these hyper-parameters has a significant influence on the network training task and the classification accuracy of the model. In this research, these hyper-parameters were optimised using a trial and error strategy, and it was demonstrated in Chapter Eight that selecting different hyper-parameter values can affect the overall performance of the CNN. Optimisation algorithms could be developed to select the optimal hyper-parameter configuration for deep CNN.
- The developed CNN-Three in this research has been evaluated using simulated and experimental vibration data. However, the weights learned by the convolutional layer in both scenarios, simulated and experimental data, were not interpreted in detail. Further research can interpret and analyse the underlying meaning of these weights in association with the monitored data sources.
- The proposed IReLU-Tanh function has been successfully applied to simulated and experimental vibration data for machinery CM. However, such a proposed activation function could be used in other domains such as image processing, speech recognition, etc. The contribution of the IReLU-Tanh function generally to DNNs could be investigated.

References

1. Sheng, S., et al., *Wind turbine drivetrain condition monitoring during GRC phase 1 and phase 2 testing*. 2011, National Renewable Energy Lab.(NREL), Golden, CO (United States).
2. Staszewski, W.J. and K. Worden, *Signal processing for damage detection.* Encyclopedia of structural health monitoring, 2009.
3. Schneider, T., N. Helwig, and A. Schütze, *Automatic feature extraction and selection for classification of cyclical time series data.* 2017.
4. Zhang, W., et al., *Comprehensive overview on computational intelligence techniques for machinery condition monitoring and fault diagnosis.* Chinese Journal of Mechanical Engineering, 2017. **30**(4): p. 782-795.
5. Xu, K., et al., *A novel adaptive and fast deep convolutional neural network for bearing fault diagnosis under different working conditions.* Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2020. **234**(4): p. 1167-1182.
6. Xie, S., G. Ren, and J. Zhu, *Application of a new one-dimensional deep convolutional neural network for intelligent fault diagnosis of rolling bearings.* Science Progress, 2020. **103**(3): p. 0036850420951394.
7. Dargan, S., et al., *A survey of deep learning and its applications: A new paradigm to machine learning.* Archives of Computational Methods in Engineering, 2019: p. 1-22.
8. Gao, J., et al., *A survey on deep learning for multimodal data fusion.* Neural Computation, 2020. **32**(5): p. 829-864.
9. Bhadane, M. and K. Ramachandran. *Bearing fault identification and classification with convolutional neural network*. in *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. 2017. IEEE.
10. Sahu, M. and R. Dash, *A Survey on Deep Learning: Convolution Neural Network (CNN)*, in *Intelligent and Cloud Computing*. 2019, Springer. p. 317-325.
11. Ahlawat, S., et al., *Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN).* Sensors, 2020. **20**(12): p. 3344.
12. Namatēvs, I., *Deep convolutional neural networks: Structure, feature extraction and training.* Information Technology and Management Science, 2017. **20**(1): p. 40-47.
13. Szandała, T., *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks.* arXiv e-prints, 2020: p. arXiv: 2010.09458.
14. Dureja, A. and P. Pahwa, *Analysis of non-linear activation functions for classification tasks using convolutional neural networks.* Recent Patents on Computer Science, 2019. **12**(3): p. 156-161.
15. Jardine, A.K., D. Lin, and D. Banjevic, *A review on machinery diagnostics and prognostics implementing condition-based maintenance.* Mechanical systems and signal processing, 2006. **20**(7): p. 1483-1510.
16. Heng, A., et al., *Rotating machinery prognostics: State of the art, challenges and opportunities.* Mechanical systems and signal processing, 2009. **23**(3): p. 724-739.
17. Zhao, X., *Data-driven fault detection, isolation and identification of rotating machinery: With applications to pumps and gearboxes.* 2012.
18. HONGTAO, X., *Study on Intelligent Condition Diagnosis Based on Vibration Information and Support Vector Machine for Plant Machinery.* 2014.
19. Goyal, D. and B. Pabla, *Condition based maintenance of machine tools—A review.* CIRP Journal of Manufacturing Science and Technology, 2015. **10**: p. 24-35.
20. Van Tung, T. and B.-S. Yang, *Machine fault diagnosis and prognosis: The state of the art.* International Journal of Fluid Machinery and Systems, 2009. **2**(1): p. 61-71.

21. Heyns, T., *Low cost condition monitoring under time-varying operating conditions*. 2013, University of Pretoria.
22. Jia, F., et al., *Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data.* Mechanical Systems and Signal Processing, 2016. **72**: p. 303-315.
23. Ince, T., et al., *Real-time motor fault detection by 1-D convolutional neural networks.* IEEE Transactions on Industrial Electronics, 2016. **63**(11): p. 7067-7075.
24. Dragomir, O.E., et al. *Review of prognostic problem in condition-based maintenance*. in *2009 European Control Conference (ECC)*. 2009. IEEE.
25. Lei, Y., et al., *Condition monitoring and fault diagnosis of planetary gearboxes: A review.* Measurement, 2014. **48**: p. 292-305.
26. Miljković, D., *Brief review of vibration based machine condition monitoring.* HDKBR INFO Magazin, 2015. **5**(2): p. 14-23.
27. Zhu, J., et al. *Survey of condition indicators for condition monitoring systems*. in *Annu. Conf. Progn. Heal. Manag. Soc*. 2014.
28. Forrester, B.D., *Advanced vibration analysis techniques for fault detection and diagnosis in geared transmission systems*. 1996, Swinburne University of Technology.
29. Aherwar, A., *An investigation on gearbox fault detection using vibration analysis techniques: A review.* Australian Journal of Mechanical Engineering, 2012. **10**(2): p. 169-183.
30. Večeř, P., M. Kreidl, and R. Šmíd, *Condition indicators for gearbox condition monitoring systems.* Acta Polytechnica, 2005. **45**(6).
31. Xu, Q., et al., *Imbalanced fault diagnosis of rotating machinery via multi-domain feature extraction and cost-sensitive learning.* Journal of Intelligent Manufacturing, 2019: p. 1-15.
32. Aherwar, A. and M.S. Khalid, *Vibration analysis techniques for gearbox diagnostic: a review.* International Journal of Advanced Engineering Technology, 2012. **3**(2): p. 04-12.
33. Riaz, S., et al., *Vibration feature extraction and analysis for fault diagnosis of rotating machinery-a literature survey.* Asia Pacific Journal of Multidisciplinary Research, 2017. **5**(1): p. 103-110.
34. Shao, R., W. Hu, and J. Li, *Multi-fault feature extraction and diagnosis of gear transmission system using time-frequency analysis and wavelet threshold de-noising based on EMD.* Shock and Vibration, 2013. **20**(4): p. 763-780.
35. Forrester, B., *Method for the separation of epicyclic planet gear vibration signatures*. 2001, US Patent US6,298,725.
36. Bechhoefer, E., M. Kingsley, and P. Menon. *Bearing envelope analysis window selection using spectral kurtosis techniques*. in *2011 IEEE Conference on Prognostics and Health Management*. 2011. IEEE.
37. Ilhan Asilturk, H.A., Ugur Ozmen, *MACHINERY MONITORING USING VIBRATION SIGNAL ANALYSIS.* 2017.
38. Zhong, J. and Y. Huang, *Time-frequency representation based on an adaptive short-time Fourier transform.* IEEE Transactions on Signal Processing, 2010. **58**(10): p. 5118-5128.
39. Zhang, X., L. Wang, and Q. Miao. *Fault diagnosis techniques for planetary gearboxes under variable conditions: A review*. in *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*. 2016. IEEE.
40. Li, S., et al. *A Review on the Signal Processing Methods of Rotating Machinery Fault Diagnosis*. in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2019. IEEE.
41. Turing, A.M., *Computing machinery and intelligence*, in *Parsing the turing test*. 2009, Springer. p. 23-65.

42. Reis, D. and N. Pati, *Applications of artificial intelligence to condition-based maintenance.* RAE-Revista de Administração de Empresas, 2000. **40**(2): p. 102-107.
43. Liu, R., et al., *Artificial intelligence for fault diagnosis of rotating machinery: A review.* Mechanical Systems and Signal Processing, 2018. **108**: p. 33-47.
44. Jia, F., et al., *Deep normalized convolutional neural network for imbalanced fault classification of machinery and its understanding via visualization.* Mechanical Systems and Signal Processing, 2018. **110**: p. 349-367.
45. He, J., S. Yang, and C. Gan, *Unsupervised fault diagnosis of a gear transmission chain using a deep belief network.* Sensors, 2017. **17**(7): p. 1564.
46. Kolar, D., et al., *Fault Diagnosis of Rotary Machines Using Deep Convolutional Neural Network with Wide Three Axis Vibration Signal Input.* Sensors, 2020. **20**(14): p. 4017.
47. Jaber, A.A. and R. Bicker, *The state of the art in research into the condition monitoring of industrial machinery.* Int. J. of Current Engineering and Technology, 2014. **4**(3): p. 1986-2001.
48. Alaskar, H., *Deep learning-based model architecture for time-frequency images analysis.* International Journal of Advanced Computer Science and Applications, 2018. **9**(12).
49. Zhang, S., et al., *Deep Learning Algorithms for Bearing Fault Diagnosticsx—A Comprehensive Review.* IEEE Access, 2020. **8**: p. 29857-29881.
50. Lei, Y., et al., *An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data.* IEEE Transactions on Industrial Electronics, 2016. **63**(5): p. 3137-3147.
51. Fink, O., et al., *Potential, challenges and future directions for deep learning in prognostics and health management applications.* Engineering Applications of Artificial Intelligence, 2020. **92**: p. 103678.
52. Alom, M.Z., et al., *The history began from alexnet: A comprehensive survey on deep learning approaches.* arXiv preprint arXiv:1803.01164, 2018.
53. Stetco, A., et al., *Machine learning methods for wind turbine condition monitoring: A review.* Renewable energy, 2019. **133**: p. 620-635.
54. Awad, M. and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. 2015: Springer Nature.
55. Wuest, T., et al., *Machine learning in manufacturing: advantages, challenges, and applications.* Production & Manufacturing Research, 2016. **4**(1): p. 23-45.
56. Neelamegam, S. and E. Ramaraj, *Classification algorithm in data mining: An overview.* International Journal of P2P Network Trends and Technology (IJPTT), 2013. **4**(8): p. 369-374.
57. de Carvalho, A.C. and A.A. Freitas, *A tutorial on multi-label classification techniques*, in *Foundations of computational intelligence volume 5*. 2009, Springer. p. 177-195.
58. Er, M.J., R. Venkatesan, and N. Wang. *An online universal classifier for binary, multi-class and multi-label classification*. in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2016. IEEE.
59. Nguyen, C.D., A. Prosvirin, and J.-M. Kim, *A Reliable Fault Diagnosis Method for a Gearbox System with Varying Rotational Speeds.* Sensors, 2020. **20**(11): p. 3105.
60. Yuan, Y., et al., *A general end-to-end diagnosis framework for manufacturing systems.* National Science Review, 2020. **7**(2): p. 418-429.
61. Ramanan, A., S. Suppharangsan, and M. Niranjan. *Unbalanced decision trees for multi-class classification*. in *2007 International Conference on Industrial and Information Systems*. 2007. IEEE.
62. Tiwari, R., et al., *Multi-class fault diagnosis in gears using machine learning algorithms based on time domain data.* International journal of COMADEM, 2017. **20**(1).
63. Sun, W., et al., *An intelligent gear fault diagnosis methodology using a complex wavelet enhanced convolutional neural network.* Materials, 2017. **10**(7): p. 790.

64. Chen, Z., et al. *Gearbox Fault Diagnosis Using Convolutional Neural Networks And Support Vector Machines*. in *2019 27th European Signal Processing Conference (EUSIPCO)*. 2019. IEEE.
65. Ide, H. and T. Kurita. *Improvement of learning for CNN with ReLU activation by sparse regularization*. in *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017. IEEE.
66. McFadden, P., *A technique for calculating the time domain averages of the vibration of the individual planet gears and the sun gear in an epicyclic gearbox.* Journal of Sound and vibration, 1991. **144**(1): p. 163-172.
67. Randall, R.B., J.J.M.s. Antoni, and s. processing, *Rolling element bearing diagnostics—A tutorial.* 2011. **25**(2): p. 485-520.
68. Fan, Q., et al., *Gear tooth surface damage diagnosis based on analyzing the vibration signal of an individual gear tooth.* Advances in Mechanical Engineering, 2017. **9**(6): p. 1687814017704356.
69. Zuber, N., R. Bajrić, and R. Šostakov, *Gearbox faults identification using vibration signal analysis and artificial intelligence methods.* Eksploatacja i Niezawodność, 2014. **16**(1): p. 61--65.
70. Huang, N.E., et al. *The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis*. in *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*. 1998. The Royal Society.
71. Junsheng, C., Y. Dejie, and Y. Yu, *Research on the intrinsic mode function (IMF) criterion in EMD method.* Mechanical systems and signal processing, 2006. **20**(4): p. 817-824.
72. Dybała, J. and R. Zimroz, *Rolling bearing diagnosing method based on empirical mode decomposition of machine vibration signal.* Applied Acoustics, 2014. **77**: p. 195-203.
73. Huang, N.E., et al., *The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis.* Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences, 1998. **454**(1971): p. 903-995.
74. Wiggins, R.A., *Minimum entropy deconvolution.* Geoexploration, 1978. **16**(1-2): p. 21-35.
75. Barszcz, T. and N. Sawalhi, *Fault detection enhancement in rolling element bearings using the minimum entropy deconvolution.* Archives of acoustics, 2012. **37**(2): p. 131-141.
76. Kedadouche, M., M. Thomas, and A. Tahan, *Monitoring machines by using a hybrid method combining MED, EMD, and TKEO.* Advances in Acoustics and Vibration, 2014. **2014**.
77. Endo, H. and R. Randall, *Enhancement of autoregressive model based gear tooth fault detection technique by the use of minimum entropy deconvolution filter.* Mechanical Systems and Signal Processing, 2007. **21**(2): p. 906-919.
78. Li, Q., X. Ji, and S.Y. Liang, *Incipient fault feature extraction for rotating machinery based on improved AR-minimum entropy deconvolution combined with variational mode decomposition approach.* Entropy, 2017. **19**(7): p. 317.
79. Endo, H., R. Randall, and C. Gosselin, *Differential diagnosis of spall vs. cracks in the gear tooth fillet region: Experimental validation.* Mechanical Systems and Signal Processing, 2009. **23**(3): p. 636-651.
80. He, L., et al. *Application of minimum entropy deconvolution on enhancement of gear tooth fault detection*. in *Prognostics and System Health Management Conference (PHM-Harbin), 2017*. 2017. IEEE.
81. Zhao, L., et al., *The incipient fault feature enhancement method of the gear box based on the wavelet packet and the minimum entropy deconvolution.* Systems Science & Control Engineering, 2018. **6**(3): p. 235-241.

82. McDonald, G.L. and Q. Zhao, *Multipoint optimal minimum entropy deconvolution and convolution fix: Application to vibration fault detection.* Mechanical Systems and Signal Processing, 2017. **82**: p. 461-477.
83. Miao, Y., et al., *Application of an improved maximum correlated kurtosis deconvolution method for fault diagnosis of rolling element bearings.* Mechanical Systems and Signal Processing, 2017. **92**: p. 173-195.
84. Alqatawneh, I., et al. *Condition Monitoring and Fault Diagnosis Based on Multipoint Optimal Minimum Entropy Deconvolution Adjusted Technique*. in *2018 24th International Conference on Automation and Computing (ICAC)*. 2018. IEEE.
85. Wang, Z., et al., *A novel method for multi-fault feature extraction of a gearbox under strong background noise.* Entropy, 2018. **20**(1): p. 10.
86. Barszcz, T. and N. Sawalhi, *Wind turbines' rolling element bearings fault detection enhancement using minimum entropy deconvolution.* Diagnostyka, 2011: p. 53-59.
87. Nose-Filho, K., C. Jutten, and J.M. Romano. *Sparse blind deconvolution based on scale invariant smoothed ℓ 0-norm*. in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*. 2014. IEEE.
88. Cai, W. and Z. Wang, *Application of an improved multipoint optimal minimum entropy deconvolution adjusted for gearbox composite fault diagnosis.* Sensors, 2018. **18**(9): p. 2861.
89. Cai, W., et al., *A new compound fault feature extraction method based on multipoint kurtosis and variational mode decomposition.* Entropy, 2018. **20**(7): p. 521.
90. Azimi, M., A.D. Eslamlou, and G. Pekcan, *Data-Driven Structural Health Monitoring and Damage Detection through Deep Learning: State-of-the-Art Review.* Sensors, 2020. **20**(10): p. 2778.
91. Hurwitz, J. and D. Kirsch, *Machine learning for dummies.* IBM Limited Edition, 2018. **75**.
92. Jing, L., et al., *A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox.* Measurement, 2017. **111**: p. 1-10.
93. You, W., et al., *An Intelligent Deep Feature Learning Method With Improved Activation Functions for Machine Fault Diagnosis.* IEEE Access, 2019. **8**: p. 1975-1985.
94. Pantula, S.R., *Automated fault diagnosis in rotating machinery*. 2014, University of Waterloo.
95. Peng, D., et al., *A novel deeper one-dimensional CNN with residual learning for fault diagnosis of wheelset bearings in high-speed trains.* IEEE Access, 2018. **7**: p. 10278-10293.
96. Çınar, Z.M., et al., *Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0.* Sustainability, 2020. **12**(19): p. 8211.
97. Chakraverty, S., D.M. Sahoo, and N.R. Mahato, *Concepts of soft computing: fuzzy and ANN with programming*. 2019: Springer.
98. Wasukar, A.R., *Artificial neural network–an important asset for future computing.* International Journal For Research In Emerging Science And Technology, 2014. **1**.
99. Cilimkovic, M., *Neural networks and back propagation algorithm.* Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 2015. **15**.
100. Bui, N.T. and H. Hasegawa, *Training artificial neural network using modification of differential evolution algorithm.* International journal of Machine Learning and computing, 2015. **5**(1): p. 1.
101. Chung, H., S.J. Lee, and J.G. Park. *Deep neural network using trainable activation functions*. in *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016. IEEE.
102. Vapnik, V., *The nature of statistical learning theory*. 2013: Springer science & business media.

103. Cortes, C. and V. Vapnik, *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.
104. Madzarov, G. and D. Gjorgjevikj. *Multi-class classification using support vector machines in decision tree architecture*. in *IEEE EUROCON 2009*. 2009. IEEE.
105. Widodo, A. and B.-S. Yang, *Support vector machine in machine condition monitoring and fault diagnosis.* Mechanical systems and signal processing, 2007. **21**(6): p. 2560-2574.
106. Bellary, J. and K.R. Eddula, *Improving multi-class support vector machines training.* Int J Comput Commun Instrum Eng, 2014. **1**: p. 119-125.
107. Zhang, C., et al., *A gearbox fault diagnosis method based on frequency-modulated empirical mode decomposition and support vector machine.* Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2018. **232**(2): p. 369-380.
108. Chao, L., C. Lu, and J. Ma, *An approach to fault diagnosis for gearbox based on reconstructed energy and support vector machine.* Vibroengineering PROCEDIA, 2017. **14**: p. 136-140.
109. Kang, J., et al. *Gearbox fault diagnosis method based on wavelet packet analysis and support vector machine*. in *Proceedings of the IEEE 2012 Prognostics and System Health Management Conference (PHM-2012 Beijing)*. 2012. IEEE.
110. Cover, T. and P. Hart, *Nearest neighbor pattern classification.* IEEE transactions on information theory, 1967. **13**(1): p. 21-27.
111. Alsharif, M.H., et al., *Machine learning algorithms for smart data analysis in internet of things environment: taxonomies and research trends.* Symmetry, 2020. **12**(1): p. 88.
112. Quinlan, J.R., *Induction of decision trees.* Machine learning, 1986. **1**(1): p. 81-106.
113. Song, Y.-Y. and L. Ying, *Decision tree methods: applications for classification and prediction.* Shanghai archives of psychiatry, 2015. **27**(2): p. 130.
114. Al-Rajab, M.M.J., *EFFICIENT ALGORITHMS FOR CANCER GENE SEARCHING AND CLASSIFICATION: COLON CANCER*. 2019, University of Huddersfield.
115. Guolian, H., et al. *Research on fault diagnosis of wind turbine control system based on Artificial Neural Network*. in *2010 8th World Congress on Intelligent Control and Automation*. 2010. IEEE.
116. Yang, Z., W.I. Hoi, and J. Zhong. *Gearbox fault diagnosis based on artificial neural network and genetic algorithms*. in *Proceedings 2011 International Conference on System Science and Engineering*. 2011. IEEE.
117. Khazaee, M., et al., *An appropriate approach for condition monitoring of planetary gearbox based on fast Fourier transform and least-square support vector machine.* International Journal of Multidisciplinary Sciences and Engineering, 2012. **3**(5): p. 22-26.
118. Praveenkumar, T., et al., *Fault diagnosis of automobile gearbox based on machine learning techniques.* Procedia Engineering, 2014. **97**: p. 2092-2098.
119. Yang, D., et al., *Gear fault diagnosis based on support vector machine optimized by artificial bee colony algorithm.* Mechanism and Machine Theory, 2015. **90**: p. 219-229.
120. Heidari, M., et al., *Fault diagnosis of gearboxes using wavelet support vector machine, least square support vector machine and wavelet packet transform.* Journal of Vibroengineering, 2016. **18**(2): p. 860-875.
121. Montaña, M.M. and M.A. Sanz-Bobi. *Anomaly Detection Indicators of a Wind Turbine Gearbox Based on Feature Extraction from its Vibration Performance*. in *PHM Society European Conference*. 2018.
122. Wang, Z., J. Wang, and Y. Wang, *An intelligent diagnosis scheme based on generative adversarial learning deep neural networks and its application to planetary gearbox fault pattern recognition.* Neurocomputing, 2018. **310**: p. 213-222.
123. Wang, J., et al., *Performance analysis and enhancement of deep convolutional neural network.* Business & Information Systems Engineering, 2019. **61**(3): p. 311-326.

124. Liu, R., et al., *Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine.* IEEE Transactions on Industrial Informatics, 2016. **13**(3): p. 1310-1320.
125. Jia, F., et al., *A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines.* Neurocomputing, 2018. **272**: p. 619-628.
126. Wang, Y., et al., *A Novel Bearing Fault Diagnosis Methodology Based on SVD and One-Dimensional Convolutional Neural Network.* Shock and Vibration, 2020. **2020**.
127. Deng, L., *A tutorial survey of architectures, algorithms, and applications for deep learning.* APSIPA Transactions on Signal and Information Processing, 2014. **3**.
128. Gong, W., et al., *A novel deep learning method for intelligent fault diagnosis of rotating machinery based on improved CNN-SVM and multichannel data fusion.* Sensors, 2019. **19**(7): p. 1693.
129. Hinton, G.E., S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets.* Neural computation, 2006. **18**(7): p. 1527-1554.
130. Guo, C., et al., *A Deep Learning Based Fault Diagnosis Method With Hyperparameter Optimization by Using Parallel Computing.* IEEE Access, 2020. **8**: p. 131248-131256.
131. Wang, T., et al., *Fault diagnosis of rotating machinery under time-varying speed based on order tracking and deep learning.* Journal of Vibroengineering, 2020. **22**(2): p. 366-382.
132. Zhao, M., et al., *Deep residual networks with dynamically weighted wavelet coefficients for fault diagnosis of planetary gearboxes.* IEEE Transactions on Industrial Electronics, 2017. **65**(5): p. 4290-4300.
133. Le, Q.V., *A tutorial on deep learning part 1: Nonlinear classifiers and the backpropagation algorithm.* Google Inc., Mountain View, CA, 2015: p. 18.
134. Toh, G. and J. Park, *Review of vibration-based structural health monitoring using deep learning.* Applied Sciences, 2020. **10**(5): p. 1680.
135. Le, Q.V., *A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks.* Google Brain, 2015: p. 1-20.
136. Shrestha, A. and A. Mahmood, *Review of deep learning algorithms and architectures.* IEEE Access, 2019. **7**: p. 53040-53065.
137. Schilling, F., *The effect of batch normalization on deep convolutional neural networks*. 2016.
138. Sze, V., et al., *Efficient processing of deep neural networks.* Synthesis Lectures on Computer Architecture, 2020. **15**(2): p. 1-341.
139. Xueyi, L., et al., *Semi-supervised gear fault diagnosis using raw vibration signal based on deep learning.* Chinese Journal of Aeronautics, 2020. **33**(2): p. 418-426.
140. Chen, Z., et al., *Vibration-based gearbox fault diagnosis using deep neural networks.* Journal of Vibroengineering, 2017. **19**(4): p. 2475-2496.
141. Yin, A., et al., *Fault Diagnosis of Wind Turbine Gearbox Based on the Optimized LSTM Neural Network with Cosine Loss.* Sensors, 2020. **20**(8): p. 2339.
142. Chen, Z., C. Li, and R.-V. Sánchez, *Multi-layer neural network with deep belief network for gearbox fault diagnosis.* Journal of Vibroengineering, 2015. **17**(5): p. 2379-2392.
143. Alom, M.Z., et al., *A state-of-the-art survey on deep learning theory and architectures.* Electronics, 2019. **8**(3): p. 292.
144. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors.* nature, 1986. **323**(6088): p. 533-536.
145. Hoang, D.-T. and H.-J. Kang, *A survey on deep learning based bearing fault diagnosis.* Neurocomputing, 2019. **335**: p. 327-335.
146. Guo, G. and N. Zhang, *A survey on deep learning based face recognition.* Computer Vision and Image Understanding, 2019. **189**: p. 102805.

147. Liu, G., H. Bao, and B. Han, *A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis.* Mathematical Problems in Engineering, 2018. **2018**.
148. Shao, H., et al., *A novel deep autoencoder feature learning method for rotating machinery fault diagnosis.* Mechanical Systems and Signal Processing, 2017. **95**: p. 187-204.
149. Williams, R.J. and D. Zipser, *A learning algorithm for continually running fully recurrent neural networks.* Neural computation, 1989. **1**(2): p. 270-280.
150. Yu, Y., et al., *A review of recurrent neural networks: LSTM cells and network architectures.* Neural computation, 2019. **31**(7): p. 1235-1270.
151. Sun, W., et al., *Fault detection and identification using Bayesian recurrent neural networks.* Computers & Chemical Engineering, 2020. **141**: p. 106991.
152. LeCun, Y., et al., *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
153. Li, H., et al., *Fault Diagnosis for Rotating Machinery Using Multiscale Permutation Entropy and Convolutional Neural Networks.* Entropy, 2020. **22**(8): p. 851.
154. González-Muñiz, A., I. Díaz, and A.A. Cuadrado, *DCNN for condition monitoring and fault detection in rotating machines and its contribution to the understanding of machine nature.* Heliyon, 2020. **6**(2): p. e03395.
155. Liu, R., et al., *Multiscale Kernel Based Residual Convolutional Neural Network for Motor Fault Diagnosis Under Nonstationary Conditions.* IEEE Transactions on Industrial Informatics, 2019. **16**(6): p. 3797-3806.
156. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* nature, 2015. **521**(7553): p. 436-444.
157. Goodfellow, I., Y. Bengio, and A. Courville, *Deep learning*. 2016: MIT press.
158. Jiang, G., et al., *Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox.* IEEE Transactions on Industrial Electronics, 2018. **66**(4): p. 3196-3207.
159. Grezmak, J., et al., *Explainable convolutional neural network for gearbox fault diagnosis.* Procedia CIRP, 2019. **80**: p. 476-481.
160. Zhang, W., et al., *A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load.* Mechanical Systems and Signal Processing, 2018. **100**: p. 439-453.
161. Chen, Z., C. Li, and R.-V. Sanchez, *Gearbox fault identification and classification with convolutional neural networks.* Shock and Vibration, 2015. **2015**.
162. Liu, C., et al., *Planetary gears feature extraction and fault diagnosis method based on VMD and CNN.* Sensors, 2018. **18**(5): p. 1523.
163. Park, P., et al., *Fault detection and diagnosis using combined autoencoder and long short-term memory network.* Sensors, 2019. **19**(21): p. 4612.
164. Zhang, L., et al., *A review on deep learning applications in prognostics and health management.* IEEE Access, 2019. **7**: p. 162415-162438.
165. Zhao, R., et al., *Deep learning and its applications to machine health monitoring.* Mechanical Systems and Signal Processing, 2019. **115**: p. 213-237.
166. Lin, M.-C., et al., *Development of Compound Fault Diagnosis System for Gearbox Based on Convolutional Neural Network.* Sensors, 2020. **20**(21): p. 6169.
167. Ding, B., H. Qian, and J. Zhou. *Activation functions and their characteristics in deep neural networks*. in *2018 Chinese Control And Decision Conference (CCDC)*. 2018. IEEE.
168. Lee, J., et al., *ProbAct: A Probabilistic Activation Function for Deep Neural Networks.* arXiv preprint arXiv:1905.10761, 2019.
169. Clevert, D.-A., T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus).* arXiv preprint arXiv:1511.07289, 2015.
170. Nair, V. and G.E. Hinton. *Rectified linear units improve restricted boltzmann machines*. in *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.

171. Maas, A.L., A.Y. Hannun, and A.Y. Ng. *Rectifier nonlinearities improve neural network acoustic models*. in *Proc. icml*. 2013.
172. Feng, K., et al., *A diagnostic signal selection scheme for planetary gearbox vibration monitoring under non-stationary operational conditions.* Measurement Science and Technology, 2017. **28**(3): p. 035003.
173. GU, X., *Influence of planet position errors and eccentricities on planetary gear dynamics.* INSA de Lyon, Diss, 2012.
174. Lei, Y., et al., *Fault detection of planetary gearboxes using new diagnostic parameters.* Measurement Science and Technology, 2012. **23**(5): p. 055605.
175. Zhang, X., L. Wang, and Q. Miao. *Fault diagnosis techniques for planetary gearboxes under variable conditions: A review*. in *Prognostics and System Health Management Conference (PHM-Chengdu), 2016*. 2016. IEEE.
176. Inalpolat, M. and A. Kahraman, *A theoretical and experimental investigation of modulation sidebands of planetary gear sets.* Journal of Sound and Vibration, 2009. **323**(3): p. 677-696.
177. Blunt, D.M. and J.A. Keller, *Detection of a fatigue crack in a UH-60A planet gear carrier using vibration analysis.* Mechanical Systems and Signal Processing, 2006. **20**(8): p. 2095-2111.
178. Lei, Y., et al., *A method based on multi-sensor data fusion for fault detection of planetary gearboxes.* Sensors, 2012. **12**(2): p. 2005-2017.
179. Mones, Z., et al. *Planetary gearbox fault diagnosis using an on-rotor MEMS accelerometer*. in *2017 23rd International Conference on Automation and Computing (ICAC)*. 2017. IEEE.
180. Wen, W., R.X. Gao, and W. Cheng, *Planetary gearbox fault diagnosis using envelope manifold demodulation.* Shock and Vibration, 2016. **2016**.
181. Yip, L., *Analysis and modeling of planetary gearbox vibration data for early fault detection*. 2011, University of Toronto.
182. Alban, L.E., *Systematic analysis of gear failures*. 1985: ASM International.
183. Glew, T.C. *Failure analysis and repair techniques for turbomachinery gears*. in *Proceedings of the 9th Turbomachinery Symposium*. 1980. Texas A&M University. Gas Turbine Laboratories.
184. Jotram Patel, G.S., and P.K. Sen,, *A study on common failure of gears.* 2015.
185. Errichello, R., *Friction, lubrication, and wear of gears.* Materials Park, OH: ASM International, 1992., 1992: p. 535-545.
186. Shipley, E.E., *Gear failures.* Mach Design, 1967. **39**(28): p. 152-162.
187. DAHAL, P. *Introduction to Convolutional Neural Networks*. [Accessed 20 April 2020]; Available from: https://deepnotes.io/intro.
188. Albelwi, S. and A. Mahmood, *A framework for designing the architectures of deep convolutional neural networks.* Entropy, 2017. **19**(6): p. 242.
189. Gu, J., et al., *Recent advances in convolutional neural networks.* Pattern Recognition, 2018. **77**: p. 354-377.
190. Bacanin, N., et al., *Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics.* Algorithms, 2020. **13**(3): p. 67.
191. Triantis, D., *Functionally weighted convolutional neural networks.* 2017.
192. Bjorck, N., et al. *Understanding batch normalization*. in *Advances in Neural Information Processing Systems*. 2018.
193. Ioffe, S. and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* arXiv preprint arXiv:1502.03167, 2015.
194. Wu, S., et al., *L1-Norm Batch Normalization for Efficient Training of Deep Neural Networks.* IEEE transactions on neural networks and learning systems, 2018. **30**(7): p. 2043-2051.

195. Karlik, B. and A.V. Olgac, *Performance analysis of various activation functions in generalized MLP architectures of neural networks.* International Journal of Artificial Intelligence and Expert Systems, 2011. **1**(4): p. 111-122.
196. Nanni, L., et al., *Stochastic Activation Function Layers for Convolutional Neural Networks.* 2020.
197. Nwankpa, C., et al., *Activation functions: Comparison of trends in practice and research for deep learning.* arXiv preprint arXiv:1811.03378, 2018.
198. Zhang, W., et al., *A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals.* Sensors, 2017. **17**(2): p. 425.
199. Liu, X., et al., *Fault diagnosis of rotating machinery under noisy environment conditions based on a 1-D convolutional autoencoder and 1-D convolutional neural network.* Sensors, 2019. **19**(4): p. 972.
200. Scherer, D., A. Müller, and S. Behnke. *Evaluation of pooling operations in convolutional architectures for object recognition*. in *International conference on artificial neural networks*. 2010. Springer.
201. Boureau, Y.-L., J. Ponce, and Y. LeCun. *A theoretical analysis of feature pooling in visual recognition*. in *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.
202. Nagi, J., et al. *Max-pooling convolutional neural networks for vision-based hand gesture recognition*. in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. 2011. IEEE.
203. Chen, H., et al., *A deep convolutional neural network based fusion method of two-direction vibration signal data for health state identification of planetary gearboxes.* Measurement, 2019. **146**: p. 268-278.
204. Jiao, J., et al., *A multivariate encoder information based convolutional neural network for intelligent fault diagnosis of planetary gearboxes.* Knowledge-Based Systems, 2018. **160**: p. 237-250.
205. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
206. Beale, M.H., M.T. Hagan, and H.B. Demuth. *Neural network toolbox™ user's guide*. The MathWorks 2010.
207. Yamashita, R., et al., *Convolutional neural networks: an overview and application in radiology.* Insights into imaging, 2018. **9**(4): p. 611-629.
208. Hinton, G., et al., *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.* IEEE Signal processing magazine, 2012. **29**(6): p. 82-97.
209. Wang, C., et al. *Depth Learning Standard Deviation Loss Function*. in *Journal of Physics: Conference Series*. 2019. IOP Publishing.
210. Shahsavarani, S., *Speech Emotion Recognition using Convolutional Neural Networks.* 2018.
211. Jian, X., et al., *Fault diagnosis of motor bearings based on a one-dimensional fusion neural network.* Sensors, 2019. **19**(1): p. 122.
212. Tabian, I., H. Fu, and Z. Sharif Khodaei, *A convolutional neural network for impact detection and characterization of complex composite structures.* Sensors, 2019. **19**(22): p. 4933.
213. Vogl, T.P., et al., *Accelerating the convergence of the back-propagation method.* Biological cybernetics, 1988. **59**(4-5): p. 257-263.
214. Tawfique, Z., *Tool-Mediated Texture Recognition Using Convolutional Neural Network*. 2016.

215. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.
216. Bottou, L., *Stochastic gradient learning in neural networks.* Proceedings of Neuro-Nımes, 1991. **91**(8): p. 12.
217. Zhang, J., *Gradient descent based optimization algorithms for deep learning models training.* arXiv preprint arXiv:1903.03614, 2019.
218. Jansson, P., *Single-word speech recognition with Convolutional Neural Networks on raw waveforms.* 2018.
219. Wu, Y., et al., *Demystifying Learning Rate Polices for High Accuracy Training of Deep Neural Networks.* arXiv preprint arXiv:1908.06477, 2019.
220. Kriesel, D., *A brief introduction on neural networks.* 2007.
221. Heaton, J., *Introduction to neural networks with Java*. 2008: Heaton Research, Inc.
222. Wilson, D.R. and T.R. Martinez. *The need for small learning rates on large problems*. in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*. 2001. IEEE.
223. Rakhecha, A. *Understanding Learning Rate*. 2019 [Accessed 22 February 2020]; Available from: https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de.
224. Raschka, S., *Model evaluation, model selection, and algorithm selection in machine learning.* arXiv preprint arXiv:1811.12808, 2018.
225. Gabralla, L.A., H. Mahersia, and A. Abraham. *Ensemble neurocomputing based oil price prediction*. in *Afro-European Conference for Industrial Advancement*. 2015. Springer.
226. McCaffery, J. *Classification and Prediction Using Neural Networks*. 2012 [Accessed 20 March 2020]; Available from: https://docs.microsoft.com/en-us/archive/msdn-magazine/2012/july/test-run-classification-and-prediction-using-neural-networks.
227. Ying, X. *An Overview of Overfitting and its Solutions*. in *Journal of Physics: Conference Series*. 2019. IOP Publishing.
228. Onwujekwegn, G. and V.Y. Yoon, *Analyzing the Impacts of Activation Functions on the Performance of Convolutional Neural Network Models.* 2020.
229. Nguyen, A., et al., *An Analysis of State-of-the-art Activation Functions For Supervised Deep Neural Network.* arXiv preprint arXiv:2104.02523, 2021.
230. Rahman, M.S., *Computations, optimization and tuning of deep feedforward neural networks.* bioRxiv, 2019.
231. Glorot, X. and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.
232. Datta, L., *A Survey on Activation Functions and their relation with Xavier and He Normal Initialization.* arXiv preprint arXiv:2004.06632, 2020.
233. Zheng, Q., et al., *Rethinking the Role of Activation Functions in Deep Convolutional Neural Networks for Image Classification.* Engineering Letters, 2020. **28**(1).
234. Epelbaum, T., *Deep learning: Technical introduction.* arXiv preprint arXiv:1709.01412, 2017.
235. Gulcehre, C., et al. *Noisy activation functions*. in *International conference on machine learning*. 2016.
236. You, W., et al., *An Intelligent Deep Feature Learning Method with Improved Activation Functions for Machine Fault Diagnosis.* IEEE Access, 2019.
237. Maguolo, G., L. Nanni, and S. Ghidoni, *Ensemble of Convolutional Neural Networks Trained with Different Activation Functions.* arXiv preprint arXiv:1905.02473, 2019.
238. Xu, B., et al., *Empirical evaluation of rectified activations in convolutional network.* arXiv preprint arXiv:1505.00853, 2015.

239. Pedamonti, D., *Comparison of non-linear activation functions for deep neural networks on MNIST classification task.* arXiv preprint arXiv:1804.02763, 2018.
240. Lu, L., et al., *Dying relu and initialization: Theory and numerical examples.* arXiv preprint arXiv:1903.06733, 2019.
241. Yang, J. and G. Yang, *Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer.* Algorithms, 2018. **11**(3): p. 28.
242. Feng, J. and S. Lu. *Performance Analysis of Various Activation Functions in Artificial Neural Networks*. in *Journal of Physics: Conference Series*. 2019. IOP Publishing.
243. Hsiao, J., K. Shivam, and T. Kam. *Fault diagnosis method for worm gearbox using convolutional network and ensemble learning*. in *Journal of Physics: Conference Series*. 2020.
244. HANSEN, C. *Activation Functions Explained - GELU, SELU, ELU, ReLU and more*. 2019 [Accessed 20 February 2020]; Available from: https://mlfromscratch.com/activation-functions-explained/#/.
245. Singh, P., M. Varshney, and V.P. Namboodiri, *Cooperative initialization based deep neural network training.* arXiv preprint arXiv:2001.01240, 2020.
246. Manessi, F. and A. Rozza. *Learning combinations of activation functions*. in *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018. IEEE.
247. Li, J., et al., *A domain adaptation model for early gear pitting fault diagnosis based on deep transfer learning network.* Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2020. **234**(1): p. 168-182.
248. Trottier, L., P. Gigu, and B. Chaib-draa. *Parametric exponential linear unit for deep convolutional neural networks*. in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2017. IEEE.
249. Wu, C., et al., *Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network.* Computers in Industry, 2019. **108**: p. 53-61.
250. Han, Y., B. Tang, and L. Deng, *An enhanced convolutional neural network with enlarged receptive fields for fault diagnosis of planetary gearboxes.* Computers in Industry, 2019. **107**: p. 50-58.
251. He, Z., et al., *Ensemble transfer CNNs driven by multi-channel signals for fault diagnosis of rotating machinery cross working conditions.* Knowledge-Based Systems, 2020: p. 106396.
252. Cao, J., et al., *An Anti-Noise Fault Diagnosis Method of Bearing based on Multi-Scale 1DCNN.* 2020.
253. Feng, Z. and M.J. Zuo, *Vibration signal models for fault diagnosis of planetary gearboxes.* Journal of Sound and Vibration, 2012. **331**(22): p. 4919-4939.
254. Szegedy, C., et al. *Going deeper with convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
255. He, K. and J. Sun. *Convolutional neural networks at constrained time cost*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
256. Zhang, B., et al., *Intelligent fault diagnosis under varying working conditions based on domain adaptive convolutional neural networks.* IEEE Access, 2018. **6**: p. 66367-66384.
257. Sadoughi, M., et al., *A deep learning-based approach for fault diagnosis of roller element bearings.* 2018.
258. Huang, S., et al., *Signal status recognition based on 1DCNN and its feature extraction mechanism analysis.* Sensors, 2019. **19**(9): p. 2018.
259. Guo, X., L. Chen, and C. Shen, *Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis.* Measurement, 2016. **93**: p. 490-502.
260. James, G., et al., *An introduction to statistical learning*. Vol. 112. 2013: Springer.
261. Farhadi, F., *Learning activation functions in deep neural networks*. 2017, École Polytechnique de Montréal.

262. Tharwat, A., *Classification assessment methods.* Applied Computing and Informatics, 2020.
263. Hossin, M. and M. Sulaiman, *A review on evaluation metrics for data classification evaluations.* International Journal of Data Mining & Knowledge Management Process, 2015. **5**(2): p. 1.

Appendices

Appendix A: Multi-Class Confusion Matrix Results for CNN-Three and Three Different CNN Architectures Using Simulated Data

Table A-1: Multi-class confusion matrix for CNN-Three, DCNN, 1D-DCNN and CNN using simulated data with SNR (-1 dB)

			Actual Class		
Predicted Class		Class	Small	Medium	Large
	Small	CNN	1161	10	0
		1D-DCNN	1184	12	0
		DCNN	1200	2	0
		CNN-Three	1195	0	0
	Medium	CNN	39	1120	18
		1D-DCNN	16	1150	14
		DCNN	0	1144	0
		CNN-Three	5	1180	0
	Large	CNN	0	70	1182
		1D-DCNN	0	38	1186
		DCNN	0	54	1200
		CNN-Three	0	20	1200
Ground Truth			1200	1200	1200

Table A-2: Multi-class confusion matrix for CNN-Three, DCNN, 1D-DCNN and CNN using simulated data with SNR (-2 dB)

			Actual Class		
	Class		Small	Medium	Large
Predicted Class	Small	CNN	1100	17	0
		1D-DCNN	1121	56	0
		DCNN	1179	7	0
		CNN-Three	1176	0	0
	Medium	CNN	100	1046	24
		1D-DCNN	79	1017	0
		DCNN	21	971	0
		CNN-Three	24	1079	0
	Large	CNN	0	137	1176
		1D-DCNN	0	127	1200
		DCNN	0	222	1200
		CNN-Three	0	121	1200
Ground Truth			1200	1200	1200

Table A-3: Multi-class confusion matrix for CNN-Three, DCNN, 1D-DCNN and CNN using simulated data with SNR (-4 dB)

			Actual Class		
	Class		Small	Medium	Large
Predicted Class	Small	CNN	943	36	0
		1D-DCNN	1052	46	0
		DCNN	1132	9	0
		CNN-Three	1064	8	0
	Medium	CNN	255	943	65
		1D-DCNN	148	842	3
		DCNN	68	767	0
		CNN-Three	135	885	0
	Large	CNN	2	221	1135
		1D-DCNN	0	211	1197
		DCNN	0	424	1200
		CNN-Three	1	307	1200
Ground Truth			1200	1200	1200

Table A-4: Multi-class confusion matrix for CNN-Three, DCNN, 1D-DCNN and CNN using simulated data with SNR (-5 dB)

			Actual Class		
Predicted Class	Class		Small	Medium	Large
	Small	CNN	810	31	0
		1D-DCNN	1053	82	2
		DCNN	1149	46	0
		CNN-Three	1034	4	0
	Medium	CNN	388	799	32
		1D-DCNN	146	732	9
		DCNN	51	686	0
		CNN-Three	166	831	1
	Large	CNN	2	370	1168
		1D-DCNN	1	386	1189
		DCNN	0	468	1200
		CNN-Three	0	365	1199
Ground Truth			1200	1200	1200

Appendix B: Multi-Class Confusion Matrix Results for Different Activation Functions Using Simulated Data

Table B-1: Multi-class confusion matrix for different activation functions using simulated data with SNR (-1 dB)

			Actual Class		
Predicted Class	Class		Small	Medium	Large
	Small	Tanh	1200	13	0
		ReLU	1195	0	0
		LReLU	1195	0	0
		ELU	1197	0	0
		IReLU-Tanh	1200	1	0
	Medium	Tanh	0	1181	48
		ReLU	5	1180	0
		LReLU	5	1184	0
		ELU	3	1197	0
		IReLU-Tanh	0	1197	0
	Large	Tanh	0	6	1152
		ReLU	0	20	1200
		LReLU	0	16	1200
		ELU	0	3	1200
		IReLU-Tanh	0	2	1200
Ground Truth			1200	1200	1200

Table B-2: Multi-class confusion matrix for different activation functions using simulated data with SNR (-2 dB)

			Actual Class		
Predicted Class	Class		Small	Medium	Large
	Small	Tanh	1182	38	0
		ReLU	1176	0	0
		LReLU	1168	0	0
		ELU	1182	1	0
		IReLU-Tanh	1190	3	0
	Medium	Tanh	18	1125	100
		ReLU	24	1079	0
		LReLU	32	1092	0
		ELU	18	1176	0
		IReLU-Tanh	10	1183	0
	Large	Tanh	0	37	1100
		ReLU	0	121	1200
		LReLU	0	108	1200
		ELU	0	23	1200
		IReLU-Tanh	0	14	1200
Ground Truth			1200	1200	1200

Table B-3: Multi-class confusion matrix for different activation functions using simulated data with SNR (-4 dB)

			Actual Class		
Predicted Class	Class		Small	Medium	Large
	Small	Tanh	1168	119	0
		ReLU	1064	8	0
		LReLU	1098	3	0
		ELU	1146	39	0
		IReLU-Tanh	1162	22	0
	Medium	Tanh	32	1053	341
		ReLU	135	885	0
		LReLU	102	926	0
		ELU	54	1109	32
		IReLU-Tanh	38	1130	10
	Large	Tanh	0	28	859
		ReLU	1	307	1200
		LReLU	0	271	1200
		ELU	0	52	1168
		IReLU-Tanh	0	48	1190
Ground Truth			1200	1200	1200

Table B-4: Multi-class confusion matrix for different activation functions using simulated data with SNR (-5 dB)

			Actual Class		
	Class		Small	Medium	Large
Predicted Class	Small	Tanh	1174	182	0
		ReLU	1034	4	0
		LReLU	1081	2	0
		ELU	1138	66	0
		IReLU-Tanh	1150	57	0
	Medium	Tanh	26	975	332
		ReLU	166	831	1
		LReLU	119	828	0
		ELU	62	1058	50
		IReLU-Tanh	50	1087	45
	Large	Tanh	0	43	868
		ReLU	0	365	1199
		LReLU	0	370	1200
		ELU	0	76	1150
		IReLU-Tanh	0	56	1155
Ground Truth			1200	1200	1200

Appendix C: Multi-Class Confusion Matrix Results for Different Activation Functions Using Experimental Vibration Data

Table C-1: Multi-class confusion matrix for different activation functions using experimental vibration data with zero load

			Actual Class				
		Class	Baseline	Sun-F1	Sun-F2	Planet-F1	Planet-F2
Predicted Class	Baseline	Tanh	657	0	0	40	28
		ReLU	648	0	0	19	2
		LReLU	640	0	0	25	0
		ELU	653	0	0	26	3
		IReLU-Tanh	674	0	0	24	11
	Sun-F1	Tanh	0	632	30	1	60
		ReLU	1	615	35	5	42
		LReLU	0	623	36	3	45
		ELU	0	647	10	6	38
		IReLU-Tanh	0	669	14	4	51
	Sun-F2	Tanh	0	31	680	0	15
		ReLU	0	40	677	0	12
		LReLU	0	26	677	0	14
		ELU	0	9	700	0	12
		IReLU-Tanh	0	5	703	0	12
	Planet-F1	Tanh	24	0	0	653	47
		ReLU	38	3	0	693	24
		LReLU	47	3	0	688	31
		ELU	40	0	0	676	39
		IReLU-Tanh	22	0	0	680	23
	Planet-F2	Tanh	39	57	10	26	570
		ReLU	33	62	8	3	640
		LReLU	33	68	7	4	630
		ELU	27	64	10	12	628
		IReLU-Tanh	24	46	3	12	623
Ground Truth			720	720	720	720	720

Table C-2: Multi-class confusion matrix for different activation functions using experimental vibration data with 25% load

			Actual Class				
		Class	Baseline	Sun-F1	Sun-F2	Planet-F1	Planet-F2
Predicted Class	Baseline	Tanh	702	0	0	35	1
		ReLU	714	0	0	20	0
		LReLU	718	0	0	9	1
		ELU	711	0	0	23	0
		IReLU-Tanh	717	0	0	20	1
	Sun-F1	Tanh	0	546	36	2	99
		ReLU	0	620	22	2	68
		LReLU	0	601	28	4	76
		ELU	0	639	21	3	63
		IReLU-Tanh	0	652	8	2	57
	Sun-F2	Tanh	0	62	674	1	12
		ReLU	0	16	685	0	9
		LReLU	0	28	680	0	5
		ELU	0	7	683	0	12
		IReLU-Tanh	0	11	701	0	7
	Planet-F1	Tanh	10	1	0	648	55
		ReLU	6	0	0	630	27
		LReLU	2	2	0	686	50
		ELU	9	0	0	642	29
		IReLU-Tanh	3	0	0	683	19
	Planet-F2	Tanh	8	111	10	34	553
		ReLU	0	84	13	68	616
		LReLU	0	89	12	21	588
		ELU	0	74	16	52	616
		IReLU-Tanh	0	57	11	15	636
Ground Truth			720	720	720	720	720

Table C-3: Multi-class confusion matrix for different activation functions using experimental vibration data with 75% load

			Actual Class				
		Class	Baseline	Sun-F1	Sun-F2	Planet-F1	Planet-F2
Predicted Class	Baseline	Tanh	714	0	0	41	1
		ReLU	707	0	0	19	3
		LReLU	690	0	0	16	1
		ELU	709	0	0	30	3
		IReLU-Tanh	694	0	0	17	1
	Sun-F1	Tanh	0	618	87	9	88
		ReLU	0	640	54	0	51
		LReLU	0	652	62	3	61
		ELU	0	661	22	2	52
		IReLU-Tanh	0	683	20	0	45
	Sun-F2	Tanh	0	60	628	0	16
		ReLU	0	41	665	0	0
		LReLU	0	36	657	0	1
		ELU	0	29	697	0	2
		IReLU-Tanh	0	11	699	0	3
	Planet-F1	Tanh	6	1	0	609	40
		ReLU	11	1	0	659	70
		LReLU	24	0	0	666	51
		ELU	8	1	0	659	65
		IReLU-Tanh	23	1	0	673	36
	Planet-F2	Tanh	0	41	5	61	575
		ReLU	2	38	1	42	596
		LReLU	6	32	1	35	607
		ELU	3	29	1	29	598
		IReLU-Tanh	3	25	1	30	635
Ground Truth			720	720	720	720	720

Table C-4: Multi-class confusion matrix for different activation functions using experimental vibration data with 90% load

			Actual Class				
		Class	Baseline	Sun-F1	Sun-F2	Planet-F1	Planet-F2
Predicted Class	Baseline	Tanh	688	0	0	29	3
		ReLU	685	0	0	13	9
		LReLU	696	0	0	17	5
		ELU	688	0	0	20	2
		IReLU-Tanh	703	0	0	14	0
	Sun-F1	Tanh	0	557	58	12	74
		ReLU	0	608	60	6	62
		LReLU	0	613	47	11	61
		ELU	0	646	33	8	41
		IReLU-Tanh	0	642	26	8	23
	Sun-F2	Tanh	0	77	647	7	14
		ReLU	0	42	657	1	7
		LReLU	0	39	673	0	7
		ELU	0	24	681	3	15
		IReLU-Tanh	0	20	692	0	5
	Planet-F1	Tanh	22	8	2	640	27
		ReLU	26	13	0	678	14
		LReLU	17	12	0	660	15
		ELU	31	9	0	664	18
		IReLU-Tanh	17	16	0	671	12
	Planet-F2	Tanh	10	78	13	32	602
		ReLU	9	57	3	22	628
		LReLU	7	56	0	32	632
		ELU	1	41	6	25	644
		IReLU-Tanh	0	62	2	27	680
Ground Truth			720	720	720	720	720