FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

AN INDUSTRIAL INTERNET OF THINGS APPLICATION FOR REAL-TIME CONDITION MONITORING

Aydil Jumaa BAPIR

Master's Thesis

DEPARTMENT OF COMPUTER ENGINEERING

JANUARY 2022

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

Department of Computer Engineering

Master's Thesis

AN INDUSTRIAL INTERNET OF THINGS APPLICATION FOR REAL-TIME CONDITION MONITORING

Author
Aydil Jumaa BAPIR

Supervisor
Assoc. Prof. İlhan AYDIN

JANUARY 2022
ELAZIG

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

Department of Computer Engineering

Master's Thesis

Title:	An Industrial Internet of Things Application for Real-Time Condition Monitoring
Author:	Aydil Jumaa BAPIR
Submission Date:	02 January 2022
Defense Date:	25 January 2022

THESIS APPROVAL

This thesis, which was prepared according to the thesis writing rules of the Graduate School of Natural and Applied Sciences, Fırat University, was evaluated by the committee members who have signed the following signatures and was unanimously approved after the defense exam made open to the academic audience.

	Signature	
Supervisor:	Assoc. Prof. İlhan AYDIN Firat University, Faculty of Engineering	Approved
Chair:	Asst. Prof. Dr. Mehmet BAYĞIN Ardahan University, Faculty of Engineering	Approved
Member:	Asst. Prof. Dr. Gülşah KARADUMAN Firat University, Faculty of Engineering	Approved

This thesis was approved by the Administrative Board of the Graduate School on

..... / ..... / 20 .....

Signature

Prof. Dr. Kürşat Esat ALYAMAÇ
Director of the Graduate School

DECLARATION

I hereby declare that I wrote this Master's Thesis titled “An Industrial Internet of Things Application for Real-Time Condition Monitoring” in consistent with the thesis writing guide of the Graduate School of Natural and Applied Sciences, Firat University. I also declare that all information in it is correct, that I acted according to scientific ethics in producing and presenting the findings, cited all the references I used, express all institutions or organizations or persons who supported the thesis financially. I have never used the data and information I provide here in order to get a degree in any way.

25 January 2022

Aydil Jumaa BAPIR

PREFACE

All devices are subject to faults. In fact, a joint study found that unplanned downtime costs an estimated 50 billion per year for industrial producers. The reason of 42 percent of this unplanned downtime is equipment faults.

Monitoring the machine condition, predictive maintenance, and frequent machinery inspection helped to detect problems at an early stage. In order to mitigate breakdowns and reduce sudden stops, predictive analytics allow us to forecast future results using past data.

All of this may sound a little overwhelming because analytics is not a key feature in a production setup, but there is support from businesses that provide Internet of Things technology solutions and Happiest Minds that provide its engineering services. With this study, we will help manufacturers concentrate on their core activities and not think too much about things like asset usage and equipment uptime.

I would like to thank my teacher and seminar advisor, my supervisor Assoc. Prof. İlhan AYDIN at the computer engineering department for his invaluable support and guidance. He ensured that work on this study was my own while steering me in the right direction as and when I needed it.

Aydil Jumaa BAPIR
ELAZIG, 2022

TABLE OF CONTENTS

Page

ABSTRACT

An Industrial Internet of Things Application for Real-Time Condition Monitoring

Aydil Jumaa BAPIR

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Computer Engineering

January 2022, Page: xiii + 75

In recent years, the production model has become widespread around the world. Smart technology and modern automation tools are increasingly changing the face of production and industry. The use of innovative maintenance techniques contributes to production by reducing disruptions and sudden stops in industry. The achievable production costs by reducing downtime and increasing maintenance performance is the most critical task. It can achieve these goals with the widespread use of low-cost smart devices that can provide a comprehensive online view of equipment operating conditions. For these reasons, an Internet of Things-based method is proposed for monitoring induction motors in this thesis. These devices analyze the vibration signals of the motor to provide conditional and continuous control of the tolerance of the motor after a fault.

As a result of artificial intelligence, production and algorithms, it will be possible to remotely monitor the state of the motor using the methods of remote clouds of the Internet of things. The sound or vibration sensors that evaluate the stability of the machine help to reduce breakdowns and faults by issuing warnings and predicting whether the machine is stable, by informing monitoring centers before any fault occurs to the motor. For this purpose, it will be possible to measure vibration signals from the induction motors and make condition analysis remotely. The proposed approach is based on measuring vibration signals with a remote sensing kit and identifying faults based on artificial intelligence. The approach consists of two methods. The first method is based on the detection of shaft imbalances based on machine learning with vibration signals measured with the kit. For this purpose, the performance of different machine learning methods was compared. The second approach is to evaluate the diagnostic performance of three different one-dimensional convolutional neural networks on motor vibration signals. Obtained results show that methods can be used to detect motor faults in real environment and multiple motors can be easily monitored remotely.

Keywords: Induction motors, bearing faults, VMD, CWRU, Machine Learning, Deep learning, Steval ST-WINKT1, IoT, DWT, FFT, CNN, LSTM, CapsuleNet.

ÖZET

Gerçek Zamanlı Durum İzleme İçin Endüstriyel Nesnelerin İnterneti Uygulaması

Aydil Jumaa BAPIR

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Ocak 2022, Sayfa: xiii + 75

Son yıllarda, üretim modeli dünya çapında yaygınlaşmıştır. Akıllı teknoloji ve modern otomasyon araçları, üretim ve endüstrinin çehresini giderek daha fazla değiştirmektedir. Yenilikçi bakım tekniklerinin kullanılması, kuruluşlardaki aksama ve ani duruşları azaltarak üretime katkıda bulunur. Arıza süresini azaltarak ve bakım performansını artırarak gerçekleştirilebilecek üretim maliyetlerini azaltmak en kritik görevdir. Ekipmanın çalışma koşullarının kapsamlı ve çevrimiçi bir görünümünü sunabilen düşük maliyetli akıllı cihazların yaygın kullanımı ile bu hedeflere ulaşabilir. Bu nedenlerle, bu tezde asenkron motorların izlenmesi için Nesnelerin İnterneti tabanlı bir yöntem önerilmiştir. Bu cihazlar, bir arıza sonrasında motorun toleransının koşullu ve sürekli kontrolünü sağlamak için motorun titreşim sinyallerini analiz eder.

Yapay zeka, üretim ve algoritmaların bir sonucu olarak, nesnelerin internetinin uzak bulutları yöntemlerini kullanarak motorun durumunu uzaktan izlemek mümkün olacaktır. Makinenin kararlılığını değerlendiren ses veya titreşim sensörleri, uyarılar vermenin ve makinenin kararlı olup olmadığını tahmin etmenin yanı sıra, izleme merkezlerini makinede herhangi bir hasar oluşmadan önce bilgilendirerek arıza ve arızaların azaltılmasına yardımcı olur. Bu amaçla asenkron motorlardan titreşim sinyallerinin ölçülmesi ve durum analizinin uzaktan yapılabilmesi sağlanacaktır. Önerilen yaklaşım bir uzaktan algılama kiti ile titreşim sinyallerinin ölçülmesi ve yapay zeka temelli arızaların belirlenmesine dayalıdır. Önerilen yaklaşım iki yöntemden oluşmaktadır. İlk yöntem kit ile ölçülen titreşim sinyalleri ile makine öğrenmesi tabanlı mil dengesizliklerinin tespitine dayalıdır. Bu amaçla farklı makine öğrenmesi yöntemlerinin performansı karşılaştırılmıştır. İkinci yaklaşım ise motor titreşim sinyalleri üzerinde üç farklı tek boyutlu evrişimsel sinir ağları ile arıza teşhis performansının değerlendirilmesidir. Elde edilen sonuçlar gerçek ortamda motor arızlarının belirlenmesi için yöntemlerin kullanılabileceğini ve uzaktan çoklu motorların kolayca izlenebileceğini göstermektedir.

Anahtar Kelimeler: Induction motors, bearing faults, VMD, CWRU, Machine Learning, Deep learning, Steval ST-WINKT1, IOT, DWT, FFT, CNN, LSTM, CapsuleNet.

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF APPENDICES

Page

SYMBOLS AND ABBREVIATIONS

Abbreviations

CWRU	: Case Western University Database
CNN	: Convolutional Neural Network
CM	: Condition Monitoring
1D CNN	: one-dimensional convolutional neural network
DWT	: Discrete Wavelet Transform
FFT	: Fast Fourier Transform
IoT	: Internet of Things
LSTM	: Long-Short Term Memory
RNN	: Recurrent Neural Network
SVM	: Support Vector Machine
TF	: TensorFlow Platform
WPD	: Wavelet Packet Decomposition

1. INTRODUCTION

1.1. Overview

The induction motors are workhorses of the industrial sector, are exposed to a range of improper stressors throughout their operating lives, resulting in flaws and setbacks [1]. A catastrophic fault of a motor in these devices may be quite costly. Due to the widespread use of automation and a reduction in computer-interface communication, finding an efficient and accurate fault diagnosis method is becoming more complex. Fault diagnosis for induction motors is thus complex due to different operating speeds and load conditions. Neural Networks, for example, is a kind of data-based model that has been more popular in electrical equipment condition monitoring during the last decade. Decades have passed since the first suggestion of using vibration, tracking to locate spinning unit faults. Rotor system health may be determined using vibration testing, which is considered the most accurate method [2],[3]. Each malfunction in a rotating system generates vibration distinct characteristics compared to reference values. Standard vibration signal analysis approaches, such as the Fast Fourier Transform, are ineffective for examining signals having a transient nature. Evaluations are also largely based on the load on the system and the identification of the high frequency of multi-link faults. A high level of resolution is required for constituents [4]. Transient signals may be tested using Wavelet, a scalable signal processing method, without depending on load. It is possible to get both low-frequency and high-frequency information using the variable window size[5],[6].

1.2. Problem Statement

Predictive maintenance has been utilized by a wide range of industries, including power plants, infrastructure, transportation frameworks, connection protocols, and disaster services[7]. For the most part, and in a broader context, it implies historical knowledge of the system to optimize support operations that anticipate faults or failures in a broken-down system by providing an appraisal of the system's condition [8].

Predictive maintenance programs can identify early indicators of the problem or failure and then support measures may be provided at the right moment. Diagnostic and prognostic data are both provided through predictive maintenance [9].

1- Predictive maintenance information has made it possible and efficient to tell what is wrong, where the error occurs, whether it signals a mistake or a malfunction, and why it occurs.

2- In addition to being utilized for repairs, predictive maintenance data may also be used for other reasons, such as predicting the operating duration of a fuel cell [10].

3- Determining the quality of second-hand goods and the wear on cutting equipment.
4- As far back as the 1940s, predictive maintenance has been in use, but in its simplest form, where an experienced support person visits the preface and uses their senses of seeing and hearing and noticing and touching to detect an early symptom of a problem, is now widely used because it is still very effective in a variety of conditions.

Sensors for the duties of monitoring, listening, sensing, and touching are now accessible at the machine level rather than at the component or device level. Using sensors and wireless network systems, predictive maintenance gathers data on device status in the most efficient manner possible. Continuous acquisition and analysis of data is required to determine the most appropriate moment to intervene in the key maintenance processes such as staffing and maintaining component order. A multi-disciplinary approach is required for predictive maintenance tasks, including methods for developing and handling. Procurement of records, processing of information, and making decisions are the three essential components of maintenance. There is now research into these three subjects and specific techniques of predictive maintenance and necessary machinery/components to be kept in mind for predictive maintenance purposes.

Accordingly, this study explains how predictive maintenance works from a technological and administrative perspective by first defining predictive maintenance and then highlighting some critical points that must be considered for the successful implementation of predictive maintenance. Condition monitoring and repair of expensive and complex equipment are required when these machines are needed to perform vital company tasks. For example, in today's global economy, manufacturing organizations are doing all they can to reduce costs and improve product efficiency in order to remain competitive [11]. An essential part of the industrial chain, rotating equipment has a considerable influence on production schedules, quality, and cost of manufacture. Machine fault may be caused by unanticipated malfunctions, collisions, and injuries. Maintenance costs can account for anywhere from 15 to 60 percent of a product's total cost of production of food-related manufacturing. In comparison, maintenance costs can account for as much as 60 percent of the total cost of production in iron and steel and other heavy industries.

1.3. Aims of the Thesis

In this thesis, the vibration of an induction motor was analyzed to acquire exact information that may be utilized to anticipate motor bearing faults. An induction motor carrying fault detection method has been tried. Machine learning techniques, in addition to wavelet transform (WT) and fast Fourier transform (FFT), an advanced signal processing approach, are employed in this study to investigate frame vibrations during initialization. IoT is the basis of today's fast technological progress. A vast number of items are networked effectively, especially in industrial-automation,

resulting in a condition, and monitoring to enhance efficiency to collect and process the parameters of induction motor, the suggested solution employs an IoT-based platform. The facts obtained may be kept in the cloud platform and accessed through a web page.

As a deep learning methodology, a one-dimensional convolution neural network (1D-CNN) is used to diagnose and identify early bearing faults based on Variational Mode Decomposition (VMD). Methods such as VMD-based feature extraction for fault detection are followed by multi-scale feature extraction for classification and diagnosis using convolutional neural networks and pooling layers. Bearing fault detection and diagnosis are evaluated using the Case Western Reserve University experimental dataset to determine its robustness and performance. Demodulation techniques and machine learning algorithms are also contrasted for their ability to identify and diagnose the fault. A VMD filter and one-dimensional convolutional neural network approach for monitoring bearing fault are clearly promising, according to the results obtained from the experiments.

1.4. Literature Review

In some cases, Federico Civerchia et al. [12] proposed a sophisticated industrial IoT solution, the NGS-Plant One system, intended to enable ubiquitous monitoring of industrial equipment with a battery-powered it sensing devices, enabling the creation of advanced predictive maintenance applications. To evaluate the created IoT system in a real-world context, the NGS-Plant One solution was first deployed and subsequently operated in a real-world electrical power plant. The installed testbed, consisting of 33 IoT sensing devices, has been running for two months, assessing transmission delays and system life via power consumption measurements. A year is the estimated average life if each IoT smart device is set to deliver gathered and elaborated data every 30 minutes.

A. Ajitha et al 2017 [13] used an Arduino board and needed sensors to create and execute IOT technology. The proposed strategy comprises an IoT-based platform to collect and process the induction motor parameters. The acquired data may be processed in the cloud network and seen through the web page, and the suggested approach includes a youth-based platform to gather and process the induction motor parameters. The data gathered may be kept in the cloud platform, which can be accessed via the web page. Additionally, the system will send out warnings in real-time if any of the monitored parameters are deviated from their intended ranges, allowing for quick remediation and saving both time and money. Advantages of this system involve continuous monitoring of the equipment, getting alarms, and data availability for predictive maintenance.

Li Da Xu et al.[14] began with an overview of what the IoT is all about and then moved on to cover some of its most critical potential technologies. The next section outlined some of the most important industrial uses of it. Research problems and future developments in IoT were then

discussed. Unlike previous out surveys, their review study focuses on industrial IoT applications and emphasizes potential future industrial researchers' obstacles and potential research possibilities.

Y. Amirat et al. [15] proposed a technique for early fault diagnosis based on variational mode decomposition (VMD) as a notch filter for dominant mode cancellation and a one-dimensional convolution neural network (1D-CNN) for fault identification and diagnosis. They suggested extracting features using VMD for fault detection, then extracting multi-scale features using CNN convolution and pooling layers for classification and diagnosis. The resilience and performance are examined using the well-known Case Western Reserve University experimental dataset. They also compared performance against well-established demodulation methods and machine learning methodologies for fault diagnostics. A 1D-CNN technique using VMD notch filters represent promise results for monitoring of bearing faults.

Wang et al. [16] created a unique hybrid fault diagnostic technique for diagnosing signals and fault classification that combines VMD with a one-dimensional convolutional neural network (1D CNN). VMD reduces stochastic noise from raw signals and improves their features. Because the model number and penalty parameter are critical in VMD, they developed a unique optimization approach and a new fitness function: the weighted signal difference average. The 1D CNN uses the recovered signals of mode components decomposed by improved VMD to create fault diagnostic models. The proposed hybrid technique was tested on rolling bearings. Based on the experimental findings, they suggested hybrid technique is preferable for fault identification of vibration data.

Eren L et al. [17] have been explored the performance of a general real-time induction bearing failure diagnostic system using a compact adaptive 1D CNN classifier in detail. While several research studies have produced accurate bearing fault detection algorithms, their findings have often been confined to tiny train/test data sets. Unlike traditional intelligent fault detection systems, their proposed system utilizes raw time-series sensor data as input and effectively discovers optimum features with sufficient training. 1D CNNs have several advantages over complex, deep architectures, compact architecture, configuration (instead of complex deep architectures) that performs only 1D convolutions, thus their method is suitable for real-time fault detection and monitoring and the ability to work without any pre-determined transformation. The 1D CNN based fault detection approach is evaluated against existing intelligent fault diagnostic methods using two widely used benchmark actual vibration data sets.

Song et al. [18] proposed an intelligent fault diagnostic system incorporating VMD, CNN, and LSTM neural networks. First, VMD decomposes the original bearing vibration signal into modal components with fault characteristics. Second, the number of local modal components was determined using the instantaneous frequency mean value. The two-dimensional feature matrix

comprises calculating local feature components and the CNN's input data. Third, the CNN extracts the fault features implicitly and adaptively, and its output is given as the LSTM layer's input. The LSTM extracts time series data from fault signals. Finally, the output layer used Softmax function to recognize numerous bearing problems. The experimental findings suggest that the proposed strategy increases diagnostic accuracy and overcomes standard diagnosis flaws.

Wang et al. [19] proposed a novel bearing-fault detection approach that combines multi-modal sensory information, such as accelerometers and microphone data, is proposed. The suggested technique employs 1D-CNN-based networks to combine raw vibration and acoustic inputs. The proposed technique is evaluated using extensive experimental findings from 10 groups of bearings. It outperforms methods based on a single-mode sensor in terms of diagnostic accuracy when the loss function and accuracy rate are analyzed. A visualization, analysis is also performed to analyze the suggested 1D-CNN-based method's inner mechanism.

Hao et al. [20] utilized 1D-CNN for feature extraction, while LSTM was used for classification. K-nearest neighbors (KNN), SVM, CNN, and BPNN have all been used to compare this method's effectiveness. Multi-sensor vibration data were collected and then merged to enhance bearing fault detection. Fewer time steps in the LSTM layers for long-term temporal feature extraction reduce the LSTM layers' computational complexity. The proposed method outperforms existing bearing fault diagnosis approaches and adapts well to different loads and low SNRs.

Liu et al. [21] proposed a new strategy combining a 1-D denoising convolutional autoencoder (DCAE) with a 1-D convolutional neural network (CNN) to overcome thier issue. For dancing, the DCAE model is trained using noisy input. The CNN model uses a global average pooling layer instead of fully linked layers to limit the number of parameters and the danger of overfitting. Also, randomly distorted signals are used to teach anti-noise diagnostics. Datasets of bearings and gearboxes with Gaussian noise confirm the technique. The proposed DCAE model effectively denoises input data while employing global average pooling and input-corrupt training increases the CNN model's anti-noise capacity. Thus, combining the DCAE and CNN models may provide a high-accuracy diagnosis in noisy environments.

Finally, Kiranyaz et al. [22] presented the concept of extracting the signal processing stage. For class discrimination, it is subsequently replaced by convolution layers and pooling. Despite machine learning outcomes for feature extraction and filtering, signal processing is still extremely beneficial for improving diagnostic accuracy, particularly in noisy environments.

1.5. Thesis Layout

The remaining parts of this thesis are divided into four chapters. More than one application has been taken in this study to detect the condition monitoring of industrial machines in the state

of the real-time fault and the suitable techniques for connecting the real-time statues (healthy or faulty) to the appropriate cloud.

The second chapter focuses on motor faults, neural network fundamentals, deep learning, CNN and RNN architectures, applications, and the Python framework with its main tools used in this thesis in addition to machine learning methods, SVM, KNN, Naive Bayes, and decision tree. The third chapter discussed the data set used in this thesis, explaining the structure of the proposed modulation models (1D CNN, LSTM, CapsuleNet), and training both modifiers' models to detect motor Fault. In the fourth chapter, the fault detection results after training and testing are discussed. The model's performance was evaluated using some evaluation scales and compared with the performance of other related works. Finally, Chapter Five discusses the main conclusions and future work.

2. BACKGROUND AND GENERAL TECHNIQUES

2.1. Introduction

Due to the development of deep learning techniques, convolutional neural networks and recurrent neural networks have been widely used in many fields. In the field of computer vision, deep learning is gaining popularity, in part due to giant data sets in various fields. This section discusses theories of bearing fault occurring in induction motors in the literature and the stimuli used in the literature. Also, it discusses the methods of feature extraction used in this thesis. Finally, this section discusses the principles of convolutional neural networks and variational mode decomposition, and we discussed the main evaluation metrics used in this thesis to control the proposed model performance.

2.2. Motor Faults

Induction motors are sturdy, low cost, low maintenance, relatively compact-sized, reasonably high efficiency, and functioning with a readily accessible power source. Despite their reliability, they are susceptible to a variety of unpleasant flaws. The bearing, stator winding, rotor bar, and shaft of an induction motor are the most fault-related sections, according to research on the device's design and performance. Faults may also emerge as a result of an uneven air gap between the stator's inner surface and the rotor's outer surface of the motor. Motor dependability, performance, and problems have all been the subject of several investigations in the past [23]. IEEE (Institute of Electrical and Electronics Engineering's) and EPRI (Electric Power Research Institute) investigations of motor faults are referenced in [24]. In this research, the proportion of various faults in relation to the overall number of faults was determined. IEEE examined a variety of industrial motors as part of their investigation. Using the IEEE Standard 493-1997, Table 2.1 represent the most prevalent errors and their incidence rates. The motor manufacturer's report was the foundation for research sponsored by EPRI and carried out by General Electric Company. Faults in induction motors can be categorized as follows:

a) Electrical faults: There are many types of faults that fall into this category. They include imbalances in voltage or current and single phasing. They also include under or over voltage, current and reverse phase sequence.
b) Mechanical faults: include broken rotor bars, mass imbalance, air gap eccentricity, bearing corrosion, and rotor and stator winding failure.

c) Environmental faults: Induction motors work better when the outside temperature is right and there isn't a lot of moisture around. The vibrations of the machine, which can be caused by anything from a bad installation to a bad foundation.

Table 2.1. Fault occurrence possibility on induction motor

Studied-by	Bearing-fault (%)	Stator-fault (%)	Rotor-fault (%)	Other faults (%)
EPRI	41	36	9	14
IEEE	42	28	8	22

There are many various forms of stator faults, as well as many different types of rotor faults. A list of induction motor issues, such as a broken bar fault, a rotor mass imbalance fault, a bowing rotor fault, a bearing fault, a stator winding fault, a single phasing fault, and so on, may help with fault diagnosis. Another problem with induction motors is the so-called creeping issue, which occurs when the motor fails to reach its rated speed, but instead operates at around one-seventh of its synchronous speed. About 8–9% of all motor faults are rotor faults. Broken bar, rotor mass imbalance, stator winding, single phasing, and crawling faults are all addressed in this study.

Multiple failures might develop concurrently in an induction motor, making it difficult to identify the root cause [25]. Overheating, torque loss, vibration, and stator current and voltage oscillations are all possible outcomes of induction motor problems of this kind [26]. As a result of these motor problems, some harmonic components of current and voltage might rise in amplitude. Any of these issues may have an impact on the performance of an induction motor. The causes and implications of various induction motor problems are addressed in the following sub-sections.

2.2.1. Stator Fault

Stator of an induction motor is exposed to a lot of different stressors [27], including mechanical, electrical, thermal, and environmental. If these stressors are strong enough, stator faults may happen. The stress of a well-designed motor can be kept under control if the correct operations and maintenance are done. The stator problems can be broken down into two groups: i) problems with the laminations and the frame of the stator, and (ii) problems with the stator winding. A lot of people have stator problems like the second one. IEEE and EPRI found that 28–36 percent of induction motor problems are caused by stator winding problems [28]. Most of these problems are caused by a combination of the stressors above.

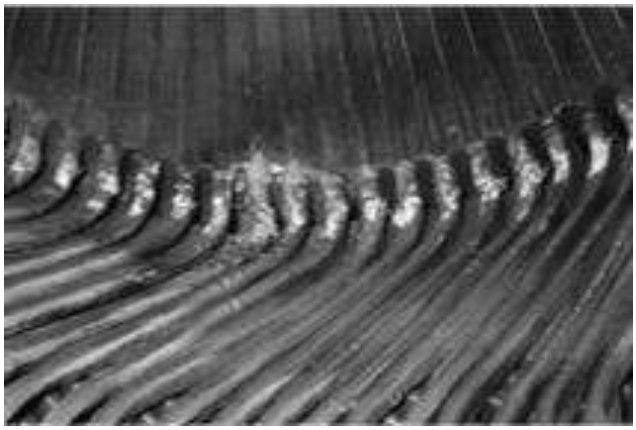


Figure 2.1. Photograph of damage stator winding [29]

In Figure 2.1, shows induction motor problems thar are caused by stator winding problems

2.2.2. Effects and Causes of Stator Faults

i. Mechanical Stresses: Mechanical faults are created when the stator coil and rotor collide. The Rotor may collide with stator owing to misalignment of rotor and stator or shaft deflection. Mechanical vibrations may cause the stator windings to detach, resulting in an open-circuit problem if the stator-laminations breach the coil insulation leads to ground fault occurring [30].

ii. Electrical Stresses: They are mostly caused by transients in the supply voltage. This transient occurs as a result of various faults (such as line-to-line, line-to-ground, or three-phase faults), as a result of lightning, as a result of circuit breakers being opened or closed, or as a result of the drives of variable-frequency [31]. This transient voltage degrades the stator winding's life and, in extreme cases, may result in a turn-to-turn or turn-to-ground failure.

iii. Thermal stresses: This is due to thermal overloading, which is the primary cause of the stator winding's insulation breakdown. Several factors may cause thermal stress, including a high temperature, a lack of ventilation, and an imbalanced supply voltage [31]. There is a rule of thumb that suggests that if there is a 3.5% voltage imbalance per phase, winding temperature will rise by 25% in the phase with the greatest current [32]. The temperature of the windings will rise if the motor is started and stopped a lot in a short period of time. It's possible that if the winding temperature rises and the motor is beyond its temperature limit, the best insulation may also fail rapidly. Thumb rule: insulation life is decreased by half for every 10 degrees Celsius rise in temperature over the stator winding temperature limit [33].

iv. Environmental stresses: If the motor is operating in an atmosphere that is too hot, too cold, or too humid, these strains may occur. The insulation of the stator winding may be contaminated by foreign material, which can limit the pace at which heat is dissipated from the motor [31]. The effect of temperature is given in Table 2.2.

Table 2.2. Effect of rise of temperature

Ambient in °C	Insulation life in hours
30	250,000
40	125,000
50	60,000
60	30,000

The area around the motor should be clear of obstructions; otherwise, the heat created by the rotor and stator will raise the winding temperature, reducing the insulation's life.

2.2.3. Bearing Fault

An induction motor's spinning shaft is supported by two sets of bearings at start and end of the rotor, they kept the rotor in place and made it easier for it to revolve freely by reducing friction. Each bearing comprises of an inner and outer ring referred to as races, as well as a set of rolling components referred to as balls that are placed between these two races. Normally, when a motor is used, the inner race is coupled to the shaft and the load is conveyed through spinning balls this reduces friction. The use of lubricant between the wheels of the race reduces friction even more. Figure 2.2. (a) shows a typical ball-bearing and Fig. 2.2.(b) shows a dissected ball-bearing [34].

A bearing fault is defined as any physical damage to the inner race, the outer race, or the surface of the balls as a result of the operation of the bearing. Bearing failure is the most common cause of induction motor failure, and it is the component that fails the most often. When it comes to induction motors, it is the single most common source of fault according to the findings of an IEEE and EPRI researches.

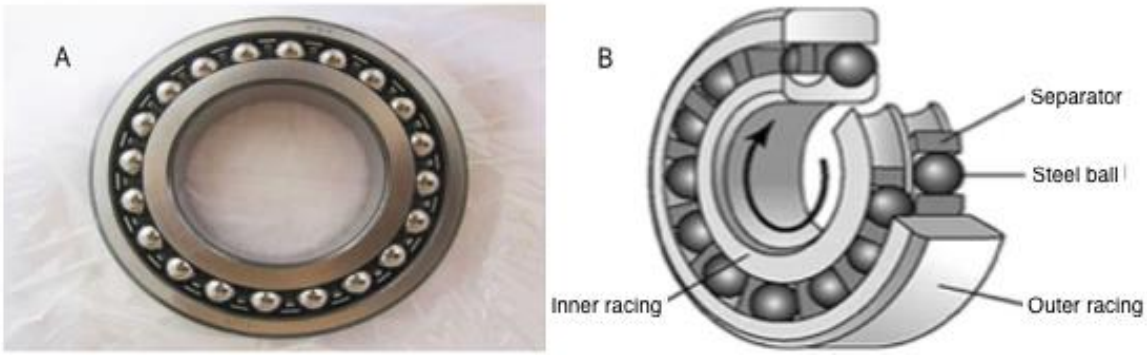


Figure 2.2. (a) Ball bearing, (b) Ball bearing (dissected) [34]

Figure 2.2, shows ball bearing and ball bearing (dissected)

Effects and causes of bearing fault:

1. Excessive loads, tight fittings, and rapid temperature increase may all cause the two races and ball materials to anneal. Additionally, they may deteriorate, if not completely destroy, the lubrication. Brinelling happens when the load exceeds the elastic limit of the bearing material.
2. Fatigue failure: This is because the bearings have been running for a long time now, which makes them wear out. It breaks down and then removes small pieces of material from the surface of races or balls. In this case, the failure of the bearings will spread when they are used again, so it is called "progressive." For this bearing to fail, the noise and vibration level of the motor will get worse.
3. Corrosion: Because of corrosive fluids (acid, etc.) or corrosive atmospheres, this effect occurs. The corrosion of bearings may occur if lubricants degrade or if the bearings are installed haphazardly. Corrosion may lead to early failure.
4. Contamination: It is one of the main reasons that bearings fail. Dirt and other foreign particles get into lubricants, which happens a lot in places where people work with machinery. Some of the effects of contamination are high vibrations and wear on the parts.
5. Lubricant failure: This occurs when the lubricant flow is limited or the temperature is high. Overheating occurs when the lubricant's properties are degraded, resulting in increased wear of balls and races. The lubricant (grease) melts and leaks out of the bearing if the temperature is too high. Lubricant failure is indicated by discolored balls and tracks.
6. Bearing misalignment: For this, the surface wear of races and balls causes the bearings to heat up.

In the event of a bearing failure, it has been shown that friction arises. This causes the temperature of the bearings to rise and the vibration of the machine to get worse. For example, bearing temperature and vibration can tell you a lot about the condition of the bearings and the health of the machine.

2.2.4. Condition Monitoring and Its Necessity

For industrial prime movers, induction motors are the most common workhorse, because of their robustness, cheap cost, minimal maintenance, compact size, great efficiency, and ability to operate with a readily accessible power source. Induction motors utilize around half of the nation's total produced electricity [35]. Induction motors are used in a large variety of applications, but they have a few drawbacks in terms of their working circumstances. If these circumstances are exceeded, stator and/or rotor early failure may occur. This failure may result in a loss of time and money, and in some cases, the whole industrial process. Therefore, it is critical to ensure that the induction motor does not fail at any point. There is a constant strain on induction motor operators and mechanics to avoid unexpected downtime as well as to lower the maintenance costs of the motors.

Three types of electrical motor maintenance are available: breakdown, regular and condition-based. It's a common practice in breakdown repair to "run the motor until it fails," which implies that only when the motor breaks down does the maintenance begin. In this situation, even if the motor may operate for a long time before it has to be serviced, it is considerably more expensive to replace the complete machine rather than just the damaged components. Due to the time it takes to get back up and running, productivity suffers. Fixed-time maintenance necessitates a lengthy downtime since the motor must be stopped for examination. Every fault must be identified appropriately by skilled and experienced technicians. Condition-based motor maintenance is required for all of these reasons. As the motor is permitted to function normally, the first hint of an incipient problem is taken care of immediately. Induction motor condition monitoring is the subject of several studies [35].

When there is a problem with a piece of equipment, the plant operator needs to have enough information to make the best possible decision about what to do next. If there is not enough information, there is a chance that the wrong diagnosis of the problem will lead to the wrong replacement of parts, and if the root of the problem isn't found, the replacement or any other action that has been taken will also fail.

In condition monitoring, motor signals are sent to a system of data acquisition all the time, and the safety of the motor is constantly checked while it is running. This is called online condition

monitoring of the motor, and it is also possible to discover issues. even as they are developing. Operators and technicians can plan ahead for preventive maintenance and get the parts they need for repairs before they need them. Thus, condition monitoring can help you plan your maintenance and keep your motors up and running as long as possible, which will make the motor more reliable. The following are some of the benefits of using condition monitoring [36]:

- Predict the faults of the motor.
- Optimize the maintenance of the motor.
- Reduce the cost of maintenance.
- Minimize machine downtime.
- Increase the dependability of the motor.

2.3. Steval-Stwinkt Kit

The STWIN Sensor Tile wireless industrial node (STEVAL-STWINKT1) is a sophisticated industrial IoT reference design and development kit for predictive maintenance and condition monitoring. The kit also comes with a core system board that includes a range of industrial grade sensors. An ultra-low power microcontroller is used to analyze vibration data from nine degrees of freedom, as well as high-precision environmental and temperature monitoring on a local level. Besides the development kit comes with a lot of software and firmware libraries, there is also a cloud-based dashboard application that can help speed up the design process for full solutions. An on-board BLE (Bluetooth Low Energy) module and a separate plug-in extension board allow the kit to connect wirelessly to the internet through Wi-Fi (STEVAL-STWINWFV1). A built-in RS485 transceiver allows for wired communication as well. Additionally, an STMod+ connection is included on the main system board for attaching appropriate small form factor STM32 family peripherals like the LTE Cell pack. [37].

Figure 2.3. STEVAL-STWINKT1 STMicroelectronics [37]

shown in figure 2.3 shows the STEVAL-STWINKT1 STMicroelectronics kit with its peripherals.

2.4. Data Presentation on Its Cloud

The STWINKT1 is used to create custom industrial condition monitoring applications. There are two methods for connecting STWIN to the cloud. The first method – the standard firmware that comes pre-flashed on STWIN – utilizes the onboard Bluetooth Low Energy Module to connect STWIN to the ST BLE sensor application for Android/iOS, which enables sensor data reading, motion and audio algorithm demos, and firmware updates over the air.

The second project develops an end-to-end sensor to cloud application and complements it with a cloud application (DSH-PREDMNT) hosted by ST and powered by AWS. We may link the STWIN to the specialized DSH-PREDMNT web-based dashboard to record and monitor sensor data and equipment status using the plugin WiFi expansion board STEVAL-STWINWFV1 (available as an add-on to the kit). The program collects and analyzes sensor data for the MP23ABS1 analog microphone (up to 80 kHz) and the IIS3DWB accelerometer (up to 5 kHz) [38].

2.5. Internet of Things

In the scenario of telecommunications of the modern wireless, the Internet of Things (IoT) is a new concept that is quickly gaining ground. The central principle of this theory is the pervasive existence of several items or objects surrounding us, such as Radio Frequency Identification (RFID) tags, sensors, actuators, and cell phones. Via specific management structures, they are able to communicate with each other and collaborate with their neighbors to achieve common

goals [39], IoT key strength definition is doubtlessly the potential effect, it would have on many aspects of actions and daily-life of potential-users. From a view of private users' point, both in the domestic and working sectors, the most apparent results of the IoT implementation would be shown. In this sense, only a few examples of potential implementation situations in which the modern model will soon play a crucial role, living assisted, e-health, and enhanced learning.

Similarly, from the view point of market users, in areas such as industrial and automation logistics, business/process, production manufacturing, intelligent transport of people and products, the most clear suggestions would be noticeable equally, It highlights potential possibilities that could emerge, starting from the idea that "public demand combined with advancements in technology could drive the widespread proliferation of an Internet of Things (IoT) that could contribute invaluably to economic growth [40], like the current Internet, The potential risks resulting from the widespread implementation of such a technology are also highlighted, Indeed, it is stressed to the degree that everyday objects become threats of information security, these risks could be spread by the youth far more broadly than the Internet has to date, many challenging problems still need to be tackled, and before the IoT concept is generally accepted, both technology and social knots must be untied, Central issues are making it possible for interconnected devices to be completely interoperable, providing them with an ever-higher intelligence degree by permitting their autonomous and adaptation actions, while ensuring confidence, privacy, and protection. The IoT definition also presents many new issues relating to the networking aspects [41].

In fact, in terms of both computational and energetic power, the internet of things presents would be characterized by low resources. Accordingly, in addition to the obvious scalability concerns, the suggested solutions must pay particular attention to resource quality.

2.6. System Design

For fault diagnosis, data is collected from an induction motor with the STWINK module. This module works wirelessly and with batteries and transmits data wirelessly. The schematic of the experimental setup set up is given in Figure 2.5.

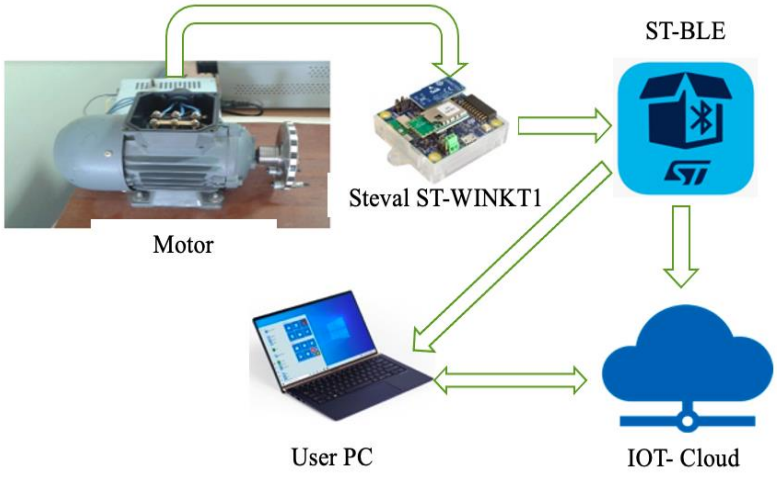


Figure 2.4. Predictive maintenance system design

In Figure 2.4. the data is first taken from the asynchronous engine and transmitted to the cloud environment. It is then transferred to a computer environment and analyzed.

2.7. Feature Extraction Methods

In this thesis, four methods have been proposed for classifying vibration signal-based feature extraction. These methods are statistical features (SF), the combination of welch power spectral density (PSD) method with a statistical feature, the combination of Fast Fourier transforms (FFT) method with a statistical feature, and the combination of statistical feature with a wavelet packet decomposition (WPD) method.

2.7.1. Statistical features

The mathematical-statistical analysis method is applied to the signal to obtain more information about the signal that cannot be obtained from the mother signal. This method is an easy method to extract features. In this thesis, six statistical features (mean, standard deviation, variance, median, maximum value, and minimum value) were applied to extract six features [42]. Equation (2.1) describes the method for calculating the mean function. Standard deviation calculates the deviation values from the mean value of the vibration samples. Equation (2.2) describes the method for calculating the standard deviation. The middle element of the array refers to the median value, as seen in equation (2.3). Variance is the average of the square deviations.

$$Avg = \frac{1}{n_e - n_s} \sum_{i=n_s}^{n_e - n_s} X_{iVb} \tag{2.1}$$

$$Std = \sqrt{\frac{1}{(n_e - n_s)-1}\sum_{i=n_s}^{n_e-n_s}(X_{iVb} - \bar{x}_{Vb})} \quad (2.2)$$

$$median(x) = \begin{cases} X\left[\frac{n}{2}\right] & if\ n\ is\ even \\ \frac{(X\left[\frac{n-1}{2}\right]+X\left[\frac{n+1}{2}\right])}{2} & if\ n\ is\ odd \end{cases} \quad (2.3)$$

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} \quad (2.4)$$

Where n_s refers to the signal's starting point while n_e refers to the signal's endpoint, $n_e - n_s$ is pointing to the samples total number. The variable I is pointing to the horizontal axis values, $\bar{x}_{Vb}$ refers to the mean value of the Vibration data, X is the median equation refers to the ordered list of values in data set. The parameter n refers the number of values in the dataset, and S^2 refers the samples of variance, x_i refers to the value of the one observation, $\bar{x}$ refers to the mean value.

2.7.2. Fast Fourier transforms

Fourier transforms (FFT) is a useful method for extracting information from vibration signals because it performs large scale arithmetic-operations on actual values from four different frequency bands (alpha, beta, gamma, and theta). The Butterworth fourth-class filter used to isolate these four prohibitions. Equation 2.5 was used for data feature extraction [42]-[43]:

$$X_f = \sum_{n=0}^{N-1} X_n e^{-i2\pi f\frac{n}{N}},\ f = 0,1,2,\ldots\ldots,N-1 \quad (2.5)$$

Where N is the total number of data samples, X_f is the coefficients of FFT, and n is pointing to the total number of points in the FFT.

2.7.3. Wavelet packet decomposition

Wavelet Packet Decomposition (WPD) is one of the techniques for extracting additional characteristics from wavelet-decomposition (WD) by signal deconstruction into multiple frequency bands. It is beneficial for extracting delicate details from signals with varying resolutions, as it may deconstruct the signals with varying resolutions. each signal is divided into two integral sub-spaces, denoted by the letters V and W, where V denotes low-frequency information and W denotes high-frequency information, respectively. A complete wave tree will be created after the signal has been wholly dissected, as seen in figure 2.2.[45], [46]. Figure 2.5 shows the steps of Wavelet packet decompositions.

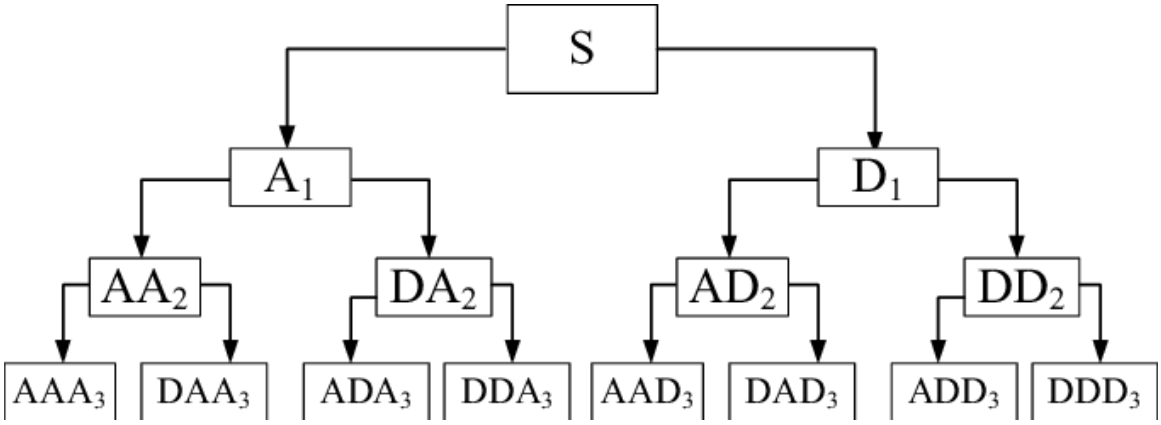


Figure 2.5. The WPD tree (A= Approximation coefficient, and D= detail coefficient) [47].

In Figure 2.5, the detail and approximation components for the S signal are obtained. These signals are obtained for certain levels and feature extraction is performed.

2.8. Deep Learning

Deep learning is a neural network with multiple-layers of nonlinear processing units. Each successive layer receives the output from the previous layer as input. The network can extract the complicated hierarchical characteristics from a vast quantity of data by employing these layers. Deep learning may represent functions with increasing complexity if the number of layers and units in a single layer is raised. Deep learning algorithms may assist humans in establishing mapping functions for operational convenience if sufficient labeled training datasets and acceptable models are available [48]. Also, Deep learning is a branch of machine-learning that trains computers to do what comes easily to humans. Machine learning algorithms employ computational approaches to "learn" information from data directly without depending on a preconceived equation as a model [49]. Deep learning is highly suitable for image identification, which is vital for tackling issues such as motion detection, face recognition, and various advanced driver-assistance technologies such as autonomous parking [50]. Auto encoder, pedestrian detection, autonomous driving and lane detection, and Convolutional Neural Networks (CNNs) are the four basic architectures used in deep learning. Convolutional and Recurrent neural networks were utilized to construct a detection model for extracting motor fault from vibration data.

2.8.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most popular categories of deep learning. First designed by Yann LeCun [51] as LeNet and this network was first implemented to recognize handwritten numbers. Convolutional Neural Networks (CNNs) can extract features from

data and classify them into a single structural model, unlike traditional Artificial Neural Networks (ANNs) [52]. With CNN's recent success in computer vision, it is clear that its use with picture data from a particular network topology is key [53]. Images and movies, both of which have high dimensionality, are common data types for this technique. The functioning process of CNN is comparable to that of an ANN in principle. There is a significant distinction between CNN and other image processing systems because CNN utilizes two-dimensional filters or weights convolved over the image data. Both supervised and unsupervised model learning may be achieved using CNNs in the same way. Figure 2.6 depicts the overall structure of CNN.

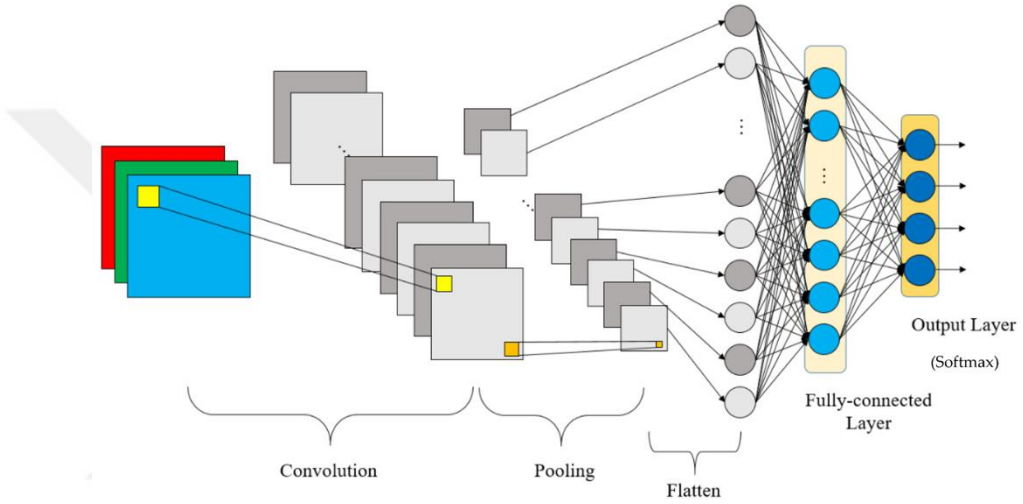


Figure 2.6. CNN Layers

Since their inception, CNNs have found widespread use in 1-D signal processing, despite their intended intent to distinguish objects in two-dimensional data like images and video frames. The most important components in the CNNs are the convolutional layers, Dropout layers, Pooling Layers, and Fully Connected Layers. The convolutional layers contain a group of filters or kernels that are convolved with the input data to extract the feature map. These filters are a group of numbers (weights) that are organized in two-dimensional (in case of single-channel) or three-dimensional (in case of three channels) matrix. The learning operation includes many iterations, the weights are initialized randomly and updated continuously according to the expected output. The filters are conniving with the convolution layer input, as shown in figure 2.6. The filter is multiplied with a highlighted patch (also 3x3) of the input feature map and sums up all values to generate one value in the feature map output. Note that the filter slides along the height and width of the feature map of the input, and this process continues until the filter cannot slide further anymore. When a feature map has been pooled, a two-dimensional filter is slid over each channel and the features in that area are summed.

By using pooling layers feature maps may be reduced in size. There are fewer parameters to learn, which in turn decreases the processing time of the network. A feature map generates from convolution layer, while features summing up by pooling layer in a specific area. So, the convolution layer generates summary characteristics rather than exact features that may be used for subsequent processing. As a result, the model is more resistant to changes in the image's feature locations. Pooling functions like max, min, and average is used to minimize the input feature maps. Figure 2.7 shows that the max-pooling is most often utilized function. Pooling operations that pick the largest or minor elements from a feature map are known as max pooling. A feature map comprising just those most prominent features in the previous one would be generated after the max-pooling [54], [55].

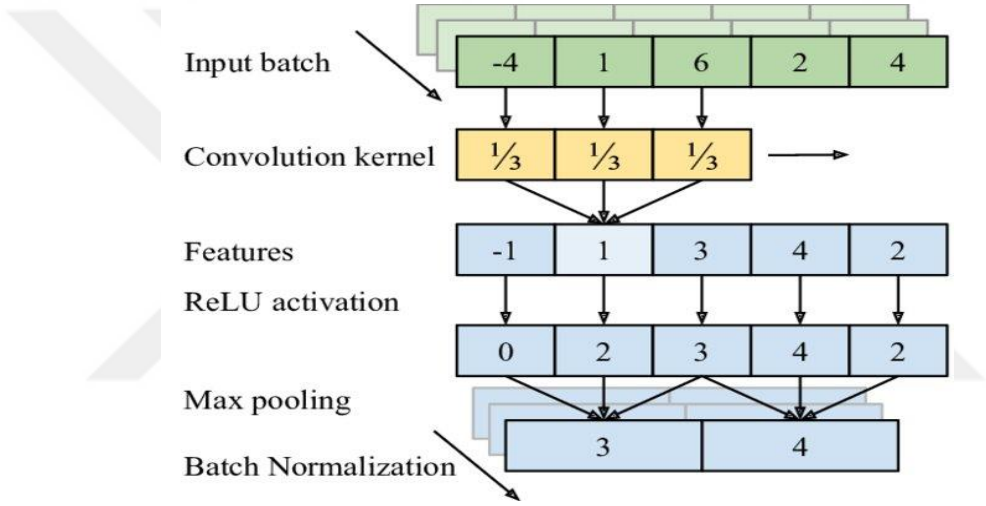


Figure 2.7. Input data (green), convolved data with 1D (yellow), ReLU activation function and max pooling layer used, applied batch normalization to the max pooling output

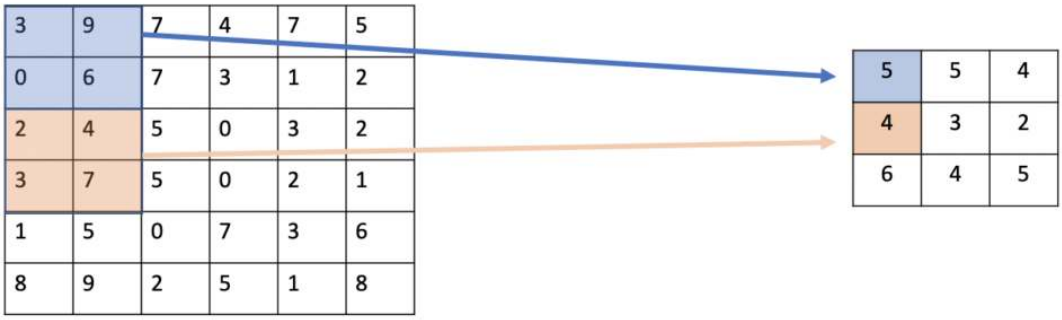


Figure 2.8. Max Pooling Operation.

As shown in Figure 2.8. Fully linked layers may be regarded as convolution layers with

weights of 1x1 dimensions. It's important to note that all of these levels' units are related to each other. Like an ANN, these layers operate by multiplying the feature maps inputs with their weights and adding the bias vector, and then applying the non-linear activation function. The use of fully linked layers in picture categorization and pattern recognition has become more common [56].

2.8.2. 1D convolution neural network

1D CNNs are used in a lot of signal processing applications, especially vibration signals. Where the signal is converted to spectrograms and analysis [56]. 1D CNNs have become a popular method of signal processing because these networks can take advantage of the signal's fine time structure. For example, Kim et al. [57] proposed a 1D CNN architecture for automatic tagging of musical signals. The basic architecture of 1D CNNs is somewhat similar to a regular neural network, but it differs from a neural network in that it receives raw data as input rather than handcrafted features. Then, it analyzes the data through several trainable convolutional layers to extract features. Finally, these networks classify the features in the same architecture [58].

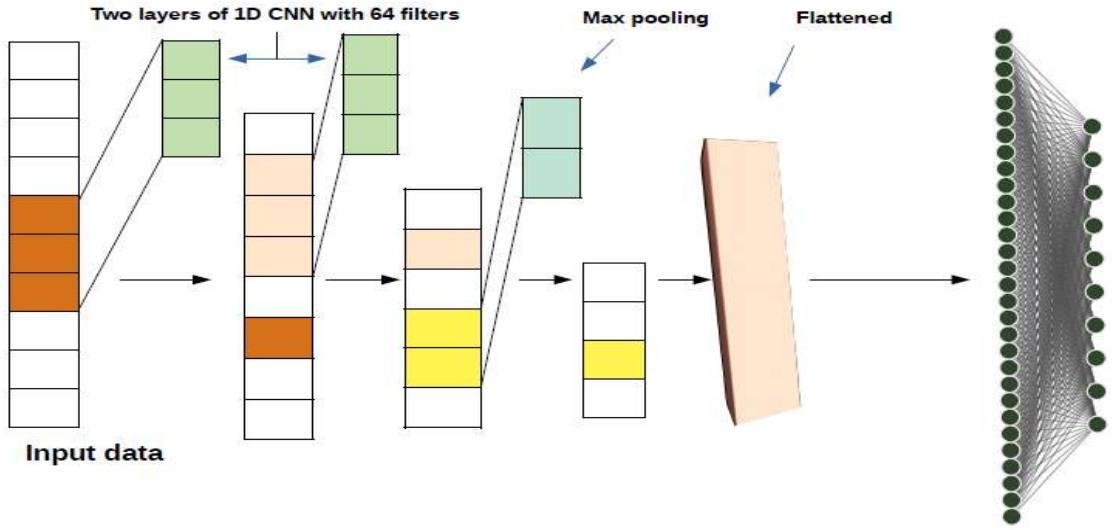


Figure 2.9. An example of the architecture of the 1D CNN.

Figure 2.9 shows the general form of the 1D CNN architecture.

2.8.3. Capsule Network

Capsule Networks (CapsNet) are networks that are capable of retrieving spatial information and other critical properties in order to compensate for the information loss that occurs during pooling operations. Consider the distinction between a capsule and a neuron. Capsule outputs a vector with direction. For instance, if the image's orientation is changed, the vector is likewise

shifted in the same direction, but the output of a neuron is a scalar number that conveys no information about its direction.

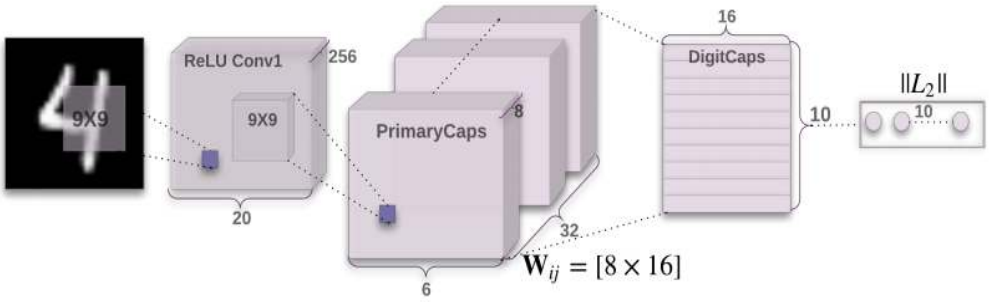


Figure 2.10. CapsuleNet Layers

Figure 2.10 shows the architecture of CapsuleNet Layers

2.8.4. Recurrent Neural Networks (RNN) with Long-Short term memory (LSTM)

A recurrent neural network (RNN) is a neural loop network that has internal memory. The RNN performs the same function for all input data, and the current input, output depends on the previous operation [59]. This means that this network deals with sequence data such as climate and text data. In the texts, the sentence consists of sequential words linked together, and in the climate, there is a relationship between today's temperature and yesterday's temperature [60]. An RNN mainly consists of multiple neurons (nodes). Each node has a time-varying value. Also, each connection has an actual weight that can be modified in any situation. The neuron's output at time t-1 will be passed to the input at time t, and add the data at time t to generate the output at time t. The formulas of RNN are shown in equations 2.6 and 2.7 [60].

$$h_t = \tanh\left(w_h h_{t-1} + w_x x_t\right) \tag{2.6}$$

$$y_t = w_y h_t \tag{2.7}$$

Where h_t refers to the hidden layer vector, x_t refers to the input vector, y_t refers to the output vector, and w_h refers to the weighting matrix.

After the output is obtained, it is sent back to the recurrent network for the appropriate decision to be made. An RNN takes into account the current inputs and the outputs learned from the previous inputs. This redundancy can make the network somewhat opaque because RNN tends to be over-processed and requires good organization. But it is no different from a normal neural network. It can be considered as multiple copies of the same network, each copy transmitting a message to the next copy. Figure 2.11 represents the loop process in the RNNs.

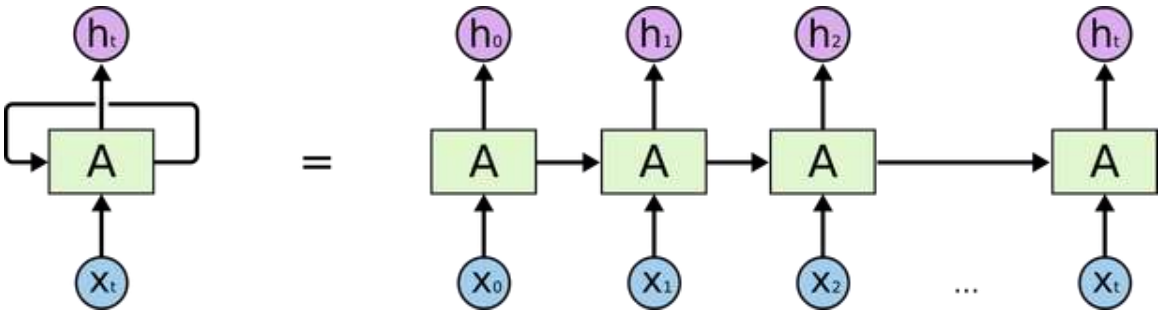
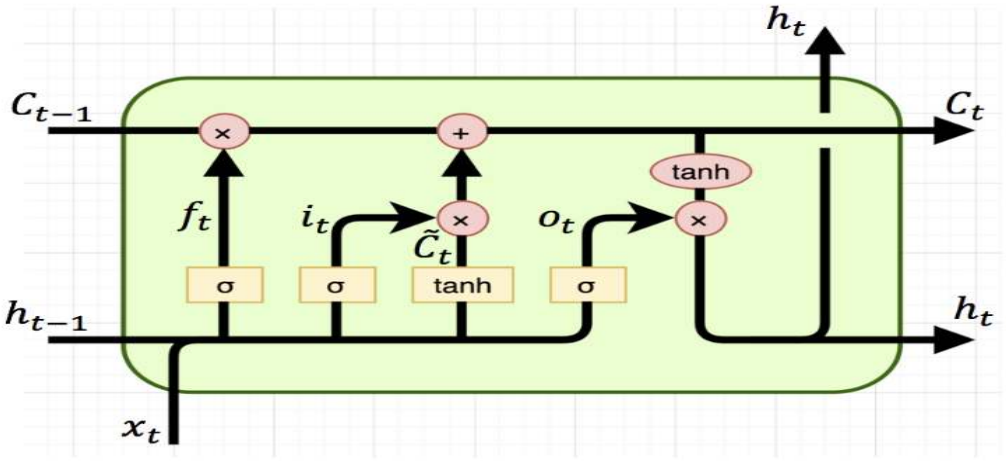


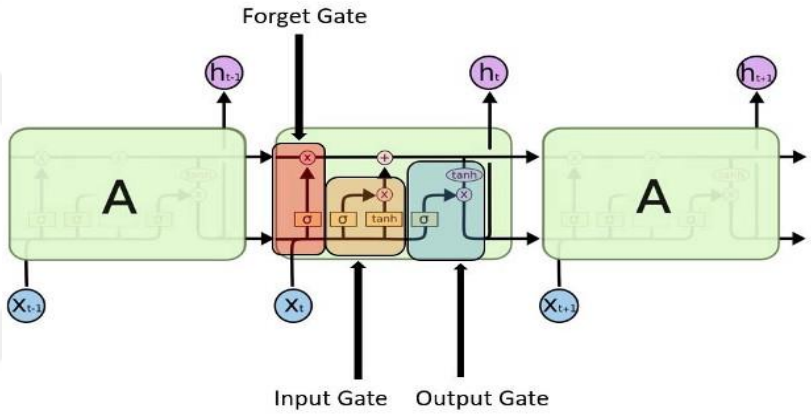
Figure 2.11. A simple RNN architecture [61]

In Figure 2.11, Part Net looks at the input (x_i), then gives the output (o_i). Some of the applications of RNN are given as language modeling [62], speech recognition [63], machine translation [64].

Long-Short term memory (LSTM) was first proposed to improve RNNs by Hochreiter and Schmidhuber (1997) [61]. The idea of an LSTM network is based on the concept of control gates that control the flowing data and prevent problems that unnecessary data may cause. Later, the LSTM network was updated by Gers et al. (2000) [65]. Forgetting gates have been added to forget useless memories from the memory cell. The number of gates in the LSTM network becomes three: input gate, forget gate, and output gate. Figure 2.8 shows the three gates of a single cell structure of the LSTM network. Also, figure 2.9 shows the recurrent process on the LSTM network. The LSTM network differs from the RNN in learning skills by using a complex gate approach. The input gates learn the importance of the input information, The network gate learns the importance of the volume of relevant information. In addition to the forgotten gate. A study was presented in 2012 [65], in which LSTM was used for a modeling task using English and French. This study concluded that LSTM outperformed RNN by 8%. Figure 2.10 shows the architecture of each gate. Also, figure 2.10 shows the architecture of the LSTM model for motor fault recognition.



(a)



(b)

Figure 2.12. (a) Single-cell of LSTM network [66], **(b)** LSTM's gates [67].

As shown in Figure 2.12 (a), the forget gate is defined using Equation 2.8. This equation eliminates unnecessary information. This is done by entering the output (h_{t-1}) at the previous unit ($t-1$), and adding the current time (t) input (x_t) into the sigmoid function $S_{(t)}$ As can be seen in equation 2.9. Where values between 0 and 1 are generated, as can be seen in figure 2.12.(b) A value of 0 indicates that this information is forgotten and a value of 1 indicates that this information is not forgotten [60].

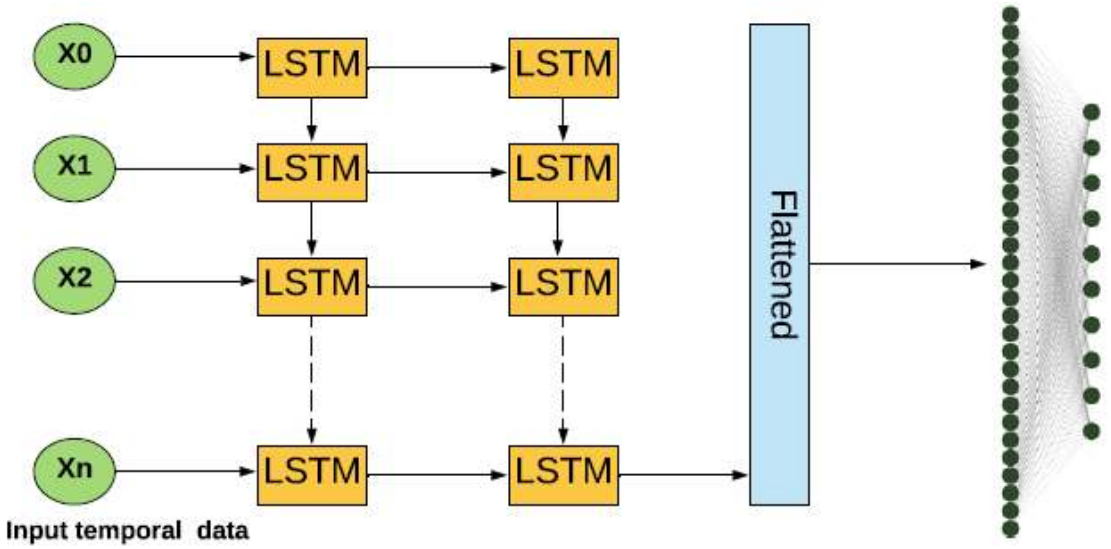


Figure 2.13. The architecture of the LSTM model for motor fault recognition [66].

Figure 2.13. shows the architecture of the LSTM model for motor fault recognition

$$f_t = \sigma(w_f \,.\, [h_{t-1}, x_t] + b_f \tag{2.8}$$

$$S_{(t)} = \frac{1}{1+e^{-t}} \tag{2.9}$$

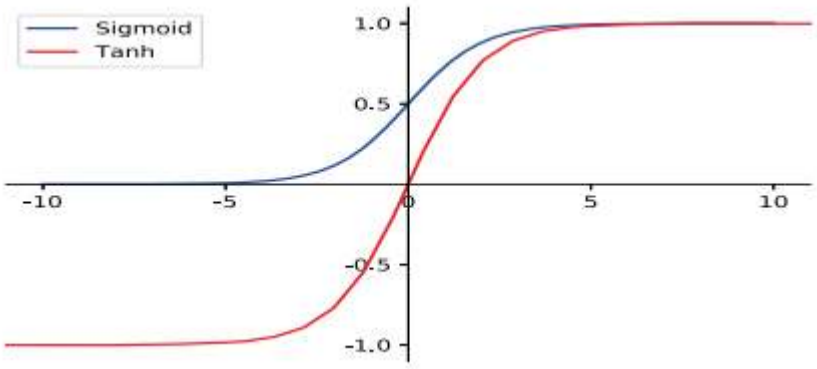


Figure 2.14. The sigmoid and tanh function [67].

Figure 2.14 shows the change in happening between Sigmoid and Tan hyperbolic function

As for the input gate, it defines the new information that must be remembered in the cell. In the input gate, equation 2.10 is used to calculate the value of (i_t) to determine the amount of state of the new information cell to remember. The value (i_t) is between 0 and 1. At the same time, the tanh layer gets an election message ($\tilde{c}_t$) using equation 2.11. Then the values of the i_t are multiplied by the values of $\tilde{c}_t$that were obtained using the Sigmoid function to get updated information, as can be seen in equation 2.12 [60].

$$i_t = \sigma(w_i \,.\, [h_{t-1}, x_t] + b_i \tag{2.10}$$

$$\tilde{c}_t = tanh(w_c \,.\, [h_{t-1}, x_t] + b_c \tag{2.11}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{2.12}$$

As for the output gate, it determines the information that will be output from the cell. Whereas, the equation 2.13 is used to generate values between 0 and 1 to determine the number of cells of information that need to be output (o_t). Then the result we got in equation 2.13 is multiplied by a value between -1 and 1 using the tanh function, as can be seen in equation 2.14. Here we will get the output (h_t) For the LSTM block at time t.

$$o_t = \sigma(w_o \,.\, [h_{t-1}, x_t] + b_o \tag{2.13}$$

$$h_t = o_t * tanh\,(c_t) \tag{2.14}$$

2.9. Evaluation Metrics of the Semantic Segmentation

The main objective of the vibration signal classification based on motor vibration signals is assigned a class label for each type of fault in the target. The evaluation process of motor fault is performed using global and class-wise metrics such as confusion matrix, sensitivity, specificity, the area under the curve (AUC), precision, recall, and f1-score. The global metrics measure the overall accuracy without respecting the classic's accuracy individually. These metrics are accurate with applications that have a convergent amount of class labels. While in other metrics, the accuracy of each is usually used by the application that has class imbalance [56]. The following metrics are used to evaluate the performance of the fault detection based on motor signals in this thesis.

2.9.1. Confusion Matrix

It is a tool that measures the performance evaluation of the classification model. It contains information about the number of actual and predicted labels. The performance of these systems is often evaluated using confusion matrix data. Table 2.3 shows the confusion-matrix for a two-class model [68]. In the context of this thesis, the elements in the confusion matrix have the following meaning:

• **TP** is the number of **correct-predictions** that an instance is positive,

• **FP** is the number of incorrect-predictions that an instance positive,

• **FN** is the number of incorrect-predictions that an instance is negative

• **TN** is the number of **correct-predictions** that an instance is negative.

Table 2.3. The Confusion Matrix for Two Class [68].

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

2.9.2. Global Accuracy, Sensitivity, and Specificity

Global accuracy is defined as the ratio between the correctly predicted values for all classes to the total number of values, as seen in equation 2.15. Sensitivity is the ratio of true positives that correctly identified overall positive assessments, as seen in equation 2.16 [69]. Specificity is used to calculate the proportion of the true negatives correctly identified overall negative assessments, as seen in equation 2.17.

$$Global\ Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.15}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.16}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.17}$$

Global Accuracy is not adequate with a class imbalanced dataset where there is a prominent disparity between the number of negative and positive labels. To overcome this issue, the accuracy is calculated for each class separately, and the mean accuracy is produced [70]. A sensitivity test is used to identify the normal state of the motor correctly. This is called the ratio of true positives that are correctly determined by the test. Whereas the specificity test is used to identify motors with fault states correctly. It is called the proportion of true negatives, which is correctly determined by the test [71].

2.9.3. Precision, Recall, and F1-Score

Precision is the proportion of correctly classified positive cases from the total number of actual positive cases. Equation 2.18 shows how to calculate the precision. Recall (Like a sensitivity test) is the ratio of normal cases that were correctly classified to all observations, as seen in equation 2.19. The F1 score is the average of both precision and recall tests. Where false positives and false negatives were taken into account. This test is helpful if the distribution of a class is uneven. While, if the false positives and false negatives have the exact cost, the accuracy test will work best. F-

score will be used when a balance needs to seek a balance between precision and recall, as in Equation 2.20.

$$Precision = \frac{TP}{TP + FP} \tag{2.18}$$

$$recall = \frac{TP}{TP + FN} \tag{2.19}$$

$$F1 - Score = 2 * (\frac{precision * recall}{precision + recall}) \tag{2.20}$$

2.10. Python Software Environment

In this thesis, the anaconda version (4.9.2), Python version (3.8.5), and Jupyter notebook version (6.1.4) software package are used to design the 1D CNN, LSTM, and 1D CNN + LSTM architectures. However, the following toolboxes have been used to prepare the main program and supported function codes.

2.10.1. Deep Learning Toolbox

There are many frameworks for deep learning in the Python environment. These libraries allow researchers and programmers to write code efficiently. The most popular of these platforms are TensorFlow and Keras [72].

2.10.2. TensorFlow platform (TF):

The TensorFlow framework is one of the most widely used open-source platforms for deep learning. The TensorFlow platform was developed by Google in November 2015 and supports deployment across CPUs and GPUs. One of the advantages of the TensorFlow platform is that it does not require the use of a graphics processing unit (GPU). It will try to use it automatically. But if there is more than one GPU, you must specify the operations that each GPU will execute by writing the following code [72]:

With TensorFlow. Device ("/gpu: 1"):

Model definition here

To install TensorFlow on anaconda, open the Anaconda prompt. And then, type the following command: pip install TensorFlow. The last version of the TensorFlow used is 2.0.

2.10.3. Keras platform:

It is a library within the Python environment consisting of a high-level neural network written in Python and using TensorFlow in the back-end. Keras is relatively easier than TF because it

enables you to run quick experiments, for example, it can execute entire code with less than 15 lines of code. Also, writing Keras in Python enables it to become a particular framework to work with. This is because the Python community has a larger community of users and supporters and is very easy to get started with. To install Keras on anaconda, open the Anaconda prompt. And then, type the following command: pip install Keras.

2.10.4. Graphic Processing Units (GPUs)

GPUs were built for rendering graphics. However, GPUs have grown popular for general-purpose high-performance computing in the past ten years, including medical image processing. Because of the enhanced programmability of these devices, along with cheap cost and good performance, this is most likely the reason. [73].

Recently, the use of GPUs became popular for general-purpose computations that require a highly data-parallel architecture. GPUs architecture consists of many processing cores, each of which has many functional units, such as arithmetic logic units (ALUs). Each functional unit includes a set of thread processors under the supervision of a shared control unit. The control unit stimulates the thread processors to implement the same instruction on each image pixel in parallel [74].

2.10.5. Collaborator (Colab):

Collaborator, collab, or Google Colaboratory is a cloud service for writing code in the Python language and executing it on your computer for free. It also allows free access to GPUs and TPU as well. The club is based on Jupyter notebook technology, which is an open-source tool that can run locally or on the cloud. It contains multiple cells fully configured for deep learning and free access to a powerful GPU. Colab contains essential machine-learning and artificial-intelligence libraries, such as Keras, Matplotlib and TensorFlow. It also offers to save Jupiter notebook files on the user's Google Drive account. Finally, the Google Colaboratory infrastructure is hosted on the Google Cloud platform [75].

3. MATERIAL AND METHODS

In this part, an induction motor condition monitoring system has been developed and implemented. A microprocessor and a vibration sensor are included in this system to look for mechanical problems that may arise in the motor, the extraction of fault data using appropriate approaches has been used to research and examine a variety of fault types, including (stator and bearing problems). The results revealed that the methods utilized and the vibration measuring technology had a high probability of accurately detecting and diagnosing the majority of mechanical faults. As a result, a diagnostic method has been created to quickly and accurately identify the health of a motor. Two methods for the tolerance of the induction motor fault based on the vibration signals has been included.

3.1. Cloud based bearing fault diagnosis of induction motors

It was done by taking vibration samples from the motor directly and performing the two ways of obtaining errors, which are (bearing 1) and (bearing 2), the first includes placing one screw on the disk connected to the motor and obtaining signals. As for the second one, it is done by placing an additional screw in succession to the first and obtaining the error named (bearing 2), then the data is analyzed and features are taken from it through the (FFT) and (DWT) operations, after that applying machine learning algorithms to it, such as the (Naive Bayes), (SVM), (KNN) algorithms and more, and comparison between them to obtain highest accuracy.

The proposed system uses the API software architectura1-style to reach the cloud, on which a cloud can retrieve the most recent data released by the IoT system. Particle Cloud is used as a connection between the IoT framework and the client cloud, as an overview of the protocol used to reach it data, as shown in Figure 3.1.

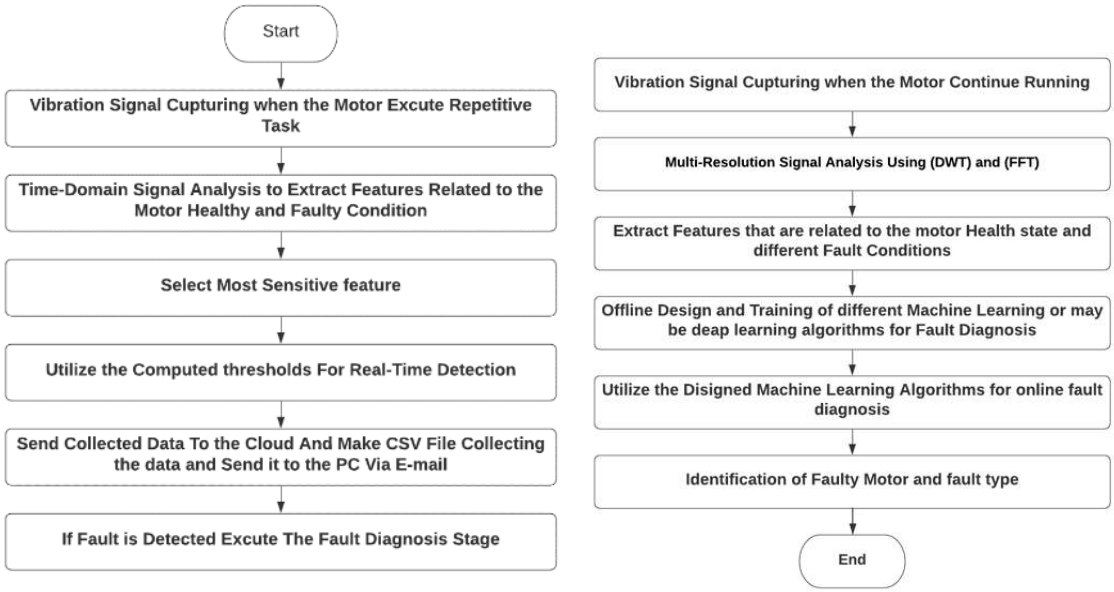


Figure 3.1. The flowchart of the first application

3.1.1 Feature extraction methods

Bearing fault detection using DWT and FFT:

- Use the wavelet-transform to input to create wavelet coefficients that allow us to discern the differences correctly.
- To better minimize noise, choose the acceptable specified threshold for each system.
- Signal analysis is used to describe physical processes and diagnose electrical machines.
- The inverse WT for the coefficients of the wavelet is used to get back to the main signal.

The estimation specifications are minimal as compared to other business instruments. Furthermore, the DWT is included in most commercial software packages, so no sophisticated calculations are needed. The Fourier-transform is applied when a signal is stationary, but the vibration signal is not necessarily stationary.

By using the time window function, a Fast Fourier Transform approach is used to depict the signal in the frequency and time domains shown in Figure 3.2.

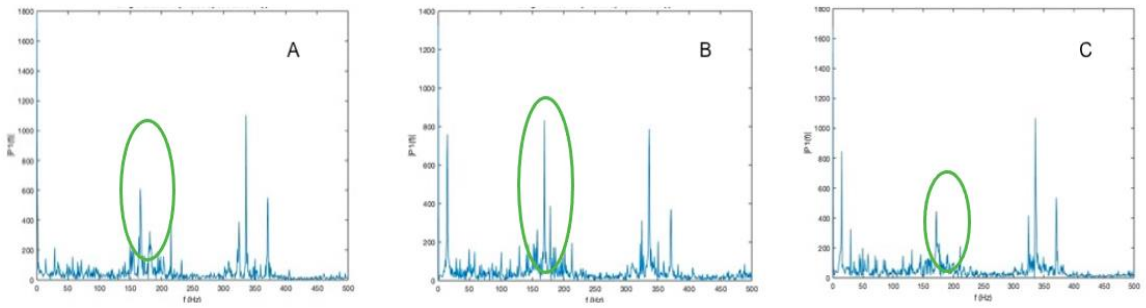


Figure 3.2. Bearing signal spectrum for (healthy), (bearing 1), (bearing 2) faults, and the detected fault points.

The discrete-wavelet-transform is defined as a result of these changes. Mother wavelets come in a variety of shapes and sizes. We will talk about the best option for producing the best outcomes. For both low pass and high pass bandwidths, the signal is separated into two equivalent bandwidths. The high-pass band can provide the signal's minor details of the concern, while the low pass bandwidth contains the relevant data. In order to obtain estimations and information, a discrete-wavelet-transform (DWT) of a signal is decomposed using a low pass filter and high pass filter in a sequential process. LP denotes the low pass filter, and HP denotes the high pass filter. The coefficients a1 (estimated from the main signal) and d1 are obtained from the first decomposition of the comprehensive shape form of the main signal. Furthermore, decomposing a_1 gives two coefficients, a_2 and d_2. As seen in Figure 3.3, a_2 can also result in other decomposition locations.

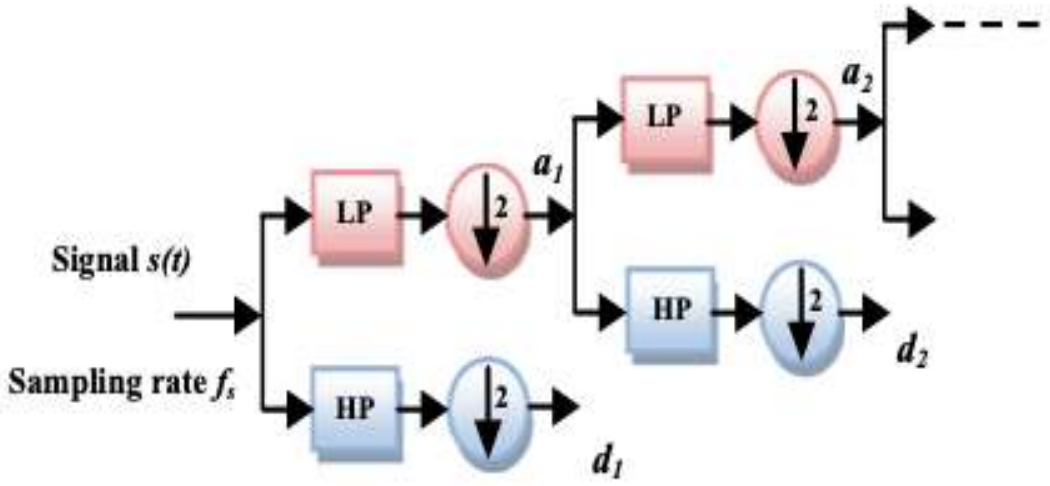
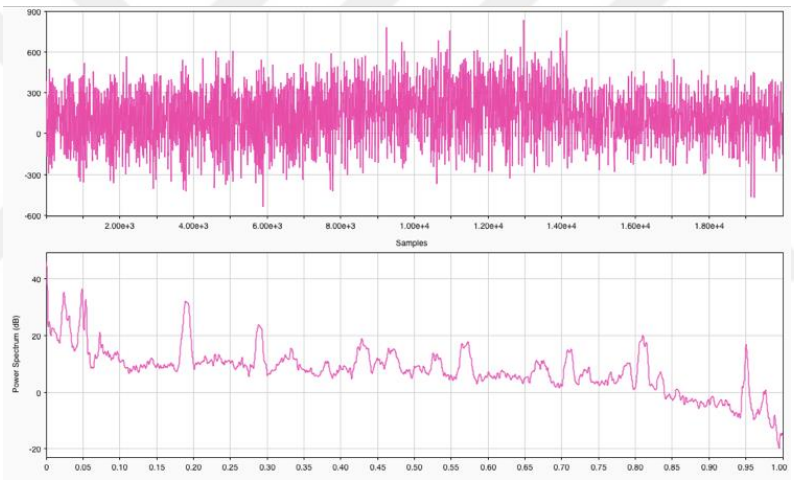


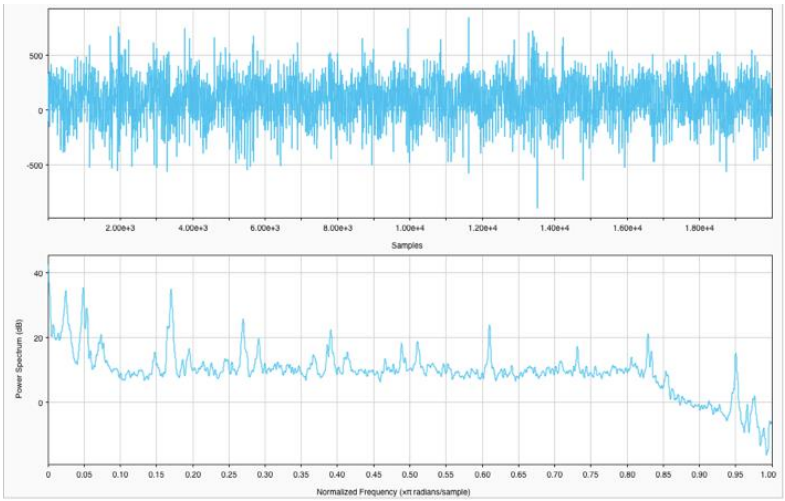
Figure 3.3. Discrete wavelet process

Fault detection is vital because stopping a device without failure will lead to a loss of sales for the company, and failing to stop a defective device can result in more damage. Any tool created for this reason needs to be extremely precise. Two methods are looked at in this segment. The first sees the fractures as intermittent, high-frequency events. As a result, irregular behavior can be observed by looking at the energy content of high-frequencies. The next one is built on resonance, which is a mechanical phenomenon. The mechanical system is designed to resist resonance to maintain normal behavior Figure 3.4 (a). If a bearing fails though the resonant-frequencies are more likely to occur as high-frequency peaks in the spectrum Figure 3.4 (b) and Figure 3.4 (c) the orientation of the frequency peaks aids in the differentiation of normal and irregular behavior.

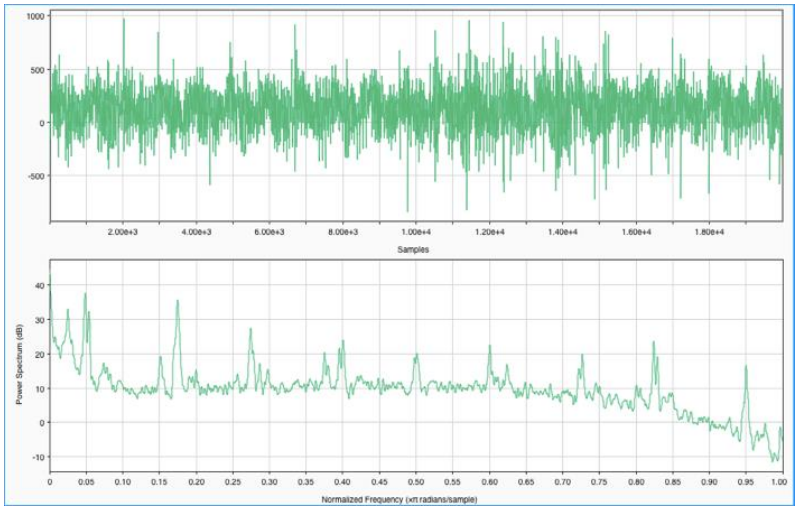
Figure 3.4 shows the frequency domain spectrum of the vibration signal token from the induction motor in each state (healthy, bearing1 and bearing 2) after Apply (mean) feature, and we can notice that the signal spectrum goes smoother when the bearing fault increases during every measurement.



(A)



(B)



(C)

Figure 3.4. Typical spectrum of vibration data after applying (mean) is featured for: (A) Healthy motor, (B) Bearing1 fault (C) Bearing2 fault

When the first method is used to analyze vibration data, the percentage of energy found that the frequencies are higher when a failure happens and low when the machine is operating normally. We performed experiments that demonstrate that a wavelet-transform (Daubechies db4) thresholds on the standardized energy found in the frequency sub-band between 0 and 1000 Hz can effectively discern between faulty and normal behavior in a ball bearing. The position of the first three frequency peaks provides good results for the second method as well.

3.1.2 Data Preprocessing and normalization

Since the data set that was used is a previously processed data set, we did not implement any of the data processing methods. We just applied the data normalization method to get data with values between 0 and 1. Learn. Preprocessing. Standard Scaler () method from the Scikit-Learn library was used to normalize our data. Data standardization aims to reduce all features to a common domain between 0 and 1 and reduce the training and testing time. Learn. Preprocessing. StandardScaler () is applied to each feature independently using Equation 3.1:

$$X_{scaled} = \frac{X - X_{mean}}{X_{std}} \tag{3.1}$$

Where X refers to the input signal, X_{mean} refers to the mean of the training samples, and X_{std} refers to the standard deviation of the training samples. The value of X_{mean} equal zero if the with mean=False, and the value of X_{std} Equal one if the with_std=False. Figure 3.5 shows the data normalization method that was used in this thesis. The variable in_Data: refers to the input features.

```
# Data normalization
normalized = preprocessing.StandardScaler().fit(in_Data)
X_scaled = normalized.transform(in_Data)
print('X_scaled=',X_scaled.shape)
```

Figure 3.5. Data normalization method using the Scikit-Learn library.

3.1.3. Data Partitioning

At this stage, the data that we obtained through various methods of feature extraction was divided into three sets: the first set is the training set with 70% of the total data we obtained, the second set is the testing set with 20% of the total data, and the third set is the validation set with 10% of the total data. The train_test_split () function from the sklearn. model_selection library was used to initially split the data into two sets (70% and 30%) for both training and testing, respectively. Figure 3.6 shows the code used to split the data. Then the test set was divided into (20% and 10%) for both testing and Validation.

```
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state=42,shuffle=True)
```

Figure 3.6. the code used to make training and testing sets

3.1.4. The building proposed models

This thesis uses machine learning models: KNN, SVM, and Naive Bayes models to detect motor fault based on detecting fault vibration signals. In this section, we will review the architecture of the models used.

3.1.5. Naive Bayes classifiers:

Are a collection of Bayes' Theorem-based categorization methods. It is a group of algorithms rather than a single algorithm. that all operate on the same concept, namely that each pair of characteristics identified is independent of the others.

3.1.6. Support Vector Machine:

The support vector machine algorithm's purpose is to determine an N-dimensional hyperplane that classifies the data points in a different manner.

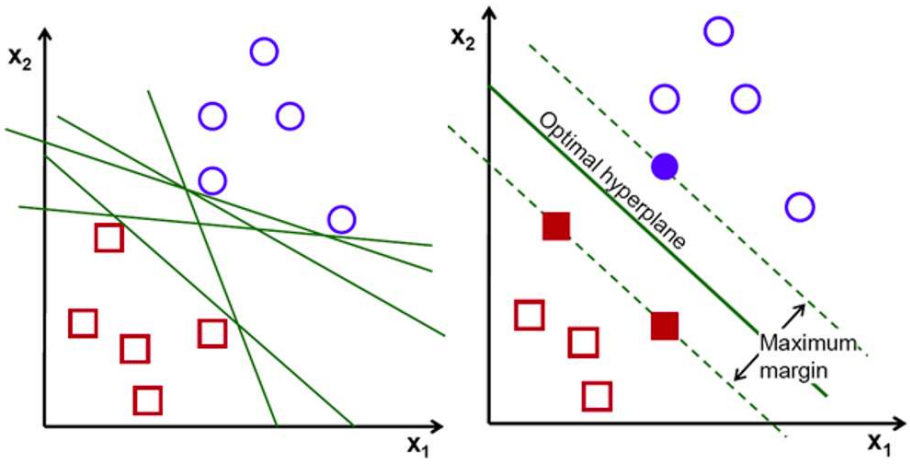


Figure 3.7. Support Vector Machine

As shown in figure 3.7. There are several hyperplanes that might be used to separate the two sets of data points. Its purpose is to determine the plane with the largest margin, that is, the plane with the greatest distance between data points. from both classes. Maximizing the margin distance reinforces the classification of subsequent data points.

3.1.7. K-nearest Neighbor:

K Nearest Neighbor (KNN) is an obvious method that is simple to implement. beginners may learn this algorithm even in the early stages of their Machine Learning studies, beginners may learn this algorithm.

3.2. A comparative Analysis of 1-D Convolutional Neural Networks for Bearing Fault Diagnosis

This method is done by taking the famous dataset that was flown from Case Western Reserve University (CWRU) database, so as (VMD) was applied to obtain the features where the data were segmented and take each part separately and then recombining them as one signal, then deep learning algorithms were applied to it via architecture 1D CNN, LSTM, and CapsuleNet several steps in terms of training dataset selection framework and network architecture design. In this part, the focus is on the data set used to train and test models. Next, a detailed overview of feature extraction methods, model steps, and details of the training process are given. Figure 3.8 represent a flowchart of the first method of study.

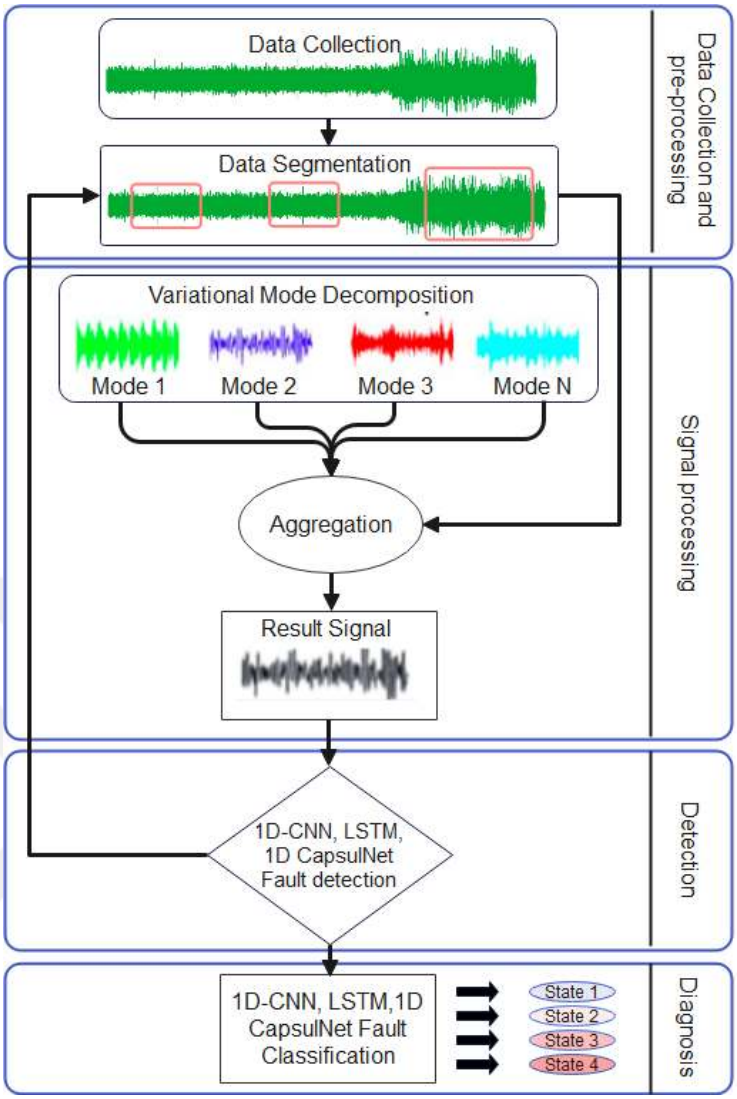


Figure 3.8. The flowchart of the for Bearing Fault Diagnosis

As can be seen from the shape of the flowchart in figure 3.8, there are five steps in the proposed method. These steps are data collection, feature extraction, feature normalization, model building, training and testing. For signal processing, VMD is applied to each raw signal. Afterward, these signals are aggregated to obtain the feature signal. The resulting signal is given to 1D-CNN and faults are classified.

3.2.3. Case Western Reserve University (CWRU) Database

The proposed VMD and 1D-CNN fault diagnosis method is assessed using the well-known Case Western Reserve University (CWRU) database, widely used for validation purposes in the literature [28]. The approach that has been proposed has been tested on accelerometer data to drive-end to 10 faults type with different intensities: inner race, Ball and outer race shown in Table. 3.1.

Table 3.1. Types of Bearing Fault

Fault	Label
Ball	C1
Inner Race	C2
Normal	C3
Outer Race	C4

CWRU empirical setup, shown in Figure 3.9, comprises a 1.49 kW (2HP) 3-phase dependence electrical motor rotating a pole on which an encoder and torque energy convertor is installed.

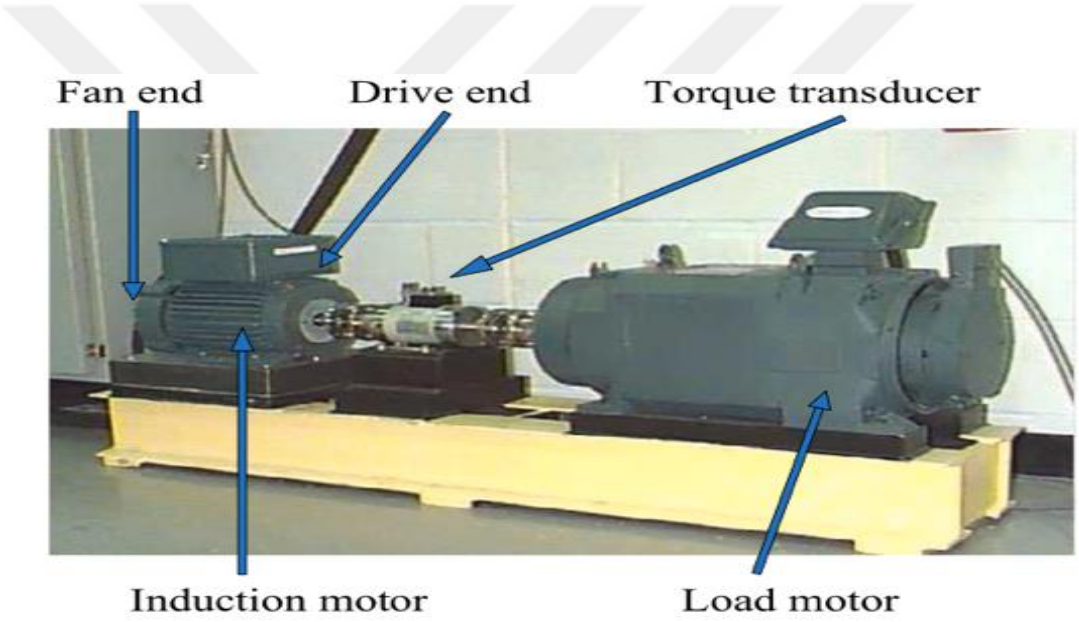


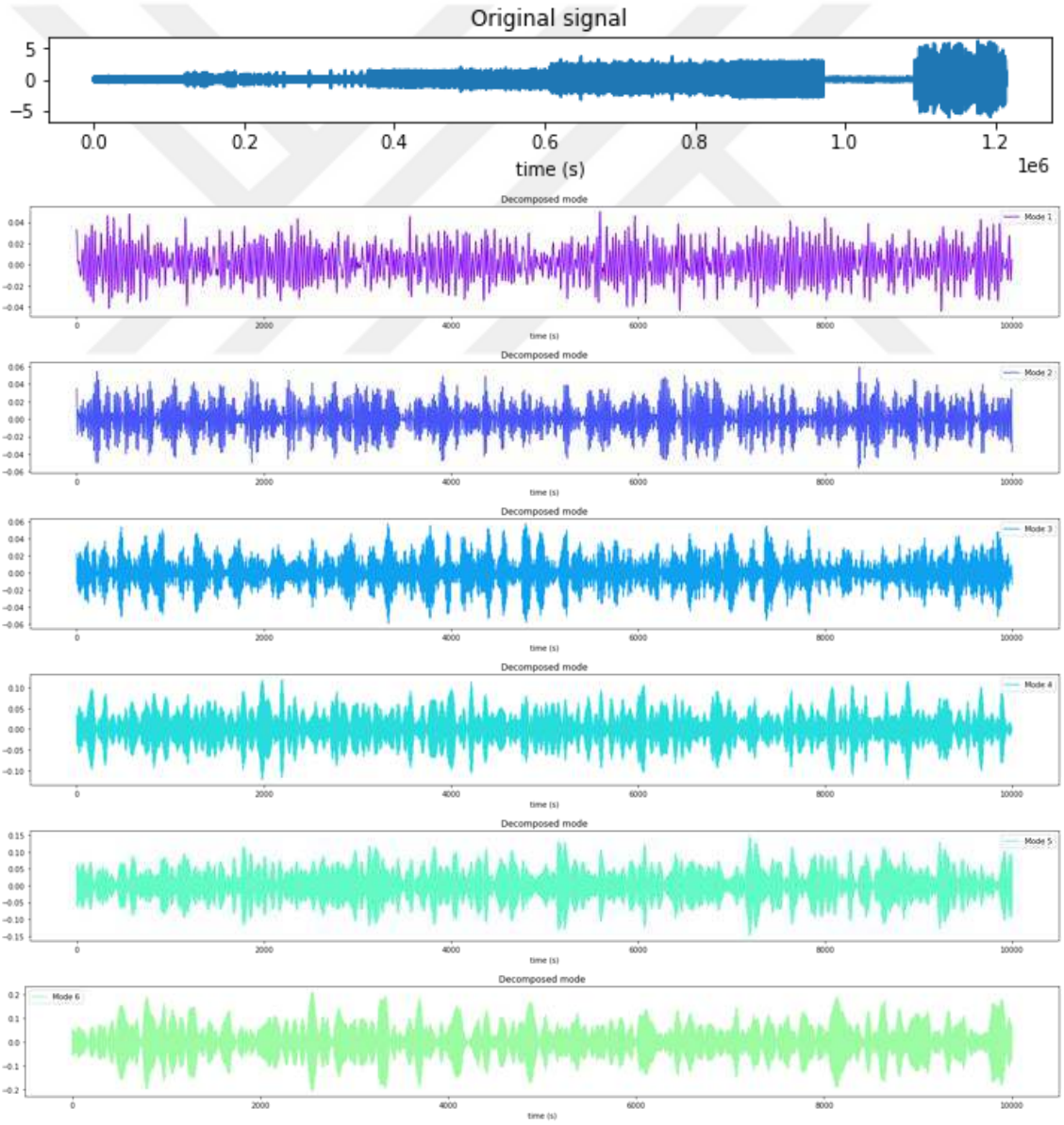
Figure 3.9. Bearing fault simulation testbed. CWRU test bench.

Tools that are used comprise a loading motor, induction motor, and a hub connected to it. CWRU dataset is enhanced with (1HP) operating mode at the speed of 1730 rpm and a sampling frequency of 12-kHz.

The above-mentioned data are segmented, as shown in the technique flowchart Figure 3.8. Data splitting with a crossover and a frame length of 10000 samples for every sub-signal constitutes pre-processing in this situation. This duration is long enough to preserve bearing-fault characteristics, while expanding the number of patterns, the overall number of segments per fault increased to 48.

3.2.2. Fault Detection based-on variational Mode Decomposition

Signal handling is the stage of selection to deal with the issue of obtaining signals damaged by harmonics and noise, as described above in the cutting-edge testimony. This is especially true when vibration signals are linked to mechanical components that produce pulses of decreased amplitude [76]. To separate these elements, signal processing methods are used. The VMD is a member of the pliable mode decomposition series. It was undoubtedly created to enhance the VMD and has since evolved into a selection method for nonlinear and non-fixed data analysis for a wide range detection of applications. VMD was able to break down complicated signals into several steady signals, which is useful. Figure 3.9 represents different modes of the signal using VMD.



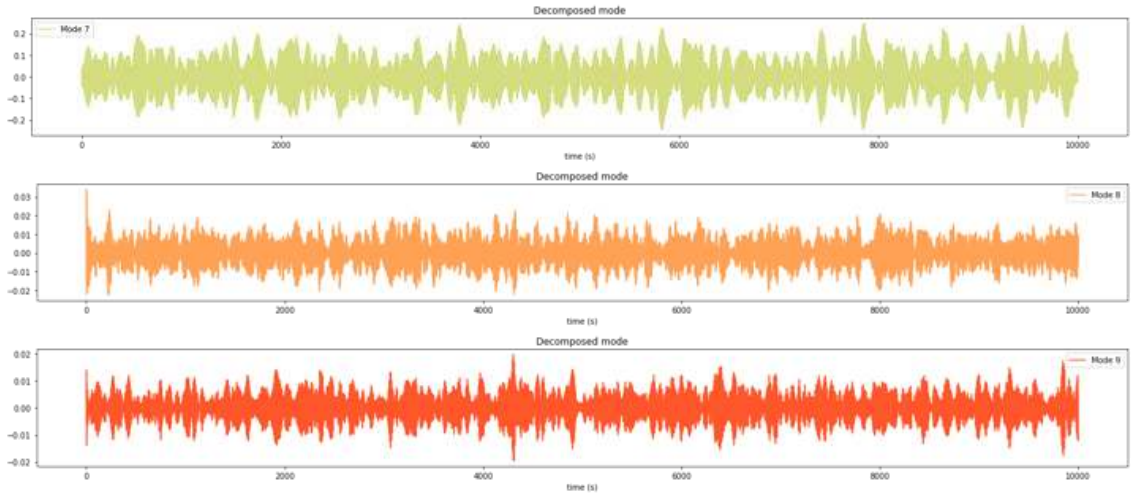


Figure 3.10. VMD modes

As shown in Figure 3.10, different modes are obtained from original signal by using VMD method. Each of this signal are aggregated to obtain feature signal for classification methods.

3.2.3. 1D CNN architecture

CNNs are widely used in human activity recognition studies. 1D CNN is one of the useful algorithms in analyzing bearing fault signals in the literature. The 1D CNN architecture proposed in this thesis is composed of two convolutional layers. Each convolutional layer consists of 16 filters and 3 kernel sizes. The feature map size was 16 x 1 at the output of max pooling using the VMD method. A dropout of 0.2 has been implemented to prevent overfitting

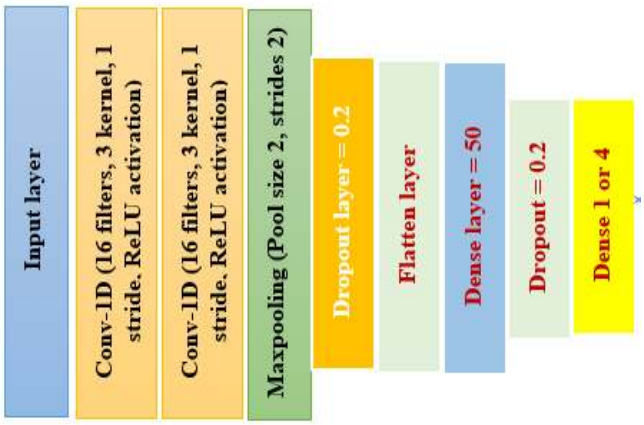


Figure 3.11. The 1D CNN architecture is used in this thesis

Figure 3.11 shows the 1D CNN architecture proposed in this thesis.

Table 3.2 shows the parameters of the layer using multi-classification (arousal-valence model). Also, APPENDIX- 1 shows the main code of 1D CNN using the VMD method. And Appendix- 2 shows the testing process for the same model

Table 3.2. The layer parameters of the 1D-CNN model VMD

Layer	VMD	
	Output	Param
Input	(None, 20,16)	-
Conv1D	(None, 20, 16)	64
Conv1D	(None, 20, 32)	784
Conv1D	(None, 20, 32)	1568
Conv1D	(None, 20, 32)	3104
Max Pooling	(None, 10, 32)	0
Flatten	(None, 320)	0
Dense	(None, 50)	16050
Dropout	(None, 50)	0
Dense	(None, 20)	1020
Activation	(None, 20)	0
Total params		22590
Trainable params		22590
Non-trainable params		0

3.2.4. LSTM architecture

LSTM networks are part of RNNs that learn patterns from time-series data. LSTM networks are used in many applications such as speech recognition, weather forecasting, or stock market forecasting. Also, LSTMs networks have been used extensively in classifying vibration signals. For example, Tan et al. [77] introduced an approach based on a combination of LSTM and CNN to detect coronary artery disease (CAD) using ECG signals. LSTM differs from RNN in solving the vanishing gradient problem. The architecture of LSTM networks consists of several cells and each cell consists of three gates, namely forget, input, and output gates. Each cell acts as a memory to store, pass, or erase information. In this thesis, there are two-layer LSTM networks, each one consisting of 16 units, with the ReLU activation function. This is followed by a Dropout

layer of 0.1 to prevent overfitting. The next layer is flattening. The last layer named as dense layer consists of 100 neurons and then the output layer with sigmoid activation in binary classification and SoftMax in multi-classing.

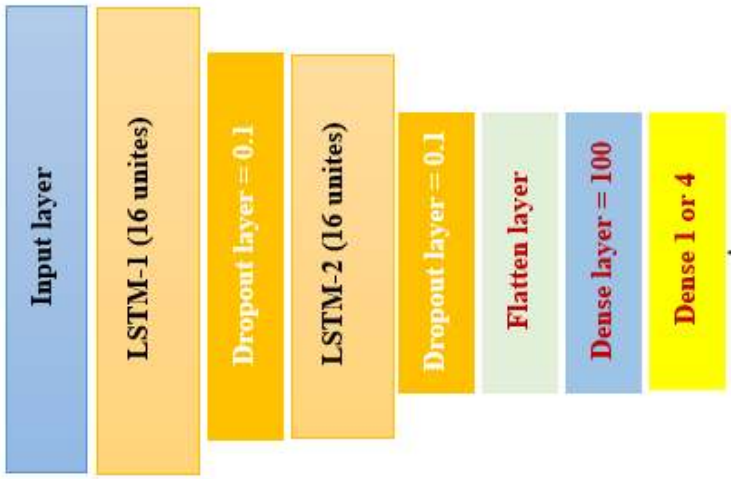


Figure 3.12. The LSTM model is used in this thesis

Table 3.3 lists the parameters of all layers implemented using the SF method and multi-classing.

Table 3.3. The layer parameters of the LSTM model using VMD

Layer	**VMD**	
	Output	Param
Input	(None, 20, 32)	-
LSTM 1	(None, 20, 32)	4352
LSTM 2	(None, 32)	8320
Dropout	(None, 32)	0
Flatten	(None, 32)	0
Dense	(None, 100)	3300
Dense	(None, 50)	5050
Dense	(None, 20)	1020
Total params		22042
Trainable params		22042
Non-trainable params		0

3.2.5. Convolutional Layers

The function of the convolutional layer includes the mode parameter of the same border, which adds zero-padding around the input to produce the feature map of the same size as layer output. For every layer deeper into the network, a larger kernel is used. Deeper layers extract more class-specific and complex information, which requires a more significant number of trainable parameters. Figures 3.12 show that the convolutional layers have kernels of sizes 3, Consequently, the receptive-field size is (3×3×1) refers to the depth of the signal input matrix

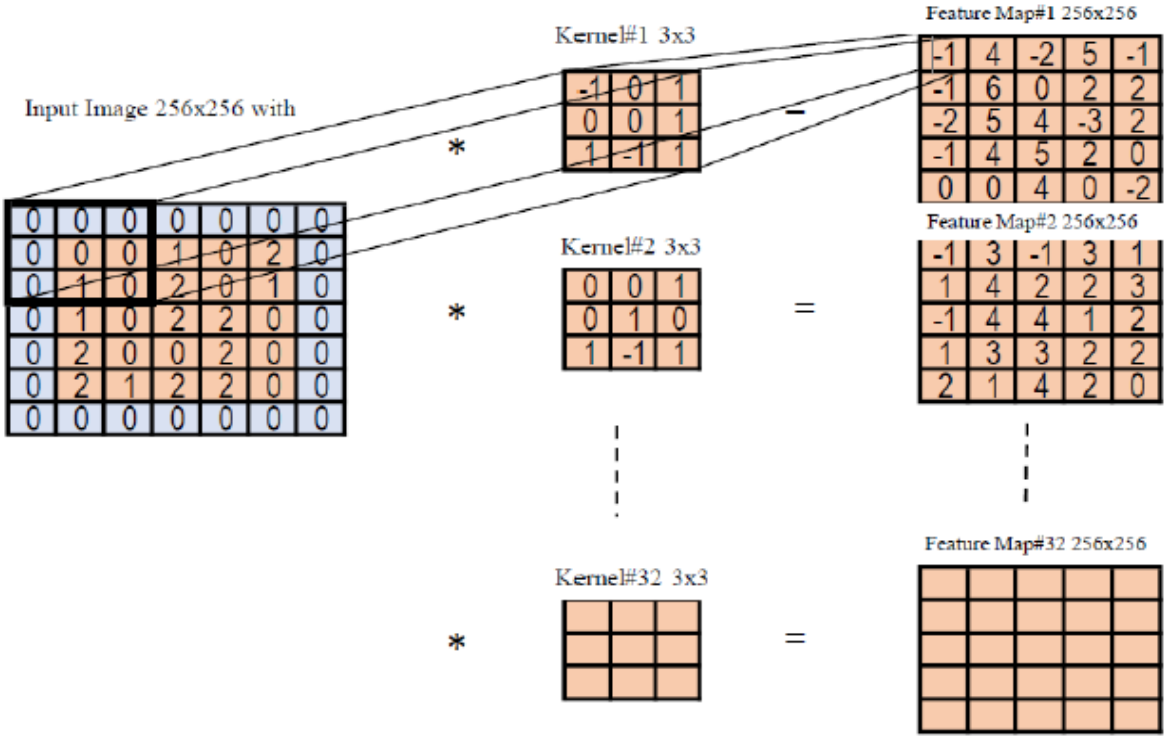


Figure 3.13. Feature Mapping from Input using 3x3 kernels.

As shown in figure 3.13, a kernel is a matrix of trainable weights; hence the total number of weights depends on its size. A single kernel of size (3×3×1) in the first convolutional layer contains 9 weights. In this convolutional layer, 32 kernels, therefore, contain 9×32 = 288 weights. Hence, every kernel adds additional trainable weights to the neural network. The number of weights for each convolutional layer is calculated using the equation 3.2:

$$No.of\ weights = (IF * K * D) + B \tag{3.2}$$

Where IF is the number of input features, K is the number of the kernels, D is the dimension of the kernel, and (B) is the bias value.

3.2.6. ReLU Activation Function Layers

Each convolutional layer in CNN is followed by a nonlinear activation function (or a piecewise linear) function. The main objective of the activation function is to squash the real-valued input to produce a non-linear output within a small range such as [0; 1] and [-1; 1]. After the weight layers, a nonlinear function must be applied to enable neural networks to learn nonlinear mappings. Stacking many weight layers is comparable to linear mapping from the input domain to the output domain in the absence of nonlinearities. As illustrated in equation 3.3 and

$$y = \begin{cases} 0 & if\ x < 0 \\ x & if\ x > 0 \end{cases} \tag{3.3}$$

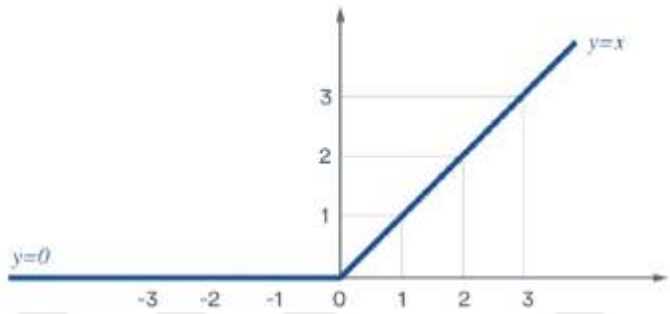


Figure 3.14. Relay activation function

Figure 3.14, the network's nonlinear activation function is a rapid and straightforward computation activation function that maps a negative input to a 0 and leaves a positive input intact.

3.2.5. Max-Pooling Layers

A pooling layer is a type of neural network layer, which has no learnable parameters. As in convolutional layers, the max-pooling operation needs to specify the size of the pooled region and the stride. In the modified 1D CNN, we use a max-pooling window 2x2 with a stride of 2. It down-samples the input features spatial size to half by selecting the maximum values within the pooling window. The max-pooling layers make network performance invariant to the small translations. Together with a convolutional layer, they are essential components of convolutional neural networks.

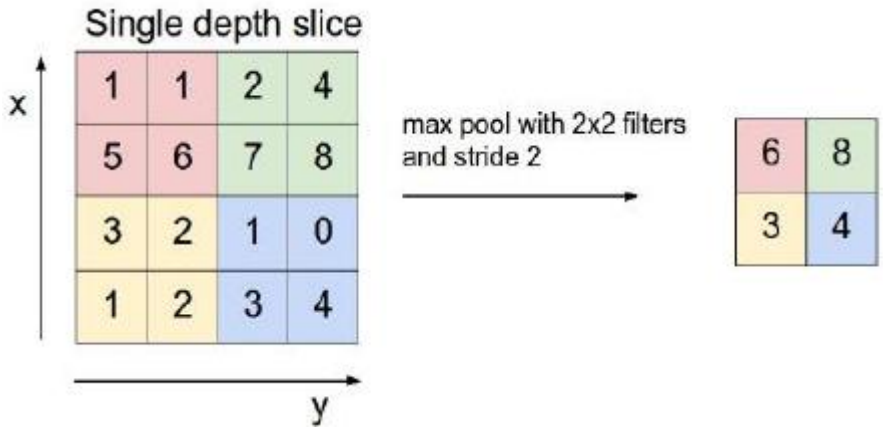


Figure 3.15. Max pooling layer

Figure 3.15 illustrates how an input feature 4x4 is minimized to half by using the max-pooling function.

3.2.6. Training our models

This thesis uses Python with Jupiter notebook software packages, deep learning, and parallel computing toolboxes to train and test the modified 1D CNN, LSTM, and hybrid 1D CNN + LSTM models' architecture to recognize induction motor fault using CWRU dataset. Besides, Python at colab cloud platform. The training process included the following steps. Before training the models, many training options were identified until we reached the options shown in Table 3.4, which are the best options for training the proposed models.

Table 3.4. Training Options of the Modified models.

Parameter	Value
Optimization Algorithm	Adam
Initial Learn Rate	1e-7
Max Epochs	100
Min Epochs	30
Learning rate	0.001
Shuffle	True
Batch size	42
metrics	Accuracy
Loss function	sparse_categorical_crossentropy

3.2.7. Evaluation and Classification

The random subsampling approach ensures learning reliability; the dataset is divided into random training / testing subsets. To guarantee more excellent dependability of predicting outcomes, training data are separated into validation and training for the cross-validation approach of the k-folds, as illustrated in Table 3.5. The evaluation is based on universal indicators like accuracy and the confusion matrix.

Table 3.5. test/train cross-validation procedure

Training Data					Test Data	
Validation	Train	Train	Train	Train	Test	Fold_1
Train	Validation	Train	Train	Train	Test	Fold_2
Train	Train	Validation	Train	Train	Test	Fold_3
Train	Train	Train	Validation	Train	Test	Fold_4
Train	Train	Train	Train	Validation	Test	Fold_5
						Average evaluation

4. EXPERIMENTAL RESULTS

In this section, we focused on the results we obtained using the modified three models. We calculated the confusion matrix. The performance of the models is measured in terms of global F1-score, recall, precision, sensitivity, specificity, and accuracy. Finally, we make a comparison between the modified models and other related works.

4.1. Cloud-based bearing fault diagnosis of induction motors

Data from an actual experimental-setup was used to validate the performance of the proposed method. An asynchronous motor with 0.37 kW provided vibration signals at various operations-frequencies for this purpose, shown in Figure 4.1.

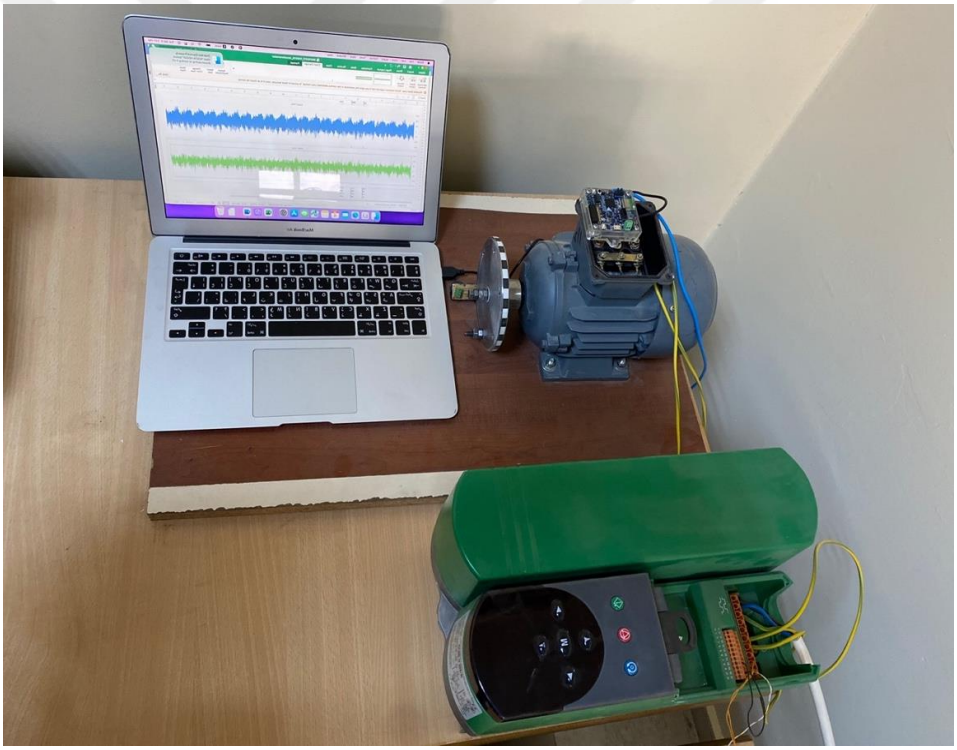


Figure 4.1. Experimental Setup

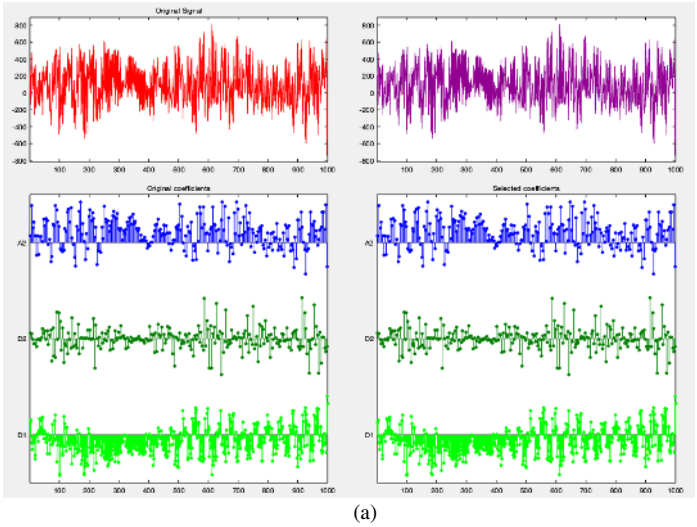
A disk causes the eccentricity flaw fixed to the motor shaft. A discrepancy was created by first adding a screw to the first hole and then to the next hole in the same disc. The vibration signals were obtained using an NI 6211 model data processing card. The sampling rate of 10 kHz has been chosen. The parameters are given in Table 4.1.

Table 4.1. Motor Parameters used in experiments

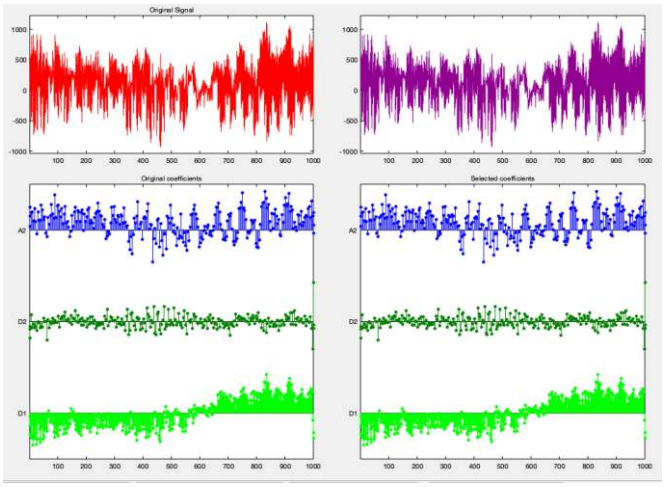
Parameter	Value
Power	0.37 kW
Full load current	1.2 A
Supply frequency	50 Hz
Number of poles	4
Full load speed	1390 rpm
Supply voltage	380 V

For specific applications, simply detecting the fault isn't enough, it's also necessary to pinpoint its location. A fault in a bearing can be found in one of three places, the ball, the outer race or the inner race. As a result, the fault locating can be viewed as a classification challenge, with each class representing one fault spot. Appropriate and accurate functionality should be derived from the data to achieve high classification accuracy. Wavelets-based techniques are discussed and adapted to vibration data in this article. The signals are decayed using DWT decay, and a Bayesian classifier is used to classify them into different classes, each describing a fault set of various types.

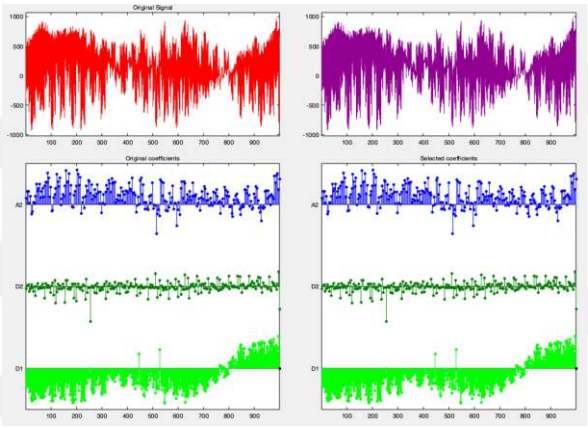
The next move is to choose the proper mother wavelet for DWT-based features. The best features derived from the DWT are considered the root-mean-square (RMS) and the norm. Sym6, Sym5, Sym4, Db6, Db5 and Db4 are all feasible alternatives for the mother wavelet. In this article, Db4 was applied to the database, and a perfect classification was accomplished.



(a)



(b)



(c)

Figure 4.2. Discrete Wavelet Transform Features (Db4) for (healthy), (bearing1 fault), (Bearing2 fault) motor respectively.

Figure 4.2. Shows the classification results for different mother-wavelets extracted using the obtained features from the Discrete Wavelet Transform of the data.

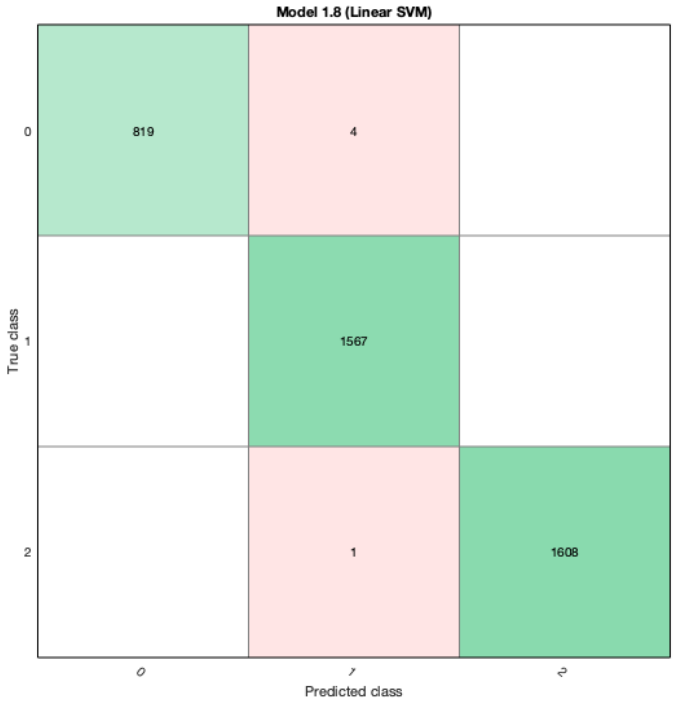


Figure 4.3. Confusion matrix (SVM classifier)

As shown in Figure 4.3. Just five samples are misclassified. The accuracy rating was found to be 99.9%. The classifier's efficiency was used in other machine learning systems. For this, the Matlab Classification-Learner tool was used. Figure 4.4 displays the comparative effects.

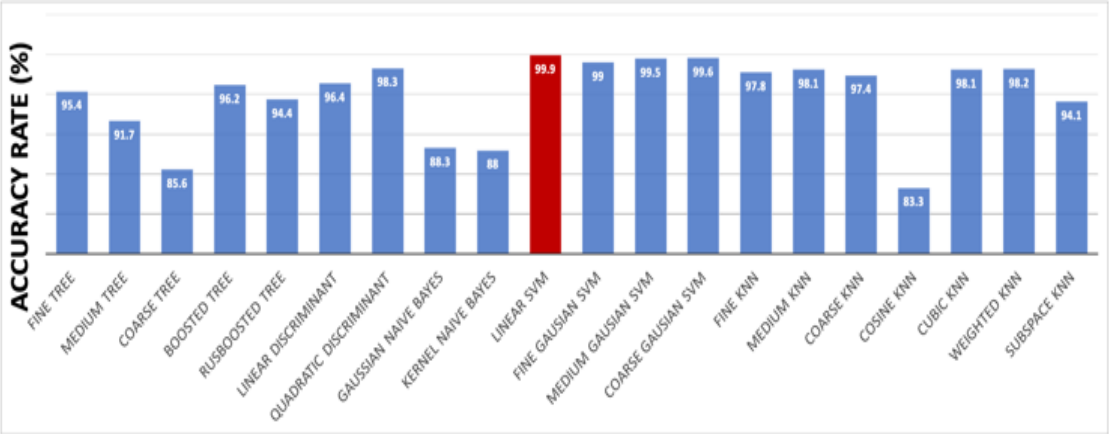


Figure 4.4. Different machine learning classification method

Figure 4.4. shows Linear SVM provided the best results. Another benefit of the suggested approach is that, since the frequency is calculated from the existing signals, the frequencies linked to the fault are automatically received.

4.2. A comparative analysis of 1-D Convolutional Neural Network for Bearing Fault Diagnosis

Bearing-faults detection has been accomplished with 100% precision for the operating setting of (1HP). The recommended fault detection method succeeded as well as effective in distinguishing healthy and balanced and malfunctioning states. The immediate result of the VMD used as a filter eliminates the lead-in setting, which is normal to healthy and faulty states. The 1D-CNN is used for enabling better feature removal, and fivefold cross recognition is utilized as shown in Figure 4.5.

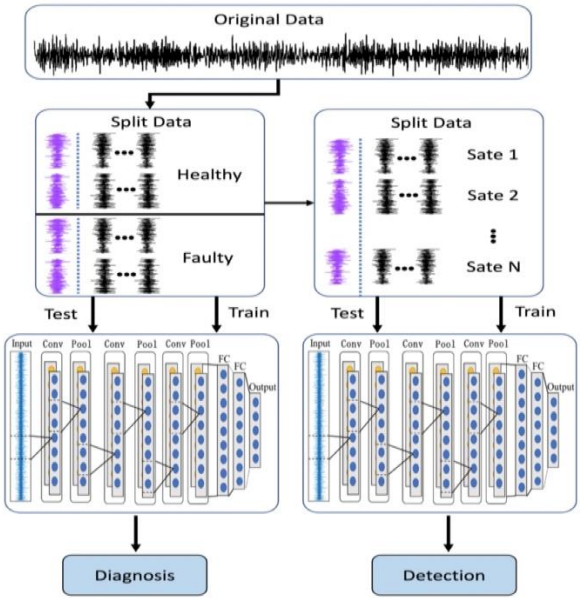


Figure 4.5. Fault reveal and diagnosis

The learning of convolution layers is also appropriately evaluated versus Methods of machine learning that are well-known, called (CapsuleNet), which specifies the features of the object and its likelihood, and LSTM, which is devoted to the analysis of time-series signals as well as the ability to learn Information that is both short-term and long-term. The results that obtained in Table 4.2. Shows clearly that 1D-CNN Attain the maximum degree of diagnostic accuracy.

Table 4.2. Accuracies of different algorithms used in this process

	1D-CNN	**CapsuleNet**	**LSTM**
Accuracy	97.04%	96.17%	96.68%
F1_Score	97.9%	99.0%	97.0%
Precision	98.0%	98.0%	96.89%
Recall	97.8%	96.7%	96.9%

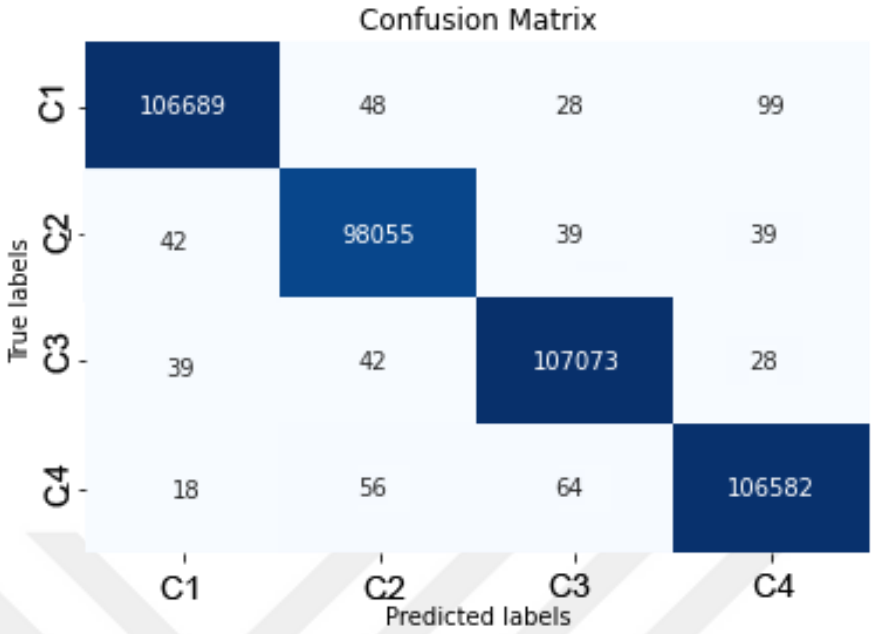


Figure 4.6. Confusion matrix for fault diagnosis using 1D-CNN

Figure 4.6. shows the confusion matrix of 1-D CNN for fault diagnosis

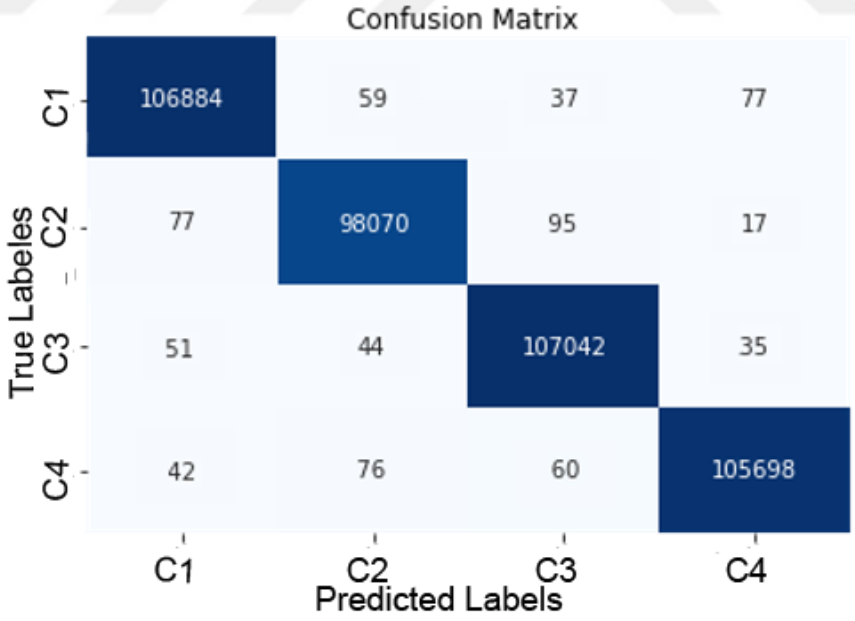


Figure 4.7. Confusion matrix for fault diagnosis using LSTM

Figure 4.7. shows the confusion matrix of LSTM for fault diagnosis

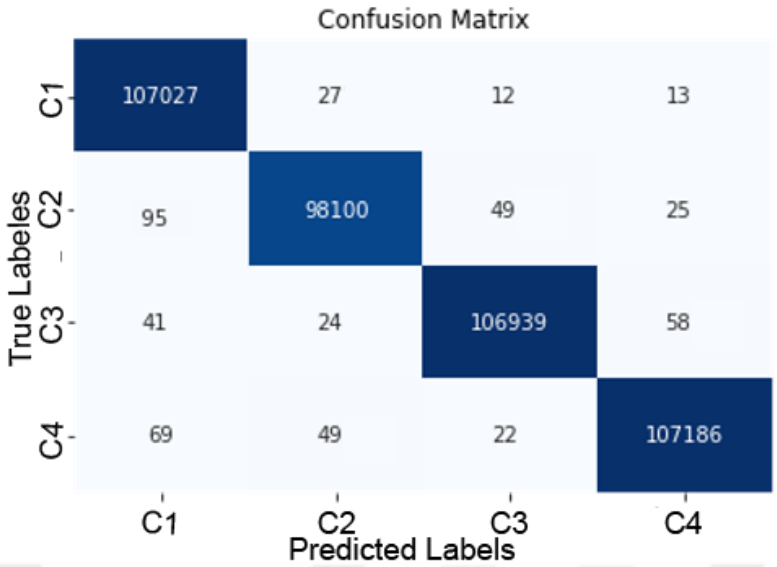


Figure 4.8. Confusion matrix for fault diagnosis using CapsuleNet

Figure 4.6. shows the confusion matrix of 1Capsule Networkfor fault diagnosis

The performances of fault diagnosis are also mentioned in Figures (4.6 & 4.7 & 4.8) in terms of the confusion matrix.

4.2.3. Training Progress Plot

After defining the training options, it's time to train the modified models using the model. Training the networks requires four predefined parameters. Model architecture (training data: input and output), number of epochs, number of batch size, and the value of validation_data

```
# Compiling the CNN
opt=keras.optimizers.Adam(
    learning_rate=0.001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
# Fitting to the training set
history = model.fit(x_train,y_train,epochs=50,batch_size=42,validation_data=(x_test, y_test))
```

Figure 4.9. Compiling and fitting 1D CNN model

The training process is done by using a Fit() function in Keras environment. This process is given in Figure 4.9

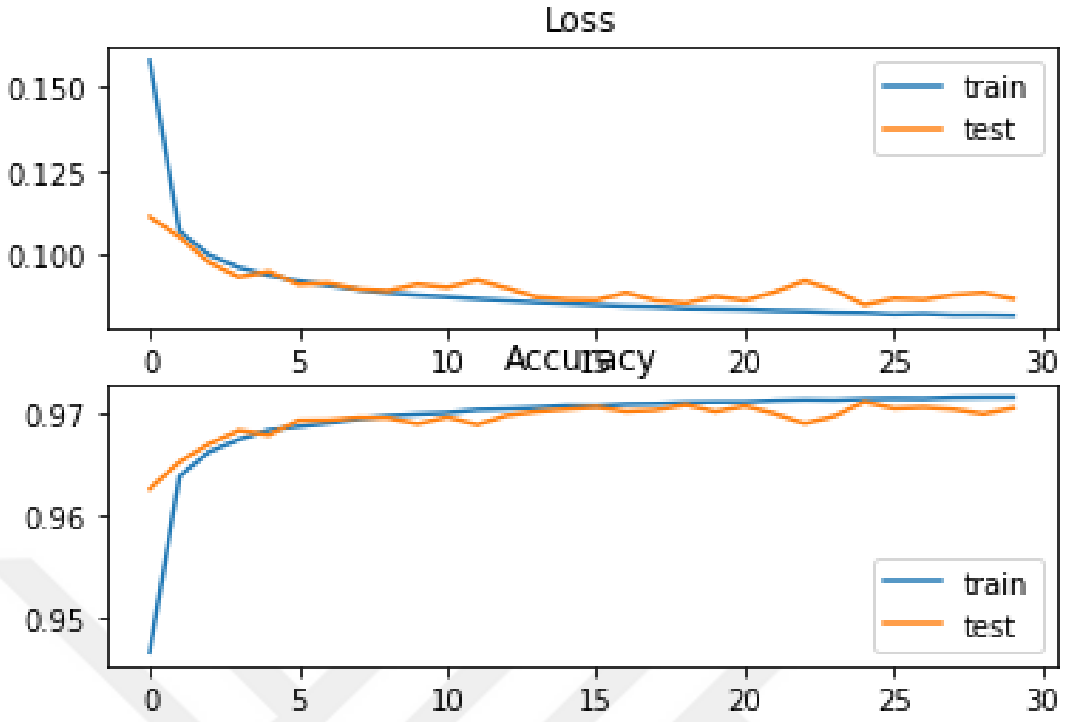


Figure 4.10. Training Progress Plot for the 1 D CNN model using VMD method.

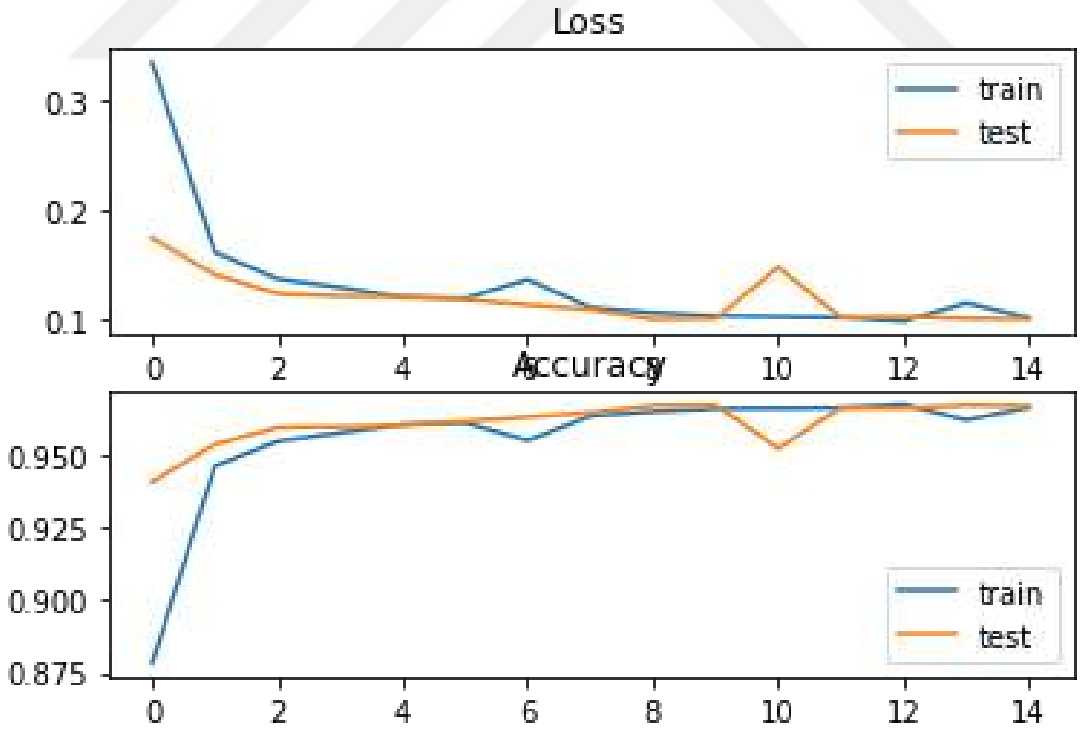


Figure 4.11. Training Progress Plot for LSTM model using VMD method.

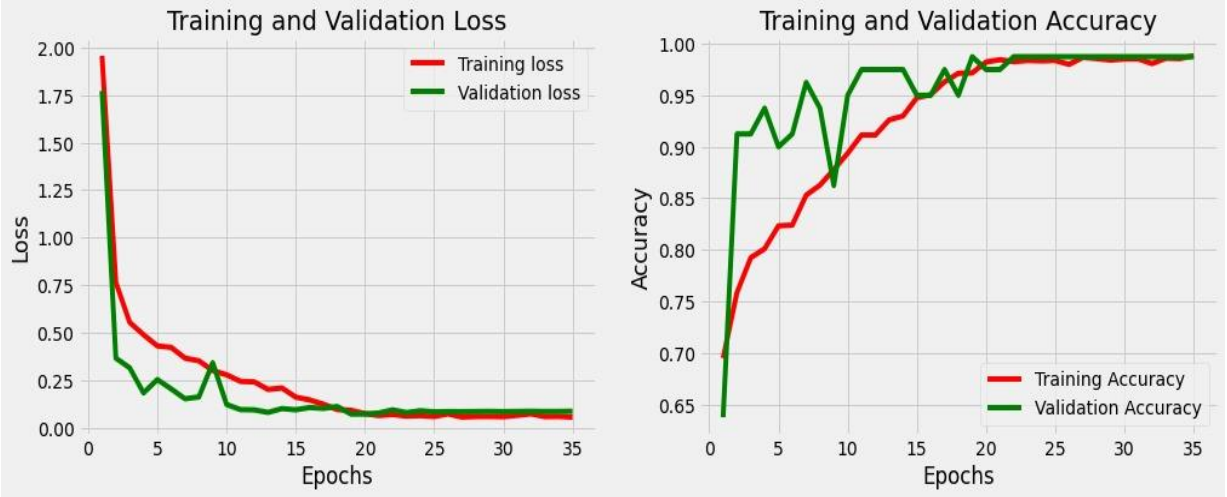


Figure 4.12. Training Progress Plot for CapsuleNet model using VMD method.

. Figures 4.10 - 4.12 represent the training curve and the loss function curve for samples of the models proposed in this thesis.

5. CONCLUSION

When faulted bearings emit vibration signals, the Discrete Wavelet Transform (DWT) has been used to detect fault features. The Fast-Fourier-Transform (FFT) is crucial in evaluating results obtained with MATLAB-Toolbox's The use of DWT as a method for detecting fault features in bearings is also sufficient on low-speed systems. However, it is recommended for DWT performance. Many of these can be monitored remotely without the need to shut down the system for repairs. As a result, this approach is recommended as an alternative technique for bearing fault detection in real-time monitoring, especially for high-speed systems.

A distinct strategy has been suggested for detecting and diagnosing bearing fault as a dominant mode cancellation filter. VMD was used that enhance fault detection. For the purposes of diagnosis, a (1D-CNN) was approved. The detection and diagnosis that are proposed for bearing faults approach were validated using the famous Case Western Reserve University experimental dataset. Significantly, the potential of the suggested diagnostic approach to identify a given fault with varying severity levels has been emphasized. The detection and diagnosis performance have also been compared to pertinent state-of-the-art related approaches. On the one hand, it has been shown that the best approach is to utilize VMD as a filter compared to traditional VMD. On the other hand, as part of the classifying feature extraction, convolution layers (1D- CNN) have been discovered better by comparing to LSTM.

Future research should look at the suggested VMD-based 1D-CNN technique for bearing fault forecast utilizing LSTM. Indeed, LSTM is devoted to monitoring the time series and has a good learning capacity of long-term and short-term relationships, which might be valuable for prediction.

REFERENCES

[1] Z. Peroutka, T. Glasberger, and M. Janda, “Main problems and proposed solutions to induction machine drive control of multisystem locomotive,” in 2009 IEEE Energy Conversion Congress and Exposition, 2009, pp. 430–437.

[2] C. Chen and C. Mo, “A method for intelligent fault diagnosis of rotating machinery,” Digital Signal Processing, vol. 14, no. 3, pp. 203–217, 2004.

[3] S. Poyhonen, P. Jover, and H. Hyotyniemi, “Signal processing of vibrations for condition monitoring of an induction motor,” in First International Symposium on Control, Communications and Signal Processing, 2004., 2004, pp. 499–502.

[4] W. R. Finley, M. M. Hodowanec, and W. G. Holter, “An analytical approach to solving motor vibration problems,” in Industry Applications Society 46th Annual Petroleum and Chemical Technical Conference (Cat. No. 99CH37000), 1999, pp. 217–232.

[5] K. K. Shukla and A. K. Tiwari, Efficient algorithms for discrete wavelet transform: with applications to denoising and fuzzy inference systems. Springer Science & Business Media, 2013.

[6] S. Abbasion, A. Rafsanjani, A. Farshidianfar, and N. Irani, “Rolling element bearings multi-fault classification based on the wavelet denoising and support vector machine,” Mechanical systems and signal processing, vol. 21, no. 7, pp. 2933–2945, 2007.

[7] G. Bucci, F. Ciancetta, E. Fiorucci, A. Fioravanti, A. Prudenzi, and S. Mari, “An IoT condition monitoring system for resilience based on spectral analysis of vibration,” in 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, 2020, pp. 38–43.

[8] M. Rausand and A. Hoyland, System reliability theory: models, statistical methods, and applications, vol. 396. John Wiley & Sons, 2003.

[9] S. Selcuk, “Predictive maintenance, its implementation and latest trends,” Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 231, no. 9, pp. 1670–1679, 2017.

[10] T. P. Carvalho, F. A. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance,” Computers & Industrial Engineering, vol. 137, p. 106024, 2019.

[11] A. Yunusa-Kaltungo and J. K. Sinha, “Effective vibration-based condition monitoring (eVCM) of rotating machines,” Journal of Quality in Maintenance Engineering, 2017.

[12] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani, and M. Petracca, “Industrial Internet of Things monitoring solution for advanced predictive maintenance applications,” Journal of Industrial Information Integration, vol. 7, pp. 4–12, 2017.

[13] D. Shyamala, D. Swathi, J. L. Prasanna, and A. Ajitha, “IoT platform for condition monitoring of industrial motors,” in 2017 2nd International Conference on Communication and Electronics Systems (ICCES), 2017, pp. 260–265.

[14] L. da Xu, W. He, and S. Li, “Internet of things in industries: A survey,” IEEE Transactions on industrial informatics, vol. 10, no. 4, pp. 2233–2243, 2014.

[15] H. Habbouche, Y. Amirat, T. Benkedjouh, and M. Benbouzid, “Bearing Fault Event-Triggered Diagnosis using a Variational Mode Decomposition-based Machine Learning Approach,” IEEE Transactions on Energy Conversion, 2021.

[16] Q. Wang, C. Yang, H. Wan, D. Deng, and A. K. Nandi, “Bearing Fault Diagnosis Based on Optimized Variational Mode Decomposition and 1-D Convolutional Neural Networks,” Measurement Science and Technology, 2021.

[17] L. Eren, T. Ince, and S. Kiranyaz, “A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier,” Journal of Signal Processing Systems, vol. 91, no. 2, pp. 179–189, 2019.

[18] R. Song and Q. Jiang, “Application of VMD Combined with CNN and LSTM in Motor Bearing Fault,” in 2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA), 2021, pp. 1661–1666.

[19] X. Wang, D. Mao, and X. Li, “Bearing fault diagnosis based on vibro-acoustic data fusion and 1D-CNN network,” Measurement, vol. 173, p. 108518, 2021.

[20] S. Hao, F.-X. Ge, Y. Li, and J. Jiang, “Multisensor bearing fault diagnosis based on one-dimensional convolutional long short-term memory networks,” Measurement, vol. 159, p. 107802, 2020.

[21] X. Liu, Q. Zhou, J. Zhao, H. Shen, and X. Xiong, “Fault diagnosis of rotating machinery under noisy environment conditions based on a 1-D convolutional autoencoder and 1-D convolutional neural network,” Sensors, vol. 19, no. 4, p. 972, 2019.

[22] X. Zhao, Y. Qin, C. He, L. Jia, and L. Kou, “Rolling element bearing fault diagnosis under impulsive noise environment based on cyclic correntropy spectrum,” Entropy, vol. 21, no. 1, p. 50, 2019.

[23] D. M. Morris, G. Uswatte, J. E. Crago, E. W. Cook III, and E. Taub, “The reliability of the wolf motor function test for assessing upper extremity function after stroke,” Archives of physical medicine and rehabilitation, vol. 82, no. 6, pp. 750–755, 2001.

[24] S. M. A. Cruz and A. J. M. Cardoso, “Stator winding fault diagnosis in three-phase synchronous and asynchronous motors, by the extended Park’s vector approach,” IEEE Transactions on industry applications, vol. 37, no. 5, pp. 1227–1233, 2001.

[25] A. Bellini, F. Filippetti, C. Tassoni, and G.-A. Capolino, “Advances in diagnostic techniques for induction machines,” IEEE Transactions on industrial electronics, vol. 55, no. 12, pp. 4109–4126, 2008.

[26] A. H. Bonnett and G. C. Soukup, “Cause and analysis of stator and rotor failures in three-phase squirrel-cage induction motors,” IEEE Transactions on Industry applications, vol. 28, no. 4, pp. 921–937, 1992.

[27] S. Karmakar, S. Chattopadhyay, M. Mitra, and S. Sengupta, “Induction motor and faults,” in Induction Motor Fault Diagnosis, Springer, 2016, pp. 7–28.

[28] S. Karmakar, S. Chattopadhyay, M. Mitra, and S. Sengupta, Induction motor fault diagnosis: approach through current signature analysis. Springer, 2016.

[29] S. Karmakar, S. Chattopadhyay, M. Mitra, and S. Sengupta, “Induction motor and faults,” in Induction Motor Fault Diagnosis, Springer, 2016, pp. 7–28.

[30] A. G. Sarigiannidis, M. E. Beniakar, P. E. Kakosimos, A. G. Kladas, L. Papini, and C. Gerada, “Fault tolerant design of fractional slot winding permanent magnet aerospace actuator,” IEEE Transactions on Transportation Electrification, vol. 2, no. 3, pp. 380–390, 2016.

[31] S. Grubic, J. M. Aller, B. Lu, and T. G. Habetler, “A survey on testing and monitoring methods for stator insulation systems of low-voltage induction machines focusing on turn insulation problems,” IEEE Transactions on Industrial Electronics, vol. 55, no. 12, pp. 4127–4136, 2008.

[32] A. W. Galli and M. D. Cox, “Temperature rise of small oil-filled distribution transformers supplying nonsinusoidal load currents,” IEEE transactions on Power Delivery, vol. 11, no. 1, pp. 283–291, 1996.

[33] A. H. Bonnett, “Operating temperature considerations and performance characteristics for IEEE 841 motors,” in Record of Conference Papers. Industry Applications Society Forty-

Seventh Annual Conference. 2000 Petroleum and Chemical Industry Technical Conference (Cat. No. 00CH37112), 2000, pp. 77–89.

[34] Y. Kaya, M. Kuncan, K. Kaplan, M. R. Minaz, and H. M. Ertunç, "Classification of bearing vibration speeds under 1D-LBP based on eight local directional filters," Soft Computing, vol. 24, no. 16, pp. 12175–12186, 2020.

[35] T. Kärkkäinen and P. Silventoinen, "Necessity of active condition monitoring in power electronic converters," 2013.

[36] R. Dyson and D. L. Doel, "CF6-80 condition monitoring-the engine manufacturer's involvement in data acquisition and analysis," in 20th joint propulsion conference, 1984, p. 1412.

[37] G. Olmo, G. Loizzi, and E. A. Gumiero, "Master's Degree in Biomedical Engineering".

[38] G. Olmo, E. Cosenza, and E. G. Tantillo, "Design of an Inertial Multi-Sensor Network for the Monitoring of the Heart Rate During Sleep using Ballistocardiographic Signals".

[39] T. Mustafa and A. Varol, "Review of the Internet of Things for Healthcare Monitoring," in 2020 8th International Symposium on Digital Forensics and Security (ISDFS), 2020, pp. 1–6.

[40] T. Mustafa and A. Varol, "Review of the Internet of Things for Healthcare Monitoring," in 2020 8th International Symposium on Digital Forensics and Security (ISDFS), 2020, pp. 1–6.

[41] M. Karabatak, T. Mustafa, and C. Hamaali, "Remote Monitoring Real Time Air pollution - IoT (Cloud Based)," in 2020 8th International Symposium on Digital Forensics and Security (ISDFS), 2020, pp. 1–6. doi: 10.1109/ISDFS49300.2020.9116339.

[42] M. Islam, T. Ahmed, S. S. Mostafa, M. S. U. Yusuf, and M. Ahmad, "Human emotion recognition using frequency & statistical measures of EEG signal," in 2013 International Conference on Informatics, Electronics and Vision (ICIEV), 2013, pp. 1–6.

[43] H. J. Nussbaumer, "The fast Fourier transform," in Fast Fourier Transform and Convolution Algorithms, Springer, 1981, pp. 80–111.

[44] H. Hindarto and S. Sumarno, "Feature extraction of electroencephalography signals using fast fourier transform," CommIT (Communication and Information Technology) Journal, vol. 10, no. 2, pp. 49–52, 2016.

[45] Z. Ye, B. Wu, and A. R. Sadeghian, "Signature analysis of induction motor mechanical faults by wavelet packet decomposition," in APEC 2001. Sixteenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No. 01CH37181), 2001, vol. 2, pp. 1022–1029.

[46] W. Ting, Y. Guo-Zheng, Y. Bang-Hua, and S. Hong, "EEG feature extraction based on wavelet packet decomposition for brain computer interface," Measurement, vol. 41, no. 6, pp. 618–625, 2008.

[47] J. Tang, X. Yin, Z. Zhang, D. You, L. Ye, and Z. He, "Fault location in distribution networks base on the use of wavelet packet analysis," in 2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011, pp. 825–830.

[48] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11–26, 2017.

[49] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015.

[50] Z. Akkus, A. Galimzianova, A. Hoogi, D. L. Rubin, and B. J. Erickson, "Deep learning for brain MRI segmentation: state of the art and future directions," Journal of digital imaging, vol. 30, no. 4, pp. 449–459, 2017.

[51] Y. LeCun et al., “Comparison of learning algorithms for handwritten digit recognition,” in International conference on artificial neural networks, 1995, vol. 60, pp. 53–60.

[52] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, “1-d convolutional neural networks for signal processing applications,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 8360–8364.

[53] J. Xu, B. Wang, J. Li, C. Hu, and J. Pan, “Deep learning application based on embedded GPU,” in 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), 2017, pp. 1–4.

[54] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, “A guide to convolutional neural networks for computer vision,” Synthesis Lectures on Computer Vision, vol. 8, no. 1, pp. 1–207, 2018.

[55] M. Yani, “Application of transfer learning using convolutional neural network method for early detection of terry’s nail,” in Journal of Physics: Conference Series, 2019, vol. 1201, no. 1, p. 012052.

[56] S. Toraman, “Automatic recognition of preictal and interictal EEG signals using 1D-capsule networks,” Computers & Electrical Engineering, vol. 91, p. 107033, 2021.

[57] T. Kim, J. Lee, and J. Nam, “Sample-level CNN architectures for music auto-tagging using raw waveforms,” in 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), 2018, pp. 366–370.

[58] S. Abdoli, P. Cardinal, and A. L. Koerich, “End-to-end environmental sound classification using a 1D convolutional neural network,” Expert Systems with Applications, vol. 136, pp. 252–263, 2019.

[59] A. Akbari Asanjan, T. Yang, K. Hsu, S. Sorooshian, J. Lin, and Q. Peng, “Short-term precipitation forecast based on the PERSIANN system and LSTM recurrent neural networks,” Journal of Geophysical Research: Atmospheres, vol. 123, no. 22, pp. 12–543, 2018.

[60] Y. Tsai, Y. Zeng, and Y. Chang, “Air Pollution Forecasting Using RNN with LSTM,” in 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2018, pp. 1074–1079. doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00178.

[61] L. Tai and M. Liu, “Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not,” arXiv preprint arXiv:1612.07139, 2016.

[62] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model.,” in Interspeech, 2010, vol. 2, no. 3, pp. 1045–1048.

[63] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in 2013 IEEE international conference on acoustics, speech and signal processing, 2013, pp. 6645–6649.

[64] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” in Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1700–1709.

[65] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM neural networks for language modeling,” 2012.

[66] R. Yuan et al., “Daily runoff forecasting using ensemble empirical mode decomposition and long short-term memory,” Frontiers in Earth Science, vol. 9, p. 129, 2021.

[67] R. S. A. Usmani, T. R. Pillai, I. A. T. Hashem, M. Marjani, R. Shaharudin, and M. T. Latif, “Air pollution and cardiorespiratory hospitalization, predictive modeling, and analysis using

artificial intelligence techniques," Environmental Science and Pollution Research, pp. 1–13, 2021.

[68] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, "What is a good evaluation measure for semantic segmentation," IEEE PAMI, vol. 26, no. 1, 2004.

[69] W. Zhu, N. Zeng, and N. Wang, "Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations," NESUG proceedings: health care and life sciences, Baltimore, Maryland, vol. 19, p. 67, 2010.

[70] F. B. Jada, A. M. Aibinu, and A. J. Onumanyi, "Performance Metrics for Image Segmentation Techniques: A Review," no. October, pp. 344–348, 2015.

[71] H. B. Wong and G. H. Lim, "Measures of diagnostic accuracy: sensitivity, specificity, PPV and NPV," Proceedings of Singapore healthcare, vol. 20, no. 4, pp. 316–318, 2011.

[72] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow. Packt Publishing Ltd, 2019.

[73] E. Smistad, T. L. Falch, M. Bozorgi, A. C. Elster, and F. Lindseth, "Medical image segmentation on GPUs–A comprehensive review," Medical image analysis, vol. 20, no. 1, pp. 1–18, 2015.

[74] H. L. Khor, S.-C. Liew, and J. M. Zain, "A review on parallel medical image processing on GPU," in 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS), 2015, pp. 45–48.

[75] T. Carneiro, R. V. M. da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. de Albuquerque, and P. P. Reboucas Filho, "Performance analysis of google colaboratory as a tool for accelerating deep learning applications," IEEE Access, vol. 6, pp. 61677–61685, 2018.

[76] S. Ma, B. Lv, C. Lin, X. Sheng, and X. Zhu, "EMG signal filtering based on variational mode decomposition and sub-band thresholding," IEEE journal of biomedical and health informatics, vol. 25, no. 1, pp. 47–58, 2020.

[77] J. H. Tan et al., "Application of stacked convolutional and long short-term memory network for accurate identification of CAD ECG signals," Computers in biology and medicine, vol. 94, pp. 19–26, 2018.

APPENDICES

APPENDIX- 1: MENDELEY ADD-ONS USAGE FOR CITATIONS

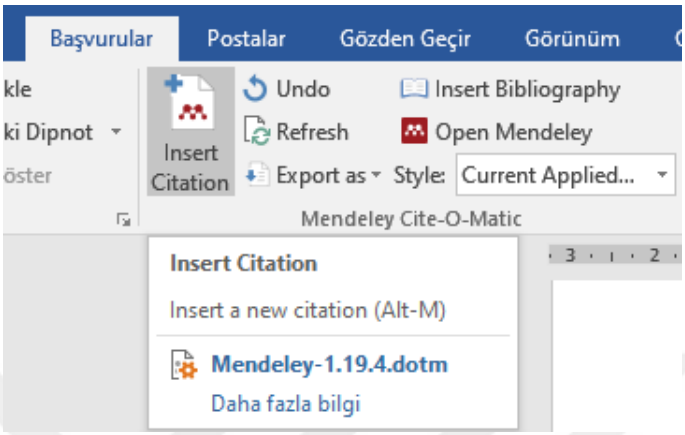


Figure A1.1. Screenshot of Mendeley macro icons in MS Word

APPENDIX- 2: SELECTION OF CITATION METHOD IN MENDELEY PROGRAM

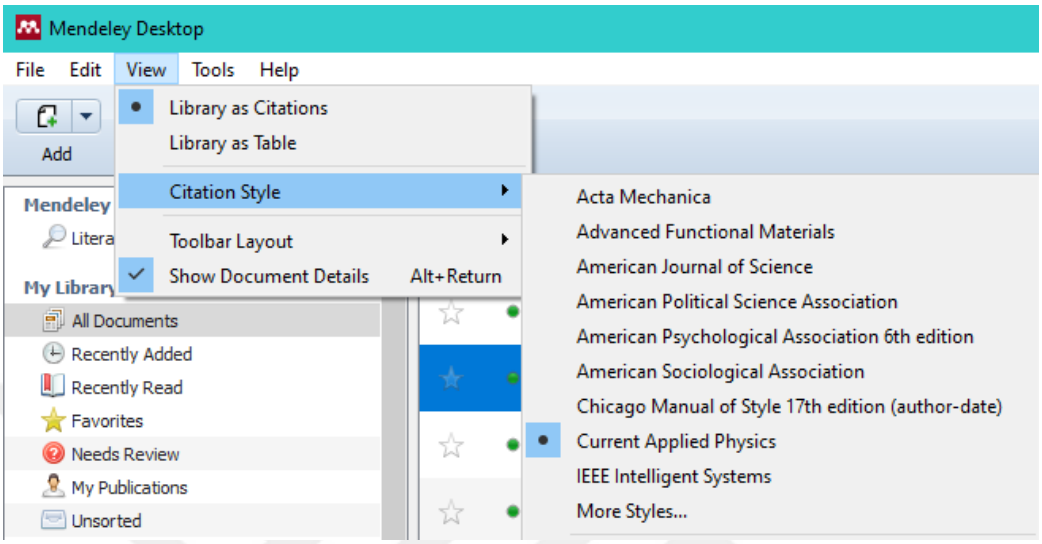


Figure A1.2. Screenshot for Mendeley citation styles in MS Word

APPENDIX- 3: THE PYTHON, MAIN CODE FOR THE 1D CNN, LSTM, CAPSULENET MODELS USING VMD METHOD.

```
!pip install vmdpy

#%% Simple example: generate signal with 3 components + noise
import numpy as np
import matplotlib. pyplot as plt
from vmdpy import VMD
import pandas as pd
from sklearn import preprocessing
import glob
import os
import pywt
import random
from tensorflow.keras import layers
from keras.layers import LSTM
import tensorflow as tf
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split # Import
train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for
accuracy calculation
from keras.models import Sequential
from keras.layers import Activation, BatchNormalization, Conv1D,
Dense, MaxPooling1D, Flatten, Dropout

#. modes
v_1 = pd.read_csv('/content/drive/MyDrive/thesis
data/new_data/ball_1hp/1hp_DE_ball_all.csv')
v_2 = pd.read_csv('/content/drive/MyDrive/thesis
data/new_data/inner_race_1hp/1hp_DE_IR_all.csv')
v_3 = pd.read_csv('/content/drive/MyDrive/thesis
data/new_data/normal_1hp/normal.csv')
v_4 = pd.read_csv('/content/drive/MyDrive/thesis
data/new_data/outer_race_1hp/1hp_DE_OR_all.csv')

f =np.append(v_3,v_4,axis=0)
f=np.append(v_2,f,axis=0)
f=np.append(v_1,f,axis=0)
print(f.shape)
```

```
#. some sample parameters for VMD
alpha = 50       # moderate bandwidth constraint
tau = 0.            # noise-tolerance (no strict fidelity
enforcement)
K = 20              # 20 modes
DC = 0              # no DC part imposed
init = 1            # initialize omegas uniformly
tol = 1e-7


#. Run VMD
seg= f[:10000]
u, u_hat, omega = VMD(seg, alpha, tau, K, DC, init, tol)
df=pd.DataFrame(u)


seg_1 = f[10000:20000]
vmd_1= u1, u_hat, omega = VMD(seg_1, alpha, tau, K, DC, init, tol)
df1=pd.DataFrame(u1)


plt.figure(figsize=(20,5))
plot1=plt.plot(u1.T)


seg_2 = f[20000:30000]
vmd_2= u2, u_hat, omega = VMD(seg_2, alpha, tau, K, DC, init, tol)
df2=pd.DataFrame(u2)


seg_3 = f[30000:40000]
vmd_3= u3, u_hat, omega = VMD(seg_3, alpha, tau, K, DC, init, tol)
df3=pd.DataFrame(u3)


seg_4 = f[40000:50000]
vmd_4= u4, u_hat, omega = VMD(seg_4, alpha, tau, K, DC, init, tol)
df4=pd.DataFrame(u4)
dff4=np.transpose(df4)

seg_5 = f[50000:60000]
vmd_5= u5, u_hat, omega = VMD(seg_5, alpha, tau, K, DC, init, tol)
df5=pd.DataFrame(u5)
dff5=np.transpose(df5)

seg_6 = f[60000:70000]
vmd_6= u6, u_hat, omega = VMD(seg_6, alpha, tau, K, DC, init, tol)
df6=pd.DataFrame(u6)
```

```
dff6=np.transpose(df6)

seg_7 = f [70000:80000]
vmd_7= u7, u_hat, omega = VMD(seg_7, alpha, tau, K, DC, init, tol)
df7=pd.DataFrame(u7)
dff7=np.transpose(df7)

seg_8 = f[80000:90000]
vmd_8= u8, u_hat, omega = VMD(seg_8, alpha, tau, K, DC, init, tol)
df8=pd.DataFrame(u8)
dff8=np.transpose(df8)

seg_9 = f[90000:100000]
vmd_9= u9, u_hat, omega = VMD(seg_9, alpha, tau, K, DC, init, tol)
df9=pd.DataFrame(u9)
dff9=np.transpose(df9)

seg_10 = f[100000:110000]
vmd_10= u10, u_hat, omega = VMD(seg_10, alpha, tau, K, DC, init,
tol)
df10=pd.DataFrame(u10)
dff10=np.transpose(df10)

seg_11 = f[110000:120000]
vmd_11= u11, u_hat, omega = VMD(seg_11, alpha, tau, K, DC, init,
tol)
df11=pd.DataFrame(u11)
dff11=np.transpose(df11)

seg_12 = f[120000:130000]
vmd_12= u12, u_hat, omega = VMD(seg_12, alpha, tau, K, DC, init,
tol)
df12=pd.DataFrame(u12)
dff12=np.transpose(df12)

seg_13 = f[130000:140000]
vmd_13= u13, u_hat, omega = VMD(seg_13, alpha, tau, K, DC, init,
tol)
df13=pd.DataFrame(u13)
dff13=np.transpose(df13)

seg_14 = f[140000:150000]
vmd_14= u14, u_hat, omega = VMD(seg_14, alpha, tau, K, DC, init,
tol)
df14=pd.DataFrame(u14)
dff14=np.transpose(df14)

seg_15 = f[150000:160000]
```

```
vmd_15= u15, u_hat, omega = VMD(seg_15, alpha, tau, K, DC, init,
tol)
df15=pd.DataFrame(u15)
dff15=np.transpose(df8)

seg_16 = f[160000:170000]
vmd_16= u16, u_hat, omega = VMD(seg_16, alpha, tau, K, DC, init,
tol)
df16=pd.DataFrame(u16)
dff16=np.transpose(df16)

seg_17 = f[170000:180000]
vmd_17= u17, u_hat, omega = VMD(seg_17, alpha, tau, K, DC, init,
tol)
df17=pd.DataFrame(u17)
dff17=np.transpose(df17)

seg_18 = f[180000:190000]
vmd_18= u18, u_hat, omega = VMD(seg_18, alpha, tau, K, DC, init,
tol)
df18=pd.DataFrame(u18)
dff18=np.transpose(df18)

seg_19 = f[190000:200000]
vmd_19= u19, u_hat, omega = VMD(seg_19, alpha, tau, K, DC, init,
tol)
df19=pd.DataFrame(u19)
dff19=np.transpose(df19)

seg_20 = f[200000:210000]
vmd_20= u20, u_hat, omega = VMD(seg_20, alpha, tau, K, DC, init,
tol)
df20=pd.DataFrame(u20)
dff20=np.transpose(df20)


seg_21 = f[210000:220000]
vmd_21= u21, u_hat, omega = VMD(seg_21, alpha, tau, K, DC, init,
tol)
df21=pd.DataFrame(u21)
dff21=np.transpose(df21)

seg_22 = f[220000:230000]
vmd_22= u22, u_hat, omega = VMD(seg_22, alpha, tau, K, DC, init,
tol)
df22=pd.DataFrame(u22)
dff22=np.transpose(df22)
```

```
seg_23 = f[230000:240000]
vmd_23= u23, u_hat, omega = VMD(seg_23, alpha, tau, K, DC, init,
tol)
df23=pd.DataFrame(u23)
dff23=np.transpose(df23)

seg_24 = f[240000:250000]
vmd_24= u24, u_hat, omega = VMD(seg_24, alpha, tau, K, DC, init,
tol)
df24=pd.DataFrame(u24)
dff24=np.transpose(df24)

seg_25 = f[250000:260000]
vmd_25= u25, u_hat, omega = VMD(seg_25, alpha, tau, K, DC, init,
tol)
df25=pd.DataFrame(u25)
dff25=np.transpose(df25)

seg_26 = f[260000:270000]
vmd_26= u26, u_hat, omega = VMD(seg_26, alpha, tau, K, DC, init,
tol)
df26=pd.DataFrame(u26)
dff26=np.transpose(df26)

seg_27 = f[270000:280000]
vmd_27= u27, u_hat, omega = VMD(seg_27, alpha, tau, K, DC, init,
tol)
df27=pd.DataFrame(u27)
dff27=np.transpose(df27)

seg_28 = f[280000:290000]
vmd_28= u28, u_hat, omega = VMD(seg_28, alpha, tau, K, DC, init,
tol)
df28=pd.DataFrame(u28)
dff28=np.transpose(df28)

seg_29 = f[290000:300000]
vmd_29= u29, u_hat, omega = VMD(seg_29, alpha, tau, K, DC, init,
tol)
df29=pd.DataFrame(u29)
dff29=np.transpose(df29)

seg_30 = f[300000:310000]
vmd_30= u30, u_hat, omega = VMD(seg_30, alpha, tau, K, DC, init,
tol)
df30=pd.DataFrame(u30)
dff30=np.transpose(df30)
```

```
seg_31 = f[310000:320000]
vmd_31= u31, u_hat, omega = VMD(seg_31, alpha, tau, K, DC, init,
tol)
df31=pd.DataFrame(u31)
dff31=np.transpose(df31)

seg_32 = f[320000:330000]
vmd_32= u32, u_hat, omega = VMD(seg_32, alpha, tau, K, DC, init,
tol)
df32=pd.DataFrame(u32)
dff32=np.transpose(df32)

seg_33 = f[330000:340000]
vmd_33= u33, u_hat, omega = VMD(seg_33, alpha, tau, K, DC, init,
tol)
df33=pd.DataFrame(u33)
dff33=np.transpose(df33)

seg_34 = f[340000:350000]
vmd_34= u34, u_hat, omega = VMD(seg_34, alpha, tau, K, DC, init,
tol)
df34=pd.DataFrame(u34)
dff34=np.transpose(df34)

seg_35 = f[350000:360000]
vmd_35= u35, u_hat, omega = VMD(seg_35, alpha, tau, K, DC, init,
tol)
df35=pd.DataFrame(u35)
dff35=np.transpose(df35)

seg_36 = f[360000:370000]
vmd_36= u36, u_hat, omega = VMD(seg_36, alpha, tau, K, DC, init,
tol)
df36=pd.DataFrame(u36)
dff36=np.transpose(df36)

seg_37 = f[370000:380000]
vmd_37= u37, u_hat, omega = VMD(seg_37, alpha, tau, K, DC, init,
tol)
df37=pd.DataFrame(u37)
dff37=np.transpose(df37)

seg_38 = f[380000:390000]
vmd_38= u38, u_hat, omega = VMD(seg_38, alpha, tau, K, DC, init,
tol)
df38=pd.DataFrame(u38)
dff38=np.transpose(df38)
```

```
seg_39 = f[390000:400000]
vmd_39= u39, u_hat, omega = VMD(seg_39, alpha, tau, K, DC, init,
tol)
df39=pd.DataFrame(u39)
dff39=np.transpose(df39)

seg_40 = f[400000:410000]
vmd_40= u40, u_hat, omega = VMD(seg_40, alpha, tau, K, DC, init,
tol)
df40=pd.DataFrame(u40)
dff40=np.transpose(df40)


seg_41 = f[410000:420000]
vmd_41= u41, u_hat, omega = VMD(seg_41, alpha, tau, K, DC, init,
tol)
df41=pd.DataFrame(u41)
dff41=np.transpose(df41)

seg_42 = f[420000:430000]
vmd_42= u42, u_hat, omega = VMD(seg_42, alpha, tau, K, DC, init,
tol)
df42=pd.DataFrame(u42)
dff42=np.transpose(df42)

seg_43 = f[430000:440000]
vmd_43= u43, u_hat, omega = VMD(seg_43, alpha, tau, K, DC, init,
tol)
df43=pd.DataFrame(u43)
dff43=np.transpose(df43)

seg_44 = f[440000:450000]
vmd_44= u44, u_hat, omega = VMD(seg_44, alpha, tau, K, DC, init,
tol)
df44=pd.DataFrame(u44)
dff44=np.transpose(df44)

seg_45 = f[450000:460000]
vmd_45= u45, u_hat, omega = VMD(seg_45, alpha, tau, K, DC, init,
tol)
df45=pd.DataFrame(u45)
dff45=np.transpose(df45)

seg_46 = f[460000:470000]
vmd_46= u46, u_hat, omega = VMD(seg_46, alpha, tau, K, DC, init,
tol)
df46=pd.DataFrame(u46)
```

```
dff46=np.transpose(df46)
seg_47 = f[470000:480000]
vmd_47= u47, u_hat, omega = VMD(seg_47, alpha, tau, K, DC, init,
tol)
df47=pd.DataFrame(u47)
dff47=np.transpose(df47)

#. Visualize decomposed modes
plt.figure(figsize=(20,10))
plt.subplot(2,1,1)
plt.plot(f)
plt.title('Original signal')
plt.xlabel('time (s)')
plt.subplot(2,1,2)
plt.plot(u.T)
plt.title('Decomposed modes')
plt.xlabel('time (s)')
plt.legend(['Mode %d'%m_i for m_i in range(u.shape[0])])
plt.tight_layout()

df_all=np.append(dff46,dff47,axis=0)
df_all=np.append(dff45,df_all,axis=0)
df_all=np.append(dff44,df_all,axis=0)
df_all=np.append(dff43,df_all,axis=0)
df_all=np.append(dff42,df_all,axis=0)
df_all=np.append(dff41,df_all,axis=0)
df_all=np.append(dff40,df_all,axis=0)
df_all=np.append(dff39,df_all,axis=0)
df_all=np.append(dff38,df_all,axis=0)
df_all=np.append(dff37,df_all,axis=0)
df_all=np.append(dff36,df_all,axis=0)
df_all=np.append(dff35,df_all,axis=0)
df_all=np.append(dff34,df_all,axis=0)
df_all=np.append(dff33,df_all,axis=0)
df_all=np.append(dff32,df_all,axis=0)
df_all=np.append(dff31,df_all,axis=0)
df_all=np.append(dff30,df_all,axis=0)
df_all=np.append(dff29,df_all,axis=0)
df_all=np.append(dff28,df_all,axis=0)
df_all=np.append(dff27,df_all,axis=0)
df_all=np.append(dff26,df_all,axis=0)
df_all=np.append(dff25,df_all,axis=0)
df_all=np.append(dff24,df_all,axis=0)
df_all=np.append(dff23,df_all,axis=0)
df_all=np.append(dff22,df_all,axis=0)
df_all=np.append(dff21,df_all,axis=0)
df_all=np.append(dff20,df_all,axis=0)
df_all=np.append(dff19,df_all,axis=0)
```

```
df_all=np.append(dff18,df_all,axis=0)
df_all=np.append(dff17,df_all,axis=0)
df_all=np.append(dff16,df_all,axis=0)
df_all=np.append(dff15,df_all,axis=0)
df_all=np.append(dff14,df_all,axis=0)
df_all=np.append(dff13,df_all,axis=0)
df_all=np.append(dff12,df_all,axis=0)
df_all=np.append(dff11,df_all,axis=0)
df_all=np.append(dff10,df_all,axis=0)
df_all=np.append(dff9,df_all,axis=0)
df_all=np.append(dff8,df_all,axis=0)
df_all=np.append(dff7,df_all,axis=0)
df_all=np.append(dff6,df_all,axis=0)
df_all=np.append(dff5,df_all,axis=0)
df_all=np.append(dff4,df_all,axis=0)
df_all=np.append(dff3,df_all,axis=0)
df_all=np.append(dff2,df_all,axis=0)
df_all=np.append(dff1,df_all,axis=0)
df_all=np.append(dff,df_all,axis=0)
df_all.shape
dd=pd.DataFrame(df_all)

# normalize the data attributes
normalized = preprocessing.StandardScaler().fit(df_all)
X_scaled = normalized.transform(df_all)
print('X_scaled=',X_scaled.shape)

x=pd.DataFrame(X_scaled)
X_scaled=[]


# select input (X) and output (y) variables
filePath= '/content/drive/MyDrive/thesis
data/new_data/output_vmd_2/ball'
read_files= glob.glob(os.path.join(filePath,"*.csv"))
np_array=[]
for files in read_files:
    filesData= pd.read_csv(files, header=0)
    y_G1 = filesData


    np_array.append(y_G1)
merge_value=np.vstack(np_array)
yG1=pd.DataFrame(merge_value)
```

```
filePath2= '/content/drive/MyDrive/thesis
data/new_data/output_vmd_2/inner'
read_files2= glob.glob(os.path.join(filePath2,"*.csv"))
np_array2=[]
for files in read_files2:
    filesData2= pd.read_csv(files, header=0)
    y_G2 = filesData2


    np_array2.append(y_G2)
merge_value2=np.vstack(np_array2)
yG2=pd.DataFrame(merge_value2)


filePath3= '/content/drive/MyDrive/thesis
data/new_data/output_vmd_2/normal'
read_files3= glob.glob(os.path.join(filePath3,"*.csv"))
np_array3=[]
for files in read_files3:
    filesData3= pd.read_csv(files, header=0)
    y_G3 = filesData3


    np_array3.append(y_G3)
merge_value3=np.vstack(np_array3)
yG3=pd.DataFrame(merge_value3)




filePath4= '/content/drive/MyDrive/thesis
data/new_data/output_vmd_2/outer'
read_files4= glob.glob(os.path.join(filePath4,"*.csv"))
np_array4=[]
for files in read_files4:
    filesData4= pd.read_csv(files, header=0)
    y_G4 = filesData4
    np_array4.append(y_G4)
merge_value4=np.vstack(np_array4)
yG4=pd.DataFrame(merge_value4)

yAllCh=np.append(yG3,yG4,axis=0)
yAllCh=np.append(yG2,yAllCh,axis=0)
yAllCh=np.append(yG1,yAllCh,axis=0)
y=yAllCh
print(y)
```

```
# Split dataset into training set and test set
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42,shuffle=True)

print(f'X_train shape  is {x_train.shape}')
print(f'X_test shape  is {x_test.shape}')
print(f'y_train shape  is {y_train.shape}')
print(f'y_test shape  is {y_test.shape}')

x_train=np.array(x_train)
x_test=np.array(x_test)


# Reshaping X_train for efficient modelling
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))

# CNN model
from tensorflow import keras
model = Sequential()
model.add(Conv1D(filters=16, kernel_size=3, strides=1,
input_shape=(20,1), activation='relu', padding='same'))
model.add(Conv1D(filters=16, kernel_size=3, strides=1,
activation='relu', padding='same'))
model.add(Conv1D(filters=32, kernel_size=3, strides=1,
activation='relu', padding='same'))
model.add(Conv1D(filters=32, kernel_size=3, strides=1,
activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2 ))
model.add(Flatten())
model.add(Dense(50, activation='relu' ))
model.add(Dropout(0.01))
model.add(Dense(20))
model.add(Activation('softmax'))
print(model.summary())


# Compiling the CNN
opt=keras.optimizers.Adam( learning_rate=0.001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
# Fitting to the training set
history = model.fit(x_train, y_train, epochs=30,
batch_size=64,validation_data=(x_test, y_test), shuffle=True)


ModelLoss, ModelAccuracy = model.evaluate(x_test, y_test)
print('Test Loss is {}'.format(ModelLoss))
```

```
print('Test Accuracy is {}'.format(ModelAccuracy ))

# LSTM Model
model = Sequential()
# First LSTM layer with Dropout regularisation
model.add(LSTM(units=32, activation='relu', return_sequences=True,
input_shape=(x_train.shape[1],1)))
# Second LSTM layer
model.add(LSTM(units=32, activation='relu'))
model.add(Dropout(0.1))
# The output layer
model.add(Dense(units=100, activation='relu'))
model.add(Dense(units=50, activation='relu'))
model.add(Dense(units=20, activation='softmax'))
print(model.summary())

# CapsuleNet Model
def CapsNet(input_shape, n_class, routings):
    x = layers.Input(shape=input_shape)

    # Layer 1: Just a conventional Conv2D layer
    conv1 = layers.Conv2D(filters=256, kernel_size=9, strides=1,
padding='valid', activation='relu', name='conv1')(x)

    # Layer 2: Conv2D layer with `squash` activation, then reshape to
[None, num_capsule, dim_capsule]
    primarycaps = PrimaryCap(conv1, dim_capsule=8, n_channels=32,
kernel_size=9, strides=2, padding='valid')

    # Layer 3: Capsule layer. Routing algorithm works here.
    digitcaps = CapsuleLayer(num_capsule=n_class, dim_capsule=16,
routings=routings,
    name='digitcaps')(primarycaps)

    # Layer 4: This is an auxiliary layer to replace each capsule
with its length. Just to match the true label's shape.
    # If using tensorflow, this will not be necessary. :)
    out_caps = Length(name='capsnet')(digitcaps)

    # Decoder network.
    y = layers.Input(shape=(n_class,))
    masked_by_y = Mask()([digitcaps, y]) # The true label is used to
mask the output of capsule layer. For training
    masked = Mask()(digitcaps) # Mask using the capsule with maximal
length. For prediction

    # Shared Decoder model in training and prediction
    decoder = models.Sequential(name='decoder')
```

CURRICULUM VITAE

Aydil Jomaa BAPIR

[redacted]

ACADEMIC ACTIVITIES

Paper:

1. A. Bapir and İ. Aydin, "Cloud based bearing fault diagnosis of induction motors," *Computer Science*, no. Special, pp. 141–146, DOI:10.53070/bbd.990814