Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

A Deep Learning Approach for Fault Diagnosis in Electrical Power Systems

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology – Software

By

Noor Hassan Fadhil Noor

Supervised by

Assist.Prof.Dr.Mohammed. H. Dosh

2022 A.C. **1443 A.H.**

(وَمَنْ يَتَّقِ اللَّهَ يَجْعَلْ لَهُ مَخْرَجًا ۝ وَيَرْزُقْهُ مِنْ حَيْثُ لَا يَحْتَسِبُ وَمَنْ يَتَوَكَّلْ عَلَى اللَّهِ فَهُوَ حَسْبُهُ إِنَّ اللَّهَ بَالِغُ أَمْرِهِ قَدْ جَعَلَ اللَّهُ لِكُلِّ شَيْءٍ قَدْرًا ۝)

(الطلاق ٢, ٣)

صدق الله العظيم

Declaration

I hereby declare that this dissertation entitled “A deep learning approach for fault diagnosis in electrical power systems ”, submitted to the University of Babylon in partial fulfillment of the requirements for the degree of Master in Information Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source is appropriately cited in the references.

Signature:

Name: **Noor Hassan Fadhil**

Date: / / 2022

Supervisor's Certification

I certify that the thesis entitled "**A Deep Learning Approach for Fault Diagnosis in Electrical Power Systems**" was prepared under my supervision at the Department of Software / College of Information Technology / University of Babylon as partial fulfillment of the requirements for the degree of Master in Information Technology - Software.

Signature:

Supervisor Name: Assist.Prof.Dr. Mohammed Husain Dosh

Date: / / 2022

The Head of the Department Certification

Given the available recommendations, I forward the thesis entitled "**A deep learning approach for fault diagnosis in electrical power systems**" for debate by the examination committee.

Signature:

Assist. Prof. Dr. Ahmed Saleem Abbass

Head of Software Department

Date: / / 2022

Certification of the Examination Committee

We, the undersigned, certify that (**Noor Hassan Fadhil**) candidate for the degree of Master in Information Technology - Software, has presented his thesis of the following title (**A Deep Learning Approach for Fault Diagnosis in Electrical Power Systems**) as it appears on the title page and front cover of the thesis that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:

Signature:
Name:
Title: Assistant Professor
Date: / / 2022
(**Chairman**)

Signature:
Name:
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name:
Title: lecturer
Date: / / 2022
(**Member**)

Signature:
Name: Dr.Mohammed H. Dosh
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name:
Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Dedications

I dedicate this thesis

To my loving mother who was and still is my source of strength, everything I am or ever will be I owe it belongs to you, my warmest home, and my paradise. I ask Allah to protect you from all evil and I will not disappoint you.

To My father, who never stops giving of himself in countless ways. I will always be your daughter who is proud of you, and I will not disappoint you, my beloved father, who stands by me when things look very difficult.

To My beloved brothers and sisters are a constant source of support and encouragement during all times.

To My husband, my life partner who stood by me and supported me, courage me.

To My dear child who supports me with her warm eyes and smile.

Acknowledgments

Foremost, I am highly grateful to Allah (God) for His unlimited blessings that continue to flow into my life, and because of You, I made this through against all odds.

I would like to take this opportunity to acknowledge my humanitarian and academic supervisor, Dr. Mohammed Dosh, who accepted supervision despite my difficult circumstances and my health crisis, I am grateful to you.

I would like to thank staff of the college, who gives me the information and achieve all procedures of my study, and the teachers who helped and guided me to successfully complete my study career and they all have the credit to make me stand here today. Also, All thanks to my institute, which give me the chance to get this study.

Last but not least, I would like to thank those who helped me with a word of encouragement, a smile on a dark day, or scientific information that benefited me in this research, thank you all.

Noor Hassan Fadhil Al rubayee

Abstract

A fault is defined as a non-desired deviation in a system or one of its components from its normal behavior. Therefore, Diagnosing fault nowadays is one of the most problems that need to a solution, especially in Electrical Power Systems (EPS). Power systems are quickly developed in the latest years because of the latest digital technologies and using them in automatic control systems. Unmanned Aerial Vehicles (UAVs) consider one of the automatic control systems. Which are characterized by the absence of a human pilot onboard and have been used in several fields. In such systems power system is considered an important and essential part. Which may be suffering from problems as a result of a fault in it or one of its components, undetected failures can lead to critical damage. Therefore, fault diagnosis System is used to eliminate problems in the system. Fault diagnosis (FD) is the process of detecting, isolating, and identifying mode faults, which is the aim of the proposed methodology.

The proposed system adopted deep learning one dimensional convolutional neural network (1DCNN) to build fault diagnosis classifiers for single and multi-faults by using "ADAPT" dataset, coming from NASA Research Center for EPS of UAV.

Finally, the classification accuracy for the proposed system for single fault classifier achieved for fault detection (96.70%), for fault isolation (98.51%), and for identifying severity of fault (98.67%). While for a multi fault classifier achieved for fault detection (99.10%), for faults isolation (99.03%), and for identifying severity of faults (99.14%).

Declaration Associated with this Thesis

Some of the works presented in this thesis have been published or accepted as below.

First Paper:

Title: Using 1DCNN for Diagnosis Fault in Electricals Power System

Author: Mohammed Husain Dosh and Noor Hassan Fadhil**,**

Journal: International Journal of Mechanical Engineering,
Volume: 7,

Number: 2,

Pages: 3127- 3135,

Year: February, 2022,

Publisher: International Journal of Mechanical Engineering.

List of Contents

List of Figures

Figure name **Page Number**

List of Tables

Table Name **Page Number**

List of Algorithms

Algorithm Name **Page Number**

Table of Abbreviation

Abbreviation	Meaning
1D	One Dimension
2D	Two Dimensions
ADAPT	Advanced Diagnostics and Prognostics Testbed
ANN	Artificial neural networks
BP	Back-Propagation
CM	Confusion Matrix
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DCNN	Deep Convolutional Neural Network
DNN	Deep Neural Network
DT	Diction Tree
EPS	Electrical Power System
FC	Fully Connected layer
FDC	Fault Diagnosis Classifiers
FLTz	Flight- simulator
FN	False Negative
FP	False Positive
GFRF	Generalized Frequency Response Function
GPU	Graphics Processing Unit
KMA	Korea Meteorological Administration
LSTM	Long Short Term Memory
MCSVM	Multi-class Support Vector Machine.
MFD	Multi Fault Diagnosis Classifier
MFDDNN	Multi Fault Detection Deep Neural Network
MFIDDNN	Multi Fault Identify Deep Neural Network
MFIDNN	Multi Fault Isolate Deep Neural Network
MSE	Mean Square Error
PCA	Principal Component Analysis
PMSM	Permanent Magnet Synchronous Motor
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Units
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SEE	SampEn-weighted Energy
SFD	Single Fault Diagnosis Classifier
SFDDNN	Single Fault Detection Deep Neural Network
SFIDDNN	Single Fault Identify Deep Neural Network
SFIDNN	Single Fault Isolate Deep Neural Network

SGD	Stochastic Gradient Descent
SNNs	Synthetic Neural Networks
S-T	Stockwell -Transform
Std	Standard Deviation
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent
TN	True Negative
TP	True Positive
TSC	Time Series Classification
UAS's	Unmanned Aircraft Systems

Chapter One

INTRODUCTION

Chapter One

INTRODUCTION

1.1 Overview

The power system generally contains devices connected to the system such as a generator, transformer, motor, contactor, and circuit breaker. An EPS has several components and sensors [1] as shown in figure (1.1). Increasing use of the terminology of fault, failure, and malfunction requires highlighting to avoid confusion. A fault is an undesirable deviation from the normal or international behavior of one or more than the component of system, Failure is the continual disruption of a system's ability to satisfy the required function under certain operating conditions, and malfunction is an intermittent anomaly in the performance of the required function of the system [2]. The faults may occur in both components and sensors. Diagnosing fault of EPSs have become very important in our society because of their use in most aspects of life and it is necessary for the proper operation of most devices in various medical, military and civil industries.

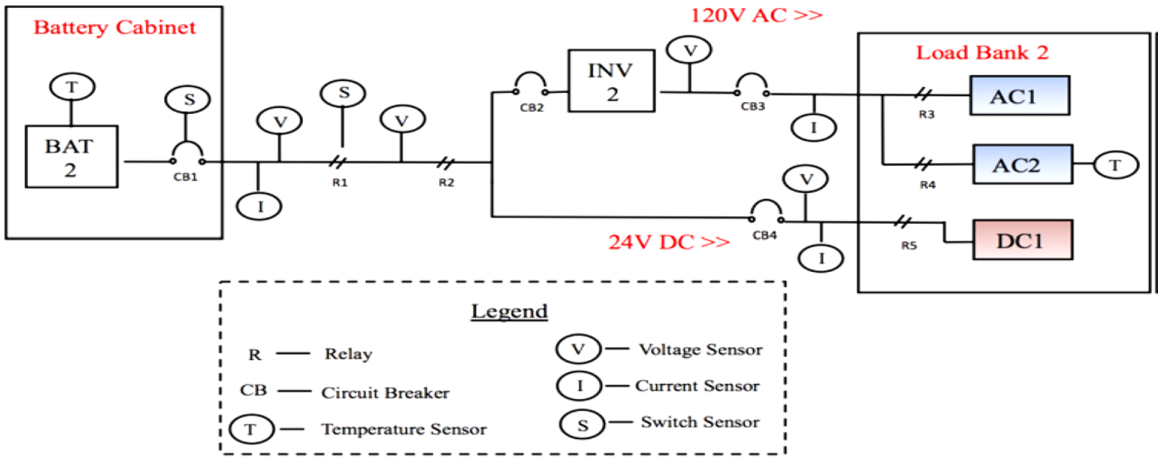


Figure 1.1: sensors and component of EPS[3].

UAVs usually rely on electric power, in whole or in part. The first UAV was manufactured by the Americans Lawrence and Sperry in 1916, Recently, UAV has been used in many fields [1][4]. The power system of the UAV includes several add-ons and sensors. Add-ons or subsystems that operate simultaneously and contain many components, which are necessary for its operation and are always exposed to electrical disturbance or failure. Sensors are widely used to obtain information by measuring the quality of the components of the system that will affect the accuracy, balance and reliability of the system [1] [5] .

Faults are expensive for every industry because they force machine overhaul, downtimes in the entire process, with consequent loss and delay in the task [6]. It is assumed that the early identification of faults in EPS is detected and properly classified, so it should be eliminated as soon as possible [4]. To overcome faults that occur in the power system, a special fault diagnosis system is used. Fault Diagnostic System (FDs) is a system capable of detecting the presence of faults in the system under observation and locating them in different words, FDs is able to show the three tasks of detecting, isolating, and identifying faults. which can be summarized as follows [7]:

• Fault detection: To make a binary choice whether everything is OK (nominal) or something has been wrong for a long time (outside of nominal).

• Fault isolation: To identify the fault area, i.e. perceive which element, sensor or actuator has become faulty.

• Determining the error: to estimate the severity, type or nature of the fault [7].

In the past (Previously), diagnosis of sensor and component faults was by implement multiple sensors of a similar parameter score at same time, and deal with the problem with the internal side effect as an error. Usually, this

approach is effective, however, excessive usage of sensors and component will now no longer be the simplest due to increasing its cost, but, in addition, makes the system complex. Then used a set of methods to diagnose faults such as knowledge based fuzzy logic, a decision tree, a support vector machine, and a fuzzy neural network. Recently, deep artificial neural networks using the deep learning approach in fault diagnosis due to their ability to handle huge data and give correct detection and identification [8] [9].

1.2 Thesis Motivation

Many researchers are motivated to solve the problem of fault that occurs in EPSs, Due to increasing requests to acquire more reliable implementation on such complex systems. Failures in components or the sensor of EPS can lead to disastrous consequences, the main motivations are:

• The human losses as a result of a malfunction and smashing in the EPS. Therefore, this problem is considered one of the main crucial motives for building fault-diagnosing systems.

• Material damage to public and private property resulting from failure to diagnose faults.

• Faults are expensive for every industry because they force machine repair, stopping periods in the whole process.

1.3 Research Problem

Due to the expansion in the industry and the great number of faults that take place in the EPS, So the main problem that is addressed in this thesis is " fault diagnosis in EPS" or in other words detect ,isolate ,and identify single and multi-fault.

1.4 Thesis Aim and Objectives

This thesis sought to diagnosis single and multiple faults in EPS of UAVs.

To achieve this key aim, the following objectives would be met:

1. Building a model with highly effective to detect if faults occur or not in single faults in EPS.
2. Building a model for isolating location and identifying fault type in single faults in EPS.
3. Building a model with highly effective to detect if faults occur or not in multi faults in EPS.
4. Building a model for isolating location of faults and identifying fault type in multi faults in EPS.

1.5 Thesis Contributions

The significant contribution in this thesis is to present a general and efficient model for detecting, isolating, and determining faults that occur in the single fault dataset and multi fault dataset that occurs in the EPS of UAV.

1.6 Challenges of the Research Problem

Many researchers addressed this issue (fault diagnosis), the challenges faced this thesis can be summarized in below point:

1. Filling missing values, the presence of missing values due to the different sensors read rates, which read in 1HZ, 2HZ, and 10HZ.
2. Normalization, The presence of different values due to the different functions of sensors such as reading voltages, temperature, and switching location.

3. Uncertainty in diagnosis states when sensors data similar for different components and return same fault type.

1.7 Related Works

Due to the development in industry and the requirements to increase the safety and correctness of EPS, the fault diagnosis algorithm has become the focus of most studies and research. Therefore a review of the most relevant work is provided and listed in Table (1.1):

1. Jing-li .yet al (2016) [10], have shown that self-validating multifunctional sensors have been used in most industries. They used gas data sample implemented by board PCI-6014, which contained four type of faults. So the method was used to detect, isolate and diagnose their faults complete ensemble empirical mode decomposition with sample entropy weighted energy and multiclass SVM (MSVM). An inadequate in this work they deal with the little type of faults mode.
2. Jiangmeng. Fu et al(2019) [11], used deep-learning approach to diagnose faults in six-rotor UAVs based on hybrid CNN-LSTM. They used four experiment of flight UAV data. The proposed method used sliding window technique and then extracts features using the networks for diagnose actuator faults. Accuracy calculated for each experiment then average accuracy calculated CNN-LSTM model. An inadequate in this work they used few datasets.
3. Aslam. M et al.(2019) [12], overviewed an application in power system, they used deep LSTM to present novel case study on the solar irradiation forecasting, single input, two hidden layers, and one output layer used in this work .KMA 'Korea Meteorological Administration' dataset is used from 2001 to 2017. Authors used data from 2001 to 2016 for predicating

2017 then comparing predicted with actual data. This work required only for yearly forecasting photovoltaic generation.

4. Jing. R et al. (2019) [13], used deep learning, CNN, for fault detection and classification in unmanned ground vehicle. Authors generated dataset by take one input signal and two output signal then converted them to 2D time-frequency images by use continuous wavelet transform (CWT) to feed them to DNN as Inputs. Authors improved ability of their model to detection faults and classification, they generate four scenario contain three type of faults as well to no fault. Accuracy calculated for each scenario, then take average accuracy. An inadequate in this work algorithm need to extend for more types of faults and investigate the fault tolerance control algorithms for unmanned ground vehicles as well.
5. Lerui.C et al.(2020) [14], used a combination of nonlinear generalized frequency response function (GFRF) and CNN for fault diagnosis in Permanent magnet synchronous motor (PMSM). PMSM data is used, which contain normal and three type of faults. Authors suggest to test the versatility of the proposed method by diagnosing other nonlinear systems as feature work. Inadequate in this work the method used was less affected by the outliers.
6. M Ganesan et al.(2020) [15], used 1DCNN for detect fault in electrical power system of satellite. They use ADAPT dataset of six univariate experiments, they take only read of sensor voltage and ignore other from these experiments. S-transform used for process data before input to CNN and compared with used of sliding windows Fast Fourier Transform (FFT) and Wavelet Transform (WT). An inadequate in this work they didn't determine type of fault but detected faults occur or not.

7. Ahmad. A and Dahrouj .Z.(2020) [16], used Decision Tree matrix C4.5(DTMatrix) for detecting fault in UAV. Instead of employing a

single large decision tree, they employed small decision trees. FLTz dataset from NASA was used, these dataset consist of 20 flights (sample of data), which have four type of faults. Authors used 15 flights of these dataset (8 for trained and 7 for tested) to build binary classifier to detect fault occurred or not. DT required time for sliding windows approach and for train and test. Inadequate in this work they used a few scenarios.

Table 1.1: Summary of the Related Work.

Authors	Technique	Dataset	Accuracy
Jing-li .yet et al [10]	CEEMD+SEE+MCSVM	data acquisition board (PCI-6014) with four type of faults	95%
Jiangmeng. Fu et al[17]	CNN-LSTM	Flight data of UAV with four scenario	92.74%
Aslam. M et al [12]	CNN-RNN-LSTM	KMA for data of 17 years	MSE= 10.4
Jing. R et al [13]	CNN	Generate dataset have four scenario with three type of faults	99.35%
Lerui. C et al[14]	GFRF+CNN	PMSM with three type of faults	98.75%
M Ganesan et al [15]	ST-CNN	ADAPT for six scenario	96.6%
Ahmad. A and Dahrouj .Z [16]	DT	flight data “FLTz of UAV for 20 scenario	0.994%

1.8 Thesis Outline

After review Chapter 1, where it represents the general introduction about the recommendation system research, literature review, problem statement, and explains the objectives of this thesis. The rest of the thesis is structured as follows:

Chapter 2: Presents an extensive description of the main principles, theoretical concepts, and technologies of deep learning that will be used in the proposed system. Moreover, the evaluation measure has been used.

Chapter 3: Clarifies the main steps of designing the proposed system, also illustrate the behavior of diagnosis algorithm in UAV.

Chapter 4: Shows the implement of the proposed system on real datasets and results of the experiments

Chapter 5: Lists the derived conclusions of this proposed system and gives suggestions for future work.

CHAPTER Two

THEORETICAL BACKGROUND

CHAPTER TWO
THEORITICAL BACKGROUND

2.1 Overview

The theoretical background of diagnosing faults that appear in power systems of UAS and the types of these faults will be explained. In addition, an explanation of Neural Networks and Deep learning will be introduced. Nowadays, Deep learning has offered excellent performance as a result of the recent swift advances in deep CNN's. It is receiving increasing attention and applied in a wide field such as object detection and recognition, aerial imagery registration, video processing, scene classification, autonomous or unmanned driving, and medical image-related applications. 1DCNN's proposed recently to solve complex problems. In several applications, such as personal biomedical data classification and diagnosis the abnormality detection and identification in electrical motor fault detection, and quickly achieved sophistication performance. Another key benefit is that due to the simple and compact architecture of one dimensional CNN.

2.2 Unmanned Aircraft Systems (UAS's):

UASs are a recent portion of the aviation system [18], UASs is defined as the system where the components comprise air vehicles and related equipment that do not load a human operator, but rather fly independently or are driven remotely. UAS must be understood in the context of a system, which comprise the command, control, and communications system, as well as personnel controlling the unmanned aircraft [19] [20][21].

UAS have seen a lot of use recently in both military and civilian applications. Its value and benefits in search and rescue, real-time monitoring, reconnoitering operations, traffic monitoring, risky site inspection, and have lately been used in the cultivation field. Furthermore, UASs are well-suited to circumstances that are too risky and hazardous for humans to observe directly. UAS is a burgeoning field in the unmanned aviation community. The use of UA technology is currently transforming commercial and humanitarian activity. Progress in this area has worked with an undeniably quick extension of UAV innovation that has begun to move into an assortment of areas [22].

EPS are critical to the operation of UAVs. The choosing of an electric power system is dictated largely by the period time of continuous power needs for the task profile of UAV. The power system, unlike many other UAV subsystems, supports both the platform and the payload [18].

2.3 Electrical Power System (EPS)

EPS is an electrical component network that is deployed to provide, transmit, and employ electric power. Which is considered real-time energy delivery systems. Electric power systems have witnessed great development in several fields, such as transmission, generation, distribution, and load systems. This development and expansion were accompanied by the emergence of many malfunctions in the devices, which caused the failure of some of these systems and the low level of quality of their performance, which led to a decrease in confidence in these energy systems and significant economic losses [23].

2.3.1 Faults in Powers Systems:

EPSs are developing quickly in the latest years, due to its increased use of it in several industries. Which is an important and essential part of industrial devices and has several components and sensors. That may suffer from faults. An abnormal state in an electric system is the definition of electrical fault induced by instrumentality failures such as transformers and spinning machinery, human mistake, and environmental factors. These faults disrupt electric flows, equipment harm, and even result in death. On the other hand, the divergence of voltages and currents from natural values or states is referred to as an electrical fault [24].

Electrical systems are an important element of the UAV power system, especially because more electrical aircraft become available [25]. Among the electrical needs on board are vital subsystems like avionics, propulsion, life support, and environmental controls. EPSs on UAVs operating in cruel circumstances are characterized by physically consolidated topologies in which high-intensity generation feeds power to electronics-related loads [3].

To demonstrate the EPS in UAV. This example represents an EPS providing power to UAS. This EPS consists of a number of components and sensors providing power from a source (battery) to a set of loads (devices). Two use cases are considered for this system. In the first one (single-string UAS), illustrated in Figure (2.1), Electrical power is supplied from a battery (BAT2) to a number of loads in the UAS. Transfer power via a number of circuit breakers (names of components starting with "CB"), relays ("EY"), and an inverter INV2 to provide alternating current power. AC483, FAN416, and DC485 are the names of the loads. Also, the system is equipped with sensors to register voltage of electrical (names starting with "E"), current electrical

("IT"), and the positions of circuit breakers and relays ("ISH", "ESH"). Finally, one sensor (fan speed, "ST") is used to report the load's operational status. The second use case is described in detail in figure (2.2) where redundant UAS is presented. In this use case, multi faults can be occurs, if any, as opposed to a single fault [26].

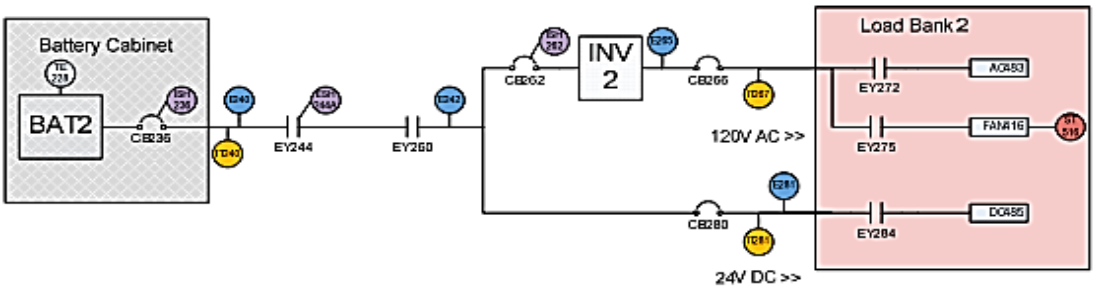


Figure 2.1: EPS schematic of single-string UAS [26].

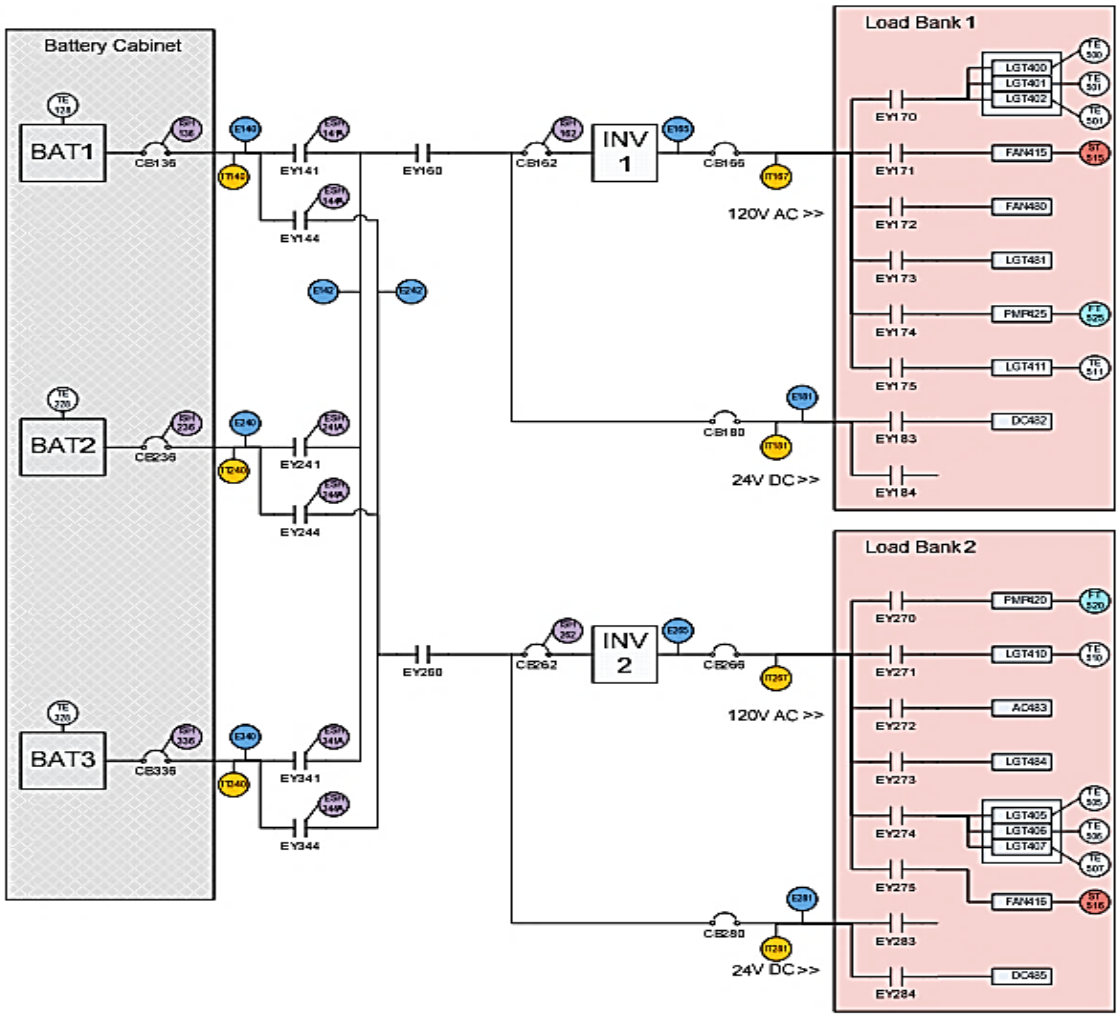


Figure 2.2: EPS schematic of redundant UAS [26].

2.3.2 Types of Faults:

Automated control systems faults can be classified as follows:

- Actuator: occur when an actuator behaves abnormally.
- Sensor: defined as severe inaccuracies in sensor measurements.
- Plant: These are alterations in the system's dynamical input/output qualities.

To clarify precisely faults, actuator and sensor faults are considered hardware faults, whereas plant faults are software faults.

Figure (2.3) illustrate automated faults [27].

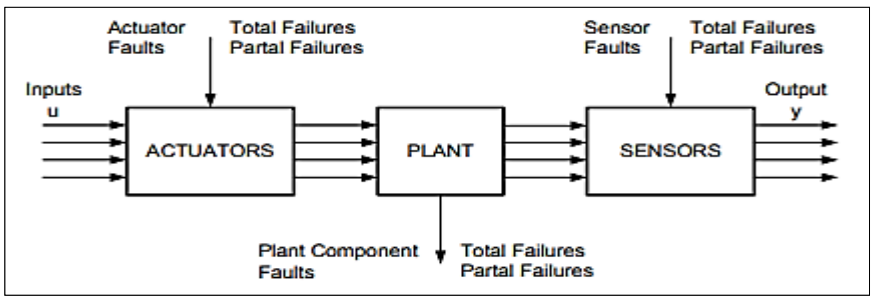


Figure 2.3: Automated Faults [2].

Faults above also can be classified based on how they are modelled, as follows:

- Additive faults: Faults that may be addressed as external inputs or biases. They generally impact the system's sensors.
- Multiplicative faults: faults whose status and input vectors are multiplied. Such failures can affect both sensors and actuators.

Figure (2.4) illustrated modelled faults.

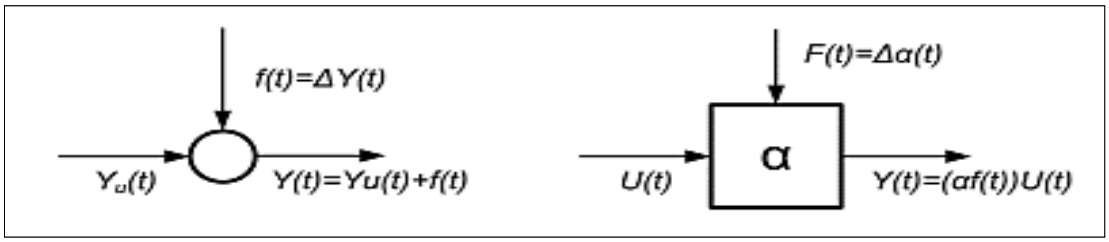


Figure 2.4: Modelled faults [2].

Regardless of whether a fault is in the actuator, sensor or plant, which can represent according to the severity of the fault as follows.

- Abrupt faults: They are variations in system parameters values that occur more quickly than in a normal dynamic process.
- Incipient faults (drift): these faults cause a small effect that may be unnoticed by the detecting system.
- Intermittent fault: This type of fault is prevalent within most systems. The identification of intermittent faults is big defiance for most detection methods due to their inconsistent nature.

Form faults (severity) have approximately 12 fault mode types. In this thesis, the highlight fault location and fault mode types for diagnosis purposes [18]. Figure (2.5) shows the form (severity) of faults.

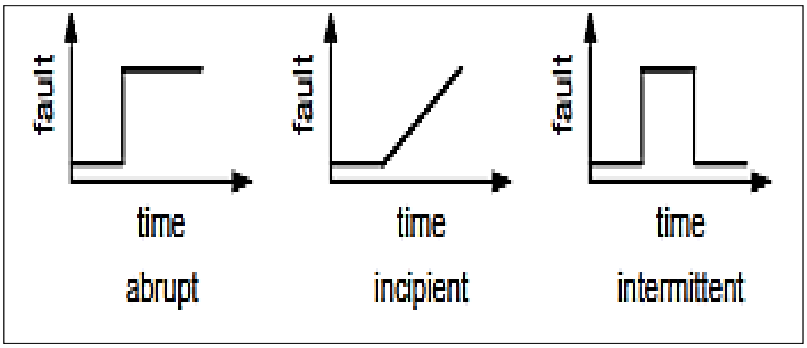


Figure 2.5: severity faults [2].

2.4 Fault Diagnosis:

Fault diagnosis can be known as the process of detecting, isolating and identifying the fault in the system based on observing its behavior. Fault detection is determined by the appearance of a fault in the monitoring system. It comprises detecting the faults based on the variance between the measurable signals to detect faults in processes, actuators, and sensors. Fault

isolation determines the location whereas fault identification determines the mode of fault [21].

The cause's latent for the rising interest in fault diagnosis in power systems are fault detection has vital in high-cost, safety-sensitive operations. Power systems fault diagnostic is of crucial significance in power system operation, in addition to the complexity and high degree of correlation that can result in a huge variety of alarms and status signals being produced as a result of disruption [28]. As a result, process monitoring and fault diagnosis are essential parts of creative and new systems of automatic managing to operate the production systems [21].

2.5 Time-Series Data

Time–series data indicates a series of data points that are arranged by time. Because the characteristics data are ordered, it makes no difference whether the order of data is chronological or not. The crucial aspect is that there are features that may be based on the ordering. Therefore, consider time-series classification problems differ from ordinary classification problems. A time series is a collection of data in which one or more variables vary over time. Univariate time series is that have only a single variable that fluctuates over time. On the other hand, a multivariate time series occurs when more than one variable changes over time [29].

2.6 Fault Dataset:

Fault datasets are split into two sets according to the number of faults that occurs at the same moment single-fault and multi-faults. In this section, the most important dataset will mention:

- **Satellite Power System (SPS)**

This dataset is univariate time series, which was acquired from NASA, that have six experiments each experiment represent a read of sensor voltage [30] .

- **Intelligent Flight Control (IFC)**

This dataset is simulated to flight and acquired from NASA in experiments. Which have twenty experiments, each experiment generated with a random path [31].

- **ADAPT Electrical Power System (EPS)**

This dataset is a simulated function of a UAV. Which acquiring from NASA Ames Research Center capturing in a number of scenarios. Each scenario consists of a sequence of time-series data collected from sensors and user commands. There are two types of scenarios: single and multi in the dataset [32].

2.7 ANNs:

ANNs are models that simulate biological neural systems in the real world, with the objective of processing data similarly to the human brain. Multiple nodes make up ANNs, and these nodes are designed to simulate real neurons. The nodes are connected together by links, which interact with one another and have weight. These nodes take input data and do basic procedures on it before passing the results to other neurons. The ANN can learn by modifying the weights' values [33].

2.7.1 Architecture of ANNs:

ANN is made up of basic computing units called neurons, each neuron executes a (typically non-linear) calculation on the input, and they are

collected together based on the different architectures. For instance, perhaps arranged in layers (multi-layer network) or have a topology of connection. The following layers make up layered networks [34]:

- Input layer: (n neurons) where each network has one input layer.
- Hidden layer/s: One or more hidden (or mediate) made up of m neurons make up the hidden layer.
- Output layer: is made up of p neurons, one for each output of the network. The connection pattern is allowed to distinguish two kinds of architectures.
- The feed-forward architecture: Which lacks feedback connections (signals only reach the neurons of the next layer).
- Feedback architecture: that reforms the connections between neurons from the last to the previous layers.

2.7.2 Models of Neural Networks:

2.7.2.1 Perceptron:

The perceptron is a simplified representation of a real neuron, which tries to emulate it by passing through the following steps: it receives the input digital signal, let's call them x1, x2,..., xn, computes a weighted sum y for these inputs, then processes it through a threshold function to result from the outputs [35]. Which can be used to solve a simple problem, as shown in figure (2.6).

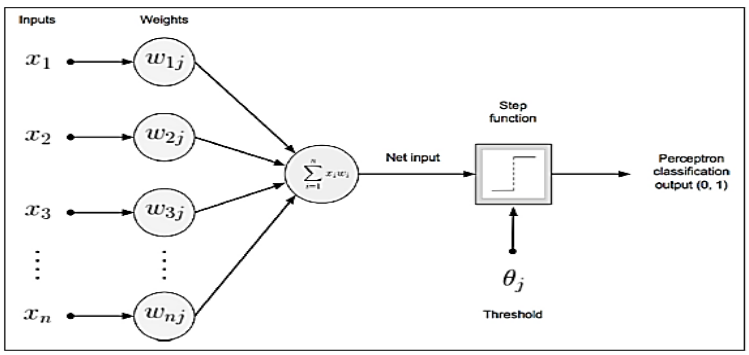


Figure2.6: Perceptron with a single layer [29].

Where x_0 is the bias, with weight w_0, Y is the activation function value, F(y) is the transfer function, $x_1, \ldots, n$ the signals of n input neurons, and $w_1, \ldots, n$ are the input weights The vector entry pass's all inputs: x1,... xn, and all weights w1...,wn into vectors xi and wi, and the output will be 1 when its dot product is positive and -1 otherwise. Equation (2.1) represent inputs and weight vectors and equation (2.2) represent output after apply activation function.

$$\mathbf{X} = \begin{matrix} \mathbf{1} \\ \mathbf{x1} \\ \cdot \\ \cdot \\ \mathbf{xn} \end{matrix}, \mathbf{w} = \begin{matrix} \mathbf{w0} \\ \mathbf{w1} \\ \cdot \\ \cdot \\ \mathbf{wn} \end{matrix} \quad \ldots\ldots\ldots (2.1)$$

$$\boldsymbol{y = f(net)} \ \ldots\ldots\ldots \ \boldsymbol{(2.2)}$$

Where net represents the input stimuli cumulative (input function) to the neuron and *f* is a nonlinear function (activation function) of the net, given by equation (2.3):

$$\boldsymbol{net = X_1 * w_1 + X_2 * w_2 + .. + X_m * w_m} \quad \ldots\ldots\ldots (2.3)$$

2.7.2.2 Multi-Layer Perceptron:

If there is a complex pattern need to be recognized, such as a visual pattern, many hidden layers are required in this case, and in this state, the network, called a Deep Neural Network (DNN)[36]. Where it contain an input layer, one or more mediate layers of neurons, and an output layer [34] as shown in figure(2.7).

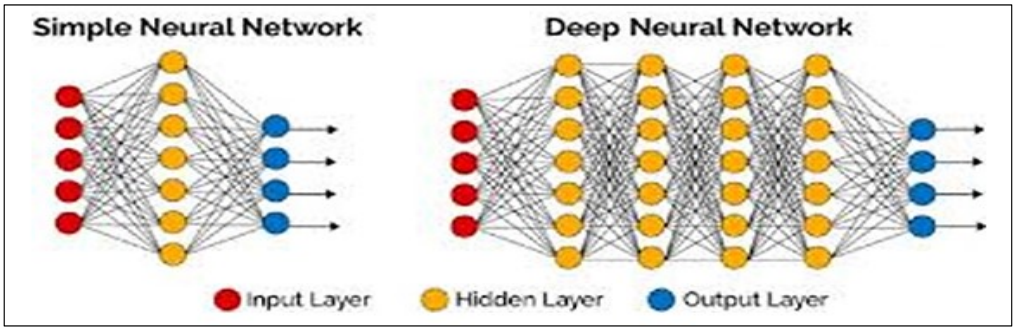


Figure2.7:Neural network with multi-layer perceptron's [34].

2.7.2.3 Backpropagation:

Backpropagation is the most important algorithm which powers ANN training [37]. In the 'standard' ANN the input of a neural network during the forward propagation will map to the output. The back-propagation algorithm aims to minimize the resulted loss error between the real and expected output by updating initial weights. This is performed by propagating the error back to the former layer. The weights are changing in the training process by the optimization functions to reach the lowest values of error rate. The results will be the output of the model [38].

Although ANNs are excellent at handling a variety of issues, they have two flaws. The first is that neural networks with few hidden layers are inadequate to solve complicated issues, therefore, necessitating the addition of more hidden layers in order to achieve better results. The second is that it's difficult to select the best features from a high dimensional input. DNNs solve these problems [39].

2. 7.3 Neural Network characteristics:

1. Number of hidden layers and neurons: Several experimental studies have found that one hidden layer is enough to approximate basic non-linear functions with satisfactory accuracy. This strategy, however, necessitates a high number of neurons, which limits the training process. In general, networks with two hidden layers are more effective, particularly for high-frequency data prediction. This is confirmed by experience, which shows how, on the other hand, networks with more than two hidden layers can result in considerable gains in network performance. It's also worth mentioning that an excessive number of neurons causes overlearning, which means that the neurons are unable to generate an accurate prediction since the contribution of inputs has been reduced. It appears that the

network has "retained" the right responses but lacks the capacity to generalize them. The minimal number of neurons, on the other hand, limits the network's learning ability [40].

2. Activation functions: One of most important parameters in ANNs. Which is used to learning and approximating reasonably of any continuous and complicated relationship among network variables. They use to get the output of the neural networks. Generally, activation functions are divided into two types: linear and non-linear activation functions. Only forward propagation neural networks and simple problems can represent linearly using the linear activation function. Linear activation functions have drawbacks, such as they don't use in back-propagation networks since the function's derivative is constant. The main component used to achieve the non-linearity output in layers of neural network is non-linear activation functions. The importance of a non-linear activation function in a neural network indicates to the availability of data that don't segregate in a linear style [41].

Therefore, a multi-layer or deep neural network does not acquire from the increased layer in it when only a linear activation function is used. The usage of the nonlinear activation function is very significant for some networks, due to its ability to map the network's output into a limited range. Therefore, a multi-layer or deep neural network does not acquire from the increased layer in it when only a linear activation function is used. The most common non-linear activation functions applied in neural networks are sigmoid, softmax, tanh, and ReLU functions. The form of output determines which activation function is chosen. Softmax activations are recommended for classification issues, whereas ReLU is favored for predictive/regression problems [36].

- Linear activation function: Activation of a linear activation function is proportional to the input, and its equation is comparable to that of a straight line. From equations (2.1) and (2.2), in case of choosing the linear, as explained in equation (2.4):

$$\mathbf{y = C\,net} \quad \ldots\ldots\ldots (2.4)$$

Where C fixed constant [42].

- ReLU (Rectified Linear Unit): Widely used in neural networks especially deep-learning models. This function returns zero if it received a negative value, but returns the same value if received any positive value[43]. Figure (2.8) show its plot. A decision has to be made while designing a neural network. From equations (2.1) and (2.2), in case of choosing the ReLU, as explained in equation (2.5):

$$\boldsymbol{y = max(0, net)} \quad \ldots\ldots\ldots (2.5)$$

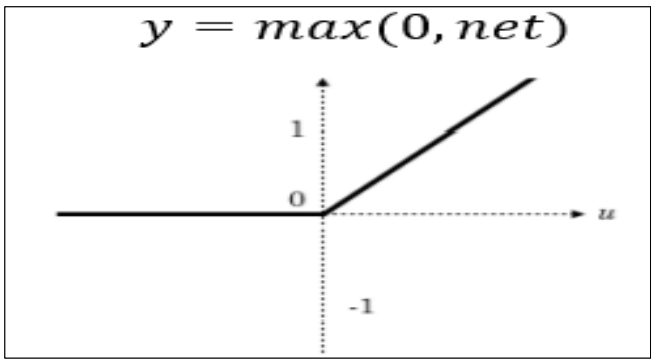


Figure 2.8: ReLU activation function [43]

- Leaky ReLU function: Instead of translating all negative values to 0, the Leaky ReLU preserves a tiny scope for negative values. It dresses the dying ReLU problem and outperforms ReLU in terms of performance. Leaky ReLU activation is given by the equation (2.6) and shown in figure (2.9).

$$y = 0.01\,net\ when\ net < 0 \qquad \ldots\ldots\ldots \quad (2.6)$$

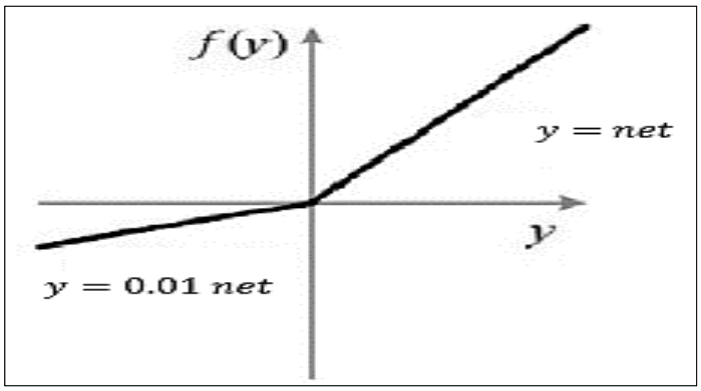


Figure 2.9: Leaky ReLU activation function [44].

- Sigmoid transforms values in the range 0 to 1 range. Which use for a binary classification neural network [45]. Sigmoid activation is given by the following equation (2.7) and shown in figure (2.10).

$$\sigma(net) = \frac{1}{1+e^{-net}} \qquad \ldots\ldots\ldots \quad (2.7)$$

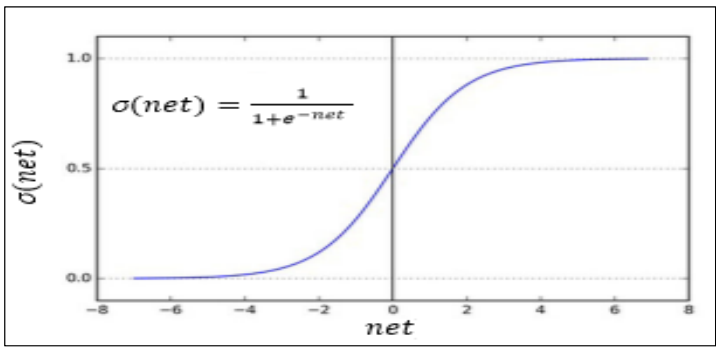


Figure 2.10: sigmoid activation function [44].

- Softmax function: is a nonlinear activation function preferred for multi-class classification. The output layer softmax function for a c-class problem that represents the c probability of input belonging to each of the c classes, is given by the following equation (2.8)[46].

$$P(class(i) = j(net)) = \frac{e^{net_j}}{\sum_{k=1}^{n} e^{net_k}} \qquad \ldots\ldots\ldots \quad (2.8)$$

3. Loss function: In neural networks, there are several formulas that have appeared for calculating error, these formulas are called loss functions, which represents the aberration of output prediction, O, and the actual output, T. When using different loss functions can lead to a different error value for the same prediction, therefore, the type of predictive modelling problem imposes limitations on the kind of loss function that can be utilized [36][41].

There are three main types of loss functions:

a) Classification use with classification problems such as binary classification two classes and multiclass classification more than two classes, some of classification loss function explain bellow

- Binary cross entropy used for the binary classification issues in DNNs, the objective function that is the binary cross entropy equation, as in logistic regression shown in Equation (2.9) [41].

$$\boldsymbol{l = \frac{1}{N}\sum_{n=1}^{N}[y_n log(\hat{y_n}) + (1 - y_n) log(1 - \hat{y_n})]} \quad \ldots\ldots\ldots (2.9)$$

Where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are the true label and the estimated output probability from the neural network respectively.

- Categorical cross entropies: (also called maximum probabilistic estimation) it is used for classification multi-class classification problems. In these problems, the target can only belong into one of several possible categories and the model must decide which one to use. Because this function is employed in networks that use the softmax activation function, it is frequently used to determine the difference between these two probability distributions. Equation (2.10) illustrates the categorical cross-entropy [41].

$$L = -\sum_{n=1}^{N}[y_n \log(\hat{y}_n)] \quad \ldots\ldots (2.10)$$

Where $\mathbf{y}$ one hot encoding vector $\hat{\mathbf{y}}$ probability distributions, and N output size.

- Sparse categorical cross entropy: The only variance between sparse categorical cross entropy and categorical cross entropy is the shape of true labels. When dealing with a single-label, multi-class classification problem, the labels for each data input are mutually exclusive, implying that each data entry can only belong to one class. Use same equation used in categorical cross entropies [41].

b) Regression is used in regression problems when the output variables are continuous such as following

- Mean Squared Error (MSE) is one of the necessary regression loss functions, which calculates the square difference between the real and expected value. Equation (2.11) illustrates the MSE [41].

$$MSE = \frac{1}{2}\sum_{i=1}^{n}(t_i - O_i)^2 \quad \ldots\ldots (2.11)$$

Where t_i actual target vector, o_i the output predicted vector, n vector length.

- Root mean squared error (RMSE) uses for evaluating the performance of the linear regression model. As a result, a high RMSE is considered "bad," whereas a low RMSE is considered "excellent." Equation (2.12) illustrates the RMSE [41].

$$L = \frac{1}{2}(\underline{\mathbf{t}} - \underline{\mathbf{O}})^T(\underline{t} - \underline{O}) \quad \ldots\ldots (2.12)$$

Where L represents the loss function.

c) Embedding is used with tasks that need to measure the similarity between two inputs. Embedding loos function such as.

- Euclidean distance loss: This loss function is an embedding loss that is typically employed in problems where two inputs must be compared, not in classification problems. It calculates the distance between two locations or vectors. The equation (2.13) illustrates of euclidean distance loss.

$$\mathbf{L} = \sqrt{\sum_{i=1}^{n}(\boldsymbol{f}(\boldsymbol{t_i}) - \boldsymbol{o_i})^2} \qquad \ldots\ldots\ldots (2.13)$$

Where $\boldsymbol{t_i}$ actual target vector, $\boldsymbol{o_i}$ the output predicted vector,n vector length [41].

4. The optimization algorithms: One of the most essential phases is choosing the algorithm to use to optimize a neural network. There are two types of optimization methods in machine learning. The first is batch or deterministic gradient methods, which handle all training instances in a big batch at the same time. The second type is stochastic or online methods, which work with only one instance at a time [41].

   Two types of optimization methods could be divided into Algorithms with constant learning rates, such as Stochastic Gradient Descent (SGD), and adaptive learning algorithms [47]. In the first group, the learning rate η is chosen manually. The task of choosing the learning rate in this type of algorithm is somewhat difficult. When choosing a relatively small learning rate, the learning process is slowed, and the training time becomes too large. While choosing a relatively large learning rate, can lead to fluctuation in the loss value around the minimum value, and this hinders the convergence process. While the algorithms of the second group, do not need to set the learning rate manually, but they have a heuristic approach that takes care of

adjusting the learning rate value, several algorithms appeared that belong to these two categories, the most important of which are illustrates as follows:

- **SGD**

SGD is one of constant learning rate algorithms that tries to update the network parameters more repeatedly. In this algorithm, the network parameters are modified on each training sample after the loss calculation. So, if the dataset contains 900 samples, the network parameters will be modified 900 times in one cycle of the dataset. The SGD equation (2.14) that is used to update parameters in a neural network is illustrated as follows:

$$\boldsymbol{w^{(k+1)} = w^{(k)} - \eta * (\nabla j(w))} \quad \ldots\ldots\ldots (2.14)$$

W is network a parameter (weights), η represents the learning rate,

$\boldsymbol{\nabla j(w)}$ is the gradient, that is obtained from the loss function [41] [48].

- **Adam**

Adam is considered one of the adaptive learning rate optimization algorithms, it measures individual learning rates for various parameters [49]. Adam sets the learning parameter automatically, this was done by using the first and second-moment estimation. But what's the moment?. The moment is the expectation of a random_variable at the power of n [50]. The moment can illustrate in equation (2.15).

$$\boldsymbol{m_n = E[X^n]} \quad \ldots\ldots\ldots (2.15)$$

Where: m is the_moment, and X is a random_variable.

The following equations uses to estimates, the first and second moment Adam[51].

$$\boldsymbol{\hat{m}_t = \frac{m_t}{1-\beta_1^t}} \qquad \ldots\ldots\ldots (2.16)$$

$$\hat{V_t} = \frac{V_t}{1-\beta^t_2} \quad \text{.....(2.17)}$$

Where m_t the previous first moment and v_t ,the previous second moment and they are initialized with 0 in the first step.

β1,β2 are new parameters inserted into the algorithm. They have default values of 0.9 and 0.999 respectively.

After calculating the value of the first and second moments, It's used to keep the network weights up to date according to the following equations(2.18).

$$w_t = w_{t-1} - n * \frac{\hat{m_t}}{\epsilon+\sqrt{\hat{v_t}}} \quad \text{..... (2.18)}$$

Where W is network weights, n is Stepsize, $\epsilon = 10^{-8}$[52].

2.8 Deep Learning:

Deep learning (defined as deep structured learning) is a branch of machine learning that interest with algorithms inspired on structure and brain functions [53]. As shown in figure (2.11).

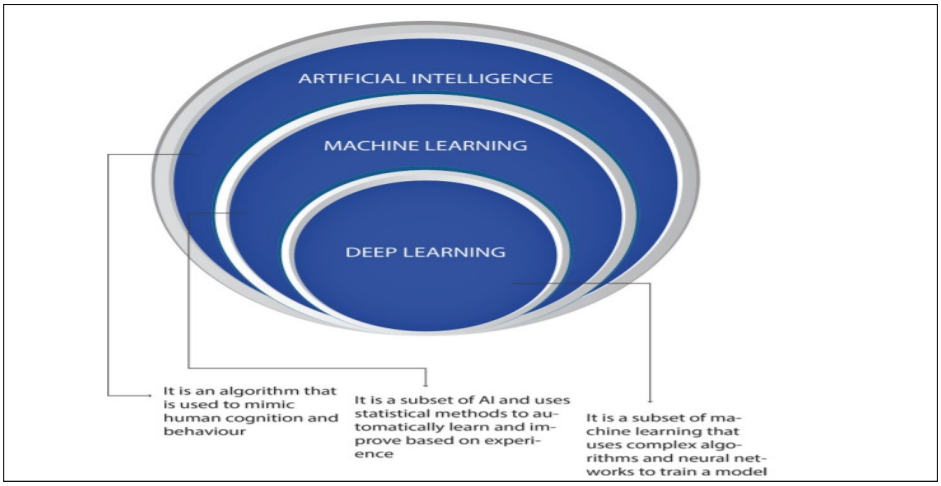


Figure 2.11: Artificial intelligent and its subfields: machine-learning and deep-learning [54].

It represents a larger group of machine learning techniques based on ANNs and representation learning consists of hierarchical architecture, where each higher layer builds on its previous lower layer, which makes it popular for the first time as hierarchical learning [47]. Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, convolutional neural networks, and recurrent neural networks have been utilized in fields, for instance, computer-vision, speech-recognition, natural-language processing, machine-translation, drug design, bioinformatics, medical image analysis, material inspection, and computer game programs, with results comparing to, or in several cases exceeding human performance. The performance of most of the old generations of learning algorithms will stabilize. It is the first class of algorithms is deep learning... which is considered scalable, its performance improves when feeds it more data[55].

The machine learning technique requires a feature vector corresponding to the raw data (raw sensor data in case of time series) as an input to detect and process it so that the handcrafted features must be extracted and later fed into a classifier. Deep learning algorithms such as CNN can quickly learn these features on their own. The goal of designing deep learning algorithms is to overcome the problems that traditional machine learning algorithms have been unable to solve [56].

The most important problems that deep learning overcame can be described as follows [34].

- **Curse of Dimensionality**

The curse dimensional problem affects a variety of domains, including numerical analysis, data processing, and machine learning. The popular theme of curse dimensional problems is that the amount of space increases so rapidly

that the data available are sparse with the dimensionality increases. Because of its vast capacity to handle numerous dimensions, deep learning has addressed this problem [57]. Deep learning algorithms have offered effective tools to handle massive volumes of unlabeled data due to their capability to handle to deal with big data analysis. Multiple neural networks types are used in deep-learning and the conventional-neural-network is one of the most important.

• Local Constancy and Smoothness Regularization

Previous notions about the type of function that machine learning algorithms can learn are required. Smoothness or local consistency is one of the most widely held ideas. Smoothness is defined as the ability of an algorithm to modify the learnt function very little within a narrow region. Because many simpler machine learning algorithms rely on local consistency assumptions to generalize successfully, they face scaling statistical problems in AI tasks [57].

2.8.1 Categories of Deep Learning:

The architectures, as well as methodologies, can be classified depending on how they are intended to use, most deep learning architecture into three types [35]:

- **Supervised Deep Learning:**

In supervised deep learning algorithms, the model is trained on labeled data. Then the model is used the learning algorithm to adjust itself, and during the testing phase, the model should determine the correct answer without relying on any label. Supervised deep learning has two main fields: classification problems and regression problems. One of the most frequently supervised models used is a CNN. The basic architecture of the CNN model which is explained later has numerous architecture-based.

Some of the CNN-based supervised deep learning architectures are AlexNet, LeNet-5, VGGNet, ZFNet, GoogleNet, and others [41][57].

- **Unsupervised deep learning**

  In the unsupervised deep learning algorithms, the model is trained on unlabeled data, and the model tries to extracting patterns and features on its own. Some of the unsupervised deep learning architectures are Restricted-Boltzmann-Machine (RBM), Deep-Belief-Network (DBN) [41]

- **Semi-Supervised Deep Learning**

  Semi-supervised deep learning takes on an intermediate stage between supervised and unsupervised learning. It takes a large amount of labeled data to support a larger set of unlabeled data. This approach is especially useful when it is difficult to extract relevant data features, and when labeling the sample a long time. The common method that uses this approach is General Adversarial Networks (GANs) [38].

2.8.2 CNNs:

CNNs are a subclass of discriminative deep architecture that have been demonstrated to be effective in data processing [58]. Deep CNN's tremendous learning capacity is partly due to the utilization of several feature extraction phases which can build hierarchical representations for the input data automatically. A CNN's performance depends largely on its architecture. Extensive knowledge both in CNNs and the examined problem field is necessary to construct a CNN with significant potential that is not always available to each interested user.

CNN implement in many with different dimensions according to the specific field application, such as one-Dimension CNN (1DCNN) usually used for signal processing, two-Dimension CNN (2DCNN) is implemented in image processing and using the 2d kernel to represent spatial features

over the image. While three-dimension CNN (3DCNN) uses in video processing [59].

2.8.2.1 CNNs Architecture:

CNN architecture is divided into two major parts as described in figure (2.12):

- **Feature extraction**

  It is the first part of the CNN architecture, and it extracts input features and transforms them into feature maps. The CNN feature extractor is comprised of one or many convolution layers, each of which has its own activation function for distinguishing the particular signal from beneficial features of every hidden layer. The first part of CNN passes through the convolution layer to convolve the filter with the input and produce feature maps which are reduced later with pooling layer, and use the previously generated feature maps to be as input feature maps and applying the same process on them, it stills go on the layer by layer and still extract powerful feature and get smaller size feature maps. Final powerful feature, reduced dimension feature maps are flattened to generate low dimensional feature vector to be fed into the prediction part [60].

- **Classification**

  After converting the features map into a vector (flatten) fed into a prediction part. The prediction part works as a traditional fully connected artificial neural network. The prediction part performs either the classification or the regression task according to the type of task that the network is training for. When working as a classifier, it returns the likelihood of class that the input may belong to it. While when works on regression task it return a vector of prediction values. The prediction

part involves the number of fully connected layers to do that purpose [60] [61].

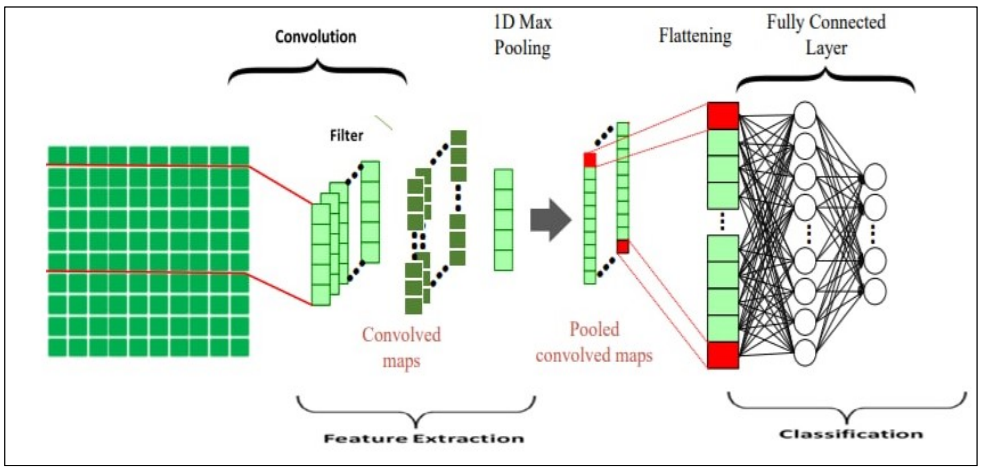


Figure 2.12: Convolution neural network architecture [60].

2.8.2.2 One-Dimensional Convolutional-Neural-Networks (1DCNNs):

1D CNNs is a modified form of 2D CNNs that was developed lately, 1DCNN convolution procedures are only performed in the one direction [62][63].

1D CNN is preferable for the following main reasons:

- A 1D CNN has remarkably lower computational complexity than a 2D CNN.
- Visibly, 1D CNN networks with perfunctory architectures are considerably much simpler to train as well as implement.
- Any CPU that can be established on a traditional computer appropriate and quite quick for training any small 1D CNNs with slight hidden layers.

In general, There are three layers to CNNs: convolutional (CONV) layers, pooling layers, and fully connected layers [60] [63].

The overall 1D-CNN layers can be explained as follows:

1. Convolutional Layer: Which is a feature extractor that extracts various features from the input data by performing convolution operations. The primary computational burden of CNNs is performed by the CONV layers, which include a series of filters (kernels) with learnable weights. The depth of the filters and inputs are the same, figure (2.13) explain the operation of convolution layer [53].

The primary parameters that affect the output feature map of the convolution layer include:

- Filter size: A filter size is represented with local receptive domain, which is a spatial region that passes over the input features vector. Which is used to extract features from the input by applying convolution operations on the input area through multiplication the filter elements with an input region's corresponding elements, these features are called feature maps.
- Number of filters: Different reasonable numbers of filters can be used, the number of feature maps is dependent on the number of filters that have been specified.
- Stride: The stride of the filter influences how it moves through the input.
- Padding: is the method of adding or logically deleting values from an input dimension such that the filter may traverse the whole dimension to control with the dimensionality of the output volume's [64].

To represent the convolution operation mathematically, let the input data of size 1×NI×N called I, convolution kernel or weight called W, bias called b. Let us suppose i filters, W, of size 1×Nw×N are convolved with the input to

produce i feature maps called O_i each with i indices. The convolution operation can be given in equation (2.19), as the following [65].

$$o_i = f(\sum_{n=1}^{N}(w_{i,n}, I_n) + b_i) \quad \ldots\ldots\ldots (2.19)$$

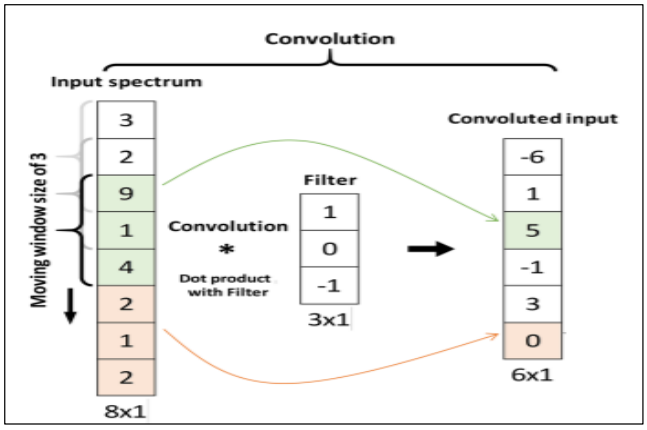


Figure 2.13: Convolution layer operation [65].

2 Activation layer (Non-Linear Layer)

Non-linear layers apply many functions such as ReLUs function', LeakyRelu, and 'softmax function'. These functions utilize to identify the significant features of each hidden layer. Figure (2.14) describes that the feature maps output from the convolution layers becomes the input to non-linear layers to convert the linear output into nonlinear [64][66].

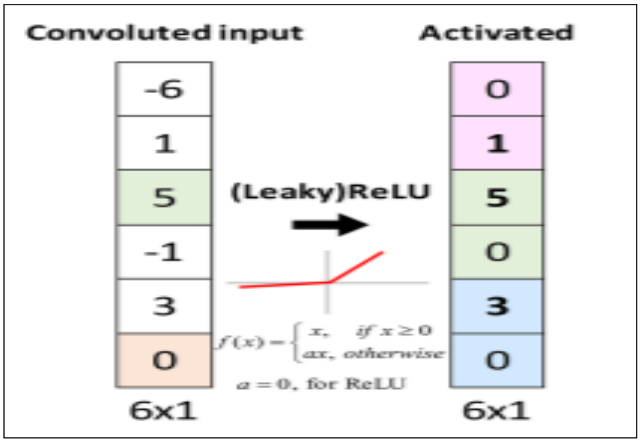


Figure 2.14: Apply activation function on feature map [65].

3 Pooling Layer:

Following The Convolutional Layer is usually by the Pooling Layer to execute a down-sample operation. The primary objective of such layer is to reduce the computational expenses by decreasing the convolved feature map's size, accomplishing this operation by reducing the connections among layers and functioning independently on each feature map. It also helps to reduce overfitting. There are many kinds of operations of pooling.

In max-pooling, the largest element is taken from the feature map as illustrate in figure (2.15). Average pooling calculates the common elements during a predefined sized input section. The total add of the elements within the predefined section is computed in sum pooling. To represent pooling operation mathematically, let the size of the pooling layer filter is 1×m and $\boldsymbol{a_i(k)}$ is the k-th item of the *i*-th feature-map input to the pooling layer to produce i roubst feature map called pi (j).

The max pooling operation can be given in equation (2. 20) as the following

$$\boldsymbol{p_i(j) = \max_{((j-1)*m<k<j*m)}(a_i(k))} \quad \ldots\ldots\ldots (2.20)$$

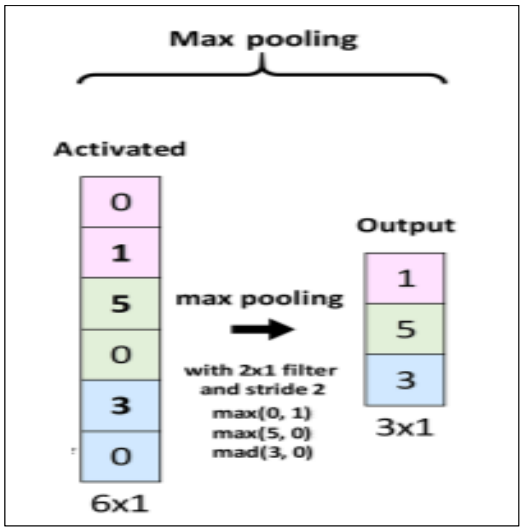


Figure 2.15: Max pooling [67].

4 Flatten:

The depth of the convolution layers output might become more than one. The flatten transforms the output of the convolution layers to generate a flat structure that may subsequently be provided as input to a fully connected layer as illustrated in Figure(2.16) [68].

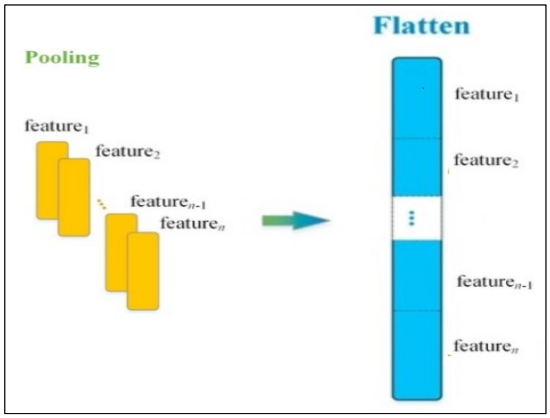


Figure 2.16: Flatten layer [69] .

5 Fully Connected Layer (FC):

The fully connected layer is usually placed as the last layer of CNN and used to perform the prediction phase of CNN. A fully connected FC layer is a structure where every neuron in one layer is connected to every neuron in the following layer [69]. FC receives the flattened output from convolution layers as input and mappings it to the output. Figure (2.17) shows the FC layer and equation (2.21) illustrate how it work. The weights, biases, and neurons, make up the FC layer, which is utilized to link neurons of two different layers. These layers may be applied before the final layer. Also known as the output layer or decision layer, is responsible to make decisions and predicting output values.

$$\boldsymbol{out = f(dot(input, w) + bias)} \quad \ldots\ldots\ldots (2.21)$$

Where dot is dot product between the weight vector w and the input. $\boldsymbol{f}$ Is softmax function which is a mathematic function that converts a vector of

numbers into a vector of possibilities. Wherever possibilities of every value measure proportionate to the relative ranking of every value within the vector. Equation (2.22) shows the performance of the function.

$$\boldsymbol{f(out)_i} = \frac{e^{out_i}}{\sum_{j=0}^{N} e^{out_i}} \qquad \ldots\ldots\ldots (2.22)$$

Where N represents the classes, out is the input vector, and $\boldsymbol{f(out)_i}$ is the output class probability[70].

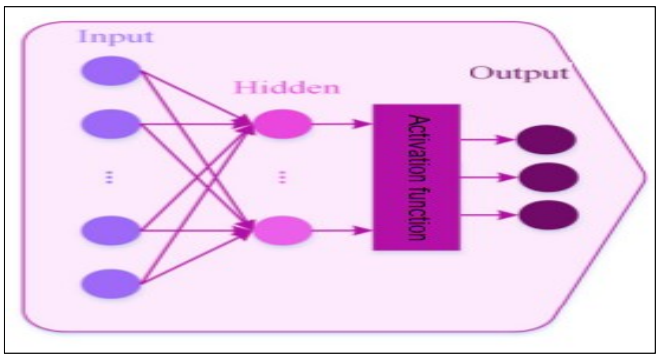


Figure 2.17:Fully connected network [69].

6 Dropout:

Dropouts are applied to avoid overfitting, which is defined as the problem of memorizing inputs rather than learning common features of the inputs. The dropout layer decreases the connection between neurons in the network through choosing neurons randomly besides sets their weights to zero in the training phase, This process occurs according to a probability value, often not more than 50%. from number neurons in layer. the dropout layer is an influential and simple method to control the model's sensitivity to noise during the training phase. The dropout process illustrates in Figure (2.18) [71].

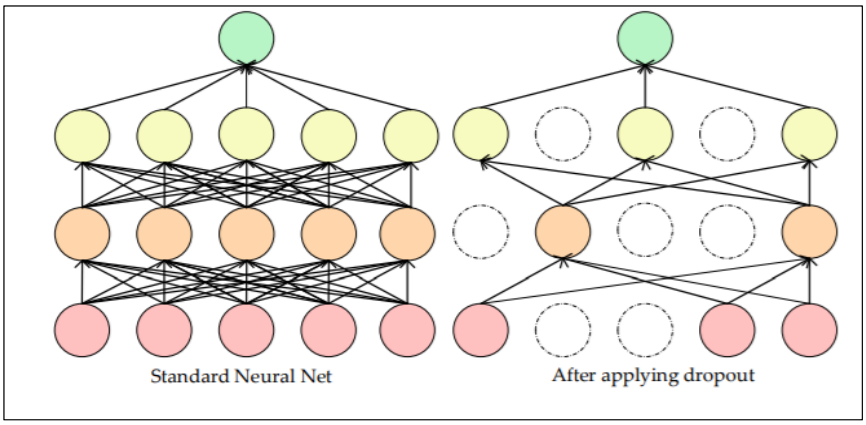


Figure 2.18: Dropout layer[71].

2.9 Handling Missing Values

Datasets of real-world may contain missing values, it is common for time series data to have missing values. Further, series values may be synchronized at an inconvenient time when collected by autonomous sensors. It is usually appropriate to have values of time series that are spaced and synchronized equally across various behavioral features for processing the data [72][73].

Methods for handling missing values

a) Interpolation

Three types of interpolation methods utilized to estimate the missing values are linear, cubic, and quadratic.

Linear interpolation: The easiest form of interpolation used to link two points of data with a linear line. As shown in equation (2.23) [73].

$$f_1(x) = f(x_0) + \frac{f(x_1)-f(x_0)}{x_1-x_0} * (x - x_0) \quad \ldots\ldots\ldots (2.23)$$

Where $f_1(x)$ value of dependent, x_0, x_1 a known values of the independent

Quadratic interpolation: This method is used if there are three data points available, as illustrated in equation (2. 24) [73].

$$f_1(x) = f(x_0) + \frac{f(x_1)-f(x_0)}{x_1-x_0} + \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0} * (x-x_0)(x-x_1) \quad \ldots\ldots\ldots (2.24)$$

Cubic Interpolation: This method is used if there are four data points are available, an equation used to calculate the Cubic dependent variable like (2.23) equation with add the fourth point.

In several cases, Quadratic and Cubic approaches do not produce considerably better outcomes than a simple linear interpolation method [74].

b) Imputation

Three types of imputation methods utilized to estimate the missing values are mean, median, and mode. All methods in imputation are employed in the same manner. Where the input vector is either a time series or a numeric. In multivariate data, the mean value of the available values for each attribute is computed, Then missing values in that attribute are replaced with the mean value. as illustrate in equation (2.25)[74][75].

$$y^{-} = \frac{1}{N}\sum_{i=1}^{N} y_i \quad \ldots\ldots\ldots (2.25)$$

Where N represent available values, y_i represent value of data.

2.10 Normalization

Normalization is a scaling approach, which transforms the original values' distribution into a new set of values with the required tributes [76]. Generally, in several scenarios, time series require to be normalized, specifically when analyzing various series simultaneously. For instance, one series may be used for temperature measurement, while another is used to pressure measure. Due to the values of series being measured on various scales, these values cannot be properly compared.

Thus, two normalization techniques are commonly utilized to adapt to these variations [72].

1. Min-Max (Range-based)

Normalization aims to scale all the numerical values of numeral features to a specified range [0,1] or [-1,1], by determining the minimum and maximum values of the features. Equation (2.26) show min_max technique[72]..

$$\boldsymbol{z = \frac{y_i - min}{max - min}} \quad \text{……. (2.26)}$$

Where $\boldsymbol{z}$ mapped value, $\boldsymbol{y_i}$ value of feature, **max** maximum value, and **min** minimum value[72].

2. Standardization

In several cases, the use of the min-max normalization cannot be applied or isn't useful. When the values of minimum or maximum attributes don't know, normalization with min-max is infeasible. Even when the values of minimum and maximum are available, the outlier's presence may bias the normalization of min-max by grouping the values and determining the precision of available digital to represent the values. Therefore standardization technique is used, the mean and standard deviation of the attributes are used for normalization as shown in equation (2.27) [77] [78].

$$\mathbf{z = \frac{x - \mu}{\sigma}} \quad \text{…… (2.27)}$$

Where z stander scalar value, μ is the mean and calculate by equation (2.28), while σ is stander deviation calculate by equation (2.29) [72].

$$\mu = \mathbf{\frac{1}{n} * \sum_{i=1}^{n} (xi)} \quad \text{…… (2.28)}$$

$$\boldsymbol{\sigma} = \sqrt{\mathbf{\frac{1}{n}} * \sum_{\mathbf{i=1}}^{\mathbf{n}} (\mathbf{x_i} - \mu)^{\mathbf{2}}} \quad \text{…… (2.29)}$$

2.11 One Hot Encoding:

Due to machine learning algorithms only accepting numerical inputs, categorical variables must be encoded into numeric values using the encoding technique. Therefore, the one hot encoding technique is used. That compares each class of the categorical variable to a set of reference classes. Which transforms one variable with observations (n) and different values (d) into binary variables (d), each having n observations. Every observation denotes whether the categorical binary variable is present (1) or absent (0) [79].

The two classes are represented by:

Class 1: [1 0]

Class 2: [0 1]

2.12 Performance Measures

Evaluating performance and efficiency is an important aspect when designing a model in machine learning. To make the machine learning model dependable, an assessment tool that is appropriate for the nature of the model's work must be chosen. Usually, when evaluating machine learning models, several scales are employed to guarantee that the model is correctly evaluated. Machine learning evaluation measures are divided into three main types: the measures that use to evaluate classification tasks, the measures that use to evaluate regression tasks, and the measures that use to evaluate clustering tasks [80].

2.12.1 Evaluation Measures for Classification Tasks

Several metrics have been used in the classification tasks such as:

- Confusion matrix is one of the most important tools that provide a complete description of the performance of the classification model. Figure (2.19) shows the confusion matrix of a multi -class classification model [81].

		Predicted Number			
		Class 1	Class 2	…	Class n
Actual Number	Class 1	x_{11}	x_{12}	…	x_{1n}
	Class 2	x_{21}	x_{22}	…	x_{2n}
	. . .	. . .	. . .	. . .	. . .
	Class n	x_{n1}	x_{n2}	…	x_{nn}

Figure 2.19: Confusion matrix for a multi class classification model [81].

Every item in a confusion matrix describes the number of predictions generated by the model which classify the classes that were correctly classified or wrongly classified. In multi-class classification needs to calculate total False-Negative (FN), False–Positive (FP), True-Positive (TP), and True-Negative (TN), as illustrate in equations (2.30), (2.31), (2.32), and (2.33) [81].

- True–Positive (TP): correct-positive prediction.
- False–Positive (FP): incorrect-positive prediction.
- True–Negative (TN): correct-negative prediction.
- False–Negative (FN): incorrect-negative prediction.

$$FN_i = \sum_{\substack{J=1 \\ j \neq i}}^{n} class_{ij} \quad \text{..... (2.30)}$$

$$FP_i = \sum_{\substack{J=1 \\ j \neq i}}^{n} class_{ji} \quad \text{..... (2.31)}$$

$$TP_i = \sum_{j=1}^{n} class_{jj} \quad \text{..... (2.32)}$$

$$TN_i = \sum_{\substack{J=1 \\ J \neq i}}^{n} \sum_{\substack{k=1 \\ k \neq i}}^{n} class_{jk} \quad \ldots\ldots\ldots (2.33)$$

- Accuracy: is the total number of correct predictions to the total amount of input samples as shown in equation (2.34) [82].

$$\mathbf{Accuracy} = \frac{TP_{all}}{TP_{all}+FP_{all}+TN_{all}+FN_{all}} \quad \ldots\ldots\ldots (2.34)$$

- Precision computed the number of true-positive predictions divide by predictions of a total number of positive. Precision metrics can calculate according to the following equation (2.35).

$$\boldsymbol{Precision} = \frac{TP_{all}}{TP_{all}+FP_{all}} \quad \ldots\ldots\ldots (2.35)$$

- Recall is used to calculate the rate of true positive in the model. Which can be calculated according to the following equation (2.36).

$$\boldsymbol{Recall} = \frac{TP_{all}}{TP_{all}+FN_{all}} \quad \ldots\ldots\ldots (2.36)$$

- F-score can represent a united mean of recall and precision. The F1 tries to balance between recall and precision, and use to measure a test's accuracy which determines the precise of how many samples it correctly classifies and robust by cannot ignore the significant number from samples of the model. Equation (2.37) represents the F-score measure.

$$\mathbf{F-score} = 2 * \frac{precision*recall}{precision+recall} \quad \ldots\ldots\ldots (2.37)$$

Chapter Three

The Proposed System

Chapter Three
THE PROPOSED SYSTEM

3.1 Overview

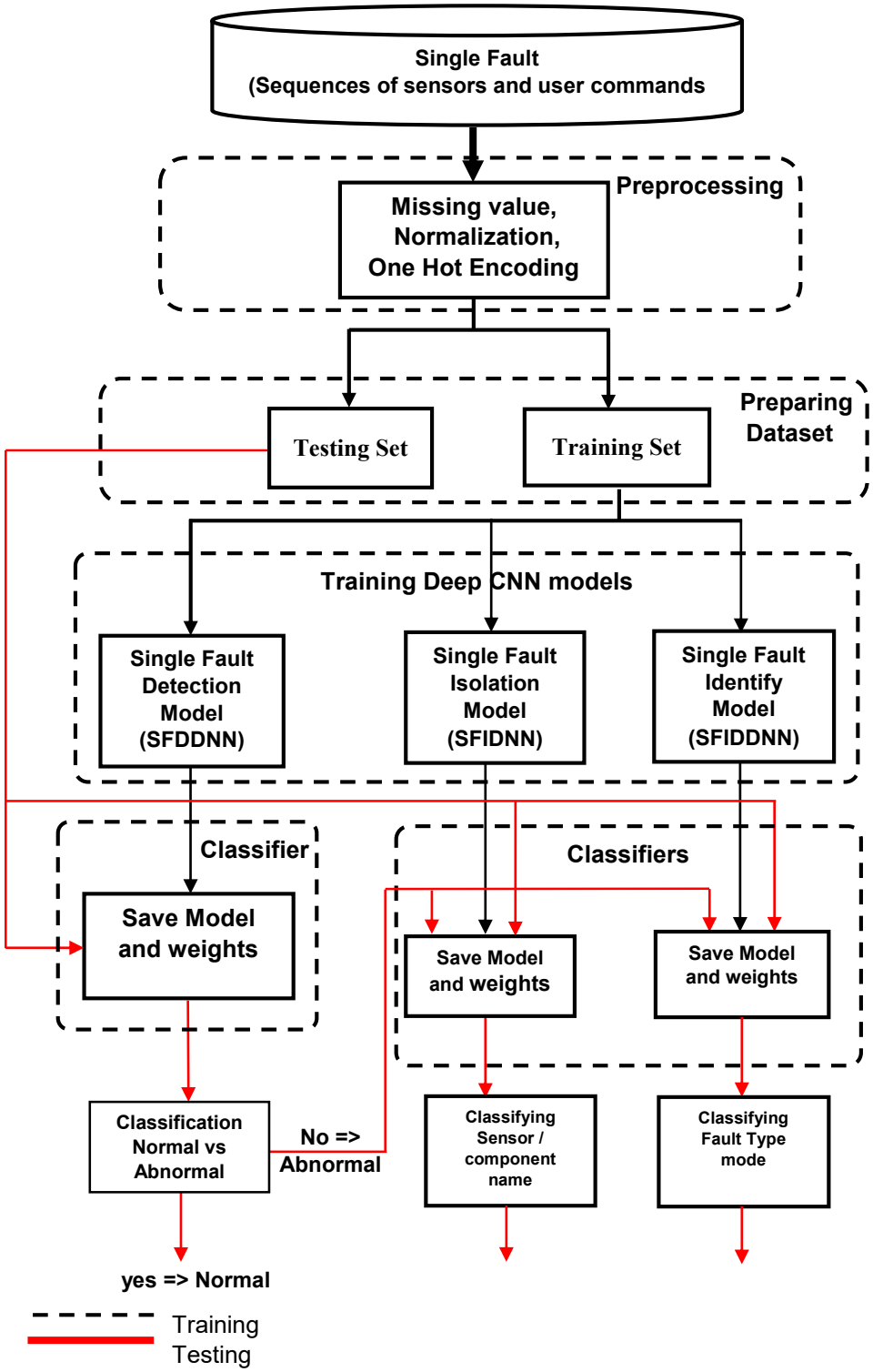
Recently, fault being's diagnosis based on EPS of UAV has received much attention and have an importance in the individual's life. In this chapter, the proposed system for fault diagnosis materials and all the followed processes will be illustrated and discussed in detail for both single faults and multiple faults. A detailed description about all the followed steps that have been used in the proposed fault diagnosis system will be introduced including the algorithms, a description of each step and its importance to get the desired goal of the thesis.

3.2 Proposed System Design

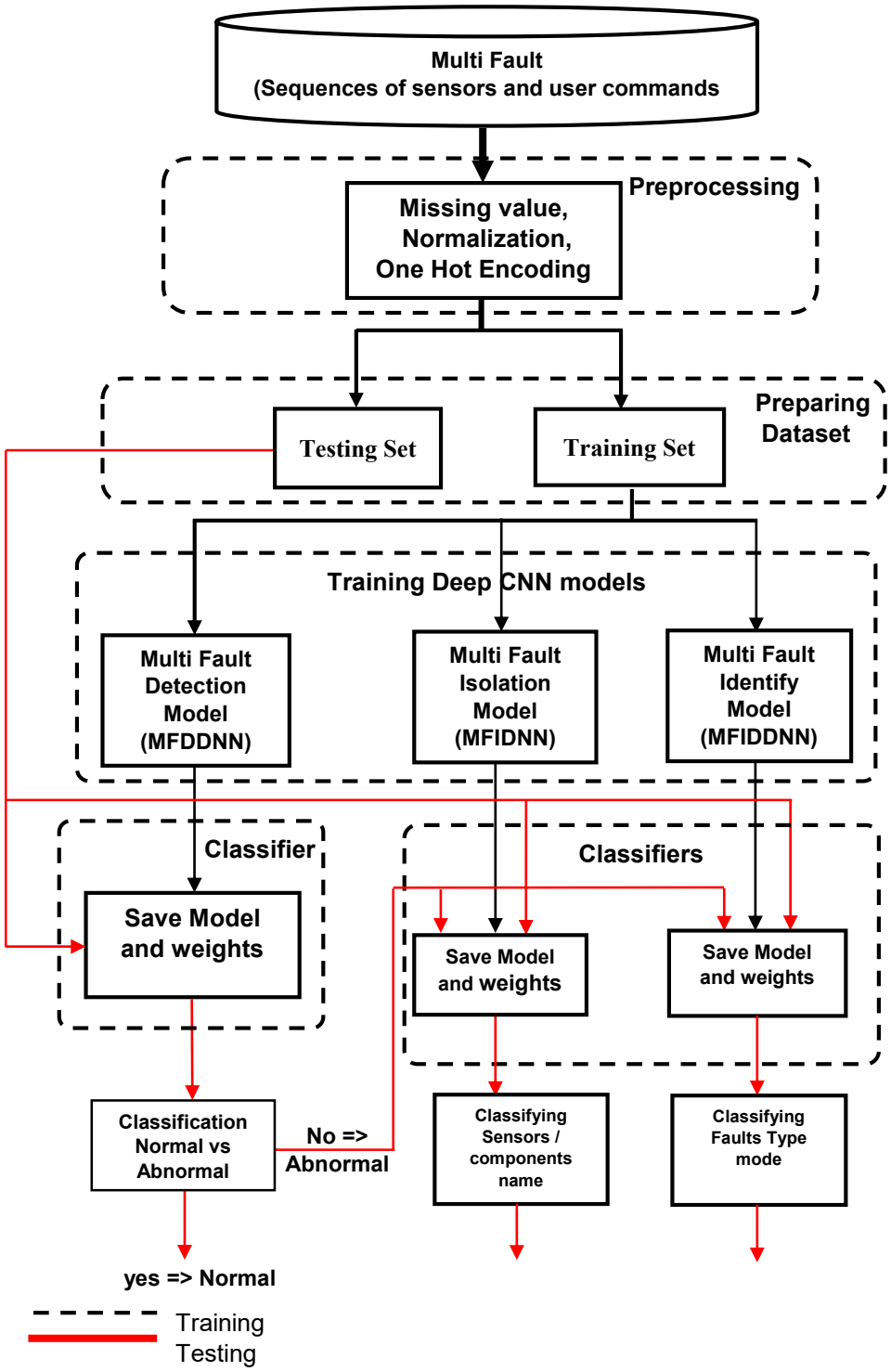
The design of any system is very important because it shows how the system works and explains the exact and practical steps that will be carried out to obtain the required need from it. The proposed diagnosis system is Suitable for working with single and multiple fault datasets. It consists of four main phases for both single fault and multiple faults, as follows: preprocessing dataset, preparing dataset, training, and testing, there are few differences in the configuration of the third phase. The first phase, which is the preprocessing includes imputing missing values, Normalized values, and encoding class label. The second phase includes splitting data into two parts training and testing. The third phase involves build diagnosis classifiers system to detect, isolate, and identified fault mode. The final phase for testing the classifier model and predicted faults location and faults mode type.

Generally, the proposed diagnosis system detects the faults and classify them on the basis of the values of sensors reads and component work into normal

or abnormal. If abnormal, isolate the location of faults then identify the fault type mode. The overall proposed diagnosis system is illustrated in figure (3.1), and algorithm (3.1).



(a)



(b)

Figure 3.1: The Block Diagram of the proposed over all fault diagnosis system:

(a) single, (b) multi.

Algorithm 3.1: The overall proposed methodology

Input: ADAPT
Output: Decision fault location and mode type or normal
Step1: Call pre-processing algorithm(ADAPT)
1 Collect all text scenario files in ADAPT into one csv file.
2 Impute missing values by mean.
3 Normalize imputed dataset by StandardScaler.
4 Encoding class label by one-hot-encoding**.**

Step2 : preparing data
5 Splitting dataset into training set, testing set.

Step3: call training algorithm.
6 Training detection model on the features of all training set.
7 Training isolation model on the features of training set without normal.
8 Training identification model on the features of training set without normal.

Step4: Testing algorithm.

9 Extract the features of the testing set as vector using detection model
10 If prediction of detection model =abnormal

- call isolation model \\ Return location of fault
- call identification model \\ Return type of fault type mode

11 Decision Normal
END

The basic idea behind the proposed system is to create a system that diagnoses single and multiple faults based on the input dataset.

3.2.1 Dataset

The Advanced Diagnostics and Prognostics Testbed (ADAPT) datasets[32] that have been used in this thesis for fault diagnosis based on the EPS of UAV will be described in this section. The dataset consists of a number of scenarios. Each scenario consists of records of different user commands and readings of sensors. There are two types of datasets for single faults and multiple faults. In a single fault dataset, each scenario consists of reading a sequence of values. There is 227 sample of scenarios (Experiment Control), each of which has approximately 2400 (instances) record of reading

the command data and sensor. The data of each scenario has been collected for about five minutes.

The number of features in all scenarios of a single fault is 12, all of 227 were collected together in one CSV file. While for multiple faults, there is 122 sample of scenarios, each scenario has approximately 2400 instances of command data and sensors. The number of features in all scenarios of multiple faults is 42 features, all of 122 scenarios were collected together in one CSV file.

Table (3.1) explains a sample of sensors and component and their name, while Table (3.2) explains a sample of the raw single dataset for one scenario, where the fault occurs in the reading value of voltage (IT240), and fault type mode (offset). Table (3.3) explains a sample of the raw multiple-fault dataset for one scenario, where the fault occurs in the reading value of voltage (IT281) and reading the value of temperature (TE505, TE501). And as a result, fault type's mode (Offset, Offset, and stuck).

Table 3.1: Sample of sensors and components with their name.

Sensors	
Sensors name	**Name symbol**
Temperature sensors	Begin with TE(TE228)
Position sensors	Begin with ISH, ESH (ISH236, ESH244A, ISH262)
Voltage sensors	Begin with E and I (E240, E242, E265, E281, IT240, IT267, IT281)
Speed sensor	Begin with S (ST512)
Light sensor	Begin with LT(LT500,LT505
Actuator sensor	Begin with ESH(ESH244A,ESH141A)
Component	
Component name	**Name symbol**
Battery	Begin with Bat(BAT1,BAT2,BAT3)
Circuit breaker	Begin with CB (CB236,CB262,CB266,CB136)
Relay	Begin with EY (EY244,EY260, EY272,EY170)
Fan	Begin with FAN (FAN416,FAN415,FAN480
Light	Begin with LGT (LGT405,LGT406,LGT407)

Table 3.2: Sample of single fault.

INSTANCES of scenario(Exp–1127–001f–pb–ADAPT–Lite)					
FEATURES	0	Sensor–Data	Antagonist–Data	Antagonist–Data	Antagonist–Data
	1	Time	2010-03-18 05:40:00.524GMT-07:00	2010-03-18 05:40:00.618GMT-07:00	2010-03-18 05:40:00.712GMT-07:00
	2	E242	24.04395	24.07324	23.98535
	3	IT240	16.819	16.78771	16.78771
	4	E265	120.5127		
	5	E240	24.11719	24.13184	24.04395
	6	ESH244A	1	1	1
	7	ISH236	1	1	1
	8	E281	23.58984		
	9	IT267	2.335221		
	10	TE228	74.1387		
	11	IT281	2.397783		
	12	ST516	900		
(Component / sensor) IT240	**Mode** Offset				

Table 3.3: sample of multiple fault.

INSTANCES of scenario (Exp_1222_pb_ADAPT)					
FEATURES	0	SensorData	AntagonistData	AntagonistData	AntagonistData
	1	Time	2010-05-05 07:51:00.845 GMT-07:00	2010-05-05 07:51:00.939 GMT-07:00	2010-05-05 07:51:01.048 GMT-07:00
	2	E181	0.014649		
	3	E242	0	0	0.014649
	4	E265	0.21973		
	5	E281	0.029297		
	6	ST515	0		
	7	IT267	-0.00352		
	8	TE500	68.83079		
	9	IT181	0.008208		
	10	IT167	-0.00743		
	11	ESH141A	0	0	0
	12	ESH344A	0	0	0
	13	IT140	0.064112	0.064112	0.048472
	14	TE501	68.31386		
	15	TE511	65.03813		
	16	TE506	66.71801		
	17	FT525	0		
	18	TE505	67.73875		
	19	IT340	-0.04537	-0.04537	-0.01409
	20	TE128	70.50492		
	21	TE510	73.18247		
	22	ESH144A	0	0	0
	23	E240	23.09178	23.09178	23.09178
	24	E340	24.23438	24.23438	24.23438
	25	TE228	70.5626		
	26	E140	23.89746	23.88281	23.89746
	27	IT281	0.008208		

	28	TE507	67.47108		
	29	ESH341A	0	0	0
	30	E142	0.014649	0	0
	31	ESH241A	0	0	0
	32	TE328	71.77386		
	33	E165	0.073243		
	34	IT240	-0.06101	-0.06101	-0.07665
	35	ESH244A	0	0	0
	36	ISH236	1	1	1
	37	FT520	0		
	38	ISH336	1	1	1
	39	ST516	0		
	40	TE502	67.61868		
	41	ISH136	1	1	1
(Component / Sensors) IT281 TE505 TE501	Mode Offset Offset Stuck				

On the other hand, the single faults' dataset consists of 12 fault type's mode in addition to normal mode (no-fault). Each scenario represents one type of fault mode or there is no fault. While in multi faults' dataset, there are 63 faults types mode in addition to normal mode (no-fault), the faults occur in one sensor reading or more than in the same scenario, or there is no fault.

3.2.2 Preprocessing dataset

The need for the pre-processing phase appears due to the entered data needs to be complete. The pre-processing improves and organizes the input data to prepare it for the training and testing phase. The pre-processing phase consists of three steps include: impute missing data, normalization, and label encoding.

3.2.2.1 Impute data

For several causes, datasets of real-world may contain missing values, often indicated as NULs or NaNs. Furthermore, values of all sensors are not reading in synchronizable, and deep learning does not deal with this form of data. Therefore, the imputation methodology (simpleImputer) uses for handling missing values by calculating the mean for each feature from the

known portion of the data as mentioned in the equation (2.25) and shown in the algorithm (3.2).

Algorithm 3.2:Mean Imputation.

Input:dataset with missing–values
Output:dataset without missing–values
1. For vi ϵ dataset \\ vi vector of feature
2. mean= calculate mean vector for vi
3. for each NaN value in vi \\ NaNs missing values
4. replace NaNs with mean
5. Endfor
6. Endfor
7.**End**

3.2.2.2 Normalization

Usually, the data set needs to be normalized when analyzing multiple sensors simultaneously. For example, one sensor may measure temperature, while another may measure voltage. Due to these values being measured on different scales Thus cannot be usefully compared. Therefore, the standardization technique uses to adjust for such differences and produce stander scalar values. Standerscalar is done according to the equation (2.27) and illustrated in algorithm (3.3)

Algorithm 3.3:Stander Scalar Normalization

Input: dataset
Output: Normalize dataset values
1. For each vi in dataset \\ vi vector of feature
2. mean= calculate mean vector for vi
3. sd=calculate stander division
4. for each value in vi to
5. Normalize vector[ij] =(vi[j]-mean) /sd
6. end for
7. End For
8.End

3.2.2.3 One Hot Encoding

Due to the dataset class label being categorical, these values cannot be directly fed to Network. Therefore, categorical must be converted to a

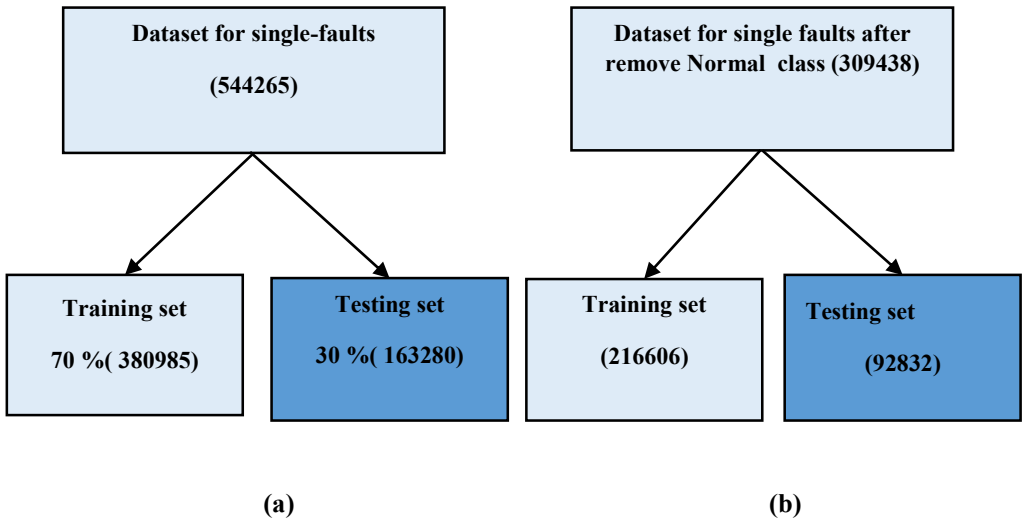
numeric value using the one-hot encoding method. In this method, the classes arrange alphabetically, then transform into binary vectors. Each class label indicates a binary vector for a length equal to the number of classes, where (0) means absence and (1) mean presence class. For instance, the class labels Normal and Abnormal represent as

Normal	[0	1]
Abnormal	[1	0]

3.2.3 Preparing Dataset phase

3.2.3.1 Split Dataset

Training and testing method is the significant part affecting the success of deep learning. The amount of data for training and testing is a significant factor in the success rate, especially when the dataset is imbalanced and huge. Therefore before building the model original dataset is split into two parts training ratio is 70% and the testing ratio is 30% for both single faults and multi faults used in the detection model while the abnormal class labels dataset is used in the isolation model and identification model, as shown in figure (3.2).



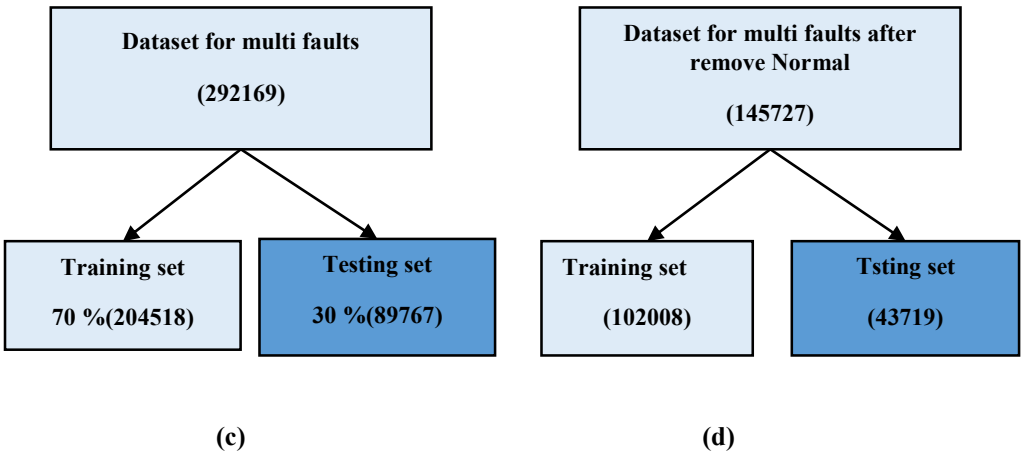


Figure 3.2: Splitting dataset of classifier.
a: Single fault dataset,
b: Only abnormal single dataset.
c: Multi dataset,
d: Only abnormal multi dataset.

3.2.4 Fault Diagnosis Classifiers (FDC)

This stage includes building general and fast FDC system. Which work in the same way for single fault diagnosis and multi fault diagnosis, and its objective to detect, isolate, and identify a fault in the EPS of UAV, which will be explained in detail in this section.

The accomplished CNN layers and their parameters used in this thesis for the FDC for single faults and multi faults are one-dimensional will explain as follows:

1. **1D convolution layer** is responsible for extracting features and consists of many filters (neurons). These filters moved over the entire input data in a sliding window manner. The output of the kernels within the convolutional layers is referred to as feature maps. The feature maps contract relationships from the input data. These feature maps from each kernel put together perfect the convolutional layer. The 1D convolution process mentioned in equation (2.19) can be formulated as an algorithm. Suppose the size of input X as (N, D) where N instances number and D features number, and the number of

weights filters or kernel (W) is k with filter size L*D. Also, suppose the filters move with S stride. Therefore the convolution process is shown in Figure (3.3) and can be written as following Algorithm (3.4).

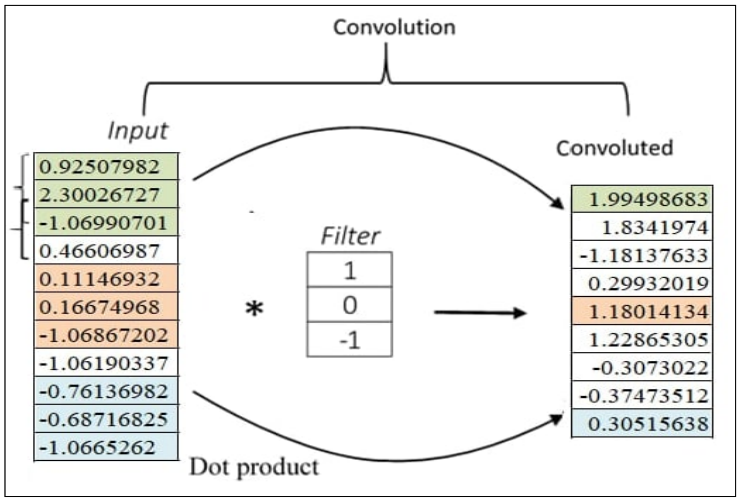


Figure 3.3: The process of convolution when stride=1.

Algorithm 3.4 Convolution Layer.
Input: sensors read (sample of dataset)
Output: Feature map of the sensors read

1. Define the main variables of convolution operation
2. N*D= sensors read \\N number of instance, D number of feature, D=1
3. K=no. of kernel (filter)
4. L = kernel size
5. W =[L*D] \\W is weights filters (kernel)
6. Initialize bias (b) = 1 \\ b is a constant which is b equal to 1
7. For i = 1 to N-L+1 \\loop to calculate feature map for each Kernel
8. net=0
9. For m=0 to L \\ loop equal to L
10. For d= 0 to D
11. net = net +[X(m,d) * W(m,d) + b] \\ calculate dot product between input and weight
12. End d
13. End m
14. Feature-map[i] = f(net) \\ f feature map resulted that equal to the number of kernel
15. End i
16. **END**

2. MaxPooling Compute the maximum value for every patch in the feature map. Therefore it's obtaining the maximum output. The process of

max pooling for 1D data downsamples the input feature map by taking the maximum value over the window, as mentioned in the equation (2.20). The max pooling operation is shown in figure (3.4) and presented in algorithm (3.5).

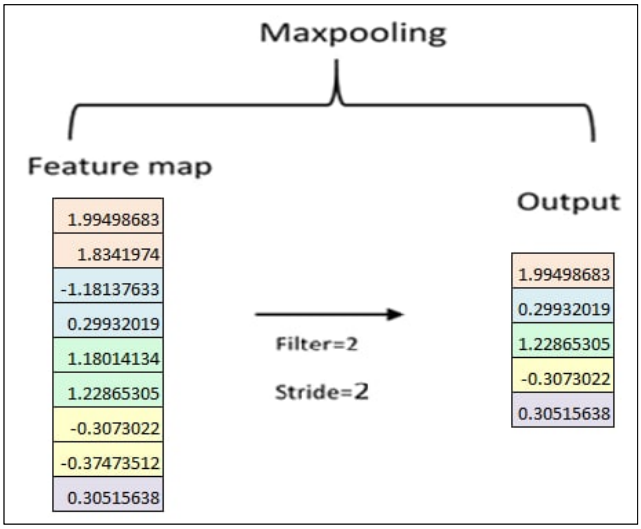


Figure 3.4: The operation of max pooling.

Algorithm 3.5:Max_ pooling layer.
Input: Feature-map
Output: output_map

```
1.  L1 = 0 \\initial index of the resulted output_map
2.  P=pool size
3. S=stride \\ stride of pool
4.  For i = 1 to Feature _map
5.        Max = Feature _map(i) \\ put first value in Max
6.              For  m = i to P \\ loop equal to P
7.                    If  Feature-map  (m) > Max
8.                         Max = Feature-map (m) \\put max value in Max
9.                    End if
10.             End m
11.       output_map[L1 ] = max \\add result to index output_map
12.       L1 = L1 + S
13.  End i
14 End
```

3. **ReLU** is a non-linear activation function used in multilayer neural networks or deep neural networks. This function returns zero if it receives a negative value, but any positive value returns the same received value. This

activation use due to nature of dataset. Equation (2.5) and figure (3.5) illustrates how it works.

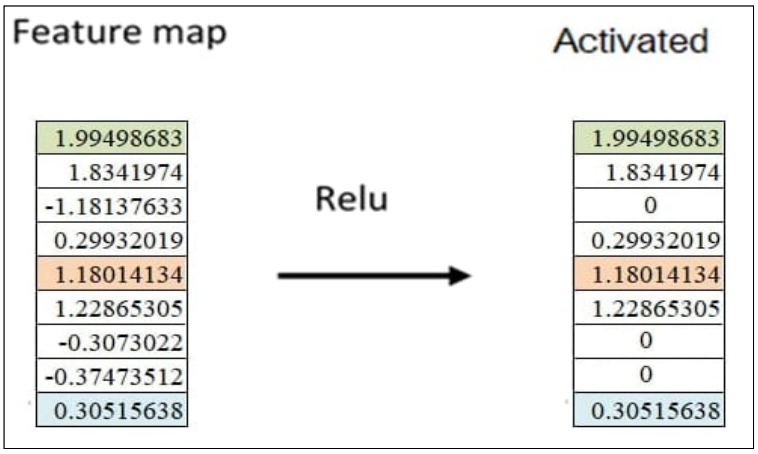


Figure 3.5: The operation of Relu activation function.

4. **Activation Function Linear** is identity function where the activation is proportionate with the input. The linear activation function transforms the neural network into only one layer, as mentioned in the equation (2.4).

Therefore, function output will not be confined between any ranges. The FDC utilize it in the last convolution to work as regression where collect strong features together

5. **Activation Function Leaky ReLu** is a non-linear function that does not make all negative inputs zero but a value near zero. FDC uses Leaky ReLu in MFD because the dataset has many negative values. The equation represents the work of this function mentioned in (2.6).
6. **Flatten layer:-**The flatten layer transforms the input data into a one-dimensional vector then feeds it to the fully connected layer.
7. **Dense layer** is a Fully-Connected (FC) layer used in the last layer, in addition, to using it either before max-pooling or after max-pooling. FDC uses dense at the last layer for classification, where the output of flatten layer utilizes as input to the Dense. The FC layer produces the classification output

equal to the number of classes. The operation of FC is illustrated in equation (2.21).

3.2.4.1. Single Fault Diagnosis Classifier (SFD)

SFD classifier is responsible for working on single fault data, which mean that a single fault occurs in one of the sensors or components. SFD consist of three effective models as described below.

- **Single Fault Detection Model(SFDDNN)**

  The SFDDNN determines if faults occur or not. SFDDNN consists of eight one dimensional 1D convolution layers with a ReLU activation function except the last layer with a linear activation function. The first six convolution layers are connected with the maxpooling layer with pool-size (1), last two convolution layers without the maxpooling layer. The numbers of filters used with the convolution layers are (16, 32, 64,128,256, 512, 1024, and 35) respectively with kernel size (3) and stride size (1).

  The 1D convolution layer performs the role of feature extract to construct the feature maps of the input sample. A flatten layer convert feature maps to vector, output of flatten layer is used as input to fully connected layers (FC) to classify a fault with probabilistic values between 0 and 1. The weights of each layer are saved to generate the output of these layers, and after that output of the current layer is passed to the next layer as inputs until reaches to finally layer. The prediction for this model determines the model's overall functioning. If it is normal, there is no need to go through isolate model and the identifier model, otherwise, it will pass two models to determine the location and type of fault. The architecture of the SFDDNN model illustrate in figure (3.6).

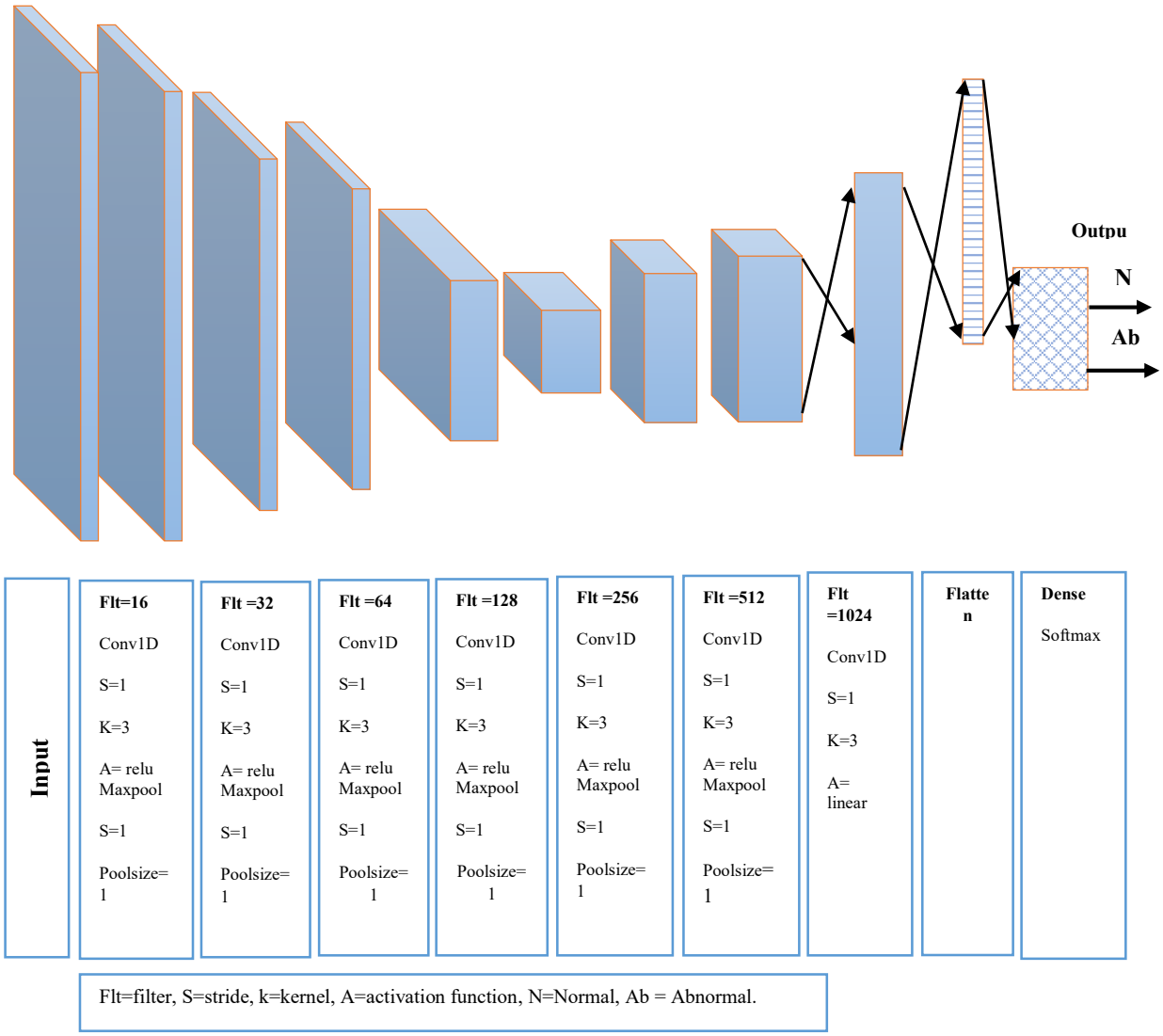


Figure3.6: The structure of SFD model.

- **Single Fault Isolate Model (SFIDNN)**

The SFIDNN model works if the prediction of the SFDDNN model is (abnormal), which is responsible for determining the location of the fault. The model layers are similar to the layers of the SFDDNN model with little difference in filters number of the last convolution layer 145 and number of class labels in the dense layer 24. Where number of classes equal to number of sensor and component as mention in figure (2.1).

- **Single Fault Identify Model (SFIDDNN)**

The last model of the SFD works in synchronization with the SFIDNN model. SFIDDNN is responsible for classifying the fault type mode, which consists of layers similar to the SFDDNN model except the last convolution layer have 85 filters and the number of classes in the Dense layer (12). The weights of each model are different from others. A trained SFD obtain after completing training SFDDNN, SFIDNN and SFIDDNN models on the training dataset.

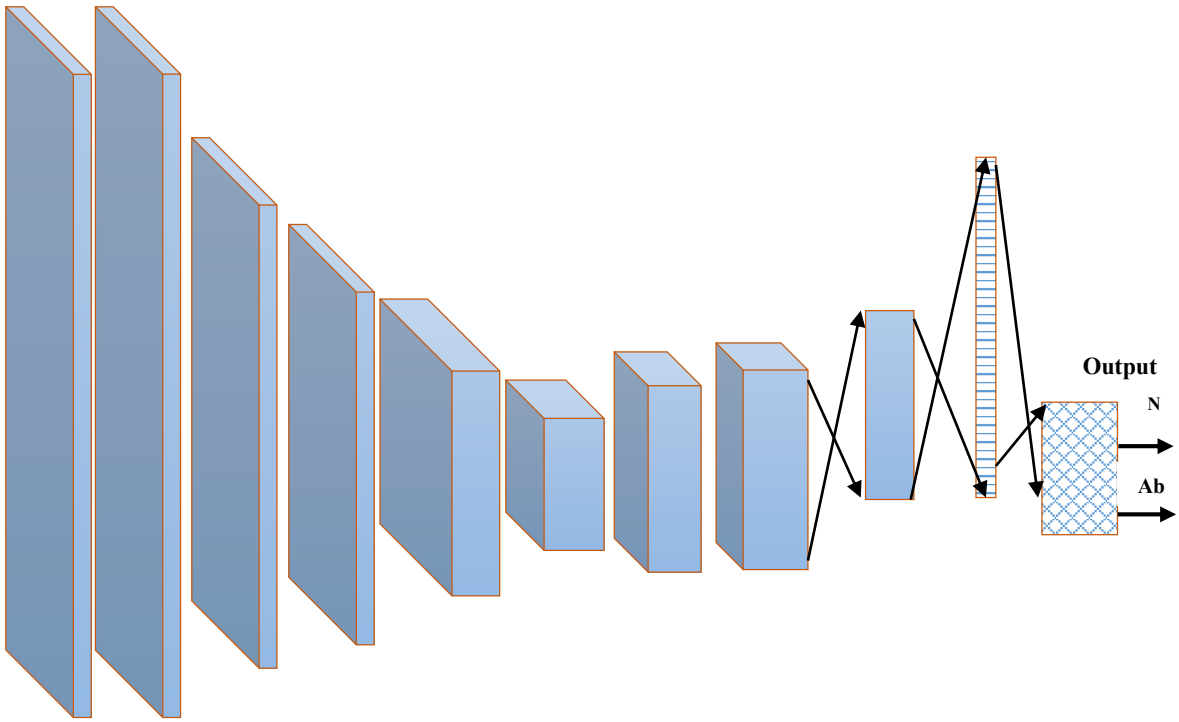
3.2.4.2. Multi Fault Diagnosis Classifier (MFD)

Before starting to describe MFD, it is necessary to clarify the basic ideas espoused by the classifier. MFD can detect, isolate, and identify one or more faults at the same time. The general MFD consists of three fast classifier models that will be explained as follows:

- **Multi Fault Detection Model (MFDDNN)**

This model determines the work of the MFD classifier as a whole, as it works to determine the presence or absence of the fault. MFDDNN is built based on 1D CNN, which consists of eight convolution layers with an activation function LeakyReLU and linked with maxpooling, except for the last convolution layer whose activation function is linear and without maxpooling. Filters of convolution layers are (16, 32, 64,128,256, 512, 1024, and 345) respectively.

The first convolution layer is the input layer, while the last layer is a dense layer to predict the output. The prediction of this model is normal or abnormal if normal there is no need to train other models used in this classifier. The architecture of the MFDDNN model is shown in figure (3.7).



Input	Flt =16 Conv1D S=1 K=3 A=Leaky-Relu Maxpool S=2 Poolsize =2 A=Leaky	Flt =32 Conv1D S=1 K=3 A=Leaky-Relu Maxpool S=1 Poolsize= 2	Flt =64 Conv1D S=1 K=3 A=Leaky-Relu Maxpool S=1 Poolsize =1	Flt =128 Conv1D S=1 K=3 A=Leaky-Relu Maxpool S=1 Poolsize= 1	Flt =256 Conv1D S=1 K=3 A=Leaky-Relue Maxpool S=1 Poolsize =1	Flt =512 Conv1D S=1 K=3 A=Leaky-Relu Maxpool S=1 Poolsize =1 A=Leaky-Relu	Flt =1024 Conv1D S=1 K=3 A= linear	Flatten	Dense Softmax

Flt=filter, S=stride, k=kernel, A=activation function, N=Normal, Ab = Abnormal.

Figure 3.7: The structure of MFD model.

- **Multi Fault Isolate Model (MFIDNN)**

MFIDNN model determines the sensors or components where faults have occurred to isolation them. This model consists of layers similar to MFDDNN layers except the last convolution layers have 850 filters and the 165 number of the classes in dese. Where number of classes equal to number of sensor and component as mention in figure (2.2).

- **Multi Fault Identify Model (MFIDDNN)**

This part of the classifier is responsible for identifying the mode faults that occur in an EPS. The layers of MFIDDNN are similar to the layers of the MFDDNN model except for number of filters in the last convolution layer 345, and number of classes' label in dense 63.

3.2.5 The FDC Training phase

Each deep neural network (DNN) in the FDC is trained separately from the other for both SFD and MFD, the goal of training the networks separately is to determine if a fault occurs or not and find the location and fault types mode. To train each network separately the training data must be provided for each network. So this classifier uses an ADAPT dataset that is sorted into two groups of datasets for single faults and multi faults. Each classifier is trained on its corresponding dataset, each model in FDC is trained for a specific classification. In SFDDNN and MFDDNN models trained to classify if the system works normal or abnormal while SFIDNN, MFIDNN models trained to classifier faults location, and SFIDDNN, MFIDDNN to classifier fault type mode. Algorithm (3.6) illustrates the training process of the FDC system.

Algorithm 3.6. Training 1D-CNN.
Input: training dataset (TRD) features, labels.
Output: fault diagnosis 1D-CNN model.
1. Initialize: biases and weights randomly. \weights of kernels
2. For each epoch Do: \\ epoch No. of training
3. Process records of TRD data.
4. Per each batch size update and save the weights. \\ Batch size number of record
5. Save model and weights.
6. End

3.2.6 The FDC Testing Phase

In the testing phase, weights of each trained model in FDC were used to test the classifier from the unseen dataset. Inputs of each model in SFD and MFD are features vectors, which uses to predict the system in normal mode

or faults, location and faults type mode. If the prediction of SFDDNN in SFD is abnormal, features vectors pass to SFIDNN to predict fault location and to SFIDDNN to predict fault type mode.

In the same way, if the prediction of MFDDNN in MFD is abnormal, features vectors pass to MFIDNN to predict fault/s location and to MFIDDNN to predict fault/s type mode. The testing phase of the FDC can be represented during the following algorithm (3.7).

Algorithm 3.7. Testing procedure for 1D-CNN model.
Input: Testing dataset (TSD) features, labels.
Output: Decision of FDC.
1 Read one record of TSD data per each time.
2 Input record on saved model.
3 Make decision.
4 End

Chapter Four

Experimental-Results and Discussions

Chapter four
Experimental Results and Discussion

4.1 Overview

This chapter presents and discusses the experimental results and effective results obtained from each stage of implementation in the EPS of UAV. Also present a description of the dataset used with the proposed system, hardware and software requirements in implementing the proposed system. The results of all stages are arranged based on their appearance in Chapter three.

4.2 System specification

The implementation of a deep learning approach requires high computing resources. So the proposed system was implemented using the following hardware and software requirements.

- **Hardware**
  1. Central Processing Unit (CPU): Intel Core i710750h.
  2. RAM: 32 GB.
  3. Graphics Processing Unit (GPU): NVIDIA 1660TI.
  4. Hard Disk: 512 GB.
- **Software**
  1. Operating System: Windows 10 64bit.
  2. Programming Language: Python3.7.

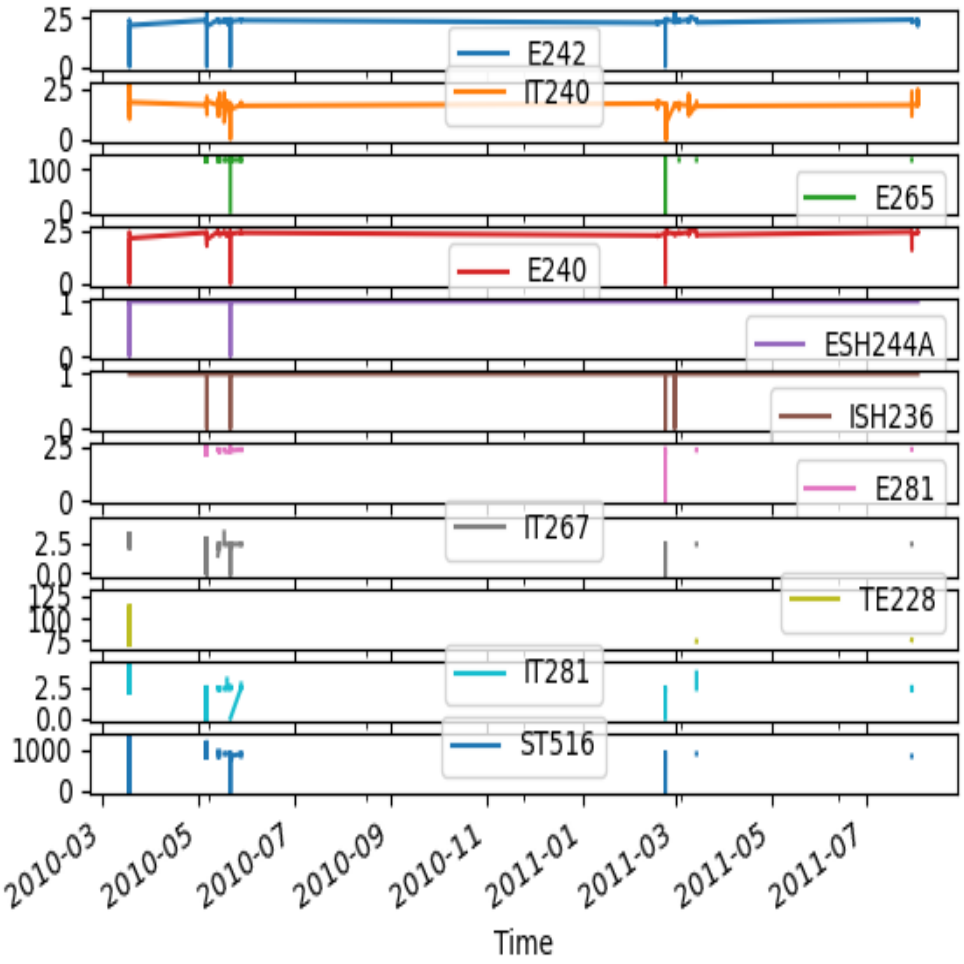
4.3 Results of the proposed system's implementation phases

In this section of the thesis, a detailed description of the fault diagnosis system based on sensors read of the EPS in UAV. A set of measures will be explained

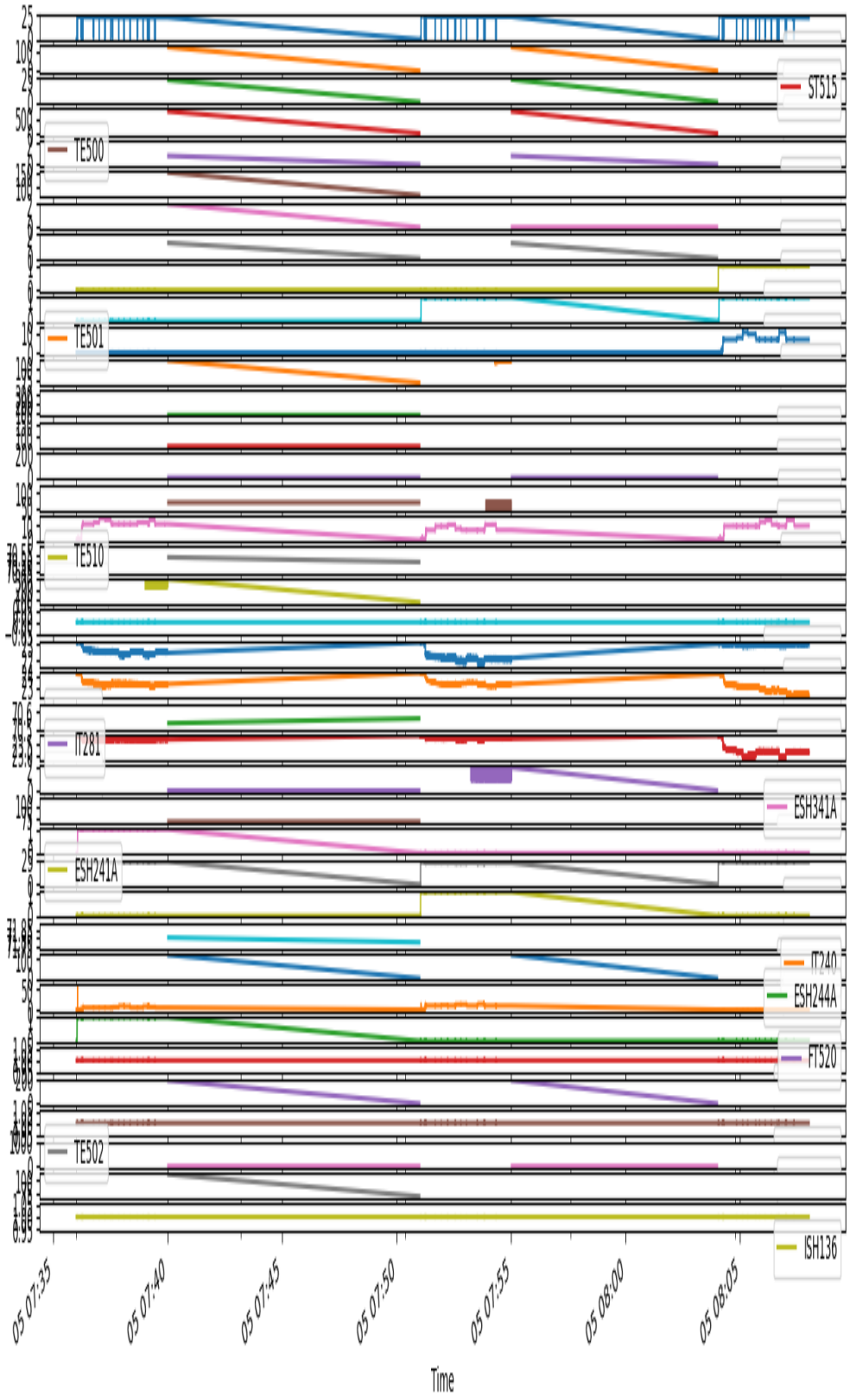
and implemented and a detailed description of each phase that the system passes through will be presented. As following.

4.3.1 Results of data preprocessing phase

This stage presented in section 3.3 is applied to both the SFD dataset and the MFD dataset. Figure (4.1) presents SFD and MFD that are entered into the proposed system before preprocessing for each sensor. All datasets that enter the proposed system must go through the pre-processing stage, whether it is in the training phase or the testing phase.



(a)



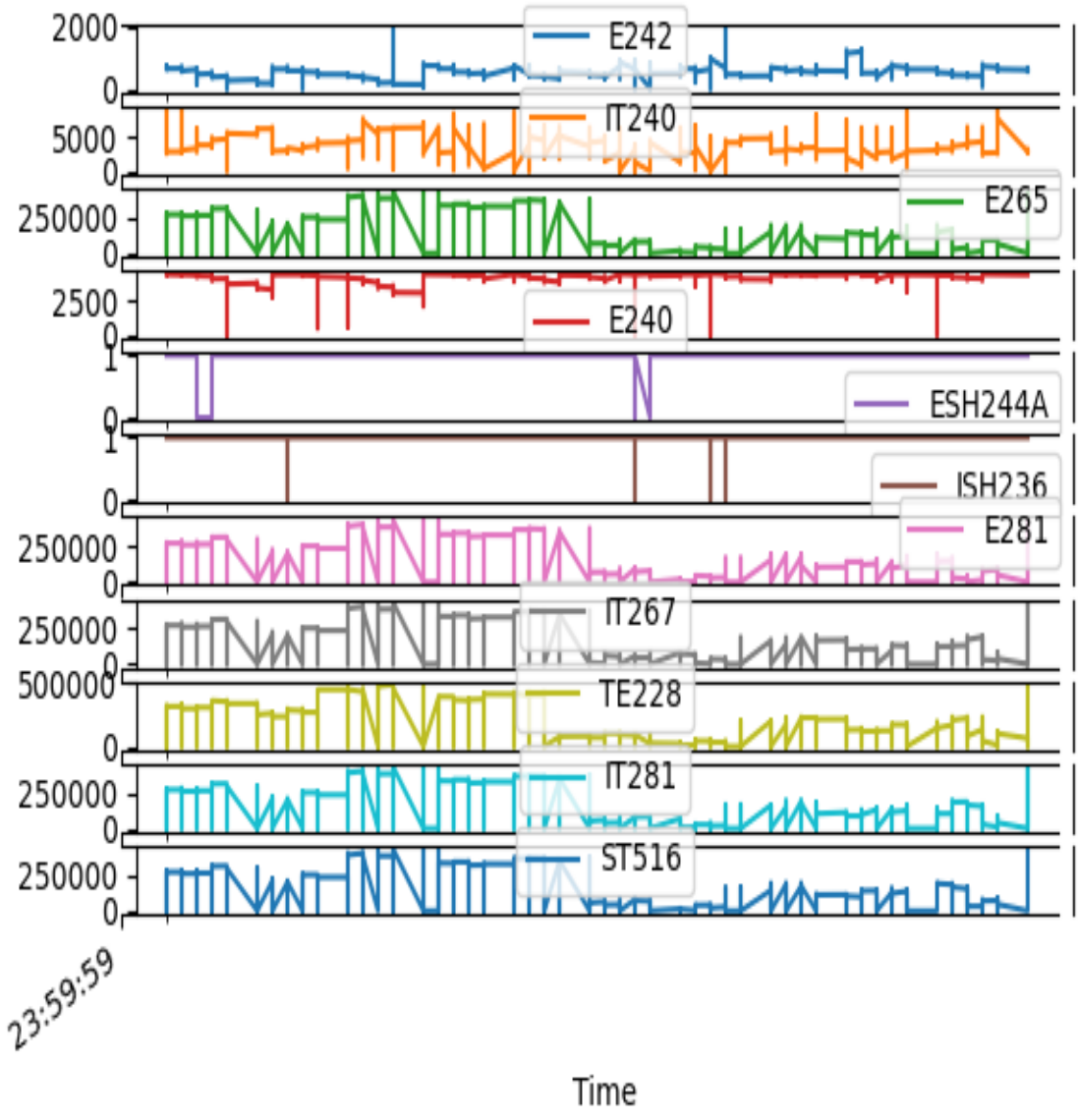
(b)

Figure 4.1: Raw dataset before pre-processing for: (a) single dataset, (b) multi dataset.

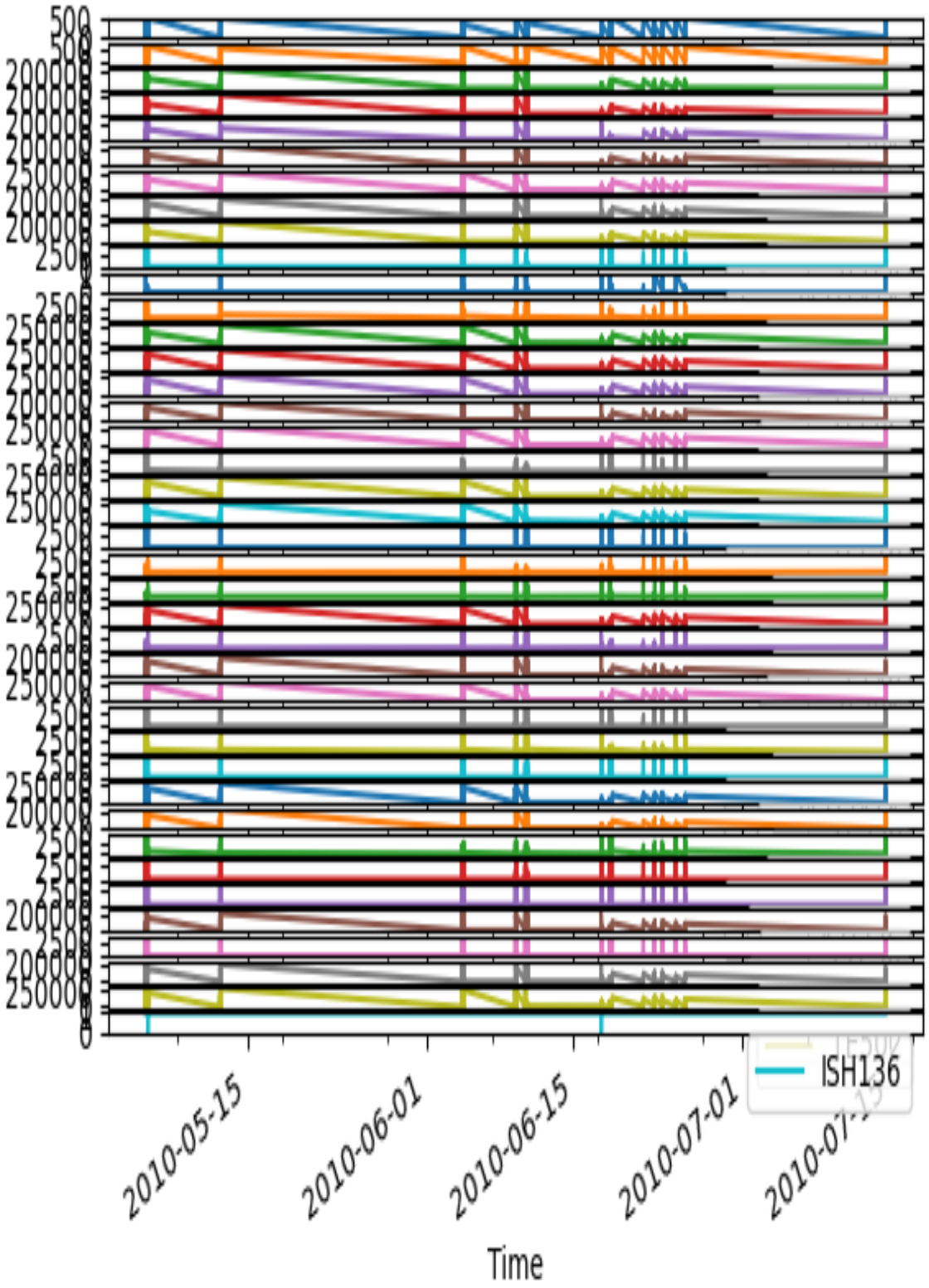
The preprocessing that includes a set of steps to give us accurate, perfect data to deal with in the classification phase consists of two steps as follow:

4.3.1.1 Cleaning data

The first and the most important step in the preprocessing is filling the missing values. Figure (4.2) illustrates single dataset and multi- dataset after imputing the missing values by using mean.



(a)



(b)

Figure 4.2: The result of apply simpleImputer with mean strategy, (a) single dataset, (b) multi dataset.

4.3.1.2 Normalization

After cleaning data by simpleimputer, the second stage attained in applying the Normalization technique. The standardScaler is applied to the cleaning dataset, whose aim is to substitute the values of numeric columns in the dataset to specific scales without changing variations in the ranges of values. The result of using different preprocessing techniques with different epochs shown in tables (4.1) and (4.2), where table (4.1) shows filling the missing values by linear Interpolation with different normalization techniques, where its explain that model suffer from underfitting where accuracy train higher than accuracy test, while table (4.2) shows the use of a mean imputation with various normalizing strategies to fill in missing variables. Figure (4.3) shows the normalized dataset.

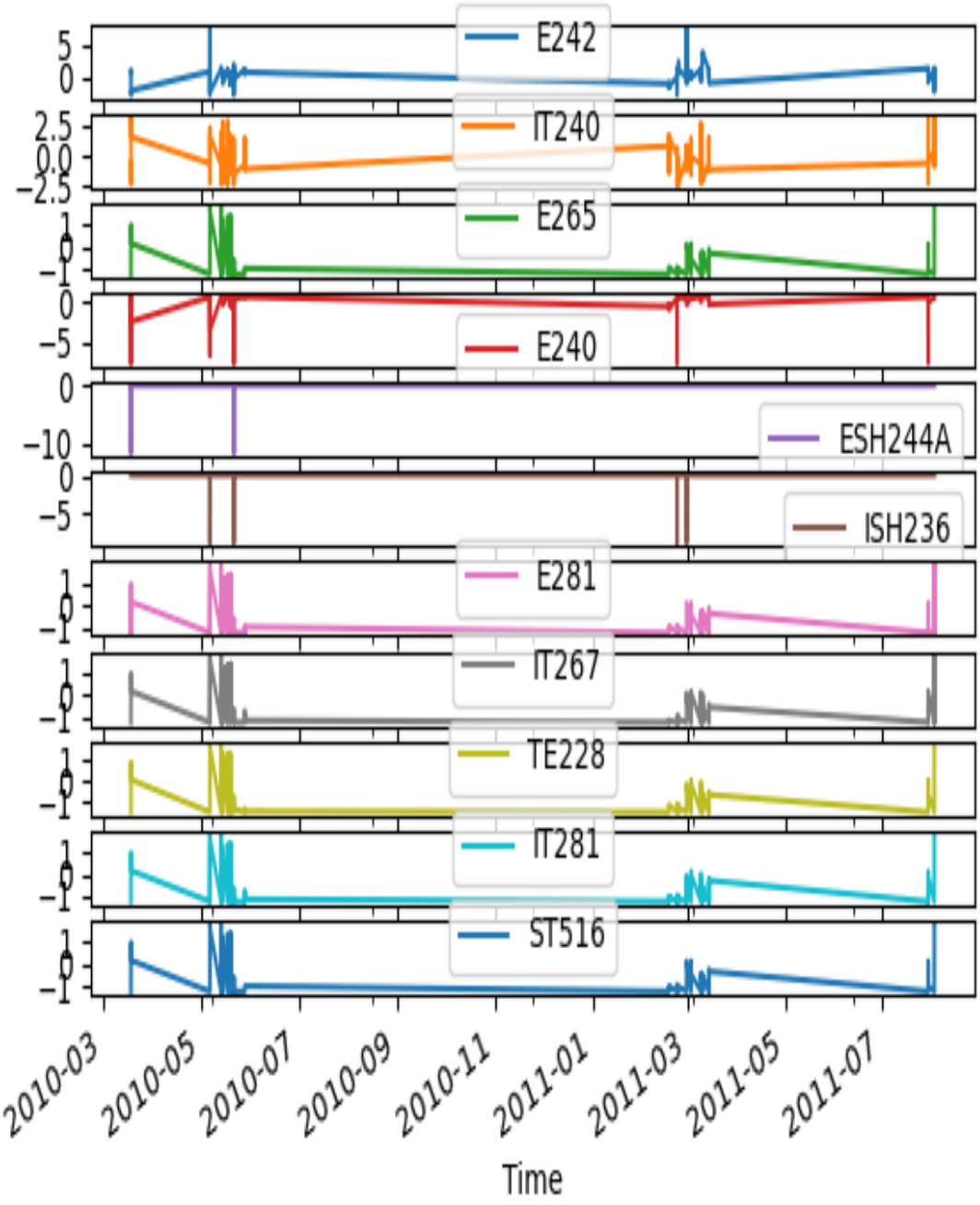
Table 4.1: Linear interpolation with different normalization techniques.

Technique	Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
StanderScaler()	100	0.9242	0.1568	0.9053	0.2500
	200	0.9522	0.1003	0.8996	0.2791
	400	0.9702	0.0631	0.8932	0.1321
MinMaxScaler(-1,1)	100	0.8685	0.2785	0.8990	0.2911
	200	0.8957	0.2161	0.8801	0.2932
	400	0.9039	0.2001	0.8887	0.4701
MinMaxScaler(0,1)	100	0.8440	0.3266	0.8360	0.3577
	200	0.8789	0.2544	0.8777	0.6010
	400	0.8954	0.2163	0.8705	0.2885

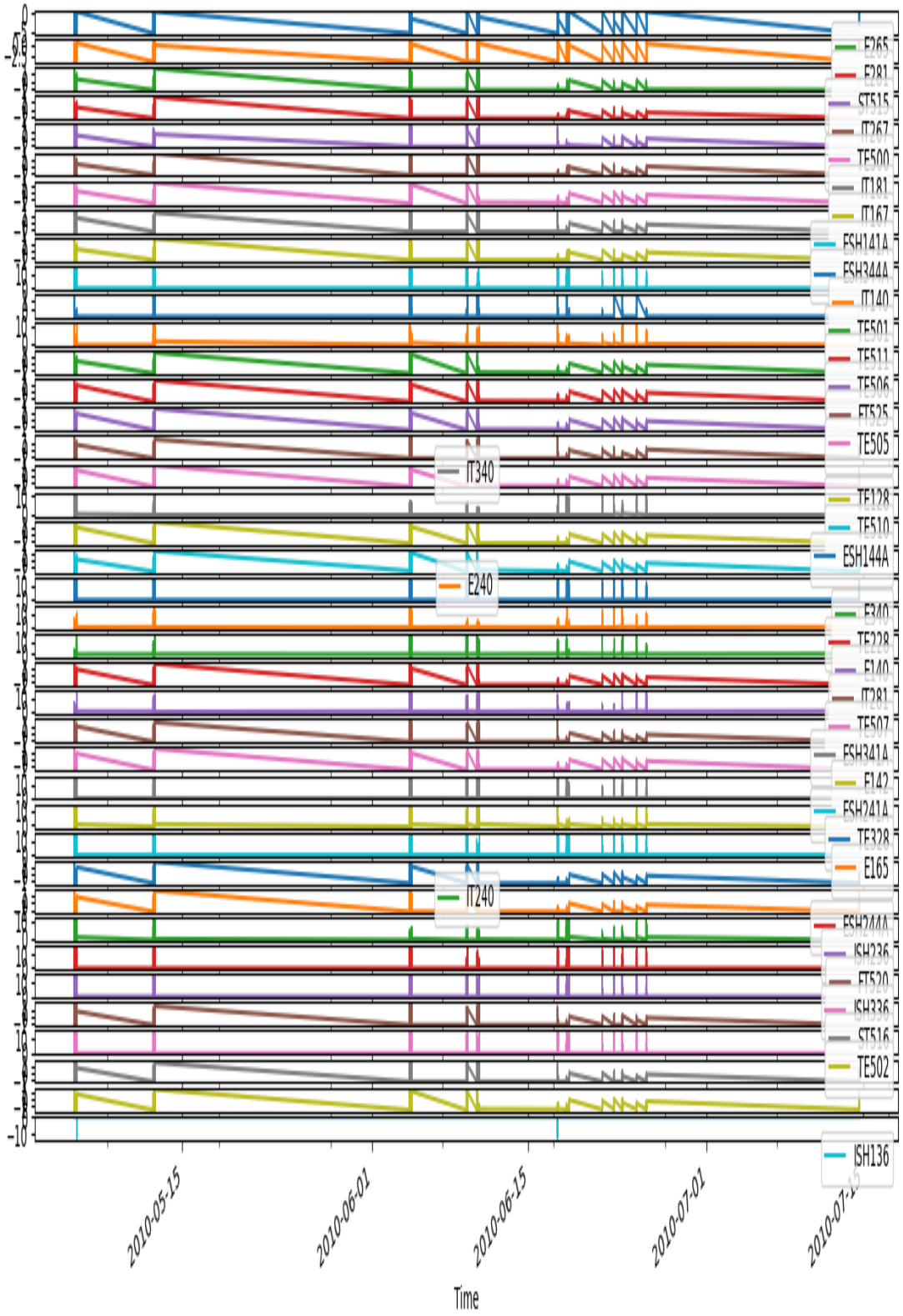
Table 4.2 Mean imputation with different normalization techniques.

Technique	Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
StanderScaler()	100	0.9187	0.1627	0.9181	0.1859
	200	0.9388	0.1221	0.9349	0.1357
	400	0.95460	0.1045	0.95411	0.1326

MinMaxScaler(-1,1)	100	0.9058	0.1862	0.9044	0.2239
	200	0.9142	0.1673	0.9106	0.2481
	400	0.9279	0.1415	0.9241	0.5276
MinMax-Scaler(0,1)	100	0.8828	0.2304	0.8676	0.2581
	200	0.9139	0.1757	0.8925	0.2619
	400	0.9100	0.1783	0.9014	0.2534



(a)



(b)

Figure 4.3: result of normalize dataset, (a) single dataset, (b) multi dataset.

4.3.1.3 One Hot Encoding

Method for handling variables categorical, where the class label for each model in the classifier is encoding, for both single dataset and multi-dataset. Table (4.3) explains applying one hot encoding for the SFD model.

Table 4.3: Apply one-hot encoding for the SFD class label.

Detection model	
Classes label	'Abnormal', 'normal'
One hot encoding normal	0 1
Isolation model	
classes label	'AC483', 'CB236', 'CB262', 'CB266', 'CB280', 'DC485', 'E240', 'E242', 'E265', 'E281', 'ESH244A', 'EY244', 'EY260', 'EY272', 'EY275', 'EY284', 'FAN416', 'INV2', 'ISH236', 'IT240', 'IT267', 'IT281', 'ST516', 'TE228'
One hot encoding IT240	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Identify model	
Classes label	'Drift', 'FailedOff', 'FailedOpen', 'IntermittentOffset', 'IntermittentResistanceOffset', 'Offset', 'OverSpeed', 'ResistanceDrift', 'ResistanceOffset', 'Stuck', 'StuckOpen', 'UnderSpeed'
One hot encoding Offset	0 0 0 0 0 1 0 0 0 0 0 0

4.3.2 Results and Experiments of SFD

The SFD is used for fault diagnosis. As indicated earlier in chapter three, this classifier consists of three models. In general, three main parameters control the training process of models used in this thesis, the number of epochs, batch size, and learning rate. The results and experiments of each model will be described below.

- **Results SFDDNN**

This model used the single dataset to classify if there is fault or normal, which consisted of (544265) samples were 70 % for training and 30 % for

testing. The following tables show the experimental results of the model for adjusting the values of these parameters during training and testing by adopting the Adam optimization with initial learning-rate 0.0016.

Table (4.4) shows the summary representation of the SFDDNN model architecture, where parameter number calculate by (number of filter*(filter size+1)). Table (4.5) shows the process of adjusting batch size by comparison of the values of the accuracy and values of loss function during training and testing when the batch size changes with the constant learning rate value and a number of epochs. Table (4.6) shows the difference between achieved results for values of accuracy and the loss function for different epochs. Table (4.7) shows the difference between achieved results for values of accuracy and the loss function for different learning rate.

Table 4.4: The summary of SFDDNN model.

Layer	Output	Parameters number
conv1d_1	(None, 10, 16)	64
Max–pooling1d_1	(None, 10, 16)	0
conv1d–2(Conv1D)	(None, 8, 32)	1568
Max–pooling1d_2	(None, 8, 32)	0
conv1d–3(Conv1D)	(None, 6, 64)	6208
Max–pooling1d_3	(None, 6, 64)	0
conv1d–4(Conv1D)	(None, 4,128)	24704
Max–pooling1d_4	(None, 4, 128)	0
conv1d–5(Conv1D)	(None, 2, 256)	98560
Max–pooling1d_5	(None, 2, 256)	0
conv1–6(Conv1D)	(None, 2, 512)	131584
Max–pooling1d_6	(None, 2, 512)	0
conv1d–7(Conv1D)	(None,2, 1024)	525312
conv1d–8(Conv1D)	(None,2, 35)	35875
Flatten–1(Flatten)	(None, 70)	0
Dense–1(Dense)	(None, 2)	142
Total–params: 824,017 **Trainable–params: 824,017** **Non–trainable–params: 0**		

Table 4.5: Results SFDDNN model of 100 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.8147	0.3479	0.8165	0.6096
256	0.9128	0.1751	0.9108	0.1847
512	0.9192	0.1614	0.9182	0.1660
1024	0.9187	0.1627	0.9181	0.1859

Table 4.6: Result SFDDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9187	0.1627	0.9181	0.1859
150	0.9378	0.1224	0.9319	0.1363
300	0.954480	0.1055	0.953771	0.1334
500	0.9677430	0.1022	0.9671	0.1248

Table 4.7: Results SFDDNN model of 100 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9204	0.1591	0.9158	0.1698
0.0001	0.8879	0.2238	0.8903	0.2229
0.0016	0.9187	0.1627	0.9181	0.1859
0.0004	0.9148	0.0699	0.9080	0.0935

The final parameters values for training the SFDDNN model are as follows

Batch size=1024

Epochs=500

Learning rate=0.0016

Result get from final parameters for training are accuracy: 0.9677, loss: 0.1022 and for testing are accuracy: 0.9671, loss: 0.1248. Prediction time for all testing dataset is 9316.659927368164 millisecond, while training time for each epoch 7 second. Figure (4.4) shows the curve of accuracy, and figure (4.5) shows the curve of loss function.

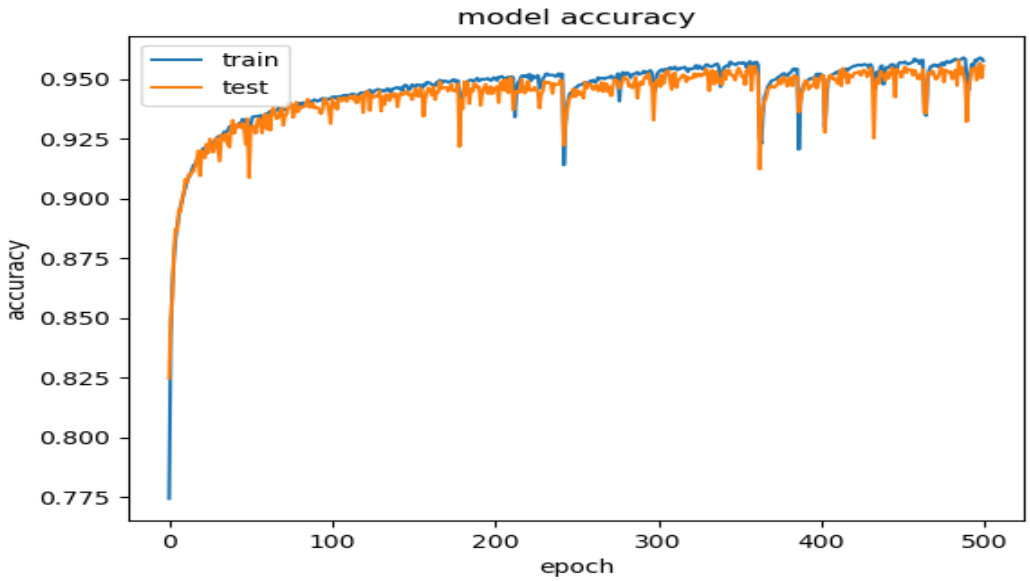


Figure 4.4: SFDDNN Accuracy curve.

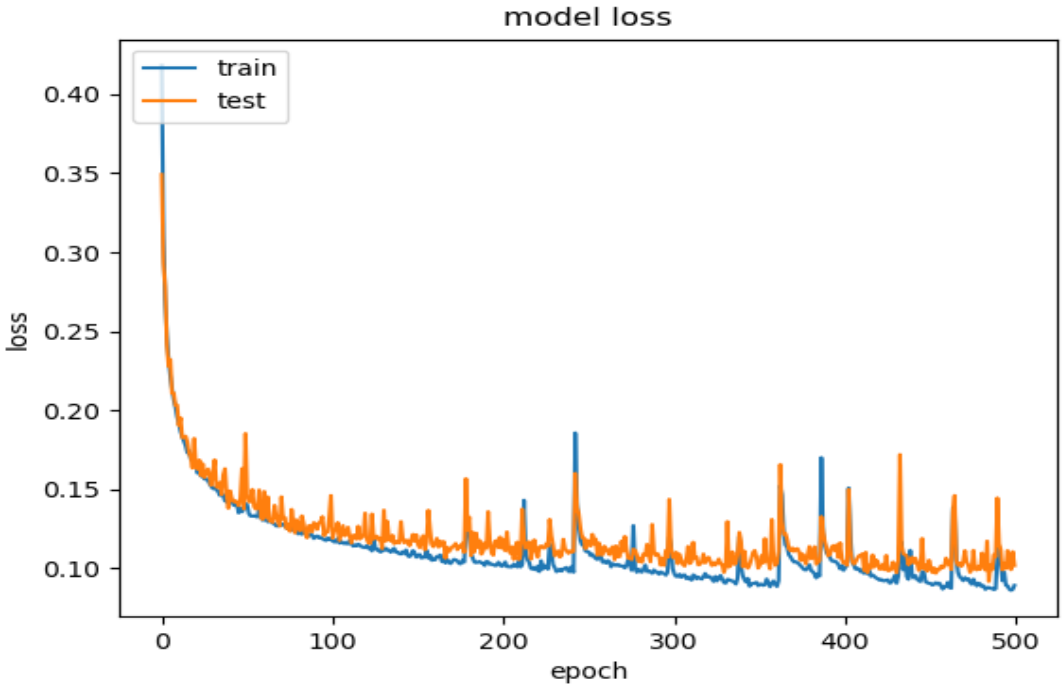


Figure 4.5: SFDDNN Loss function curve.

- **Results SFIDNN**

This model is triggered if the prediction of the first model is abnormal. SFIDNN trains on the same dataset used in SFDDNN after dropping all instances with Normal classes. The summary representation of the SFIDNN model architecture is shown in table (4.8), table (4.9) shows the process of adjusting batch size by comparison of the values accuracy and values of the loss function during training and testing when the batch size changes with the constant learning rate value and a number of epochs.

Table (4.10) shows the difference between achieved results for values of accuracy and the loss function for different epochs with the constant values for batch size and learning rate. Table (4.11) shows the difference between the achieved results for values of accuracy and the loss function for different learning rates with constant values for batch size and epochs.

Table 4.8: The summary of SFIDNN model.

Layer type	Output	Parameters number
conv1d–1	(None , 9, 16)	64
Max‹-pooling1d_1	(None , 9, 16)	0
conv1d–2 (Conv1D)	(None , 7,32)	1568
Max–pooling1d_2	(None , 7,32)	0
conv1d–3 (Conv1D)	(None , 5, 64)	6208
Max–pooling1d_3	(None , 5, 64)	0
conv1d–4 (Conv1D)	(None , 3, 128)	24704
Max–pooling1d_4	(None , 3, 128)	0
conv1d–5 (Conv1D)	(None , 1, 256)	98560
Max–pooling1d_5	(None , 1, 256)	0
conv1d–6 (Conv1D)	(None , 1, 512)	131584
Max–pooling1d_6	(None , 1, 512)	0
conv1d–7 (Conv1D)	(None , 1, 1024)	525312
conv1d–8 (Conv1D)	(None , 1, 145)	148625
Flatten–1 (Flatten)	(None , 145)	0
Dense–1 (Dense)	(None , 24)	3504
Total–params: 940,129 **Trainable–params: 940,129** **Non-trainable–params: 0**		

Table 4.9: Results SFIDNN model of 100 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.9403	0.1392	0.9297	0.1769
256	0.9449	0.1321	0.9130	0.4207
512	0.9512	0.1161	0.9314	0.1933
1024	0.9513	0.1175	0.9458	0.1371

Table 4.10: Results SFIDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9519	0.1181	0.9524	0.1236
150	0.9636	0.0924	0.9604	0.1028
300	0.9634	0.0900	0.9614	0.1078
500	0.9843	0.0400	0.9851	0.0418

Table 4.11: Results SFIDNN model of 100 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9538	0.1120	0.9464	0.1301
0.0001	0.9029	0.2437	0.9002	0.2469
0.0016	0.9532	0.1123	0.9512	0.1157
0.0004	0.9508	0.1178	0.9307	0.1703

The final parameters values for training the SFIDNN model are as follows:

Batch size=1024

Epochs=500

Learning rate=0.0016

Result get from final parameters for training are accuracy: 0.9843, loss: 0.0400 and for testing are accuracy: 0.9851, loss: 0.0418. Prediction time for all testing dataset is 4004.8723220825195 millisecond, while training time for each epoch 4 second. Figure (4.6) shows the curve of accuracy, Figure (4.7) shows the curve of the loss function.

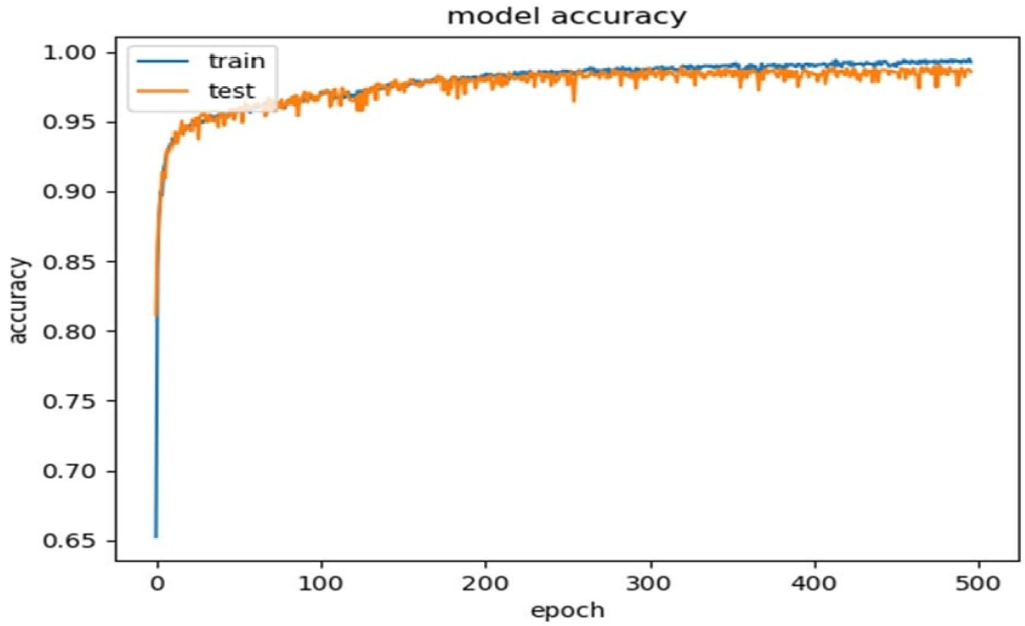


Figure 4.6**:** Accuracy curve of SFIDNN model.

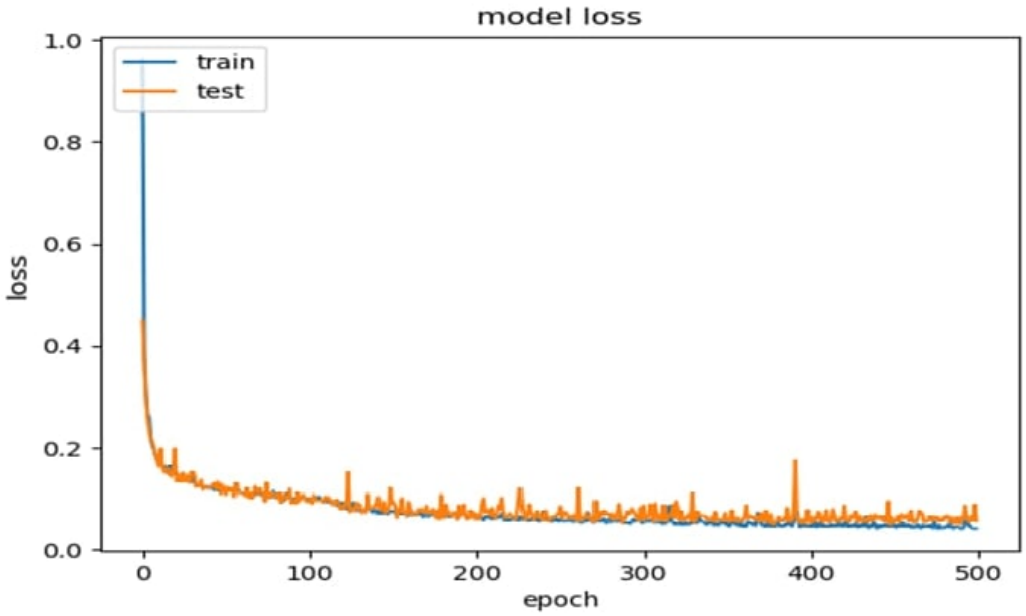


Figure 4.7: Loss function curve of SFIDNN model.

- **Results SFIDDNN**

After locating the fault in the fault isolation model, this module works to determine the type of fault (fault mode). SFIDDNN model is trained on the same data that the isolation model was trained. The summary of the SFIDDNN model is shown in Table (4.12).

Table 4.12: The summary of SFIDDNN model.

Layer type	Output	Parameters number
conv1d–1	(None,9, 16)	64
Max–pooling1d_1	(None,9, 16)	0
conv1d–2 (Conv1D)	(None,7, 32)	1568
Max–pooling1d_2	(None,7, 32)	0
conv1d–3 (Conv1D)	(None,5, 64)	6208
Max–pooling1d_3	(None,5, 64)	0
conv1d–4 (Conv1D)	(None,3, 128)	24704
Max–pooling1d_4	(None,3, 128)	0
conv1d–5 (Conv1D)	(None,1, 256)	98560
Max–pooling1d_5	(None,1, 256)	0
conv1d–6 (Conv1D)	(None,1, 512)	131584
Max–pooling1d_6	(None,1, 512)	0
conv1d–7 (Conv1D)	(None,1, 1024)	525312
conv1d–8 (Conv1D)	(None,1, 85)	87125
Flatten–1 (Flatten)	(None,85)	0
Dense–1 (Dense)	(None,12)	1032
Total–params: 876,157 **Trainable–params: 876,157** **Non-trainable–params: 0**		

The experimental results of this model during the stages of training and testing showed the difference between achieved results for the accuracy and loss function as described tables below.

Table (4.13) show the result of using different batch size, table (4.14) show the result of using different epochs, and table (4.15) show the result for different learning rate.

Table 4.13: Results SFIDDNN model of 100 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.9467	0.1264	0.9426	0.1411
256	0.9522	0.1140	0.9531	0.1134
512	0.9517	0.1114	0.9386	0.1459
1024	0.9412	0.1427	0.9208	0.2682

Table 4.14: Results SFIDDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9522	0.1140	0.9531	0.1134
150	0.9577	0.1017	0.9543	0.1202
300	0.9653	0.0841	0.9578	0.1251
500	0.9869	0.0343	0.9868	0.0352

Table 4.15: Results SFIDDNN model of 100 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9543	0.1078	0.9510	0.1176
0.0001	0.9155	0.2008	0.9126	0.2084
0.0016	0.9522	0.1140	0.9531	0.1134
0.0004	0.9567	0.1011	0.9520	0.1140

The final parameters values for training the SFIDDNN model are as follows

Batch size=512

Epochs=500

Learning rate=0.0016

Result get from final parameters for training are Accuracy: 0.9869, loss: 0.0343and for testing are accuracy: 0.9868, loss: 0.0352. Prediction time for all testing dataset is 3806.285858154297, while training time for each epoch 5 second. .Figure (4.8) shows the curve of accuracy, Figure (4.9) shows the curve of the loss function.

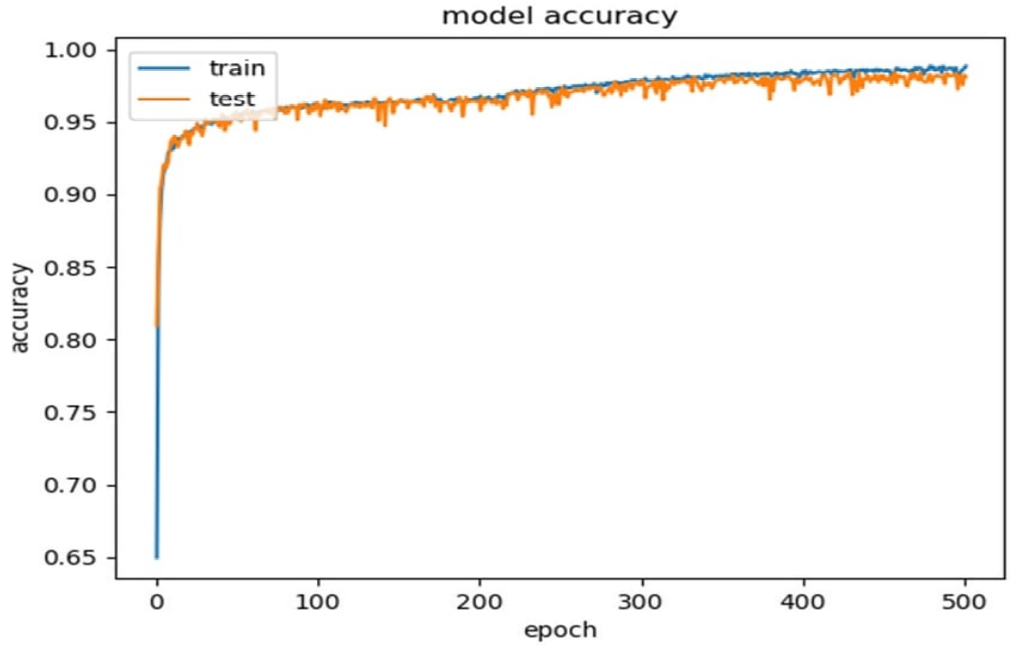


Figure 4.8: Accuracy curve of SFIDDNN model.

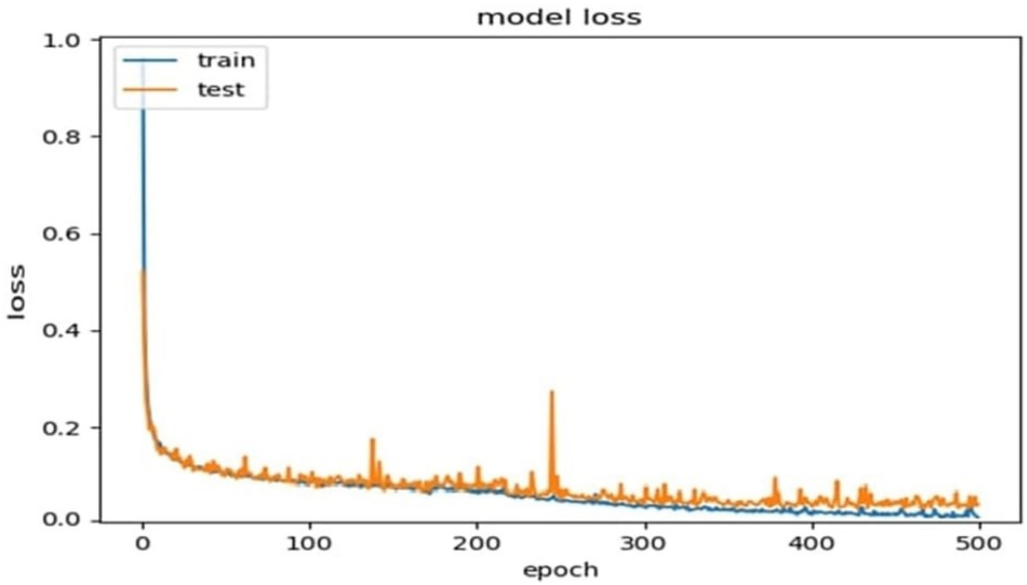


Figure 4.9: Loss curve of SFIDDNN model.

4.3.3 Results and Experiments of MFD

The MFD is used for diagnosis multi Fault. This classifier consists of three models as indicated in chapter three. Also, this classifier has parameters that affect and control the training and testing process of models used in this thesis. These parameters are the number of epochs, batch size, and learning rate. The results and experiments of each model will be described below.

- **Results MFDDNN**

This model work for binary classification (normal, abnormal), and used the multi faults Dataset, which consisted of (292169) samples were 70 % for training and 30 % for testing. The summary representation of the MFDDNN model architecture is shown in table (4.16).

Table 4.16: The summary of MFDDNN model.

Layer (type)	Output	Parameters number
conv1d–1	(None,38,16)	64
Leaky–relu–1	(None,38,16)	0
Max–pooling1d–1	(None,19,16)	0
Leaky–relu–2	(None,19,16)	0
conv1d_2	(None,17,32)	1568
Max–pooling1d–2	(None,8,32)	0
conv1d–3	(None 6,64)	6208
Leaky–relu–3	(None,6,64)	0
Max–pooling1d–3	(None,6,64)	0
conv1d–4	(None,4,128)	24704
Leaky–relu–4	(None,4,128)	0
Max–pooling1d–4	(None,4,128)	0
conv1d–5	(None,2,256)	98560
Leaky–relu–5	(None,2,256)	0
Max–pooling1d–5	(None,2,256)	0
conv1d–6	(None,2,512)	131584
Leaky–relu–6	(None,2,512)	0
Max–pooling1d–6	(None,2,512)	0
Leaky–relu–7	(None,2,512)	0
conv1d–7	(None,2,1024)	525312

conv1d–8	(None,2,35)	35875
Flatten–1	(None,70)	0
Dense–1	(None,2)	142
Total–params: 824,017 **Trainable–params: 824,017** **Non-trainable–params: 0**		

The experimental results of this model for adjusting the values of parameters during training and testing are shown in the table (4.17) which explains the result of using different batch sizes, table (4.18) which explains the result of using different epochs, and table (4.19) which explains result for different learning rate.

Table 4.17: Results MFDDNN model of 50 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.9707	0.1061	0.9814	0.0474
256	0.9726	0.0860	0.9724	0.0928
512	0.9756	0.0697	0.9806	0.0548
1024	0.9842	0.0390	0.9835	0.0415

Table 4.18: Results MFDDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9882	0.0274	0.9879	0.0350
150	0.9903	0.0225	0.9871	0.0342
200	0.9904	0.0224	0.9900	0.0276
300	0.9917	0.0183	0.9910	0.0270

Table 4.19: Results MFDDNN model of 50 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9841	0.0387	0.9843	0.0407
0.0001	0.9788	0.0561	0.9809	0.0551
0.0016	0.9839	0.0396	0.9851	0.0376
0.0004	0.7917	0.4517	0.8056	0.4366

MFDDNN model final parameters values for training is batch size= 1024, epochs=300, and learning rate=0.0016. Results get from them for training are accuracy: 0.9917, loss: 0.0183, and for testing is accuracy: 0.9910, loss: 0.0270. Prediction time for all testing dataset is 6737.812280654907 millisecond, while training time for each epoch 4 second. Figure (4.10) shows the curve of accuracy, figure (4.11) shows the curve of the loss function.

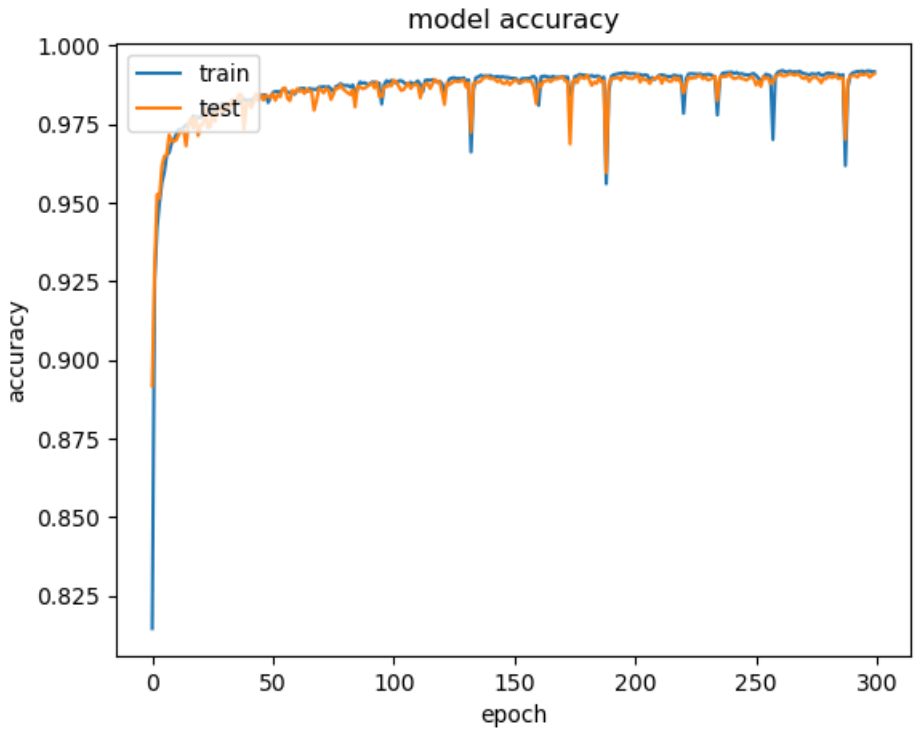


Figure 4.10: MFDDNN accuracy curve.

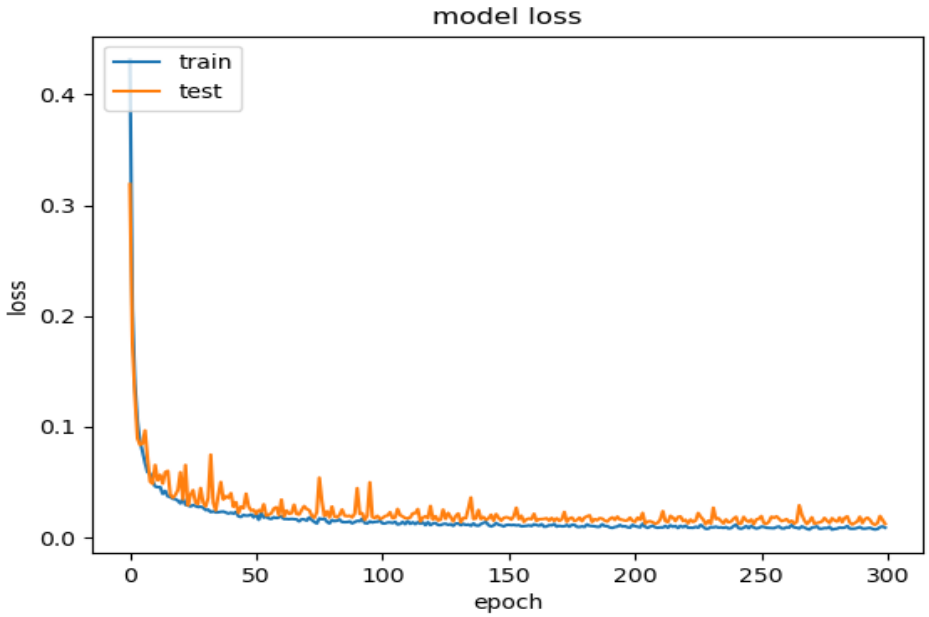


Figure 4.11: MFDDNN loos curve.

- **Results MFIDNN**

This model works based on the results of the previous model. The following tables show the results of experiments to determine the location of the fault for the purpose of isolation.

The summary of the MFIDNN model is shown in table (4.20).Table (4.21) shows the batch size adjustment process by comparing the accuracy values and loss function values during training and testing when the batch size changes with the value of the constant learning rate and the number of periods. Table (4.22) shows the difference between the results achieved for accuracy and the values of the loss function for the different epochs. Table (4.23) shows the difference between results achieved values of accuracy and the loss function for different learning rates.

Table 4.20**:** The summary of MFIDNN model.

Layer (type)	Output	Parameters number
conv1d–1	(None,38,16)	64
Leaky–relu–1	(None,38,16)	0
Max–pooling1d–1	(None,19,16)	0
Leaky–relu–2	(None,19,16)	0

conv1d–2	(None,17,32)	1568
Max–pooling1d–2	(None,8,32)	0
conv1d–3	(None,6,64)	6208
Leaky–relu–3	(None,6,64)	0
Max–pooling1d–3	(None,6,64)	0
conv1d–4	(None,4,128)	24704
Leaky–relu–4	(None,4,128)	0
Max–pooling1d–4	(None,4,128)	0
conv1d–5	(None,2,256)	98560
Leaky–relu–5	(None,2,256)	0
Ma—pooling1d–5	(None,2,256)	0
conv1d–6	(None,2,512)	131584
Leaky–relu–6	(None,2,512)	0
Max–pooling1d–6	(None,2,512)	0
Leaky–relu–7	(None,2,512)	0
conv1d–7	(None,2,1024)	525312
conv1d–8	(None,2,850)	871250
Flatten–1	(None,1700)	0
Dense_1	(None,165)	280665
Total–params: 1,939,915 **Trainable–params: 1,939,915** **Non-trainable–params: 0**		

Table 4.21: Result MFIDNN model of 50 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.9625	0.1550	0.9661	0.1753
256	0.9598	0.1774	0.9682	0.1296
512	0.9471	0.2958	0.9508	0.2640
1024	0.9657	0.1009	0.9688	0.1180

Table 4.22: Result MFIDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9843	0.0456	0.9803	0.0792

150	0.9856	0.0370	0.9862	0.1128
200	0.9863	0.0357	0.9855	0.0848
300	0.9910	0.0361	0.9903	0.0421

Table 4.23: Result MFIDNN model of 50 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9784	0.0631	0.9808	0.0683
0.0001	0.9697	0.0944	0.9726	0.1283
0.0016	0.9825	0.0462	0.9809	0.0794
0.0004	0.9769	0.0647	0.9718	0.1000

MFIDNN model final parameters values for training is Batch size=1024, Epochs=300, Learning rate=0.0016. Results get from them for training are accuracy: 0.9910, loss: 0.0361, and for testing is accuracy: 0.9903, loss 0.0421. Prediction time for all testing dataset is 2620.15438079834 millisecond, while training time for each epoch 3 second. Figure (4.12) shows the curve of accuracy, figure (4.13) shows the curve of the loss function.

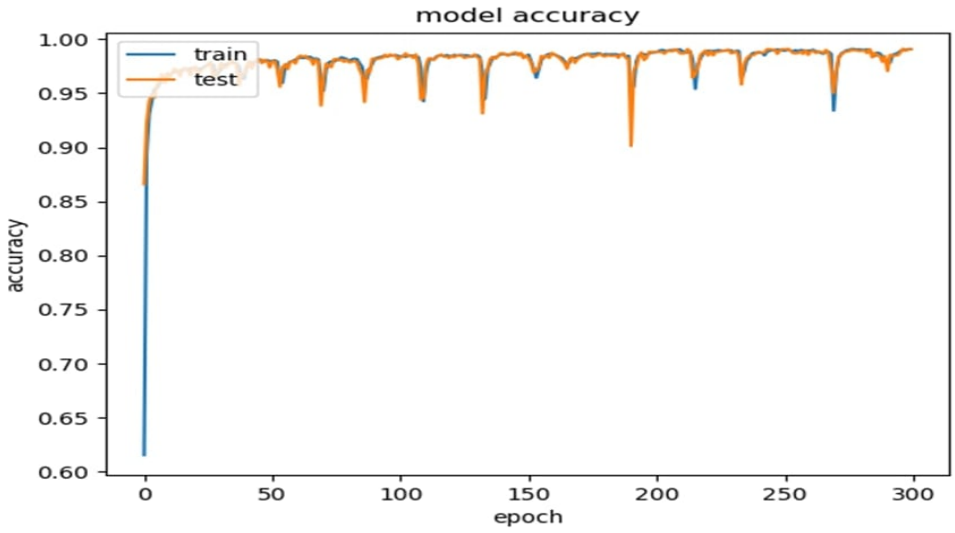


Figure 4.12: curve of accuracy of MFIDNN model.

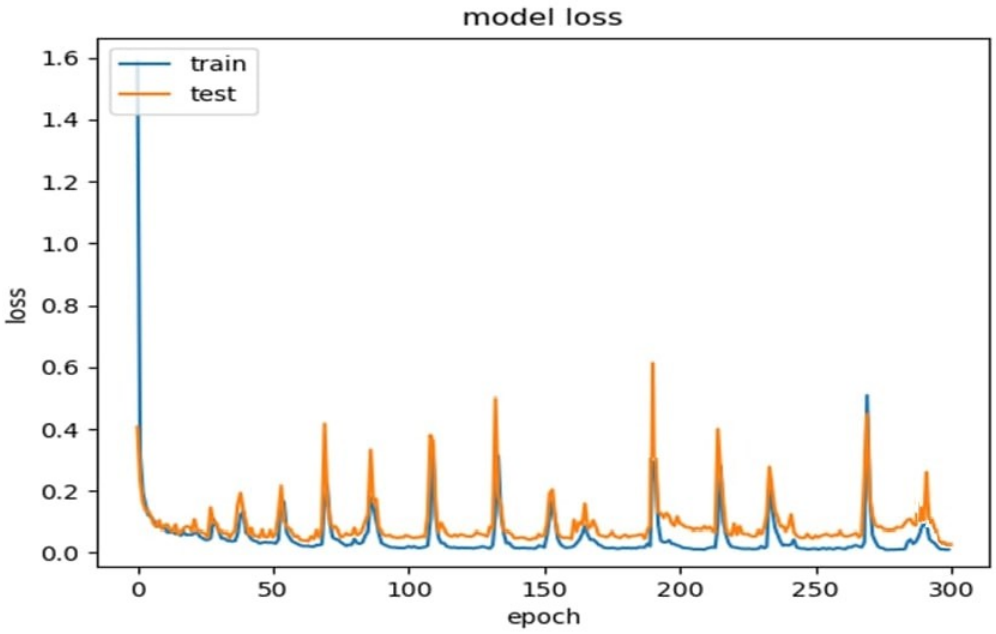


Figure 413. curve of loss for MFIDNN model.

- **Results MFIDDNN**

The working principle of this model is similar to the SFIDDNN that determines the type of fault. It works on the same data as the MFIDNN model. The summary of the MFIDDNN model is shown in table (4.24). Table (4.25) shows the batch size adjustment process by comparing the accuracy and loss function values during training and testing when the batch size changes with the value of the constant learning rate and the number of periods. Table (4.26) shows the difference between the results achieved for values of accuracy and the loss function for the different epochs. Table (4.27) shows the difference between the results achieved for values of accuracy and loss function for the different learning rates.

Table 4.24: The summary of MFIDDNN model.

Layer (type)	Output	Parameters number
conv1d–1	(None,38,16)	64
Leaky–relu–1	(None,38,16)	0
Max–pooling1d–1	(None,19,16)	0
Leaky–relu–2	(None,19,16)	0

conv1d–2	(None,17,32)	1568
Max–pooling1d–2	(None,8,32)	0
conv1d–3	(None,6,64)	6208
Leaky–relu–3	(None,6,64)	0
Max–pooling1d–3	(None,6,64)	0
conv1d–4	(None,4,128)	24704
Leaky–relu–4	(None,4,128)	0
Max–pooling1d–4	(None,4,128)	0
conv1d–5	(None,2,256)	98560
Leaky–relu–5	(None,2,256)	0
Max–pooling1d–5	(None,2,256)	0
conv1d–6	(None,2,512)	131584
Leaky–relu–6	(None,2,512)	0
Max–pooling1d–6	(None, 2,512)	0
Leaky–relu–7	(None, 2,512)	0
conv1d–7	(None,2,1024)	525312
conv1d_8	(None,2,340)	348500
Flatten_1	(None,680)	0
Dense_1	(None,63)	42903
Total–params: 1,179,403 **Trainable–params: 1,179,403** **Non-trainable–params: 0**		

Table 4.25: Result MFIDDNN model of 50 epochs for different batch size.

Batch size	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
128	0.9675	0.1140	0.9635	0.1465
256	0.9689	0.1229	0.9672	0.1606
512	0.9784	0.0677	0.9799	0.0911
1024	0.9818	0.0490	0.9809	0.0686

Table 4.26: Result MFIDDNN model of 1024 batch size for different epochs.

Epochs	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
100	0.9847	0.0425	0.9821	0.0725
200	0.9880	0.0310	0.9854	0.0795

300	0.9812	0.0915	0.9796	0.1263
500	0.9916	0.0230	0.9907	0.0855

Table 4.27: Result MFIDDNN model of 50 epochs for different learning rate.

Learning rate	Train-Accuracy	Train-Loss function	Test-Accuracy	Test-Loss function
0.001	0.9815	0.0534	0.9753	0.0852
0.0001	0.9583	0.1374	0.9538	0.1447
0.0016	0.9818	0.0490	0.9809	0.0686
0.0004	0.9777	0.0642	0.9775	0.0774

MFIDDNN model final parameters values for training is batch size= 256, epochs=500, and learning rate=0.0016. Results get from them for training are accuracy: 0.9912, loss: 0.0232, and for testing is accuracy: 0.9909, loss: 0.1075. Prediction time for all testing dataset is 2447.04270362854 millisecond, while training time for each epoch 2 second. Figure (4.14) shows the curve of accuracy, figure (4.15) shows curve of loss function.

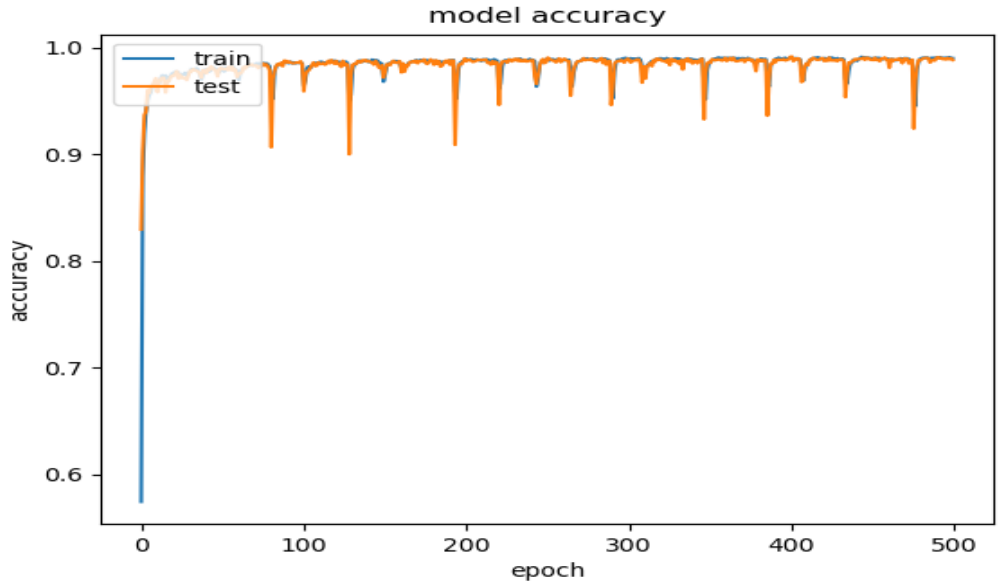


Figure4.14: curve of accuracy of MFIDDNN model.

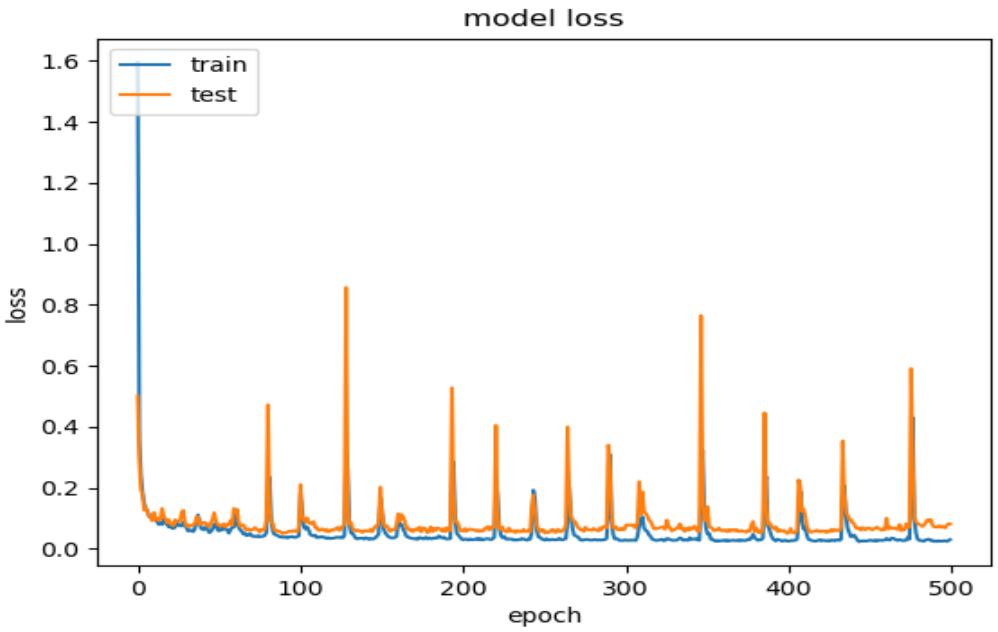


Figure 4.15 curve of loss for MFIDDNN model.

4.4 Evaluation Overall System

Chapter three explained the overall system consists of two classifiers, each of them consisting of three models. Evaluating the overall FDS performance requires evaluating the performance of each model in the system on the testing dataset. The metrics used in the evaluation are accuracy, precision, recall, and F1-Mesure. The overall performance of the system is evaluated in two classifiers: evaluation of the SFD performance on the single dataset, and evaluation of the MFD performance on the multi dataset.

4.4.1 Evaluation the SFD Performance

The results of the evaluation of the SFD are presented through the terms of the confusion matrix for SFDDNN, samples of classes' label of SFIDNN, and samples of classes' label of SFIDDNN shown in table (4.28), and performance metrics are shown in table (4.29).

The confusion matrix of SFIDDNN is shown in figure (4.16) which explains how to calculate (TP, FP, and FN) according to equations (2.30, 2.31, and 2.32)

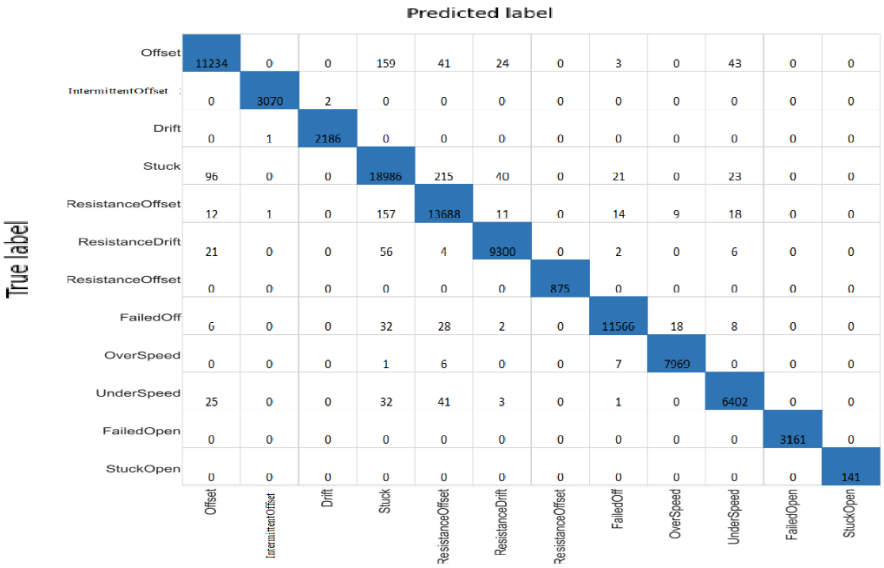


Figure 4.16 confiusion matrix of SFIDDNN model.

Table 4.28: the confusion matrix of SFDC.

model	Number of samples	Classes	TP	FP	FN
SFDDNN	163280	Normal	86926	2873	2503
		Abnormal	70978	2503	2873
SFIDNN	89767	IT240	16905	159	233
		IT281	813	0	0
		ST516	152	6	3
		E242	307	0	0
		IT267	306	0	0
		TE228	17762	197	164
		ESH244A	2970	85	37
		E240	1220	0	0

		E281	2333	73	77
		ISH236	2226	152	85
SFIDDNN	89767	Offset	11234	270	160
		IntermittentOffset	3070	3072	2
		Drift	2186	1	2
		Stuck	18986	395	437
		ResistanceOffset	13688	222	335
		ResistanceDrift	9300	89	80
		IntermittentResistanceOffset	875	0	0
		FailedOff	11566	94	48
		OverSpeed	7969	14	27
		UnderSpeed	6402	102	98
		FailedOpen	3161	0	0
		StuckOpen	141	0	0

Table 4.29: The performance matrix of SFDC.

Model	Number of samples	Accuracy	precision	recall	F1-measure
SFDDNN	163280	0.96707	0.96710	0.96708	0.96708
SFIDNN	89767	0.98513	0.985127	0.98513	0.98512
SFIDDNN	89767	0.98675	0.98677	0.98675	0.98676

4.4.2 Evaluation MFD Performance

The results of the evaluation of the MFD are presented through the terms of the confusion matrix for MFDDNN, the sample of class's label of MFIDNN, and a sample of class's label of MFIDDNN shown in table (4.30) and performance metrics are shown in Table (4.31). In addition, to comparing the proposed system to with M Ganesan et al [15] as shown in table (4.32).

Table 4.30: The confusion matrix of MFDC.

Model	Number of samples	Classes	TP	FP	FN
MFDDNN	87651	Normal	43325	417	370
		Abnormal	43539	370	417
MFIDNN	43719	TE510	96	0	0
		IT281	157	0	0
		IT281TE505	99	0	0
		IT281TE505TE501	1322	0	0
		TE500	32	0	2
		TE511	198	1	0
		TE511ST516	404	0	0
		TE501	455	0	2
		TE501TE500	360	1	0
		IT240	477	1	0
MFIDDNN	43719	Offset	1375	0	0
		OffsetOffset	39	0	1
		OffsetOffsetStuck	188	1	0
		Stuck	4662	3	12

		StuckStuck	1046	3	1
		AbruptParasiticLoad	563	2	3
		AbruptParasiticLoadFailedOff	321	0	0
		AbruptParasiticLoadFailedOffFailed Open	343	1	0
		StuckOverSpeed	234	13	5
		StuckFailedOffUnderSpeed	338	5	13

Table 4.31: The performance matrix of MFDC.

Model	Number of samples	Accuracy	precision	recall	F1-measure
MFDDNN	87651	0.9910	0.9920	0.9915	0.9917
MFIDNN	43719	0.9902	0.9903	0.9902	0.9899
MFIDDNN	43719	0.9914	0.9916	0.9914	0.9913

Table 4.32 show compare between proposed approache with other method.

	M Ganesan et al [15]		**Proposed***	
Dataset	Univariate		Multivariate	
	ADAPT single fault dataset (six experiments) (120 samples for each experiment)		ADAPT single,multi-fault dataset (227 experiments for single,2400 samples for each experiment) (122 experiments for multi,2400 samples for each experiment)	
Pre-processing	Stockwelltransform with remove missing values.		Impute missing values and StanderScaler	
Approach	1DCNN for detection		1DCNN for diagnosis(detection , isolation, Identify)	
layers	Convolution, pooling and ReLU		Convolution, maxpooling , ReLU/Leaky Relu, and linear	
	Learning rate	0.001	Learning rate	0.0016

Hyperparameter						
	Batch Size		16	Batch Size		1024
	Number of Filters		50	Number of Filters		(16-1024)
	Strides		1	Strides		1
	Drop out		0.2			
Accuracy and Loss		Accuracy	Loss	Single	Accuracy	Loss
	Train	96.6	0.114	Train	96.77	0.1022
	Test	96.7	0.118	Test	96.71	0.1248
Not*	When remove missing value in our model **SFDDNN** accuracy= but **SFIDNN,SFIDDNN** suffer from overfitting			Train	97.68	0.4221
				Test	97.77	0.4170

Chapter Five

Conclusion and Future Works

Chapter Five

Conclusion and Future Works

5.1 Research Conclusions

Fault diagnosis is an urgent issue in most of applications in real world and in the last few years, fault diagnosis in EPS of UAV has become a significant topic to obtain safety. Preprocessing step is an important to get the desired result from the system. In this thesis several conclusions achieved during the design and implementation of the proposed system, where the proposed system proved its success by adopting one of the famous deep learning techniques known as a CNN as mention in table (4.29) and table (4.31).

The method implemented in preprocessing is: using the mean method for imputing missing values gives better results than the interpolation method, and normalization of the dataset using standardization gives better results than mini-max methods due to the nature of the dataset as mention in table (4.1) and table (4.2). Where using 1DCNN gives better performance than the statistical methods and other machine learning methods in fault diagnosis in EPS of UAV. In CNN layers, adding the Max-pooling after some convolutional layers in SFC and MFD to reduce dimensionality and extract the best feature from the previous layer. ReLU activation function is used in SFD because of the nature of its data to extract salient features and neglect weak features. While The Leaky ReLU activation function is used in MFD because its data contains more negative values to preserve a tiny scope for negative values in order to extract salient features and neglect weak features, and adding linear activation function in last convolution layer to aggregate close features from each other.

5.2 Future Works

1. Reducing the training time of the networks and obtain higher accuracy with fewer epochs.
2. Applying features reduction and optimization to reduce the number of parameters of the networks in the system models and preserve the required memory.
3. Applying the proposed model to UAV online in cooperation with an engineering team, So that the proposed system receives data directly from the electrical circuits system, and therefore the monitoring and handling of errors in a manner commensurate with the technological development.
4. Making the proposed system perform an alarm to the operator when a fault occurs.
5. Applying the proposed model to determine which action must be done based on the severity of fault mode, which is classified by the identifying model.

References

References

[1] K. Keller, A. Del Amo, and B. Jordan, “Aircraft electrical power systems prognostics and health management (AEPHM),” *SAE Tech. Pap.*, no. 724, 2004, doi: 10.4271/2004-01-3162.

[2] D. Miljković, “Fault detection methods: A literature survey,” *MIPRO 2011 - 34th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. - Proc.*, pp. 750–755, 2011.

[3] R. Telford and S. Galloway, “Fault classification and diagnostic system for unmanned aerial vehicle electrical networks based on hidden Markov models,” *IET Electr. Syst. Transp.*, vol. 5, no. 3, pp. 103–111, 2015, doi: 10.1049/iet-est.2014.0042.

[4] M. Jamil, S. K. Sharma, and R. Singh, “Fault detection and classification in electrical power transmission system using artificial neural network,” *Springerplus*, vol. 4, no. 1, 2015, doi: 10.1186/s40064-015-1080-x.

[5] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, and S. Uckun, “Diagnosing faults in electrical power systems of spacecraft and aircraft,” *Proc. Natl. Conf. Artif. Intell.*, vol. 3, no. 650, pp. 1699–1705, 2008.

[6] D. Guo, Y. Wang, M. Zhong, and Y. Zhao, “Fault detection and isolation for Unmanned Aerial Vehicle sensors by using extended PMI filter,” *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 818–823, 2018, doi: 10.1016/j.ifacol.2018.09.669.

[7] C. F. Silvio Simani and and R. J. Patton, *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. .

[8] G. K. Fourlas and G. C. Karras, “A survey on fault diagnosis and fault-

tolerant control methods for unmanned aerial vehicles †," *Machines*, vol. 9, no. 9, 2021, doi: 10.3390/machines9090197.

[9] D. Li, Y. Wang, J. Wang, C. Wang, and Y. Duan, "Sensors and Actuators A : Physical Recent advances in sensor fault diagnosis : A review," *Sensors Actuators A. Phys.*, vol. 309, p. 111990, 2020, doi: 10.1016/j.sna.2020.111990.

[10] J. L. Yang, Y. S. Chen, L. L. Zhang, and Z. Sun, "Fault detection, isolation, and diagnosis of self-validating multifunctional sensors," *Rev. Sci. Instrum.*, vol. 87, no. 6, 2016, doi: 10.1063/1.4954184.

[11] J. Fu, C. Sun, Z. Yu, and L. Liu, "A hybrid CNN-LSTM model based actuator fault diagnosis for six-rotor UAVs," *2019 Chinese Control Decis. Conf.*, pp. 410–414, 2019.

[12] A. Muhammad, J. M. Lee, S. W. Hong, S. J. Lee, and E. H. Lee, "Deep Learning Application in Power System With Case Study On Power Solar Irradiation Forecasting," *2019 Int. Conf. Artif. Intell. Inf. Commun.*, p. https://ieeexplore.ieee.org/abstract/document/8668, 2019.

[13] J. Ren, R. Ren, M. Green, and X. Huang, "A Deep Learning Method for Multiple Faults Detection and Classification of Unmanned Ground Vehicles," *12th Int. Conf. Comput. Sci. Inf. Technol. CSIT 2019*, pp. 108–111, 2019, doi: 10.1109/CSITechnol.2019.8895161.

[14] L. Chen, Z. Zhang, and J. Cao, "A novel method of combining generalized frequency response function and convolutional neural network for complex system fault diagnosis," *PLoS One*, vol. 15, no. 2, pp. 1–17, 2020, doi: 10.1371/journal.pone.0228324.

[15] M. Ganesan, R. Lavanya, and M. Nirmala Devi, "Fault detection in satellite power system using convolutional neural network," *Telecommun. Syst.*, vol.

76, no. 4, pp. 505–511, 2021, doi: 10.1007/s11235-020-00722-5.

[16] A. Alos and Z. Dahrouj, “Decision tree matrix algorithm for detecting contextual faults in unmanned aerial vehicles,” *J. Intell. Fuzzy Syst.*, vol. 38, no. 4, pp. 4929–4939, 2020, doi: 10.3233/JIFS-191575.

[17] J. Fu, C. Sun, Z. Yu, and L. Liu, “A hybrid CNN-LSTM model based actuator fault diagnosis for six-rotor UAVs,” *Proc. 31st Chinese Control Decis. Conf. CCDC 2019*, pp. 410–414, 2019, doi: 10.1109/CCDC.2019.8832706.

[18] ICAO, *Unmanned Aircraft Systems (UAS)*. USA: Approved by the Secretary General and published under his authority, 2011.

[19] E. G. Fernández, “MASTER THESIS Management System for Unmanned Aircraft System Teams,” *Technology*, no. May. 2010.

[20] W. L. Perry and J. Moffat, *Applications for NAVY UNMANNED AIRCRAFT SYSTEMS*. 2004.

[21] Civil Aviation Authority, “Unmanned Aircraft System Operations in UK Airspace – Guidance,” *Cap 722*, no. 8, p. 238, 2020, [Online]. Available: www.caa.co.uk/CAP722.

[22] S. G. Gupta, M. Ghonge, and P. M. Jawandhiya, “Review of Unmanned Aircraft System (UAS),” *SSRN Electron. J.*, no. April, 2019, doi: 10.2139/ssrn.3451039.

[23] S. W. Blume, H. Lane, T. Chen, T. G. Croda, S. Welch, and A. Editor, “Electric Power System Basics for the Nonelectrical Professional (IEEE Press Series on Power Engineering),” p. 260, 2007, [Online]. Available: http://www.amazon.com/Electric-System-Nonelectrical-Professional-Engineering/dp/0470129875.

[24] A. Abbaspour, S. Mokhtari, A. Sargolzaei, and K. K. Yen, “A survey on active fault-tolerant control systems,” *Electron.*, vol. 9, no. 9, pp. 1–23, 2020, doi: 10.3390/electronics9091513.

[25] M. John and M. Nas, “Classifying Unmanned Aircraft Systems : Developing a Legal Framework for the Purposes of Airworthiness Certification,” 2015.

[26] I. Track and D. P. Descriptions, “Second International Diagnostic Competition (DXC ’ 10) Industrial Track Diagnostic Problem Descriptions,” *Time*.

[27] H. Hamadi, “Fault-tolerant control of a multirotor unmanned aerial vehicle under hardware and software failures,” 2021, [Online]. Available: https://tel.archives-ouvertes.fr/tel-03116827.

[28] J. P. N. Gonzalez, L. E. G. Castañon, R. Morales-Menendez, A. El Hajjaji, and A. Rabhi, “Fault diagnosis of electrical power systems using soft computing,” *IFAC Proc. Vol.*, pp. 863–868, 2013, doi: 10.3182/20130204-3-FR-2033.00037.

[29] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Min. Knowl. Discov.*, vol. 31, no. 3, pp. 606–660, 2017, doi: 10.1007/s10618-016-0483-9.

[30] NASA, “Satellite,” 2012. https://c3.ndc.nasa.gov/dashlink/topic/diagnostic-challenge-competition/%5D%28https%3A/dashlink.arc.nasa.gov/topic/diagnostic-challenge-competition/%29/.

[31] NASA, “FLTz dataset,” 2011. https://c3.ndc.nasa.gov/dashlink/resources/294/.

[32] NASA, “ADAPT EPS of UAV dataset,” 2020. https://data.nasa.gov/dataset/ADAPT-Dataset/6gjh-n6gb.

[33] X. Guo, L. Chen, and C. Shen, “Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis,” *Meas. J. Int. Meas. Confed.*, vol. 93, pp. 490–502, 2016, doi: 10.1016/j.measurement.2016.07.054.

[34] M. Grundland and N. A. Dodgson, “The decolorize algorithm for contrast enhancing, color to grayscale conversion,” *Computer (Long. Beach. Calif).*, no. 649, 2005.

[35] L. Deng, “An overview of deep-structured learning for information processing,” *APSIPA ASC 2011 - Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. 2011*, pp. 138–151, 2011.

[36] H. Wang, R. Czerminski, and A. C. Jamieson, “Neural Networks and Deep Learning,” *Mach. Age Cust. Insight*, pp. 91–101, 2015, doi: 10.1108/978-1-83909-694-520211010.

[37] R. Agrawal, C. Faloutsos, and A. Swami, “E cient Similarity Search In Sequence Databases 1 Introduction,” *Syst. Res.*, vol. 8958546, pp. 1–16, 1994.

[38] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep Models under the GAN: Information leakage from collaborative deep learning,” 2017, doi: 10.1145/3133956.3134012.

[39] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[40] S. Theodoridis, “Neural Networks and Deep Learning,” in *Machine Learning*, 2015.

[41] R. E. Neapolitan and X. Jiang, “Neural Networks and Deep Learning,” in *Artificial Intelligence*, 2018, pp. 389–411.

[42] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1237, no. 2, 2019, doi: 10.1088/1742-6596/1237/2/022030.

[43] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, “Deeper networks for pavement crack detection,” *ISARC 2017 - Proc. 34th Int. Symp. Autom. Robot. Constr.*, no. July, pp. 479–485, 2017, doi: 10.22260/isarc2017/0066.

[44] S. Nasiri, J. Helsper, M. Jung, and M. Fathi, “DePicT Melanoma Deep-CLASS: A deep convolutional neural networks approach to classify skin lesion images,” *BMC Bioinformatics*, vol. 21, no. Suppl 2, pp. 1–13, 2020, doi: 10.1186/s12859-020-3351-y.

[45] M. J. Brown, L. A. Hutchinson, M. J. Rainbow, K. J. Deluzio, and A. R. De Asha, “A comparison of self-selected walking speeds and walking speed variability when data are collected during repeated discrete trials and during continuous walking,” *J. Appl. Biomech.*, vol. 33, no. 5, pp. 384–387, 2017, doi: 10.1123/jab.2016-0355.

[46] B. Gao and L. Pavel, “On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning,” pp. 1–10, 2017, [Online]. Available: http://arxiv.org/abs/1704.00805.

[47] “Re-Coloring Images for Gamuts of,” 2005.

[48] S. Ruder, “An overview of gradient descent optimization algorithms,” pp. 1–14, 2016, [Online]. Available: http://arxiv.org/abs/1609.04747.

[49] C. C. Aggarwal and Neural, *Neural Networks and Deep Learning*. 2018.

[50] K. Janocha and W. M. Czarnecki, “On loss functions for deep neural

networks in classification," *Schedae Informaticae*, vol. 25, pp. 49–59, 2016, doi: 10.4467/20838476SI.16.004.6185.

[51] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[52] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam Optimization Algorithm for Wide and Deep Neural Network," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, p. 41, 2019, doi: 10.17977/um018v2i12019p41-46.

[53] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Comput. Intell. Neurosci.*, vol. 2018, 2018, doi: 10.1155/2018/7068349.

[54] P. Khan *et al.*, "Machine Learning and Deep Learning Approaches for Brain Disease Diagnosis: Principles and Recent Advances," *IEEE Access*, vol. 9, no. March, pp. 37622–37655, 2021, doi: 10.1109/ACCESS.2021.3062484.

[55] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, 2020, doi: 10.1109/TVT.2020.3034800.

[56] E. Alpaydin, "Neural Networks and Deep Learning," *Mach. Learn.*, 2021, doi: 10.7551/mitpress/13811.003.0007.

[57] Dr. Juliansyah Noor, *Encyclopedia of Computational Biology Bioinformatics and*, vol. 53, no. 9. 2019.

[58] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "NNs Archtectures review," *Elsevier*, pp. 1–31, 2017, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216315533.

[59] S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, 2013, doi: 10.1109/TPAMI.2012.59.

[60] T. Liu, H. Xu, M. Ragulskis, M. Cao, and W. Ostachowicz, “A data-driven damage identification framework based on transmissibility function datasets and one-dimensional convolutional neural networks: Verification on a structural health monitoring benchmark structure,” *Sensors (Switzerland)*, vol. 20, no. 4, pp. 1–25, 2020, doi: 10.3390/s20041059.

[61] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” *FPGA 2015 - 2015 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, pp. 161–170, 2015, doi: 10.1145/2684746.2689060.

[62] O. Avci, O. Abdeljaber, M. S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman, “Efficiency validation of one dimensional convolutional neural networks for structural damage detection using a SHM benchmark data,” *25th Int. Congr. Sound Vib. 2018, ICSV 2018 Hiroshima Call.*, vol. 8, no. July, pp. 4600–4607, 2018.

[63] O. Abdeljaber, O. Avci, M. S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman, “1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data,” *Neurocomputing*, vol. 275, pp. 1308–1317, 2018, doi: 10.1016/j.neucom.2017.09.069.

[64] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, “Convolutional neural networks for time series classification,” *J. Syst. Eng. Electron.*, vol. 28, no. 1, pp. 162–169, 2017, doi: 10.21629/JSEE.2017.01.18.

[65] “Handbook_of_Medical_Image_Computing_and_Computer_Assisted_Int

ervention.pdf.” .

[66] S. Kevin Zhou, G. Fichtinger, and D. Rueckert, “Handbook of medical image computing and computer assisted intervention,” *Handb. Med. Image Comput. Comput. Assist. Interv.*, pp. 1–1043, 2019, doi: 10.1016/C2017-0-04608-6.

[67] W. Lee, A. T. M. Lenferink, C. Otto, and H. L. Offerhaus, “Classifying Raman spectra of extracellular vesicles based on convolutional neural networks for prostate cancer detection,” *J. Raman Spectrosc.*, vol. 51, no. 2, pp. 293–300, 2020, doi: 10.1002/jrs.5770.

[68] M. H. Mozaffari and L.-L. Tay, “A Review of 1D Convolutional Neural Networks toward Unknown Substance Identification in Portable Raman Spectrometer,” 2020, [Online]. Available: http://arxiv.org/abs/2006.10575.

[69] H. Chen *et al.*, “A deep learning CNN architecture applied in smart near-infrared analysis of water pollution for agricultural irrigation resources,” *Agric. Water Manag.*, vol. 240, no. May, p. 106303, 2020, doi: 10.1016/j.agwat.2020.106303.

[70] D. Huang, S. Li, N. Qin, and Y. Zhang, “Fault Diagnosis of High-Speed Train Bogie Based on the Improved-CEEMDAN and 1-D CNN Algorithms,” *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, doi: 10.1109/TIM.2020.3047922.

[71] C. C. Chen, Z. Liu, G. Yang, C. C. Wu, and Q. Ye, “An improved fault diagnosis using 1d-convolutional neural network model,” *Electron.*, vol. 10, no. 1, pp. 1–19, 2021, doi: 10.3390/electronics10010059.

[72] C. C. Aggarwal and Data, *Data Mining*. 2021.

[73] S. Moritz and T. Bartz-Beielstein, “imputeTS: Time series missing value

imputation in R," *R J.*, vol. 9, no. 1, pp. 207–218, 2017, doi: 10.32614/rj-2017-009.

[74] M. N. Noor, A. S. Yahaya, N. A. Ramli, and A. M. M. Al Bakri, "Filling missing data using interpolation methods: Study on the effect of fitting distribution," *Key Eng. Mater.*, vol. 594–595, no. May, pp. 889–895, 2014, doi: 10.4028/www.scientific.net/KEM.594-595.889.

[75] W. L. Junger and A. Ponce de Leon, "Imputation of missing data in time series for air pollutants," *Atmos. Environ.*, vol. 102, pp. 96–104, 2015, doi: 10.1016/j.atmosenv.2014.11.049.

[76] S. G. K. Patro and K. K. sahu, "Normalization: A Preprocessing Stage," *Iarjset*, pp. 20–22, 2015, doi: 10.17148/iarjset.2015.2305.

[77] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, vol. 72. 2015.

[78] S.A. Alasadi and W. S. Bhaya, "Review of Data Preprocessing Techniques in Data Mining," *J. Eng. Appl. Sci.*, vol. 12, no. 16, pp. 4102–4107, 2017.

[79] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, 2017, doi: 10.5120/ijca2017915495.

[80] T. Sarheed Alshammari, "Biologically-Inspired Machine Intelligence Technique for Activity Classification in Smart Home Environments Evaluating Machine Learning Techniques for Activity Classifica-tion in Smart Home Environments," no. August 2019, 2019.

[81] C. Manliguez, "Generalized Confusion Matrix for Multiple Classes Generalized Confusion Matrix for Multiple Classes The total numbers of false negative (TFN), false positive (TFP), and true negative (TTN) for

each class i will be calculated based on the Generalized," no. November, pp. 2–4, 2016, doi: 10.13140/RG.2.2.31150.51523.

[82] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," pp. 1–17, 2020, [Online]. Available: http://arxiv.org/abs/2008.05756.

Appendix A

The Published Paper

Using 1DCNN for Diagnosis Fault in Electricals Power System

Noor Hassan

Software, College of Information Technology, University of Babylon, Babylon, Iraq.

Mohammed Dosh*

Computer Science, College of Education for Girls, University of Kufa, Al-Najaf, Iraq.

Abstract

A fault is a non-desirable deviation of a system or one of its components from normal behaviour. Fault diagnosis is the process of detecting, isolating, and identifying such a fault.

In this paper, a method to diagnose the fault in Unmanned Aircraft Vehicles (UAVs) systems is proposed. These systems consist of several components along with sensors to measure some properties such as current, voltages, and temperatures. In such a system, faults are unavoidable and could lead to the failure of the whole system without timely treatment, and cause human losses. The proposed method adopts a deep learning technique to build a model called diagnose. Convolutional Neural Network (CNN) made.

This thesis adopted the "ADAPT" dataset, acquired from NASA Research Center will be considered. for a case study to represent an electrical power system of an Unmanned Aircraft Vehicle (UAV). Which contains 227 scenarios for a single fault. Finally, the classification accuracy for the proposed system for single fault classifier achieved for detection 96.70%, for isolation 98.51%, and for identifying 98.67%.

Keywords: Deep Learning, One Dimensional Convolutional Neural Network (1DCNN), Fault Diagnosis, unmanned aircraft vehicle (UAV).

.

1. Introduction

Recently, Unmanned Aircraft Vehicles (UAVs) have been used in several fields, such as load transport, archaeology, hobbies, defences, filmmaking, and recreational. The UAVs are aircraft characterized by the absence of a human pilot on board [1]. Power systems are quickly developing in the latest years because of the implication of the latest digital technologies [2]. The automatic control of such systems is considered an essential part. This may be suffered by problems as a result of a fault in it or one of its components, undetected failures can lead to critical damage. To eliminate problems in the system Diagnosis Algorithms (DA) are used. The DA uses observations coming from the system being monitored to determine whether a fault has occurred (fault detection) and which fault has occurred (fault classification). Thus, fault diagnosis plays a substantial role in ensuring system safety and reliability [1].

An electrical power system has several components and sensors. Faults in the power system of UAV may occur in both components and sensors. This brings more difficulty and uncertainty to the problem of fault diagnosis. Assuming that there exists a set fault types F1,…Fn, each of which has a different number of fault modes. Then, we say that a fault of type Fi occurs if a fault of a certain mode in that typically occurs. In the first place, we will concentrate on the diagnosis of a single fault.

Faults are very expensive for every industry because they impose a repair of the machine, and stop periods in the entire process, with consequent loss of and delay in mission. To overcome these difficulties, companies resort to predictive repair. Fault detection and isolation play a significant role in ensuring system reliability and safety. To achieve precise control and navigation of the UAV, the reliability of the sensor system must be ensured [3]. The use of fault diagnosis (FD) techniques is becoming increasingly important to ensure high levels of safety and reliability in automated systems and autonomous or remotely controlled systems. The main purpose of the FD algorithm is to monitor the system during its operation to detect the occurrence of faults (fault detection), locate faults (fault isolation), and determine their temporal evolution (fault identification) [1][4]. We use one-dimensional convolutional neural network for Diagnosis.

1. Related Works

Due to the development in industry and the requirements to increase the safety and correctness of EPS, the fault diagnosis algorithm has become the focus of most studies and research. Therefore a review of the most relevant work is provided.

1. Jing-li .yet al (2016)[5], have shown that self-validating multifunctional sensors have been used in most industries. They used gas data sample implemented by board PCI-6014, which contained four type of faults. So the method was used to detect, isolate and diagnose their faults complete ensemble empirical mode decomposition with sample entropy weighted energy and multiclass SVM.

2. Jiangmeng. Fu et al(2019).[6], used Deep-learning approach to diagnose faults in six-rotor UAVs based on hybrid CNN-LSTM. They used four experiment of flight UAV data. The proposed method used sliding window technique and then extracts features using the networks for diagnose actuator faults. Accuracy calculated for each experiment then average accuracy calculated CNN-LSTM model. An inadequate in this work they used few datasets.

3. Aslam. M et al.(2019) [7], overviewed an application in power system, they used deep LSTM to present novel case study on the solar irradiation forecasting, single input, two hidden layers, and one output layer used in this work .KMA 'Korea

KALAHARI JOURNALS

INTERNATIONAL JOURNAL OF MECHANICAL ENGINEERING

ISSN: 0974-5823

ACCEPTANCE LETTER

Date: 07-01-2022

PAPER TITLE: Using 1DCNN for Diagnosis Fault in Electricals Power System

PAPER ID: KLHJ5823/12-030124

AUTHOR'S NAME: **Noor Hassan, Mohammed Dosh***

Software, College of Information Technology, University of Babylon, Babylon, Iraq.
Computer Science, College of Education for Girls, University of Kufa, Al-Najaf, Iraq.

Congratulations!

We are pleased to inform you that, after the peer review, your article entitled "**Using 1DCNN for Diagnosis Fault in Electricals Power System**" has been accepted for publication in **INTERNATIONAL JOURNAL OF MECHANICAL ENGINEERING** ISSN: **0974-5823** in the upcoming issue February, 2022.

Notification of publication of your article will be sent to you once your article is online.

Thanking You

Regards,

Editor

INTERNATIONAL JOURNAL OF MECHANICAL ENGINEERING
ISSN: 0974-5823
KALAHARI JOURNALS

الخلاصة

يُعرَّف الخطأ بأنه انحراف غير مرغوب فيه في النظام أو أحد مكوناته عن السلوك الطبيعي أو المرغوب. لذلك ، في الوقت الحاضر يعد تشخيص الأخطاء من أكثر المشكلات التي تحتاج إلى حل ، خاصة في أنظمة الطاقة الكهربائية **(EPS).** أنظمة الطاقة تطورت بسرعة في السنوات الأخيرة بسبب التقنيات الرقمية الحديثة واستخدامها في أنظمة التحكم الآلي. تعتبر الطائرات بدون طيار **(UAVs)** أحد أنظمة التحكم الآلي. والتي تتميز بغياب طيار بشري على متنها وتم استخدامها في عدة مجالات. في مثل هذه الأنظمة ، يعتبر نظام الطاقة جزءًا مهمًا وأساسيًا. قد يعاني نظام الطاقة من مشاكل نتيجة عيب فيه أو في أحد مكوناتة ،عدم كشف هذه الاعطال قد يؤدي إلى أضرار جسيمة. لذلك ، يتم استخدام نظام تشخيص الأخطاء للتخلص من هذة المشاكل . تشخيص خطاء هو عملية الكشف عن الأخطاء وعزلها وتحديدها ، وهذا هو الهدف من المنهجية المقترحة.

إعتمد النظام المقترح التعلم العميق للشبكة العصبية التلافيفية أحادية البعد **(1DCNN)** لبناء مصنفات لتشخيص الأخطاء المفردة والمتعددة باستخدام مجموعة بيانات **"ADAPT"** قادمة من مركز أبحاث ناسا لنظام طاقة كهربائية لمركبة بدون طيار.

أخيرًا ، تم تحقيق دقة التصنيف للنظام المقترح لمصنف الخطأ المفرد للكشف عن الخطأ (٩٦,٧٠٪)، ولعزل الخطأ (٩٨,٥١٪)، ولتحديد شدة الخطأ(٩٨,٦٧٪). بينما حقق مصنف الأخطاء المتعددة للكشف عن الاخطاء (٩٩,١٠٪) ولعزل الاخطاء (٩٩,٠٣٪) ولتحديد شدة الاخطاء (٩٩,١٤٪).

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعـــة بـابـــل
كلية تكنولوجيا المعلومات - قسم البرمجيات

تقنية التعلم العميق لتشخيص الاخطاء في انظمة الطاقة الكهربائية

رسالة مقدمة إلى
مجلس كلية تكنولوجيا المعلومات - جامعة بابل كجزء من متطلبات
نيل درجة الماجستير في تكنولوجيا المعلومات / البرمجيات

من قبل
نور حسن فاضل نور

بإشـراف
أ.م.د. محمد حسين دوش

١٤٤٣ هـ ٢٠٢٢ م