Addis Ababa University

Addis Ababa Institute of Technology (AAiT)

School of Electrical and Computer Engineering

Use of artificial intelligence for predictive maintenance and management of Addis Ababa Light Rail Transit

By

Liban Ali Abdi

Advisor

Mr. Abebe Teklu

A Thesis Submitted to Addis Ababa University, Institute of Technology, in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Electrical Engineering (Railway)

December 2017

Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology (AAiT)

School of Electrical and Computer Engineering

Use of artificial intelligence for predictive maintenance and management of Addis Ababa Light Rail Transit

By

Liban Ali Abdi

Approval by Board of Examiners

_____	_____	_____
Chairman, Department Graduate Committee	Signature	Date
Ato Abebe Teklu	_____	_____
Advisor	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Abstract

Condition based monitoring is gaining much importance in the industry because of the need to increase machine reliability and reduce the potential loss of production due to breakdowns caused by different defects. In this thesis, we are interested in a condition based monitoring techniques using artificial neural network approach, especially Multiple Layer Perceptron (MLP). The multiple layer perceptron networks trained with backpropagation algorithm are very frequently used to solve a classification problems. In order to keep the machine performing at its best, one of the principal tools for the diagnosis of signaling equipment problems is the acoustic analysis and also vibration analysis which can be used to extract the fault features and then identify the fault patterns. In addition, there is a demand for techniques that can make decision on the running health of the machine automatically and reliably.

Artificial intelligent techniques have been successfully applied to automated detection and diagnosis of railway signaling equipment conditions. They largely increase the reliability of fault detection and diagnosis systems. Accordingly, the aim of this paper is to apply a MLP to classify a large number of faulty signals acquired from turn out in different states: crack signal and fatigue signal. The extracted parameters is the peak ratio, one of the best indicators. The main impact of this neural network is to generate answers that give the combined state of crack and fatigue simultaneously whereas most of previous neural networks have focalized mainly on gears or on bearings alone.

Information about the signaling equipment obtained in the form of time signal indicators is converted into a frequency signal indicator using an algorithm designed and coded using MATLAB. The frequency data obtained using the algorithm then is used as an input for continuous learning by an artificial neural network. Based on this learning outcome, the state of signaling equipment can easily be defined and classified. We chose the renowned Multi-layer perceptron (MLP) an artificial neural network for the classification phase. From simulation, we obtained a learning rate of 98% showing our algorithm and equipment state classification as per the signal generated during operation is acceptable.

Keywords: *Condition based monitoring, signal processing, artificial neural network, multi-layer perceptron*

Acknowledgment

First and foremost, I would like to thank GOD for all the blessings He has thrown my way. I would also like to take this time to acknowledge those individuals who have been guiding and supporting influences throughout my academic career.

My special thanks are extended to my advisor, Abebe Teklu, for his support, guidance, patience and constructive criticism through the course of my graduate study and preparation of this dissertation.

The most substantial contributions to my entire education are, of course, from my parents and family who have sacrificed greatly in allowing me to pursue my education far away from home. They have taught me to love learning and let effort and determination lead my way and with their unconditional love and faith, they have given me the courage to dream. I sincerely dedicate this dissertation to them.

List of figure

List of table

List of abbreviation

AALRT- Addis Ababa light rail transit.

AI- Artificial intelligence.

ANN- Artificial neural network.

CBM- Condition based on monitoring.

FFNN- Feed Forward neural network

GDA- Gradient descent backpropagation.

LM- Levenberg-Marquardt.

NN- Neural network.

MLP- Multilayer perceptron.

PM- Predictive maintenance.

PMM- Predictive maintenance management.

MSE- Mean squared error.

RBF- Radial basis function.

Rprop- Resilient backpropagation.

SVM- Support vector machine.

TMB- Time based maintenance

Trainlm- Levenberg-Marquardt training algorithm.

Trainrp- Resilient backpropagation training algorithm.

Traingda- Gradient descent backpropagation training algorithm.

Chapter one
Introduction

1.1. Background

The three main approaches applied in the industry to maintenance are [1]:

1. Corrective maintenance,
2. Preventive maintenance,
3. Predictive maintenance.

1.1.1. Corrective maintenance

The main objective of the corrective maintenance is to identified earlier the specific signs like uncommon noises, shorter cycle time or obvious change in machine, run like motion speed and fluency, baseless stopping of machinery, overheating parts, etc.

1.1.2. The preventive maintenance

The role of the preventive maintenance is about how to inspect periodically the equipment by maintenance personnel, regardless of its condition. This process attempts to slow down wear processes leading to failures, and replace components at intervals shorter than their expected useful lifetimes. However, this method does not guarantee that a failure will never occur and the determination of the inspection intervals is not a simple task. It is important for an effective maintenance program to have advance warning of upcoming failures so the necessary planning can be made.

1.1.3. The predictive maintenance

The predictive maintenance (PM) is a philosophy and a new look on the maintenance strategies to achieve the maximum lifetime of machines while the risk of machine failure is minimized. The main idea is to ensure maximum operational life and minimum down time within predefined cost, safety, and availability constraints. Predictive maintenance [1] is performed as follows:

1. Identification of a relevant condition parameter,
2. Identification of a relevant maintenance parameter,
3. Determination of the actual conditions under which the components operates,
4. Performing the tests designed to simulate the actual operating conditions,
5. Establishing an appropriate maintenance strategy.

There are five techniques that are normally used for predictive maintenance management (PMM) [2]:

1. Vibration monitoring,
2. Process parameter monitoring,
3. Thermography,
4. Acoustic analysis,
5. Visual inspection.

This method is about how to minimize by identifying and detecting the problems before they become serious, there is lot of application for a predictive maintenance to identify and detect by different technique.

1.1.4. Artificial neural network applied to the predictive maintenance

The goal of this research is to present a neural network applied to fault diagnosis and fault detection, this kind of problem is called a classification problem. The classification is to predict the class y from the observations x. In our case, the observations x are the vibratory signals obtained during the monitoring phase (previously we explain the different techniques as vibration analysis), the goal of classification consists essentially to classify vibration signals and to predict in which class the signals belong. Indeed, there exist several types of classifiers that have been developed, the three famous classifiers are:

1. Multi layer perceptron (MLP).
2. Radial basis function (RBF).
3. Support vector machine (SVM).

Artificial neural network (ANN) is a method of artificial intelligence inspired by the human brain [3]. Similarly to this structure the artificial neural network is built of body called processing element, inputs and outputs, the basis model of a single artificial neuron consisted of a weight summer and an activation function [4].

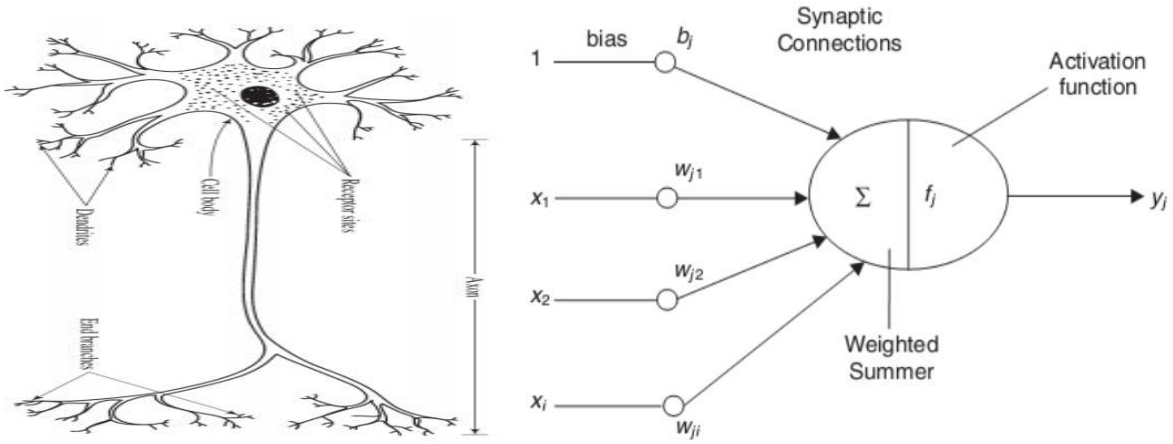


Figure 1. 1: Biological and artificial neuron *[4]*.

The MLP architecture is one of the most typical *feed-forward* neural network model or FFNN. The term feed-forward is used to identify basic behavior of such neural models, in which the impulse is propagated always in the same direction, e.g. from neuron input layer towards output layer, through one or more hidden layers (the network brain), by combining weighted sum of *weights associated to all neurons* (except the input layer) [4].

- A neuron in J_{th} layer, where is describe by:

$$x_1, x_2, x_3 \ldots\ldots\ldots\ldots\ldots\ldots x_i \text{ are input}$$

$$w_{j1}, w_{j2}, w_{j3} \ldots\ldots\ldots\ldots w_{ji} \text{ are weight}$$

$$b_i, \ldots\ldots\ldots\ldots \text{ biais}$$

$$f_i \text{ is the activation function}$$

$$y_i \text{ is the output}$$

- The weight sum S_i is therefore.

$$S_i = \sum_{i=1}^{N} w_{ji} * x_i + b_i \qquad (1.1)$$

- In matrix from.

$$S_i(t) = W_i * x + b_i \qquad (1.2)$$

The purpose of the learning phase's basically the manipulation or the iteration of input signal weights to ensure the best computational results are achieved. ANN tools are massively used for predictions in economics, weather forecast, chemistry, medicine and pharmacy and of course in industry.

Artificial neural networks show promising results as a robust tool for evaluation these data in order to support predictive maintenance activities. There exist a lot of application of ANN in maintenance. Mainly multi-layer perceptron's (MLP) are used for fault diagnosis of bearings,

induction motors, in railway equipment like track circuit, pantograph and track. Artificial neural networks have been intensively studied during the last two decades and successfully applied to dynamic system modelling as well as fault detection and diagnosis.

Problem Statement

The railway signaling equipment, as one of the most common types of railway equipment, these signaling equipment recently are getting more complicated, precise and expensive, fault diagnosis techniques for them have become more and more significant [3]. Most of the used signaling equipment for example turn out operates by means of crack/fatigue, strain and other signaling equipment frequently develop faults. These faults may cause the machine to break down and decrease its level of performance [4]. In order to keep the machine performing at its best and avoid personal casualties and economical loss, different methods of fault diagnosis have been developed and used effectively to detect and localize the faults in the specified element at an early stage. One of the principal tools for diagnosing signaling equipment problems are the vibration analysis and the acoustic analysis [5, 6]. Through the use of some processing techniques of vibration signals, it is possible to obtain vital diagnosis information. These techniques are used to extract the fault features and then identify the fault patterns. Many conventional methods such as Fourier analysis and time domain analysis are studied in the recent researches and executed in many applications [3]. However, many techniques available presently require a good deal of expertise to apply them successfully. Simpler approaches are needed which allow relatively unskilled operators to make reliable decisions without a diagnosis specialist to examine data and diagnose problems. Therefore, there is a demand for techniques that can make decision on the running health of equipment's automatically and reliably [7, 8, 9]. Artificial intelligent techniques, such as artificial neural networks (ANNs) and fuzzy logic, etc., have been successfully applied to automated detection and diagnosis of equipment conditions. They largely increase the reliability of fault detection and diagnosis systems.

Accordingly, in our application, we used a great deal of data composed of a large number of vibration signals acquired from turn out in different states: normal signal and abnormal signal. Among multiple techniques we have chosen the Frequency analysis, based on the Fourier transform specially Fast Fourier Transforms (FFT) as a tool for the vibration signal processing. We decided to extract multiple peak ratio value parameters from the vibration signals. After extracting all the parameters for all the signals, we aimed to develop a way to determine if these signals are acquired from a faulty or normal equipment, and to localize the fault by determining the faulty element (bearing or gear).

Due to their capability of learning and their capacity of classification and generalization, the ANNs are considered an ideal solution for this fault detection and classification. So we have conceived Feed Forward Neural Networks that take the power spectral density parameters as an input and generate answers that give the state of the machine, whether it's at a normal state or having a defected bearing or defected gear or both at the same time. As a result we have conceived a method to diagnose a rotating machine by detecting the fault and localizing it using simple spectral parameters with feed forward artificial neural networks. In this paper we present first the used signals in this application, and then define the parameters extracted from these signals. In addition, we give a brief explanation on the feed forward neural networks used and finally we present the results.

1.3. Objective

1.3.1. General Objective

The main objective of this thesis is to implement a predictive maintenance application, based on a neural network technique.

1.3.2. Specific Objective

The specific objectives of this thesis are:

1. To investigate condition based monitoring for signaling equipment.
2. To investigate predictive maintenance for railway signaling equipment
3. To investigate and develop Neural Network Multi-Layer Perceptron (MLP) for predictive maintenance.
4. Conduct MATLAB simulation for the developed MLP

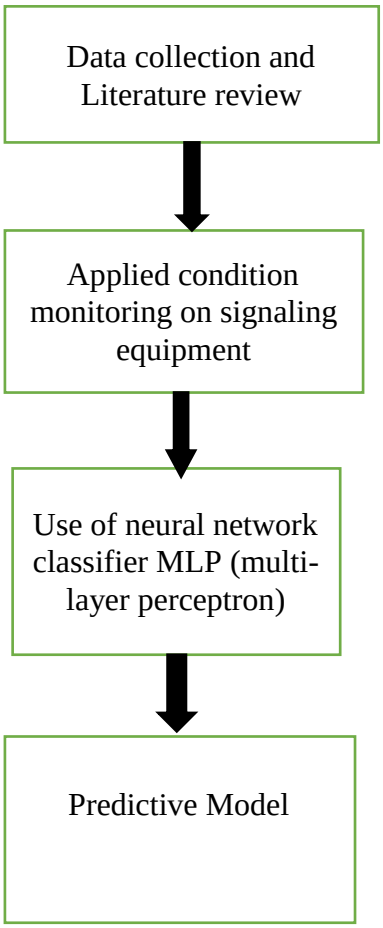
1.4. Research method

Initially, a thorough literature review was done about ANN application for predictive maintenance, generally, the process to solve a fault detection and diagnosis base on artificial neural network are:

1. Data collection (this phase consist to collect the vibration signal).
2. Classification phase (this is the training network).
3. Fault detection, diagnosis.

Neural network models have a significant advantage over analytic models, though, because they require only the indicator signal history as input and no assumptions. ANN tools are massively used for predictions in economics, weather forecast, chemistry, medicine and pharmacy and of course in industry. The signal indicators were classified in three classes: no defect or normal

signal; probable deficiency and abnormal signal or defect. This procedure advances from the fact, that it is robust against the differences of subject emissivity on measuring its vibration. Artificial neural networks show strong potential in industrial applications, especially in predictive maintenance tasks. Due to need of well-trained person for ANN computations, necessity of special software and availability of sensors able to capture and systematically store the collected data it has to be properly evaluated and the collected data can be also used as input to the ANN more precisely MLP (multilayer perceptron).



1.5. Scope of the Thesis

The scope of this thesis are:-

1. Using fast Fourier transforms algorithm to extract all the parameters for all the signals.
2. The type of ANNs, we used the multilayer perceptron (MLP), these networks are mostly used for classification because of their simplicity and their capacity of learning.
3. The desired goal is to get a high learning performance for all the training data, in order to identify the signals.
4. Discussing the result.
5. Finally, conclude the result obtained.

1.6. Thesis organization

This thesis is organized into five chapters, each chapter of this thesis explains and give as tools to understand well the studied problem. The first chapter defines the subject of the thesis otherwise it include the problem statement and the research method to solve the existing problem. Finally, the main goal of this thesis is to design and investigate neural network technique aid to fault detection and diagnosis.

The second chapter present the materials and equipment sensor and existing different sensor to measure of each material or signaling equipment. The various kinds of faults and the protection techniques that are currently available and employed are briefly discussed. Some important results from the research on the existing signaling equipment it also present in this thesis

The third chapter introduces the concept behind artificial intelligence and how to train a neural networks. A few ANN architectures that are usually employed are discussed and the various learning strategies employed in the training process of the neural networks along with the critical factors that affect the size and output of a trained network are discussed in this chapter.

The fourth chapter deals with the actual implementation and development of the neural networks and their architectures proposed for the three different parts of the condition based maintenance using artificial neural network process namely fault detection, classification and fault location. An overview of the training and testing processes employed with neural networks in this work has been outlined in this chapter. In chapter presents series of simulation results that have been obtained using MATLAB and the Artificial Neural Networks Toolboxes in Simulink in detail to emphasize the efficiency and accuracy factors of the proposed diagnosis

fault. Several neural networks with varying configurations have been trained, tested and their performances have been analyzed in this chapter.

The fifth chapter concludes the entire research work and the thesis. It discusses the results obtained in the previous chapters. Moreover, the scope for recommendation and possible extensions to this work has been outlined briefly in this chapter.

Chapter two
Literature review

2.1. Related work

The fault diagnoses usually follows after the fault detection wherein the kind, size and location of the fault will be determined. In the past two decades, the techniques of neural networks are growing, as a data-driven method, which provides a totally new perspective to fault diagnosis. Neural networks can be applied for fault diagnosis using different approaches. Pattern recognition approach and residual generation followed by decision making are the most common ones. The second approach is generally more suitable for dynamic systems. Different neural network architectures have been tried for fault diagnosis. The [5] have used the Hopfield network for identification of system parameters. The obtained parameters are further passed to Adaptive Resonance Theory (ART) network for fault diagnosis. The problem with this method is the choice of the optimal window size to detect the system parameters. In this paper [6] have proposed a model based fault diagnosis method to detect and isolate faults in the robot arm control system. The fault in the system is detected when the error (i.e. difference between the system output and the estimated output) exceeds a predetermined threshold. Once the fault is detected the estimated parameters are transferred to the fault classifier. The [7] have proposed a new neural network for fault diagnosis of rotating machinery which synthesizes the ART and the learning strategy of Kohonen network. The [8] have proposed a novel method for fault diagnosis of analog circuits with tolerance using wavelet packet decomposition and probabilistic neural networks. The fault feature vectors are extracted and fed to the probabilistic neural network. The BPN is probably the most widely used neural network structure for classification. BPN can be viewed as the gradient descent technique, used to minimize the total squared error of the output and therefore possesses a high degree of credibility.

In [9] apply acoustic emission on wooden sleepers to monitor this materiel and he chooses two neural network and one learning algorithm for classification phase. MLP, RBF and SVM this is renowned algorithm. Famous for theirs high learning ability, prediction and classification. The pattern recognition and classification approach is taken to automate such intuitive human skills for the development of more robust and reliable testing methods. Features were extracted from the impact acoustic emissions of wooden sleepers and were used for pattern classification. Time-frequency based feature extraction techniques such Short-time Fourier Transform and Discrete Wavelet Transform yielded good results. In thesis, [] test and compare the learning

performance of three learning algorithm, this algorithm is a multi-layer perceptron, radial basis function neural networks and support vector machine classifiers. Further classifier fusion was investigated by considering the output of single best classifiers as input to a new classifier with an aim of improving performance. He got a good results experimentally and demonstrate his classification accuracy of around 84%.

In [10] use a classifier Dempter-shafter for fault detection and isolation in railway track circuits. His calculation parts it's so interesting because he explain well the theoretical background and mathematical part of this classifiers and he also works on one of important material of signaling equipment. A track circuit can be considered as a large-scale system composed of a series of trimming capacitors located between a transmitter and a receiver. A defective capacitor affects not only its own inspection data (short circuit current) but also the measurements related to all capacitors located downstream (between the defective capacitor and the receiver). Here, the global fault detection and isolation problem is broken down into several local pattern recognition problems, each dedicated to one capacitor. The outputs from local neural network or decision tree classifiers are expressed using Dempster-Shafer theory and combined to make a final decision on the detection and localization of a fault in the system. Experiments with simulated data show that correct detection rates over 99 % and correct localization rates over 92% can be achieved using this approach, which represents a major improvement over the state of the art reference method.

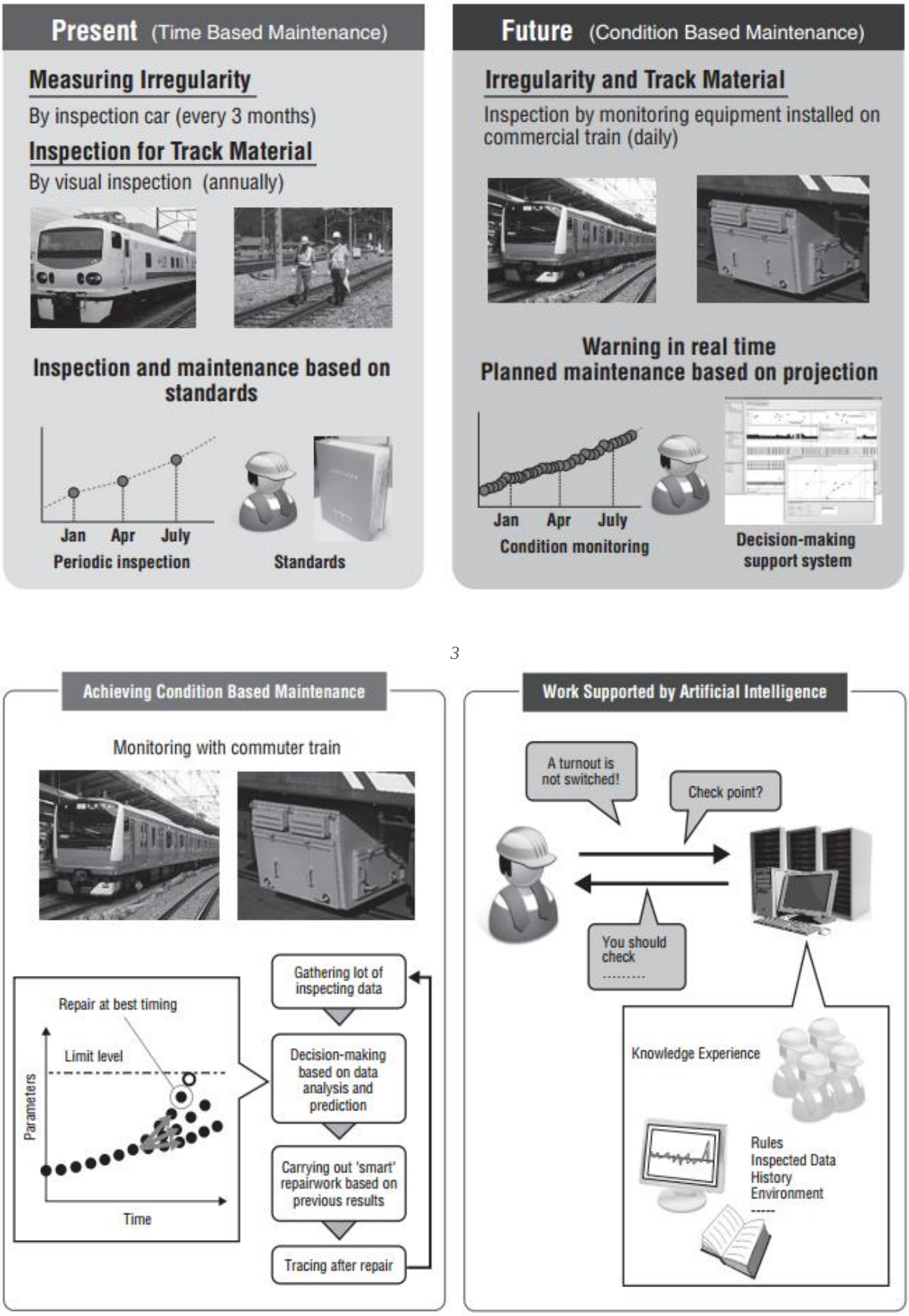
In [7], this paper is best study, I never read on condition based on monitoring for rotating machine and they use genetique algorithm to optimize the learning performance of the artificial neural network. Due to this paper, we opted to optimize our learning performance and we choose component principle analysis (CPA) to reduce the dimension of the training data. The fault diagnosis techniques have become more and more significant. In order to keep the machine performing at its best, one of the principal tools for the diagnosis of rotating machinery problems is the vibration analysis, which can be used to extract the fault features and then identify the fault patterns. In addition, there is a demand for techniques that can make decision on the running health of the machine automatically and reliably. Artificial intelligent techniques have been successfully applied to automated detection and diagnosis of machine conditions. They largely increase the reliability of fault detection and diagnosis systems. Accordingly, the aim of this paper is to apply a feed-forward efficient neural network to classify a large number of vibration signals acquired from rotating machinery in

different states: normal, good gear but faulty bearing, good bearing but faulty gear and faulty gear and bearing. The parameters given to the neural networks have been extracted from the power spectral density of the signals. The main impact of this neural network is to generate answers that give the combined state of gears and bearings simultaneously whereas most of previous neural networks have focalized mainly on gears or on bearings alone.

2.2 Condition based maintenance

Condition monitoring is important for safely prolonging the life of costly assets [1]. However, many condition monitoring systems produce too much data for engineers to view and assess, leading to useful indicators of health being overlooked. This could be solved with a condition monitoring architecture capable of anomaly detection, diagnosis, and prognosis, extracting as much information as possible from condition data.

First it's changing the basis of maintenance from Time Based Maintenance (TBM) to Condition Based Maintenance. (CBM).This entails a major change in the philosophy of maintenance. By achieving the change, much more streamlined maintenance than now will be possible. The difference is explained as follows using an example of maintenance for tracks, a typical type of railway facility. As shown in Fig. 5, inspections in TBM up to now are conducted at regular cycles (once every three months for conventional line track) to obtain data on track irregularity. Decisions on whether or not to conduct repairs are made based on this data and predetermined rules.



3

Figure 2. 1: Difference between TBM and CBM [11].

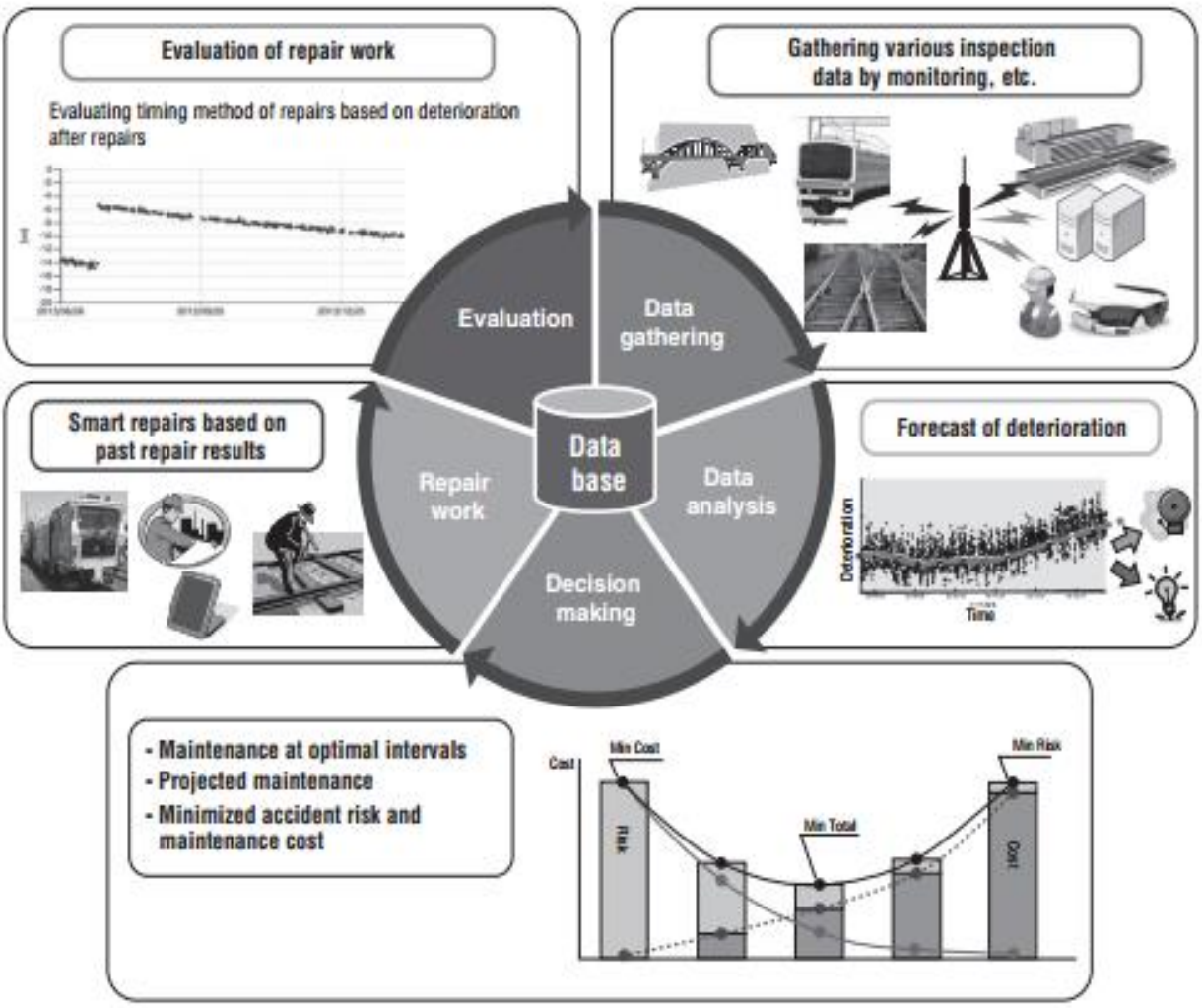


Figure 2. 2: Condition maintenance cycle [6].

Such criteria are based on maximum progression of irregularity taking into consideration the track irregularity at which derailment could occur from past data and the three month inspection cycle. As derailment must not occur, repair criteria needs to be set with much leeway, taking into consideration maximum progression of irregularity under the assumption that inspections are performed in a set cycle. For example, it is said that over 40mm longitudinal track irregularity causes derailment with overwhelming probability in Japanese conventional line based on previous studies. On the other hand, if over 23mm one (this value is showed in Figure 2.2) was measured, repair works must be done in 15 days based on rules.

Conversely, CBM is based on monitoring of large volumes of data obtained rather than performing inspections at set intervals. In the example of tracks, obtaining track displacement data from trains in operation would allow such displacement data to be obtained every day. Analyzing that data would allow us to identify the speed at which track deteriorates (i.e., the condition of the equipment) in units of 1m. Decisions can thus be made on when to conduct

repairs at the optimum timing while accurately predicting track irregularity by individual location, allowing very streamlined preventive maintenance.

Changing from TBM to CBM means a fundamental change in justification for decision-making for the priority matter in maintenance of "when to conduct what sort of repairs" (Figure 2.2).

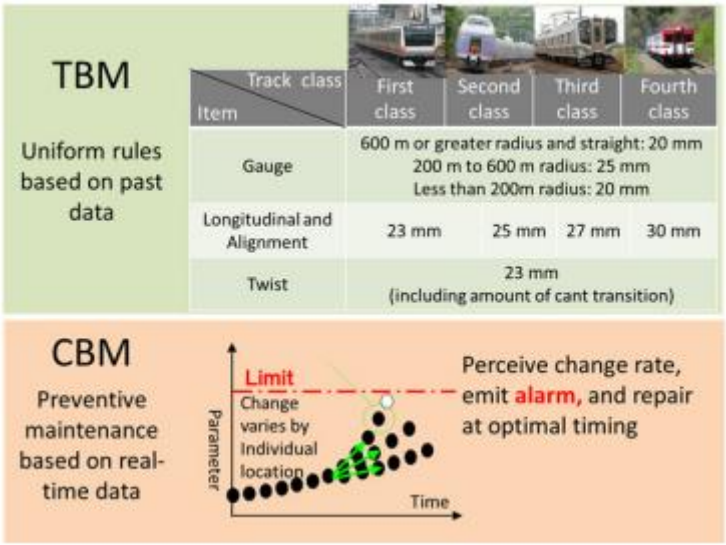


Figure 2. 3: TBM and CBM forecasting [6].

With CBM, the cycle shown in Figure 2.3 can be followed on a daily and dynamic basis. That cycle entails obtaining data, identifying the state of deterioration by data analysis, making decisions regarding time/method/location of repairs, conducting repairs, and confirming and evaluating the results of repairs. The more data is accumulated, the smarter the important decision-making process in maintenance is. Justification for decision-making in TBM is prescribed by rules (internal regulations, etc.), so there is inevitably little awareness of reviewing on a daily basis. In fact, maintenance criteria for track maintenance have not changed in about 50 years.

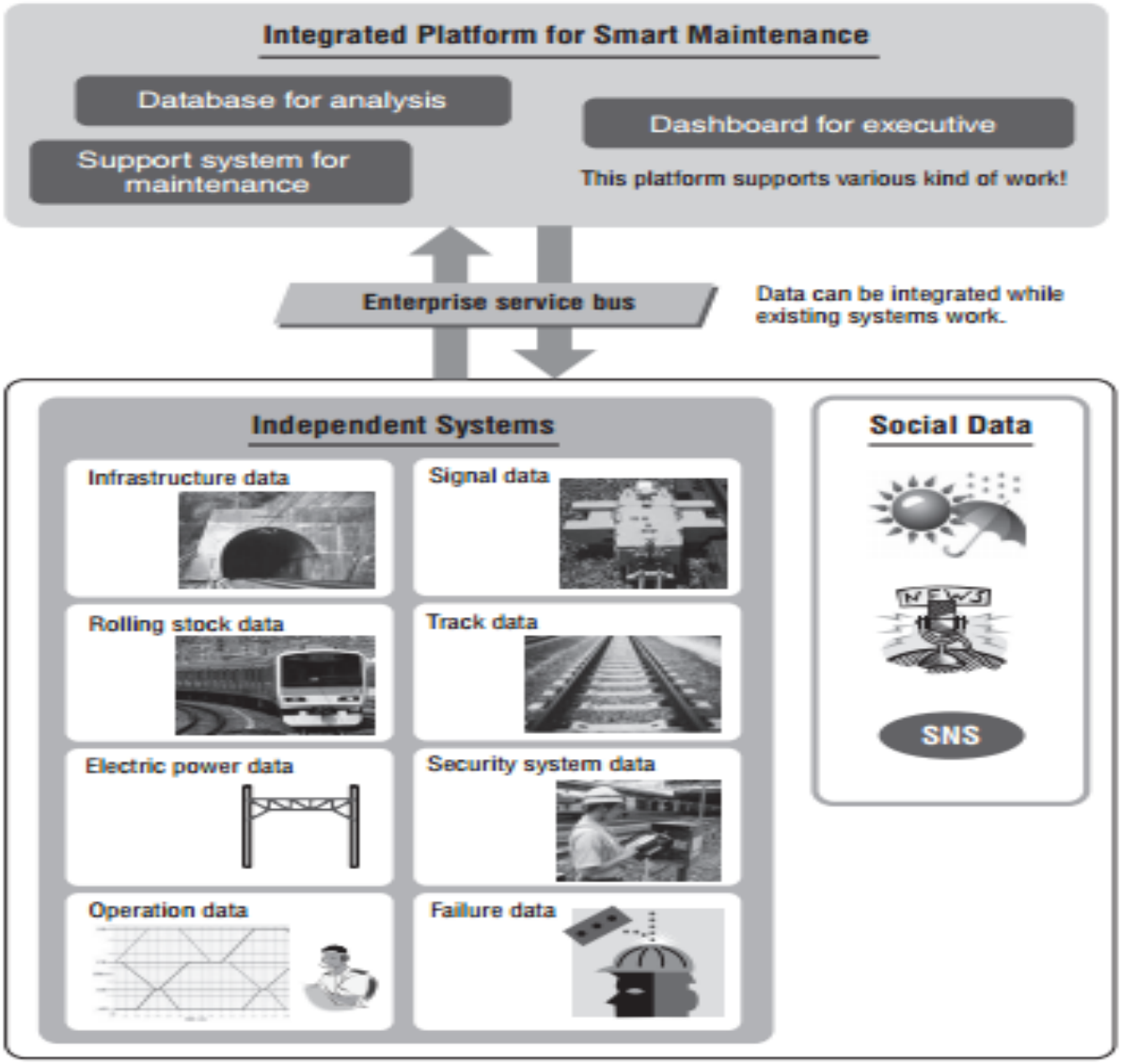


Figure 2. 4: Integrated platform for smart maintenance [6].

There are a number of promising directions for further research in condition monitoring in the railways. One future direction is a move toward holistic integrated systems [31], [48], [48] which provide near real-time information and alerts. These approaches will integrate data from different sensor systems using sophisticated modeling techniques. They will also incorporate more contextual data into the modeling, including the ambient conditions, the route, the journey time, the weight of the train, and more. The models will use multistage data fusion. This fuses measurements from a variety of sources (sensors and contextual data) and over a range of time epochs to generate a consolidated state history of the object being monitored [38]. The models will also mitigate data dependencies (physical dependencies) across the physical objects being monitored. An integrated multifaceted approach should improve prediction quality. For example, train monitoring data can be combined with route data and GPS data. If multiple trains detect a vibration fault at exactly the same GPS location, then

the fault is more than likely in the rail infrastructure such as uneven track. However, if only one train identifies a vibration anomaly at a particular GPS location, then the fault is more likely to be in the rolling stock [64].

Table 2. 1 Table detailing the sensor device used in railway condition monitoring

Object monitored	Measurement	Sensor	Citation
Track	Crack/fatigue detection	Acoustic emission	[12] [13] [14]
	Out of round wheel	Acoustic emission	[12] [13] [14]
		Accelerometer	[15]
	Stresses	Strain gauge	[16] [17] [18]
	Vibration (dynamic)	Accelerometer	[19] [20]
	Settlement and twist	Inclometer	[21]
	Incline	Inclometer	[22]
Track infrastructure	Pressure	Piezoelectric pressure sensors	[23]
	Strain	Fiber bragg strain gauge	[24]
	Displacement	Magnetic	[25]
	Stress	Strain gauge	[26]
	Temperature	Strain gauge	[26]
		Thermocouples	[26]
		SAW temperature	[26]
	Vibration	Accelerometers	[26]
Wheel	Vibration across surface	Piezoelectric	[27]
	Lateral acceleration	Piezoelectric	[27]
		Accelerometer	[28], [29]
	Wheel acceleration	Gap sensor	[30]
	Lateral (contact) force	Magnetostrictive displacement sensor	[30]
	Vertical (contact) force	Gyro	[29]
Sleepers	Crack/fatigue detection	Accelerometer	[28]
	Stresses	Strain gauge	[16, 14] [17] [18]
	Vibration (dynamic)	Accelerometer	[19] [20]
	Pressure	Piezoelectric pressure sensors	[23]
Track Circuit	Current flow	Magnetoelectric current	[31]
	Strain	Fiber bragg strain gauge	[24]
Axel Counter	Strain	Fiber bragg strain gauge	[24]
	Vibration (dynamic)	Accelerometer	[19] [20]

2.3 Signal processing

Signal processing is a very important step to detect failure and also commonly used in the predictive maintenance to monitor a real-world systems by sensor data. In fact, the data obtained, it is important to know which information is relevant. These signals must be processed in order to be replaced by a vector of parameters to simplify the classification procedure. So, we had to choose some features that can include the most important information contained in the signal and then extract them in order to prepare the matrices of learning and testing for the neural networks.

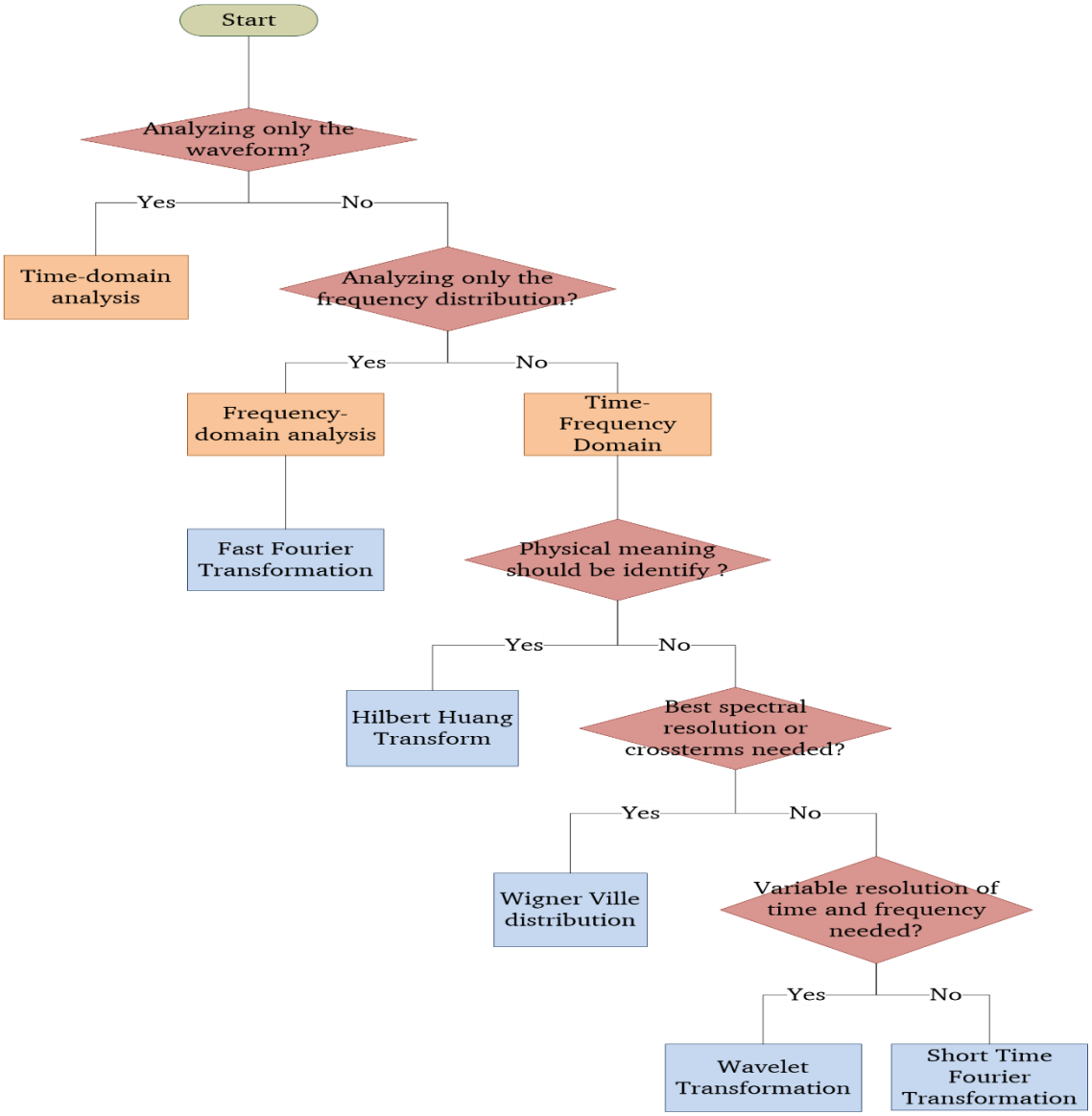


Table 2. 2: Literature review used signal processing

Reference	Time Domain (TD)	FFT	STFT	WT	HHT	WV
[32]		X				
[33]				X		
[34]		X		X		
[5]	X					
[35]			X			
[36]				X		
[37]				X		
[38]						X
[39]						X
[40]					X	
[41]					X	
[40]						X
[5]					X	
[41]					X	
Sum	**1**	**2**	**1**	**4**	**4**	**3**

Over time is involved, the time domain analysis can be used to extract more information as a new value. If only the change in the frequency of sensor value is relevant, the frequency domain analysis must be used. As mentioned in Table 2.2, the most discussed signal processing technique in the research area of failure type detection and predictive maintenance is the time frequency analysis. Here, the value change over time and the frequency of value changes are relevant. The Short Time Fourier Transform (STFT) is discussed as an introduction to the terms of time frequency analysis. The STFT works with window functions where the window slides over time, and processes the sensor data inside the window. The drawback of the STFT is the fixed time and the frequency resolution. A technique related to the STFT is the Wavelet Transformation which has wavelets instead of windows functions. These wavelets can have different time resolutions for different frequencies. A drawback of Signal processing techniques with windows functions, or wavelets, is that leakage effects can occur at the beginning or the end of the time window / wavelet [33].

The Wigner ville distribution (WVD) has the best spectral resolution of all time frequency methods, and can analyze two signals at a time. Two signals that are compared by the WVD give rise to cross terms. These cross terms are generally not preferable, but they could be interesting in terms of failure type detection and predictive maintenance. Hilbert huang transformation (HHT) consists of two steps, first, the decomposition into a finite number of intrinsic mode functions, and second, their transformation. It is possible to identify the physical meaning of the signal using this signal processing technique. This is a very interesting feature for failure type detection and predictive maintenance, but it is also expensive, and it is very special for a real-world system to define a HHT which interprets the physical meaning. A decision tree can help to choose and to find the right signal processing technique for a given failure type detection and predictive maintenance problem.

Chapter three
Research methodology

3.1. Feature extraction

The data used in this thesis was obtained at the University of Paris VIII in France. The two signal presented see in section 3.1.1, those signal is obtained by the accelerometer sensor, which was placed and tested on turn out. In our study we have used the accelerometer signals only. Since errors can take place either in bearing elements or in crack, the acquisition of signals was done in two different cases:

1. Crack signal,
2. Fatigue signal.

In addition, the bearing defection can be either a ball defection or inner race defection or outer race defection, which leads us to eight different cases instead of four. In each case there are about thirteen signal acquired for a different shaft speed and a different load, speeds were 3Hz, 6Hz or 10 Hz, and loads were 25Nm 50Nm, 75Nm or 100Nm. To increase the number of elements in each case (or class) we divided each signal into 4 segments, thus we have a minimum of 52 signals representing each case. The classification of bearing defection was done in previous work by using feed forward neural networks with a high performance [10].

Plus, once any element of a bearing was defected the entire bearing will be replaced, thus we don't need to determine the defected element in the bearing. Consequently, the classification we aimed to do was to determine whether the gear or the bearing was defected or not and the signals of each class were chosen randomly from the subclasses.

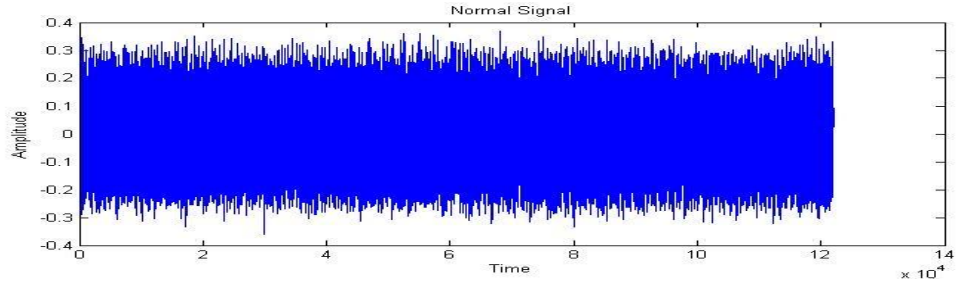


Figure 3. 1: Crack signal collected by the sensor

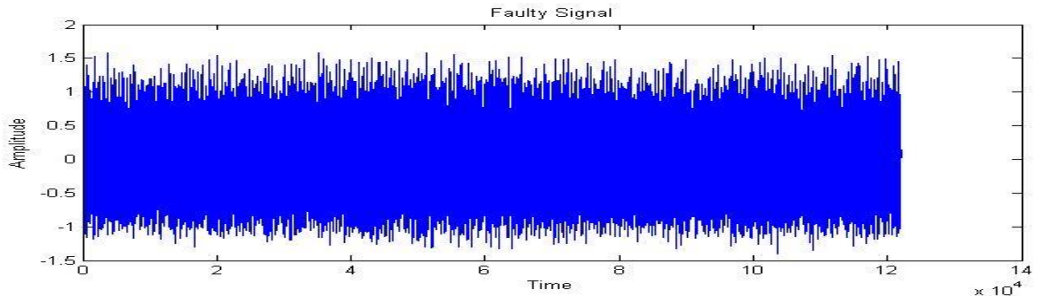


Figure 3. 2: Fatigue signal collected by the sensor

3.2. Solving algorithm

3.2.1. Fast Fourier transforms

According to the set of signals already presented, each crack and fatigue is represented by a set of vibration signals. These signals must be processed in order to be replaced by a vector of parameters to simplify the classification procedure. So, we had to choose some features that can include the most important information contained in the signal and then extract them in order to prepare the matrices of learning and testing for the neural networks. Frequency analysis has become a fundamental tool for vibration signal processing. It is based on the Fourier Transform that allows the passage from time domain to frequency domain. This transformation allows knowing the variation of power of the signal with the frequency f. Thus, it can detect the presence of a bearing or gear defection generating a periodic shock at a determined frequency [11]. Therefore, we decided to extract the peak ratio frequency parameters from the vibration signals we already have. Before the extraction, we divided each signal into 4 segments to have enough elements representing each class (bearing and gear state). Then, we extracted the parameters of the power spectral density (PSD) listed below (every parameter with its formula):

1. Power of the Signal:

$$M_r = 2 * \int_0^{\infty} f^r * S_x(f) df \qquad (3.1)$$

2. Skewness :

$$CD = \frac{M_3^*}{\sqrt{M_2^{3*}}} \qquad (3.2)$$

$$M_r^* = 2 * \int_0^{\infty} (f - MPF)^r * S_x(f) df \qquad (3.3)$$

3. Kurtosis:

$$CA = \frac{M_4^*}{M_2^{2*}} \tag{3.4}$$

4. Median Frequency:

$$\int_0^{Fmed} S_x(f)df = \int_{Fmed}^{Fmax} S_x(f)df \tag{3.5}$$

5. Relative energy by frequency band (10 values):

$$W_n = \frac{\int_{f_{n-1}}^{f_n} S_x(f)df}{M_0} \tag{3.6}$$

$$f_n = \frac{n}{N} * f_{max} \tag{3.7}$$

6. Deciles (8 values):

$$\int_{f_{n-1}}^{f_n} S_x(f)df = k \int_0^{Fmed} S_x(f)df \tag{3.8}$$

7. Spectral entropy:

$$H = -\int_0^{f_{max}} S_x(f) * \ln\big(S_x(f)\big)\, df \tag{3.9}$$

8. Peak ratio:

$$P_r = \frac{N * \sum_{j=1}^{n} P_h}{\sum_{k=1}^{N} A_i} \tag{3.10}$$

3.2.1.2. Fast Fourier transforms in Matlab

```
load x130 %load the file.txt
mou=X130(:,2);
[m,n]=size(X130);
x130freq=fft(mou);%calculate fast fourier transforms
%calculate the magnitude
for i=1:m
        module_X130(i)=abs(x130freq(i));
end
        module_X130=module_X130';
        frequ_X130=X130(:,1)/0.00008;%calculate the frenquency
        save ('X130','frequ_X130','module_X130')% save the matrix
Frequmodule
load X130.mat % load the matrix frequ_module
          frequ_X130=frequ_X130;
          module_X130=module_X130;
          fid=fopen('X130.txt','w');% write the file txt
for i=1:length(frequ_X130);fprintf(fid,'%5.5f
%5.5f\n',frequ_X130(i),module_X130(i));
end
fclose(fid)
          s=0;
          nb_mesure=0;
          somA=0;
          somAf=0;
          Pr=0;
          fid=fopen('x238.txt');
while 1
tline=fgetl(fid);
if ~ischar(tline),break,end;
          nb_mesure=nb_mesure+1;
end;
fclose(fid);
s=load('x238.txt');
disp(s);
          f= input('Frenquency f=');

for i= 1 : nb_mesure
          somA=somA+s(i,2);
if(rem(s(i,1),f)==0)
          somAf=somAf+ s(i,2);
```

```
            end
end
          Pr=nb_mesure*(somAf/somA);
disp(sum of the global amplitue:');
disp(somA);
disp(sum of the associate amplitude:');
disp(somAf);
disp('diplay the peak ratio value:');
disp(Pr);
```

3.2.2. Multilayered perceptron with backpropagation algorithm

Biological neural networks are very complex. In contrast, the mathematical model of the network is much more simplified and is based on several assumptions:

1. All neurons are synchronized. That means that the signal passing from one neuron to another takes the same time for all connections. Signal processing is also synchronized and is the same for all neurons.
2. Every neuron has a so-called transfer function which determines neuron's output signal depending on the input signal strength. That function is time-independent.
3. When the signal passes the synapse, it changes linearly, i.e., the signal value is multiplied by some number. That number is called **synaptic weight**. The very important property of the synaptic weight is that it changes in time. That feature makes it possible for brain to react differently on the same input in different moments. Or, in other words, to learn.

Of course, those assumptions simplify the initial biological neural network very much. For example, brain signal transmission time naturally depends on the distance between neurons. But despite those simplifications, artificial networks still preserve the most important characteristics of biological networks - adaptability and ability to learn. The first mathematical model of the neuron was introduced more than a half century ago, but did not change much since then. First of all, the neuron is seen as a simple "automate" that transforms input signals into the output signal (see Fig. 1.1).

The model functions as follows: inputs of the neuron's synapses receive N signals $\{ X_1, \dots \dots, X_n,\ n \epsilon \mathbb{N} \}$. Then every synapse makes a linear modification of the signal using its synaptic weight. After that neuron's body (soma) receives signals $\{ X_1 * w_1, \dots \dots, X_n * w_n,\ n \epsilon \mathbb{N} \}$ where w_i is the corresponding synaptic weight) and sums those signals:

Function name	Formula	Values range
Linear	$F(S) = kS, k \in \mathbb{R}_+$	$(-\infty, \infty)$
Semi linear	$F(S) = \begin{cases} kS, S > 0, k \in \mathbb{R}_+ \\ 0, S \leq 0 \end{cases}$	$[0, \infty)$
Sigmoid	$F(S) = \frac{1}{1+e^{-\alpha S}}$	$(0, 1)$
Bipolar sigmoid	$F(S) = \frac{2}{1+e^{-\alpha S}} - 1$	$(-1, 1)$
Hyperbolic tangent	$F(S) = \frac{e^{\alpha S} - e^{-\alpha S}}{e^{\alpha S} + e^{-\alpha S}}$	$(-1, 1)$
Exponential	$F(S) = e^{-\alpha S}$	$(0, \infty)$
Sinusoidal	$F(S) = \sin(S)$	$[-1, 1]$
Fractional	$F(S) = \frac{S}{\alpha + \lvert S \rvert}$	$[-1, 1]$
Step	$F(S) = \begin{cases} 1, S \geq 0 \\ 0, S < 0 \end{cases}$	$[0, 1]$
Signature	$F(S) = \begin{cases} 1, S \geq 0 \\ -1, S < 0 \end{cases}$	$[-1, 1]$
Binary step	$F(S) = \begin{cases} -1, S \leq -1 \\ S, -1 < S < 1 \\ 1, S \geq 1 \end{cases}$	$[-1, 1]$

Figure 3. 3: MLP Activiation function [3].

1. **Input neurons:** Those neurons are taking input vector that encodes some action or information about the external environment. Input neurons don't perform any type of computation, but only pass the input vector to subsequent neurons.

2. **Output neurons** receive signals from the preceding neurons and transform it using formulas 2.1 and 2.2. Those values represent output of the whole neural network.

3. **Hidden neurons** are the basis of the neural network. Those neurons receive the signal from the input neurons or preceding hidden neurons, process it in accordance with equation 1.1 and 1.2 and then pass result signals to the subsequent (hidden or output) neurons.

4. **Learning rate.** Learning rate is a number between 0 and 1 that determines how fast the neural network adjusts itself to the patterns in the training data. It does not have to be a constant and can also dynamically decrease or increase with time. That parameter must be chosen carefully - too small one will make the learning process slow, and too large one might lead to the divergence. Jacobs in [15] suggest modifying learning rate

dynamically and increasing it as long as gradient keeps pointing the same direction, but lowering it when the gradient changes its sign.

5. **Momentum:** Momentum term influences the way of how previous weights affect the current one. It helps the algorithm in preventing being stuck in a local minimum. That value must also be chosen carefully and should be determined experimentally. The use of momentum can be omitted. However it may improve neural network's performance greatly and therefore it is commonly used. The back propagation process can suffer from several problems:

- Reaching of the global error function minimum is not guaranteed by the method. As there can be several local minima, the algorithm might possibly remain at one of them.
- Back propagation method might lead to overfitting: in cases where learning was performed for too long or where the training examples are rare, the network may adjust to very specific random features of the training data that have no causal relation to the target function. The other possible cause of overfitting is having too many hidden neurons.
- There is no exact way of how to determine that the algorithm has found the best solution (i.e., global minimum). That problem can be generally approached by restarting the algorithm several times with some changes (e.g., shuffling training set) and therefore increase the probability that the reached minimum is the global one.

Despite those problems, back propagation is a very popular algorithm that seems to be useful in many areas ([3], [1]). Moreover, other training methods are also available – conjugate gradient descent, Quasi-Newton algorithm and Levenberg-Marquardt algorithm. However, those are not widely used. Another alternative training method is based on the genetic approach. The main principle is the following: weights of the network are not adjusted by the back propagation algorithm, but by the genetic algorithm. At the beginning, the initial population contains randomly generated sets of weights. An evolution algorithm is then applied to the population, it generates set of new weights and keeps only the most appropriate ones. That solution generally prevents the algorithm from being stuck in a local minimum and also shows a good performance. Networks that employ that approach are generally called Genetic Algorithm Neural Network or GANN. Kim in [17] gives an example and evaluates those networks in financial forecasting.

3.2.2.1. Multilayered perceptron in Matlab

```
%% Clear Variables, Close Current Figures, and Create Results Directory
clc;
clear all;
close all;
mkdir('Results//'); %Directory for Storing Results

%% Configurations/Parameters
dataFileName = 'sharky.spirals.points'; %sharky.linear.points - sharky.circle.points - sharky.wave.points - sharky.spirals.points
nbrOfNeuronsInEachHiddenLayer = [10 10]; %linear:[4] - circle:[10] - wave,spirals:[10 10]
nbrOfOutUnits = 2;
unipolarBipolarSelector = 0; %0 for Unipolar, -1 for Bipolar

learningRate = 0.15;
nbrOfEpochs_max = 500000;

enable_resilient_gradient_descent = 1; %1 for enable, 0 for disable
learningRate_plus = 1.2;
learningRate_negative = 0.5;
deltas_start = 0.9;
deltas_min = 10^-6;
deltas_max = 50;

enable_decrease_learningRate = 0; %1 for enable decreasing, 0 for disable
learningRate_decreaseValue = 0.0001;
min_learningRate = 0.05;

enable_learningRate_momentum = 0; %1 for enable, 0 for disable
momentum_alpha = 0.05;

draw_each_nbrOfEpochs = 100;
```

```
%% Read Data
importedData = importdata(dataFileName, '\t', 6);
Samples = importedData.data(:, 1:length(importedData.data(1,:))-1);
TargetClasses = importedData.data(:, length(importedData.data(1,:)));
TargetClasses = TargetClasses - min(TargetClasses);
ActualClasses = -1*ones(size(TargetClasses));

%% Calculate Number of Input and Output NodesActivations
nbrOfInputNodes = length(Samples(1,:)); %=Dimention of Any Input Samples
% nbrOfOutUnits = ceil(log2(length(unique(TargetClasses)))) + !; %Ceil(Log2( Number of Classes ))

nbrOfLayers = 2 + length(nbrOfNeuronsInEachHiddenLayer);
nbrOfNodesPerLayer = [nbrOfInputNodes nbrOfNeuronsInEachHiddenLayer nbrOfOutUnits];

%% Adding the Bias as Nodes with a fixed Activation of 1
nbrOfNodesPerLayer(1:end-1) = nbrOfNodesPerLayer(1:end-1) + 1;
Samples = [ones(length(Samples(:,1)),1) Samples];

%% Calculate TargetOutputs %TODO needs to be general for any nbrOfOutUnits
TargetOutputs = zeros(length(TargetClasses), nbrOfOutUnits);
for i=1:length(TargetClasses)
   if (TargetClasses(i) == 1)
      TargetOutputs(i,:) = [1 unipolarBipolarSelector];
   else
      TargetOutputs(i,:) = [unipolarBipolarSelector 1];
   end
end

%% Initialize Random Wieghts Matrices
Weights = cell(1, nbrOfLayers); %Weights connecting bias nodes with previous layer are useless, but to make code simpler and faster
Delta_Weights = cell(1, nbrOfLayers);
ResilientDeltas = Delta_Weights; % Needed in case that Resilient Gradient Descent is used
```

```
for i = 1:length(Weights)-1
   Weights{i} = 2*rand(nbrOfNodesPerLayer(i), nbrOfNodesPerLayer(i+1))-1; %RowIndex: From Node Number, ColumnIndex: To Node Number
   Weights{i}(:,1) = 0; %Bias nodes weights with previous layer (Redundant step)
   Delta_Weights{i} = zeros(nbrOfNodesPerLayer(i), nbrOfNodesPerLayer(i+1));
   ResilientDeltas{i} = deltas_start*ones(nbrOfNodesPerLayer(i), nbrOfNodesPerLayer(i+1));
end
Weights{end} = ones(nbrOfNodesPerLayer(end), 1); %Virtual Weights for Output Nodes
Old_Delta_Weights_for_Momentum = Delta_Weights;
Old_Delta_Weights_for_Resilient = Delta_Weights;

NodesActivations = cell(1, nbrOfLayers);
for i = 1:length(NodesActivations)
   NodesActivations{i} = zeros(1, nbrOfNodesPerLayer(i));
end
NodesBackPropagatedErrors = NodesActivations; %Needed for Backpropagation Training Backward Pass

zeroRMSReached = 0;
nbrOfEpochs_done = 0;

%% Iterating all the Data
MSE = -1 * ones(1,nbrOfEpochs_max);
for Epoch = 1:nbrOfEpochs_max

   for Sample = 1:length(Samples(:,1))
      %% Backpropagation Training
      %Forward Pass
      NodesActivations{1} = Samples(Sample,:);
      for Layer = 2:nbrOfLayers
         NodesActivations{Layer} = NodesActivations{Layer-1}*Weights{Layer-1};
         NodesActivations{Layer} = Activation_func(NodesActivations{Layer}, unipolarBipolarSelector);
```

```
        if (Layer ~= nbrOfLayers) %Because bias nodes don't have weights connected to previous layer
            NodesActivations{Layer}(1) = 1;
        end
    end

    % Backward Pass Errors Storage
    % (As gradient of the bias nodes are zeros, they won't contribute to previous layer errors nor delta_weights)
    NodesBackPropagatedErrors{nbrOfLayers} = TargetOutputs(Sample,:)-NodesActivations{nbrOfLayers};
    for Layer = nbrOfLayers-1:-1:1
        gradient = Activation_func_drev(NodesActivations{Layer+1}, unipolarBipolarSelector);
        for node=1:length(NodesBackPropagatedErrors{Layer}) % For all the Nodes in current Layer
            NodesBackPropagatedErrors{Layer}(node) = sum( NodesBackPropagatedErrors{Layer+1} .* gradient .* Weights{Layer}(node,:) );
        end
    end

    % Backward Pass Delta Weights Calculation (Before multiplying by learningRate)
    for Layer = nbrOfLayers:-1:2
        derivative = Activation_func_drev(NodesActivations{Layer}, unipolarBipolarSelector);
        Delta_Weights{Layer-1} = Delta_Weights{Layer-1} + NodesActivations{Layer-1}' * (NodesBackPropagatedErrors{Layer} .* derivative);
    end
  end

  %% Apply resilient gradient descent or/and momentum to the delta_weights
  if (enable_resilient_gradient_descent) % Handle Resilient Gradient Descent
    if (mod(Epoch,200)==0) %Reset Deltas
        for Layer = 1:nbrOfLayers
```

```
            ResilientDeltas{Layer} = learningRate*Delta_Weights{Layer};
        end
    end
    for Layer = 1:nbrOfLayers-1
        mult = Old_Delta_Weights_for_Resilient{Layer} .* Delta_Weights{Layer};
        ResilientDeltas{Layer}(mult > 0) = ResilientDeltas{Layer}(mult > 0) *
learningRate_plus; % Sign didn't change
        ResilientDeltas{Layer}(mult < 0) = ResilientDeltas{Layer}(mult < 0) *
learningRate_negative; % Sign changed
        ResilientDeltas{Layer} = max(deltas_min, ResilientDeltas{Layer});
        ResilientDeltas{Layer} = min(deltas_max, ResilientDeltas{Layer});

        Old_Delta_Weights_for_Resilient{Layer} = Delta_Weights{Layer};

        Delta_Weights{Layer} = sign(Delta_Weights{Layer}) .* ResilientDeltas{Layer};
    end
  end
  if (enable_learningRate_momentum) %Apply Momentum
    for Layer = 1:nbrOfLayers
        Delta_Weights{Layer} = learningRate*Delta_Weights{Layer} +
momentum_alpha*Old_Delta_Weights_for_Momentum{Layer};
    end
    Old_Delta_Weights_for_Momentum = Delta_Weights;
  end
  if (~enable_learningRate_momentum && ~enable_resilient_gradient_descent)
    for Layer = 1:nbrOfLayers
        Delta_Weights{Layer} = learningRate * Delta_Weights{Layer};
    end
  end

  %% Backward Pass Weights Update
  for Layer = 1:nbrOfLayers-1
    Weights{Layer} = Weights{Layer} + Delta_Weights{Layer};
  end
```

```
% Resetting Delta_Weights to Zeros
for Layer = 1:length(Delta_Weights)
    Delta_Weights{Layer} = 0 * Delta_Weights{Layer};
end

%% Decrease Learning Rate
if (enable_decrease_learningRate)
    new_learningRate = learningRate - learningRate_decreaseValue;
    learningRate = max(min_learningRate, new_learningRate);
end

%% Evaluation
for Sample = 1:length(Samples(:,1))
    outputs = EvaluateNetwork(Samples(Sample,:), NodesActivations, Weights, unipolarBipolarSelector);
    bound = (1+unipolarBipolarSelector)/2;
    if (outputs(1) >= bound && outputs(2) < bound) %TODO: Not generic role for any number of output nodes
        ActualClasses(Sample) = 1;
    elseif (outputs(1) < bound && outputs(2) >= bound)
        ActualClasses(Sample) = 0;
    else
        if (outputs(1) >= outputs(2))
            ActualClasses(Sample) = 1;
        else
            ActualClasses(Sample) = 0;
        end
    end
end

MSE(Epoch) = sum((ActualClasses-TargetClasses).^2)/(length(Samples(:,1)));
if (MSE(Epoch) == 0)
    zeroRMSReached = 1;
```

```
    end

    %% Visualization
    if (zeroRMSReached || mod(Epoch,draw_each_nbrOfEpochs)==0)
        % Draw Decision Boundary
        unique_TargetClasses = unique(TargetClasses);
        training_colors = {'y.', 'b.'};
        separation_colors = {'g.', 'r.'};
        subplot(2,1,1);
        cla;
        hold on;
        title(['Decision Boundary at Epoch Number ' int2str(Epoch) '. The max number of Epochs
is ' int2str(nbrOfEpochs_max) '.']);

        margin = 0.05; step = 0.05;
        xlim([min(Samples(:,2))-margin max(Samples(:,2))+margin]);
        ylim([min(Samples(:,3))-margin max(Samples(:,3))+margin]);
        for x = min(Samples(:,2))-margin : step : max(Samples(:,2))+margin
            for y = min(Samples(:,3))-margin : step : max(Samples(:,3))+margin
                outputs = EvaluateNetwork([1 x y], NodesActivations, Weights,
unipolarBipolarSelector);
                bound = (1+unipolarBipolarSelector)/2;
                if (outputs(1) >= bound && outputs(2) < bound) %TODO: Not generic role for any
number of output nodes
                    plot(x, y, separation_colors{1}, 'markersize', 18);
                elseif (outputs(1) < bound && outputs(2) >= bound)
                    plot(x, y, separation_colors{2}, 'markersize', 18);
                else
                    if (outputs(1) >= outputs(2))
                        plot(x, y, separation_colors{1}, 'markersize', 18);
                    else
                        plot(x, y, separation_colors{2}, 'markersize', 18);
                    end
                end
```

```
            end
        end

        for i = 1:length(unique_TargetClasses)
            points = Samples(TargetClasses==unique_TargetClasses(i), 2:end);
            plot(points(:,1), points(:,2), training_colors{i}, 'markersize', 10);
        end
        axis equal;

        % Draw Mean Square Error
        subplot(2,1,2);
        MSE(MSE==-1) = [];
        plot([MSE(1:Epoch)]);
        ylim([-0.1 0.6]);
        title('Mean Square Error');
        xlabel('Epochs');
        ylabel('MSE');
        grid on;

        saveas(gcf, sprintf('Results//fig%i.png', Epoch),'jpg');
        pause(0.05);
    end
    display([int2str(Epoch) ' Epochs done out of ' int2str(nbrOfEpochs_max) ' Epochs. MSE = '
num2str(MSE(Epoch)) ' Learning Rate = ' ...
        num2str(learningRate) '.']);

    nbrOfEpochs_done = Epoch;
    if (zeroRMSReached)
        saveas(gcf, sprintf('Results//Final Result for %s.png', dataFileName),'jpg');
        break;
    end
  end
display(['Mean Square Error = ' num2str(MSE(nbrOfEpochs_done)) '.']);
```

```
%% Activation Function
function fx = Activation_func(x, unipolarBipolarSelector)
   if (unipolarBipolarSelector == 0)
      fx = 1./(1 + exp(-x)); %Binary
   else
      fx = -1 + 2./(1 + exp(-x)); %Bipolar
   end
end
%% Activation Function
function fx_drev = Activation_func_drev(fx, unipolarBipolarSelector)
   if (unipolarBipolarSelector == 0)
      fx_drev = fx .* (1 - fx); %Binary
   else
      fx_drev = 0.5 .* (1 + fx) .* (1 - fx); %Bipolar
   end
end
function outputs = EvaluateNetwork(Sample, NodesActivations, Weights,
unipolarBipolarSelector)

nbrOfLayers = length(NodesActivations);

NodesActivations{1} = Sample;
for Layer = 2:nbrOfLayers
   NodesActivations{Layer} = NodesActivations{Layer-1}*Weights{Layer-1};
   NodesActivations{Layer} = Activation_func(NodesActivations{Layer},
unipolarBipolarSelector);
   if (Layer ~= nbrOfLayers) %Because bias nodes don't have weights connected to previous
layer
      NodesActivations{Layer}(1) = 1;
   end
end
outputs = NodesActivations{end};

end
```

Chapter four

Matlab simulation and discussion of simulation result

4.1 Matlab simulation

The Neural Network toolbox in Simulink by The Math Works divides the entire set of data provided to it into three different sets namely the training set, validation set and the testing set. The training data set as indicated above is used to train the network by computing the gradient and updating the network weights. The validation set is provided during to the network during the training process (just the inputs without the outputs) and the error in validation data set is monitored throughout the training process. When the network starts overfitting the data, the validation errors increase and when the number of validation fails increase beyond a particular value, the training process stops to avoid further overfitting the data and the network is returned at the minimum number of validation errors [44].

Two important steps in the application of neural networks for any purpose are training and testing. The first of the two steps namely training the neural network is discussed in this section. Training is the process by which the neural network learns from the inputs and updates its weights accordingly. In order to train the neural network, we need a set of data called the training data set which is a set of input output pairs fed into the neural network. Thereby, we teach the neural network what the output should be, when that particular input is fed into it. The ANN slowly learns the training set and slowly develops an ability to generalize upon this data and will eventually be able to produce an output when a new data is provided to it. During the training process, the neural network's weights are updated with the prime goal of minimizing the performance function. This performance function can be user defined, but usually feedforward networks employ Mean Square Error as the performance function and the same is adopted throughout this work. For the task of training the neural networks for different stages, sequential feeding of input and output pair has been adopted. In order to obtain a large training set for efficient performance. The two kinds of faulty indicator and one normal indicator has been simulated. All data collected in this simulation concerned turnout a signaling equipment, we teach this data our multilayer perceptron.

4.2 Simulation result and discussion

Fig 4.1 shows decision boundary and MSE, this training algorithm plot of the neural network 2-10-1 (2 neurons in the input layer, 1 hidden layer with ten neurons in it and one neuron in the output layer), and show as separate into two parts (yellow and blue). We can say and it can be seen that the network did not achieve since the desired Mean Square Error (MSE) goal by the end of the training process.

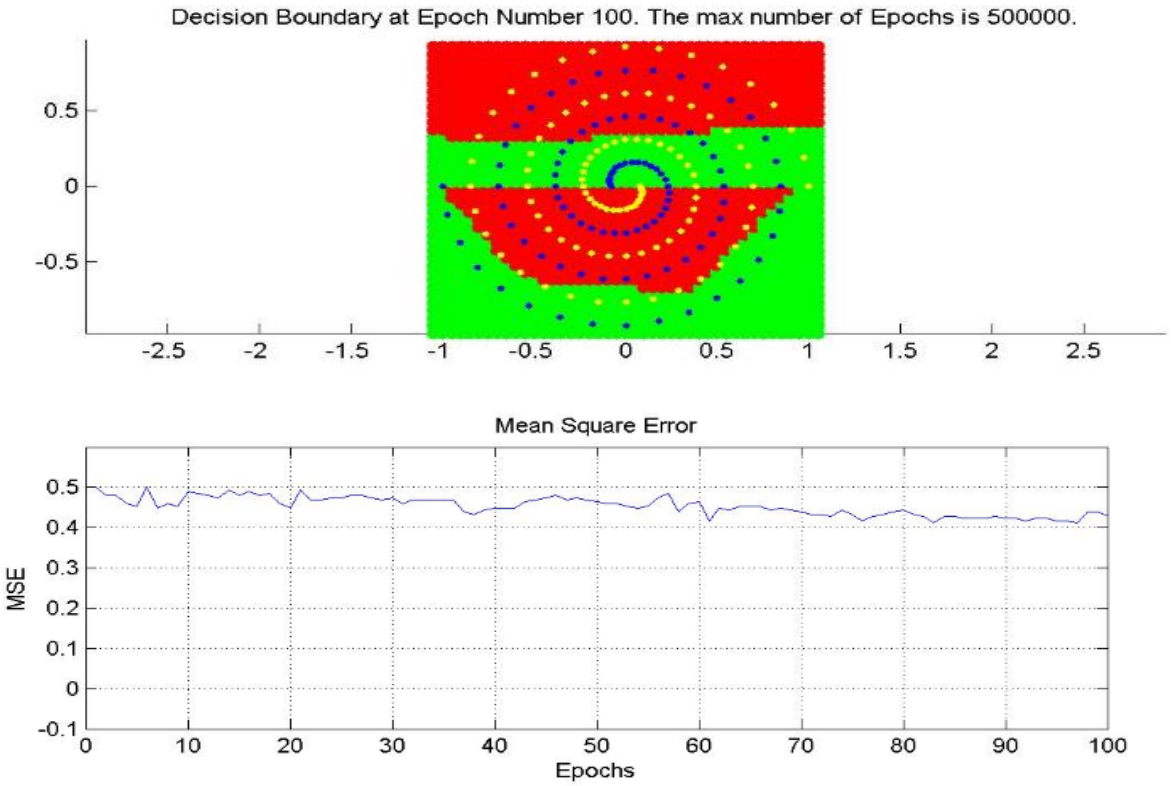


Figure 4. 1: Decision boundary and MSE

Fig 4.2 shows decision boundary and MSE, this training algorithm plot of the neural network 2-10-1 (2 neurons in the input layer, 1 hidden layer with ten neurons in it and one neuron in the output layer), and show as separate into two parts. As we see the MSE isn't stable thus the MSE goal didn't achieved.

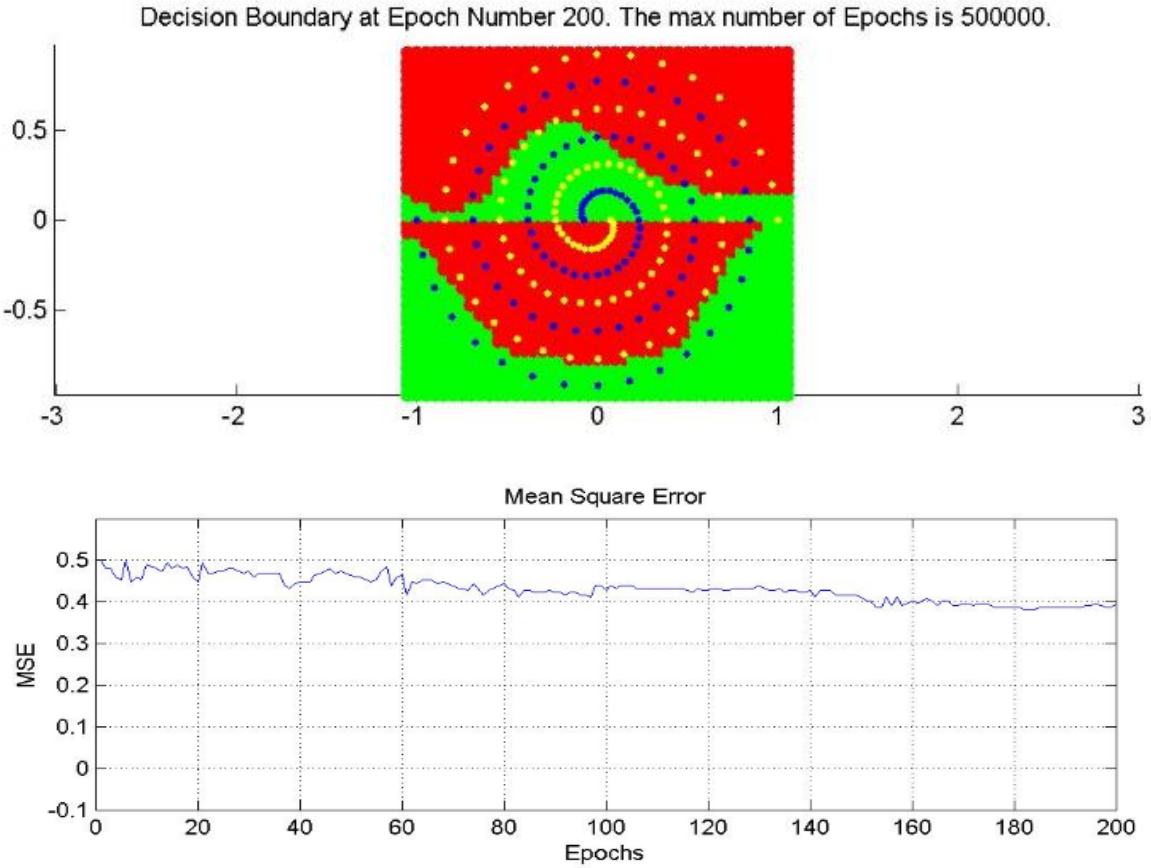


Figure 4. 2: Decision boundary and MSE

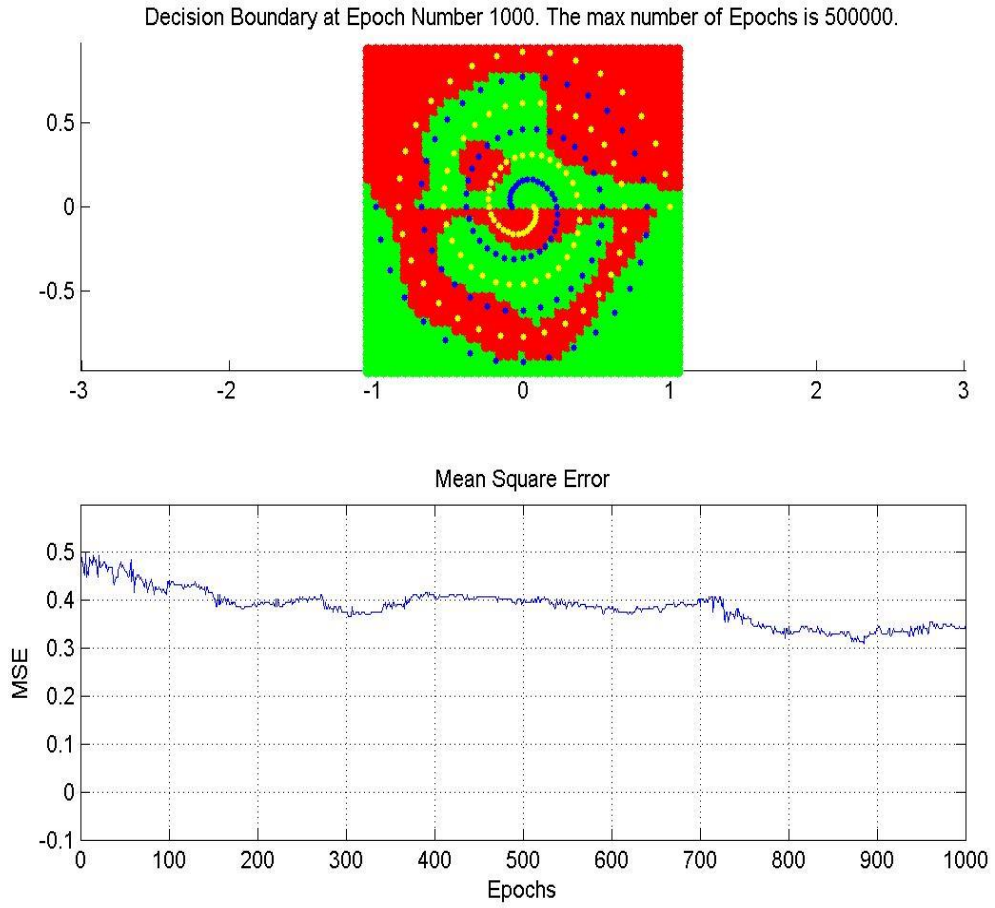


Figure 4. 3: Decision boundary and MSE

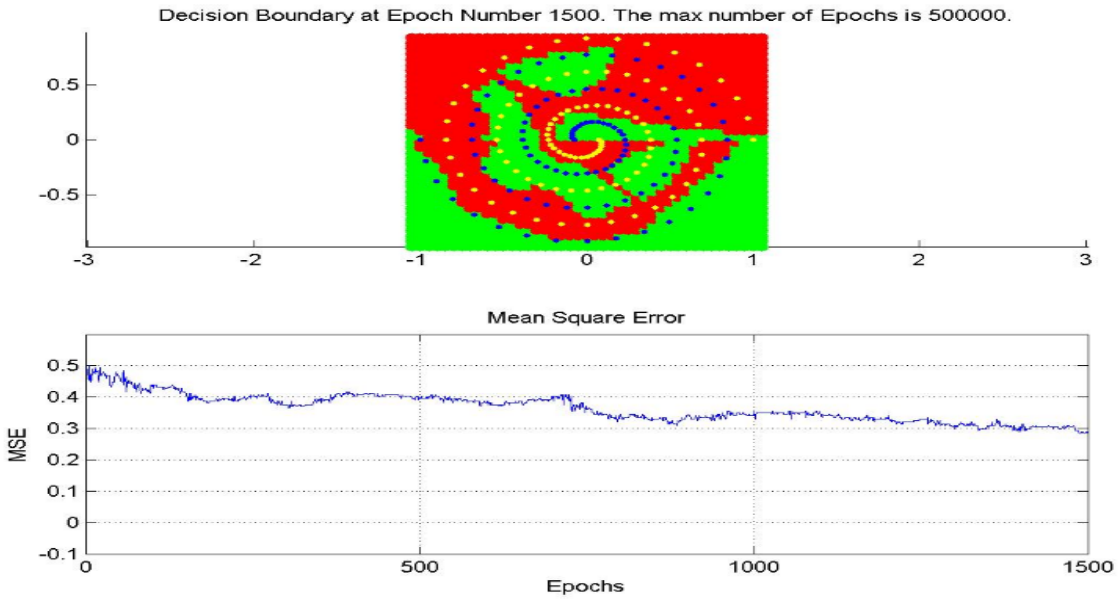


Figure 4. 4: Decision boundary and MSE

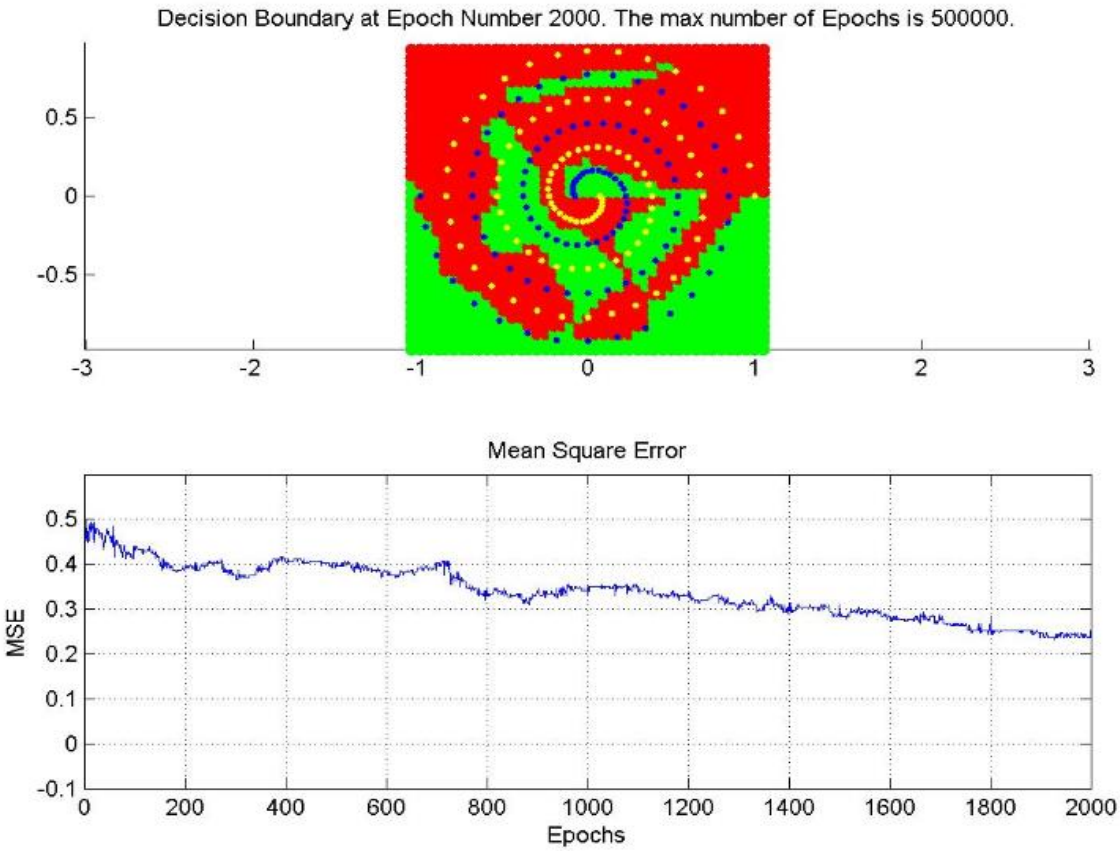


Figure 4. 5: Decision boundary and MSE

Figure 4.5 shows decision boundary and MSE, this training algorithm plot of the neural network 2-10-1 (2 neurons in the input layer, 1 hidden layer with ten neurons in it and one neuron in the output layer), and show as separate into two parts. As we see the MSE stable therefore the training phase is finish as we see in our Figure 4.5 the algorithm separate correctly the yellow point and blue point. Certain neural networks that achieved satisfactory performance are presented first along with their error performance plots.

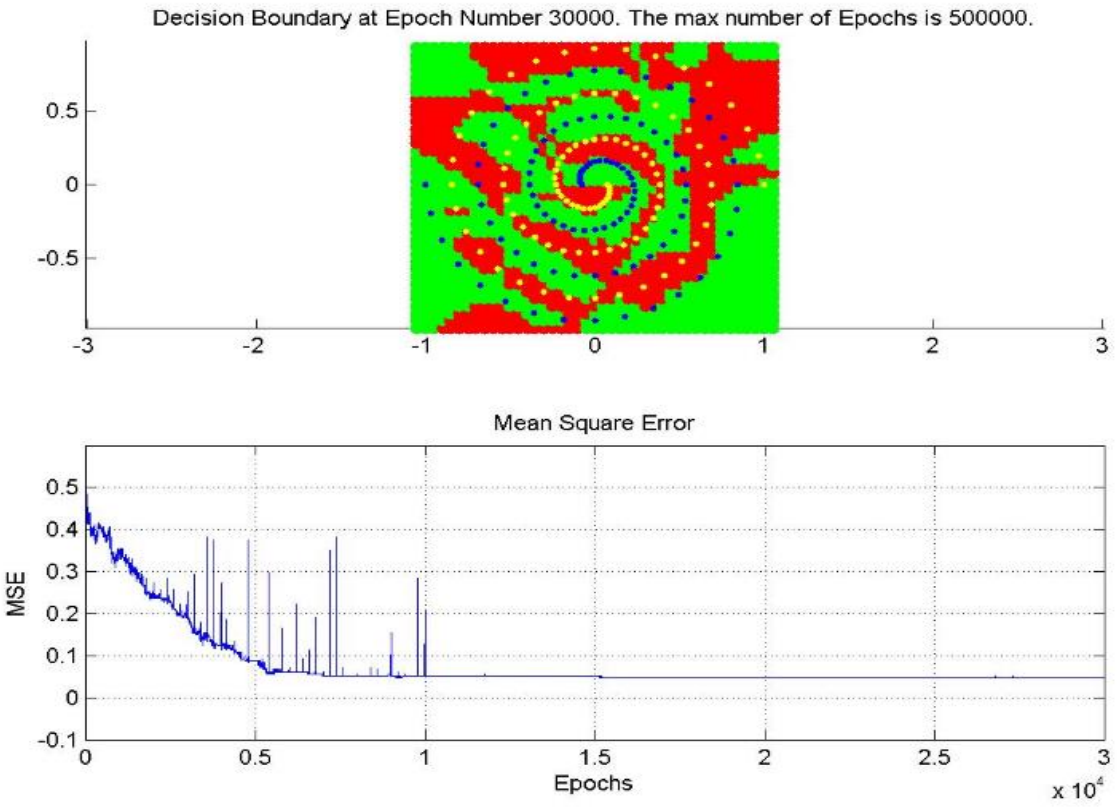
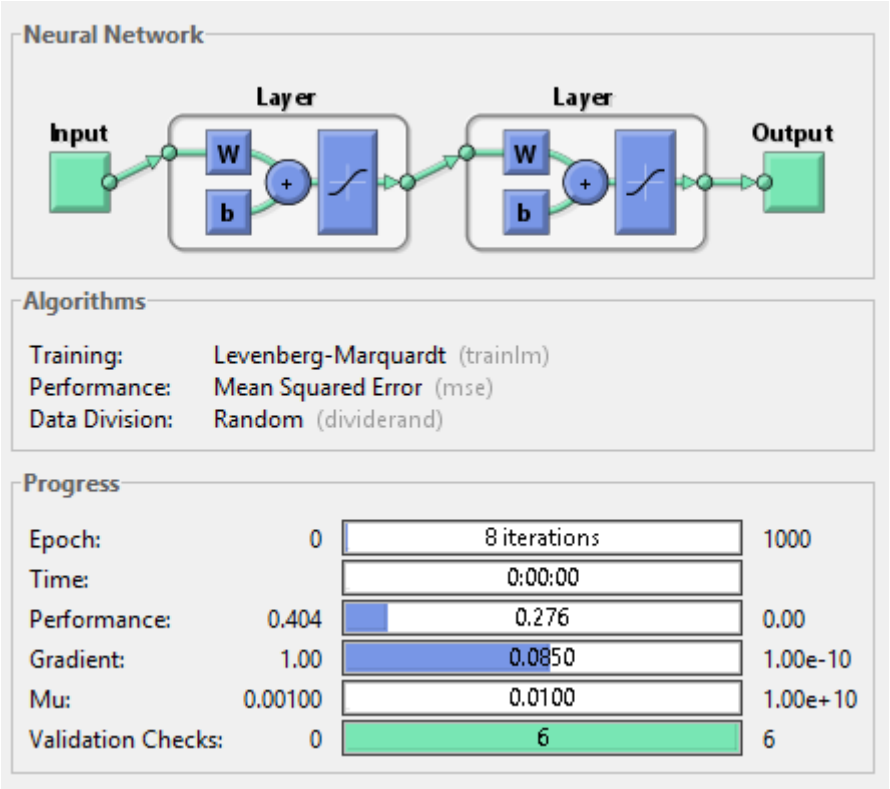


Figure 4. 6: Decision boundary and MSE

Figure 4.7 shows an overview of the chosen ANN and it can be seen that the training algorithm used is Levenberg-Marquardt algorithm. The performance function chosen for the training process is mean square error. Figure 5.7 plots the plots the best linear regression fit between the outputs and the targets and the correlation coefficient for the same has been found to be 0.65165 which is a decently good regression fit.



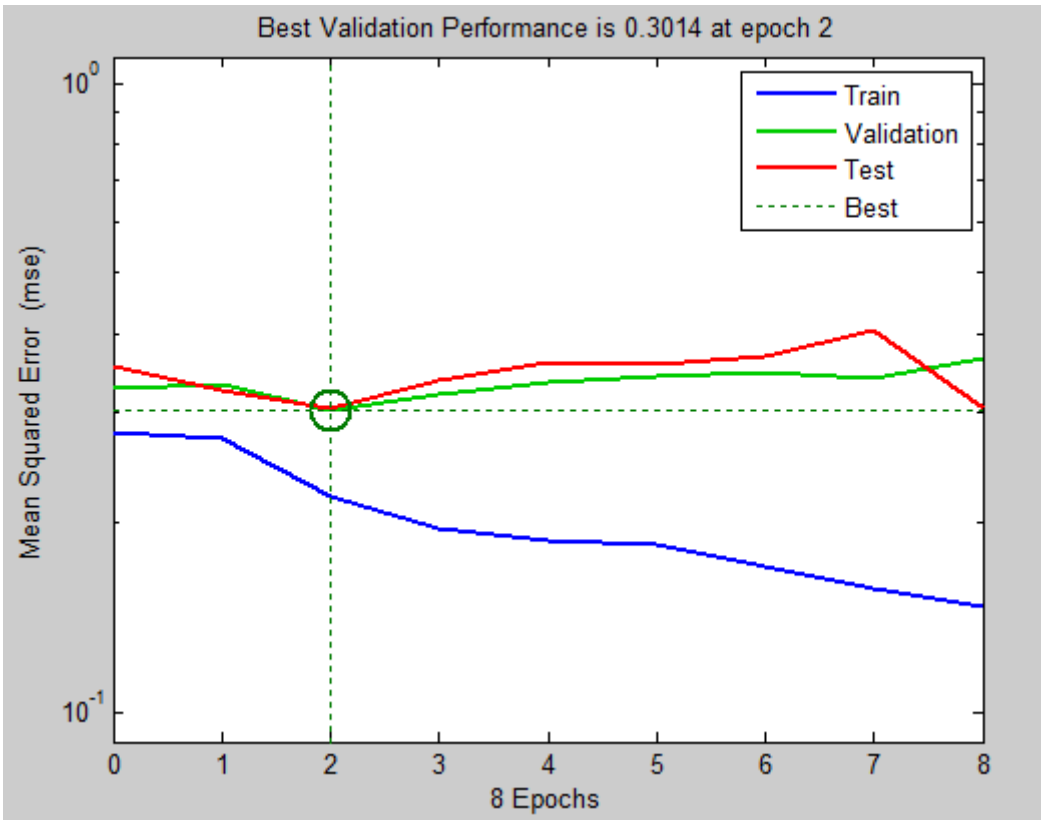


Figure 4. 7: MSE Graph

The third factor that is considered while evaluating the performance of the network is the correlation coefficient of each of the various phases of training, validation and testing. Figure 4.9 shows the regression plots of the various phases such as training, testing and validation. It can be seen that the best linear fit very closely matches the ideal case with an overall correlation coefficient of 0.34723.

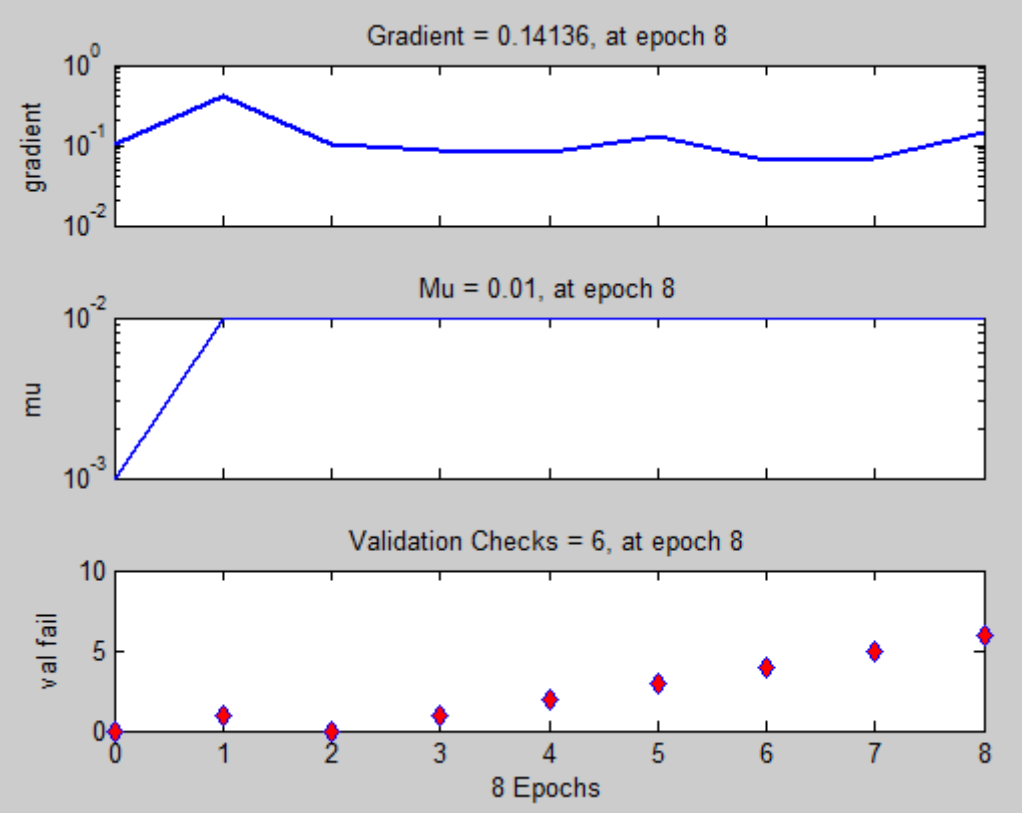
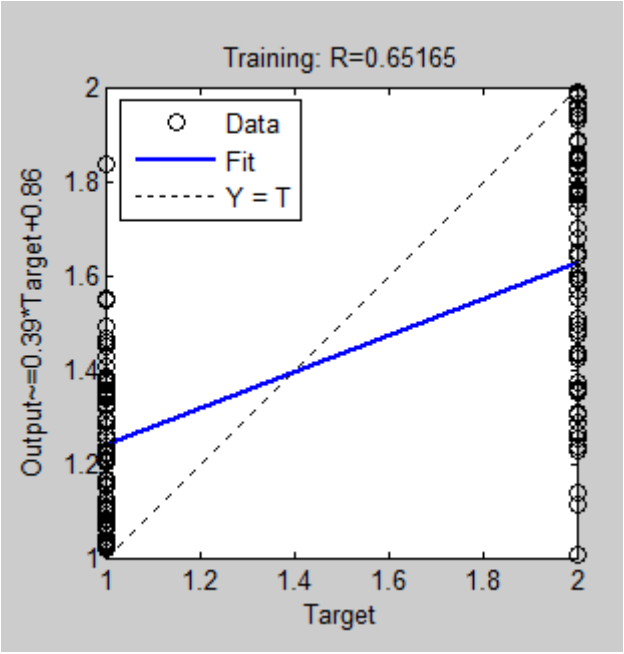


Figure 4. 8: Gradient and Validation plot



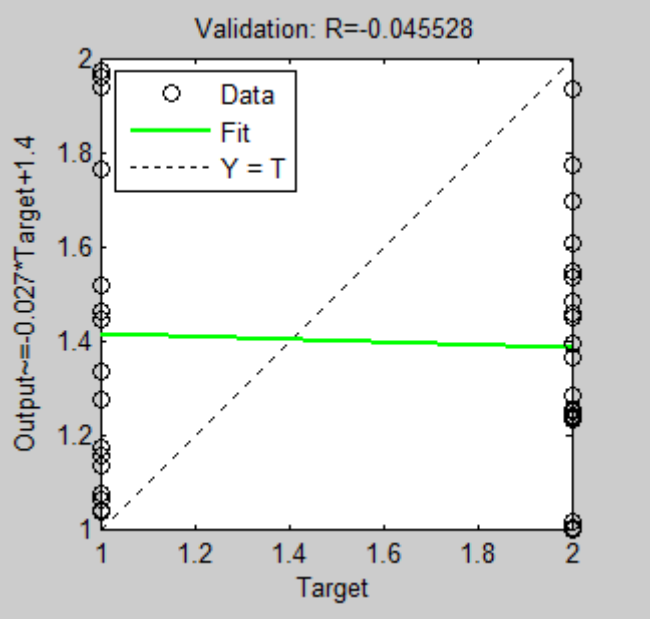


Figure 4. 9: Training and Validation

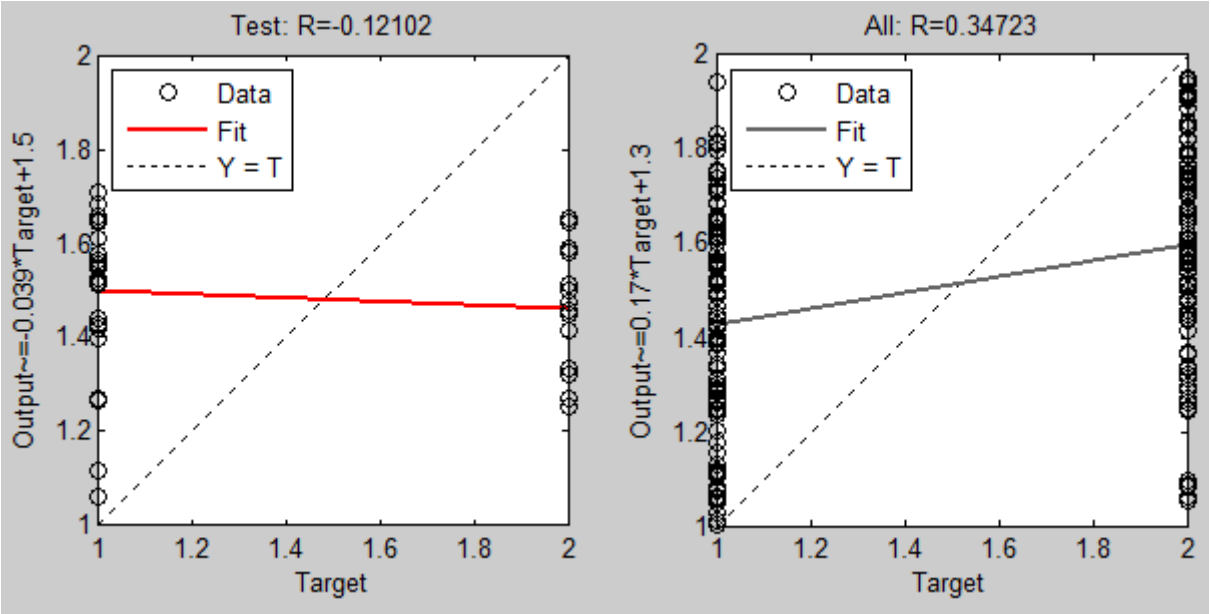


Figure 4. 10: Test graph and all

In Figure 4.12 and 4.13, we reinitialize the weight to get a good performance. It can be seen that the training algorithm is good if we compare it to Figure 4.7. The learning performance is high and in Figure 4.13 increased as shown.

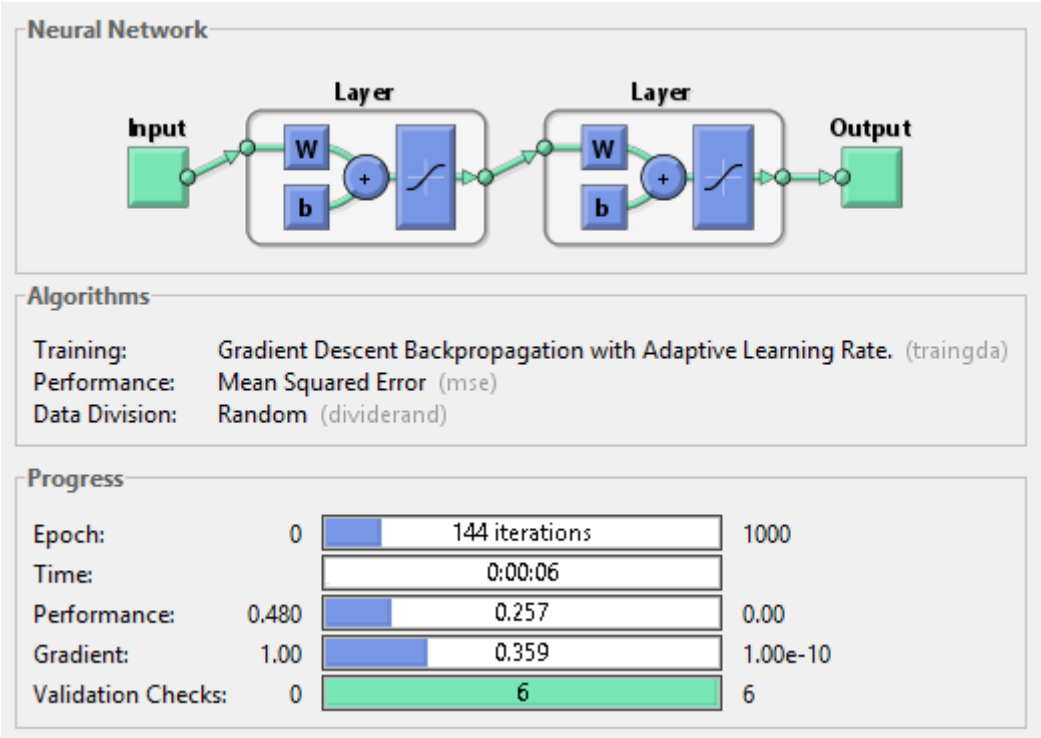


Figure 4. 11: Training with LM

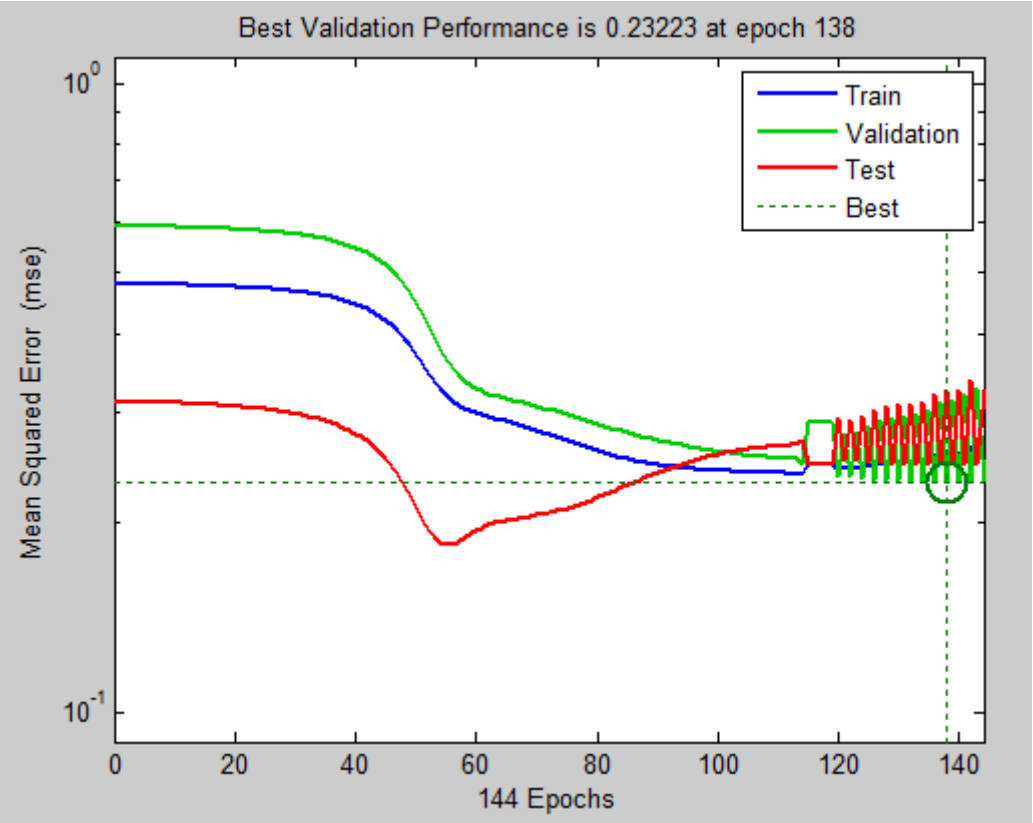


Figure 4. 12: MSE Graph

In Figure 4.17, we repeat the same experience but we change only the learning algorithm and it can be seen that the training algorithm used is Gradient Descent Backpropagation with Adaptive learning rate algorithm (traingda). The performance function chosen for the training process is mean square error and we get a good performance compared to LM algorithm (Previous Figure).

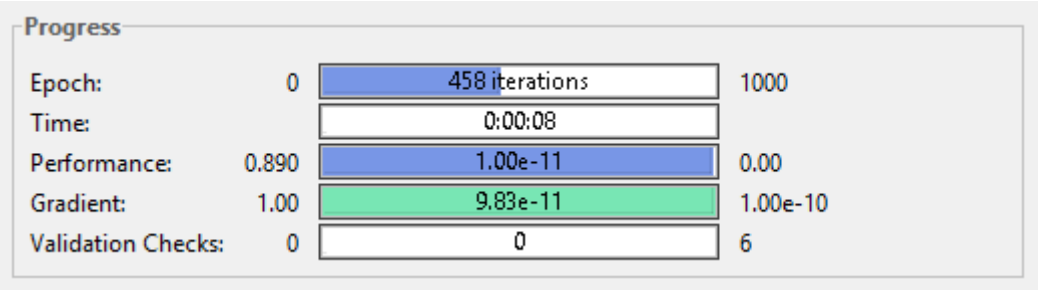


Figure 4. 13: Traingda Screenshot

Now that the neural network has been trained, the next important step is to analyze the performance of this network which is called testing. The methods and means by which this neural network has been tested are discussed here under. One important factor that helps test the network is the test phase performance plot as shown in Figure 4.17. It is to be noted that both the average as well as the maximum error percentages are in acceptable levels and hence the networks performance is satisfactory. Another means of determining the efficiency of a

trained neural network is to check the gradient and validation performance plot as shown in Figure 4.18. It can be seen that there is a steady decrease in the gradient and also that the maximum number of validation fails is 3 during 82 the training process. This indicates efficient training because the validation phase follows the test phase closely if the number of validation fails is low. This further implies that the neural network can generalize new data fed into it more effectively.

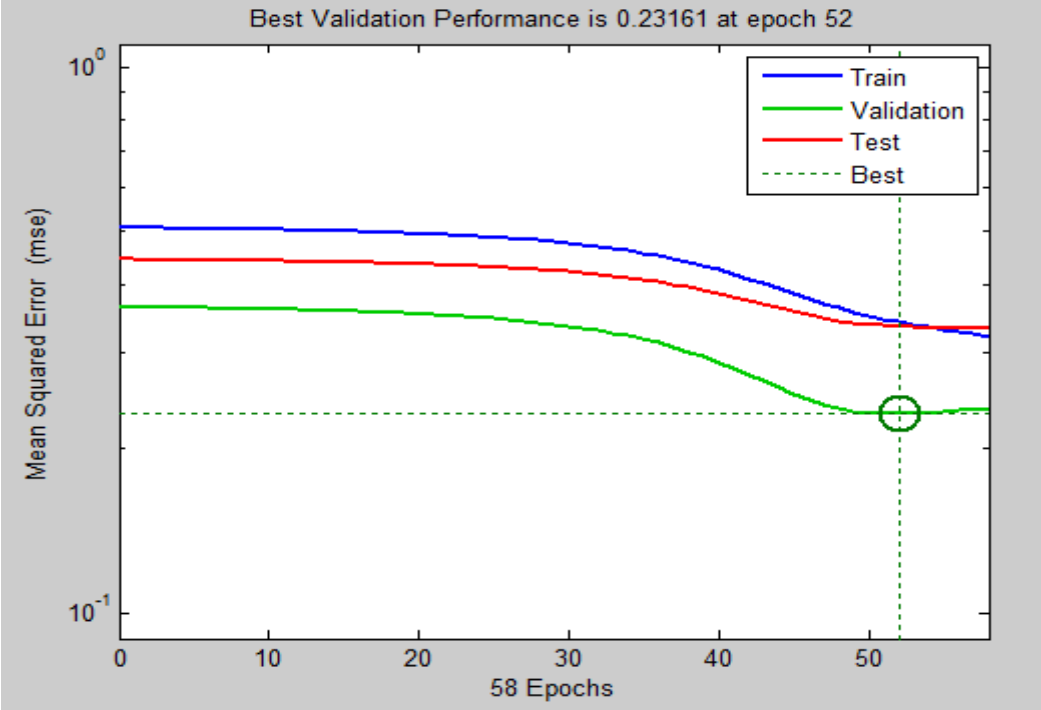


Figure 4. 14: Traingda Screenshot

The third factor that is considered while evaluating the performance of the network is the correlation coefficient of each of the various phases of training, validation and testing. Figure 4.19 shows the regression plots of the various phases such as training, testing and validation. It can be seen that the best linear fit very closely matches the ideal case with an overall correlation coefficient of 0.99329.

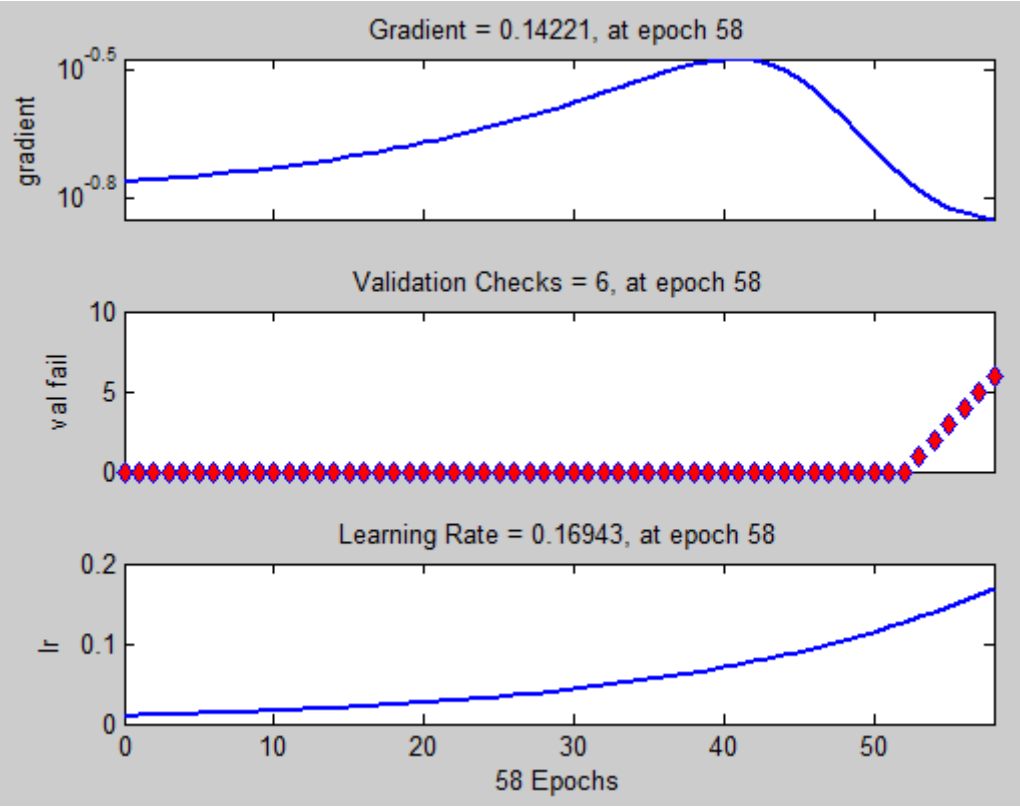


Figure 4. 15: Gradient and Validation plot

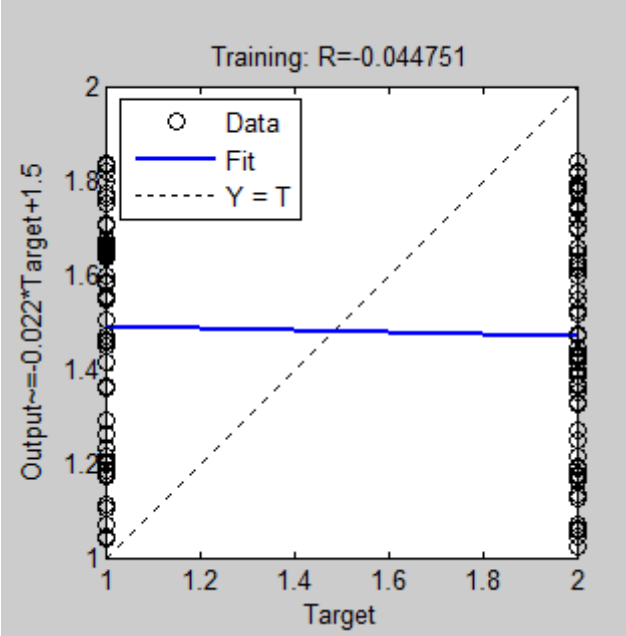


Figure 4. 16: Training Graph

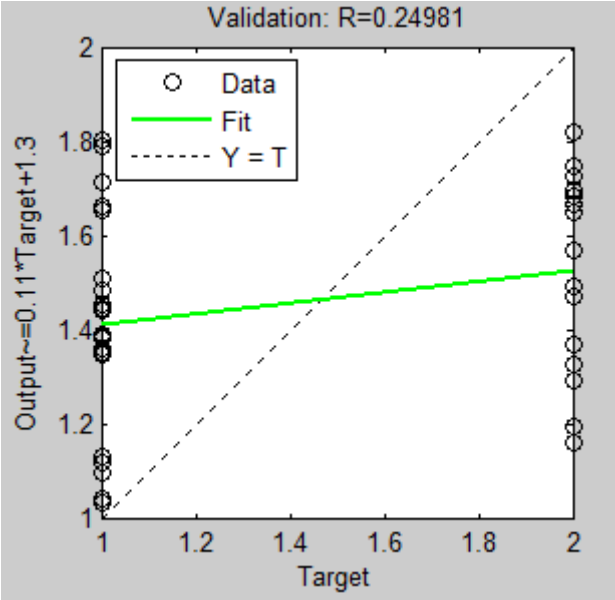


Figure 4. 17: Validation Graph

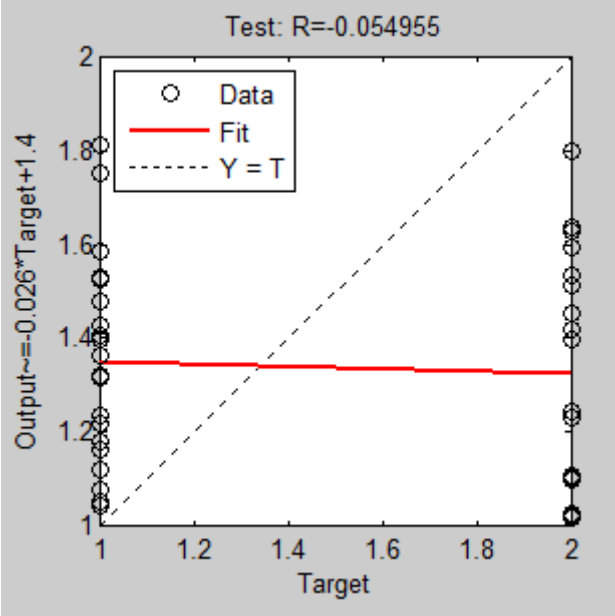


Figure 4. 18: Test Graph

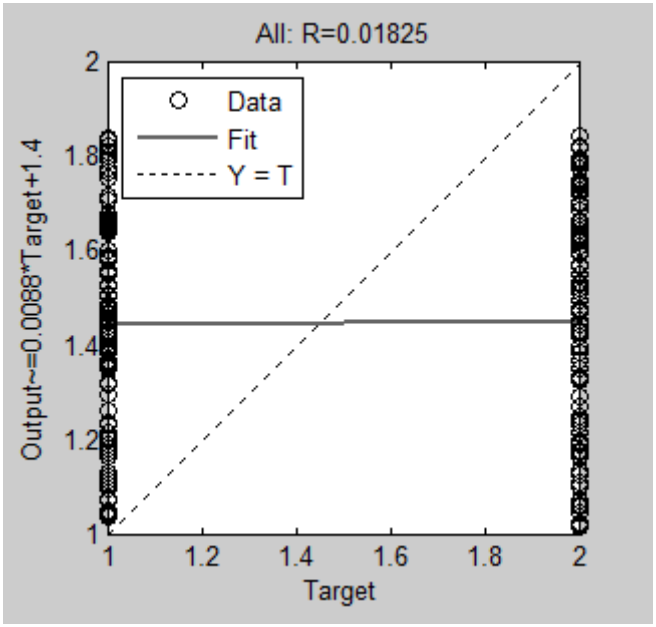


Figure 4. 19: Overall Graph

The third factor that is considered while evaluating the performance of the network is the correlation coefficient of each of the various phases of training, validation and testing. Figure 4.23 shows the regression plots of the various phases such as training, testing and validation. It can be seen that the best linear fit very closely matches the ideal case with an overall correlation coefficient of 0.99329.

Figure 4.24 and 4.25 represent the best performance obtained during the experience.

Figure 4. 20: Traingda Screenshot

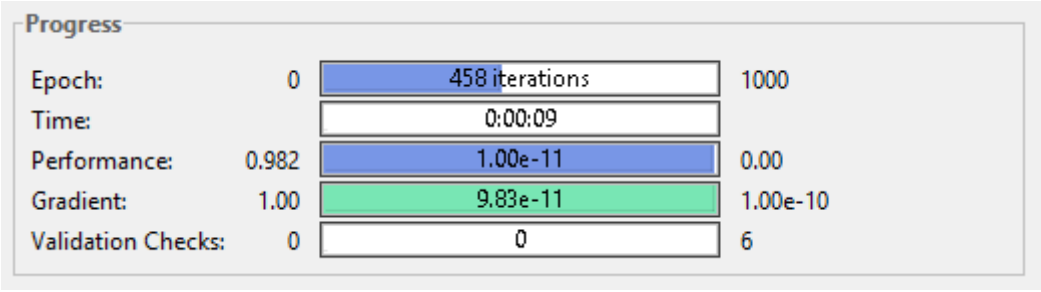


Figure 4. 21: Best Training Screenshot

Figure 4.26, 4.27 and 4.28 in this experience, we repeat the same experience, we change just the training algorithm and it can be seen that the training algorithm used is Resilient-propagation algorithm.

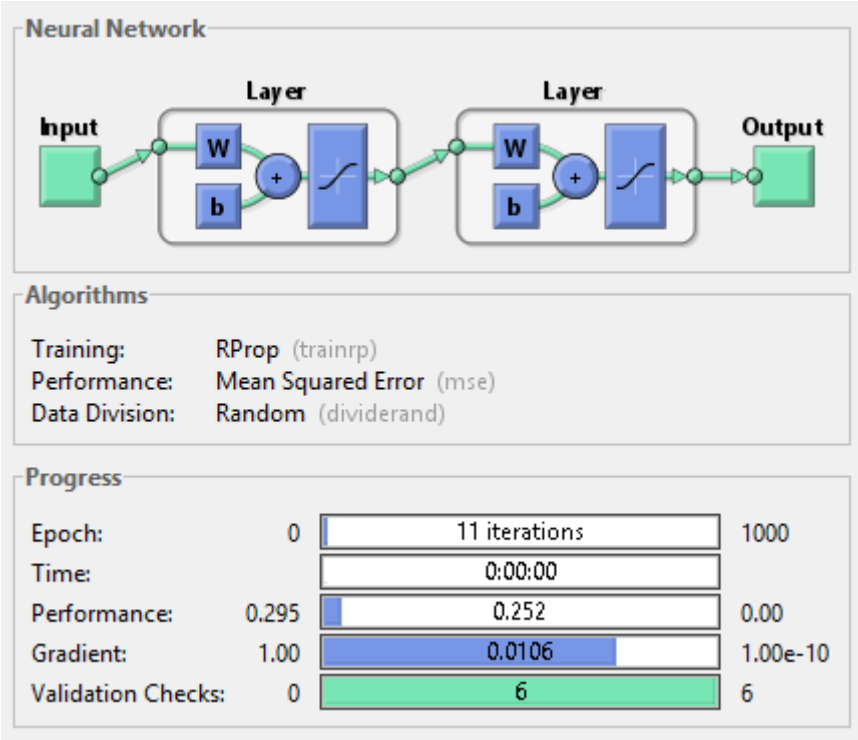


Figure 4. 22: Trainingrp Screenshot

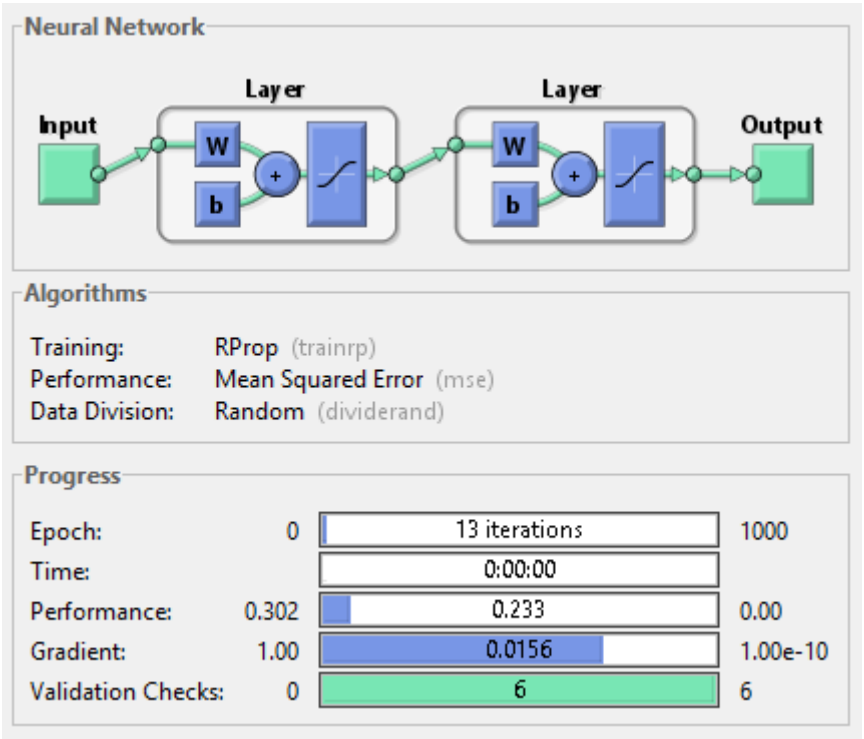


Figure 4. 23: Trainingrp Screenshot

As shown performance is very

Table 4. 1: Results of neural network for the three input case

Network	Training RMS	Training Performance	Number of hidden neurons in the network
MLP With traingda	.101	.98.2	10

Table 4. 2: Results of neural network for the 24 input case

Network	Training RMS	Training Performance	Number of hidden neurons in the network
MLP	0.00674	0.04419	25

RMS is root mean squared error. In Table 3.1, we get a good performance for 10 layer, when we put the biggest data in Table 3.2 for 25 layer our performance isn't like the previous performance, when the input is biggest we have to optimize the input data.

Once the neural network has been trained, its performance has been tested by three different factors. The first of these is by plotting the best linear regression that relates the targets to the outputs as shown in Figure 4.23.

The correlation coefficient (R) is a measure of how well the neural network's targets can track the variations in the outputs.

Figure 4.6 presents a snapshot of the trained ANN with the 3–10–1 configuration and it is to be noted that the number of iterations required for the training process were 191. It can be seen that the mean square error in fault detection achieved by the end of the training process was 9.43e-5 and that the number of validation check fails were zero by the end of the training process.

The classifiers based on neural networks have been extensively proposed and used in the past and almost all of these classifiers made use of multilayer perceptron neural network and employed the back-propagation learning strategy. Although backpropagation learning strategy is inherently slow in learning and poses difficulty in choosing the optimal size of the network, it is undoubtedly the ideal strategy to be employed when there is a large training set available because back-propagation algorithm can provide a very compact distributed representation of complex data sets.

It can be seen that there is a steady decrease in the gradient and also that the number of validation fails are 0 during the entire process which indicates smooth and efficient training.

The third factor that is considered while evaluating the performance of the network is the correlation coefficient of each of the various phases of training, validation and testing. Figure 4.23 and Figure 4.24 shows the regression plots of the various phases such as training, testing and validation. It can be seen that the best linear fit very closely matches the ideal case with an overall correlation coefficient of 0.99924.

Chapter five
Conclusion and recommendation

5.1. Conclusion

This thesis has studied the usage of neural networks as an alternative method for the detection, classification and diagnosis of faults on turnout. The methods employed make use data sensor collected and we extracted a signal processed to obtain the BA or learning matrix. For our case study, we chosen the peak ratio as inputs to the neural networks, then we took two class normal signal and the faulty signal for classification purpose. The class A represent normal signal and the class B represent faulty ones. As we saw in simulation part if there is a lot of input we have to optimize first the input, one of technique used is PCA (Principal component analysis) or genetic algorithm since this kinds of techniques reduce the noise and the dimension of the BA or learning matrix. All the neural networks investigated in this thesis belong to the back-propagation neural network architecture.

The simulation results obtained prove that satisfactory performance has been achieved by all of the proposed neural networks in general. As further illustrated, depending on the application of the neural network and the size of the training data set, the size of the ANN (the number of hidden layers and number of neurons per hidden layer) keeps varying.

The importance of choosing the most appropriate ANN configuration, in order to get the best performance from the network, has been stressed upon in this work. The choice of such rang in frequency (720 Hz-750 Hz) is reflected in our high learning performance and high accuracy. The sampling frequency adopted for sampling the current waveforms in this thesis is just between 720 Hz-750Hz which is very low compared to what has been used in the literature (a major portion of the works in literature utilized 2 kHz – 5 kHz). Our design showed a high classification rate for MLP tools in turnout fault diagnosis as shown with good test measurements and a good validation.

This has significant importance because, the lower the sampling frequency, the lesser the computational burden on the industrial PC that uses the neural networks. This means a lot of energy savings because a continuous online detection scheme of this kind consumes a large amount of energy, a major portion of which is due to the continuous sampling of waveforms. The above mentioned are some significant improvements that this thesis offers over existing neural network based techniques for turnout fault diagnosis.

To simulate the entire turnout line model and to obtain the training data set, MATLAB R2009a has been used. In order to train and analyze the performance of the neural networks, the Artificial Neural Networks Toolbox has been used extensively. Some important conclusions that drawn from this thesis are:

- Neural Networks are indeed a reliable and attractive scheme for an ideal turnout fault diagnosis scheme especially in view of the increasing complexity of the modern diagnosis technique.
- It is very essential to investigate and analyze the advantages of a particular neural network structure and learning algorithm before choosing it for an application because there should be a trade-off between the training characteristics and the performance factors of any neural network.
- Back Propagation neural networks are very efficient when a sufficiently large training data set is available and hence Back Propagation networks have been, chosen for all the three steps in the fault location process namely fault detection, classification and fault diagnosis.

5.2. Recommendation

Further studies will be carried out to combine pattern recognition and model-based approaches in order to improve the effectiveness of the final results, especially in the case of multiple defects. Further studies will be carried out to combine pattern recognition and model-based approaches in order to improve the effectiveness of the final results, especially in the case of multiple defects. The assessment of the severity of the defects is also an interesting subject to consider, particularly in a condition-based maintenance context. Further studies are being carried out to handle multiple faults and assess their severity, which may be useful in a predictive maintenance context. The functionality of the system will be extended to discriminate between benign faults, faults that need to be monitored and very serious faults that require immediate maintenance action.

Although the method has been developed in the context of railway track circuit diagnosis, we believe that it can be transferred to other application domains involving the diagnosis of large scale systems composed or linearly organized subsystems, such as other infrastructure networks.

References

[1] R. a. K. J. Bland, "A Practical Application of a New Method for Condition Based Maintenance," *Maintenance Management International,* pp. 31-35, 1987.

[2] R. Mobley, An Introduction to Predictive Maintenance, New York: Van Nostrand Reinhold, 1990.

[3] T. Kohonen, Self-Organizing Maps, 2nd.Ed., Springer, 1997.

[4] A. Negash, Intelligent control system, Addis Ababa: University Addis Ababa, 2016.

[5] M. A. T. a. H. R. K. Jafar Zarei, "Vibration analysis for bearing fault detection and classification using an intelligent filter," *Mechatronics,* vol. 2, no. 24, p. 151–157, 2014.

[6] A. a. D. M. Bemieri, "A Neural Network Approach for Identification and Fault Diagnosis on Dynamic Systems," *IEEE Transactions on Instrumentation and Measurement,* pp. 867-873, 1994.

[7] A. R. M. K. Mayssa Hajar, "Bearing and Gear Fault Detection Using Artificial Neural," Lebanese University.

[8] R. Isermann, Fault-Diagnosis Systems, Berlin: Springer, 2006.

[9] S. G. M. D. Siril Yella, "Condition monitoring of wooden railway sleepers," *Proceedings of the 2009 IEEE International Conference on Systems,* October 2009.

[10] A. D. T. D. P. A. Latifa Oukhellou, "Fault diagnosis in railway track circuits using Dempster-Shafer," Université Paris XII, Paris, France, 2014.

[11] A. Yokoyama, "Innovative changes for maintenance of railway by using ICT," East Japan Railway Co, Tokyo, Japan, 2015.

[12] R. R. S. N. K. a. K. Y. V. Venkatasubramanian, "A review of process fault detection and diagnosis Part III, Process history based methods," *Computers and Chemical Engineering,* no. 27, pp. pp. 327-346, 2003.

[13] K. S. K. H. G. K. a. S. H. C. Seong Soo Choi, "Development of a non-line fuzzy expert system for integrated alarm processing in nuclear power plants," *IEEE Transactions on Nuclear Science,* no. 42, p. 1406–1418, 1995.

[14] C.-H. H. B.-C. Z. D. l. X. a. Y. w. C. Zhi-Jie Zhou, "Hidden behavior prediction of complex systems based on hybrid information," *IEEE Transactions on Cybernetics,* no. 43(2), p. 402–411, 2013.

[15] T. R. a. J. M. P. Tomasz Bujlow, "A method for classification of network traffic based on c5.0 machine learning algorithm," *In Computing, Networking and Communications (ICNC),* no. 12, p. 237–241, 2012.

[16] p. ,. Kuanfang He and Xuejun Li. ., "A quantitative estimation technique for welding quality using local mean decomposition and support vector machine," *Journal of Intelligent Manufacturing,* p. 1–9, 2014.

[17] J.-S. J. Anfis, "adaptive-network-based fuzzy inference system," *IEEE Transactions on Man and Cybernetics Systems,* no. 23, p. 665–685, May 1993.

[18] L. R. a. B.-h. H. Juang, "An introduction to hidden markov models," *IEEE Acoustics, Speech, and Signal Processing (ASSP) Magazine,* no. 3, p. 4–16, 1986.

[19] E. B. a. M. Kingsley, "A review of time synchronous average al- gorithms," *In Annual conference of the prognostics and health management society,* vol. 23, p. 24, 2009.

[20] K. L. Butler, "An expert system based framework for an incipient failure detection and predictive maintenance system," *International Conference onIntelligent Systems Applications to Power Systems (ISAP),* p. 321–326, 1996.

[21] K. B. a. A. W. Chris Van Hoof, "The best materials for tiny," *Science clever sensors,* vol. 306(5698), p. 986–987, 2004.

[22] A. B. D. K. W. S. a. R. S. Jürgen Ackermann, "Robust control: the parameter space approach," *Springer Science & Business Media,* 1993.

[23] Z. H. a. Y. Z. Yaguo Lei, "A new approach to intelligent fault diagnosis of rotating machinery. Expert Systems with Applications," vol. 23, no. 4, p. 1593– 1600, 2008.

[24] P.-E. D. E. d. mapping, "Euclidean distance mapping," *Computer Graphics and image processing,* p. 227–248, 1980.

[25] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, p. 674–693, Jul 1989.

[26] Z. H. a. G. W. Shu-Guang He, "Online monitoring and fault identification of mean shifts in bivariate processes using decision tree learning techniques," *Journal of Intelligent Manufacturing,* vol. 24(1), p. 25–34, 2013.

[27] M. S.-M. a. E. D. Houari Toubakh, "Advanced pattern recognition approach for fault diagnosis of wind turbines," *In International Conference on Machine Learning and Applications (ICMLA),* vol. volume 2, p. 368– 373, 2013.

[28] C. h. P. M. X. L. a. H. L. C. Bin Hu, "A two stage equipment predictive maintenance framework for high-performance manufacturing systems," *In Industrial Electronics and Applications (ICIEA),* p. 1343– 1348, 2012.

[29] S. E. S. a. D. A. Dickey, "Testing for unit roots in autoregressive moving average models of unknown order," *Biometrika,* vol. 3, no. 71, p. 599–607, 1984.

[30] T. P. B. a. S. Das, "Multisensor data fusion using support vector machine for motor fault detection," *Information Sciences,* p. 96–107, 2012.

[31] J. S. R. a. J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy. American," *Journal of Physiology- Heart and Circulatory Physiology,* vol. 6, no. 276, pp. 76-79, 2000.

[32] A. W. a. L. M. Parikshit Mehta, "Condition based maintenance systems integration and intelligence using bayesian classification and sensor fusion," *Journal of Intelligent Manufacturing,* vol. 1, p. 16, 2013.

[33] M. D. a. D. He, "Hidden semi-markov model-based methodology for multi-sensor equipment health diagnosis and prognosis," *European Journal of Operational Research,* vol. 3, p. 178, 2007.

[34] Y. W. a. K. W. Zhenyou Zhang, "Fault diagnosis and prognosis using wavelet packet decomposition, fourier transform and artificial neural net- work," *Journal of Intelligent Manufacturing,* vol. 6, no. 24, p. 1213–1227, 2013.

[35] T. P. B. a. S. Das, "Multisensor data fusion using support vector machine for motor fault detection," *Information Sciences,* no. 217, p. 96–107, 2012.

[36] Q. L. a. M. Dong, "Online health management for complex nonlinear systems based on hidden semi-markov model using sequential monte carlo methods," *Mathematical Problems in Engineering,* 2012.

[37] K. M. N. Z. a. G.-. a. T. Diego Alejandro Tobon-Mejia, "A data-driven failure prognostics method based on mixture of gaus- sians hidden markov models," *IEEE Transactions on Reliability*, vol. 2, no. 61, p. 491–503, 2012.

[38] K. M. N. Z. a. G.-. a. T. Diego Alejandro Tobon-Mejia, "A data-driven failure prognostics method based on mixture of gaussians hidden markov models," *IEEE Transactions on Reliability*, vol. 2, no. 61, p. 491–503, 2012.

[39] Y. Z. a. Z. Z. Chengdong Wang, "Fault diagnosis for diesel valve trains based on time frequency images," *Mechanical Systems and Signal Processing*, vol. 8, no. 22, p. 1981–1993, 2008.

[40] J. D. W. a. S. Y. Liao, "A self-adaptive dat aanalysis for fault diagnosis of an automotive air-conditioner blower," *Expert Systems with Applications*, vol. 1, p. 545– 552, 2011.

[41] J. D. W. a. C. H. Liu, "An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network," *Expert systems with applications*, vol. 3, no. 36, p. 4278–4286, 2009.

[42] R. Isermann, «Model-Based Fault Detection And Diagnosis-Status And,» *Annual Reviews in Control*, pp. Vol. 29, Is. 1, 2005.

[43] C. R. M. J. Fuente, A Comparative Study of Neural Networks Based, Kingston Upon Hull, UK, 1997.

[44] C. R. C. H. K. S. L. a. K. W. M. M. A. Hussain, "Application of Artificial Intelligence Technique in Process Fault Diagnisis," *Journal of Engineering Science and Technology*, pp. 260-270, 2007.

[45] M. a. M. K. M.R.Othman, "Process Fault Detection Using Hierarchical Artificial Neural Network Diagnostic Strategy," *Jurnal Teknologi*, p. 11–26, 46(F) Jun 2007.

[46] A. D. T. D. P. A. Latifa Oukhellou, "Fault diagnosis in railway track circuits using Dempster-Shafer classifier fusion," Paris XII University, Paris XII.

[47] a. L. P. Becraft. W.R., "An Integrated Neural Network/Expert System Approach for Fault Diagnosis," 1993.

[48] a. K. G. Javadpour R., "A Fuzzy Neural Network Approach to Machine Condition Monitoring," in *Proceedings o f the 25& International Conference on Computers & Industrial Engineering*, 29-31 March 1999.

[49] G. W. H. Knapp, "Machine Fault Classification: A Neural Network," *Approach, International Journal o f Production Research*, pp. 811-823, 1992.

[50] S. a. P. S. Mitra, "Fuzzy Multi-Layer Perceptron, Inferencing and Rule Generation," *IEEE Transactions on Neural Networks, ,* pp. 51-63, 1995.

[51] J. a. S. H. Luxhoj, "Comparison of Proportional Hazards Models and Neural Networks for Reliability Estimation," *Journal o f Intelligent Manufacturing*, no. 8, pp. 227-234, 1997.

[52] J. a. K. B. ,. Lee, "Analysis of Machine Degradation Using a Neural Network Based Pattern Discrimination Model," *Journal o f Manufacturing Systems*, no. 12:5, pp. 379-387, 1993.

[53] J. a. K. G. Nelson, "Using CMAC Neural Networks and Optimal Control," *IEEE*, pp. 2386-2390, 1995.

[54] G. G. S. a. R. J. Carpenter, "ARTMAP: Supervised Real Time Learning and Classification of Nonstationary Data by a Self-Organizing Network," Technical Report CAS/CNS-TR-91-001, Boston, February,1991.

[55] J. a. H. Y. Buckley, "Hybrid Neural Nets Can Be Fuzzy Controllers and Fuzzy Expert Systems,," *Fuzzy Sets and Systems*, no. 60, pp. 135-142, 1993.

[56] S. a. H. R. Marriott, "A Modified Fuzzy ARTMAP Architecture for the Approximation of Noisy Mappings," *Neural Networks*, no. 8:4, pp. 619-641, 1995.

[57] Y. a. S. A. Wong, Learning Convergence in the Cerebellar Model, 1992.

[58] G. J. R. a. W. H. Knapp, "An ARTMAP Neural Network Based Machine Condition Monitoring System," *Accepted to be published in Journal*, 2000.

[59] L. M. M. a. R. L. Vena, "Neural Network Architectures for Industrial Applications," *Biosensors & Bioelectronics*, no. 10, pp. 231-236, 1995.

[60] T. a. T. W. Lee, "Real-Time Parallel Adaptive Neural Network Control For Nonlinear Servomechanisms - An Approach Using Direct Adaptive Techniques," *Mechauonics*, no. 3:6, pp. 705-725, 1993.

[61] A. a. R. S. Farell, "Framework for Enhancing Fault Diagnosis Capabilities of Artificial Neural Networks," *Computers and Chemical Engineering*, no. 18:7, pp. 613-635, 1994.

[62] H. B. Z. P. J. a. S. P. Chung, "Incipient Multiple Fault Diagnosis in Real-Time with Application to Large-scale Systems," *IEEE Transactions on Nuclear Science*, no. 41:4, pp. 1692-1703, 1994.

[63] C. a. C. C. Lin, "Learning Convergence of CMAC Technique," *IEEE Transactions on Neural Networks*, no. 8:6, pp. 1281-1292, 1997.

[64] C. M. bishop, Neural Networks for pattern recognition, Birmingham, UK: C L A R E N D O N PRESS • O X F O R D, 1995.

Appendix A: Signal collected by the sensor

Normal signal	Abnormal signal
0.0792274183976261	-0.277601636726547
0.00716225519287834	-0.0443447904191620
-0.0402373887240356	0.117603033932136
-0.0117090801186944	-0.145054570858283
-0.0105421958456973	-0.111430499001996
-0.0302182789317507	0.130922714570858
0.00619655786350148	0.0328118962075848
0.0245045697329377	-0.197033812375250
0.0352479525222552	-0.0748825948103793
0.0462327596439169	0.00958367265469062
0.0111859940652819	-0.128161317365269
-0.0168594658753709	0.178678642714571
-0.0210441543026706	0.392280838323353
0.0246252818991098	0.0484056686626747
0.0772155489614243	0.0604258682634731
0.0908157863501484	0.116953293413174
0.0903329376854599	-0.0735831137724551
0.0832913946587537	0.0531162874251497
0.0532340652818991	0.234556327345309
0.0215672403560831	0.176729421157685
-0.0306608902077151	-0.0800805189620758
-0.0651845697329377	0.00568522954091816
-0.0520671810089021	0.124100439121756
-0.0969721068249258	-0.341925948103792
-0.0764912759643917	-0.655263313373254
-0.0202796439169139	-0.185663353293413
-0.0401166765578635	0.296444111776447
-0.0315863501483680	0.114354331337325

-0.00744391691394659	0.0305378043912176
-0.0154511572700297	0.0575020359281437
0.0519464688427300	0.0328118962075848
0.127109910979229	0.183876566866267
0.113107299703264	0.0266393612774451
0.161714065281899	-0.273053453093812
0.130811750741840	0.00568522954091816
0.0344432047477745	0.463102554890220
-0.00321899109792285	0.377824111776447
-0.0570566172106825	0.163247305389222
-0.0539583382789318	0.00828419161676647
0.0169399406528190	-0.0644867465069860
0.0736344213649852	0.0319997205588822
0.101921305637982	-0.107856926147705
0.0758877151335312	-0.223510738522954
-0.0516245697329377	-0.0919382834331337
-0.116125103857567	-0.0251774451097804
-0.103490563798220	-0.0540908982035928
-0.104737922848665	0.0144567265469062
-0.0661502670623145	0.0605883033932136
0.0142037982195846	-0.247876007984032
0.0449854005934718	-0.318535289421158
0.0172216023738872	0.127024271457086
-0.00144854599406528	0.198495728542914
-0.0863494362017804	0.00227409181636727
-0.0945578635014837	0.150577365269461
0.00152902077151335	0.237480159680639
0.0644602967359051	0.0281012774451098
0.162518813056380	0.0198170858283433
0.177809020771513	0.0583142115768463
0.0567347181008902	-0.250799840319361

-0.0200382195845697	-0.224647784431138
-0.105100059347181	0.260870818363273
-0.204687596439169	0.190211536926148
-0.158334124629080	-0.266718483033932
-0.103731988130564	-0.0882022754491018
0.00667940652818991	0.0841413972055888
0.173946231454006	-0.231470059880240
0.178694243323442	-0.221074211576846
0.0876370326409496	-0.0259896207584830
-0.0424906824925816	-0.0194922155688623
-0.129886290801187	0.0313499800399200
-0.0705763798219585	0.249987664670659
0.0183482492581602	0.302941516966068
0.00675988130563798	0.0596136926147705
0.0642591097922849	-0.0862530538922156
0.134714777448071	-0.0609131736526946
0.0973342433234422	-0.0461315768463074
0.0542400000000000	-0.0729333732534930
-0.0585454005934718	-0.0334616367265469
-0.0736344213649852	0.0711465868263473
0.0189115727002967	0.100709780439122
0.0210039169139466	0.0190049101796407
0.0223719881305638	0.00227409181636727
0.0217281899109792	0.0230657884231537
-0.0281259347181009	-0.169419840319361
-0.0491700890207715	-0.202881477045908
-0.0844582789317508	0.0713090219560878
-0.137008308605341	0.119227385229541
-0.0811588130563798	-0.134658722554890
0.0127150148367953	-0.130110538922156