CRANFIELD UNIVERSITY

YIT WEY TAM

GAS TURBINE GAS PATH DIAGNOSTICS USING ARTIFICIAL NEURAL NETWORK

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Thermal Power

MSc
Academic Year: 2021–2022

Supervisor: Dr. Yiguang Li
August 2022

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Thermal Power

MSc

Academic Year: 2021–2022

YIT WEY TAM

Gas Turbine Gas Path Diagnostics using Artificial Neural Network

Supervisor: Dr. Yiguang Li
August 2022

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

Abstract

A gas turbine is a deeply complex rotating machinery which is used to generate thrust or power depending on its application. A gas turbine comprises of three main components, the compressor, turbine and combustor. Every component are meticulously designed and manufactured to achieve high overall efficiency therefore any unwanted and unexpected alterations in the gas path will cause the performance of the gas turbine to deteriorate. The alterations to the gas path can be caused by the various different type fo degradation a gas turbine may experience throughout its lifetime. If the degradation is left unnoticed and untreated, the degradation may built up overtime and eventually cause the gas turbine to break down. This would result in loss of revenue for the business as they are unable to operate. The downtime of the gas turbine would also be long since the necessary manpower and repair parts needed to repair the gas turbine would not be ready on site when the engine breaks down since it was an unexpected breakdown. In order to prevent this scenario from happening, an effective maintenance strategy would need to be in place.

Various maintenance strategies has been developed and used in the past, however the go to maintenance strategy for modern gas turbine is predictive maintenance. The development of predictive maintenance strategy was enabled due to the advancements made in technology and computing power in the last few decades. Predictive maintenance uses measurements obtained using the sensors installed in the gas turbine to monitor the health of the engine. This allows the operator to know the health of the engine at all times therefore they can schedule for maintenance more efficiently to reduce engine downtime. One of the most important component that is essential for predictive maintenance is the diagnostic system.

The diagnostic system is responsible for taking in the measurement data from the sensors and provide a diagnosis of the gas turbine. In order to use predictive maintenance strategy, the diagnostic system used must be robust and capable of providing accurate diagnosis. Various different diagnostic systems have been developed and used in the past such as expert systems, gas path analysis, genetic algorithms and etc. With computing power increasing at a rapid pace, artificial neural networks became widely used for gas turbine diagnostics. Artificial neural networks are mathematical models which tries to imitate how the human brain works. Once trained, the network will be able to carry out the task it is trained on very quickly. Artificial neural networks are also capable of handling noisy and bias data therefore making it compatible for gas turbine diagnostics since the sensors installed on board the gas turbine are exposed to high temperature and pressure operating environment making it susceptible to picking up noisy measurements.

An artificial neural network based diagnostic system for the Trent-900 turbofan engine was developed in this thesis. The artificial neural network was developed using the TensorFlow library which is an open sourced machine learning and artificial intelligence library developed by Google. The network developed has the capability of detecting, isolating and quantifying a single component degradation along the gas path of a Trent-900 engine. The result obtained from the network shows that it is capable of accurately detecting and isolating the degraded component with an average accuracy of 99.4%. As for quantifying the degradation level for the components along the gas path, the networks used to quantify the degradation level for the fan and intermediate compressor needs to be improved further as the degradation level estimated by the networks have a significant difference from the target value. As for the rest of the components, the network used were able to estimate the degradation level close to the target value with only minute differences between the estimate value and the target value. In the process of developing the neural network, various hyperparameter optimisation methods were tested and Bayesian optimisation seems to be the method that is most capable of producing the optimal combination of hyperparameters. Lastly, a binary classification neural network was trained using genetic algorithm. This method showed some potential as the network achieved was able to achieve a prediction accuracy of 91%. To achieve a better performing network using this method, further study of the different hyperparameters would be required.

Keywords

Gas Turbine, Gas Path Diagnostic, Artificial Neural Network, Trent-900, Neuoevolution, Genetic Algorithm

Acknowledgement

First I would like to thank my thesis supervisor, Dr Li, for being very supportive throughout my time at Cranfield. He would regularly takes the time out of his busy schedule to have our weekly meeting where he would share his knowledge and expertise. It has been a great privilege to be supervised by you and I will forever be thankful for the guidance and advice you provided.

I would also like to thank the Thermal Power team for all the hard work they put in behind the scenes to ensure the program runs smoothly and the students have an enjoyable learning experience.

To my Cranfield friends, I would like to thank you for making the journey fun and filled with laughters.

Lastly, I would like to thank my parents and siblings for their support and encouragement through this challenging yet rewarding journey.

Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
MLP	Multilayer Perceptron
HPC	High Pressure Compressor
TET	Turbine Entry Temperature
BPR	Bypass Ratio
OPR	Overall Pressure Ratio
SP	Selective Pressure
GA	Genetic Algorithm
GPA	Gas Path Analysis
EHMS	Engine Health Monitoring System
DP	Design Point
OD	Off Design
SFC	Specific Fuel Consumption
ISA	International Standard Atmosphere
IPC	Intermediate Pressure Compressor

HPT	High Pressure Turbine
HP	High Pressure
IPT	Intermediate Pressure Turbine
IP	Intermediate Pressure
LPT	Low Pressure Turbine
LP	Low Pressure
EHM	Engine Health Monitoring

Chapter 1

Introduction

1.1 Background

Ever since its invention, turbofan engines has became the go to engine for commercial airliners across the world, as it is able to generate high level of thrust while maintaining good fuel efficiency. At the core of any turbofan engine is a gas turbine. A simple gas turbine is made up of three main components, the compressor, combustor and turbine. The gas turbine operates by ingesting large amount of air through its intake, before using the compressor to compressed to air to increase the air pressure to a predefined level. The pressurised air is then fed into the combustor where fuel is introduced into the system, mixes with the air and burned to increase the overall temperature of the air. The combusted air is then expands in the turbine section of the gas turbine where thermal energy is then convert to mechanical energy to drive the shaft connected to the turbine and compressor as well as thrust at the exhaust of the gas turbine.

Due to the hostile operating environment the components in a gas turbine operates in, the components are susceptible to degradation which will lead to performance deterioration. Performance deterioration can increase the operating cost for the operator as they would need to add more fuel into the system to achieve the same output as a clean engine. The

longer the degradation to the gas path is left unnoticed, the more expensive the maintenance cost will get as the damage to the component caused by degradation is too severe to be recovered therefore it needs to be replaced. Since the components in a gas turbine are interconnected with each other, degradation in one component may lead to a domino effect and cause other components to degrade as well and may lead to a non-recoverable engine.

With the demand for air travel continues to increase for the foreseeable future, the number of turbofan engine operating at any given time will increase as well therefore number of degraded engine operating in any given time will increase as well. The rapid advancement made in computational power has enabled the development of more advanced technique such as artificial intelligence (AI) based engine health monitoring system used to combat degradation in gas turbine's gas path. These systems are used to aid original equipment manufacturer (OEM) to schedule maintenance more efficiently, therefore reducing the downtime of the engine and increase the engine's reliability which leads to higher customer satisfaction. However, with computational power still continue to increase at a rapid pace, this opens up the possibility for more advanced AI based diagnostics methods which are yet to be tested.

1.2 Aims & Objectives

This study aims to developed an artificial neural network based diagnostics system for a chosen engine. Additionally, this study also aims to develop a not so commonly used artificial neural network to evaluate its performance and capability for gas turbine diagnostics. The procedure to achieve the aim of this study is as follow:

- → Carry out a literature review to understand the different gas turbine degradation and maintenance strategy used.
- → Identify work the different gas turbine diagnostic methods used in the past

→ Develop an accurate model of the Trent-900 in PYTHIA.

→ Generate dataset which will be used to train, test and validate the neural network.

→ Select the structure of the nested neural network.

→ Develop and test the nested neural network.

→ Develop a graphical user interface for the nested neural network.

→ Develop a binary classification network using Neuroevolution.

→ Compare the performance of the network obtained from Neuroevolution against the feedforward backpropagation network

1.3 Thesis Structure

The structure of the thesis is divided into 5 main chapters to cover the theory and work done in this project

Chapter 1 - Introduction

In this chapter, a description outlining the need for gas turbine diagnostics is provided followed by the aims and objectives of the thesis.

Chapter 2 - Literature Review

This chapter starts by describing the effect of gas path degradation and the different type of degradation a gas turbine may encounter during its lifetime and its effect on the gas turbine's performance. Then follow by the different maintenance strategies used for gas turbine before providing a detailed description of predictive maintenance and difficulty faced when using predictive maintenance. Lastly, this chapter will conclude by comparing diagnostic methods that have been previously used by other authors.

Chapter 3 - Methodology

This chapter will explain the theory and concept used in this project to achieve the aims and objectives of the thesis. It starts by explaining the concept of Artificial neural network and

the theory behind it followed by the different activation functions. Then description on the different data feature scaling method and neural network training method is provided before a detailed description of backpropagation learning algorithm is described together with Neuroevolution. Lastly the chapter will end with the different hyperparameter optimisation method and the performance metrics used for neural network.

Chapter 4 - Application

This chapter will first describe the Trent-900 engine and development process for the engine model. After the engine model development process, the chapter will cover how the data used to train, test and validate the nested neural network is generated, before moving on to the development of the nested neural network. After the development of the nested neural network, the development of the diagnostic system's graphical user interface is described. This chapter will end with the description of the result obtained from using genetic algorithm to tune the weight of a given neural network architecture for binary classification.

Chapter 5 - Conclusion & Future Work

This chapter will summaries the work done and main findings in this thesis before providing recommendation for future work.

Chapter 2

Literature Review

2.1 Turbofan Gas Path Degradation

Components in a turbofan engine are subjected to high thermal and mechanical stresses due to the harsh operating environment it is operating in additionally turbofan engine operates in various operating environment as well therefore making the gas path of a turbofan to be more susceptible to degradation. Degradation along the gas path will lead to deterioration in the engine's performance therefore increasing fuel consumption as more fuel will be needed to achieve the same output of a clean engine. By increasing the fuel consumption, the TET will also increase and subsequently will lead to reduction in component's life. If it is left uncared for, the degradation level will increase overtime and eventually the engine will require a major overhaul therefore increasing the maintenance cost and downtime.

There are various type of degradation that may occur along the gas path but all of them can be categorised into three main categories, recoverable and non-recoverable degradation. The difference between recoverable and non-recoverable degradation is the amount of effort needed to recover the engine back to baseline performance. Recoverable degradation is normally caused by the built up of contaminants along the gas path. This type of

degradation is commonly referred to as fouling. When fouling occurs, the degraded engine can be recovered using online washing and this can be done between flights as it only takes a few minutes. It is also a good way to prevent fouling from occurring [33]. Non-recoverable degradation can be caused by degradation such as corrosion, erosion or increase in tip clearance. These type of degradation would require the engine to be overhaul as the degraded component will need to be replacement in order to restore the lost performance. This would subsequently increase the downtime and maintenance cost.

2.2 Different types of Degradation

2.2.1 Fouling

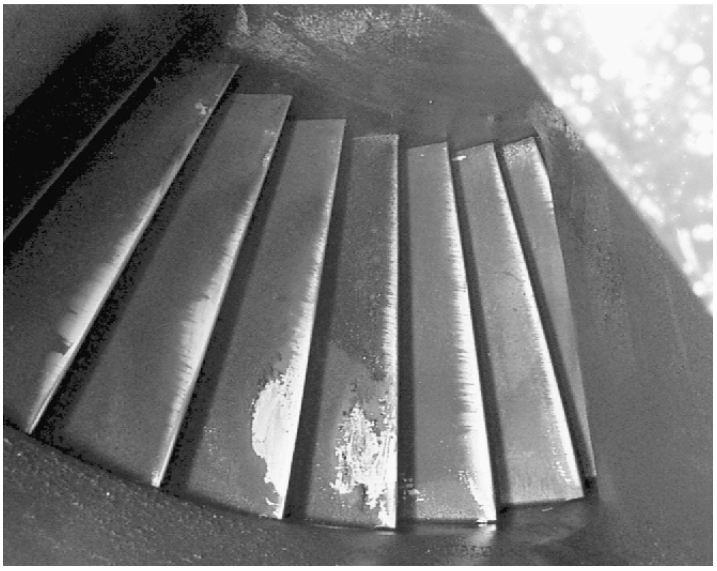


Figure 2.1: Compressor Fouling [33]

Fouling attributes to almost 70% of all degradation a gas turbine may experience during its operation [38]. It is caused by the built up of contaminants particles such as sand, heavy hydrocarbon and sea salt along the gas path [26]. Fouling occurs more prominently at the colder section of the gas turbine as the compressor is the first component the intake air will have to pass through first therefore any containment ingested will be deposited along the compressor. The build up of containment particles along the gas path may cause problems such as changes in the blade's profile, reduction in throat area and increase in

blade surface roughness [26]. This would lead to reduction in compressor outlet pressure and mass flow rate and consequently reduction in total power/thrust generated by the gas turbine. To remove the built up of containments, online washing is used periodically by injecting de-mineralised water that has been mixed with detergent into the gas turbine. If the built up of containment is substantial, it may cause a reduction in surge margin and eventually the compressor may surge if the containments are not removed [33]. Manual cleaning will required when there is a large built up of contaminant in order to wash the component thoroughly. This method would require the component to be disassemble from the gas turbine therefore significantly increasing the downtime of the gas turbine.

2.2.2 Corrosion

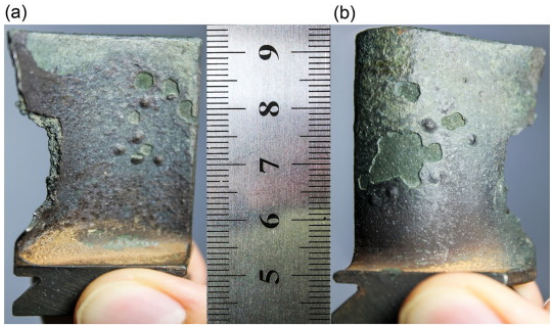


Figure 2.2: Gas Turbine Blade Corrosion [47]

Corrosion is a non-recoverable degradation that occurs in the hot section of the gas turbine. It is caused by the reaction between the material of the turbine blade and the products from the combustion process such as sulphur and vanadium [47] or contaminants ingested from the operating environment such as the detergent used for washing in a high temperature environment. When a blade suffers from corrosion, it looses some its material therefore weakening its mechanical integrity. Additionally the surface of the blade may change therefore reducing its aerodynamic efficiency [55]. This type of corrosion is defined as hot corrosion. Another type of corrosion that may occur in the hot section of the gas turbine is oxidation. Oxidation is the reaction between the metal of the turbine blade and the air it is exposed to causing the metal to be positively charged [26]. Subsequently, the surface of the

metal oxidises hence changing the mechanical properties of the metal [26]. Oxidation and hot corrosion can be prevented by applying thermal barrier coating and oxidant resistance coating to the turbine blades and nozzle guide vanes.

2.2.3 Erosion

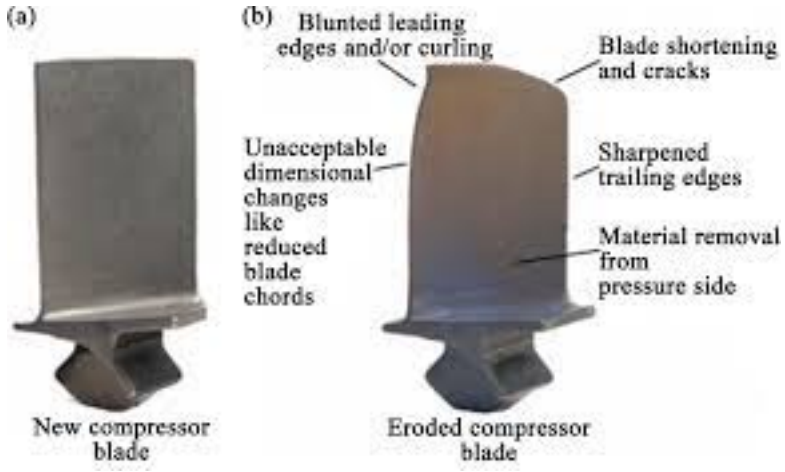


Figure 2.3: Clean vs Eroded Compressor Blade [5]

Erosion is also another type of non-recoverable degradation and is the loss of materials from components which is caused by the impact of contaminant particles that has a diameter larger than $10\mu m$ in the gas path [26]. These impact causes materials to be chipped away from the component therefore changing the surface smoothness of the blade, the profile of the blade and increasing the tip clearance between the tip of the blade and the casing. When a compressor suffers from erosion, it will cause a reduction in compressor outlet pressure and mass flow rate and subsequently a reduction in power/thrust generated. On the other hand, when a turbine suffers from erosion, the flow capacity will increase while the turbine efficiency will reduce and consequently attributing to lost of power/thrust generated from the gas turbine [38]. Since erosion is a permanent degradation, the eroded component would need to be replaced in order to restore the engine and the cost of the component can be very expensive [26].

2.2.4 Object Damage

Figure 2.4: Foreign Object Damage [52]

Loose object along the gas path will cause damage to the components as the object comes into contact with the components while in operation. Object Damage can be differentiate between foreign object damage (FOD) and domestic object damage (DOD) [38]. Foreign object damage is caused by the ingestion of objects from the gas turbine's operating environment such as birds and rocks. These foreign debris will then cause damage to the first few stages of the gas turbine and may cause the engine to surge. On the other hand, domestic object damage is caused by internally detached object within the gas turbine. This detached object is then carried downstream therefore damaging downstream components. Foreign object damage can also lead to domestic object damage. For example if a piece of the fan blade detaches from the fan due to bird strike, this piece of fan blade is then carried downstream therefore causing domestic object damage. Any kind of object damage would require the damaged component to be replaced. If the damage to the engine is too extensive, then the engine may not be recoverable.

2.3 Gas Turbine Maintenance

2.3.1 Different Maintenance Strategies

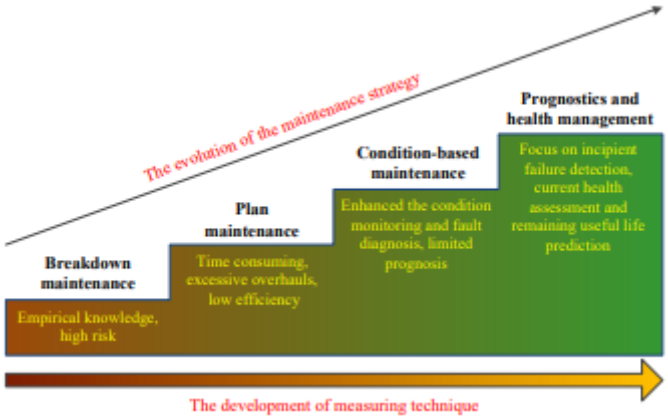


Figure 2.5: Evolution of gas turbine maintenance strategy [57]

Maintenance is essential in order to preserve the health of the engine and keep the engine operating at its optimal level. Advancement in technology have enable the development of modern maintenance strategies like predictive maintenance. This has become the standard maintenance strategy used in the industry. Before predictive maintenance, various other strategies have been adopted in the past starting with breakdown maintenance. Breakdown maintenance takes on a reactive approach, where essentially the engine is allowed to run to its failure before any maintenance is carried out [19]. This maintenance strategy does not guarantee the reliability and safe operation of the gas turbine [57] as the operator does not know when the engine will fail and how catastrophic the failure will be when the engine fails.

The rise of plan maintenance came after World War 2 where technological and industrial evolution made during the war gave rise to newly found engineering concepts such as reliability, availability, maintainability and operational cost optimisation [19]. Businesses started taking a more proactive approach to reduce their downtime by using plan maintenance. Plan maintenance carries out maintenance after a predefined number of operating

hours to prevent the engine from reaching failure. Even though this strategy increases the reliability of the engine by having more regular maintenance, it can also lead to unnecessary downtime since the health of the engine is not known to the operator until the engine is shut down.

2.3.2 Predictive Maintenance

The development of predictive maintenance was prompted by the fact that businesses were searching for more effective ways to schedule maintenance to reduce downtime to remain competitive. Predictive maintenance such as condition-based maintenance was also made available due to the technological advancement made to sensors and computing power in the last few decades. This strategy uses measurements data which are collected from the engine using sensors to monitor the health of the engine. This strategy allows the operator to know the health of the gas turbine without needing to shut it down, therefore the operator can monitor the degradation level of the engine and determine when maintenance is needed. This helps reduce the operational cost by reducing the number of hours the engine operates with degraded components therefore reducing the amount of fuel used and subsequently less emission is being released into the atmosphere .

Additionally, it also helps reduce maintenance cost since maintenance is only carried out when it is needed therefore avoiding any unnecessary downtime. Since the health of the gas turbine is known to the operator, the operator can also plan ahead by pre-ordering the necessary component and request ahead of time for the additional manpower needed on site to carry out the maintenance. This helps to reduce the downtime of the engine since everything will be readied on site to carry out the maintenance. Since the performance of the gas turbine is being monitored, this also allows OEMs to continuously implement new design changes during the maintenance to improve the performance of the gas turbine.

A condition monitoring system is made up of 4 different systems:

- Data Acquisition System
- Diagnostics System
- Data Fusion System
- Maintenance Management System

The data acquisition system is responsible for logging all the different engine measurements collected by the sensors installed on board the engine such as the fuel flow, rotational speed and temperature. These measurements will then be used by the diagnostics systems to determine the health status of the engine by detecting and isolating degraded components before quantifying the component's degradation level. The data fusion system is responsible for fusing all the various data and information from multiple different sources together to provide a easily understandable conclusion of the diagnostic result for the operator. The maintenance management system will be used to log information about the maintenance which has been carried out such as the description of the maintenance work done, the cause for the maintenance and the ID of the personnel assigned to carried out the maintenance work.

2.4 Gas Turbine Diagnostics

Due to the complexity of modern gas turbine, gas turbine diagnostics is essential to increase the reliability for the modern engine as it alerts operators to imminent failures. The gas turbine diagnostics process can be broken down into 3 main processes detecting, isolating and quantifying of an imminent failure condition while keeping the gas turbine in operation [21]. The process of detecting is to identify anomalous behaviour in the gas turbine which may signifies an imminent failure condition. The isolating process then determines the cause for the anomalous behaviour by identifying the type and location of the defect as each component degradation is associated with a fault signature. Lastly the quantifying process will estimate the severity of the defect. This process provides the operator with a more holistic view of the health of the gas turbine, therefore they can make a more inform

decision on what needs to be done. Some of the gas turbine diagnostics method are visual inspection, vibration monitoring and borescope inspection [28].

2.5 Performance Based Diagnostic Systems

Another gas turbine diagnostics method is performance based diagnostic systems. Performance based diagnostic systems evaluates the measurements provided by the sensors installed along the gas path to determine the components degradation level [28]. However in order for the performance based diagnostics system to provide accurate diagnosis, it must be able to overcome a few challenges. These challenges includes:

- **Non linear relationships -** This is due to the fact the relationship between the measurements obtained from the sensors and the engine's performance variables are highly non-linear [21], therefore the diagnosis system must be able to handle non linear relationships.
- **Limited number of sensors reading available -** Due to weight and limited suitable location where sensors can be installed, the number of sensors installed in the engine is limited, therefore only a limited number of measurements can be obtained from the engine. The diagnostic system must be able to make use of the limited number of engine measurements which are available to accurately diagnose the gas turbine.
- **Sensors readings inaccuracies -** The readings obtained from the sensors will also contain some level of noise and biases. Noise is caused by the high temperature and pressure operating environment in the gas turbine, therefore adding meaningless information into the data. On the other hand, bias is a fixed error caused by fault within the sensor itself leading to higher or lower measurement values [21]. If the diagnostic system proposed is not capable of handling noisy and bias data, the system may wrongly diagnose a clean component to be faulty.
- **Variation in operating condition -** The engine will not operate at a fixed power level and the same ambient condition everyday therefore changes in power level

and ambient condition will affect the measurement obtained from the sensors. As seen in Figure 2.6 where x-axis is the two different operating condition and y-axis is the deviation in performance, the degraded engine has a lower performance measurement. The degraded engine measurement at operating condition b has a similar measurement as the clean engine at operating condition A, therefore if this is not taken into account when designing the diagnostics system, when the degraded engine is operating at condition B, the diagnostic system may miss classified the engine to be clean and operating at condition A. Equation 2.1 are used to correct the degraded engine measurement and the clean engine measurement at condition B to the baseline condition. In this case, point A is the selected as the baseline condition.

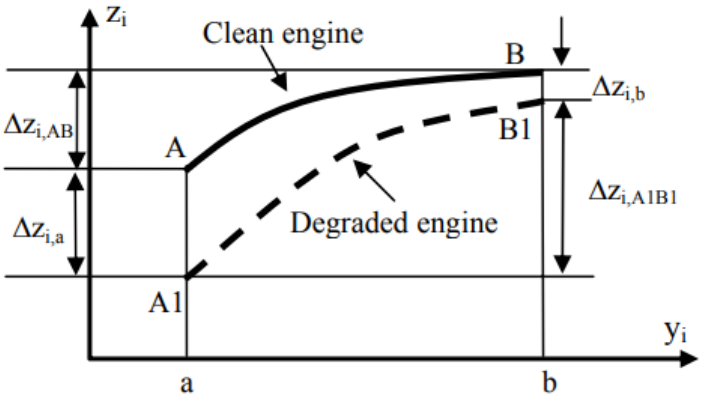


Figure 2.6: Data Correction [29]

$$
\begin{aligned}
\Delta Z_{i,a} + \Delta Z_{i,AB} &= \Delta Z_{i,b} + \Delta Z_{i,A1B1} \\
\Delta Z_x &= \Delta Z_{i,A1B1} - \Delta Z_{i,a} \\
\Delta Z_{i,a} &= Z_A - Z_{A1} \\
\Delta Z_{i,b} &= Z_B - Z_{B1} \\
\text{Clean Engine Correction } &\rightarrow \Delta Z_{i,AB} = Z_B - Z_A \\
\text{Degraded Engine Correction } &\rightarrow \Delta Z_{i,A1B1} = Z_{B1} - Z_{A1}
\end{aligned}
\tag{2.1}
$$

ΔZ_A = Performance deviation of **clean Engine @ Operating Condition A**

$\Delta Z_A 1$ = Performance deviation of **degraded Engine @ Operating Condition A**

ΔZ_B = Performance deviation of **clean Engine @ Operating Condition B**

$\Delta Z_B 1$ = Performance deviation of **degraded Engine @ Operating Condition B**

- **The presence of multiple faults in the gas path occurring at the same time**
  - There is a possibility for multiple components to degrade at the same time due to the harsh operating environment the components in the gas path are exposed to. The measurements from multi-component degradation may be similar to single component degradation, therefore the system needs to differentiate between them if not the component may identify a single component degradation as multi-component degradation or vice versa.

Various diagnostic systems has been developed and used in the past and they can be differentiated into two different category, model based (MB) and artificial intelligence (AI) based diagnostics systems. MB diagnostic systems uses the engine's thermodynamic model to diagnose the engine while AI based diagnostic systems uses simulated human intelligence on a machine to diagnose the engine [21].

2.5.1 Gas Path Analysis

One of the MB method is gas path analysis (GPA). Within GPA, there is also linear and non-linear GPA. Linear GPA assumes a linear relationship between the dependent variables and independent variables [28] and it can be express using Equation 2.2, where $\vec{z}$ is the measurable variables , $\vec{x}$ is non-measurable variables and H is the Influence Coefficient Matrix (ICM) [21]. In linear GPA it is also assumed that the engine is operating at fixed operating point with no changes to the ambient condition and power setting. [21].

$$\vec{Z} = H \cdot \vec{x} \qquad (2.2) \qquad\qquad \Delta\vec{x} = H^{-1} \cdot \Delta\vec{z} \qquad (2.3)$$

The influence coefficient matrix represents the relationship between the measurement and non measurable engine variables at a fixed operating condition [28]. If the number of measurable and non measurable variables are the same, the influence coefficient matrix can be inverted. The inverse of the influence coefficient matrix is fault coefficient matrix (FCM), this matrix then can be used to determine the deviated non-measurable engine variables. Since linear GPA assumes the relationship between the measurable and non measurable variables to be linear, it may struggle to converge when the level of degradation is high [21]. Additionally, developing the influence coefficient matrix requires prior knowledge about the change in component behaviour cause by different degradation in order to model the relationships between the variables accurately as the accuracy of the linear GPA heavily depends on the influence coefficient matrix [21].

$$\Delta\vec{z} = H^{-1} \cdot \Delta\vec{x} \quad (2.4) \qquad RMS = \sqrt{\frac{\sum_{j=1}^{M} \left(\frac{Z_{j,predicted} - Z_{j,actual}}{Z_{j,actual}} \right)^2}{M}} \quad (2.5)$$

Non linear GPA considers the non-linear relationship between the measurable and non-measurable engine variables and it can be expressed using Equation 2.4, where $\Delta\vec{z}$ is the measurable variables measurement deviation in percentage, $\Delta\vec{x}$ is the non measurement engine variables and H is the influence coefficient matrix. Non linear GPA uses the Newton-Raphson iterative method to look for the best solution by minimising a objective function [21]. Figure 2.7 illustrates how a non linear GPA which uses Newton Raphson method searches for the best solution. It starts from the baseline where the first influence coefficient matrix is generated by introducing a small change to the performance of the component. The generated influence coefficient matrix is then inverted to obtained the fault coefficient matrix which will then be used to calculate the non measurable variables, $\Delta\vec{x}$, by multiple the fault coefficient matrix with deviated measurable variables, $\Delta\vec{z}$. Then another influence coefficient matrix is developed and the process repeats itself until a solution has been found [21]. Root mean squared error can be used to evaluate the accuracy of the prediction and it is calculated using Equation 2.5 where M is the number of measurable

variables. Non-linear GPA has similar disadvantages to linear GPA, it is also desirable to have a large amount of sensors installed into the engine [21]. However this is not feasible due to the weight restriction and limited number of suitable location available where sensors can be installed.

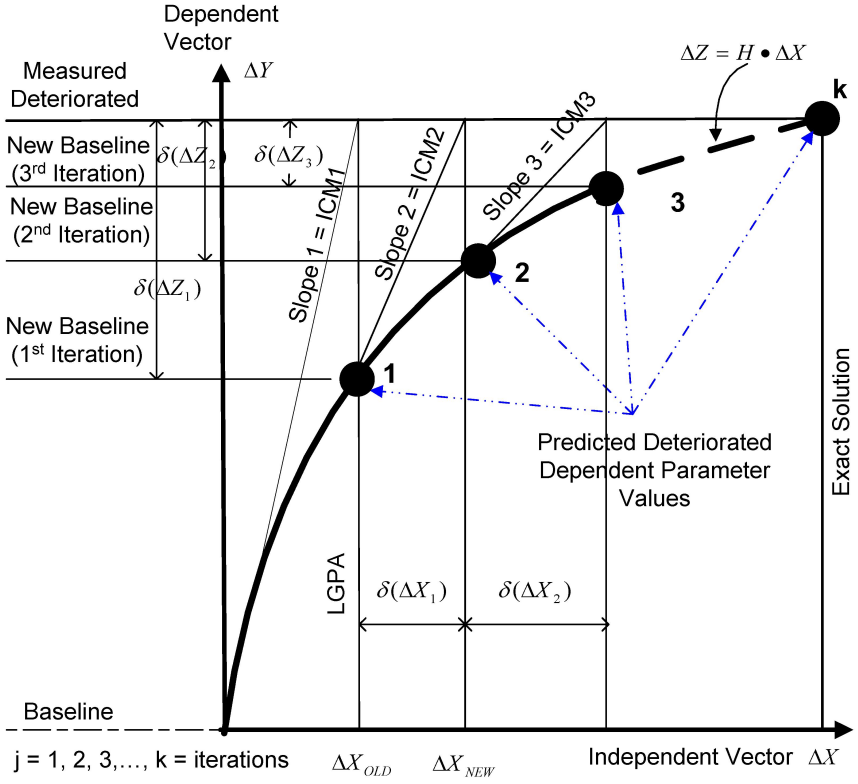


Figure 2.7: Non Linear GPA Newton Raphson Method [21]

2.5.2 Fuzzy Logic Method

The rise of computing power pave the way for AI based diagnostic methods. AI based diagnostic method uses computer to simulates human intelligence in order to solve a problem [21]. One of the AI based method is fuzzy logic. This method is widely used to determine the relationship between measurable and non-measurable variables [21]. A fuzzy logic system is made up of 4 components, a Fuzzifier, a Fuzzy Rules, an Inference Engine and a Defuzzifier [21]. The purpose of the Fuzzifier is to transform the signals from the input to a fuzzy set. This fuzzy set is then passed the Inference Engine where it will be determine which fuzzy set combines with each other according to the fuzzy rules.

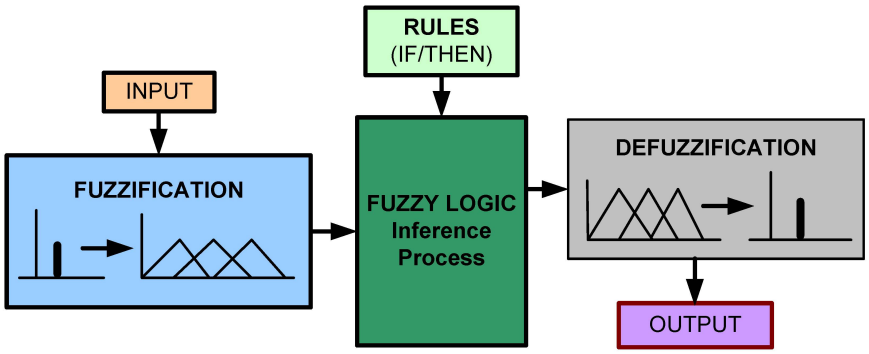


Figure 2.8: Fuzzy Logic System [21]

It is then passed onto the Defuzzifier where fuzzy set will be transformed back into the original form. However this method would also require prior expert knowledge to build up the fuzzy rules in order to achieve accurate diagnosis [21].

Chapter 3

Methodology

3.1 Artificial Neural Network

3.1.1 Introduction

Advancement in computational processing capability during the last few decades has enable advancement in technologies such as artificial intelligence (AI), which is now widely integrated into our daily life ranging from self driving cars to the use of facial recognition to unlock our phones. One sub-field of AI is machine learning [17]. Machine learning uses algorithms that imitates how the human brain process information to lean the trends and patterns from data. One of the commonly used machine learning algorithm is Artificial Neural Network (ANN).

Artificial neural networks (ANNs) are mathematical models that is biologically inspired by the human brain [1] and it was first conceptualised in 1943 by McCulloch, a neurophysiologist and Pitts, a mathematician. They presented a mathematical model of a neuron which was then used to develop the perceptron model (Figure 3.1) [48]. The perceptron model works by multiplying each input node (x) with the a weight value (w) before summing all the weighted inputs together. The sum of all the weighted input is then compared to a threshold limit. If the total sum of the weighted input is greater than the user defined

threshold, the output from the preceptron model will be one, if it is less than the user defined threshold, the output will be zero [48].

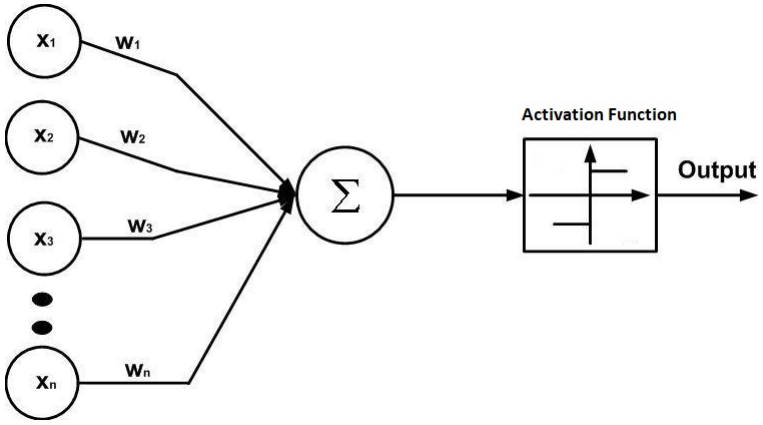


Figure 3.1: McCulloch and Pitts perceptron model [32]

The preceptron model that was developed by McCulloch and Pitts provided Rosenbaltt the foundation for his work. Rosenbaltt presented a theory that claimed that the perceptron models are capable of learning anything that is presented with [48]. However this was not the case as numerous compelling models that was developed by researchers in the 1960s was not capable of learning simple tasks [48]. Only after 1977, models such as Multi-layer perceptron, feedforward backpropagation algorithm and self organising maps were introduced. These models attracted more researchers into the field and it led to the rapid development in neural network which is now being used in almost every sector.

3.1.2 The Human Brain

In order to understand the how artificial neural networks (ANN) works, we must first understand how the human brain learns and process information since it is where ANN got its inspiration from. The human brain is a incredibly flexible massive neural network that consist of more than 10 billion neurons. It is capable of receiving signals from various body's sensory organs, process the signal and produce an output signal to the muscles accordingly. The neurons are all interconnected with each other by linking their synapses and dendrites together to. Dendrites are branch like features that are branch out of the cell

body. Its function is to receive electrical signal from the synapses. The synapses are branch liked features that is extended from one end of the axon and its function is to transmit electrical signals between two neurons. The other end of the axon connected to the soma also known as cell body. The axon is a single long fibre [1] and its purpose is to transmit electrical signals from the neuron that is contained within the cell body to the synapses which will then transmit that electrical signal to the dendrites of another cell body.

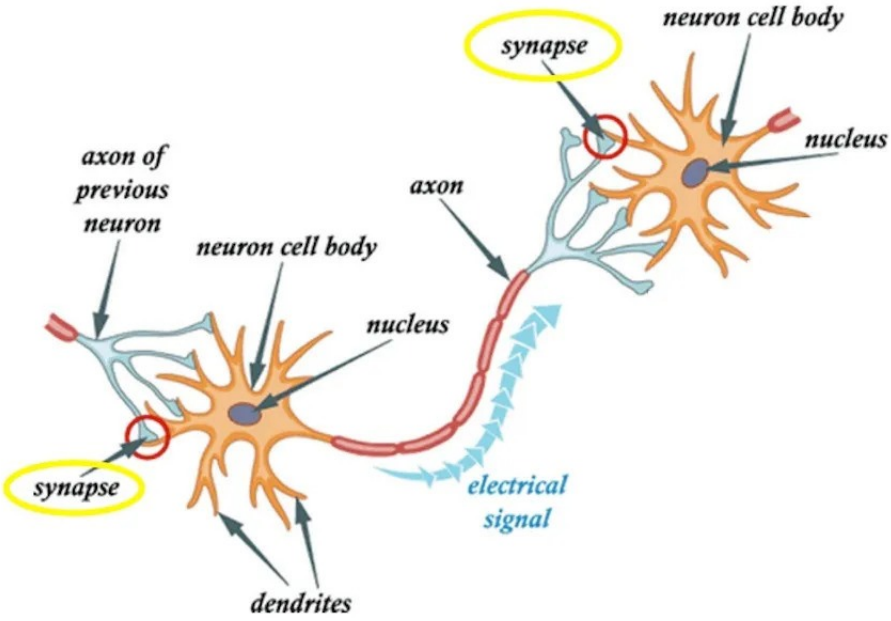


Figure 3.2: Biological Neuron Connections [25]

3.1.3 Artificial Neural Network's Concept

Artificial neural networks (ANN) are biologically inspired computation model developed using mathematical models that tries to replicate in a smaller scale how information are process in the human brain [48]. An ANN mainly comprise of 3 types of layers, an input layer, a or multiple hidden layers and an output layer. Each layer consist of nodes that are interconnected between each layer. Nodes are mathematical modes that tries to replicate the neurons in the human brain and the connection between two nodes represents the synapses [1]. How strongly two nodes are connected is determined by the connection weights. The connection weights determines how much influence a node have on the

network's output. The larger the connection weight is, the more important that node will be therefore the greater influence that node will have on the network's output [18] and this is tuned during the training process.

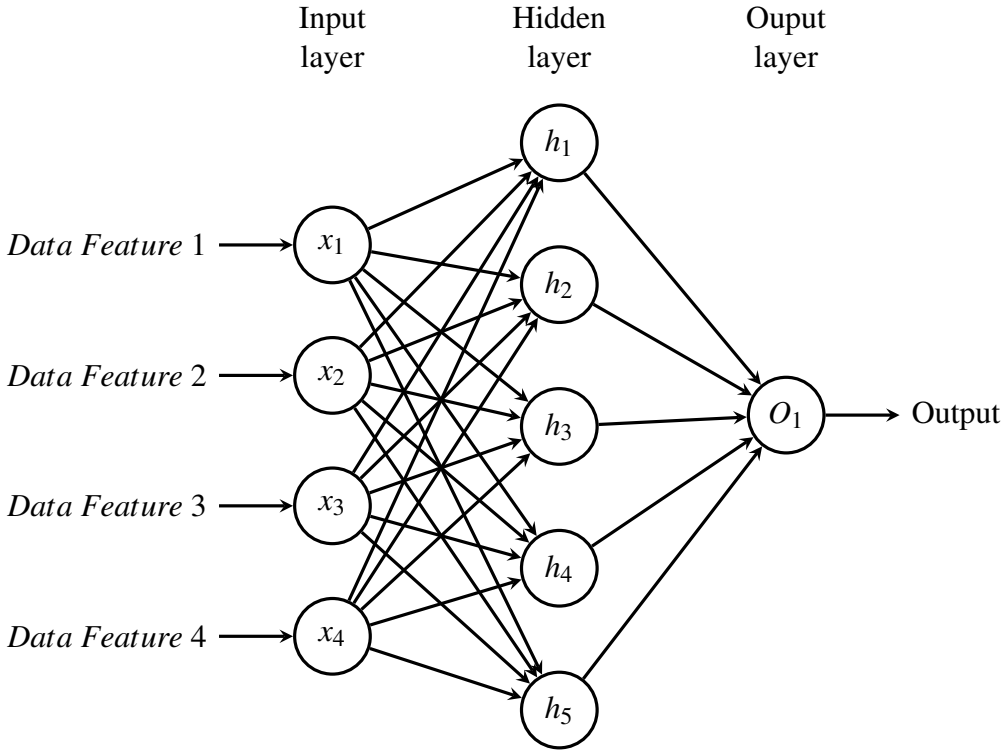


Figure 3.3: Simple FeedForward Neural Network Architecture

Nodes in different layer have different functionality. Figure 3.3 is a 3 layer feedforward neural network where the signals will only travel in the direction indicated by the arrows. The input layer is the starting point of the neural network and the nodes in this layer is responsible for receiving the input data and sending it to the hidden layer. Each node in this input layer represents a different feature of the data, therefore the number of nodes in the input layer will be the same as the number of features in the data. After the input layer is the hidden layer. The nodes in the hidden layer is responsible for summing all the weighted input it receives from the input layer and applying a transformation to the total sum using an activation function to introduce non-linearity into the neural network. The output of a hidden node (h_n) can be calculated using Equation 3.1, where b_n is the bias associated to the node, ω_n is the connection weight and x_n is the input. Lastly the nodes in

the output layer is responsible from adding all the outputs from the hidden layer by using the same equation (Equation 3.1) that is being used by the hidden node. The total sum calculated is then transform to output the prediction made by the neural network.

$$h_n = b_n + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \ldots + \omega_n x_n \tag{3.1}$$

3.2 Nested Neural Network

Neural networks are only train to carry out a specific task efficiently therefore when there are multiple different task that needs to be solve in order to resolve a problem, a possible solution might be to use nested neural networks. Nested neural network resolves a multi step problem by employing several neural network. Each neural network will be responsible for solving certain step of the problem therefore when they are combined together, they can be used to solve the problem. Figure 3.4 is an example of a nested neural network architecture for a single spool gas turbine diagnostics system. First, the user will input the necessary engine sensor measurements into the nested neural network. The measurement will then go through the first classification neural network. This first network will determine if the engine is clean or degraded. If it determines the engine to be clean, the diagnostic process is complete. If it determines the engine to be degraded, the measurements will be passed onto a second classification neural network. This neural network will be used to isolate the component that is degraded. Once the degraded component has been isolated, it will go through the respective regression neural network. The regression neural network will then estimate the component's level of flow capacity and efficiency degradation to complete the diagnostic process.

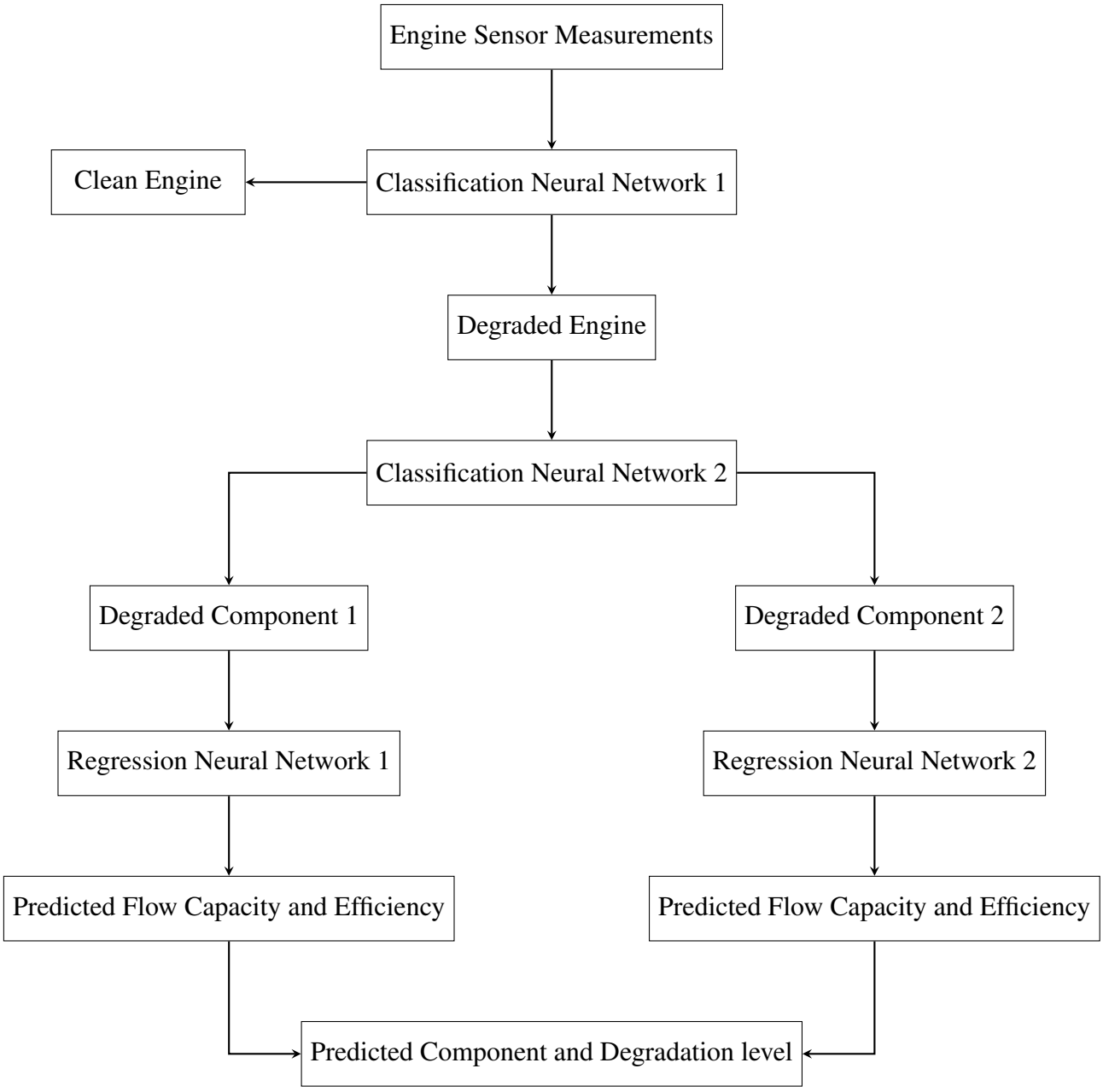


Figure 3.4: Example of Nested Neural Network for Gas Turbine Diagnostics

3.3 Activation Functions

Activation functions are functions used to transform the output of the nodes in the hidden and output layer to introduce non-linearity into the neural network. This is necessary for the neural network to learn the complex relationship between each features in a multi-dimensional dataset [43]. Without using activation functions in the hidden and output layer, the output of the neural network will have behave linearly to the weights, biases and input as the data passing through the neural network will only have gone through Equation 3.1, which is a linear equation therefore the neural network will not be able to pickup non-linear relationships within the data. Additionally, each activation function have their own characteristics, therefore it is important that the activation function chosen for the hidden and output layer is suitable for the problem the neural network is trying to solve. If the wrong activation function is chosen, it will affect the network's capability and performance.

3.3.1 Sigmoid Activation Function

There is a host of activation functions available but the most commonly used activation functions are Sigmoid Function, Rectified Linear Unit (ReLU) and Hyperbolic Tangent Function (TanH). The sigmoid function transforms the output value of the node so that it falls between 1 and 0. From Equation 3.2 it can be seen that the output of the sigmoid activation function is dependent on h_n and from Equation 3.1, h_n is dependent on the bias, weight of the connections and inputs. Since inputs is dependent on the data and assuming bias of the node to be fixed, it can be determined that the output of the sigmoid activation function is dependent on the weight of the connections. Subsequently, increasing the connection weight will increase h_n therefore the output of the sigmoid activation function will trend towards 1. Alternatively, reducing the connection weight will reduce h_n therefore the output of the sigmoid activation function will trend towards 0.

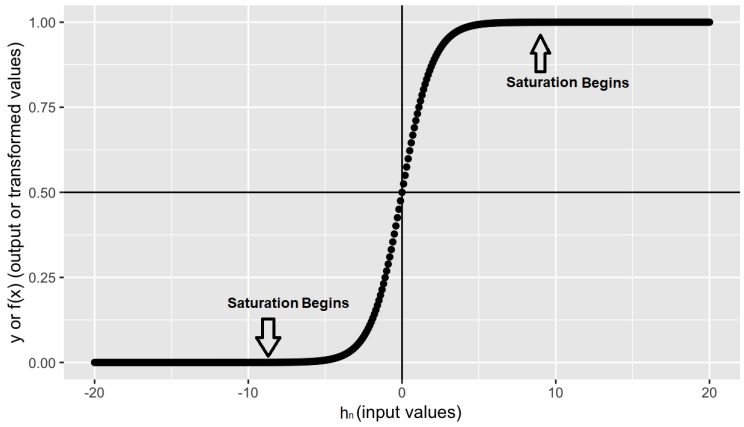


Figure 3.5: Sigmoid Activation Function [31]

$$S(h_n) = \frac{1}{1+e^{-h_n}} \tag{3.2}$$

From Figure 3.5 it can be observed that if h_n value does not fall within the -5 to 5, the output from the sigmoid activation function will be either 0 or 1 no matter how far outside the range the h_n value is. This will cause the output of the sigmoid function to become saturated. When the output of a node becomes saturated, the training process will be compromised as the training algorithm used to optimised the weights of the connection may not get any feedback on whether the changes to the weights positively or negatively affected the network's performance, as h_n may still fall outside of the active range therefore the output of the node will still be either 1 or 0 [42]. When this happen the neural network will stop learning. Another characteristic of sigmoid activation function is its non zero centroid. A non centroid activation function will always output either all positive or negative outputs only, therefore more epochs will be required to optimise the connection weights during the training process [11].

3.3.2 Hyperbolic Tangent Activation Function

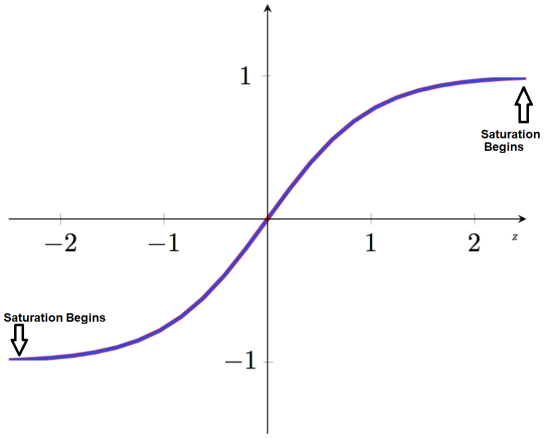


Figure 3.6: Hyperbolic Tangent Activation Funciton [37]

$$\text{TanH}(x) = \frac{e^{h_n} - e^{-h_n}}{e^{h_n} + e^{-h_n}} \tag{3.3}$$

One activation function that behaves similarly to sigmoid activation function while having a zero centroid characteristic is the hyperbolic tangent (TanH) activation function. Compare to Sigmoid activation function, the output of a hyperbolic tangent activation function has a wider range as it ranges from -1 to 1. Its zero centroid characteristic enable the mean of the output to be close to zero therefore far away from the saturation regions. Additionally, hyperbolic tangent activation have a faster convergence rate and higher classification accuracy compared to sigmoid activation function [13]. This is why hyperbolic tangent is preferred over sigmoid activation function. However hyperbolic tangent also suffers vanishing gradient and saturation problem similar to sigmoid activation function [13].

3.3.3 Rectified Linear Unit Activation Function

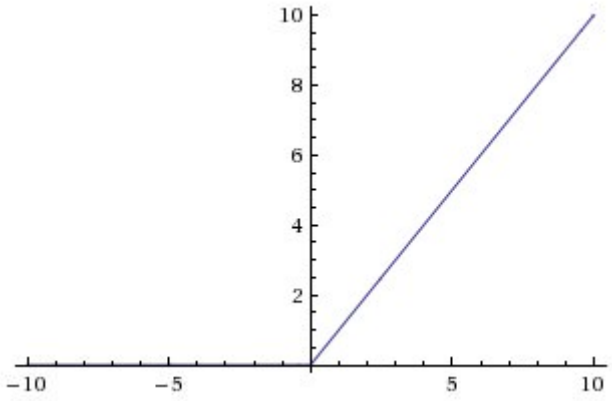


Figure 3.7: Rectified Linear unit Activation Function [10]

Rectified linear unit activation function is the most popular activation function for hidden layer amongst the three activation function [13] as it is able to overcome the problem hyperbolic tangent and sigmoid activation function suffers from. Rectified linear unit activation function overcomes vanishing gradient by having 0 output when the input value is negative and behave linearly where the gradient is 1 when the input value is positive [11]. This allow rectified linear unit activation function to be computational cheaper than sigmoid and hyperbolic tangent activation function while addressing the vanishing gradient faced by the other two activation function [13]. The downside of rectified linear unit activation function is its unbound upper limit therefore the output of a node may shift the bias of the nodes in the next layer [13]. Another issue with rectified linear unit activation function is the zero gradient for any negative inputs. This may cause the nodes to not update at all during the training process or response to variation in the input data therefore leading to a phenomenon called ”Dying ReLu” [20].

Activation Function	ReLU	Sigmoid	TanH
Output Value Range	0 to ∞	0 to 1	-1 to 1
Vanishing Gradient	No	Yes	Yes
Characteristic or Behaviour	Non Linear	Non Linear	Non Linear
Centroid = 0	No	No	Yes

Table 3.1: Comparing Hidden Layer Activation Function

3.3.4 Softmax Activation Function

$$\sigma(\overrightarrow{h}) = \frac{e^{h_i}}{\sum_{j=1}^{n} e^{h_j}} \tag{3.4}$$

For a classification problem with multiple different classes, softmax activation function is normally used in the output layer. Softmax transforms all the values in the input vector it receives from the previous layer to output values between 1 and 0 regardless of the value's polarity. Additionally, the sum of all transformed value should be equal to 1. The value transformed by softmax activation function can also be view as the probability [54] predicted by the neural network for each class. The closer the transformed value is to 1, the higher probability the neural network predicted that class to be the output class. Equation 3.4 is the mathematical definition for softmax activation function where h_i is i^{th} element in the vector and the denominator of the equation is the sum of all the exponential input. This is to ensure the output of the activate function falls between 1 and 0 and the sum of all output values is equal 1 [54].

3.4 Feature Scaling

Feature scaling is a important pre-process step needed to prepare the data before it is used to train and test a neural network. This is to ensure all data is scaled to a certain range using various scaling techniques [15]. By scaling all the data to a certain range, it helps prevent the neural network to be bias towards features that has comparative larger values than others. Additionally, feature scaling also helps to speed up the training process as well. Some feature scaling techniques includes standardisation and normalisation. Normalisation is done using Equation 3.5 where X is the initial value, X_{min} is the minimum value of the feature and X_{max} is the maximum value of the feature. When using normalisation, the scaled value will fall between 1 and 0 with 1 being the maximum and 0 being the maximum value.

Another scaling method is standardisation. Standardisation scales the data so that the mean distribution of the feature is zero and the standard deviation is one [15]. This is done by using Equation 3.6 where X is the initial value, μ is the feature's mean value and ε is the standard deviation of the feature. The selection of feature scaling technique is dependent on the problem being investigated and the algorithm used, it is important that suitable feature scaling technique is used to scale the data, as it may influence on the performance of the neural network. One method to determine which feature scaling technique to use is to first scale the data, then use it to train and test the neural network before evaluating and comparing the performance of the neural network between various scaling techniques.

$$\text{Normalisation} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (3.5)$$

$$\text{Standardisation} = \frac{X - \mu}{\varepsilon} \qquad (3.6)$$

3.5 Neural Network Training

For a neural network to "learn" the patterns within a dataset and provide an accurate prediction, it is necessary to train the neural network using training algorithms. During the training process, the weight of the connections are adjusted accordingly to emphasis on features within the dataset that the neural network deem to be important in achieving an accurate prediction. This is done by assigning larger weight values to those connections connected to the important feature to enable important features to have a greater influence on the network's output.

3.5.1 Supervised Learning

The learning algorithm used the the training process will determine how the neural network will learn from the dataset. The two main learning algorithm used to train a neural network is supervised learning and unsupervised learning. However there are also other learning algorithms such as reinforcement learning and stochastic learning [41]. In supervised

learning, the dataset used to trained the neural network will contain the output labels. For a classification problem, the output labels will be the class associated and for a regression problem the output labels will be the target value. Depending on the training algorithm used, the output labels will be used to calculate various performance indicator of the neural network such as its accuracy or prediction error [4]. These performance indicator will then be used to tuned the weight of the connections accordingly to improve the network's prediction accuracy [41]

3.5.2 Unsupervised Learning

The main difference between supervised and unsupervised learning is the use of output labels. In unsupervised learning, the neural network learns the pattern within the dataset without the aid of labelled outputs, therefore it is able to develop its own understanding on how the two features are linked together [4]. Since unsupervised learning does not require labelled outputs, it can be used to label unlabelled datasets by identifying patterns within the dataset and cluster data together based on their similarity [4]. This would allow the dataset to be used in supervised learning once it has been labelled. Some of the commonly used neural network that uses unsupervised learning are self organising maps and autoencoders.

3.6 Backpropagation Learning algorithm

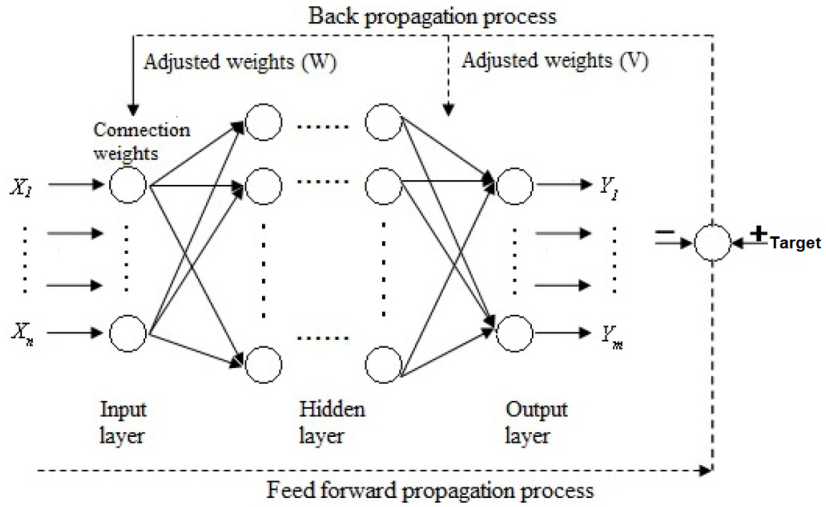


Figure 3.8: FeedForward Backpropagation Neural Network Example [27]

The most commonly used supervised learning algorithm for a feedforward neural network is backpropagation. Backpropagation learning algorithm is widely used classification and regression problems due to its simplicity and capability in achieving good performing neural network for non-linear problems [45]. This leaning algorithm improves the performance of a network by employing a gradient descent technique or hill climbing technique to find the optimal sets of weights. It propagates the error calculated between the prediction made by the network and the desired target output back into the network to make adjustments to the connections weight to improve the accuracy of the network. This process is repeated multiple times until it has met the predefined stopping criteria either be it the number of epochs have be met or a good performing neural network has been achieved [4].

3.7 Neural Network Performance Metrics

3.7.1 Loss Functions

Loss functions are function used to evaluate the performance of the neural network by calculating the average error between the prediction made and the actual target to determine how well the network is able to model the data during training process. During the training

process, the calculated error will be used by the optimiser to adjust with weights and biases of the connection to improve the performance of the neural network. Loss functions can be classified into two main categories, loss functions for classification problems and loss functions for regression problems [36].

Mean Squared Error (MSE) is used for regression problems [36] and it is calculated using Equation 3.7 where n is the total number of data points in the dataset, y_i is the value of the actual target and $\widetilde{y}$ is the value predicted by the neural network. As the error calculated in MSE is being squared, it makes MSE very sensitivity towards outliers therefore one big outliers can have a big effect on the MSE value [23]. Additionally, MSE is also sensitive towards the mean and pattern of the prediction [23]. From Equation 3.7, it can be seen that the orange line was able to identify the trend of within the dataset, however the mean of the prediction value is lower. On the other hand, the green line was able to get the mean right, but it did not get the pattern right. An variant of the MSE is root mean squared error (RMSE). RMSE as seen in Equation 3.8 square roots the mean squared error therefore it has the same scale as the predicted value hence a better performance indicator for regression neural networks.

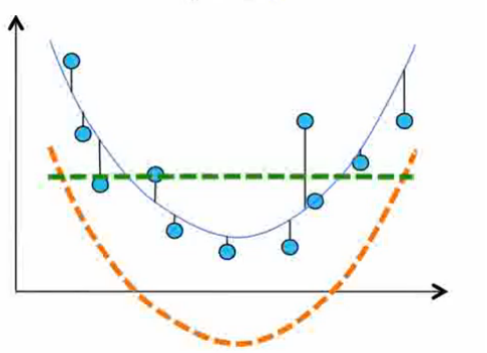


Figure 3.9: Mean Squared Error Graph [23]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2 \qquad (3.7) \qquad\qquad RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2} \qquad (3.8)$$

The loss function for classification neural network is log loss or also known as cross

entropy. In binary classification the log loss is calculated using Equation 3.9 where y_i is the actual class label and p is the predicted class label. In multi-classification, log loss function is slightly different from the one in binary classification. In multi-classification, the log loss is calculated using Equation 3.10, where every the log loss of every class is calculated individually before summing them together.

$$L_{log}(y,p) = -\frac{1}{n}\sum_{i=1}^{n}(y_i\ log(p) + (1-y_i)\ log(1-p)) \tag{3.9}$$

$$L_{log}(y,p) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{i,j}\log p_{i,j} \tag{3.10}$$

3.7.2 Performance Metric for Classification Models

There are various performance metric that can be used to evaluate the performance of a classification neural network. The simplest performance metric would be the network's accuracy. The network's accuracy is an indication of the network's ability to correctly predict the output class and it can be calculated very quickly by using dividing the number of correct prediction with the total number of prediction and multiple by 100. Using accuracy alone to evaluate the performance of a classification neural network may not be sufficient especially when there are more than 2 classes, as the accuracy calculated would be the average accuracy amongst all the classes. If a scenario where the neural network has a big difference in the prediction accuracy between the classes, the designer would not be able to identify this difference if they evaluate the performance of the neural network solely based on its accuracy.

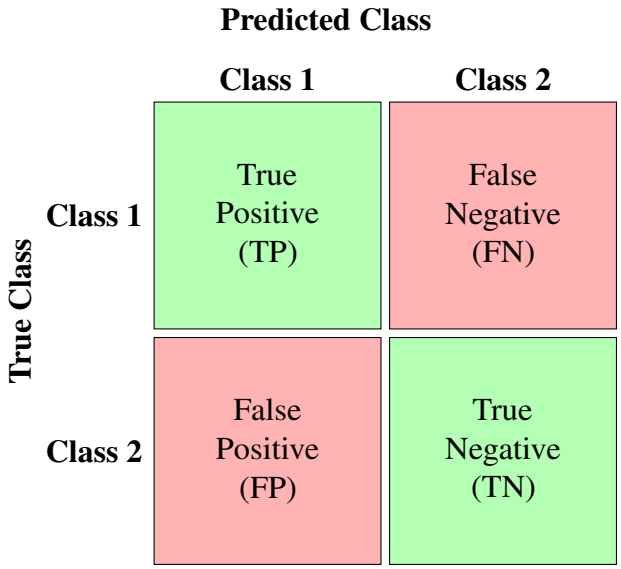


Figure 3.10: Confusion Matrix Example

To achieve a more precise breakdown of the network's accuracy for each class, a confusion matrix is used. Not only does a confusion matrix display the correct prediction for each class, it also displays the break down of incorrect predictions made by the neural network for each class. This would allow the designer to know how good or poorly the neural network is predicting for each classes. Figure 3.10 is an example of a confusion matrix for a classification problem with 2 classes. Where the True Positive (TP) indicates how many class 1 sample is predicted correctly by the neural network whereas True Negative (TN) indicates how many class 2 sample is predicted correctly by the neural network. False Negative (FN) indicates how many wrong prediction made by the neural network for class 1 and False Positive (FP) indicates how many wrong prediction made by the neural network for class 2.

$$\text{Acc} = \frac{TP+TN}{TP+TN+FP+TN} \tag{3.11}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3.12}$$

$$\text{Hit Rate} = \frac{TP}{TP+FN} \tag{3.13}$$

The model's accuracy can be calculated using Equation 3.11. Other than the accuracy, the precision and hit-rate can also provide useful information about the model. The precision describes the model's ability to predict all true positives correctly and this is calculated using Equation 3.12. If the model has a precision value of 1, it means that the model is able to correctly predict all true positives. The model's hit rate describes how many correct predictions the model made compared to the total number of predictions it made therefore it can be describe as the model's prediction correction. This is calculated using Equation 3.13. Once the precision and hit rate is known, the F1 score can be calculated (Equation 3.14). The F1 score describes the network's accuracy on the dataset by combining the precision and hit rate calculated to find the harmonic mean of the model [54].

$$F_1 = 2 \times \frac{\text{precision} \times \text{hit rate}}{\text{precision} + \text{hit rate}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.14}$$

Using the confusion matrix a receiver operating characteristic (ROC) curve can be developed. The ROC curve illustrate the model's performance at different threshold by plotting the false positive rate (FPR) against the true positive rate (TPR). The TPR (Equation 3.15) describes the model's ability to correctly identify all positive samples of a class while FPR (Equation 3.16) is the percentage at which negative samples of a class is incorrectly predicted the network as positive. Lastly the false negative rate (FNR, Equation 3.17) is the percentage at which positive samples of a class is incorrectly predicted as negative by the neural network.

$$\text{TPR} = \frac{TP}{TP + FN} \tag{3.15}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{3.16}$$

$$\text{FNR} = \frac{FN}{FN + TP} \tag{3.17}$$

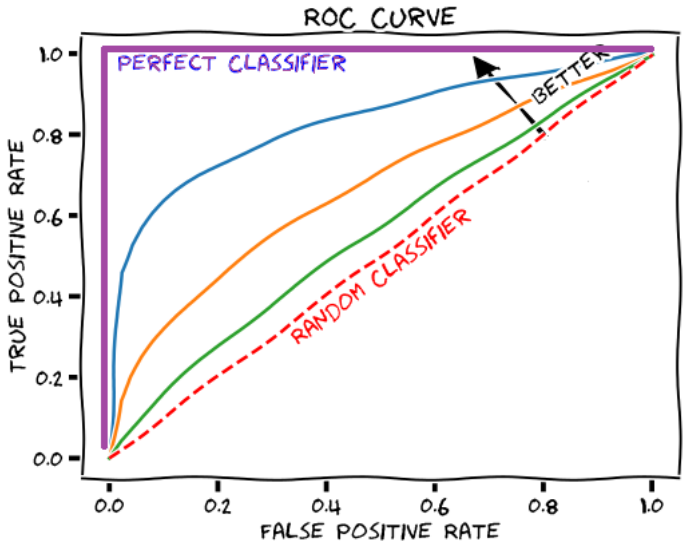


Figure 3.11: ROC Curve [7]

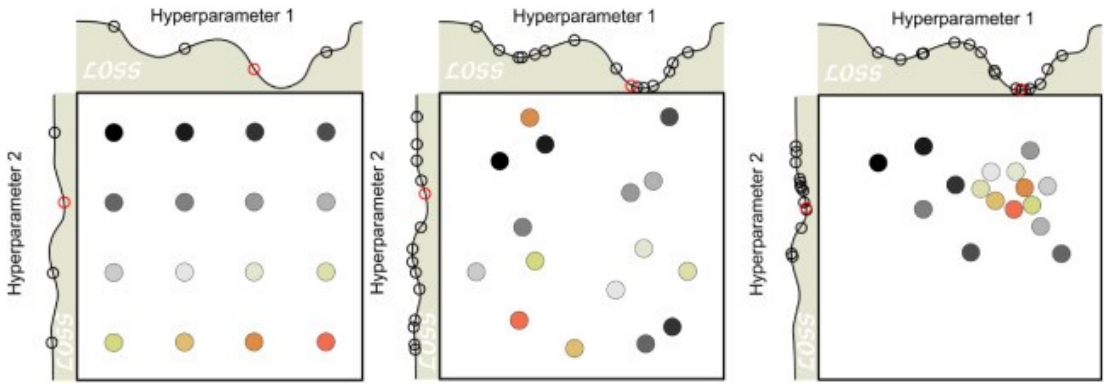
The threshold is the limit at which the predicted probability must exceed in order to be consider as that specific class. Having a high threshold would reduce the FPR but at the same time increasing FNR hence the optimal threshold should allow the classifier to achieve a high TPR while maintaining a low FPR. If the curve were to fall on the diagonal red line, it means the model would not be able to make educated prediction instead it is just randomly predicting. The area under the curve (AUC) of the ROC describes the classifier's ability to correctly differential between each classes. An AUC value of 1 means the classifier is able to predict all the samples correctly while an AUC value of 0 means the classifier would classify all positive classes to be negative and negative classes to be positive. If the AUC value is 0.5, the classifier would not be able to differential between the two different classes.

3.8 Hyperparameter Optimisation

Hyperparameters such as the number of hidden layers, number of nodes in the hidden layer, the learning rate and the number epochs has a influence on the performance of the neural network. In order to achieve the best performing neural network, an optimisation process is needed to find the best combination of hyperparameters for the neural network.

The hyperparameter optimisation process basically loops through different combination of hyperparameters, train and test the neural network to determine the combination of hyperparameter that is able to produce the neural network that has the lowest error.

Manually tuning the hyperparameter of a neural network can be an option to the designer if they are able to draw from their experience which proven combination of hyperparameter can produce a good performing neural network for the problem they are trying to solve. If it is a brand new problem or dataset the designer is working on, then an automated optimisation process is needed since the combination of hyperparameter varies between problems and datasets [40]. To achieve the best performing neural network, all of the network's hyperparameter should be optimised. However this may not be feasible due to computational and time limitation as by increasing the number of hyperparameter that needs to be optimised will exponentially increase the number of hyperparameter combinations that will need to be evaluate to find the best performing combination of hyperparameter. Therefore, only hyperparameters that have significant influence on the network's performance will be optimised and the less influential hyperparameters will be fixed [40].



(a) Grid Search (b) Random Search (c) Bayesian Optimisation

Figure 3.12: Comparison of the three hyperparameter optimisation method [40]

3.8.1 Grid Search

One of the simplest hyperparameter optimisation method is grid search. In grid search every possible hyperparameter combination within the search space is evaluated. For example from Figure 3.12a, there is 4 different values for hyperparameter 1 and 2, therefore a total of 16 different combinations of hyperparameters. Grid search will systemically train the neural network and evaluate its performance for 16 times but with different hyperparameter combination each time to determine which combination yields the best performing network. This method requires the designer to defined a very specific search space where they believe the best performing combination of hyperparameter is located at. If the defined search space is too large, the optimisation process will take a long time to evaluate every combination. Additionally, if the interval between each point of interest in the search space is too large, the optimal combination may fall between two point of interest therefore the optimal solution will not be discovered by using grid search. This is illustrated in Figure 3.12a where the optimal value for hyperparameter 1 lies somewhere between the third and fourth point. Another downside of using grid search is that the number of combination increases exponentially when adding more hyperparameter to optimise. Consequently, this will drastically increase the time needed to evaluate every combinations [40].

3.8.2 Random Search

Another method for hyperparameter optimisation is random search. In random search, different combination of hyperparameters will be randomly sampled from the search space for a predefined amount of trials in order to find the optimal combination. If the specific search space is not known to the designer, this method has a better chance of finding the optimal combination of hyperparameters compare to grid search as the point of interest in random search is not user defined like how grid search is [40]. From Figure 3.12b, it can be seen that point of interest are randomly spread throughout the search space. Since hyperparameters are randomly sampled from the search space, it will require a certain

degree of luck for the best combination of hyperparameter to be sampled. Additionally, there will also be high variance between each search therefore it may not be able to achieve the same solution again. This method is commonly used to discover new combination of hyperparameter which is not know to the designer.

3.8.3 Bayesian Optimisation

The two method which have been describe does not take into account the performance of the previous model when choosing the next combination of hyperparameters since grid search just systemically test all combination while random search just randomly chooses the next combination of hyperparameter. A hyperparameter optimisation method that uses the performance of the previous models to determine the next combination of hyperparameters is Bayesian optimisation. Bayesian optimisation uses a targeted search approach by identifying an area of interest first by taking into account previous combination and its performance score. Once an area of interest has been identified, Bayesian optimisation will focus its search solely in that area only.

Bayesian optimisation will first randomly initial a few random combination of hyperparameter and evaluate their performance to build up a history record. This history record will contain of all the previously evaluated combinations and the performance score associated with each combination. A ” cheaper” probabilistic surrogate model of the objective function is then developed using the history record. This surrogate model is used during the optimisation of the acquisition function [12]. The acquisition function is responsible for identify area of interest within the search space where it believe the best combination of hyperparameter is located at as well as selecting the next combination of hyperparameter to evaluate. The next combination of hyperparameter is selected base on maximising the acquisition function [40]. Once the newly selected combination of hyperparameter is evaluated using the objective function, the hyperparameter combination and performance score will be added into the history record and subsequently the probabilistic surrogate model will then be updated [40]. This process is repeated until either a optimal model has

been achieved or the it has met the maximum number of iterations.

3.9 Neuroevolution

3.9.1 Introduction

Another option to Feedforward Backpropagation neural networks that could possibly deliver accurate result is Neuroevolution neural networks. Neuroevolution neural networks draws its inspirations from biological evolution of human brains where evolutionary algorithms such as Evolutionary Strategy, Evolutionary Programming and Genetic Algorithms [14] are used to optimise the weights, structure and hyperparameters of the neural network [50] by using a evolutionary process to produce better performing individuals. In this paper, the author explored the use of Neuroevolution to optimise the weights of the neural network, however work done by Risi and Stanley [46] show the it is possible to use Neuroevolution to optimise the structure and parameter of a network as well.

Neuroevolution that uses genetic algorithm to optimise the weights and biases for a given neural network structure are population based algorithms. Every individual within the population is a point within the search space [22]. Each individuals within the population is made up of a chromosome and each chromosome contains a set number of genes. The genes are description of that individual in this case the different weights and biases. Individuals compete and exchange information with each other within the population [14].

The first generation of individuals is created by assigning random weights and biases values for each individuals [44]. The individuals in the first generation will then carry out the predefined task and their performance will be measured and ranked using a fitness function. Fitness functions are used to determines how fit or good an individual is at solving the predefined task by calculating a fitness score. This function is normally constructed according to the problem that is being investigated. For example the fitness function for

a binary classification problem can be constructed using equation Equation 3.18, where binary cross entropy (Equation 3.9) will be used to calculate the loss value. While for a regression problems MSE (Equation 3.7) or RMSE (Equation 3.8) could be used to calculate the loss value.

$$\text{Fitness Score} = \frac{1}{Loss} \tag{3.18}$$

The fitness score calculated is then used for the parent selection process as it determines how well an individual is at solving the problem. How the parents are selected is dependent on the selection method used. Once the parents for the next generation has been selected, crossover and mutation of the parents chromosome will be carried out to produce the offspring. This process is repeated multiple times until the stopping criteria is met. The stopping criteria can be defined to be when a solution that has satisfied the requirement has been found or the number of generation generated has met the limit set by the user. Figure 3.13 show the general procedure of using genetic algorithm to optimise the weight and biases of a neural network.

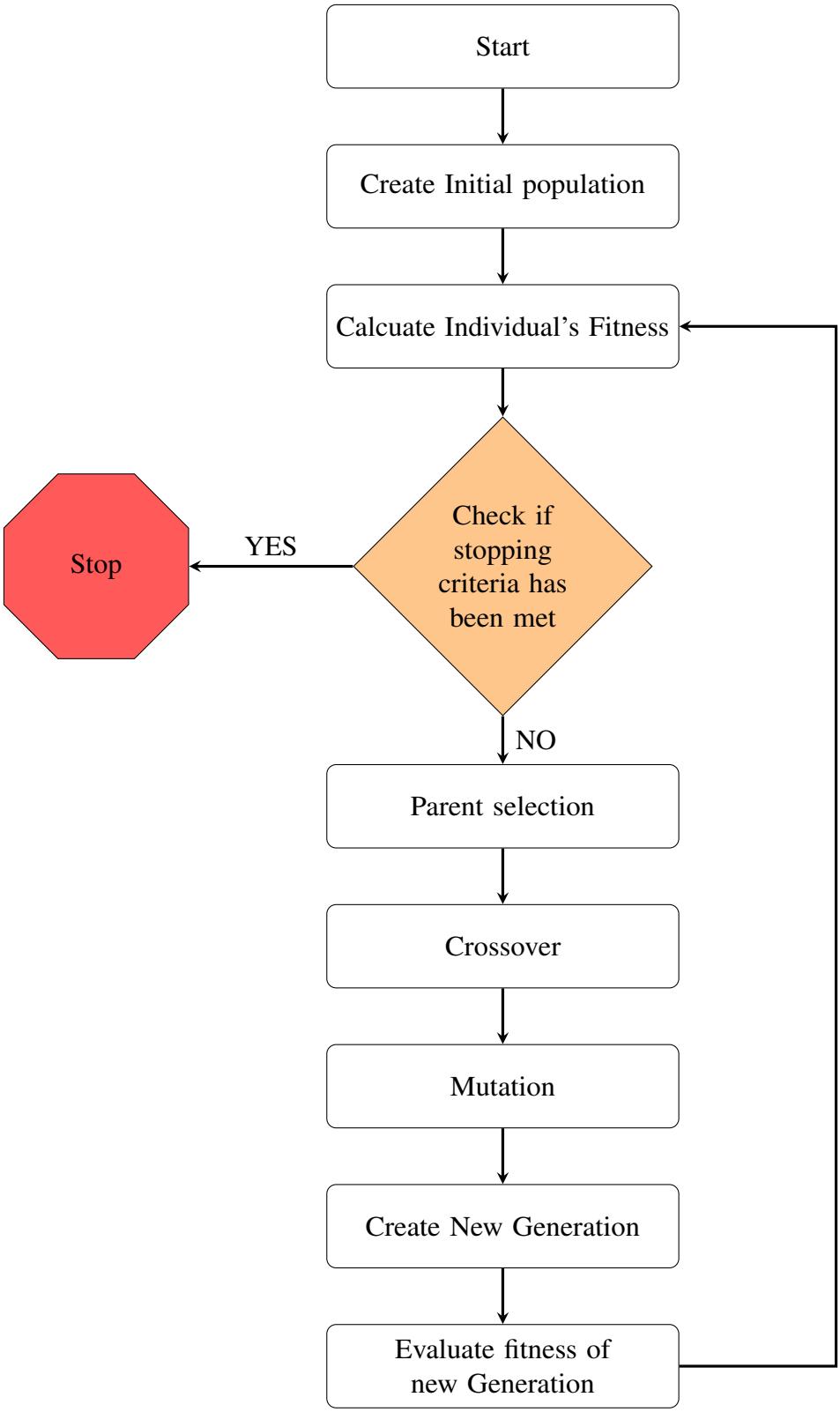


Figure 3.13: Genetic Algorithm Flow Chart

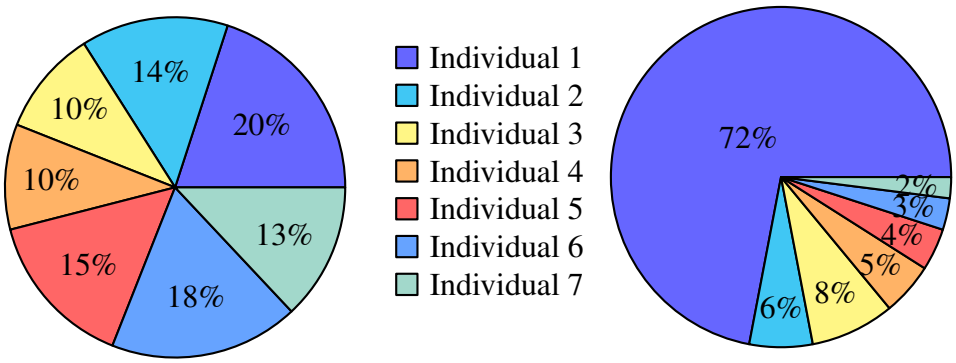
3.9.2 Parent Selection

The objective of the parents selection process is to preserve individuals that have good performance while eliminating individuals that did not perform as well [34], therefore elements that made those individuals to perform better can be passed onto the next generation. It is also beneficial to give individuals that did not perform as well a chance to be selected, as they may consist of some elements which maybe useful for the future generation [34]. As there are various selection methods and each method selects the parents differently, it is therefore important that the selection method used is suitable for the problem that is being investigated [9] so that a good mix of good and bad performing individuals are selected [34]. Using poor selection method can lead to problems such as premature convergence or low convergence rate. Premature convergence will result in the solution found to be suboptimal while low convergence rate will increase the time needed to find the optimal solution [9].

3.9.3 Roulette Wheel Selection

The simplest method is Roulette wheel selection method [34]. In the Roulette wheel selection method, each individual in the population is given a slice of the roulette wheel. The size of the slice of the roulette wheel an individual gets is determine by its fitness score. Individuals with higher fitness score will have a larger portion of the roulette wheel therefore this increases the probability of the wheel stopping at individuals that has fitness score [34]. Figure 3.14a shows an example of how the share of the roulette wheel is distributed among a population size of 7, with Individual 1 having the highest fitness score and Individual 4 having the lowest score. The number of times the wheel spun is determine by the set number of parents. If the number of parents is set to 2 by the user, then the wheel will spin 2 times to select two individuals. This method is the easiest to implement and it does not take up much time [34], however this method does not guarantee that the best individual will be chosen all the time as there is a certain degree of randomness involved. Additionally it is also bias towards individual with high fitness score, therefore

individuals with much higher fitness score compare to the other individuals will dominate the wheel. This will reduce the chances for other individuals to be chosen which may lead to premature convergence and resulting in finding the local optimums [34]. Figure 3.14b shows where the wheel is being dominated by Individual 1.



(a) No Dominant Individuals.

(b) With Dominant Individuals.

Figure 3.14: Roulette Wheel Share Distribution Examples.

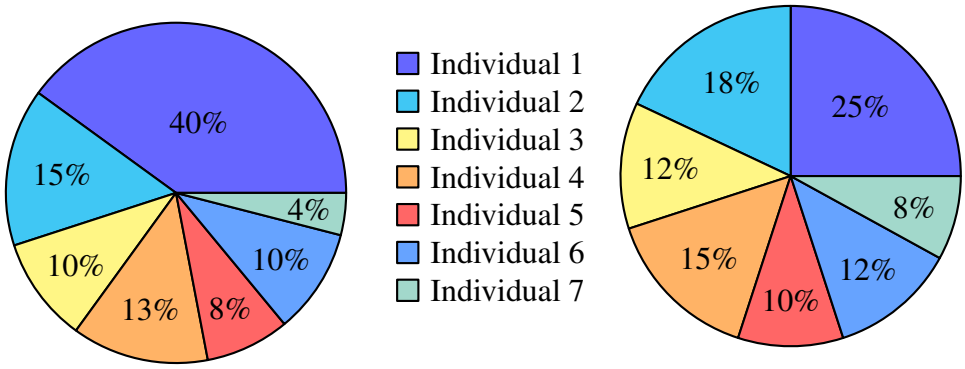
3.9.4 Rank Selection

Another selection method that is commonly used is rank selection method. Rank selection is very similar to Roulette wheel selection in how they select individuals to be the parents for the next generation. The difference between the two method is how to size the each individual on the wheel is determined. Instead of using fitness score to directly determine the size of each individual on the wheel, in rank selection the size of each individual is determined using a selection probability [34]. To determine the selection probability, individuals are first ranked based on their fitness score with rank 1 indicating the worst performing individual and rank N (number of individuals) for the best performing individual [51] . The rank assigned to each individual will be unique even if they have the same fitness score [51]. This rank will then be used to determine the selection probability of each individual by using a function that links the rank of individuals and selection probability together [34]

$$\text{Scaled Rank} = 2 - SP + \left[2(SP-1)\frac{i-1}{N-1} \right] \tag{3.19}$$

Equation 3.19 is the mapping function used in linear rank selection where selective pressure SP is used to control the selection biasness towards good performing individual. The value for SP should be such that $1 \leq SP \leq 2$ and the rate at which the best individual is being sample is equals to SP while the sampling rate for worst individual is $2 - SP$ [34]. i is the rank of that individual and N is the total number of individuals in the population. Since the mapping function used is linear, once the best and worst individual's scaled rank are determined, the rest of the individual's scaled rank within the population can be determined using linear interpolation [34].

Using rank selection method can prevent individuals that has exceptionally high fitness score compare to other individuals to dominate the wheel therefore preventing premature convergence from occurring [34]. This comes at the cost of increase in computational cost as the populations needed to be rank first, then follow by calculating individual scaled rank before the selection process can begin [34]. Additionally, the scaled rank used make it so that there isn't much differences between the best performing individual and the rest of the population hence lowering the convergence rate as well [34]. Figure 3.15 shows how the wheel is distributed between a population size of 7 with 1 dominant individual for Roulette wheel selection and rank selection. It can be seen that Individual 1 takes up a larger portion of the wheel in the Roulette Wheel selection (Figure 3.15a), while using rank selection method (Figure 3.15b), the wheel is much more evenly distributed.



(a) Roulette Wheel Selection. (b) Rank Selection

Figure 3.15: Comparison Between Roulette Wheel Selection and Rank Selection with 1 Dominant individual example

3.9.5 Tournament Selection

Another popular method that is used for parent selection is the Tournament Selection method [9]. The tournament selection method selects the parents for the next generation by hosting tournaments [9]. A number of individuals are randomly selected for each bracket of the tournament where they will compete with each other to be the winner of that bracket by comparing their fitness score and determine who has the higher fitness score. Winner from that bracket will then move on to the next round of the tournament where that winner will compete with another winner from another bracket. This process is repeated until the tournament winner has been determined. The tournament winner will then be included in mating pool with other tournament winners [9].

This selection method is less likely to be taken over by good performing individuals as all individuals are given a chance to be chosen to compete in the tournament regardless of their fitness score [34]. However, this comes at a cost of lower convergence rate, therefore increase the time needed to find the optimal solution [34]. Additionally having tournament size that is too large will lead to the reduction in diversity as fewer individuals will be able to make it into the mating pool [34]. The loss of diversity can also be a result of individual not getting selected for the tournament or individual got selected for the tournament but did not win the tournament [34]. Figure 3.16 shows an example of a tournament selection method

where 4 individuals are competing with each other to become the winner of the tournament.

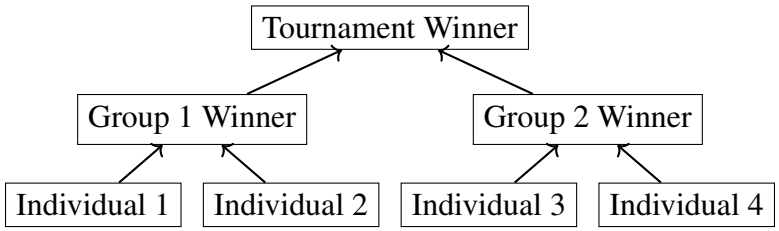


Figure 3.16: Tournament Selection Example

3.9.6 Crossover

After the number of parents required has been selected using the chosen selection method, crossover of the parents chromosome will be carried out to produce new individuals for the next generation [24]. The purpose of crossover is to have a diverse population and this can be achieve by exchanging the parent's chromosome with each other to produce a offspring that will outperform the parents. This is done by randomly selecting a or multiple points along the parents chromosome and these points will determine where the parents will exchange part of their chromosome with each other [24].

How the chromosome information and the number of points that will be chosen is determine by the type of crossover that is being used. The simplest type of crossover is single-point crossover. In single point crossover (Figure 3.17), only one points is randomly selected along the parents chromosome where the parents will split their chromosome and exchange with each other [24]. Another type of crossover is multi-point crossover (Figure 3.18), where instead of choosing only one point, multiple points will be randomly chosen to split and exchange the parents' chromosome at those points to produce the offspring. Multi-point crossover can prevent offspring to have the same chromosome as their parents therefore genes are preserved for the future generation instead of the chromosomes.

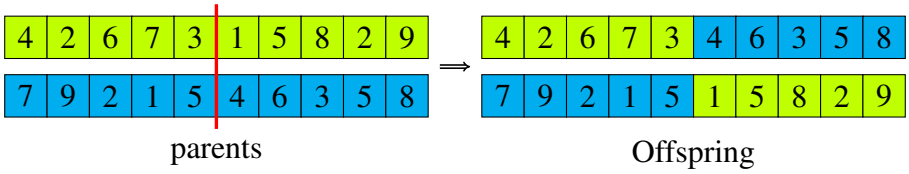


Figure 3.17: Single-Point Crossover Example

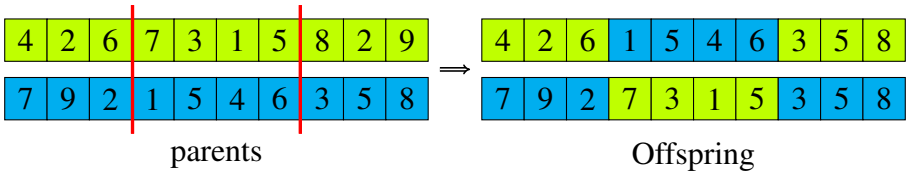


Figure 3.18: Multi-Point Crossover Example

3.9.7 Mutation

If only crossover was used to produce the offspring, the genetic algorithm may get stuck at the local minimum hence why the mutation process is needed [24]. The mutation process promotes diversity within the population by randomly altering one or more genes in the newly constructed chromosome to introduce new combination of genes [39]. The rate of mutation is determined using Equation 3.20, where L is the length of the chromosome [39]. The chosen mutation rate should be appropriate as increasing the mutation rate would increase the search area therefore increasing the time needed to find the optimal solution. Additionally if the mutation rate is too high, the genetic algorithm will behave like a random search [24], therefore it may not be able to find the optimal solution. On the other hand if the mutation rate is too low, it will also cause premature convergence [24].

$$\text{Mutation Rate} = \frac{1}{L} \tag{3.20}$$

How the mutation is done and how many genes are mutated is determined by the mutation method used. Bit Flip Mutation, Swap Mutation and Scramble Mutation are some of the

commonly used mutation methods. Bit Flip Mutation is used for binary chromosomes, where a or multiple genes are flip from 1 to 0 or 0 to 1. In Swap Mutation, a pair of genes will be randomly selected along the chromosome and the pair will then swap position with each other. Scramble Mutation is done by segmenting a portion of the chromosome and the genes within this segment will randomly swap positions amongst themselves.

Chapter 4

Application

4.1 Rolls Royce Trent-900

Figure 4.1: Airbus A380 [3]

The Trent-900 engine civil turbofan engine that is manufactured by Rolls Royce. This engine is used to power the largest passenger aircraft, the Airbus A380 (Figure 4.1). The A380 is a wide-body aircraft that is capable of carrying up to a maximum of 853 passengers [2]. Even-though the A380 program has ended since 2021, the aircraft still continues to take to the sky with approximately 129 A380s still flying during the last week of June this

year [49]. This is approximately 50% of the total delivered A380s. However, this aircraft has a high operating cost due to the fact that it uses 4 engines to generate the required thrust therefore any degradation in the engine would further increase the operating cost and may push operators to stop using the aircraft as it becomes unprofitable to operate.

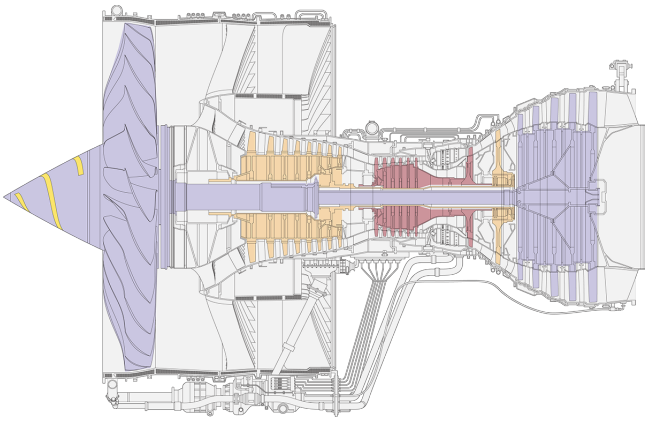


Figure 4.2: Trent-900 Cross sectional diagram [6]

Take Off Condition	Net Thrust	341.41 kN
	Intake Mass Flow Rate Range	1204-1245 kg/s
	BPR Range	8.5 - 8.7
	HPC Exit Temperature	953K
Cruise Condition	Altitude	35000 ft
	Mach number	0.85
	Net Thrust	65.40 kN
	SFC	14.67 g/Ns

Table 4.1: Published Trent-900 data @ ISA+15°C [16] [8]

The Trent-900 is a high bypass turbofan engine that uses a 3 spool configuration with an OPR of 38.5. The engine's fan has a dimater of 2.946m, a total engine length of 5.4775m and it has a dry weight of 6246kg [16]. The engine fan consist of 24 double curved leading edge titanium-alloy blades [8]. The purpose of the double curved leading edge is to reduce the mach number of the flow and to prevent stalling at the blade tip [8]. With this large intake diameter, the engine has a intake mass flow rate range of 1204 kg/s to 1245 kg/s and a BPR range of 8.5 to 8.7 at sea level. The compressor has a total of 14 stages in which 8 stages makes up the intermediate pressure compressor and 6 stages makes up the

high pressure compressor. Air for wing anti-icing and aircraft's environmental control system are from different stages of the compressor depending on the current flight phase the aircraft is in. During take-off, climb and cruise the air is bled from the exit of the IPC while during descent or ground idle, the air is bled from HPC exit [16]. The transition from IPC bleed to HPC bleed when the aircraft starts to descent happens when the aircraft is in its holding pattern. To achieve a higher aerodynamic efficiency while at the same time reduce the jet noise of the engine, the HP shaft rotates in the opposite direction of the LP shaft [8]. The flow from the HPC exit will then be fed into the combustion system. The combustion system consist of 20 burners with tiled chambers [8]. After the combustor is the turbine stages. The Trent-900 has a total of 7 turbine stages. The first stage is the HPT, which is used to drive the HP shaft that is connected to the HPC. The second stage is the IPT. The IPT is used to drive the IP shaft connected to the IPC and the last 5 stages of the turbine is used to drive the fan.

4.2 Engine modelling software

4.2.1 PYTHIA

To generate training, testing and validation data for the artificial neural network, a engine model was developed using a engine modelling software named PYTHIA. PYTHIA is a software developed at Cranfield University and it is capable of engine performance simulation, diagnostics and life monitoring [30]. Another in-house developed package called TURBOMATCH is also integrated into PYTHIA for gas turbine performance simulation.

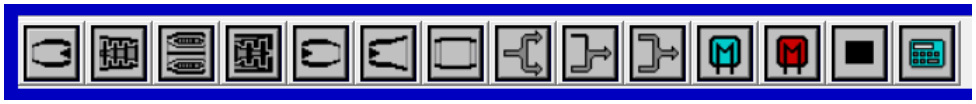


Figure 4.3: Engine Component Brick

In PYTHIA, engine components are represented using bricks. These bricks are pre-programmed to calculate the thermodynamic process for the respective component they

represent. For a user to construct their desired engine model in PYTHIA, the user would have to select and arrange the brick it such a way that it matches with the configuration of the engine they are modelling. Once all the necessary bricks have been, the user would need to define the station vector for each brick. Station vectors provides the flow path's information by connecting all the bricks together to form the engine model. The user would also need to fill in the necessary brick data for each brick. Brick data are component related data. For example the some of the brick data needed for the compressor brick would be the design pressure ratio, the compressor map and shaft number. The component's map will determine the performance and operational limit of the component therefore it is crucial that the map selected will be suitable for the component's operability and at the same time produce desired performance. Different brick would require different component related data. Once all the brick data and station vector has been defined, the engine model is now complete and can be used for design point (DP) performance simulation.

If the result of the engine model obtained from the DP performance simulation has big difference when comparing with published data, PYTHIA also offers the capability of performance adaptation. Performance adaptation improves the accuracy of the engine model by calculating the error between the targeted performance parameters and the performance result of the initial engine model. Targeted performance parameters are the published performance data of the real engine. Once the error is calculated, adjustments to to-be-adapted parameters are made. To-be-adapted parameters includes the turbine entry temperature, intake mass flow rate, compressor pressure ratio, compressor isentropic efficiency and etc [30]. Once the to-be-adapted parameters have been adjusted, the performance of the adapted model will be calculated and then evaluated again to determine the new error. The number of time is process is carried out depends on the chosen adaptation method. If linear adaptation is used, this process will only be done once. On the other hand, if non-linear adaptation method is used, this process will repeat multiple iterations until the best performing engine model is achieved [30]. Once the result of the engine model from DP simulation matches or is close to the published data, the next step

is to carry out off-design (OD) performance simulation. To simulate for OD performance, a handle must be selected first. The handle represents the parameter used to control the power/thrust generated by the engine during operation. This can either be the rotational speed of the shaft or the turbine entry temperature.

Once a good performing engine model has been achieved, the model can be used for degradation study. PYTHIA offers the capability of implanting degradation into the component by altering the component's flow capacity and efficiency to simulate the effect of different component degradation. Once the degradation has be implanted into the component, PYTHIA can calculate the performance of the degraded engine. PYTHIA also offers the capability of generating large amount of degraded and clean engine data which is needed when developing a ANN for gas turbine diagnostics. To generate degraded engine data, user would need to defined the minimum and maximum flow capacity and efficiency degradation for each component as well as the interval between each degradation level. PYTHIA will then generate the degraded engine measurement data for every possible combination of flow capacity and efficiency within the predefined range set by the user. Therefore the smaller the interval between each value of flow capacity and efficiency is, the greater the number of combination of flow capacity and efficiency will be which also mean the larger the generated dataset will be. PYTHIA is also able to implant noise and bias into the generated measurement data to make the data more realistic since the sensors in a gas turbine are susceptible to measurement errors due to its harsh operating environment.

4.3 Development of engine model

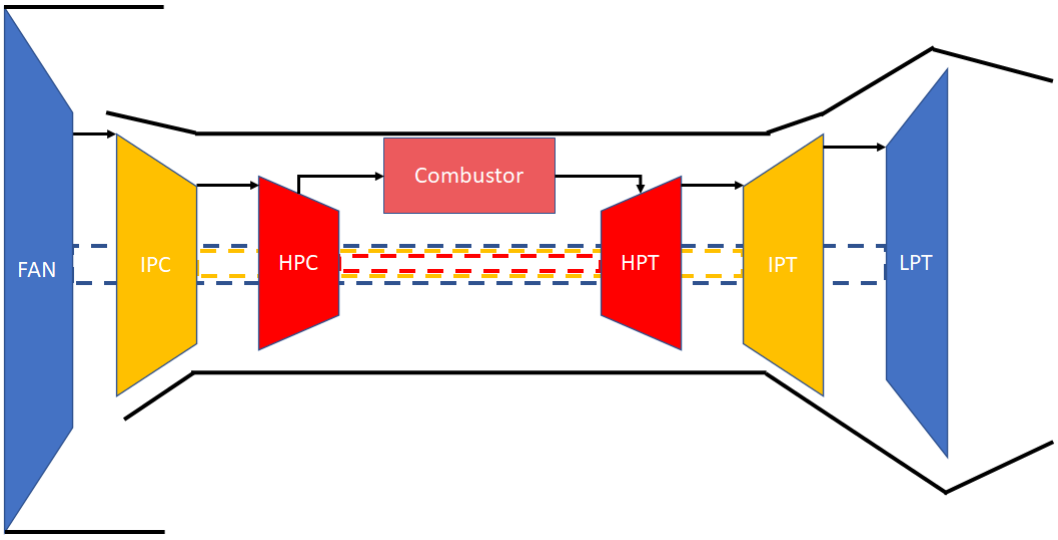


Figure 4.4: Trent-900 Schematic

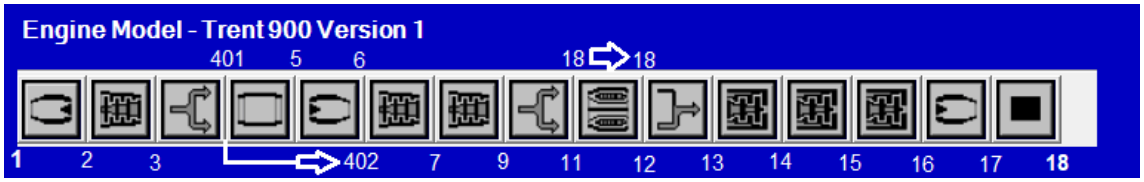


Figure 4.5: Trent-900 Model in PYTHIA (refer to Appendix B for brick name and details on the component each brick represents)

A Trent-900 engine model was developed in PYTHIA using the data in Table 4.1 which was collected from various sources. A schematic of the engine seen in Figure 4.4 was derived from Figure 4.2. The engine schematic was then used to developed the engine model in PYTHIA. Figure 4.5 describes the engine model developed in PYTHIA, with the numbers in white indicating the station vectors for each brick. Since not all of the brick data such as the intake mass flow rate, TET and HPC pressure ratio was not available in the public domain, a parametric study was carried out. The optimal fan pressure ratio was determined by varying the fan pressure ratio from 1.2 to 1.8 to calculate the net thrust and sfc for each fan pressure ratio. Then a graph of fan pressure ratio vs net thrust and sfc is plotted to find the optimal value. From Figure 4.6 it can be seen that the optimal fan

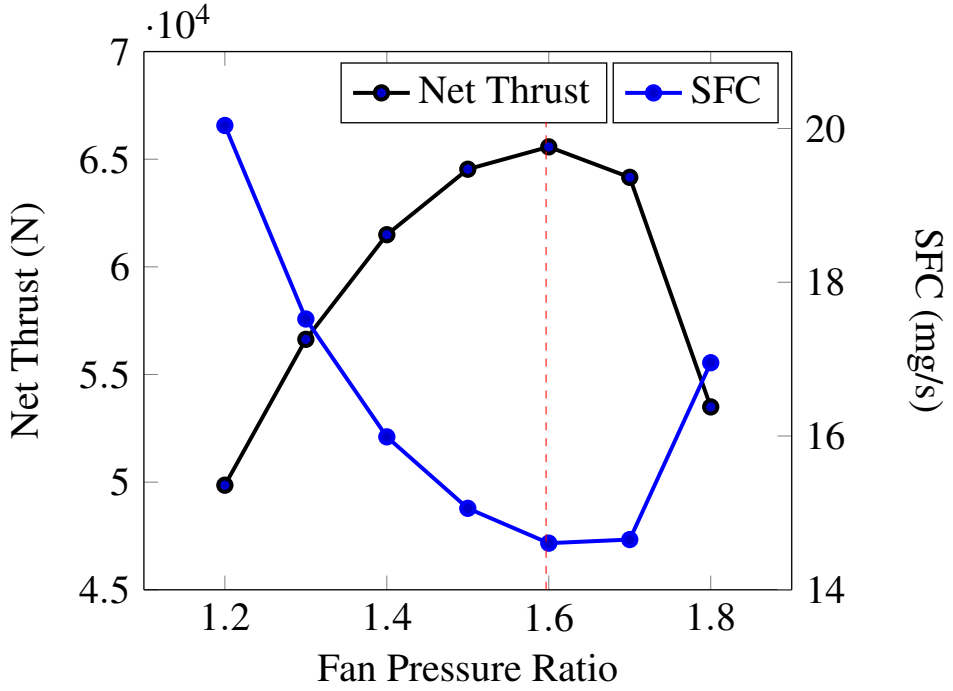


Figure 4.6: Optimal Fan Pressure Ratio for BPR of 8.6

pressure ratio is at 1.6, where the net thrust is the highest and at the same time sfc is the lowest.

Once the initial fan pressure ratio has been determine the remaining pressure ratio which is 24.0625 was split between the HPC and IPC. Since the HPC would have a higher pressure ratio, the initial HPC pressure ratio was 6 and subsequently the initial IPC pressure ratio was 4.0104. The initial isentropic efficiency for both of the compressor was set to 88% and the fan efficiency was 90%. The combustor has a efficiency of 99% and a pressure loss of 5%. To cool the HPT, 15% of cooler air is bled from the HPC exit. To determine the intake mass flow rate and TET, a multiple design points study was done by varying the TET and intake mass flow rate to find the TET and intake mass flow rate that will produce the net thrust and sfc that matches closely to the published data. Once an initial engine model has been achieved, non linear DP performance adaptation was done to improve the accuracy of the model. From Table 4.2, it can be seen that even though the initial engine model was close to the published data, the initial model have a higher SFC. After applying non-linear adaptation, the final model was able to achieve the same sfc as the published data and the

difference in net thrust between the adapted model and published data is smaller. For the full text output file produced by PYTHIA, refer to Appendix A.

Variables	Published Data	Initial Model	Adapted Model
Altitude (ft)	35000	35000	35000
Mach	0.85	0.85	0.85
Net Thrust (kN)	65.4	65.6	65.38
SFC (mg/s)	14.67	15.63	14.67
TET (K)		1580	1537.3
Mass Flow Rate (kg/s)		540	535.59

Table 4.2: Engine model Comparison

4.4 Degraded Engine Performance

To understand how different component degradation affects the various engine performance parameters, a degradation study was done by altering the efficiency and flow capacity of each component in PYTHIA to simulate the different type of degradation describes in section 2.2. Figure 4.7 shows the effect of fan and compressor flow capacity and efficiency degradation of -1% with varying ISA temperature deviation. Appendix C has figures that describes the effect of compressor and fan flow capacity and efficiency degradation on other engine parameters as well.

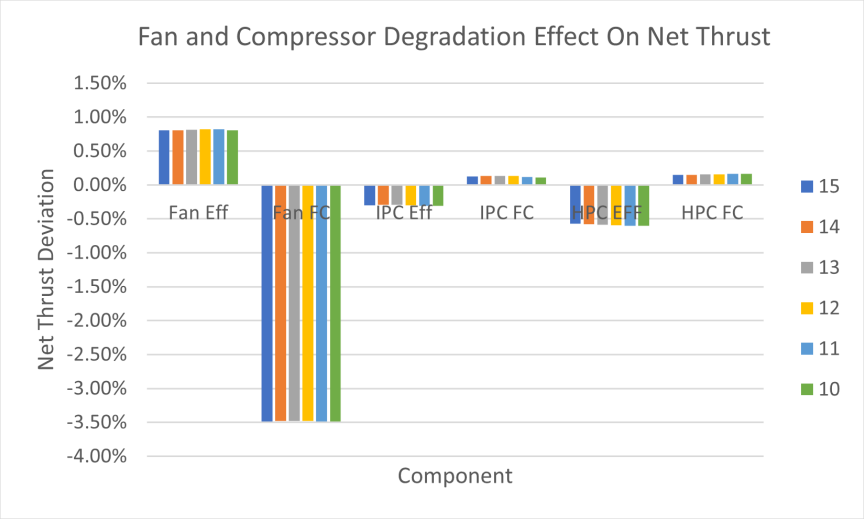


Figure 4.7: Net Thrust Deviation Due to Compressor and Fan Degradation

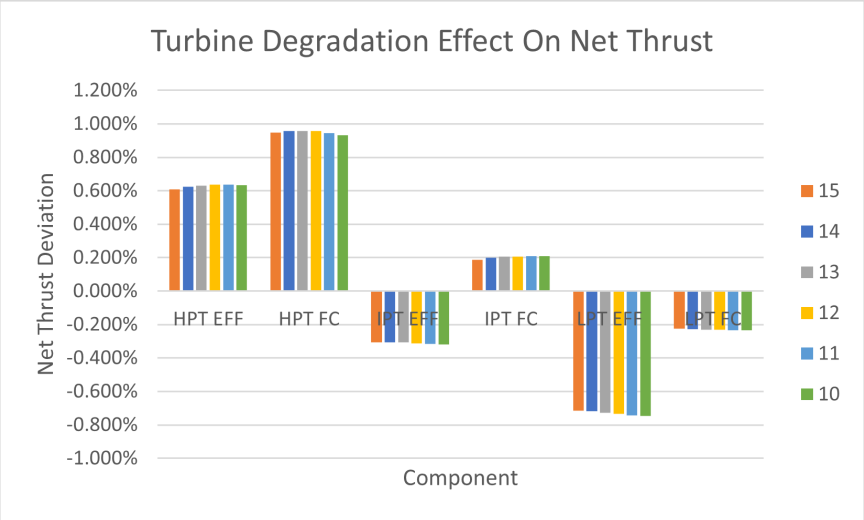


Figure 4.8: Net Thrust Deviation Due to Turbine Degradation

A degradation study was also done to determine the effect of turbine degradation on the engine. Figure 4.8 illustrates the deviation in net thrust caused by -1% flow capacity and efficiency degradation in each turbine. To view the affect turbine degradation has on other engine parameters refer to Appendix C. A correlation heatmap was also illustrated in Appendix D to determine the correlation between each parameter. If two parameter have a positive correlation, when one of the parameter increases, the other parameter will increase as well. On the other hand, when two parameter have a negative correlation, when one of the parameter increases, the other parameter will decrease. The correlation value indicates how closely correlated the two parameter are with each other. When two parameter has a correlation value of 1, it means that the two parameter has a strong positive correlation. On the other hand when two parameter has a correlation value of -1, it means that the two parameter has a strong negative correlation. For example SFC and CN1 as a correlation value of approximately -0.75 therefore they are highly negatively correlated. The correlation heatmap can also be used to determine which measurable variable can be used by the neural network to predict engine degradation such as shaft rotational speed since it is correlated with net thrust, therefore any changes to shaft rotational speed will also affect net thrust.

4.5 Data Generation

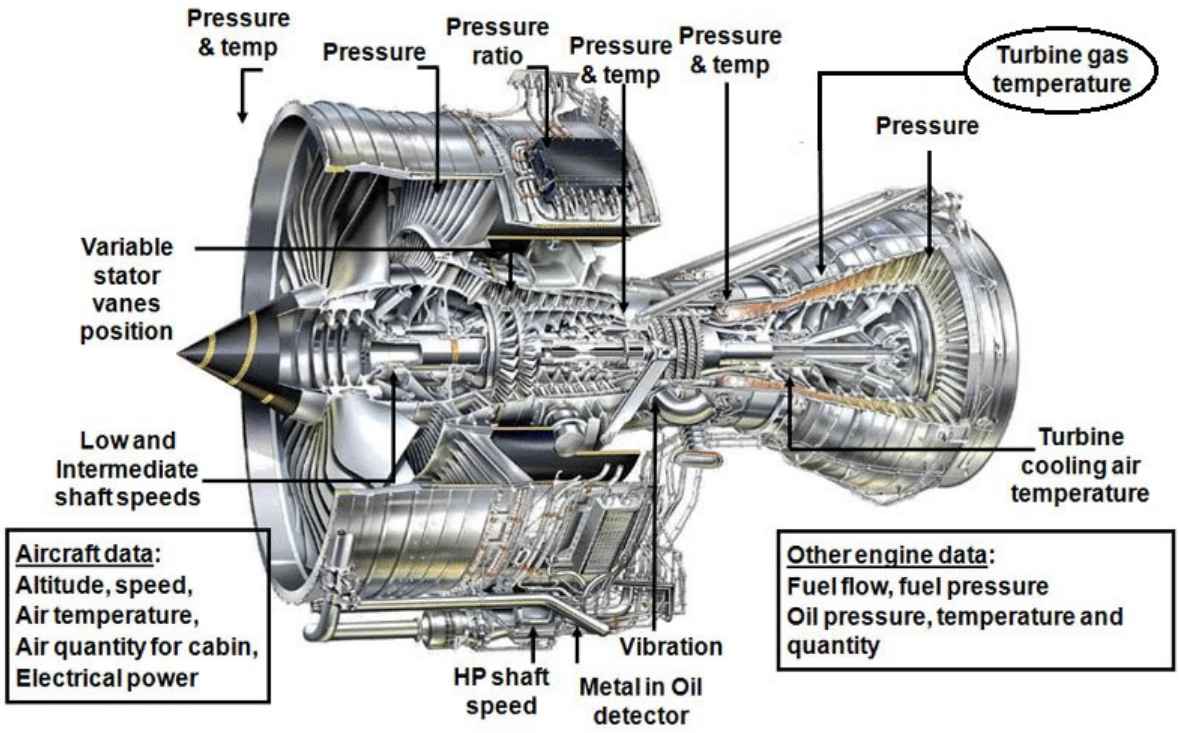


Figure 4.9: Trent-900 EHM Sensor Location [56]

After an engine model that is able to produce performance data similar to the published data is achieved, the model is then used to generate degraded and clean data which will be used to train the ANN. In order to generated degraded and clean engine data, the location and type of sensors available on board the engine has to be determine first since there is only a limited number of sensors that is installed in the engine therefore it is unrealistic to use all the measurement available in the engine model to train the ANN. Figure 4.9 illustrate the different sensors installed in a Trent-900 which is used to collect measurement data for engine health monitoring. Another consideration when choosing which measurement data to collect is the measurement availability in PYTHIA since the data for this study will be source from PYTHIA only. Using Figure 4.9 and also taking into account the measurement availability in PYTHIA, the dataset which will be used to develop the ANN will consist of the following engine measurements:

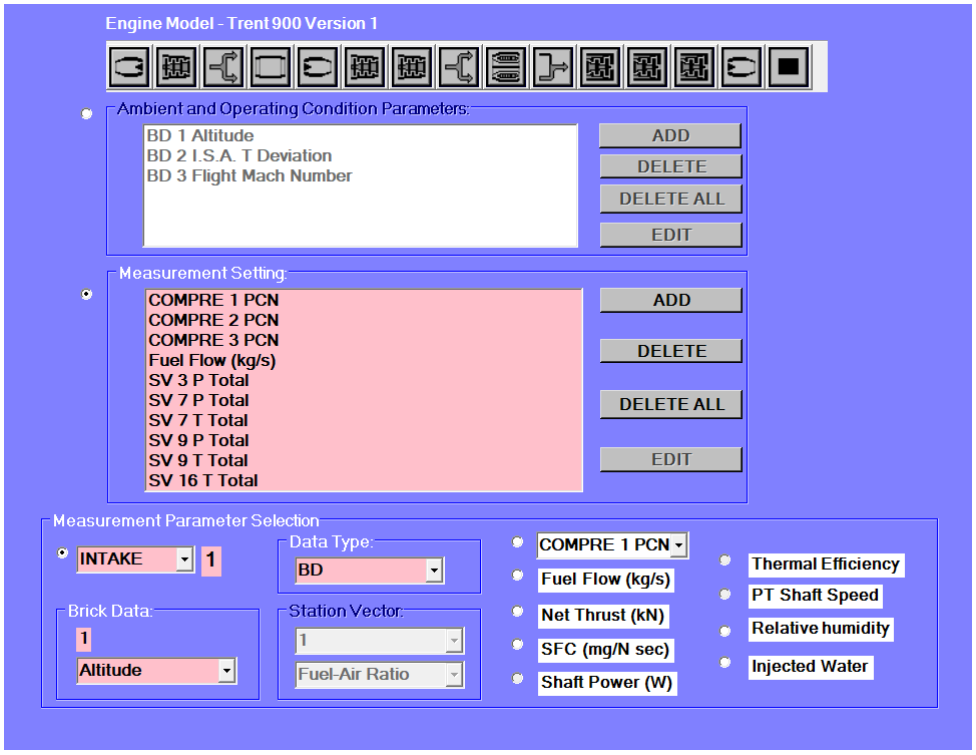


Figure 4.10: PYTHIA measurement selection setup

1. LP Shaft Speed (COMP 1 PCN)
2. IP Shaft Speed (COMP 2 PCN)
3. HP Shaft Speed (COMP 3 PCN)
4. Fuel Flow
5. Fan Exit Pressure (P Total 3)
6. IPC Exit Pressure (P Total 7)
7. IPC Exit Temperature (T Total 7)
8. HPC Exit Pressure (P Total 9)
9. HPC Exit Temperature (T Total 9)
10. LPT Exit Temperature (T Total 16)

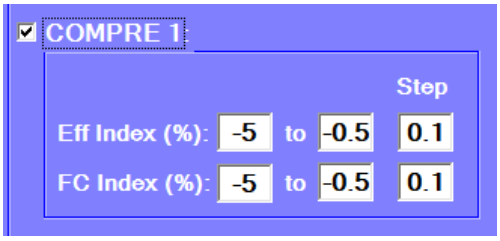


Figure 4.11: Flow Capacity and Efficiency Degradation Range

Component	Degradation Range	Steps	Size of Data	Target Output
Clean	-	-	250000	0
Fan	Γ and η range from -5 to -0.5	0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5
IPC	Γ and η range from -5 to -0.5	0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5
HPC	Γ and η range from -5 to -0.5	0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5
HPT	η range from -5 to 5 Γ range from -5 to 5	0.1 0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5 -5,-4.9,-4.8, $\cdots$ 5
IPT	η range from -5 to 5 Γ range from -5 to 5	0.1 0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5 -5,-4.9,-4.8, $\cdots$ 5
LPT	η range from -5 to 5 Γ range from -5 to 5	0.1 0.1	105800	-5,-4.9,-4.8, $\cdots$ -0.5 -5,-4.9,-4.8, $\cdots$ 5

Table 4.3: Training, testing and validation data generated from PYTHIA

Once the measurement setting has been setup, the next step is to set the flow capacity and efficiency degradation range and the steps between each degradation value within the range for each component. PYTHIA will then generate a list of different flow capacity and efficiency values within the range set. Figure 4.11 is the degradation setting for the fan. The flow capacity and efficiency degradation ranges from -5 to -0.5 with steps of 0.1 therefore there is 46 different efficiency and flow capacity degradation level, allowing 2025 unique combinations of flow capacity and efficiency. The number of measurement and target sample generated for each unique combination of flow capacity and efficiency is set to 50, therefore making the total number of sample for each component to be 105800. By having a large amount of noisy sample for each unique combination of flow capacity and efficiency, it helps to increase the robustness of the neural network as the measurement input is slightly different each time therefore the neural network will not be able to memorise from the training samples. Table 4.3 contains all the information about the degradation setting used to generate measurement and target samples for each component.

Since the measurement and target generated by PYTHIA for each component are separate text files totalling up to 20 different text files, a python script was developed to automatically combine all the text file together to create one big dataset. This was achieving by using a naming convention for the text output files. For example the name for the fan measurement and target text output file is ”Fan Degradation Measurement” and ”Fan Degradation Target”. From the file name, the code would be able to tell if the file it is reading is the target or measurement. In the process of combining the text files together, two different type of targets were also added to the dataset as well. This is to enable the dataset to be used for binary and multi-classification neural networks as the selected target generated by PYTHIA is only suitable for regression neural networks. The target for binary classification is 1 and 0. This was added by check if the file name contains the word ”clean”. If it does contains the word ”clean” the target output will be 0 if it does not the target output will be 1. For multi-classification problem, each component must have its own label. This is done by assigning a unique number for each component. The code then determines which component does the data belongs to by referring to the file name and assign the corresponding class label. Refer to Appendix E for the full code. The data is also normalised before training to prevent big variation in the range values between each features.

Network Type		Target
Binary Classification	Clean	0
	Degraded	1
Multi Classification	Fan	0
	HPC	1
	HPT	2
	IPC	3
	IPT	4
	LPT	5
Regression	Flow Capacity	-5 to 5
	Efficiency	-5 to -0.5

Table 4.4: Neural Network Targets

4.6 Development and Result of Nested Neural Network

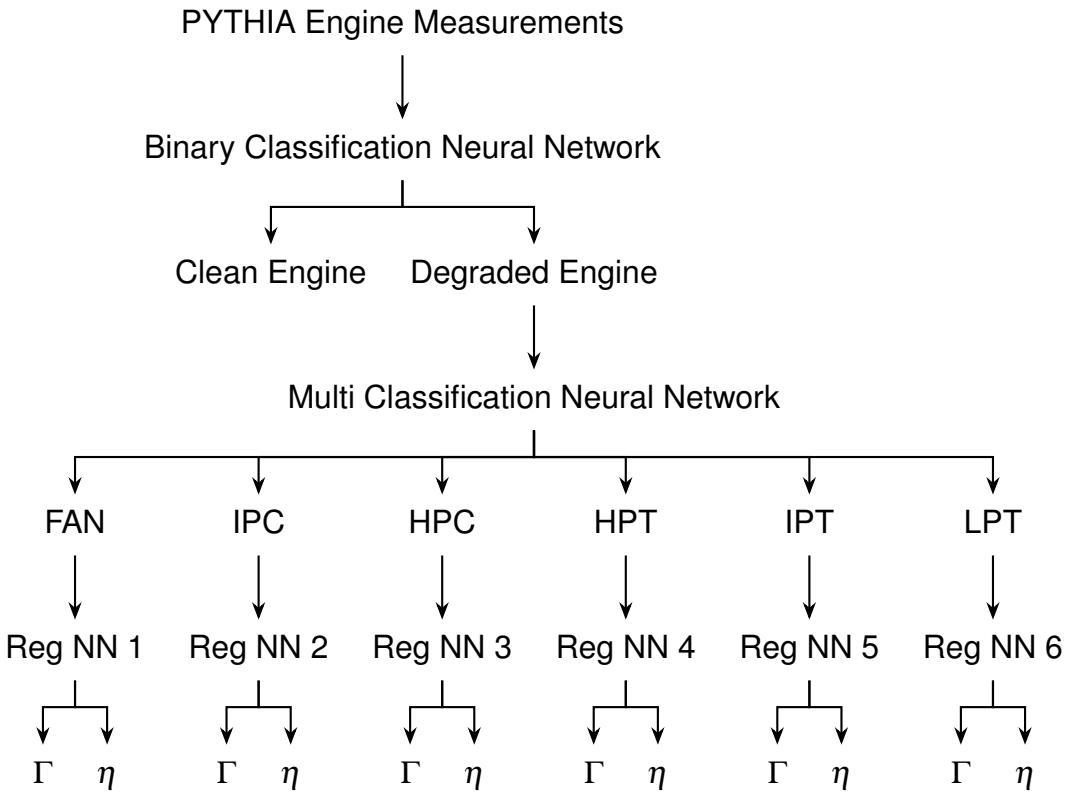


Figure 4.12: Nested Neural Network Architecture

Once the data generation process has complete, the development of the diagnostics system for the Trent-900 engine model can begin. Due to the complex nature of diagnosing a gas turbine, it means that one neural network will not be sufficient for the task explained in section 3.2, therefore a nested neural network was used. Figure 4.12 is the architecture of the nested neural network use for the diagnostic system. It consist of 1 binary classification neural network follow by a multi-classification neural network then 6 regression neural networks for each component. The purpose of the binary classification neural network is to determine if the engine is clean or degraded. If the engine is clean the diagnostic process will stop here, if the binary classification neural network predicts the engine to be degraded, the measurements is then passed onto the multi-classification neural network. The multi-classification neural network will then isolate the component that is degraded

before using a regression neural network to predict the component's flow capacity and efficiency degradation level. Multi component and sensor degradation was not explored in this thesis.

4.6.1 Keras & TensorFlow Library

The nested neural network was developed using the TensorFlow library in python. This library is an open source machine learning library developed by Google Brain team and it is widely used in large companies such as GE, NASA, Airbus and many more [35] due to its flexibility, high performance level [53] and a user-friendly API named Keras. The Keras API also uses other machine learning library such as Theano and MXNet. The Keras API was develop to be concise, easily written and readable [53], therefore it allows beginners to learn more easily and it also allows user to develop and deploy their machine learning models within few lines of codes. The model created using Keras are all made up of individual building blocks therefore this allows the individual building blocks to be combined in various different ways to different models [53].

4.6.2 Binary Classification Neural Network

The first network in the nested neural network is a binary classification neural network. The purpose of the binary classification neural network is to determine if the input engine measurement belongs to a clean or degraded engine. A dataset with 857500 noisy samples and target values of 1 and 0 was used to train, test and validate the binary classification neural network. 70% of the samples from the dataset were used to training and the other 30% was split equally for testing and validating the neural network. The validating samples are used during the training process to evaluate the models performance while the training samples are used once the network has finished training. Table 4.5 describes the size of the dataset used for training,testing and validating the binary classification neural network as well as the corresponding target for each class.

Engine Status	Total	Training	Testing	Validating	Target
Clean	250000	175000	37500	37500	0
Degraded	634800	444360	95220	95220	1
Total	884800	619360	132720	132720	

Table 4.5: Data Distribution for Binary Classification Neural Network

Since there isn't any proven good performing neural network architecture for differentiating clean and degraded engine measurements for the Trent-900, the first architecture for the binary classification neural network was randomly chosen to be 10-15-2. The first network only consist of 3 layers, the input layer, a hidden layer and output layer. The nodes in the hidden layer uses a rectified linear unit activation function to transform its output and the nodes in the output layer uses a softmax activation function therefore the output from the binary classification neural network is the predicted probability for each class. The loss function used for the binary classification is the binary cross-entropy.

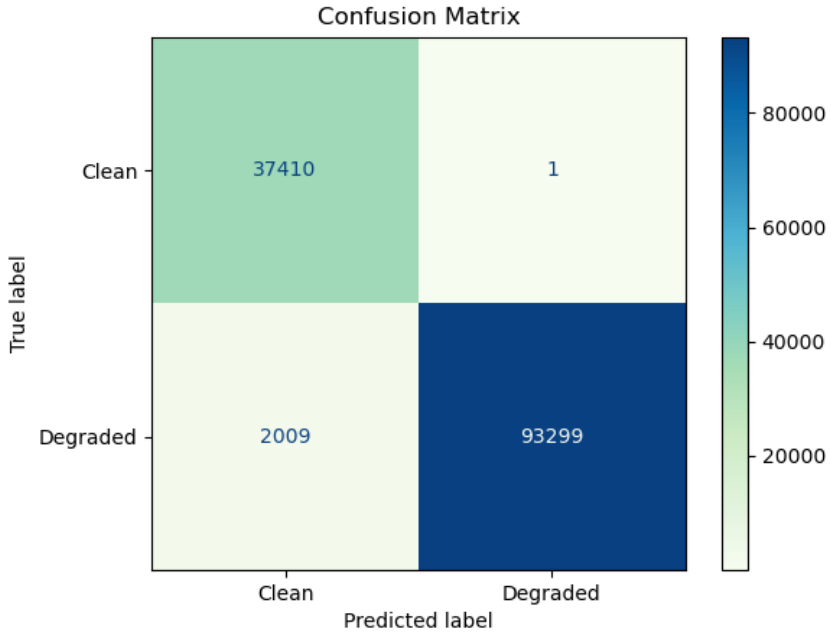


Figure 4.13: Initial Binary Classification Confusion Matrix

After training, testing and validating the 10-15-2 binary classification neural network it was able to achieve an average accuracy of 98.4% and a loss value of 0.0499. From looking at these two performance metric, it can be said that the performance of the network is

good. However by evaluating the confusion matrix (Figure 4.13) of the network, there is a difference in the network's accuracy between the two classes. From the confusion matrix (Figure 4.13), the calculated accuracy for the clean class is 99.9% while the calculated accuracy the degraded class is only 97.8%. The difference in accuracy between the two class could be down to the architecture of the network not being optimal since it was randomly chosen therefore a hyperparamter optimisation was carried out. Manual tuning and grid search is not viable since the area in the search space where the best solution is located at is not known to the author therefore the first hyperparameter optimisation process will be done out using random search.

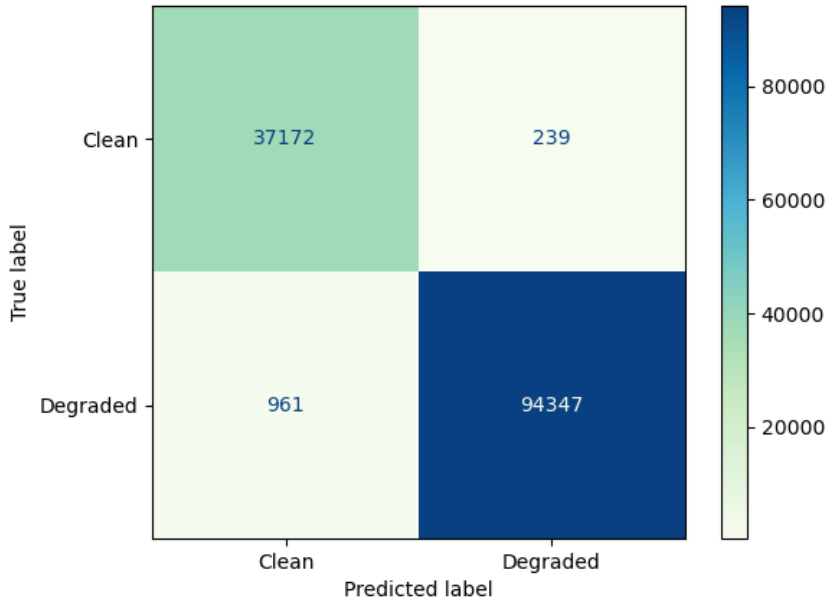


Figure 4.14: Random Search Binary Classification Confusion Matrix

During the random search hyperparameter optimisation process, 100 different combination of hyperparameters were randomly chosen and evaluated. The random search found a network with an architecture of 10-18-26-20-2 to be the best performing network. This network have an average accuracy of 99.1% and a loss value of 0.031, therefore a better performing network compared to the initial network. By evaluating the confusion matrix in Figure 4.14, the accuracy of the new network for the degraded class improved by

1.1% but at a cost of 0.5% reduction in accuracy for the clean class. Even though the performance of the network found by random search is an improvement from the first 10-15-2 network, it may not be the optimal network since random search just randomly chooses the combination of hyperparameter therefore to achieve the optimal combination of hyperparameter, Bayesian optimisation was carried out as well. Bayesian optimisation will use a systematic approach explained in subsection 3.8.3 to search in the search space 100 times to determine for the best combination of hyperparameters. From Figure 4.15, it can be observed that the Bayesian optimisation searches the search space in more specific area while random search is spread throughout the search space randomly. The Bayesian optimisation found a network with an architecture of 10-23-30-10-10-2 to be the optimal solution. This network has an average accuracy of 99.4% and a loss value of 0.021 therefore it has the best performance amongst the three network developed using different approaches. Additionally, it has the best accuracy for the degraded class while still maintaining a 99.8% accuracy for the clean class therefore this network is chosen to be used for binary classification. Refer to Table 4.6 for performance comparison between the three network.

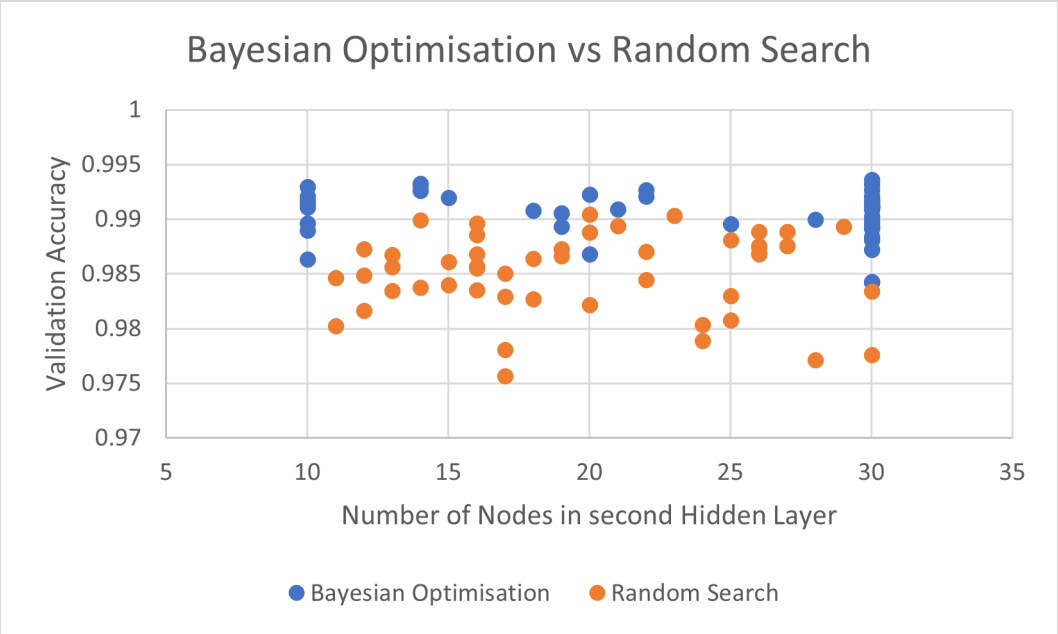


Figure 4.15: Bayesian Optimisation vs Random Search search pattern

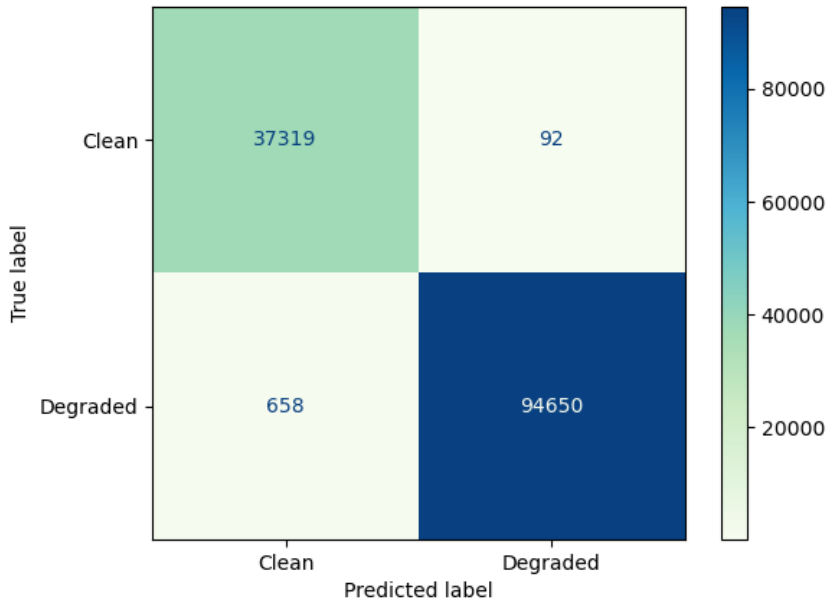


Figure 4.16: Bayesian Optimisation Binary Classification Confusion Matrix

	Initial	Random Search	Bayesian
Network Architecture	10-15-2	10-18-26-20-2	10-23-30-10-10-2
Activation Function	ReLU	ReLU	ReLU
Epochs	25	25	25
Tunning Iteration	0	100	100
Average Test Accuracy	98.4%	99.1%	99.4%
Binary Cross-entropy	0.0499	0.031	0.021
Clean Class Accuracy	99.9%	99.4%	99.8%
Degraded Class Accuracy	97.8%	98.9%	99.3%

Table 4.6: Binary Classification Network Comparison

4.6.3 Multi-Classification Neural Network

The network after the binary classification neural network in the nested network is the multi-classification neural network. The multi-classification neural network will be used to isolate the degraded component. The process used to determine the optimal network is similar to the process used to determine the optimal network for binary classification. Except random search is not used for hyperparameter optimisation process since it does not guarantee the best combination of hyperparameter, therefore the process will only involve

the random selection of initial network as the starting point and Bayesian optimisation to find the optimal solution. The initial network selected for the multi-classification neural network has an architecture of 10-15-6, which is similar to the same the initial network used for binary classification but the only exception is the modification made to the number of nodes in the output layer as there is 6 different engine component therefore it needs 6 output classes and the loss function used in the output layer is ”sparse categorical cross entropy”. Table 4.7 describes the size of the dataset used for training,testing and validating the multi-classification neural network as well as the corresponding target for each class.

Component	Total	Training	Testing	Validation	Target
Fan	105800	74060	15870	15870	0
IPC	105800	74060	15870	15870	1
HPC	105800	74060	15870	15870	2
HPT	105800	74060	15870	15870	3
IPT	105800	74060	15870	15870	4
LPT	105800	74060	15870	15870	5
Total	634800	444360	95220	95220	

Table 4.7: Data Distribution for Multi-Classification Neural Network

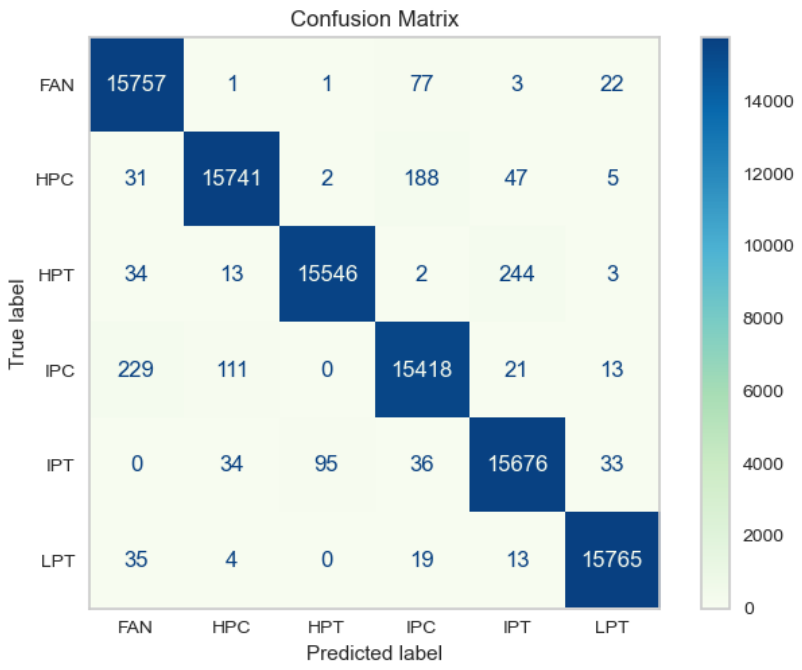


Figure 4.17: Confusion Matrix for Initial Multi-classification network

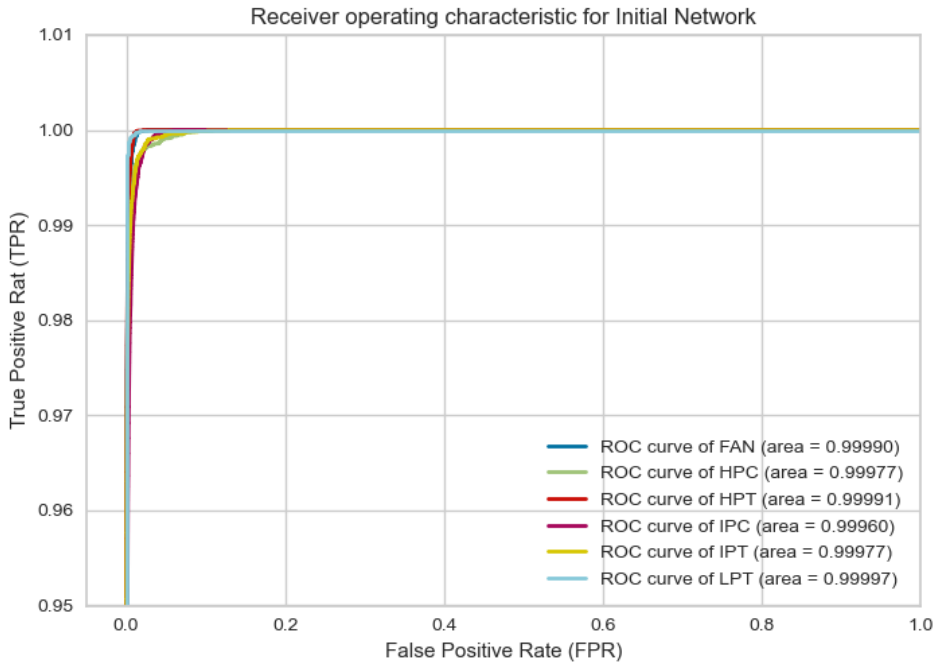


Figure 4.18: ROC Curve fir Initial Multi-classification network

The 10-15-6 network was able to achieve an average testing accuracy of 98.1% and loss value of 0.055. The accuracy calculated for each class using the confusion matrix in Figure 4.17 found to be closed to the average accuracy therefore the average accuracy of the network is not bias towards a class. The class which the network's prediction accuracy was the least is the IPC where the network was able to achieve an accuracy of 97.6%. This can also be seen in the Figure 4.18, where the AUC for the IPC ROC curve was the smallest therefore the network is less able to identify the pattern associated with IPC degradation.

Even though the performance of the 10-15-6 network is good, however it is not the optimal network since the network's hyperparameters are randomly chosen therefore they may still be room for improvement. The network found using Bayesian optimisation has an architecture of 10-21-30-30-30-6. This network was able to achieve an average testing accuracy of 99.3%. From Table 4.8, it also can be seen that the difference in individual class accuracy and the average network accuracy is much smaller, therefore it means the

network is able to predict every class accurately and is not bias towards a class.

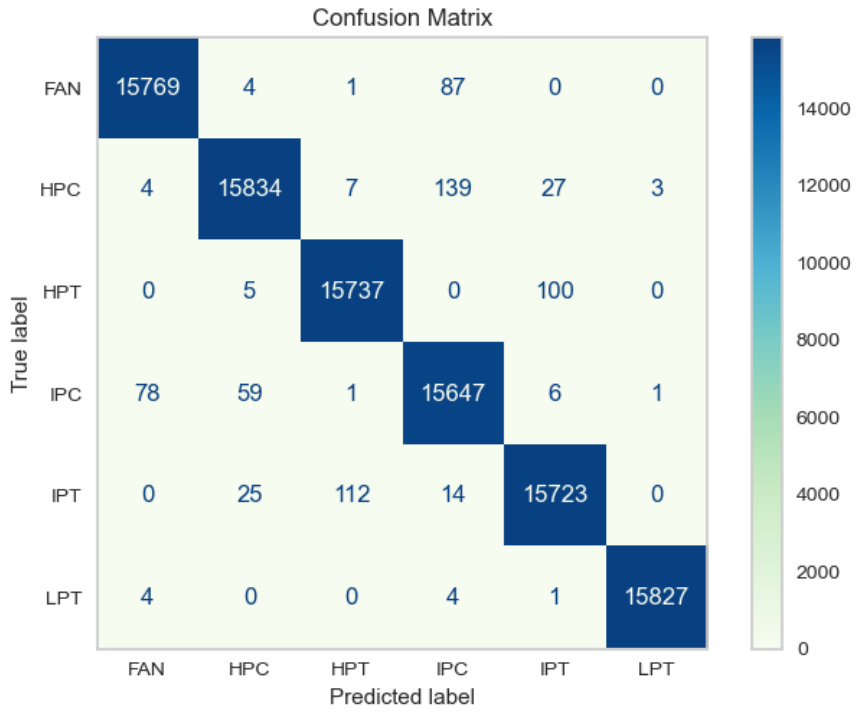


Figure 4.19: Confusion Matrix for Final Multi-classification network

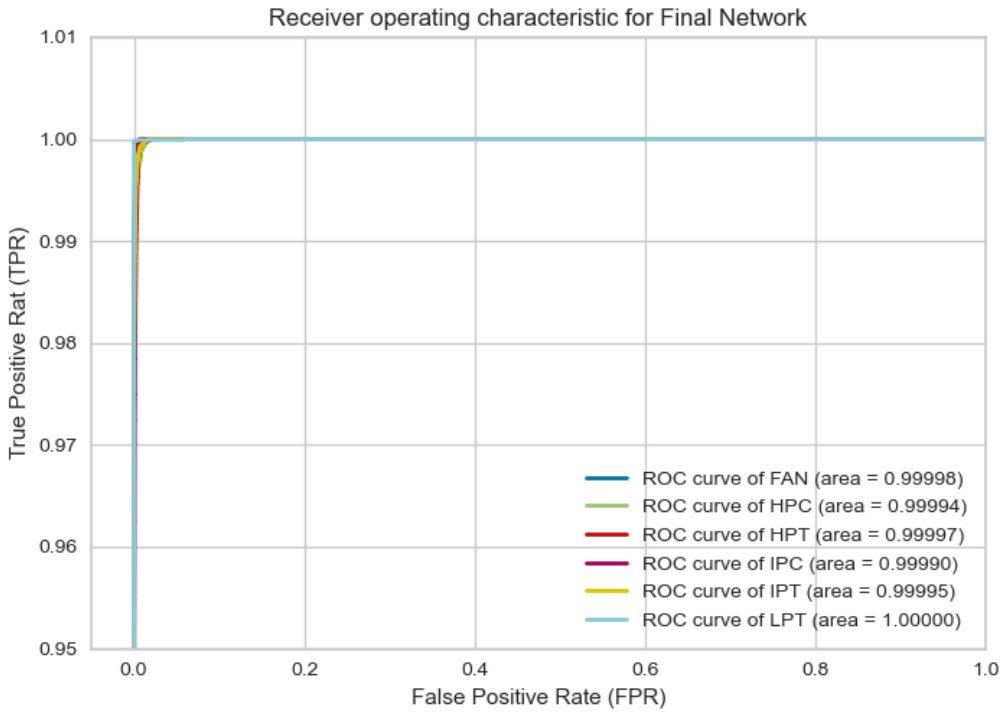


Figure 4.20: ROC Curve fir Final Multi-classification network

	Initial	Bayesian
Network Architecture	10-15-6	10-21-30-30-30-6
Activation Function	ReLU	TanH
Epochs	25	25
Average Accuracy	98.1%	99.3%
Class Accuracy		
Fan	99.3%	99.4%
IPC	97.6%	99.1%
HPC	98.3%	98.8%
HPT	98.1%	99.3%
IPT	98.7%	99.0%
LPT	99.5%	99.9%

Table 4.8: Multi-classification network comparison

4.6.4 Regression Neural Network

Once the degraded component has been isolated by the multi-classification neural network, a regression network is then used to estimate the degraded component's flow capacity and efficiency degradation level. As seen in Figure 4.12, the nested neural network consist of 6 different regression networks. Each network is assigned to a component and it will only be responsible for that component. The development of the 6 regression network are very similar to each other but it is different from the process used to develop the binary and multi-classification neural network. Instead of creating an initial model like how it was done for binary and multi-classification neural network, the 6 regression network was developed by using Bayesian optimisation method to determine the best combination of hyperparameters. Table 4.3 shows the size of the dataset and target value used to train, test and validate the regression neural networks. 70% of the dataset is used to train the neural network while the other 30% is split evenly for testing and validating

Components	Network Architecture	Activation Function	RMSE Value
Fan	10-40-28-10	tanH	0.452
IPC	10-17-15-10-2	Sigmoid	0.453
HPC	10-26-10-16-2	tanH	0.173
HPT	10-35-35-35-2	relu	0.217
IPT	10-27-26-23-2	relu	0.141
LPT	10-35-10-2	relu	0.028

Table 4.9: Regression network performance

From Table 4.9 it can be observed that RMSE value for the HPC, HPT, IPT and LPT regression network is low with the LPT regression network achieving the lowest RMSE value. This means that the regression network is able to estimate the component's flow capacity and efficiency degradation level close to the target value. This can also be observed in the regression plots in Appendix F, where 100 different data sample was plotted alongside the true target to evaluate the error between them, the blue markers in the HPC, HPT, IPT and LPT plots lies very close to the red line. The blue markers represents the degradation level estimated by the neural network while the red line indicates the true degradation level, therefore the closer the blue marker is to the red line, the smaller the difference between the estimated and true degradation level will be, therefore the better the network is at estimating the degradation level.

However, the same could not be said for the regression network used to estimate the fan and IPC degradation level. As seen in Figure F.2 in Appendix F, the blue markers are very far off the red line, therefore there is a significant difference between the estimate and true degradation level. The same can also be observed in the two IPC regression plots. This shows that the regression network for the IPC and Fan was not able to picked up on the change of trend in the engine measurements to accurately estimate the degradation level. One possible solution is to increase the number of samples used during the training process or alternatively, split the regression task into two separate task and use one neural network for each regression task so each network is able to focus on solving one task instead of two which is currently doing. However, by increasing the number of neural networks will also

increase the complexity of the nested neural network as well.

Case	Component	Target Γ	Estimated Γ	ΔΓ	Target η	Estimated η	$\Delta\eta$
Case 1	Fan	-2.00	-2.62	0.62	-1.3	-2.22	0.92
	IPC	-1.7	-2.77	1.07	-2.8	-3.70	0.9
	HPC	-2.20	-2.36	0.16	2.80	-2.88	0.08
	HPT	-4.00	-4.11	0.11	-3.50	-3.59	0.09
	IPT	-2.80	-2.80	0.00	-3.50	-3.33	-0.17
	LPT	2.60	2.62	0.02	-3.50	-3.44	-0.06
Case 2	Fan	-2.60	-3.05	0.45	-0.7	-1.05	0.35
	IPC	-2.60	-3.05	0.45	-0.7	-1.05	0.35
	HPC	-4.50	-4.52	0.02	-4.30	-4.39	0.09
	HPT	2.50	2.60	0.1	-0.5	-0.55	0.05
	IPT	0.80	0.88	0.08	-1.00	-0.99	0.01
	LPT	-1.80	-1.77	-0.03	1.30	-1.31	0.01

Table 4.10: Network Prediction vs Actual Target Comparison

4.7 Development Graphical User Interface for Diagnostic System

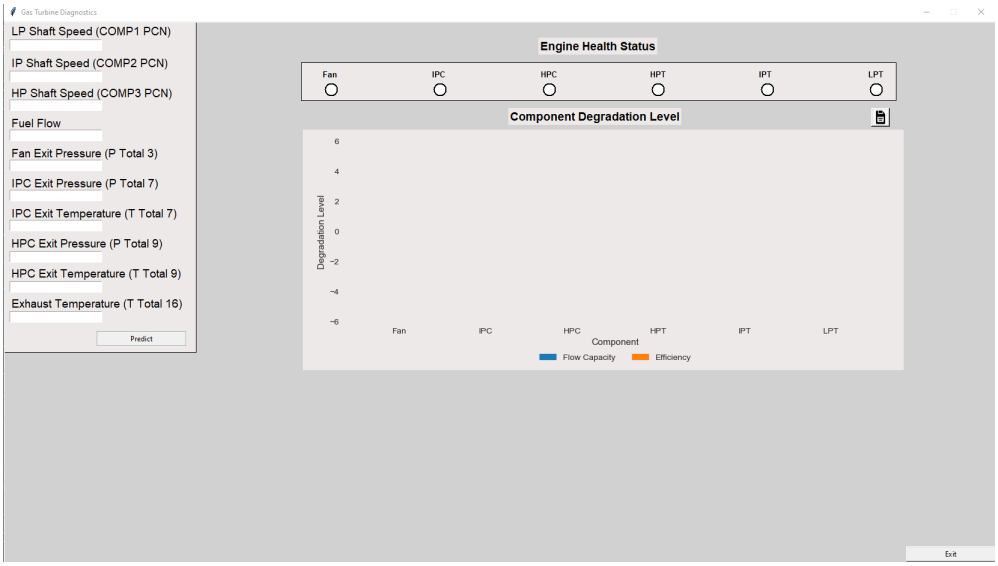


Figure 4.21: Diagnostics System Graphical User Interface (refer to Appendix G for bigger image)

Once all of the 8 of the individual networks which will be used to make up the nested neural network have been developed, a graphical user interface (GUI was built to combine

all the individuals network together and provide a easier way of using the nested network since it was developed it is in code form. Figure 4.21 shows the GUI of the diagnostics systems for the Trent-900 and it was developed using a GUI package in python called Tkinter. On the left side of the GUI, the user would input the PYTHIA generated Trent-900 engine measurement the user would like to diagnose. Once all the measurements has been entered into its respective box, the user just need to press the predict button to use the nested neural network to diagnose the status of the engine. If the nested neural network determines the engine to be clean, the engine health status indicator will all be green. If the nested neural network determines the engine to be degraded, the indicator of the degraded component will turn to red and the bar chart in the middle will show the estimated flow capacity and efficiency degradation level for the degraded component and the diagnostic process is complete.

4.8 Binary Classification Neural Network using Genetic Algorithm

A binary classification neural network was developed and trained using genetic algorithm. This is done to evaluate the feasibility of using genetic algorithm to optimise the weights of the connections by comparing its performance with the binary classification neural network obtained in subsection 4.6.2 which was trained using backpropagation algorithm. The flow chart for using genetic algorithm to optimise the weights of the connection can be seen in Figure 3.13. The python library used is the PyGAD library which is an open source machine learning and genetic algorithm library.

The architecture of the network was chosen to be the same as the binary classification network architecture obtained using Bayesian optimisation in subsection 4.6.2. After defining the architecture of the network, a fitness function was constructed using the loss function. Equation 4.1 is the fitness function which will be used to evaluate the performance of the network and it was constructed using binary cross entropy as the loss function since

it is a binary classification problem. A small value is added to the binary cross entropy to prevent a zero denominator from happening. The code for the fitness function can be seen in Appendix H. Once the fitness function and the layers of the network have been constructed, the data is normalised before it is fed into the neural network, however due to computational limitation only 225000 sample was used to train the the neural network and another 75000 was used to test the neural network. The parent selection method used was steady state selection and the type of crossover is single point crossover. The probability for crossover was set to 0.85 and the mutation probability was 0.01.

$$\text{Fitness Function} = \frac{1}{-\frac{1}{n}\sum_{i=1}^{n}(y_i\ log(p) + (1-y_i)\ log(1-p)) + 0.000001} \tag{4.1}$$

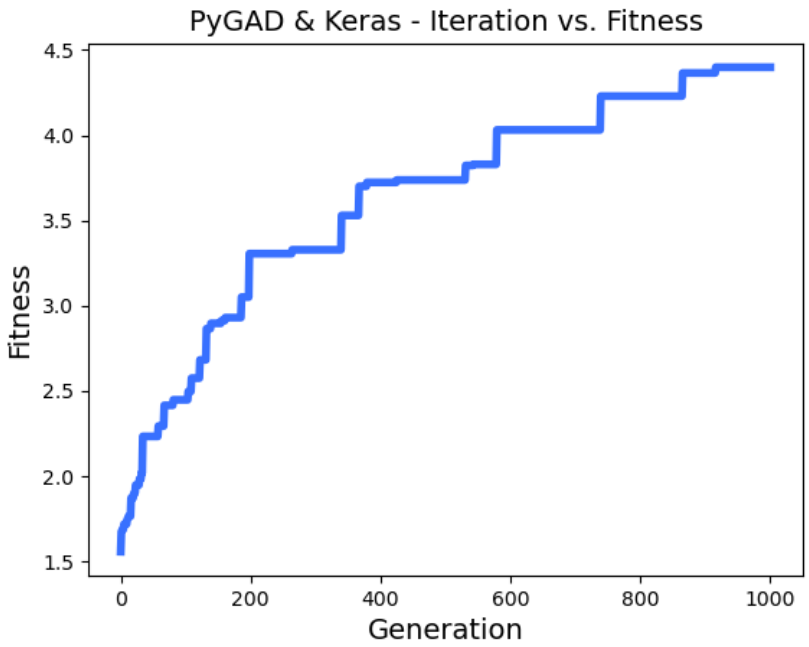


Figure 4.22: Fitness of each generation

Network Architecture	10-23-30-10-10-2
Activation Function	ReLU
Training Samples	225000
Testing Samples	75000
Parent Selection Method	Steady State Selection
Crossover type	Single Point
Crossover Probability	0.85
Mutation Probability	0.01

Table 4.11: Genetic Algorithm neural Network Parameters

After 1000 generations, the network was able to achieve a prediction accuracy of 91% on the testing dataset. This demonstrates the potential capability of using genetic algorithm to optimise the weights of the connection as an alternative to the commonly used backpropagation algorithm. However, when comparing the performance of this network to the network which was trained using backpropagation algorithm in subsection 4.6.2, this network was 8.4% less accurate than the network which uses backpropagation algorithm. This could be due to several factors such as the number of generations. The method can be very time consuming hence why the number of generations was set to 1000. With 1000 generations, it took roughly around 8 hours to complete which is significantly longer than backpropagation algorithm. It can be observed from Figure 4.22 that even at 1000 generations, there is still a possibility better solution can be found by increasing the number of generations. By increasing the number of generations would also mean increasing the time needed for the generation process complete. Additionally, there are still other variables such as the crossover rate, mutation probability, different parent selection methods and crossover type which needs investigated further to determine the best combination of those variables as only one combination of those variables were tested in this case. Furthermore, due to computing power limitations, only 34% of the total samples that was used during the backpropagation training was used to train the genetic algorithm neural network, therefore there is still area in which improvements can be made in an attempt to achieve a more accurate network.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

This thesis was able to achieve the aims set out in section 1.2, which was to developed a performance based diagnostic system using artificial neural network and develop a not so commonly used artificial neural network and evaluate its performance and capability for gas turbine diagnostics. The aims were broken down into multiple objectives which needed to be met in order to for the aim to be fulfilled.

The first objective is to carry out a comprehensive literature survey in order to understand the different type of component degradation a gas turbine may experience throughout it lifetime and the effect it has on the its performance. Then various maintenance strategies which has been in the past was explored before focusing on predictive maintenance strategy which is the go to maintenance strategy for modern gas turbine. Then a review on some of the performance based diagnostic system which has been developed and used in the past was carried out to identify the strength and weaknesses of the different diagnostic system before moving onto the methodology chapter.

As this research is based on using artificial neural network, a detailed review on the concept of artificial neural network and its theory was first carried out in order to build a

foundational understand for artificial neural network. The concept of nested neural network was then explored next as this will be the network used in the diagnostic system before exploring the different activation functions, feature scaling technique, training method and loss function associated to artificial neural network. After reviewing the basic concepts, the different hyperparameter optimisation methods was also analysed and to conclude the methodology section, a comprehensive study was done to understand the concept of neuroevolution and the different parent selection method and crossover type associated with it.

In order to develop the diagnostic system, the engine which the research will be based on must first be chosen in order to develop the engine model needed to generate the training, testing and validating dataset. For this thesis, the Rolls Royce Trent-900 was chosen. After choosing the engine, a model was developed using PYTHIA, a in-house engine modelling software at Cranfield University. As seen in Table 4.2, the performance of the engine model developed using PYTHIA was able to match the performance obtained from published sources. The engine model is then used to generate samples which will be used to train, test and validate the neural network. Table 4.3 contains all the information regarding the samples generated using the engine model. After sufficient amount of data has been generated, the development of the nested neural network can begin.

The first network developed was the binary classification neural network. This network will be used to detect if the engine is clean or degraded. As seen in Table 4.6, the initial network which has an randomly chosen network architecture of 10-15-2 was able to achieve an average testing accuracy of 98.4%. However, there is a 2.1% difference in prediction accuracy between the two classes. Rather than manually tuning the hyperparameters of the network to reduce the accuracy difference between the two classes, a random search method was used. From the random search, a network architecture of 10-18-26-20-2 was discovered and it was able to achieve a higher average testing accuracy of 99.1% while at the same time reducing the accuracy difference between the two classes to 0.5%.

However, since random search just randomly chooses the point in the search space, it does not guarantee the optimal solution, hence why Bayesian optimisation was used. A network architecture of 10-23-30-10-10-2 was discovered by Bayesian optimisation to be the optimal solution. This network was able to achieve an average testing accuracy of 99.4% and accuracy for both the classes were also above 99%. Table 4.5 contains all the information regarding the size of the dataset used to train, test and validate the binary classification neural network.

After obtaining the optimal network for binary classification, the next network which was developed was the multi-classification neural network. This network will be used to isolate the degraded component and information on the dataset used train, test and validate the multi-classification neural network can be obtained from Table 4.7. Since random search does not guarantee the optimal solution, only Bayesian optimisation was used to compare against the initial network. As seen in Table 4.8, the network obtained using Bayesian optimisation were able to predict all the classes more evenly as the biggest difference between the class and average accuracy is 0.6% while for the initial network the difference is 1.4%. Therefore in the initial network the network is more bias towards one class than the others.

To estimate the degradation level for each component, 6 different regression neural networks was developed. As seen in Table 4.9, Table 4.10 and the regression plots in Appendix F, the regression network for the fan and IPC was unable to accurately estimate the flow capacity and efficiency degradation level, while the regression network for the HPC, HPT, IPT and LPT were able to estimate the degradation level very close to the actual target. A potential solution to reduce the estimation error for the fan and IPC is to either increase the number of data or split the regression task into two separate task and have two regression neural network, one for estimating the flow capacity and the other for estimating the efficiency. This would help simplified the task for the neural network, however it would mean increase complexity in the nested neural network. Once all 8 neural networks have

been developed, a graphical user interface for the diagnostic system was developed to bring all the networks together and have a user friendly interface for ease of usage. The graphical user interface tor the diagnostic system can be seen in Appendix G.

Lastly, a study on the capability and feasibility of using genetic algorithm to optimise the weights of the connection was done on a binary classification neural network. The architecture of the binary classification network used in this study is the architecture obtained using Bayesian optimisation. After optimising the weights of the neural network using genetic algorithm, the network were able to achieve a testing accuracy of 91% which is far from 99.4% accuracy achieved using backpropagation. This could be due to several factors:

- **The use of smaller dataset -** Due to computational limitations only 300,000 samples were used to train the network.
- **Number of generations set -** Due to time limitations only 1000 generations were generated and the process took around 8 hours to complete. However from Figure 4.22, it can be seen that there is still a possibility for fitter solutions.
- **Genetic algorithm variables -** Only one combination of parent selection method, mutation probability, crossover probability and type was used in this study. A more detailed study of these variables would be needed to determine the optimal combination which would produce fitter solutions.

In conclusion, this thesis were able to fulfilled the aims and objectives of this research by successfully developed a artificial neural network based diagnostics system which is capable of diagnosing a gas turbine. Additionally, the use of genetic algorithm to optimise the weights of the connection was also explored as an alternative to the widely used backpropagation algorithm.

5.2 Future Work

The following work is proposed by the author which can be carried out in the future to improve on what has been done in this thesis, this includes:

- **Further investigate the use of genetic algorithm for optimising the weights of the neural network connection -** As mentioned in section 4.8, only one type of crossover and parents selection method was covered in this thesis, therefore a potential future work is to study the various selection method and crossover type covered in subsection 3.9.1 to determine which selection method and crossover type would produce a better performing network.
- **Study the effect of mutation probability and crossover rate -** The chosen mutation probability and crossover rate chosen in this thesis may not be optimal, therefore a potential future work is to study the effect of them and determine the optimal mutation probability and crossover rate.
- **Add dual component faults and sensors fault into the diagnostic system -** To have a more complete diagnostic system, dual component fault and sensor fault should also be included in the diagnostic system.
- **Development of a prognosis system -** A prognosis system maybe developed to determine the remaining useful life of the engine.

References

[1] Ajith Abraham. *129: Artificial Neural Network*. Ed. by Peter H Sydenham and Richard Thorn. 2005. URL: http://wsc10.softcomputing.net/ann_chapter.pdf.

[2] Airbus. *Airbus A380 Facts and Figures*. Dec. 2021. URL: https://www.airbus.com/sites/g/files/jlcbta136/files/2021-12/EN-Airbus-A380-Facts-and-Figures-December-2021_0.pdf.

[3] Airbus. *ANA Group selects the A380*. Jan. 2016. URL: https://www.airbus.com/en/newsroom/press-releases/2016-01-ana-group-selects-the-a380.

[4] Mohamed Alloghanio et al. *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*. Ed. by Michael W. Berry, Azlinah Mohamed, and Bee Wah Yap. 2020. DOI: 10.1007/978-3-030-22475-2.

[5] Jasem Alqallaf et al. "Solid Particle Erosion Behaviour and Protective Coatings for Gas Turbine Compressor Blades—A Review". In: *Processes* 8 (8 Aug. 2020), p. 984. ISSN: 2227-9717. DOI: 10.3390/pr8080984.

[6] Danish Anwar. *How doe a turbofan engine work?* URL: https://www.behance.net/gallery/27888995/How-does-a-turbofan-engine-work.

[7] Shubham Baghel. *Understanding AUC-ROC*. June 2020. URL: https://medium.datadriveninvestor.com/understanding-auc-roc-clearly-explained-74c53d292a02.

[8] Gunston Bill. *Jane's Aero engines*. 2014.

[9] Miller. Brad and Goldberg. David. “Genetic Algorithms, Tournament Selection and the Effects of Noise”. In: *Complex Systems* 9 (3 1995), pp. 193–212. URL: https://wpmedia.wolfram.com/uploads/sites/13/2018/02/09-3-2.pdf.

[10] DANB. *Rectified Linear Units (ReLU) in Deep Learning*. 2018. URL: https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook.

[11] Leonid Datta. “A Survey on Activation Functions and their relation with Xavier and He Normal Initialization”. In: (Mar. 2020). URL: https://arxiv.org/pdf/2004.06632.pdf#:~:text=Zero%2Dcentered%3A%20A%20function%20is,always%20positive%20or%20always%20negative..

[12] Ian Dewancker, Michael McCourt, and Scott Clark. “Bayesian Optimization Primer”. In: *SIGOPT* (). URL: https://static.sigopt.com/b/20a144d208ef255d3b981ce419667ec25d8412e2/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf.

[13] Bin Ding, Huimin Qian, and Jun Zhou. “Activation functions and their characteristics in deep neural networks”. In: IEEE, June 2018, pp. 1836–1841. ISBN: 978-1-5386-1244-6. DOI: 10.1109/CCDC.2018.8407425.

[14] Shifei Ding et al. “Evolutionary artificial neural networks: a review”. In: *Artificial Intelligence Review* 39 (3 Mar. 2013), pp. 251–260. ISSN: 0269-2821. DOI: 10.1007/s10462-011-9270-6.

[15] Thara D.K., PremaSudha B.G, and Fan Xiong. “Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques”. In: *Pattern Recognition Letters* 128 (Dec. 2019), pp. 544–550. ISSN: 01678655. DOI: 10.1016/j.patrec.2019.10.029.

[16] EASA. *TYPE-CERTIFICATE DATA SHEET for RB211 Trent 900 series engines*. European Union Aviation Safety Agency, 2019. URL: https://www.easa.europa.eu/sites/default/files/dfu/TCDS%20E.012_Issue%2010.pdf.

[17] IBM Cloud Education. *Machine Learning*. July 2020. URL: https://www.ibm.com/cloud/learn/machine-learning.

[18] IBM Cloud Education. *Neural Networks*. Aug. 2020. URL: https://www.ibm.com/uk-en/cloud/learn/neural-networks.

[19] Jeremiah Eyoh and Roy S Kalawsky. "Evolution of maintenance strategies in oil and gas industries: the present achievements and future trends". In: Forum for Engineering, Applied Sciences & Technology, July 2018.

[20] Jianli Feng and Shengnan Lu. "Performance Analysis of Various Activation Functions in Artificial Neural Networks". In: *Journal of Physics: Conference Series* 1237 (2 June 2019), p. 022030. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1237/2/022030.

[21] Fentaye et al. "A Review on Gas Turbine Gas-Path Diagnostics: State-of-the-Art Methods, Challenges and Opportunities". In: *Aerospace* 6 (7 July 2019), p. 83. ISSN: 2226-4310. DOI: 10.3390/aerospace6070083. URL: https://www.mdpi.com/2226-4310/6/7/83.

[22] Edgar Galván and Peter Mooney. "Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges". In: *Evolutionary Programming* 39 (June 2020). DOI: 10.1109/TAI.2021.3067574. URL: http://arxiv.org/abs/2006.05415http://dx.doi.org/10.1109/TAI.2021.3067574.

[23] Nigel Goddard. *Mean Squared Error and Outliers*. Sept. 2016. URL: https://media.ed.ac.uk/media/Mean+Squared+Error+and+Outliers/1_af8giwoz.

[24] Ahmad Hassanat et al. "Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach". In: *Information* 10 (12 Dec. 2019), p. 390. ISSN: 2078-2489. DOI: 10.3390/info10120390.

[25] *How Our Brain Learns*. June 2015. URL: http://www.brains-explained.com/how-our-brains-learn/.

[26] Rainer Kurz, Cyrus Meher-Homji, and Klaus Brun. “Gas Turbine Degradation”. In: Texas A & M Engineering Experiment Station, 2014. URL: https://core.ac.uk/download/pdf/87264032.pdf.

[27] Chunhui Li and Chuanhua Yu. “Performance Evaluation of Public Non-Profit Hospitals Using a BP Artificial Neural Network: The Case of Hubei Province in China”. In: *International Journal of Environmental Research and Public Health* 10 (8 Aug. 2013), pp. 3619–3633. ISSN: 1660-4601. DOI: 10.3390/ijerph10083619. URL: https://www.mdpi.com/1660-4601/10/8/3619.

[28] Y. G. Li. “Performance-analysis-based gas turbine diagnostics: A review”. In: *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 216 (5 Jan. 2002), pp. 363–377. ISSN: 0957-6509. DOI: 10.1243/095765002320877856. URL: https://journals.sagepub.com/doi/pdf/10.1243/095765002320877856.

[29] Y.G Li and R. Singh. “AN ADVANCED GAS TURBINE GAS PATH DIAGNOSTIC SYSTEM - PYTHIA”. In: *International Society of Air Breathing Engines* (2005).

[30] Yiguang Li. *PYTHIA 5.1 USER’S GUIDE*. Cranfield University.

[31] Hause Lin. *Gentle intro to logistic regression*. Apr. 2019. URL: https://hausetutorials.netlify.app/posts/2019-04-13-logistic-regression/.

[32] Sweety Mahanta, M. Chandrasekaran, and Sutanu Samanta. “EDM of Al7075-B 4 C-flyash hybrid metal matrix nano-composites and optimization of sustainable measures using genetic algorithm”. In: Dec. 2017.

[33] Cyrus B Meher-Homji. “Gas Turbine Axial Compressor Fouling and Washing”. In: Texas A & M University, 2004. URL: https://oaktrust.library.tamu.edu/bitstream/handle/1969.1/163249/t33-18.pdf?sequence=1&isAllowed=y.

[34] Razali. Noraini Mohd and Geraghty. John. “Genetic Algorithm Performance with Different Selection Strategies in Solving TSP”. In: International Association of

Engineers, 2011. ISBN: 978-988-19251-4-5. URL: http://www.iaeng.org/publication/WCE2011/WCE2011_pp1134-1139.pdf.

[35] Laurence Moroney and Gordon Josh. *Bringing Machine Learning to every developer's toolbox*. June 2022. URL: https://blog.tensorflow.org/2022/06/%20bringing-machine-learning-to-every-developers-toolbox.html.

[36] Vidya Muthukumar et al. "Classification vs regression in overparameterised regmise: Does the loss function matter?" In: *Hournal of Machine Learning Research* 22 (July 2021). URL: https://dl.acm.org/doi/pdf/10.5555/3546258.3546480.

[37] McCullum Nick. *Deep Learning Activation Functions*. URL: https://nickmccullum.com/python-deep-learning/deep-learning-activation-functions/.

[38] Stephen Ogaji and Riti Singh. *Artificial Neural Networks in Fault Diagnosis: A Gas Turbine Scenario*. Ed. by Professor Lakhmi Jain and Professor Xindong Wu. 2006. URL: https://link.springer.com/content/pdf/10.1007/978-1-84628-631-5.pdf.

[39] Abdoun. Otman, Abouchabaka. Jaafar, and Tajani. Chakir. *Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem*. LaRIT Laboratory. URL: https://arxiv.org/ftp/arxiv/papers/1203/1203.3099.pdf.

[40] Dário Passos and Puneet Mishra. "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks". In: *Chemometrics and Intelligent Laboratory Systems* 223 (Apr. 2022), p. 104520. ISSN: 01697439. DOI: 10.1016/j.chemolab.2022.104520. URL: https://www.sciencedirect.com/science/article/pii/S0169743922000314.

[41] Sathya R. and Abraham Annamma. "Comparison of Supervised and Unsupervised learning algorithms for pattern classification". In: *International Journal of Advanced Research In Artificial intelligence* 2 (2 2013). ISSN: 2165-4069. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.5274&rep=rep1&type=pdf#page=41.

[42] Anna Rakitianskaia and Andries Engelbrecht. “Measuring Saturation in Neural Networks”. In: IEEE, Dec. 2015, pp. 1423–1430. ISBN: 978-1-4799-7560-0. DOI: 10.1109/SSCI.2015.202. URL: http://vigir.missouri.edu/~gdesouza/Research/Conference_CDs/IEEE_SSCI_2015/data/7560b423.pdf.

[43] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. “A Review of Activation Function for Artificial Neural Network”. In: *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2020, pp. 281–286. DOI: 10.1109/SAMI48414.2020.9108717.

[44] University of Regina. *Neuro-Evolution*. URL: http://www2.cs.uregina.ca/~dbd/cs831/notes/neural-networks/neuroevolution/.

[45] Durbin Richard and Golden Richard. *Backpropagation: The Basic Theory*. Ed. by Chauvin Yves and Rumelhart David E. 1995.

[46] Sebastian Risi and Kenneth O. Stanley. “Enhancing es-hyperneat to evolve more complex regular neural networks”. In: ACM Press, July 2011, pp. 1539–1546. ISBN: 9781450305570. DOI: 10.1145/2001576.2001783. URL: https://dl.acm.org/doi/pdf/10.1145/2001576.2001783.

[47] B. Salehnasab et al. “Hot corrosion failure in the first stage nozzle of a gas turbine engine”. In: *Engineering Failure Analysis* 60 (Feb. 2016), pp. 316–325. ISSN: 13506307. DOI: 10.1016/j.engfailanal.2015.11.057. URL: https://www.sciencedirect.com/science/article/pii/S1350630715301783.

[48] Subana Shanmuganathan. *Artificial Neural Network Modelling*. Ed. by Sandhya Samarasinghe. Vol. 628. Springer International Publishing, 2016. ISBN: 978-3-319-28493-4. DOI: 10.1007/978-3-319-28495-8.

[49] Andreas Spaeth. *Airbus A380 superjumbo makes a comeback as passenger numbers soar*. July 2022. URL: https://www.dw.com/en/airbus-a380-superjumbo-makes-a-comeback-as-passenger-numbers-soar/a-62394101.

[50] Kenneth O. Stanley et al. “Designing neural networks through neuroevolution”. In: *Nature Machine Intelligence* 1 (1 Jan. 2019), pp. 24–35. ISSN: 2522-5839. DOI: 10.1038/s42256-018-0006-z.

[51] Blickle Tobias and Thiele Lothar. *A Comparison of Selection Schemes used in Genetic Algorithms*. Swiss Federal Institute of Technology, Dec. 1995. URL: https://tik-old.ee.ethz.ch/file/6c0e384dceb283cd4301339a895b72b8/TIK-Report11.pdf.

[52] Robert Wilke. *Unexpected Costs from a Foreign Object Damage Event*. 2015. URL: https://avrisk.net/unexpected-costs-from-a-foreign-object-damage-fod-event/.

[53] Mike Wolfe. *TensorFlow vs Keras: A Comparison*. July 2021. URL: https://towardsdatascience.com/tensorflow-vs-keras-d51f2d68fdfc.

[54] Thomas Wood. *Softmax Function*. URL: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[55] Koichi Yonezawa et al. “Influence of blade corrosion on aerodynamic characteristics of a gas turbine”. In: *Energy* 230 (Sept. 2021), p. 120665. ISSN: 03605442. DOI: 10.1016/j.energy.2021.120665. URL: https://www.sciencedirect.com/science/article/pii/S0360544221009142.

[56] Martha Arbayani Zaidan. “Bayesian Approaches for Complex System Prognostics”. University of Helsinki, Mar. 2014. DOI: DOI:10.13140/RG.2.2.21966.10560. URL: https://www.researchgate.net/publication/328769067_Bayesian_Approaches_for_Complex_System_Prognostics.

[57] Ningbo Zhao, Xueyou Wen, and Shuying Li. “A Review on Gas Turbine Anomaly Detection for Implementing Health Management”. In: American Society of Mechanical Engineers, June 2016. ISBN: 978-0-7918-4968-2. DOI: 10.1115/GT2016-58135.

Appendices

Appendix A

PYTHIA Design Point Text Output File

```
================= PYTHIA V5.0 ========================
FOR AERO/INDUSTRIAL/MARINE GAS TURBINE ENGINE
DESIGN POINT/OFF-DESIGN PERFORMANCE CALCULATIONS
=====================================================
!PYTHIA DESIGN FILE
!ENGINE TYPE: Trent 900 Version 1
////
DP KE FP
-1
-1
INTAKE S1,2 D1,2,3,4,5,6 R300
COMPRE S2,3,0,0 D7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26 R301 V7
PREMAS S3,401,402 D27,28,29,30,31,32,33 V27
DUCTER S401,5 D34,35,36,37,38,39 R0
NOZCON S5,6,0 D40,41,42,43,44,45 R303
```

COMPRE S402,7,701,0 D46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65 R304 V46

COMPRE S7,9,0,0 D73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92 R305 V73 V74

PREMAS S9,11,18 D100,101,102,103,104,105,106

BURNER S11,12 D107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122 R306

MIXEES S12,18,13

TURBIN S13,14 D123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138 V124

TURBIN S14,15 D139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154 V140

TURBIN S15,16 D155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170 V156

NOZCON S16,17,0 D171,172,173,174,175,176 R307

PERFOR S1,0,0 D177,178,179,180,303,300,306,307,0,0,0,0,0

CODEND

DATA////

1 10670.0

2 15.0

3 0.85

4 -1.0

5 0.0

6 72.0

7 -1.0

8 -1.0

9 1.0

10 1.611

11 0.929

12 0.0

13 4.0

14 1.0

15 1.0

16 1.0

17 1.0

18 0.0

19 0.0

20 0.0

21 0.0

22 0.0

23 -1.0

24 1.0

25 -1.0

26 -1.0

27 0.896740740740741

28 0.0

29 1.0

30 0.0

31 -1.0

32 -1.0

33 -1.0

34 0.0

35 0.01

36 0.0

37 0.0

38 0.0

39 0.0

40 -1.0

41 1.0

42 -1.0

43 -1.0

44 -1.0

45 1.0

46 -1.0

47 -1.0

48 1.0

49 4.044

50 0.888

51 0.0

52 10.0

53 2.0

54 1.0

55 1.0

56 1.0

57 1.0

58 0.0195

59 0.0

60 0.0

61 0.0

62 -1.0

63 1.0

64 -1.0

65 -1.0

73 -1.0

74 -1.0

75 1.0

76 6.053

77 0.889

78 0.0

79 10.0

80 3.0

81 1.0

82 1.0

83 1.0

84 0.0

85 0.0

86 0.0

87 0.0

88 0.0

89 -1.0

90 1.0

91 -1.0

92 -1.0

100 0.85

101 0.0

102 1.0

103 0.0

104 -1.0

105 -1.0

106 -1.0

107 0.05

108 0.99

109 -1.0

110 0.0

111 0.0

112 0.0

113 1.0

114 -1.0

115 0.0

116 -1.0

117 0.0

118 -1.0

119 -1.0

120 -1.0

121 -1.0

122 -1.0

123 1000.0

124 -1.0

125 -1.0

126 0.938

127 -1.0

128 3.0

129 7.0

130 -1.0

131 1.0

132 1.0

133 1.0

134 0.0

135 1.0

136 0.99

137 -1.0

138 1.0

139 1000.0

140 -1.0

141 -1.0

142 0.93

143 0.0

144 2.0

145 7.0

146 -1.0

147 1.0

148 1.0

149 1.0

150 0.0

151 1.0

152 0.99

153 -1.0

154 1.0

155 0.0

156 -1.0

157 -1.0

158 0.986

159 0.0

160 1.0

161 5.0

162 -1.0

163 1.0

164 1.0

165 1.0

166 0.0

167 1.0

168 0.99

169 -1.0

170 1.0

171 -1.0

172 1.0

173 -1.0

174 -1.0

175 -1.0

176 1.0

178 1.0

179 0.0

180 0.0

-1

1 2 535.591

12 6 1537.327

-1

==========

THE PARAMETER UNITS:

Temperature = K Pressure = Atmospheres Length = m

Area = sq m Mass Flow = kg/s Velocity = m/sec

Thurst = N Sp. Thrust = N/kg.sec

Power = W Sp. Power = W/kg.sec

s.f.c.(Thrust) = kg/N.s s.f.c.(Power) = kg/W.s

==========

***** AMBIENT AND INTAKE 1 PARAMETERS *****

Alt. = 10670.0000 I.S.A. T Dev. = 15.0000

ISA P Dev. = 0.0000 Mach No. = 0.8500 Etar = 1.0000

Rel.Humidity = 72.0000 Momentum Drag = 139524.9555

***** COMPRESSOR 1 PARAMETERS *****

PRSF = 0.9410 ETASF = 1.1143 WASF = 4.3334

DGPRSF = 1.0000 DGWACSF = 1.0000 DGETASF = 1.0000

Z = 0.8000 PR = 1.6150 ETA = 0.9300

PCN = 1.0000 CN = 1.0377 COMWK = 22727623.1832

SM = 0.2000 WA = 535.3370

***** PREMAS 1 PARAMETERSS *****

Lambda (W) = 0.8967 BPR = 0.1151 1/BPR = 8.6844

***** DUCTER 1 PARAMETERSS *****

Delta P = 0.0100 ETA = 0.0000 Fuel = 0.0000

Afterburner Temperature = 0.0

***** CONVERGENT NOZZLE 1 PARAMETERS *****

PRSF = 0.5702 ETASF = 1.1510 WASF = 7.2037

DGPRSF = 1.0000 DGWACSF = 1.0000 DGETASF = 1.0000

Z = 0.8000 PR = 1.6110 ETA = 0.9290

PCN = 1.0000 CN = 1.0377 COMWK = 22636935.3856

SM = 0.2000 WA = 535.5910

***** PREMAS 1 PARAMETERSS *****

Lambda (W) = 0.8967 BPR = 0.1151 1/BPR = 8.6844

***** DUCTER 1 PARAMETERSS *****

Delta P = 0.0100 ETA = 0.0000 Fuel = 0.0000

Afterburner Temperature = 0.0

***** CONVERGENT NOZZLE 1 PARAMETERS *****

NCOSF = 1.0000 DGNCOSF = 1.0000

Area = 3.42938 Exit Velocity = 322.1108 Gross Thrust = 180018.1057

Nozzle Coeff. = 0.9780 EXIT MACH No = 1.0000 PR Coef = 1.0000

***** COMPRESSOR 2 PARAMETERS *****

PRSF = 0.4929 ETASF = 1.0317 WASF = 0.9834

DGPRSF = 1.0000 DGWACSF = 1.0000 DGETASF = 1.0000

Z = 0.8000 PR = 4.0440 ETA = 0.8880

PCN = 1.0000 CN = 0.9646 COMWK = 9500486.5827

SM = 0.2000 WA = 55.3047

***** COMPRESSOR 3 PARAMETERS *****

PRSF = 1.2642 ETASF = 1.0568 WASF = 0.3686

DGPRSF = 1.0000 DGWACSF = 1.0000 DGETASF = 1.0000

Z = 0.8000 PR = 6.0530 ETA = 0.8890

PCN = 1.0000 CN = 0.7756 COMWK = 19541445.6681

SM = 0.2000 WA = 54.2263

***** PREMAS 2 PARAMETERSS *****

Lambda (W) = 0.8500 BPR = 0.1765 1/BPR = 5.6667

***** COMBUSTOR 1 PARAMETERSS *****

ETASF = 0.9900 DGETASF = 1.0000

ETA = 0.9900 Pressure Drop = 0.7439 Fuel Flow = 0.9590

Water Flow = 0.0000

ETA = 0.9900 Pressure Drop = 0.7439 Fuel Flow = 0.9590

Water Flow = 0.0000

***** TURBINE 1 PARAMETERS *****

CNSF = 0.0441 ETASF = 1.2375 TFSF = 0.6761 DHSF

= 56774.0640

DGETASF = 1.0000 DGTFSF = 1.0000 DGDHSF = 1.0000

LoadCoef = 1.0000

SHAFTEFF = 0.9900 SHAFTLOSS = 195414.4567 Referred

AUXWK = 1658.3189

TF = 148.0991 DH = 248.5562 ETA = 0.9380 CN

= 0.6001 AUXWK = 1000.0000

***** TURBINE 2 PARAMETERS *****

CNSF = 0.0494 ETASF = 1.2270 TFSF = 1.7061 DHSF

= 34576.2825

DGETASF = 1.0000 DGTFSF = 1.0000 DGDHSF = 1.0000

LoadCoef = 1.0000

SHAFTEFF = 0.9900 SHAFTLOSS = 95004.8658 Referred AUXWK = 1658.3189

TF = 373.6945 DH = 151.3746 ETA = 0.9300 CN = 0.6001 AUXWK = 1000.0000

***** TURBINE 3 PARAMETERS *****

CNSF = 0.0176 ETASF = 1.2715 TFSF = 3.7636 DHSF = 132122.0897

DGETASF = 1.0000 DGTFSF = 1.0000 DGDHSF = 1.0000 LoadCoef = 1.0000

SHAFTEFF = 0.9900 SHAFTLOSS = 226369.3539 Referred AUXWK = 0.0000

TF = 642.2172 DH = 413.0709 ETA = 0.9860 CN = 1.8000 AUXWK = 0.0000

***** CONVERGENT NOZZLE 2 PARAMETERS *****

NCOSF = 1.0000 DGNCOSF = 1.0000

Area = 0.77504 Exit Velocity = 460.1862 Gross Thrust = 24958.7658

Nozzle Coeff. = 0.9806 EXIT MACH No = 1.0000 PR Coef = 1.0000

Station	F.A.R.	Mass Flow	Pstatic	Ptotal	Tstatic	Ttotal	Vel	Area	W.A.R.
1	0.0000	535.59	0.2352	0.3773	233.8	267.6	260.6	xxxxxx	0.0004
2	0.0000	535.59	xxxxxx	0.3773	xxxxxx	267.6	xxxxxx	xxxxxx	0.0004
3	0.0000	535.59	xxxxxx	0.6078	xxxxxx	309.7	xxxxxx	xxxxxx	0.0004
401	0.0000	480.29	xxxxxx	0.6078	xxxxxx	309.7	xxxxxx	xxxxxx	0.0004
402	0.0000	55.30	xxxxxx	0.6078	xxxxxx	309.7	xxxxxx	xxxxxx	0.0004
5	0.0000	480.29	xxxxxx	0.6018	xxxxxx	309.7	xxxxxx	xxxxxx	0.0004
6	0.0000	480.29	0.3179	0.6018	258.1	309.7	322.1	3.4294	0.0004
7	0.0000	54.23	xxxxxx	2.4581	xxxxxx	479.1	xxxxxx	xxxxxx	0.0004
701	0.0000	1.08	xxxxxx	2.4581	xxxxxx	479.1	xxxxxx	xxxxxx	0.0004
9	0.0000	54.23	xxxxxx	14.8790	xxxxxx	817.8	xxxxxx	xxxxxx	0.0004
11	0.0000	46.09	xxxxxx	14.8790	xxxxxx	817.8	xxxxxx	xxxxxx	0.0004
18	0.0000	8.13	xxxxxx	14.8790	xxxxxx	817.8	xxxxxx	xxxxxx	0.0004
12	0.0208	47.05	xxxxxx	14.1350	xxxxxx	1537.3	xxxxxx	xxxxxx	0.0004
13	0.0177	55.19	xxxxxx	14.1350	xxxxxx	1439.0	xxxxxx	xxxxxx	0.0004
14	0.0177	55.19	xxxxxx	5.0052	xxxxxx	1148.8	xxxxxx	xxxxxx	0.0004
15	0.0177	55.19	xxxxxx	2.7214	xxxxxx	1003.0	xxxxxx	xxxxxx	0.0004
16	0.0177	55.19	xxxxxx	0.4424	xxxxxx	636.9	xxxxxx	xxxxxx	0.0004
17	0.0177	55.19	0.2359	0.4424	538.2	636.9	460.2	0.7750	0.0004

Gross Thrust = 204976.87

Momentum Drug = 139591.16

Net Thrust = 65385.72

Fuel Flow = 0.9590

S.F.C. = 14.6668

Sp. Thrust = 122.08

PCN 1 = 1.0000

CN 1 = 1.0377

PCN 2 = 1.0000

CN 2 = 0.9646

PCN 3 = 1.0000

CN 3 = 0.7756

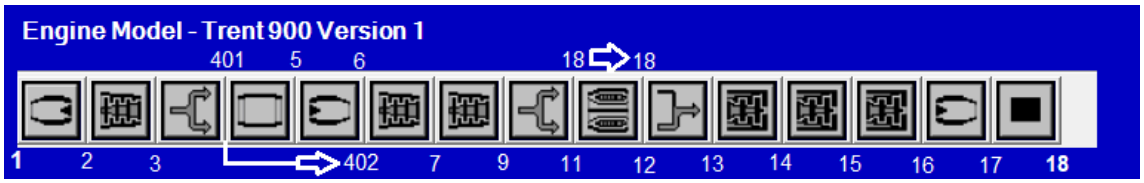
TET 1 = 1537.3

****

-3

Appendix B

Trent-900 Brick



Station	Brick Name	Component in engine
1 - 2	INTAKE	Engine Intake
2 - 3	COMPRE	Fan
3 - 401,402	PREMAS	Bypass Splitter
401 - 5	DUCTER	Bypass Duct
5 - 6	NOZCON	Cold Exhaust
402 - 7	COMPRE	IPC
7 - 9	COMPRE	HPC
9 - 11,18	PREMAS	Bled 15% air for cooling
11 - 12	BURNER	Combustor Chamber
12,18 - 13	MIXEES	Mix cooler air and combusted air
13 - 14	TURBIN	HPT
14 - 15	TURBIN	IPT
15 - 16	TURBIN	LPT
16 - 17	NOZCON	Hot Exhaust
17 - 18	PERFOR	Calculate engine’s performance

Appendix C

Affect of Compressor and Fan degradation on Engine's Performance

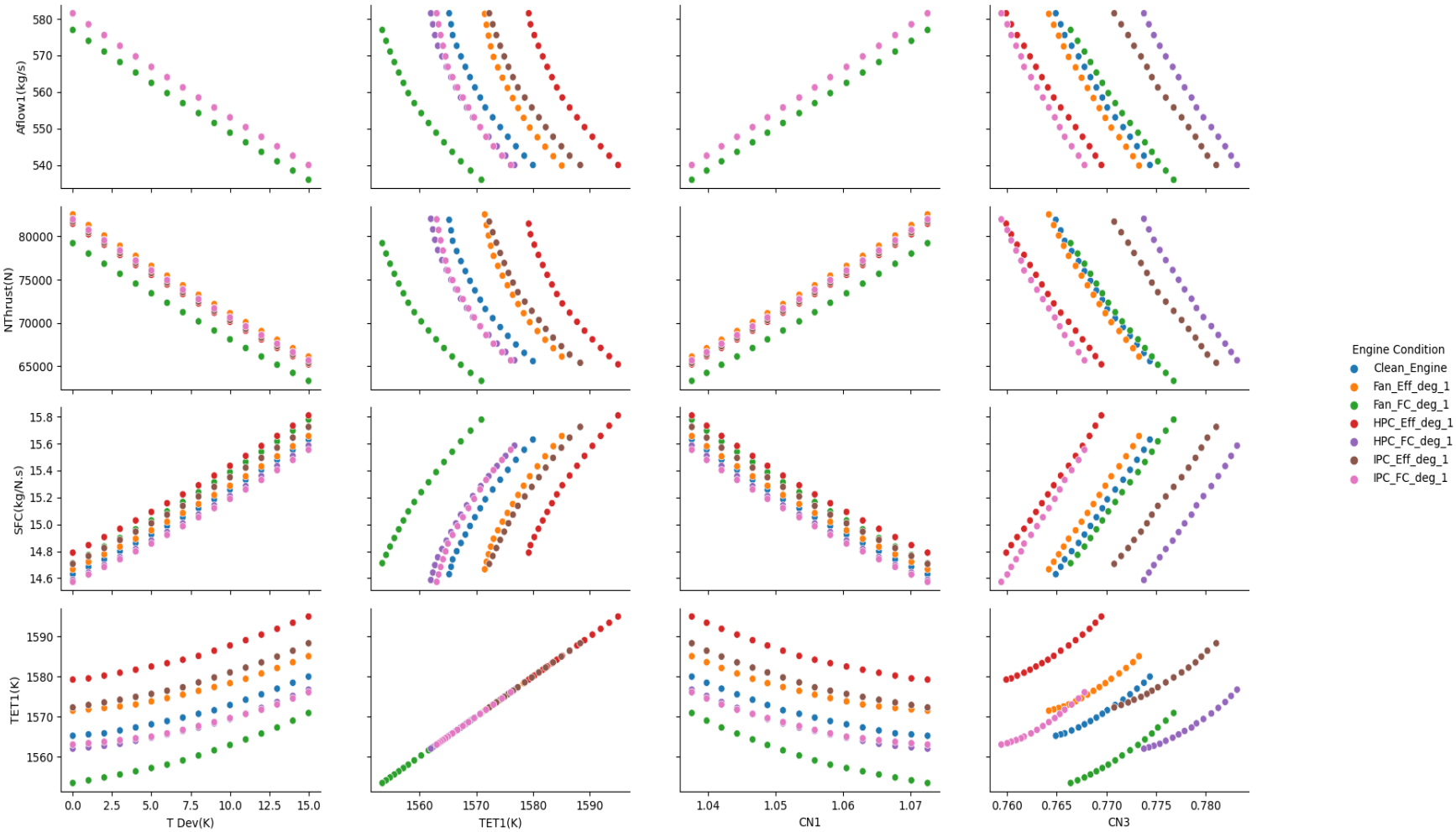


Figure C.1: Cold Section Pair-Plot

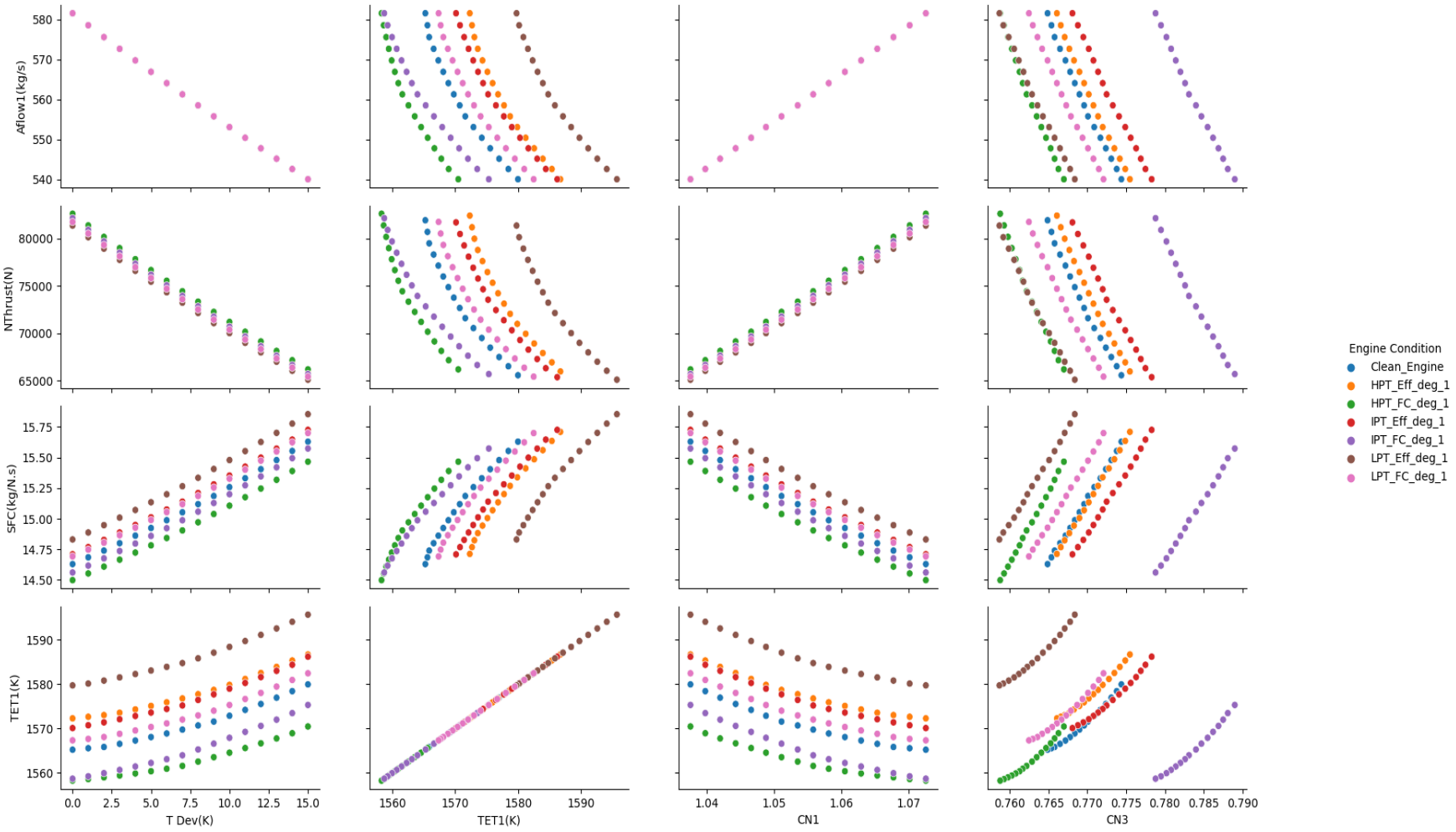
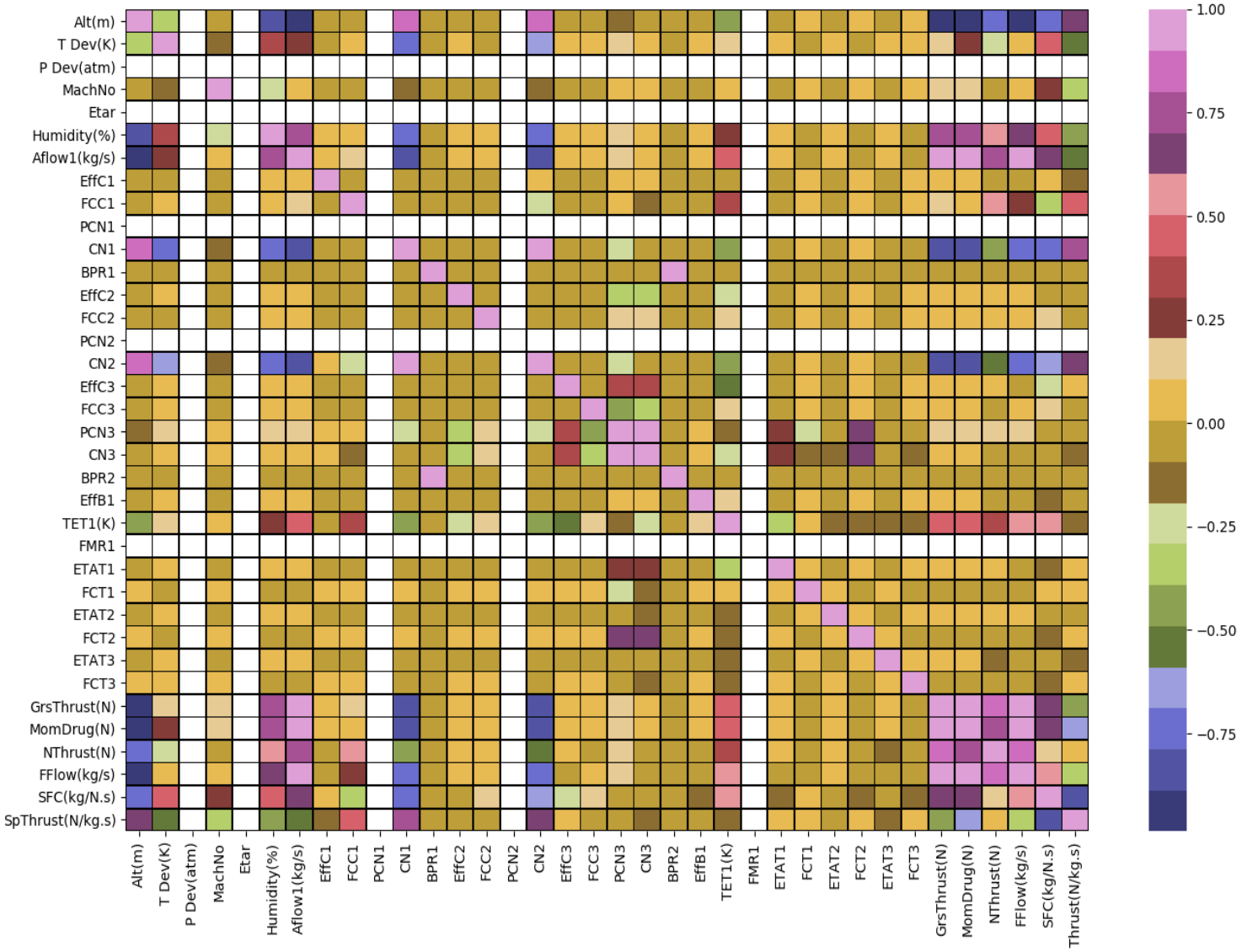


Figure C.2: Hot Section Pair-Plot

Appendix D

Correlation Heatmap



Appendix E

Python code to read PYTHIA File

```
import pandas as pd
import numpy as np
from pandas import read_csv
import glob
from matplotlib import pyplot as plt
import csv

plt.close("all")

path = r"C:\Users\Work\Desktop\OneDrive - Cranfield University\
Cranfield University\Personal Project\Neural Network Data"
file = glob.glob(path + "\*.txt")

df = []
Measurement = []
Degrade = []
Clean = []
Fan = []
```

```
IPC = []
HPC = []
HPT = []
LPT = []
IPT = []

Comp_dict = {}
num = 0

for count, filename in enumerate(file, start = 1):
    ToM = filename.split("\\")
    ToM = ToM[-1].lower()
    ToM = ToM.replace(".txt", "")
    if any (TM in "measurement" for TM in ToM.split("_")):
        df_temp = open(filename)
        df_temp = read_csv(filename)
        df_temp = df_temp.iloc[:,:-1]
        Component = ToM.split("_")[0].upper()
        if any(CE in "clean" for CE in ToM.lower().split("_")):
            df_temp["Component"] = Component
            if Component in Comp_dict:
                num = Comp_dict[Component]
            else:
                num += 1
                Comp_dict.update({Component : num})

            df_temp["Component Number"] = num
            Clean = df_temp
```

```
            Clean.append(Clean)

        else:
            df_temp["Component"] = Component
            if Component in Comp_dict:
                num = Comp_dict[Component]
            else:
                num += 1
                Comp_dict.update({Component : num})

            df_temp["Component Number"] = num
            Measurement = df_temp

    elif any(TM in "target" for TM in ToM.split("_")):
        if any(CE in "clean" for CE in ToM.lower().split("_")):
            df_temp = open(filename)
            df_temp = read_csv(filename)
            df_temp.columns = ["Target"]
            df_temp["Target0"] = 1
            df_temp["Target1"] = 0
            df_temp["Flow Capacity"] = 0
            df_temp["Efficiency"] = 0
            Clean = pd.concat([Clean, df_temp], axis = 1)
            df.append(Clean)

        else:
            df_temp = open(filename)
            df_temp = read_csv(filename)
```

```
            df_temp = df_temp.reset_index()
            df_temp.columns = df_temp.iloc[0]
            df_temp.columns = ["Flow Capacity", "Efficiency"]
            df_temp = df_temp[1:]
            df_temp["Target0"] = 0
            df_temp["Target1"] = 1
            Measurement = pd.concat([Measurement, df_temp], axis = 1)
            Measurement = Measurement.iloc[1:-1,:]
            df.append(Measurement)

df = pd.concat(df)
df.reset_index()
```

Appendix F

Regression Plots

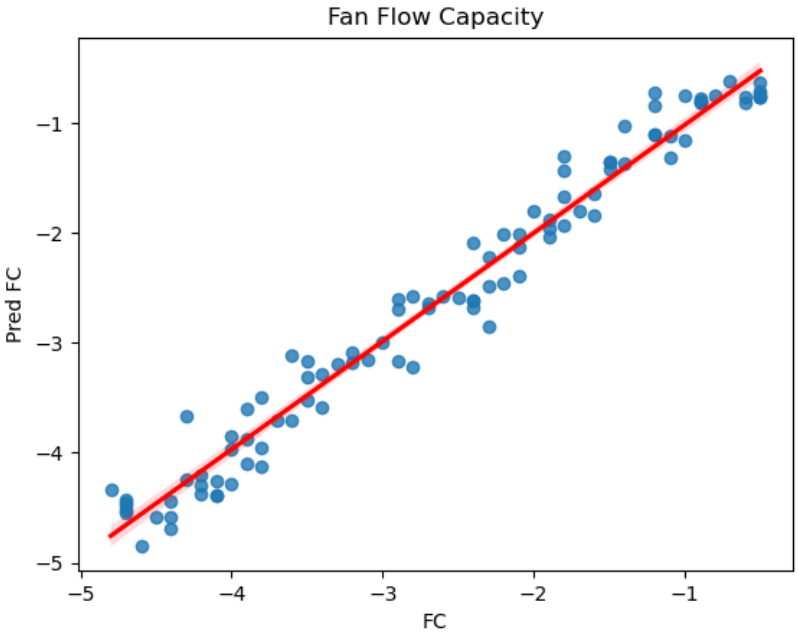


Figure F.1: Fan Flow Capacity Regression Plot

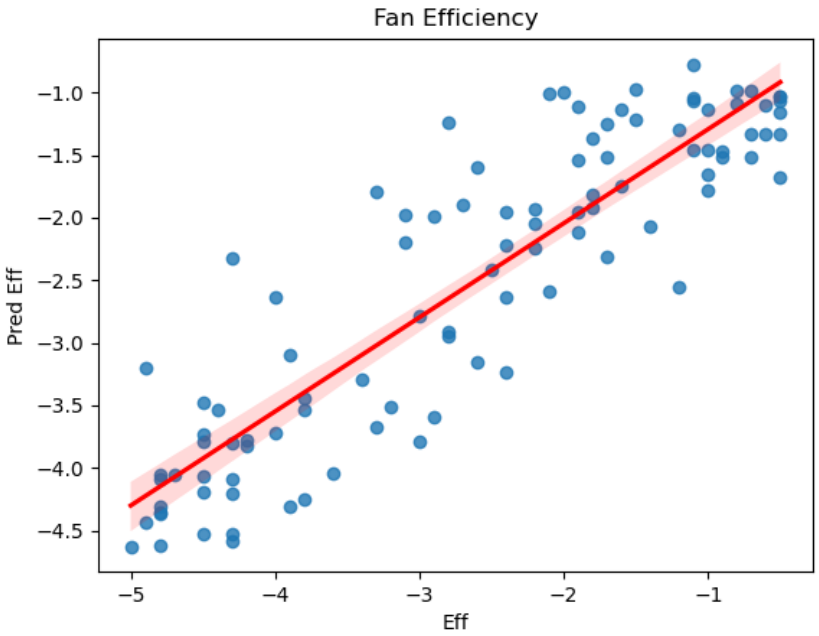


Figure F.2: Fan Efficiency Regression Plot

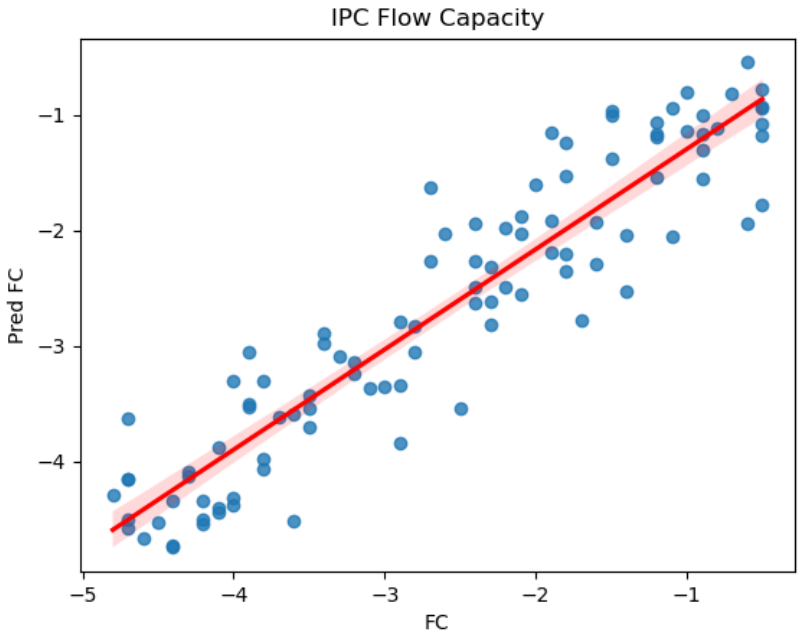


Figure F.3: IPC Fow Capacity Regression Plot

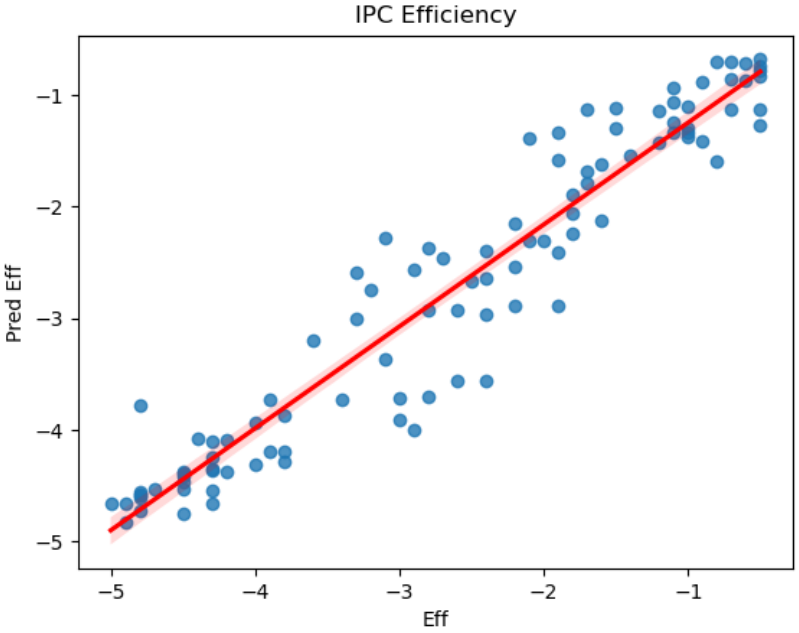


Figure F.4: IPCEfficiency Regression Plot

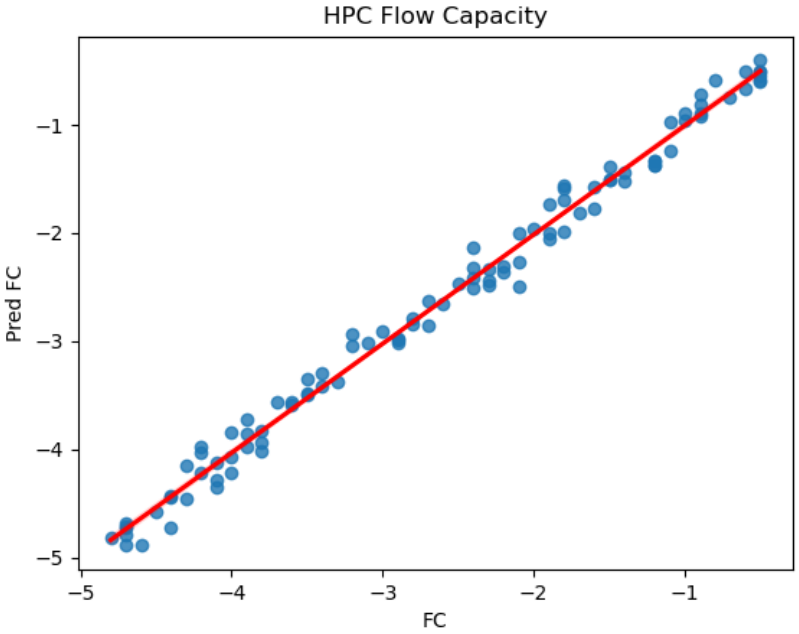


Figure F.5: HPC Flow Capacity Regression Plot

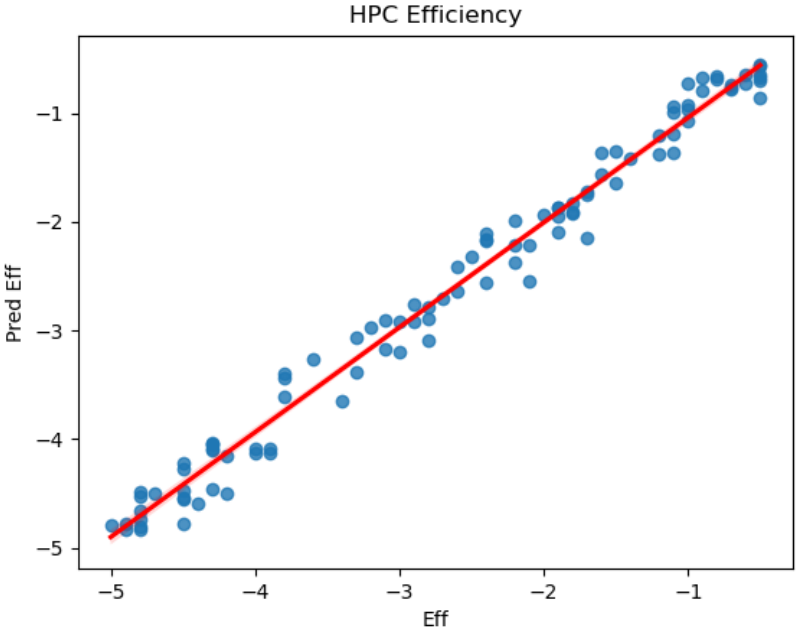


Figure F.6: HPC Efficiency Regression Plot

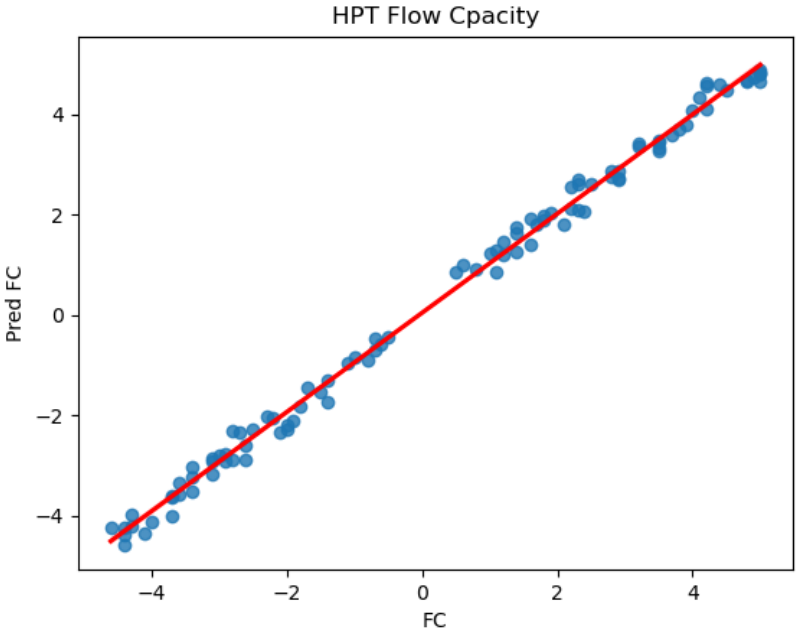


Figure F.7: HPT Flow Capacity Regression Plot

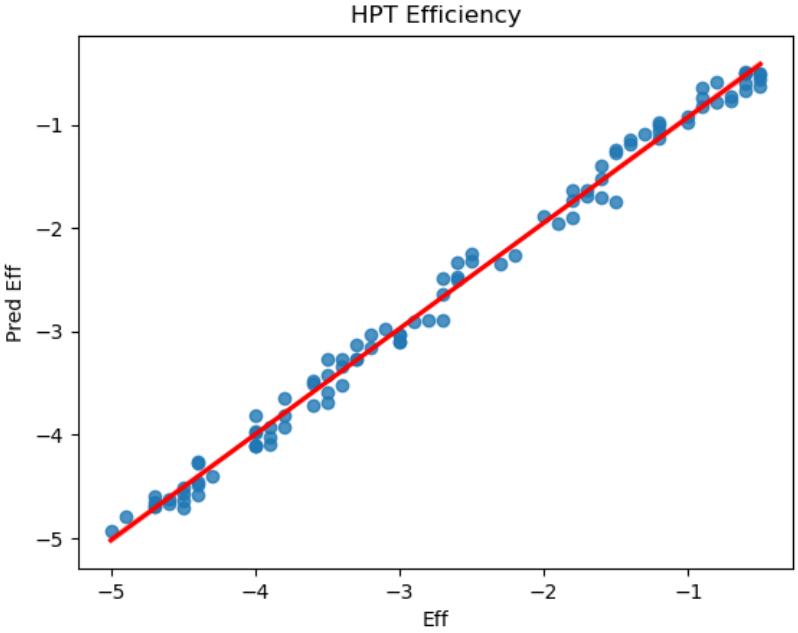


Figure F.8: HPT Efficiency Regression Plot

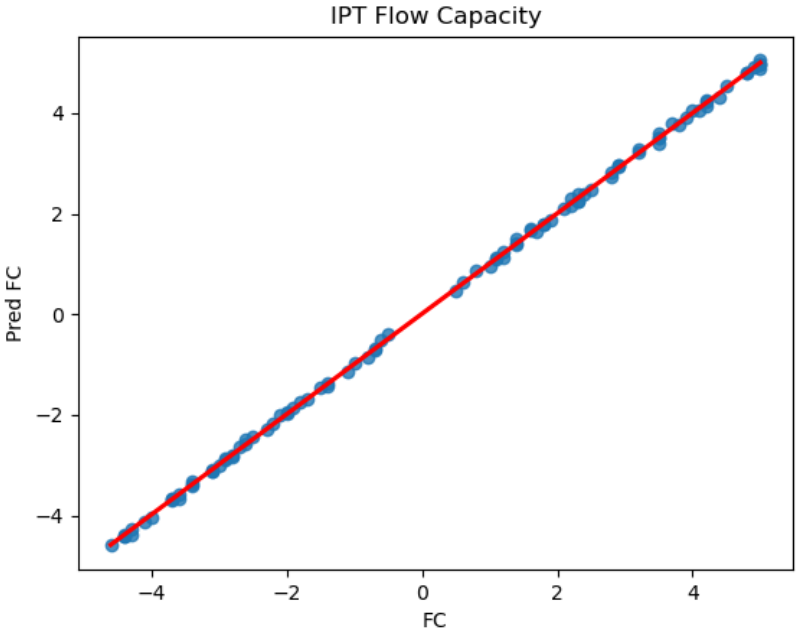


Figure F.9: IPT Flow Capacity Regression Plot

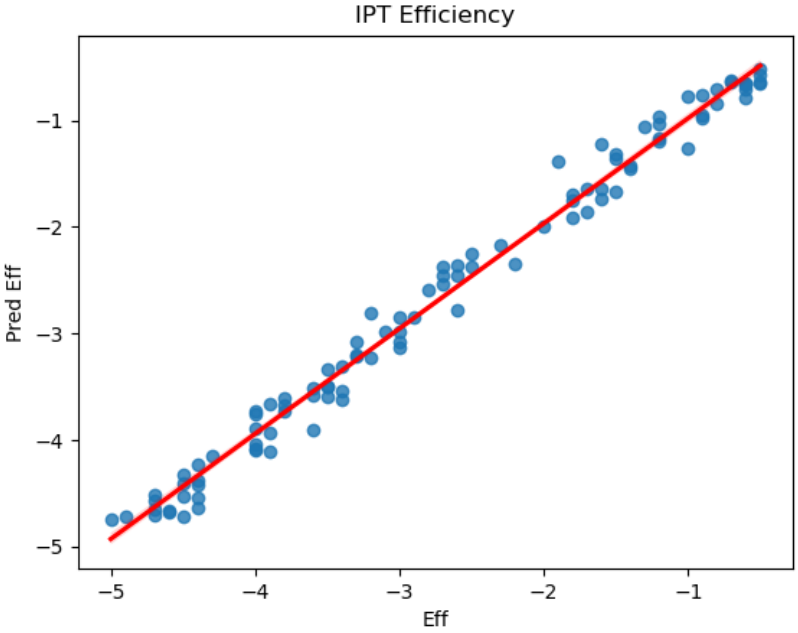


Figure F.10: IPT Efficiency Regression Plot

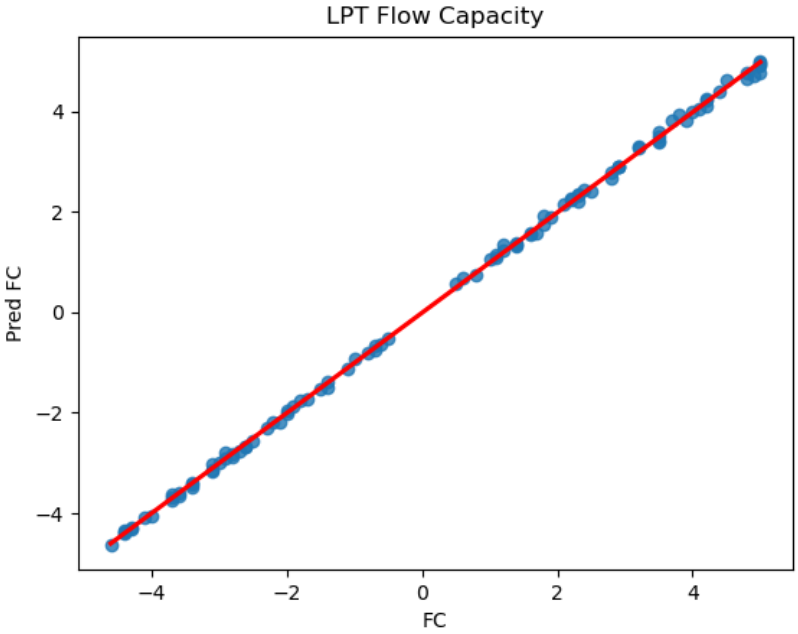


Figure F.11: LPT Flow Capacity Regression Plot

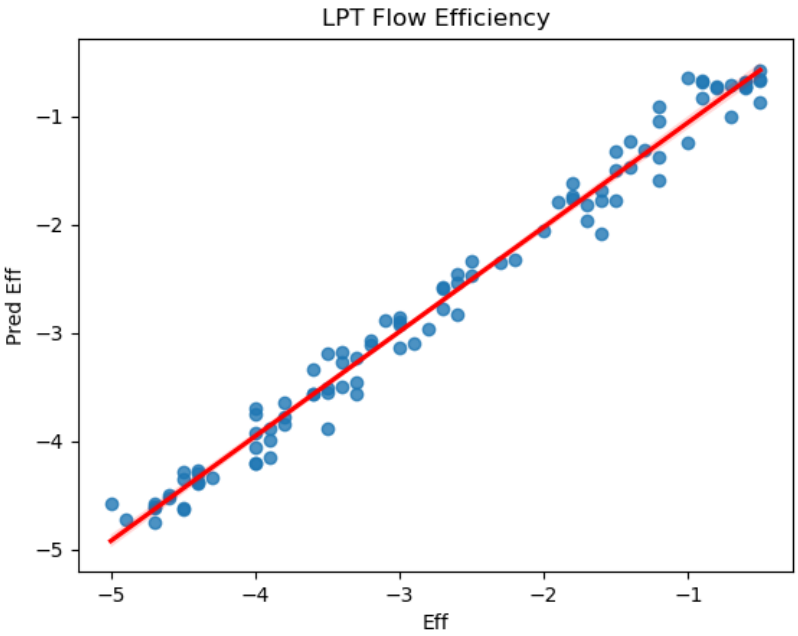


Figure F.12: LPT Efficiency Regression Plot

Appendix G

Diagnostic System Graphical User Interface

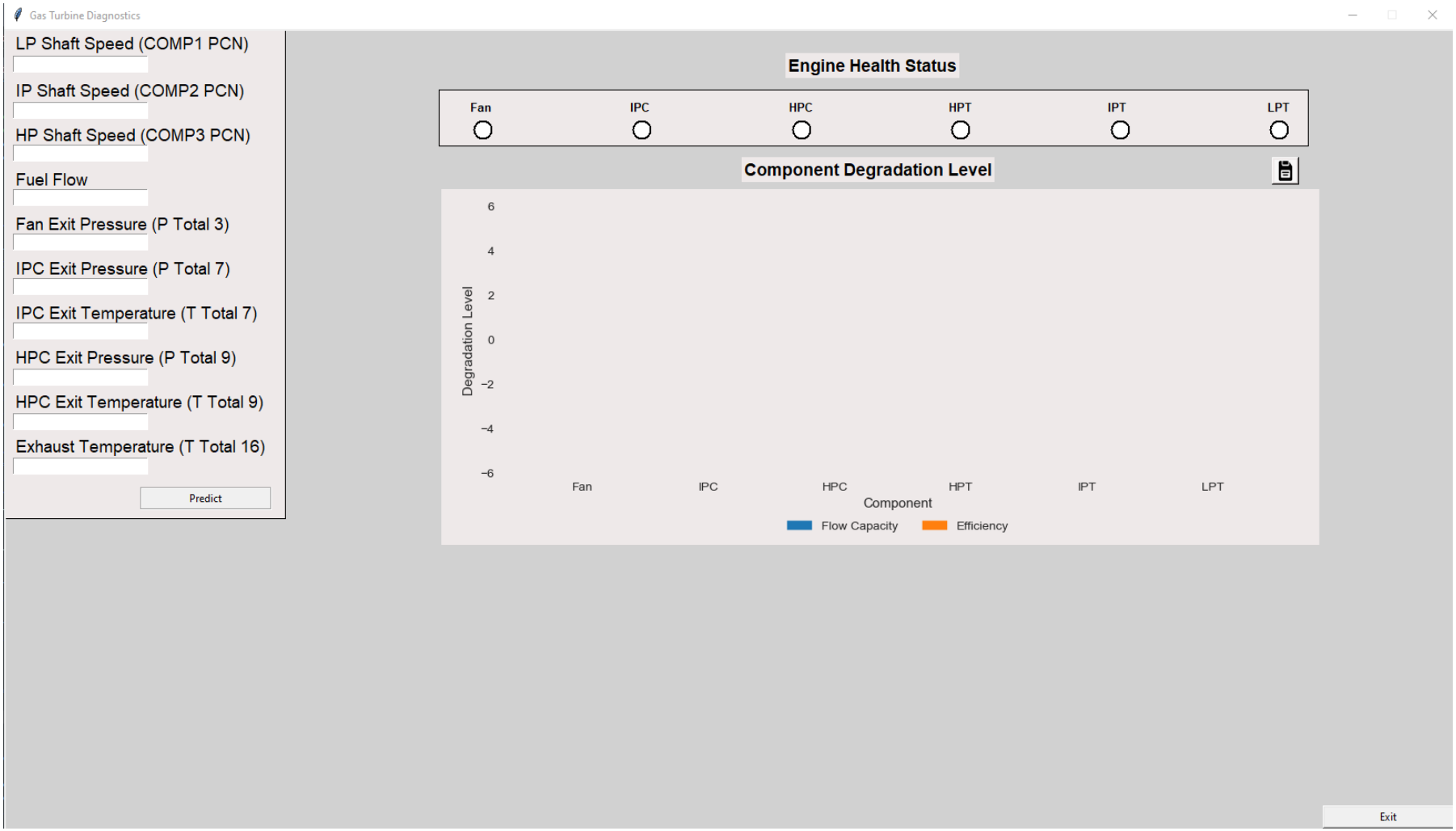


Figure G.1: Graphical User Interface

Appendix H

Fitness Function

```
def fitness_function(solution, sol_idx):
    global x_train, x_test, keras_ga, model, y_test, y_train

    Network_weight = model_weights_as_matrix(model = model,
                                             weights_vector = solution)

    model.set_weights(weights = network_weight)

    predict = model.predict(x_train)

    loss = BinaryCrossentropy()
    Model_fitness = 1/loss(y_train,predict).numpy() + 0.000001
    return Model_fitness
```