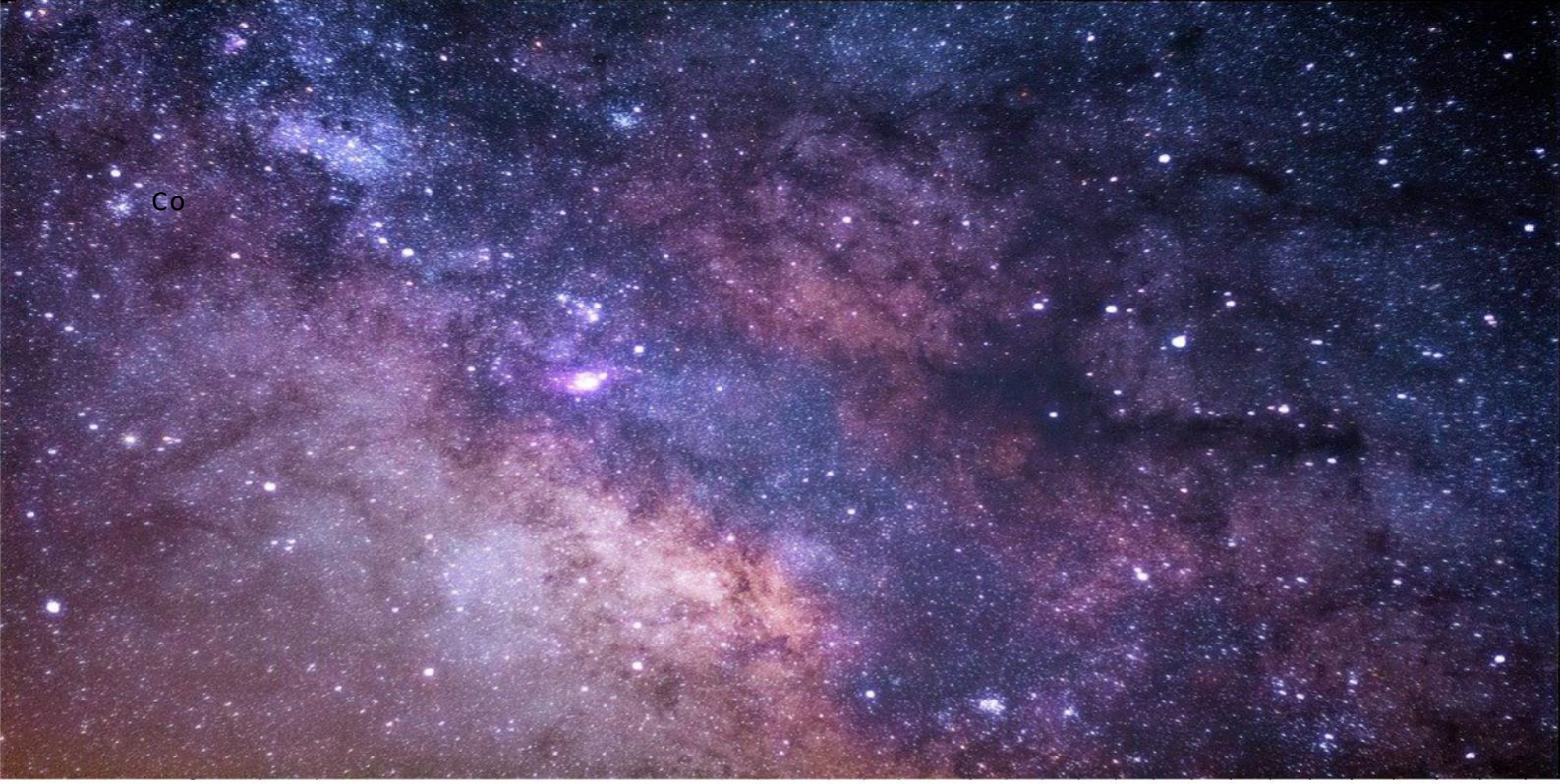


Fault detection of a controlled rectifier using machine learning

Mohamed Ameen

Sustainable Energy Engineering, OES10-03, 2020-06

Master’s Thesis

Fault detection of a controlled rectifier using machine learning

Mohamed Ameen

Sustainable Energy Engineering, OES10-03, 2020-06

Master's Thesis

Energy Technology
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Fault detection of a controlled rectifier using machine learning

Theme:
Fault detection and machine learning

Project Period:
Spring Semester 2020

Project Group:
OES-10-03

Participant:
Mohamed
Ameen

Supervisor(s):
Mohsen Soltani

Page Numbers: 70 including appendices

Date of Completion:
May 26, 2020

Abstract:

Power electronics are an essential part of the offshore industry, so it is essential that effective and accurate fault detection systems are put in place. This project focuses on the development of a machine learning algorithm for fault detection of the various components of a controlled rectifier. Data is collected from the rectifier in healthy and various faulty states, which is then used to train a neural network, which is a combination of convolutional neural networks and artificial neural networks for fault detection. The final algorithm has achieved an accuracy of 99.5% using simulation data.

Contents

PREFACE

This report is part of a research project done in the Offshore Energy Engineering department at AAU Esbjerg as part of a long master's thesis from the 9th semester to the 10th semester. The focus of this thesis is to develop a fault detection algorithm using machine learning for a controlled rectifier.

I would like to thank my supervisor Mohsen Soltani for always providing support when needed and pointing me in the right direction. I would also like to thank Henry Enevoldsen, who was helping me setup the required apparatuses before the COVID 19 crisis caused the shutdown. I would also like to thank my family for their constant and unwavering support.

Aalborg University Esbjerg, May 27, 2020

_____

Mohamed Ameen

<mameen18@student.aau.dk>

1. INTRODUCTION

This project report elucidates the various goals and methods that were used during the project. This chapter will give a brief overview of the project and set up the goals of the project. In this chapter the following will be discussed:

- A brief overview of power electronics and their condition monitoring
- Formulation of the objective
- Definition of the goals and delimitations of the project
- Steps involved in the project

1.1 USE OF CONDITION MONITORING AND POWER ELECTRONICS IN OFFSHORE ENERGY

Condition monitoring of wind turbines is becoming more and more prevalent in the wind industry to keep turbines running at optimum performance especially offshore wind turbines, since they are harder and more expensive to keep running optimally[1]. Condition monitoring involves the observation of a wind turbines components to see if any changes in their behaviour can tell us of an incoming fault, and it has also been seen that robust condition monitoring has to potential to reduce operations and maintenance costs significantly by detecting a fault before it is severe[2].

When considering existing condition monitoring methods of a wind turbine it can be divided into different types based on the method of approach. Some main ways it can be divided are as follows[3]:

- **Granularity:** A condition monitoring system can be applied at the smallest subsystem level, such as power electronics or it can also be applied on a much larger scale like a whole wind farm.
- **Intrusiveness:** A system can also be classified on the basis of impact it has on the component being monitored. So a system involving vibration and oil debris analysis would be considered as intrusive but a system involving power signal monitoring would be considered non-intrusive.
- **Prediction**: If a system is being designed to predict faults then it would be a fault prognosis system whereas if it was being designed to detect faults as they happen it would be a fault detection system.

When deciding which components to apply condition monitoring to, the failure rates and downtimes per failure need to be considered. Based on that, priority needs to be given to components that are more likely to fail or those components whose failure can lead to long downtimes and high costs.

Condition monitoring of wind turbines is not a very recent development, so traditional methods of fault detection and diagnosis have been applied, but with the advent of increased computational power, one method in particular seems to have a significant advantage when it comes to condition monitoring and fault detection and that is Machine Learning.

Machine learning is a method that involves building a model from a limited set of data and without much intervention from engineers. As the name suggests, a model based on machine learning will be able to find underlying patterns which can be used to enhance the capability of the system to detect faults, this process is a cyclical one. Machine Learning can also be divided into two broad categories, supervised learning, which predicts an output based on previously labelled data which it has trained on and unsupervised learning, which is able to classify data without labelled inputs.

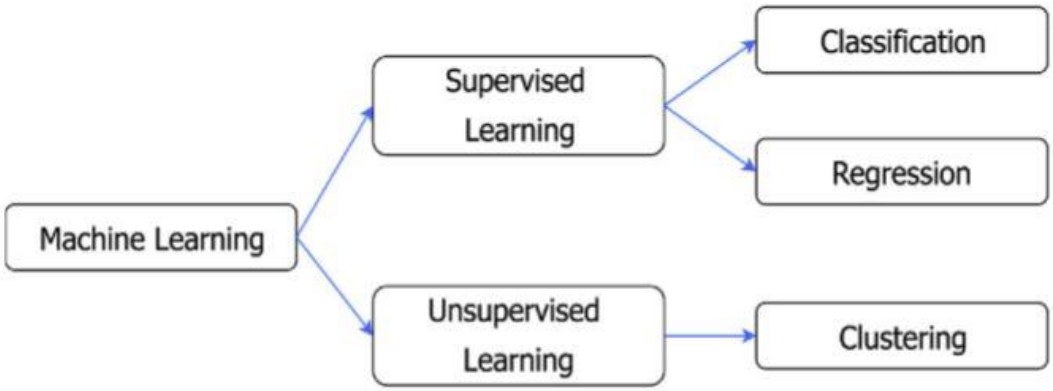


Figure 1 Basic taxonomy of Machine Learning

1.2 GOALS AND OBJECTIVE OF THE PROJECT

Based on the criteria stated in the previous section it was decided that the system would be implemented on a single phase rectifier using a non-intrusive method which involves measuring the voltages in the system and that it would be a fault detection system, being able to detect the faults as they occur.

So the main objective of this project is: ”to design a fault detection system for a single phase controlled rectifier using machine learning”

Based on the above objective the goals of this project are as follows:

- Obtain practical and application based knowledge about machine learning.
- Design and implement a simple single phase controlled rectifier.
- Collect data on healthy and faulty systems.
- Gain an understanding into how the algorithms detect the faults in the system.
- Validate the algorithm using the data collected.

1.3 DELIMITATIONS OF THE PROJECT

The purpose of this project is to implement a fault detection system based on machine learning, but one of the main drawbacks of machine learning is that it requires a lot of data on healthy systems and faulty systems. Since there are no freely available datasets for wind turbines it was decided that the data would be collected in a lab setup and a simulation. Both would then be subject to emulated faults to collect data on faulty systems. The device was chosen to be a single phase controlled rectifier because the focus of this project is to be on the machine learning aspect and not on the modelling and control of the devices and the single phase controlled rectifier is relatively simpler than a three phase controlled rectifier to model and introduce faults into.

1.4 METHOD OF THE PROJECT

To accomplish the goals that have been set for this project, the following steps were devised:

Step 1: Studying machine learning algorithms: Machine learning algorithms need to be studied to determine which algorithms are the best suited for the task set. A comparison between algorithms will be done at a later stage.

Step 2: Preparation of set up: The setup needs to be prepared and research needs to be done regarding the device to be tested and it's various faults. Also a safe method of introducing the various faults must be determined.

Step 3: Data Collection: Data needs to be collected on both healthy systems and unhealthy or faulty systems, this data also needs to be labelled correctly for training the machine learning algorithms.

Step 4: Data processing: The data collected needs to be cleaned of outliers and divided into 2 batches for training the algorithm and for testing it.

Step 5: Training algorithms: The batch of data set for training is used to train various machine learning algorithms, based on the most efficient and unbiased way to do so to avoid false positives in the testing.

Step 6: Testing algorithms: The batch of data set aside for testing is used to test the machine learning algorithms and it is noted how accurate each algorithm is using various metrics.

Step 7: Comparison and conclusion: The accuracy metrics in the previous step for each algorithm need to be compared to see which algorithm has the highest accuracy, and that is the best candidate for the task. This loops back to step 4 until an algorithm of the highest accuracy for detection is achieved.

2. TOPOLOGY AND EXPERIMENTAL SETUP

In this chapter the device that condition monitoring system is chosen to be implemented on is discussed in detail along with the design and implementation of the experimental setup.

2.1 SINGLE PHASE RECTIFIER

The device that was chosen to be examined was a single phase controlled rectifier. Rectifiers are used in the offshore wind energy to convert the generated alternate current into DC for more efficient transmission to onshore substations where it is converted back to AC for supplying the grid. However single phase rectifiers are not used in the industry as much but the methodology used in this project could very easily be used to scale up the algorithm to a 3 phase rectifier provided that data is provided on healthy and faulty systems.

A single phase controlled rectifier has an inductor connected in series with the source to allow for stepping up of the output current to desired values. This rectifier is like a regular full wave rectifier in terms of diode placement but it also has 4 antiparallel IGBTs in order to enable the stepping up and phase control of the system. The advantage of this system is that it is able to step up the output voltage unlike a normal full wave rectifier and it can also be used for controlling the power factor of the system to allow for better power efficiency. So basically, the PWM controlled rectifier has more potential and ability to "shape" the output voltage than an ordinary rectifier.

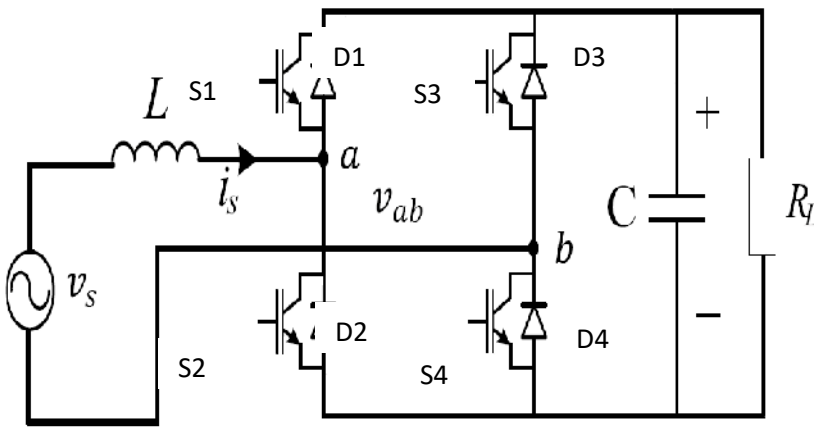


Figure 2 Circuit diagram of PWM Single Phase Rectifier

The basic principle behind the PWM single phase rectifier is that the voltage being supplied to the rectifier has a source voltage and the voltage that is separated by the input inductance, and due to the nature of the rectifier allowing bidirectional power flow, this inductive voltage is also used in the system allowing for more ability to shape the output voltage than an ordinary rectifier. The working of the rectifier will be explained in more detail later.

From the diagram it can be seen that there are 4 main components that could fail or that could have faults introduced, those are the capacitor, the inductor, the diodes and the transistors. So when looking to describe a fault detection algorithm, these devices must be carefully considered. Moreover in surveys conducted it has been found that switching devices or power devices account for 38% of faults in variable speed drives[4] and that capacitors and semiconductor devices together account for around 50% of all power electronics faults[5].

When choosing the values of the components for the controlled rectifier the most important one to be chosen carefully is the capacitor on the DC side. The capacitor is chosen based on calculations made for the ripple voltage, the formula for the ripple voltage is given as:

$$V_{ripple} = \frac{I(load)}{f * C} \tag{1}$$

The above formula is rearranged as follows:

$$C = \frac{I(load)}{f * V_{ripple}} \tag{2}$$

In the above formula, if $V_{ripple} = 0.8\,V$ and I=0.04 A and f=50Hz, the value of the capacitance required is C=1000 µF at R= 100 Ω.

Based on the above calculations the choice of the values of the components are as follows:

Variable	Value	Description
L	1e-3 H	Input inductance
C	1000µF	Output capacitance
R	100 Ω	Load

Table 1 Value of the components used

2.2 MODELLING AND CONTROL

Before the setup is implemented, the system was first designed and implemented in Simulink along with basic control to check if it is working. The system was modelled for a very low power consumption of around 2-3 watts and the values of the components were chosen accordingly.

2.2.1 WORKING PRINCIPLE

In figure 2, the boost part of the rectifier is handled by the inductor and the 4 power electronic switches and the rectification part is handled by the 4 anti-parallel diodes with suitable current carrying capacity in a H-bridge configuration, the capacitance on the DC side of the rectifier is used to smoothen out the DC output voltage and a suitable value must be chosen for it in accordance with how much ripple voltage is desired. Assuming that the supply voltage v_s is sinusoidal, the equation for the AC side of the converter under steady state conditions can be given by the following equation:

$$v_s = j\omega L i_s + v_{ab} \tag{3}$$

The equation governing the voltage on DC side of the converter generated by PWM can be given by the following equation:

$$v = m v_{DC} \tag{4}$$

Here m is the modulation index or the ratio of the reference voltage to the triangular or sawtooth carrier wave for the PWM, given by the following equation:

$$m = \frac{V_{sin}}{V_{tri}} \tag{3}$$

The inductor L in the circuit will store energy when either switch S2 or S4 is closed and will release the energy when those switches are open allowing for a boost in the output DC voltage. In the positive half when cycle switch S2 is closed and the circuit is completed with the current flowing through D4 to complete the circuit and charge the inductor, then when the switch S2 is opened, the current will flow through D1 to the load and back through D4 to the source. Similarly during the negative half cycle when switch S4 is closed the current flows through the inductor and back through D2, and when the switch is opened the stored energy is discharged through D3 and back through D2. From equation 1, the following phasor diagram is obtained

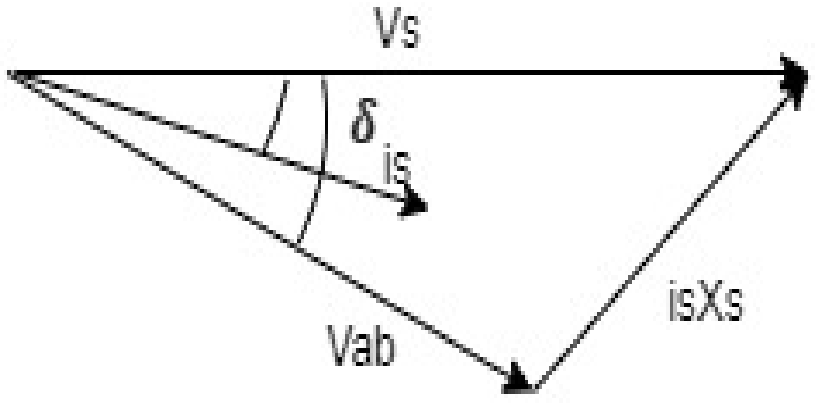


Figure 3 Phasor diagram of the rectifier

According to the above phasor diagram the power delivered from the input to the output side is given by the following equation:

$$P = \frac{v_s v_{ab}}{2\pi f L} \sin\delta = v_s i_s \cos\varphi \quad (4)$$

2.2.2 CONTROL DESIGN

The goal of the control to be designed is twofold:

- Control the power factor to unity
- Control the DC voltage to a desired value

From equation 4 and the phasor diagram it can be seen that in order to control the power factor to unity the source current must be in phase with the voltage and in order to control the DC output voltage to a desired value, the modulation index m is to be manipulated. Based on the above a simple PI controller was designed to fulfil the objectives of the control as seen in the following figure.

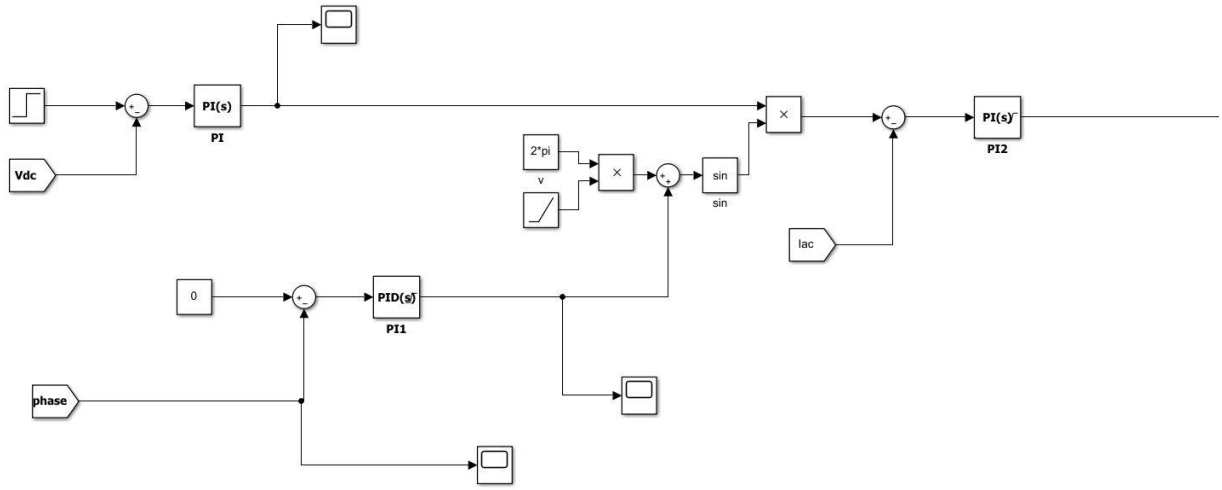


Figure 4 Simulink model of the control

The results of the Simulink model with the above defined control are shown below and it can be seen that in the simulation the designed control is fulfilling the objectives it was designed for.

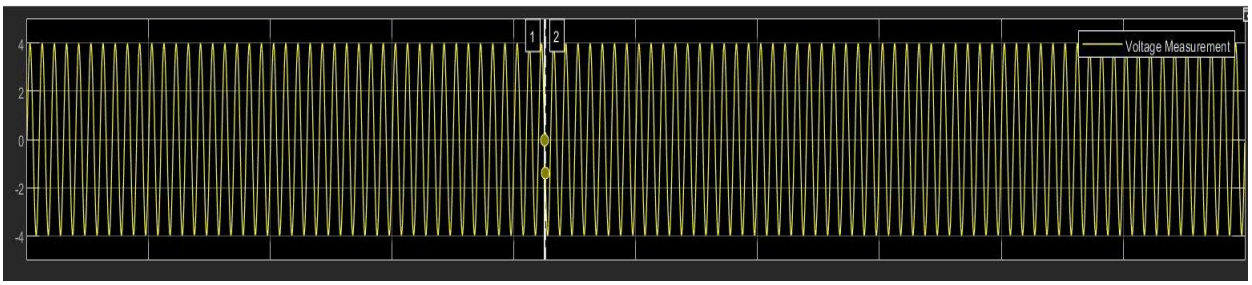


Figure 5 AC input voltage

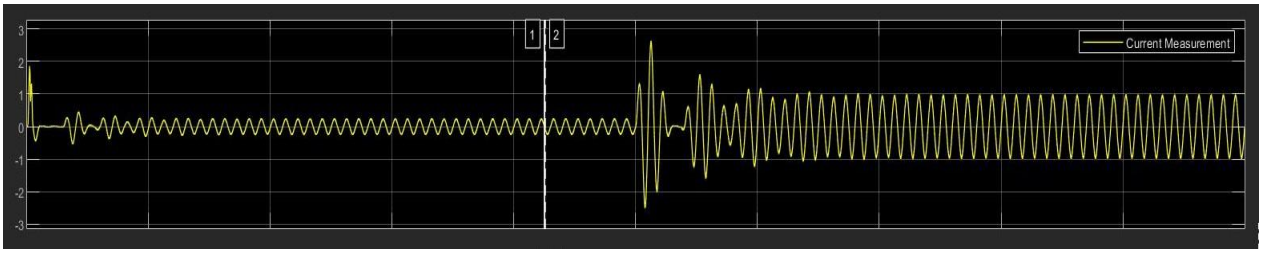


Figure 6 AC input current

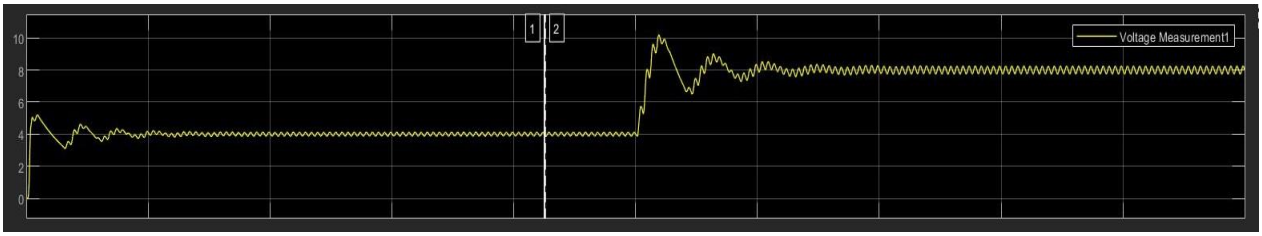


Figure 7 DC output voltage

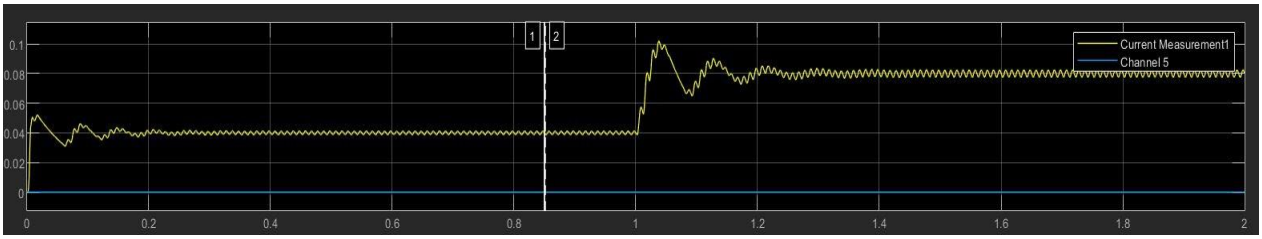


Figure 8 DC output current

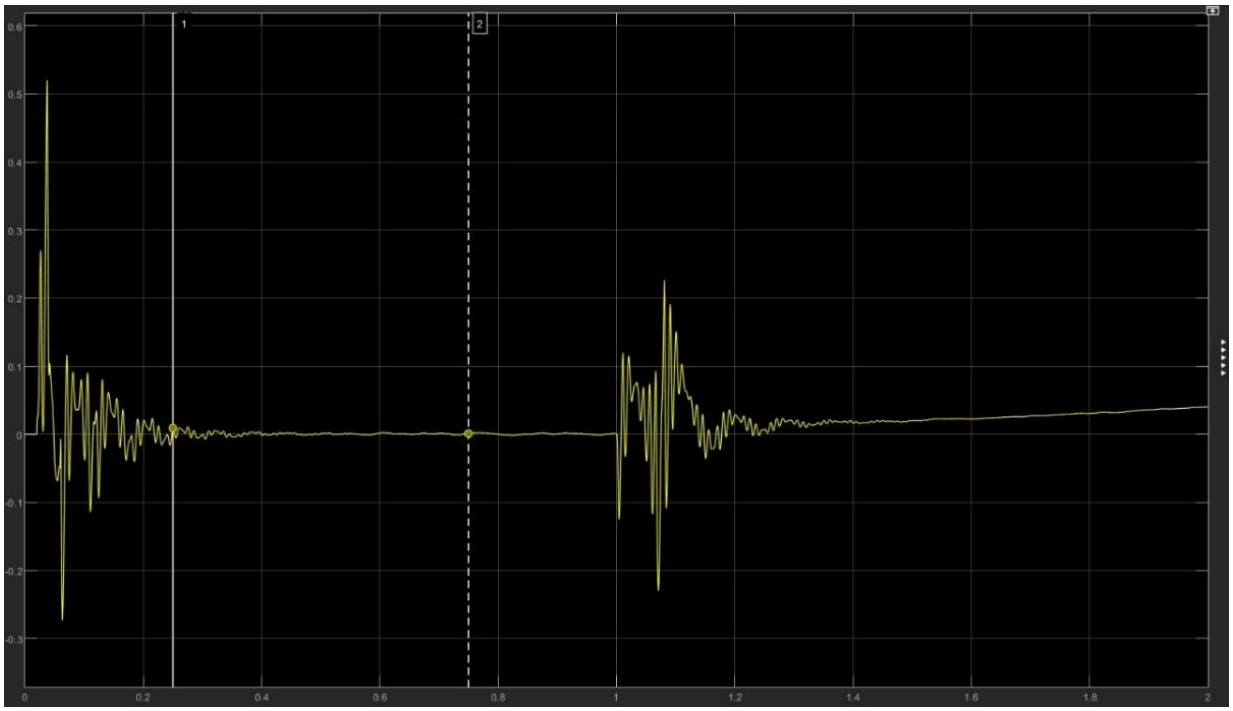


Figure 9 Phase difference between voltage and current

Since the designed controller was working in the simulation, the control was then implemented on a real system in the laboratory, to check if it was working with the following component values

Variable	Value	Description
L	1e-3 H	Input inductance
C	1000µF	Output capacitance
R	100 Ω	Load

Table 2 Values of the components used

The following results were obtained for the experiment,

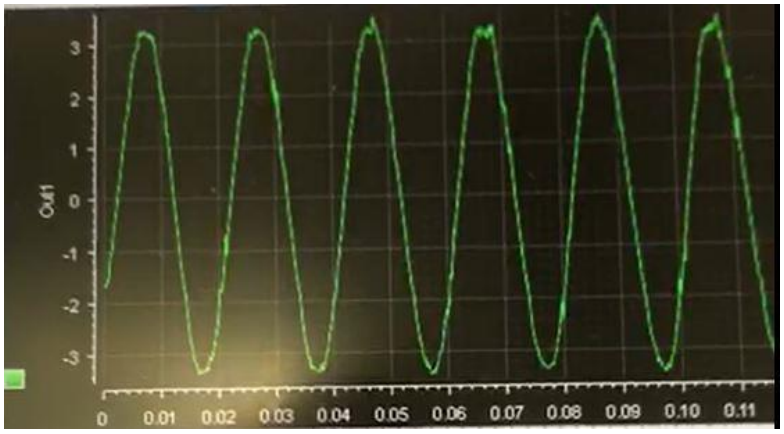


Figure 10 Input voltage

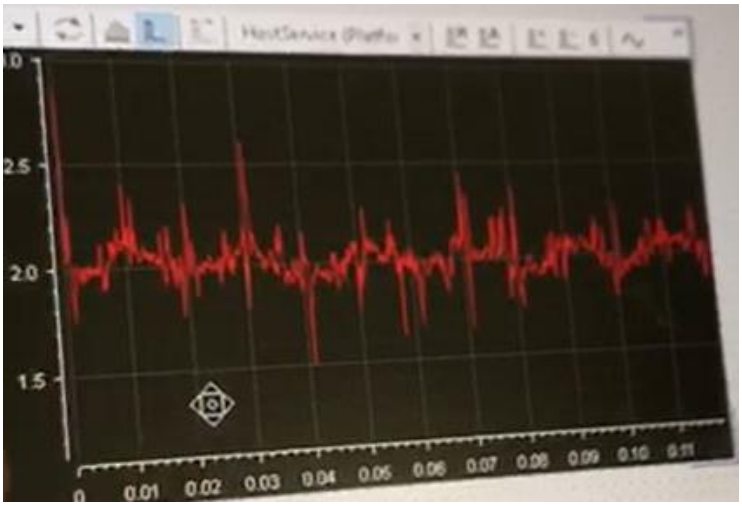


Figure 11 Output Voltage

The control is working in the simulation and was verified as well on a pre-existing setup in the lab, so the control is ready for use in the data collection process.

3. MACHINE LEARNING AND NEURAL NETWORKS

3.1 INTRODUCTION TO MACHINE LEARNING

Despite the recent surge of interest in them, machine learning and specifically neural networks have been around for around 50 years. Due to a rapid increase in computing power, neural networks are becoming more and more powerful tools in today's society, and this has seen their popularity increase over the past decade. As the name implies neural networks are networks of programmed "neurons" that are based on the way that biological neurons behave in a brain. In fact the first inspiration for a neural network came when scientists examined a cat's visual cortex. Neural networks that have been programmed are called Artificial Neural Networks or ANNs, there are many variations of the ANN, but most have a very similar structure.

Artificial neural networks learn from a set of sample data, and after their "training", if the network is given data that it has not been exposed to before, the output will be something very close to the desired output, so it is not just memorization of the values but also learning work with new data as well, the steps to train the network are called learning algorithms[6]. Machine learning takes advantage of computational power and doesn't need to rely on mathematical models to work. If the sample of training data is large and varied the algorithm continuously improves itself as it is trained, however that is also a drawback of machine learning, a large sample of data is needed for the network to be trained accurately.

As briefly mentioned in the introduction and figure 1, machine learning can be broadly divided into 2 categories: Unsupervised and supervised learning. An unsupervised learning algorithm involves using unlabelled data to see if any pre-existing patterns that are not detectable by humans might be present and supervised learning involves giving the model a set of labelled or known data and getting the response for new or unknown data, an example for this is to use the data of the last 10 years' seasonal electricity load to predict or forecast what the load for the next year will look like.

3.1.1 Neurons

Before discussing the structure of a neural network, the most basic part of an ANN must be defined, the neuron. A neuron is the basis of a neural network and it's structure can be seen in the figure below .

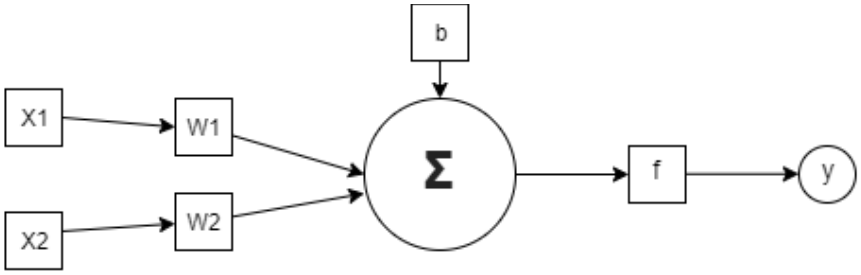


Figure 12 Single neuron

Here $X1$ and $X2$ are inputs, that are then multiplied by their respective weights $W1$ and $W2$ and are then added together with a bias b, this is called the activation potential and is denoted by i, then it is used as the input for a chosen activation function f, the output of which is the output of the neuron, most neurons follow this similar structure. In mathematical form the neuron can be written as

$$i = X1 * W1 + X2 * W2 + b \tag{5}$$

$$y = f(i) \tag{6}$$

The above equations are usually written in matrix form to allow for easier representation when the structure of the network becomes too complex to write down in equation form.

$$X = \begin{bmatrix} X1 \\ X2 \\ 1 \end{bmatrix} \tag{7}$$

$$W^1 = [W1\ W2\ b] \tag{8}$$

$$i = W^1 * X \tag{9}$$

In the above equation X is the input matrix and W^1 is the weight matrix, the superscript on the W is to denote the neuron number which is important when the structure of the neural network is discussed later in this section of the report. The other final aspect to consider in the structure of a neuron is the activation function. The activation function is the final calculation being done before the output of the neuron is determined, activation functions are of many different types and ranges and each pose their own advantages and disadvantages, some activation functions and their graphs are shown in the table and figure below.

Name	Function	Range
Hyperbolic	$f(i) = \tanh(i)$	$f(i)\ \epsilon\ (-1,1)$
Sigmoid	$f(i) = \frac{1}{1+e^i}$	$f(i)\ \epsilon\ (0,1)$

Softplus	$f(i) = \log(e^i + 1)$	$f(i)\ \epsilon\ (0,1)$
RELU	$f(i) = \max(0, i)$	$f(i)\ \epsilon\ (0, \infty)$
None(identity)	$f(i) = i$	$f(i)\ \epsilon\ (-\infty, \infty)$

Table 3 Comparison of different activation functions

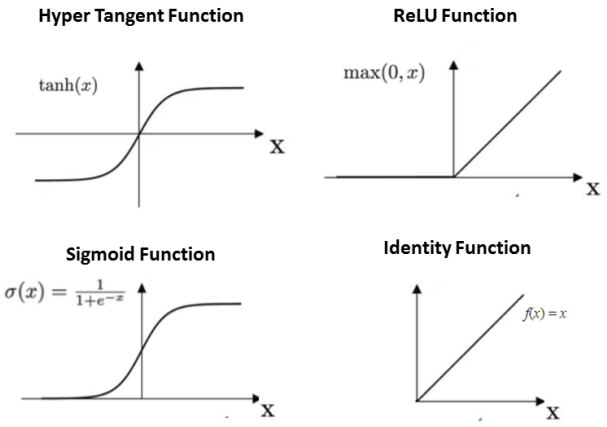


Figure 13 Common activation functions

Single neurons by themselves are not very powerful computational tools and not much other than basic classification can be done by them, this is why multiple neurons are used in tandem and parallel with each other to make layers and these layers are stacked in order to construct what is known as an artificial neural network.

3.1.2 Neural Networks

The basic structure of a neural network is generally similar regardless of the type of neural network and can be seen in the figure below.

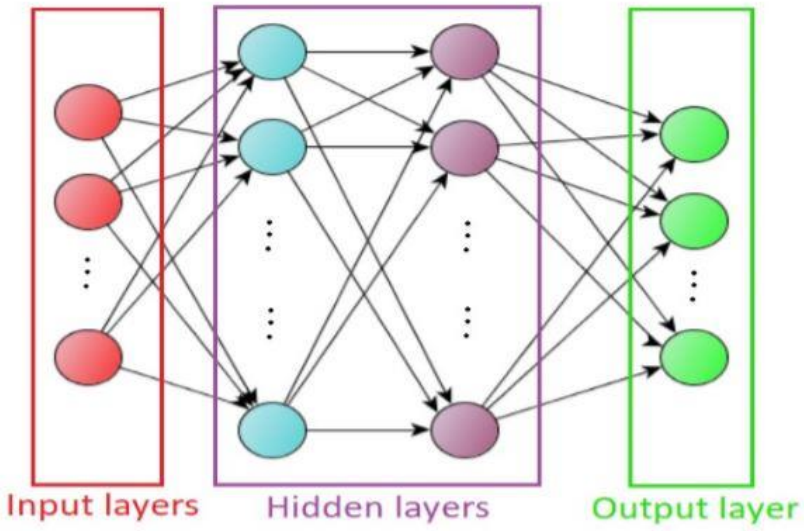


Figure 14 Structure of a neural network

In general as can be seen from figure 14 a neural network can be divided into 3 main parts:

- **Input Layer**: This layer is the one that receives the data to be analysed such as signals or measurements from the system, there can only be one input layer, but the number of neurons in the layer are not limited.
- **Hidden Layer**: The hidden layer can be made of multiple layers of neurons depending on what is needed from the network, this layer is responsible for "feature extraction" or determining the patterns present in the input data that can be analysed. There is no limit to the number of hidden layers that can be used in a neural network but it is important to keep in mind that with an increase in the number of hidden layers the complexity of the network also increases and so does computation time.
- **Output Layer**: This layer is made of neurons that determine the final output of the network and like the input layer there can only be one, usually the number of neurons in the output layer is much lesser than the number of neurons in the hidden layers and sometimes the input layer as well.

One of the most common types of neural networks that are being used and are used in this project is called a multilayer feedforward neural network, which is made of multiple interconnected neurons that "learn" the relationships between the inputs and the outputs[6]. Given enough layers in the hidden layers part of the network, it could theoretically learn the relationship between any 2 sets of inputs and outputs but in reality after some number of hidden layers it can lead to overfitting of the data and slow computation times due to the large number of calculations. A simple 3 input neural network with 2 hidden layers is shown in the diagram below, this was an arbitrary choice to illustrate the working of a neural network, however when designing a neural network the designer has to specify the following parameters[6]:

- Number of neurons in each layer
- Activation functions of the neurons
- Number of hidden layers in the network

The other parameters are initialized randomly and are taken care of by the program. A simplified diagram of the previously mentioned neural network can be seen in figure 15, the structure of the neurons is as described in section 3.1.1 and though the number of inputs to each neuron in the hidden layers changes, the neurons are otherwise similar to each other. Every neuron has one output and one activation function and the activation functions for each neuron can be changed if it desired to do so. In the figure above the first hidden layer is represented by H and the second hidden layer is represented by H', the bias, weights and activation function are not represented in the diagram.

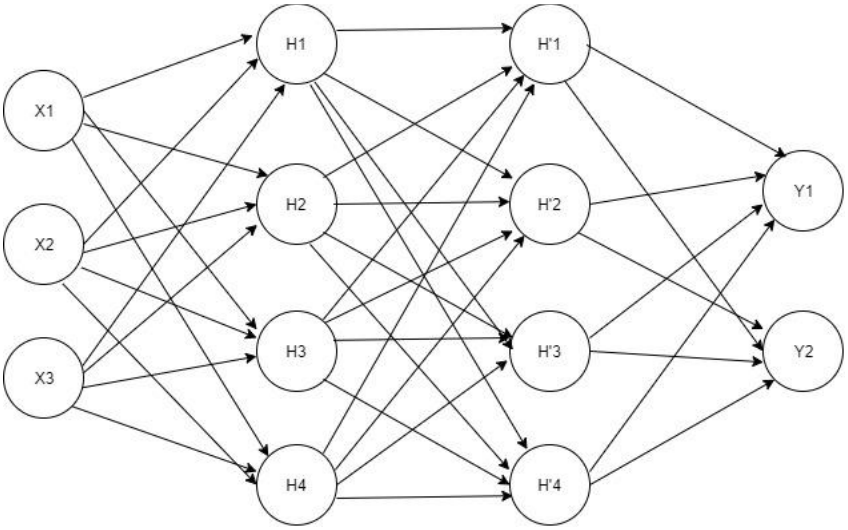


Figure 15 A 3 input 2 hidden layer neural network

Now to mathematically represent the neural network in figure 16, the same matrix methodology that was applied in section 3.1.1 to mathematically define a neuron is used here. Each neuron when it is connected to a neuron in another layer, it is multiplied by a weight as explained previously.

$$X = \begin{bmatrix} X1 \\ X2 \\ X3 \\ b \end{bmatrix} \tag{10}$$

$$W^1 = \begin{bmatrix} W11 & W12 & W13 & 1 \\ W21 & W22 & W23 & 1 \\ W31 & W32 & W33 & 1 \\ W41 & W42 & W43 & 1 \end{bmatrix} \tag{11}$$

$$W^2 = \begin{bmatrix} W11 & W12 & W13 & W14 & 1 \\ W21 & W22 & W23 & W24 & 1 \\ W31 & W32 & W33 & W34 & 1 \\ W41 & W42 & W43 & W44 & 1 \end{bmatrix} \tag{12}$$

$$W^3 = \begin{bmatrix} W11 & W12 & W13 & W14 & 1 \\ W21 & W21 & W23 & W24 & 1 \end{bmatrix} \tag{13}$$

The above are the weight matrices for the figure 6 and the first number following the W denotes the output neuron for that weight in the next layer and the number after that denotes the number of the neuron from the input layer so for example W13 in W^3 is referring to the weight connecting the 3rd neuron in the 3rd layer to the 1st neuron in the 4th layer. Also it can be seen that the last column in each of the weight matrices are all ones, this is because that column represents the weight of the bias b and as seen previously the bias has no defined weight. Now going forward using the method established in the previous section, the activation potentials are calculated and then given to the activation function this is then fed to the next layer and the process is repeated until the last layer is reached.

$$I1 = W^1 * X = \begin{bmatrix} i11 \\ i12 \\ i13 \\ i14 \end{bmatrix} \tag{14}$$

$$H = f(I1) = \begin{bmatrix} f(i11) \\ f(i12) \\ f(i13) \\ f(i14) \end{bmatrix} \tag{55}$$

A bias is added to the vector H as an additional row and the process is repeated until the last layer is reached where the following relation is determined

$$I2 = W^2 * H = \begin{bmatrix} i21 \\ i22 \\ i23 \\ i24 \end{bmatrix} \tag{16}$$

$$H' = f(I2) = \begin{bmatrix} f(i21) \\ f(i22) \\ f(i23) \\ f(i24) \end{bmatrix} \tag{17}$$

$$I3 = W^3 * H' = \begin{bmatrix} i31 \\ i32 \end{bmatrix} \tag{18}$$

$$Y = \begin{bmatrix} Y1 \\ Y2 \end{bmatrix} = \begin{bmatrix} f(i31) \\ f(i32) \end{bmatrix} \tag{19}$$

The above was a mathematical explanation of how a feed forward neural network works, the weights of the neurons are adjusted by minimizing the error by use of a cost function and the delta rule, which will be explained in a later section.

Based on this various types of neural networks can be constructed by making some tweaks to the structure of the network, but throughout these various types of networks the working principle and basic structure remains the same.

3.2 OTHER TYPES OF NEURAL NETWORKS

Owing to the structure of a neural network there are various other types of neural networks that can be made by combining neurons in multiple different ways and by adding additional functionality, some examples of these are recurrent neural networks, perceptrons and deep neural networks, the structure of some of these can be seen in the figure below

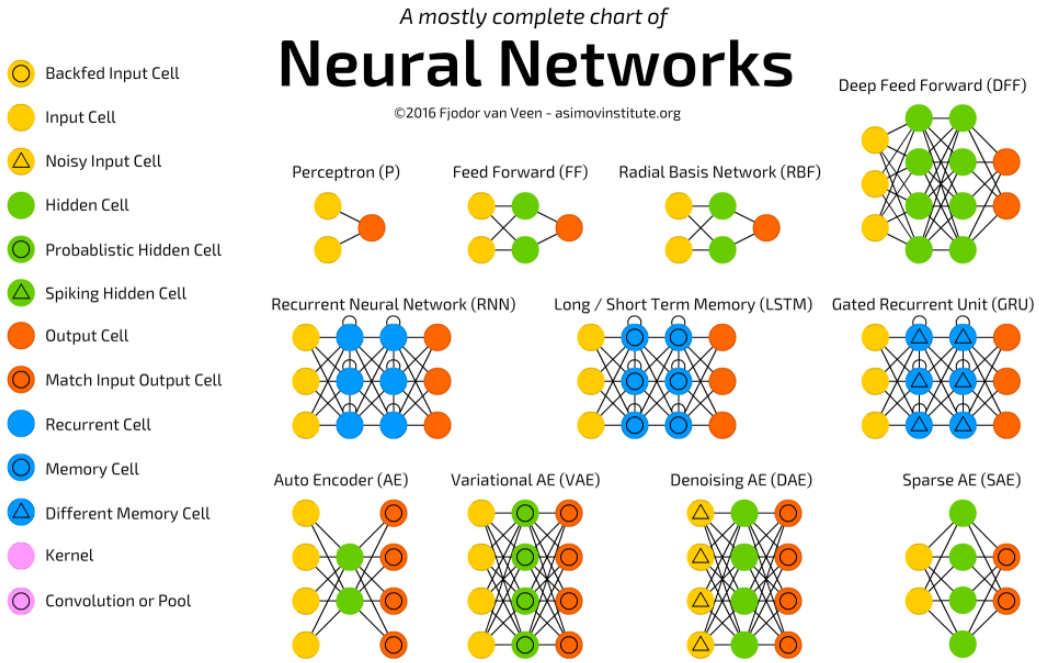


Figure 16 Some examples of neural networks[10]

Of these multiple types of neural networks, focus must be drawn to 2 types of neural networks that are currently being widely used in the field of machine learning, those are deep neural networks and convolutional neural networks, both of these can be combined with each other as well giving rise to a deep convolutional neural network

3.2.1 Deep Neural Networks

Deep learning employs deep neural network architecture such as convolutional neural networks or Boltzmann machines. The architecture is markedly different that ANNs because of the large increase in the number of layers of neurons between the input and the output, this in turn allows the neural network to progressively extract higher level features as the information travels further and further through the network. An example of the feature extraction can be seen in the figure below where a deep neural network is trained for image recognition and it extracts simple shapes like lines and the features extracted become more and more complex.

Deep neural networks are generally used in the field for applications such as image recognition, speech to text conversion and machine translation and recently they have also been used in the medical field for detection of tumors and also in board games where they have been shown to be more successful than humans[7]. However these neural networks face 2 main drawbacks, firstly the require a large amount of data for them to be able to be trained accurately and secondly they can suffer from overfitting of the data which can lead to false positives or wrong outputs.

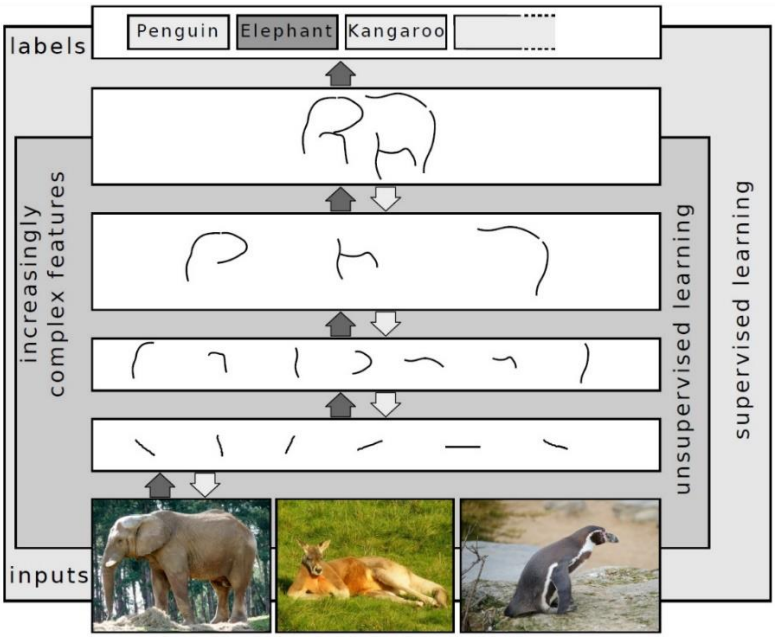


Figure 17 Feature extraction in a deep neural network[11]

3.2.2 Introduction to Convolutional Neural Networks

Convolutional neural networks (CNNs or Convnets) are a type of deep learning architecture that are widely used in classification problems owing to them using a set of unique layers called convolutional layers. These neural networks are also ideal for the problem statement of this project and may be used effectively in the classification of faults in a system. The basic structure of a CNN is almost the same as an ANN in that they have one input and output layer and they have multiple hidden layers in the middle, but some of the hidden layers are convolutional layers, the basic working principle behind the convolutional layer is that a filter is used when propagating information from one layer to the next, this filter is key to feature extraction as is seen in the figure 17. The full working of the CNN will be explained in the next section.

3.2.3 Convolutional Neural Networks

When explaining the working of CNNs it is easiest to do so in the realm of image recognition, but it must be mentioned that CNNs are not only applied for image recognition but also for various other applications. If an image needs to be given as the input to an ANN, the easiest way that this can be done is by flattening the data into a vector since the image can be represented as a matrix of various values, the values are just rearranged into a vector of dimensions Nx1 and that is used as the input layer, as shown in the figure below.

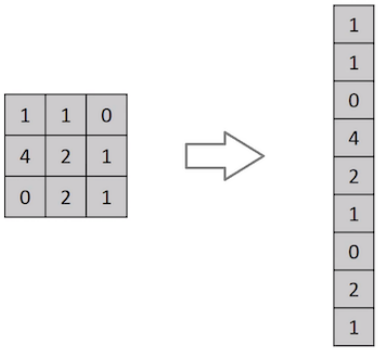


Figure 18 Flattening of a matrix

This method is useful for smaller images with binary values (black and white) but if larger images with higher resolution and multiple colour channels must be considered then this becomes highly inaccurate in its detection, this same principle holds true for noisy time series data that will be analysed in this project. So to overcome this limitation a convolutional layer is added which has 2 processes called filtering and pooling which are represented in the figure below and will be explained in the next section

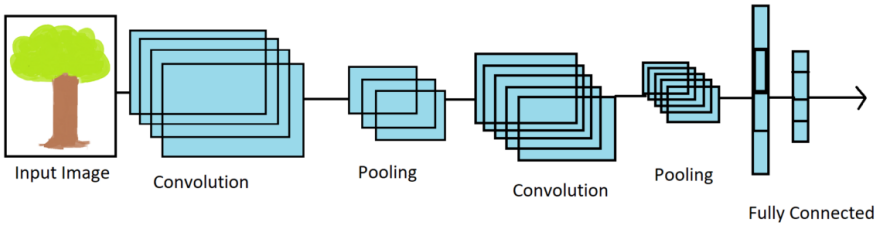


Figure 19 Diagrammatic representation of a convolutional layer[12]

3.2.3 Filters and Filtering

As previously explained filtering is a part of the convolution process in a CNN. Filters are not restricted in terms of the number of dimensions but they are usually with 2 or 3 dimensions and their dimensions are smaller than the input data that they are filtering, the figure below shows the process of filtering in a CNN.

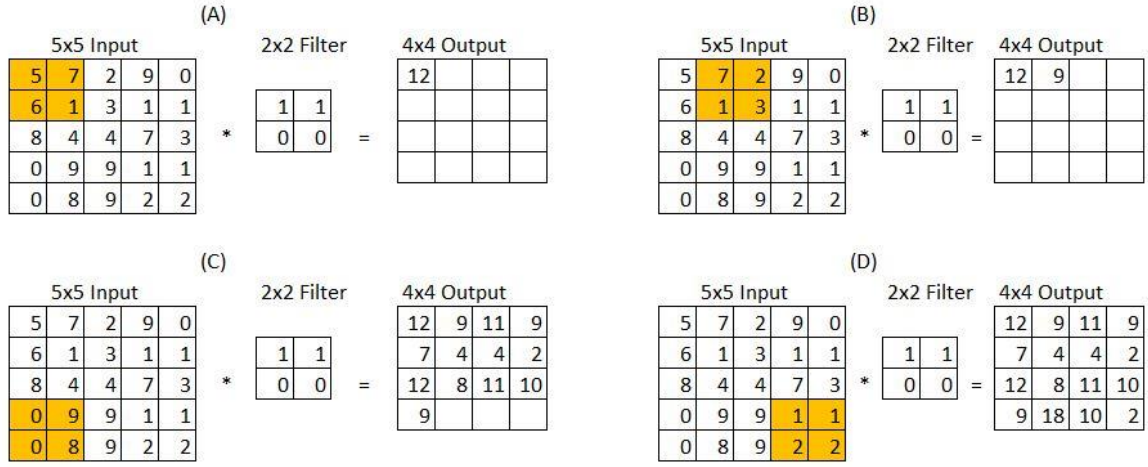


Figure 20 Working of filters in a CNN

In the above diagram the input matrix is a 5x5 matrix and it is being filtered with a 2x2 matrix which gives an output matrix of 4x4, the dimensions of the output matrix depends on the size of the input and the filter, the padding and the striding. As it can be seen in the above diagram the highlighted portion of the input matrix is the part that is multiplied with each element of the filter matrix element wise and then added together to get the output value, the calculations of the above diagram are shown below.

$$(A) \quad (1 * 5) + (1 * 7) + (0 * 6) + (0 * 1) = 12 \tag{20}$$

$$(B) \quad (1 * 7) + (1 * 2) + (0 * 1) + (0 * 3) = 9 \tag{21}$$

$$(C) \quad (1 * 0) + (1 * 9) + (0 * 0) + (0 * 8) = 9 \tag{22}$$

$$(D) \quad (1 * 1) + (1 * 1) + (0 * 2) + (0 * 2) = 2 \tag{23}$$

In figure 20 the filter is moving to the right by one element every time an output element is calculated, the amount of elements by which the filter moves over the input matrix to obtain each element of the output matrix is called striding, so in figure 20 the filter stride is 1. Moreover another thing to discuss is padding, which is augmenting the input matrix with an additional layer of zeros wrapped around the input matrix, by doing this it is ensuring that the data on the edges of the matrix is also captured well, and it will lead to the output matrix having higher dimensions than if there was no padding, the same matrix in fig 8 if padded will look like the figure below.

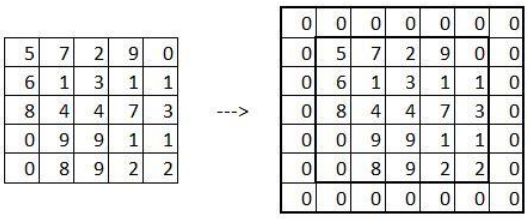


Figure 21 Padding of the input matrix

Striding is not only horizontal but also vertical, so the amount by which the filter moves horizontally and vertically needs to be defined, the figure below shows the filtering of the padded matrix with horizontal striding set at 2 and vertical string set at 1

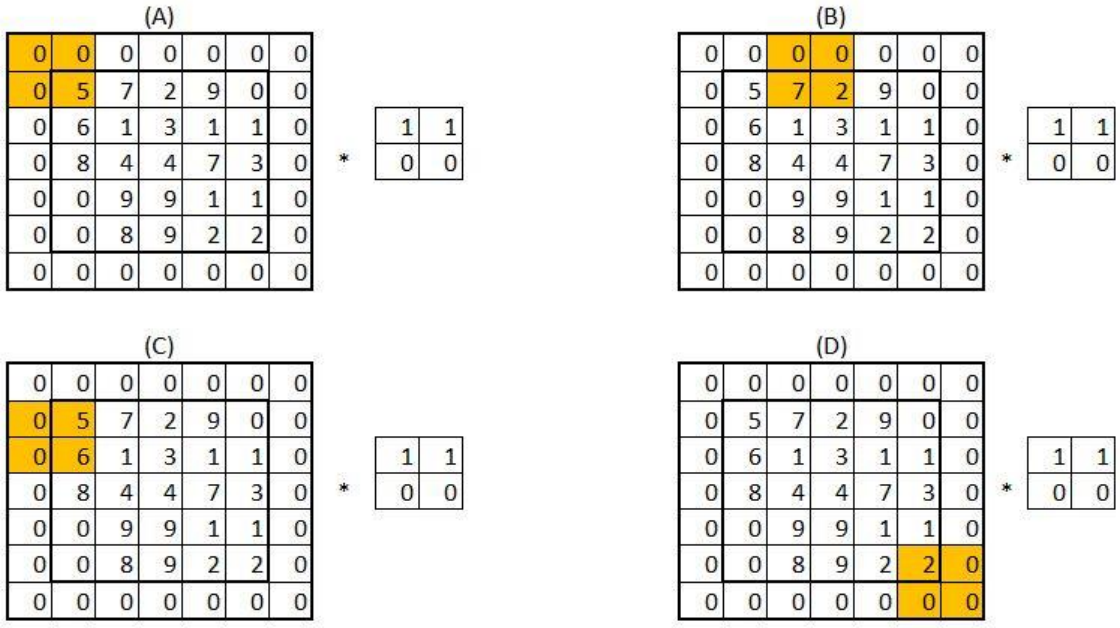


Figure 22 Padding and striding

From fig 10 it can be seen as the filter moves from (A) to (B) it moves over 2 elements and when it goes from (B) to (C) it moves by one element vertically, the striding can also be represented in vector form as [2 1] denoting [horizontal striding vertical striding]. The process shown is just shown for one node of a CNN, this process needs to be repeated for each node or neuron on the convolutional layer where each filter is different and might be responsible for extracting different features from the input matrix, like horizontal or vertical lines, and as the information traverses farther through the neural network higher level features might be extracted using different filters like parabolas or hyperbolas, the filters may be inputted by the programmers or they may be randomized initially and optimized by the computer using gradient descent. The dimensions of the output matrix are given by the following equations:

$$Out_x = \frac{in_x - f_x + 2 * P}{S} + 1 \quad (24)$$

$$Out_y = \frac{in_y - f_y + 2 * P}{S} + 1 \quad (25)$$

Where in_x and in_y are the x and y dimensions of the input matrix and f_x and f_y are the x and y dimensions of the filter and P and S are padding and striding respectively.

3.2.4 Pooling

As it was previously mentioned pooling is the next process that happens in a convolutional layer, it is usually done to reduce the size of the output of the layer and to reduce the computational power required to process the data. Pooling can also extract the more prominent or important features in the data and it also acts as a noise suppressant giving cleaner data to the next layer.

There are 2 main types of pooling max pooling and average pooling. Pooling involves selection of a window size and selection of the horizontal and vertical striding, as explained in the previous section, after selecting the type of pooling. Average pooling involves taking the average of the elements captured by the window and giving that as the output and max pooling involves finding the maximum number in the window and giving that as the output examples of both max and average pooling are shown in the figure below.

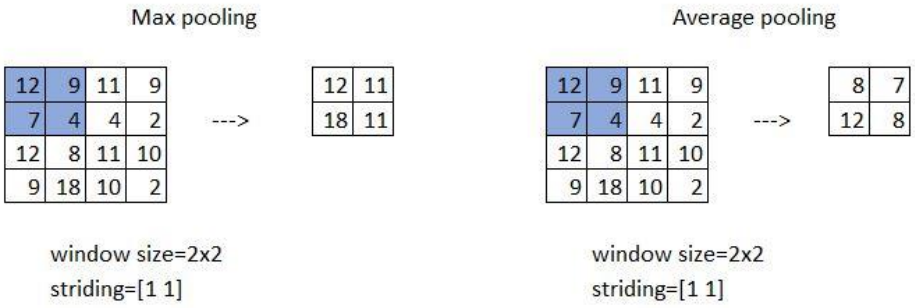


Figure 23 Illustration of max and average pooling

As can be seen from the above figure the outputs are clearly different but the dimensions of the output are the same, the dimensions of the output can be calculated by using the equations 24 and 25 as well. Also it must be mentioned that the noise suppression characteristic of pooling is better when max pooling is done because it discards the noisy activations (which are usually the smallest values), but average pooling also does noise reduction but it stems from the fact that it is reducing the dimensions of the output given to the next layer. Before the output of the CNN is given to the classification layers, the output is flattened to a vector, this vector is then the input layer of the next layer. So it can be said that when using a CNN for a classification problem, the neural network is essentially made up of 2 parts, the first part is the feature extraction which is done using the convolutional layers

and the second part is the classification which takes the extracted features and learns to classify them accurately.

3.2.5 Softmax Layer

Generally when training a neural network for classification, the final output layer of the network will have N number of neurons and N will also be the number of classification states available for the system. When the network is training, the activation of the output neurons will not be strictly binary, so some neurons may activate partially as well, this becomes a problem for the neural network.

So to overcome this problem a Softmax layer is added as the final layer, with the number of outputs of the Softmax layer being the number of system states needed to be classified. The operating function of the Softmax layer is defined as[6]:

$$y_i = \frac{e^i}{\sum_{i=1}^{N} e^i} \tag{26}$$

The above equations gives the probability of the i^{th} neuron being activated and if we calculate $\sum_{i=1}^{N} y_i$ it will be equal to 1. This last layer makes it much easier for the network to assign a value to the input that has been given to it to a particular output (classification of the system state) since the system is defined to only have one state at a time while training.

3.3 TRAINING OF NEURAL NETWORKS

In this section the training of neural networks is explained along with some important concepts such as gradient descent, cost functions and back propagation, these concepts are essential to understanding how the networks learn from the training data.

3.3.1 Gradient Descent and Cost Function

Consider a neural network with 4 outputs, whose weights and biases have been randomly initialized, for a particular set of input, the neural network gives the following output

$$y = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.2 \\ 0 \end{bmatrix} \tag{27}$$

But since the data is labelled the desired output is known and is given by the following

$$y_d = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{28}$$

Now the programmers need a way to tell the neural network that the output that it has given is wrong and also to tell it the magnitude by which it is wrong, this is where a cost function comes into the picture. A cost function is a way for the neural network to measure how wrong it is in its guess for the value of the output as a function of the weights and biases it currently has, a formula for the cost function is given as

$$C(w,b) = \frac{1}{2*N} * \sum_{i=1}^{N} (y_{di} - y_i)^2 \tag{29}$$

Here the cost function is a function of w and b which are the weights and the biases of the neural network and N is the total number of training samples used to train the network and y_{di} is the desired values for the output for the i^{th} sample and y_i is the actual values that the network gives for the i^{th} sample. The function will give the cost for each output and if the sum of the vector is taken the total cost for the neural network will be obtained.

Now the goal of the network is to minimise the cost and maximise the accuracy of the network. So the network is trained with batches of data and the weight matrices as shown in equations 11,12 and 13 are adjusted so as to reduce the total cost. The total cost with a new batch of data is compared to the total cost with the preceding batch of data and the difference between them can be defined as ϵ and the network is constantly trained until ϵ becomes small enough, if ϵ is increasing then the network is being badly trained, but care must be taken to not define ϵ to be too small as it runs the risk of ϵ never reaching the value.

$$\epsilon < C_{new} - C_{old} \tag{30}$$

This process of changing the weights and biases to reduce the cost function is known as optimization, where many different algorithms like stochastic gradient descent or adam can be used[6]. The algorithm is adjusting the weights and biases so the network minimises the prediction error by stepping in the direction of the steepest gradient descent. Another thing to note is that since the network is not being given all of the data at once and only in batches, it is not calculating the exact optimum gradient descent but only for that particular batch of data, so the descent is not following the optimum path, this is done to save computing power and is also known as stochastic gradient descent. The next section explores how the network actually adjusts it's weights and biases according to the cost function based on a concept known as backpropagation.

3.3.2 Backpropagation

Back propagation is the way the neural network learns from its mistakes during the training phase. Backpropagation involves using the error calculated by the cost function and using gradient descent to slowly minimise that error by changing the weights and biases of the layers preceding the output layer. It is called back propagation because it starts from the last layer or the output layer where the cost is calculated using the cost function and the error is distributed backwards layer by layer applying the same processes for each layer.

Considering the example in the previous section again, the example neural network is defined to have 2 hidden layers and the output layer has 4 outputs with the hidden layers having n nodes each. It can be seen that the 3rd output neuron must be fully activated for the output to be correct. So the 3rd neuron is considered

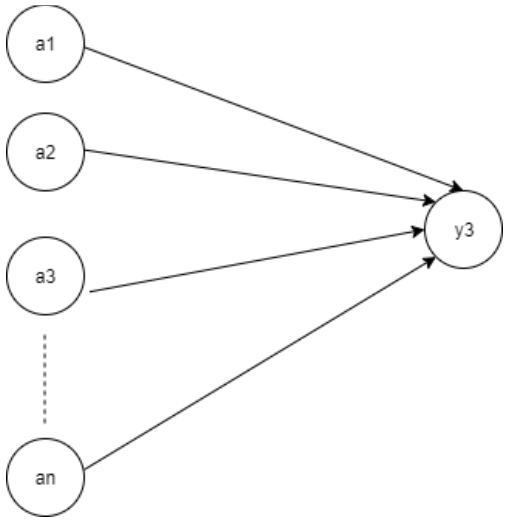


Figure 24 Single neuron connected to previous layer

So according to equation 5 the activation of the output neuron y3 is affected by the activation of the neurons in the previous layer, the weights and the biases. Since the activation of the neurons is something that cannot directly be affected the weights and biases are changed to adjust it to the required output, this is done for the other layers in the output layer as well. After this is done for the output layer and the layer preceding it, the same method is applied to the next pair of layers again and again until the input layer is reached, hence the name backpropagation.

4. FAULTS AND THEIR EFFECTS

4.1 RECTIFIER FAULT ANALYSIS

In this chapter, the various faults that can affect the different parts of the rectifier are examined, the faults and how they are implemented in the simulation and their effects on the measurements will be discussed in the next chapter. Faults need to be introduced in various parts of the setup and the AC current and DC voltage is measured for each of the faults introduced. In the controlled rectifier, each component is examined closely to see what kind of fault can be introduced in it and what effect it has on the system measurements. The schematic of the single phase rectifier can be seen in the diagram below:

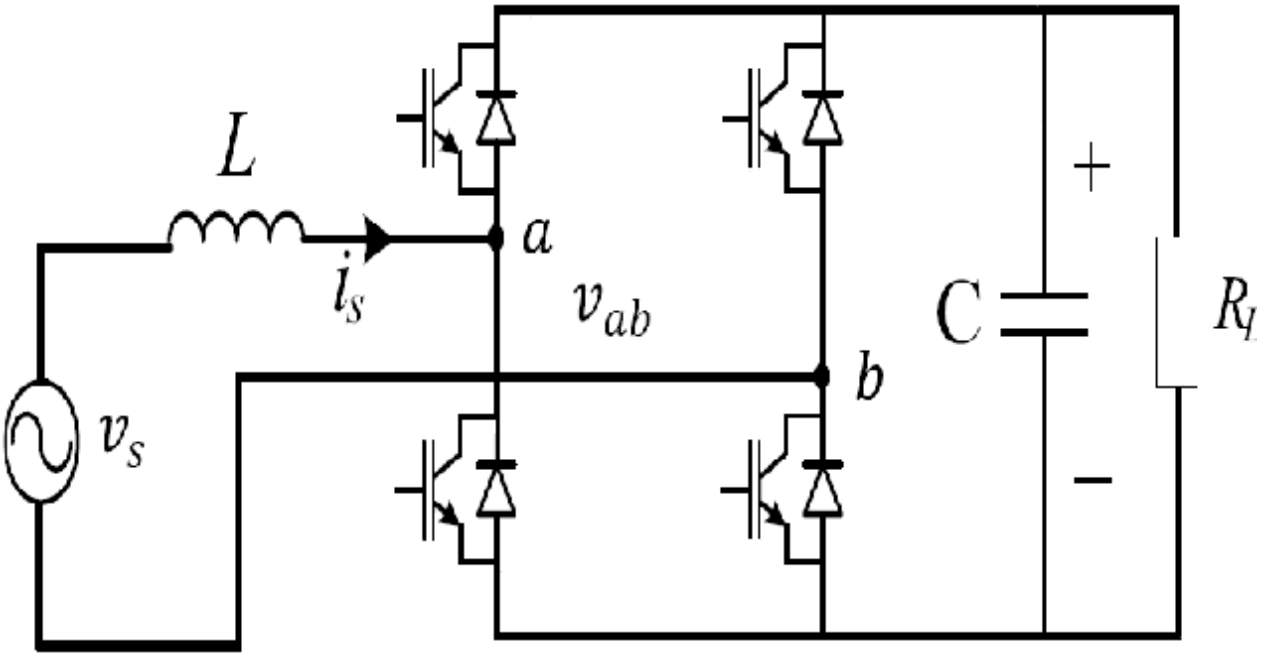


Figure 25 Diagram of controlled rectifier

From the diagram it can be said that the main components in which faults can occur are:

-Power Switches

-Diodes

-Inductor

-Capacitor

Each of these components will be examined to see what kinds of faults most commonly affect them and how the system changes according to these faults. Now another thing to keep in mind is that there can be faults that do not directly affect the components as well such as a DC offset in the supply voltage or a change in the control algorithm, these are defined as external faults and will also be discussed further in this chapter.

4.1.1 POWER SWITCH FAULTS

As mentioned in the introduction power switches are the most likely component to fail in any power electronic as such it is important to analyse the various faults and their effects through the power switches that are used in the rectifier. Generally faults in power switches can be classified into either open circuit or short circuit faults. Open circuit faults can happen due to improper connections, loose wire, aging of wires or failure of the driver for the switches, in the case of open circuit faults the rectifier can still work but not as it is designed to, without adverse effects on the system as a whole, but it can lead to excess stress on the healthy switches in the system. However in the case of a short circuit fault which is caused when the current doesn't take a path through the switch, flowing freely, this can lead to high currents in the system, adversely affecting other components and leading to propagation of faults into other components of the system[8].

Some of the faults in a power switches along with their effects are shown in the table below[8][9]:

Fault	Effects and Features
Overheating	Maximum operating temperature is exceeded causing relatively low collector current
Overcurrent	Maximum operating current is exceeded causing high currents at the DC link, peaks in the current can also be caused by incorrect switching
Short Circuit	Very high rate of change of current in the collector causing high current at the DC link and desaturation of the transistor
Open Circuit	Usually caused by loose connections or improper switching which leads to improper load current placing stress on other switches
Earth Fault	Collector current is dependent on earth inductance and driving voltage desaturation of the transistor is dependent on fault current value

Table 4 Table of faults affecting power switches

The effect that some of these faults have on the measured quantities, namely AC current and DC voltage will be examined more closely in a later chapter concerning data collection.

4.1.2 DIODE FAULTS

The diode is also a component which is likely to fail, however diodes are lesser likely to fail than power switches because they are not an active component (no switching required) so there is one less way that the use of a diode can go wrong. In the case of the controlled rectifier if one diode fails it will affect the system as a whole and place more stress on the other diodes in the system, however if the currents are below the maximum rated current it shouldn't propagate the fault to the other diodes, but both the DC voltage and the AC current will be noticeably affected. Again as is the case with power switches, diode faults can also mainly be classified into open circuit and short circuit faults. When an open circuit fault happens on a diode the resistance of that diode essentially becomes infinity and doesn't allow any current to pass through and this is usually caused by loose connections in the system and with short circuit fault, the resistance of the diode essentially becomes zero in both directions allowing current to also pass in both directions, this is more dangerous as it can lead to high currents in the system and usually happens when the diode has been exposed to voltages or currents higher than their rated values.

4.1.3 CAPACITOR FAULTS

A capacitor in a circuit is usually never just a pure capacitance, and it can be represented as a capacitor in series with an inductor and a resistance, so the equivalent circuit of a capacitor looks like as shown in the diagram below

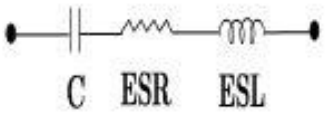


Figure 26 Equivalent model of capacitor

In the above diagram ESR is the equivalent series resistance and ESL is the equivaled series inductance, the values of the ESR and ESL depend mainly on the temperature and other physical and environmental conditions[13]. A change in these values due to a change in the environmental conditions can place stress on the capacitor and lead to changes in the output voltage and the amount of ripple that is present and can also lead to instability in the control of the controlled rectifier.

4.1.4 INDUCTOR FAULTS

The inductor in the controlled rectifier is used to store energy, to be released at particular intervals in order to boost the voltage as explained in chapter 2. So it can be said that the inductor is an integral part of the boosting action of the rectifier. The inductor stores the energy in a magnetic field around it, so anything that disrupts that magnetic field can be viewed as causing a fault in the inductor as it prevents it from working as intended, such as poor wiring, poor isolation and also solar flares, however one of the main ways it could be disrupted is if another inductor is placed nearby while the rectifier is operating as it would transfer some of its energy to that inductor leading to a loss in energy, besides this a change in the environmental conditions of the inductor such as temperature or humidity can lead to a change in the value of the inductor which can change the input AC current noticeably.

4.1.5 OTHER EXTERNAL FAULTS

There are some other faults that can occur, such as high input current or a DC offset in the input voltage, these faults are defined as external faults since they do not arise from a particular faulty component.

The following is a table of the faults chosen to implement in the simulation:

No.	Component	Fault
1	Power Switches	Short circuit
2	Power Switches	Open circuit
3	Power Switches	Overheating
4	Diodes	Short circuit
5	Diodes	Open circuit
6	Diodes	Overheating
7	Capacitor	Change in ESR
8	Capacitor	Open circuit
9	External	DC offset in input voltage
10	External	Short circuit load
11	External	Open circuit load

Table 5 Table of faults chosen to implement

4.2 EFFECTS OF FAULTS

The faults shown in the table above need to be implemented in the simulation, and their effect on the measured current and voltage is observed, but before examining the faulty systems, the healthy system must be observed to see how the faults affect the measurements. All of the faults were implemented such that they occur at 1 second in the simulation time.

4.2.1 HEALTHY SYSTEM

A Healthy system state needs to be defined as a state of the system when no faults are occurring and the system is operating as intended. When no faults have occurred, the system is set to operate at an output voltage of 4 V and an input current of 0.5 A and an input voltage of 4V at 50 Hz. The voltage and current measurements of the healthy system can be seen in the following figures.

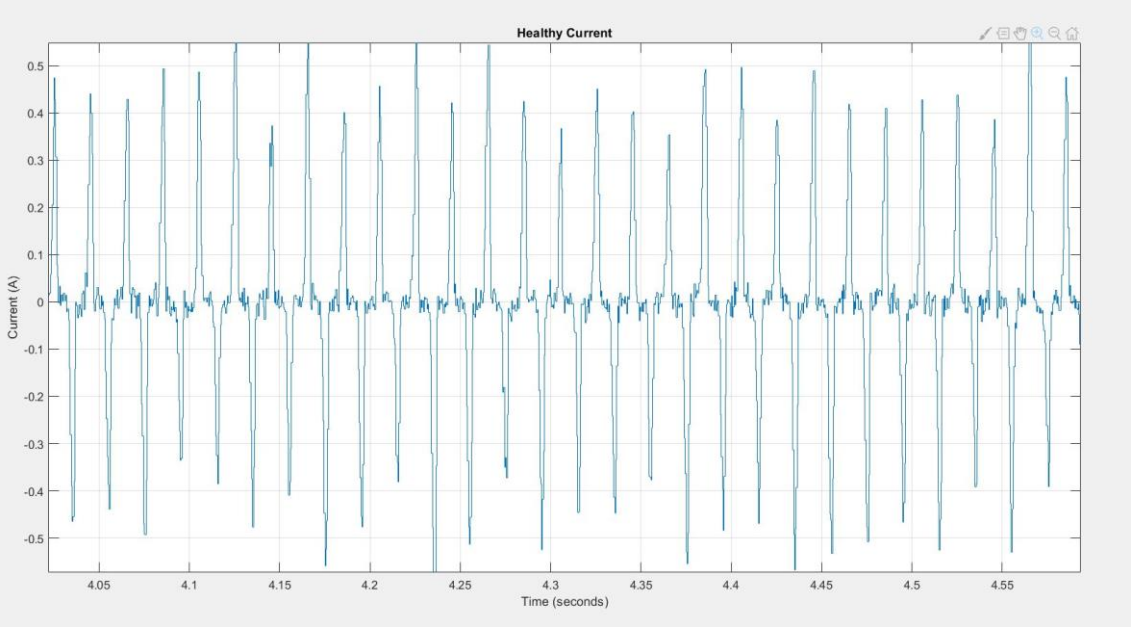


Figure 27 Healthy Current

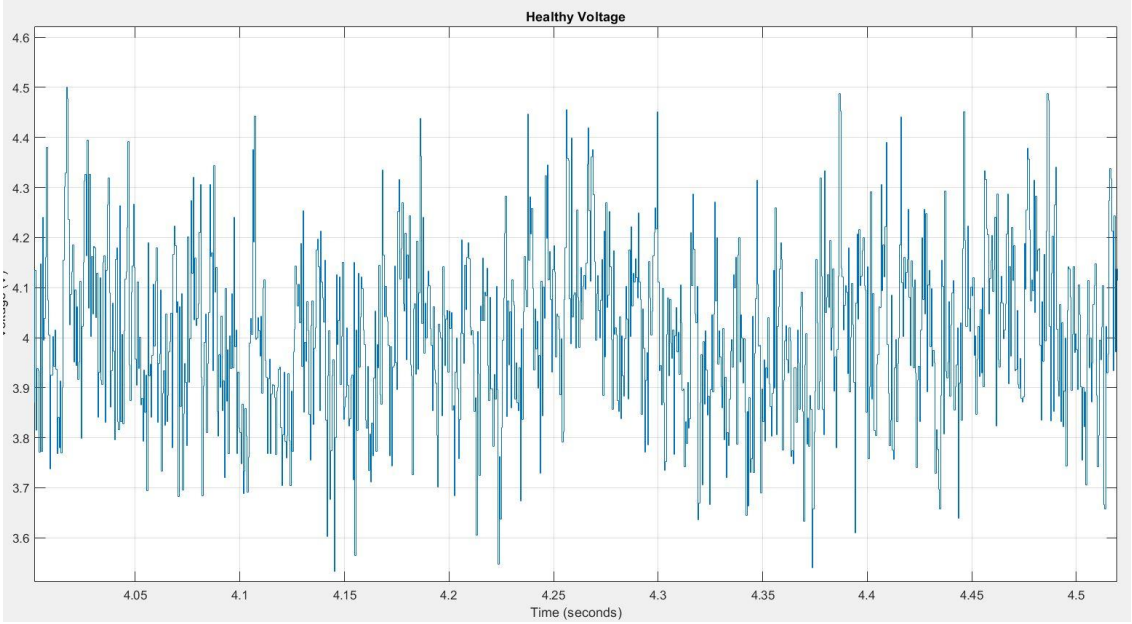


Figure 28 Healthy Voltage

The voltage in the system has been controlled to 4V DC and the current has been controlled to be in phase with the input voltage. This data sets the standard for a healthy system for the fault detection algorithm.

4.2.2 IGBT FAULTS

There were 3 types of faults chosen to simulate on the IGBT due to their ease of simulation as compared to the other faults, which are:

a. Short Circuit
b. Open Circuit
c. Overheating

SHORT CIRCUIT FAULT

This fault is simulated by removing an IGBT in the system and just short circuiting the connection, the rectifier after implementing the fault after 1 second is shown below:

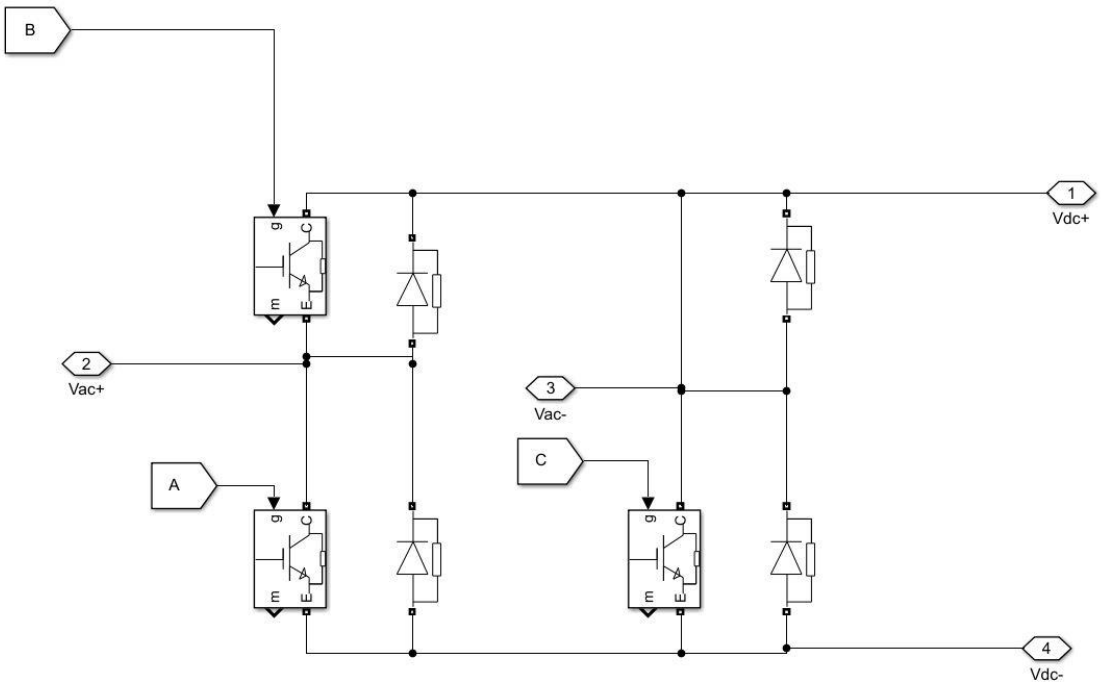


Figure 29 Rectifier with short circuit IGBT

This fault severely impacts the system, making it unable to deliver the voltage required and also delivering high currents in the input side making it dangerous to operate in this condition. The effect of this fault on the measurements can be in the following figures.

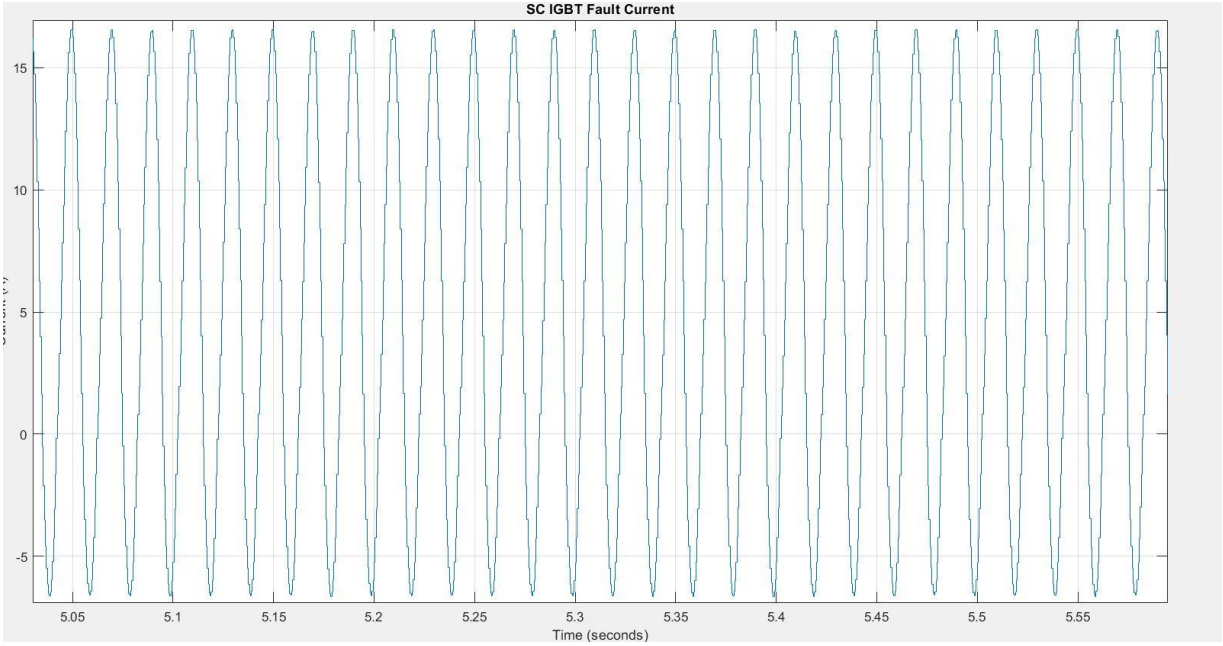


Figure 30 IGBT Short Circuit Current

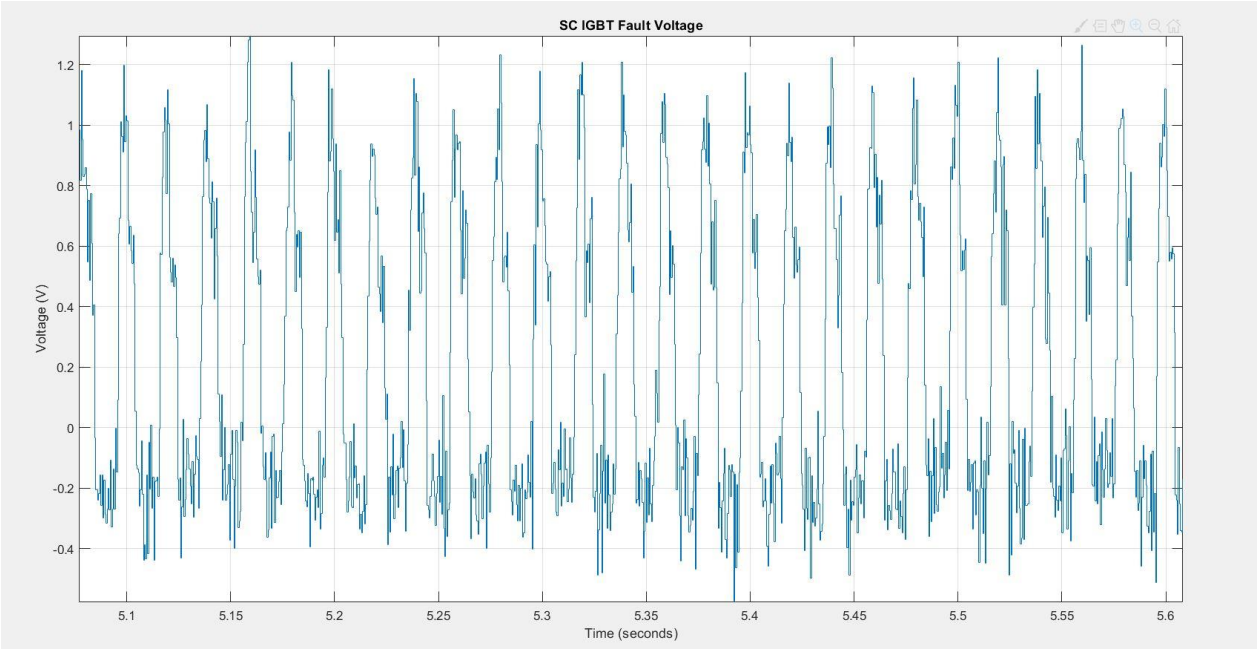


Figure 31 IGBT SC Fault Voltage

From the measurements, it can be seen that the short circuit allows high current to pass through the inductor and also the lack of an IGBT means that the waveforms will not be in phase. Moreover the voltage is around 1.1V, the short circuit IGBT is preventing the boost action of the rectifier and hence the voltage is at a low level.

OPEN CIRCUIT FAULT

In this fault the IGBT is removed from the rectifier and the connection is opened in the simulation at 1 second, the rectifier then looks as follows from the input side.

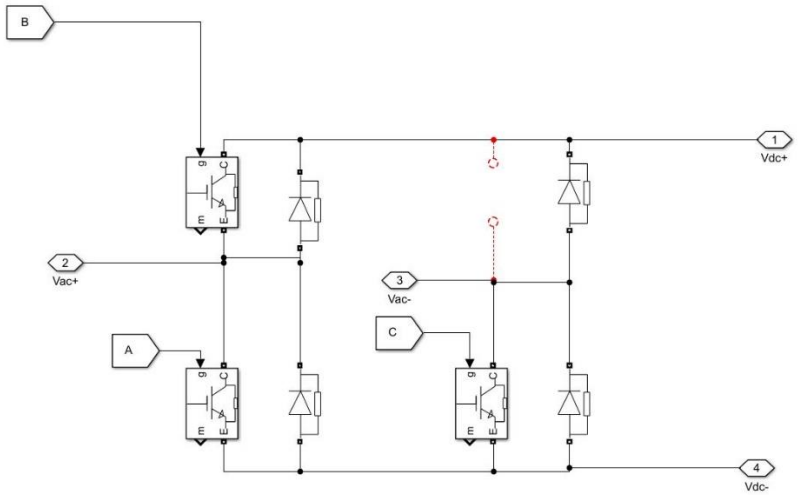


Figure 32 System with Open Circuit IGBT

When there is a fault like this, the boost action might be affected, with the other switches taking more stress. However, looking at the measurements, the voltage is more or less controlled to what is required, but the current waveform is significantly different.

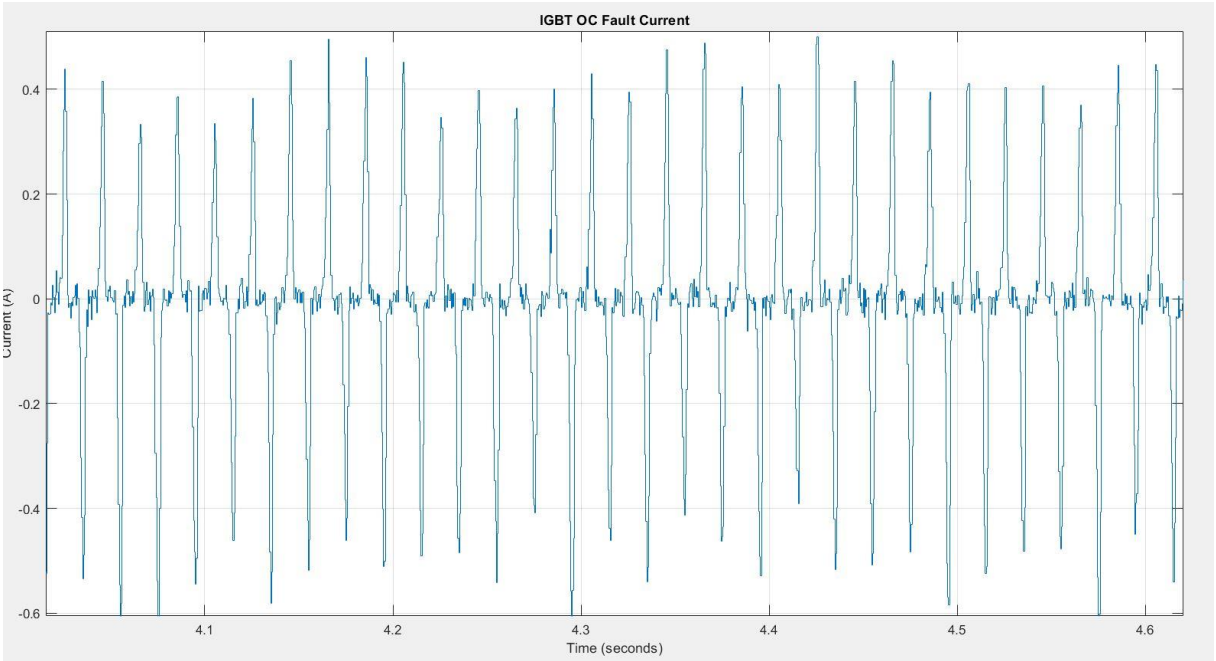


Figure 33 IGBT Open Circuit Current

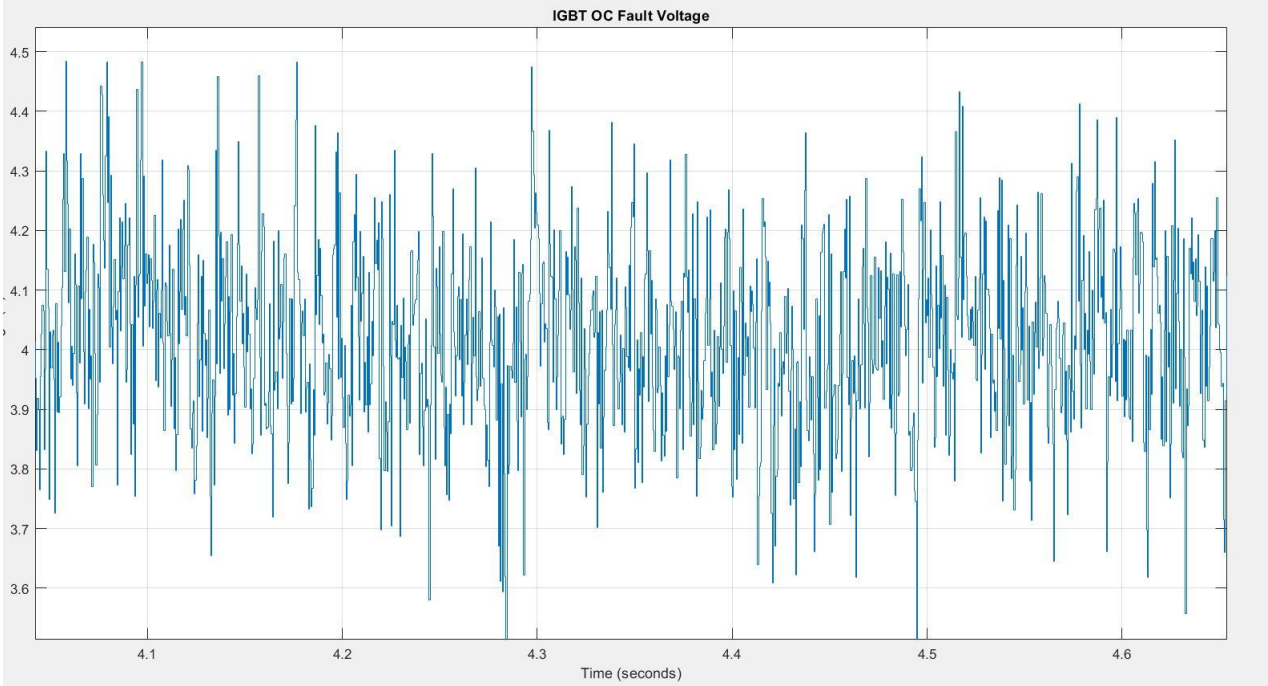


Figure 34 IGBT Open Circuit Voltage

OVERHEATING FAULT

Overheating of an IGBT will affect the internal characteristics of the switch, so this will usually effect things like resistance, forward voltage and power dissipation. To see what effect heat has on the IGBT the model of the IGBT in the simulation must be examined. The Simulink model of the IGBT is shown in the figure below.

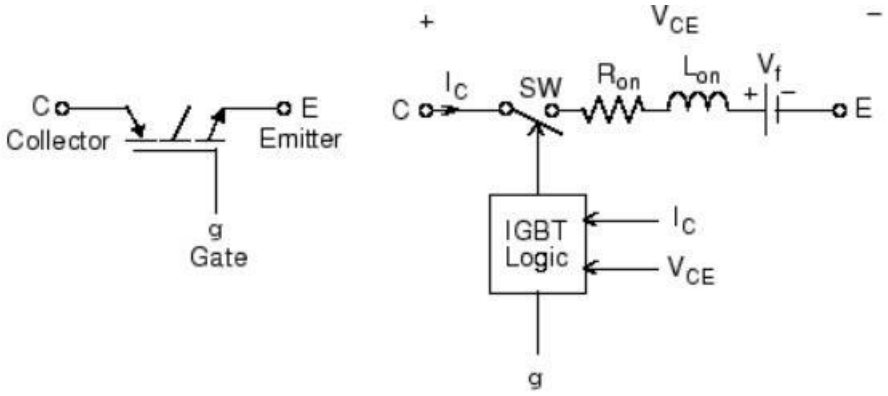


Figure 35 Internal model of IGBT in Simulink

From the internal model of the IGBT, it can be seen that the main things that the overheating will effect is R_{On} and V_f which is the on resistance and the forward voltage respectively, according to the datasheet of the IGBT IRGB6B60KPbF, from the following figure the on resistance and the forward voltage at room temperature.

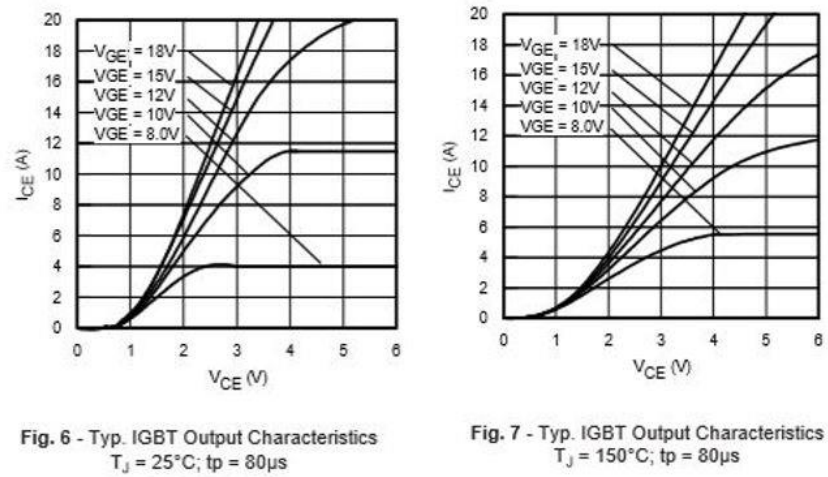


Fig. 6 - Typ. IGBT Output Characteristics
T_J = 25°C; tp = 80µs

Fig. 7 - Typ. IGBT Output Characteristics
T_J = 150°C; tp = 80µs

Figure 36 IGBT Characteristics at different temperatures[14]

From the above graphs the forward voltage at room temperature is approximately 0.8V and the resistance is 0.25 Ohm. The overheated IGBT at 150 Degrees has a forward voltage of approximately 0.4 V and the resistance is calculated to be 0.5 Ohm[14]. These changes were implemented in the simulation and the collected data is shown below.

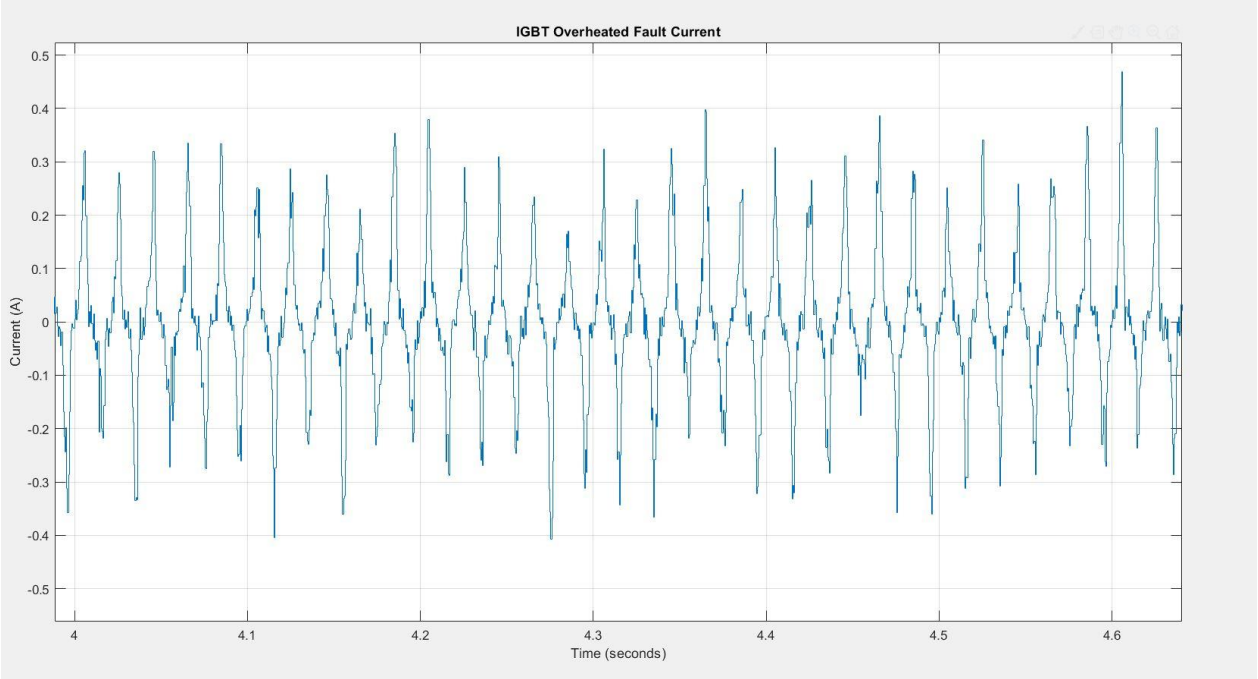


Figure 37 IGBT Overheated Fault Current

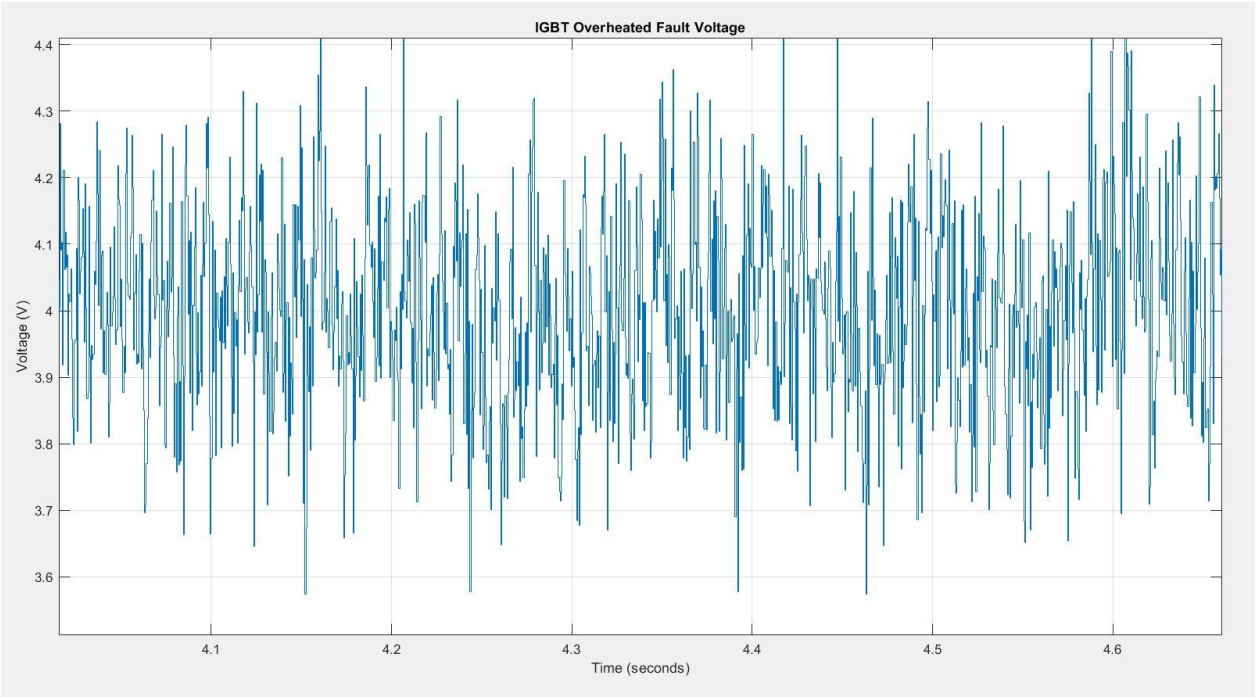


Figure 38 IGBT Overheated Fault Voltage

forward voltage The fault doesn't seem to have adversely affected the voltage of the rectifier however the waveform of the current has a lower amplitude, presumably because of the increase in the resistance due to the heat.

4.2.3 DIODE FAULTS

The faults for the diode are very similar to the faults that occur for the IGBT, but these faults effect the measurements differently than with the IGBT. The faults introduced for the diode are as follows:

- Short Circuit
- Open Circuit
- Overheating

SHORT CIRCUIT FAULT

To simulate this fault a diode was removed from the rectifier and a normal connection was made in its place, just as with the IGBT short circuit fault. The current in the AC side is very high and uncontrolled and the voltage is also not being controlled to the required voltage, this is because the short circuit in the rectifier is allowing current to pass through when it is not supposed to.

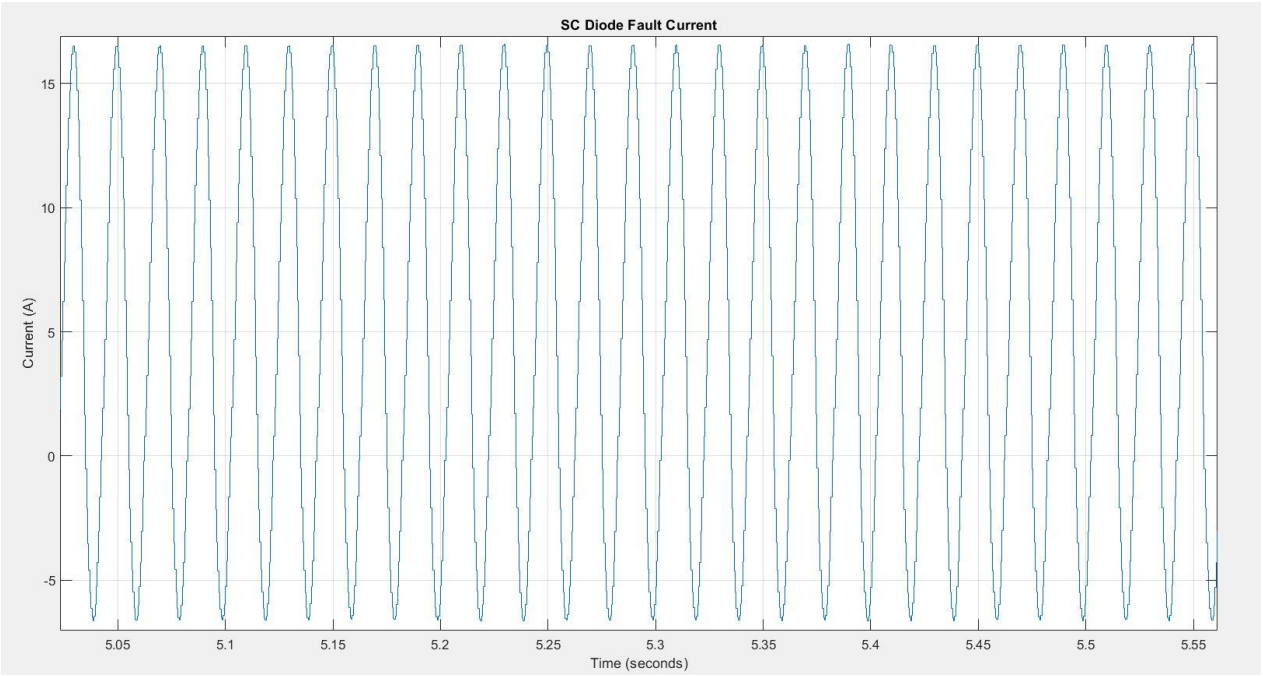


Figure 39 Diode short circuit current

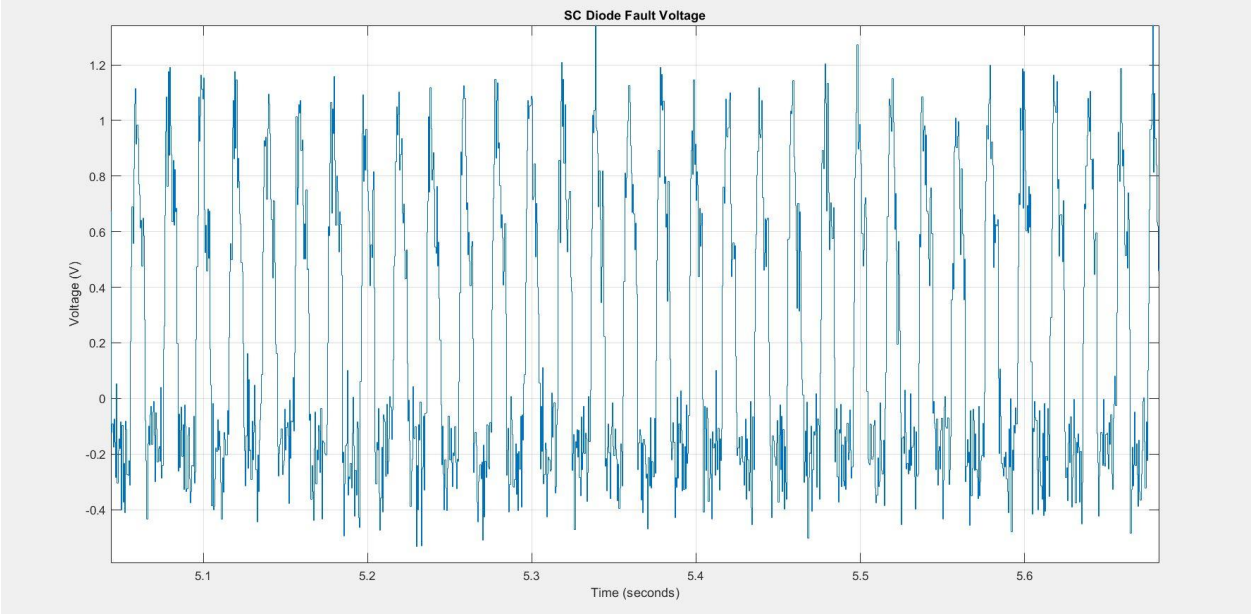


Figure 40 Diode short circuit voltage

OPEN CIRCUIT FAULT

To simulate the open circuit fault the diode was removed and the connection was left open. Understandably this would affect output voltage, since there is one diode effectively absent in the rectifier. This kind of fault also places more stress on the rest of the components in the rectifier which may lead to more faults in those components at a later time.

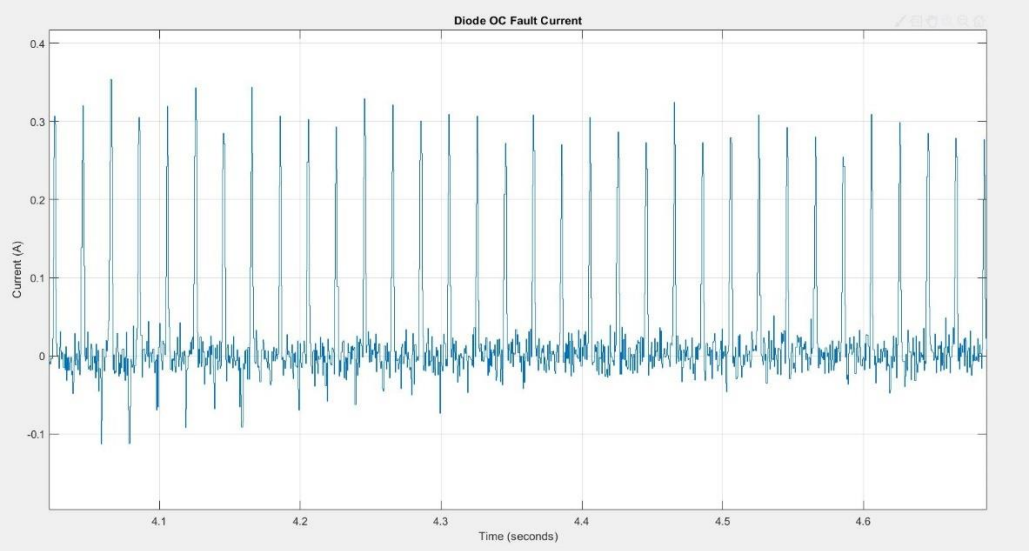


Figure 41 Diode open circuit current

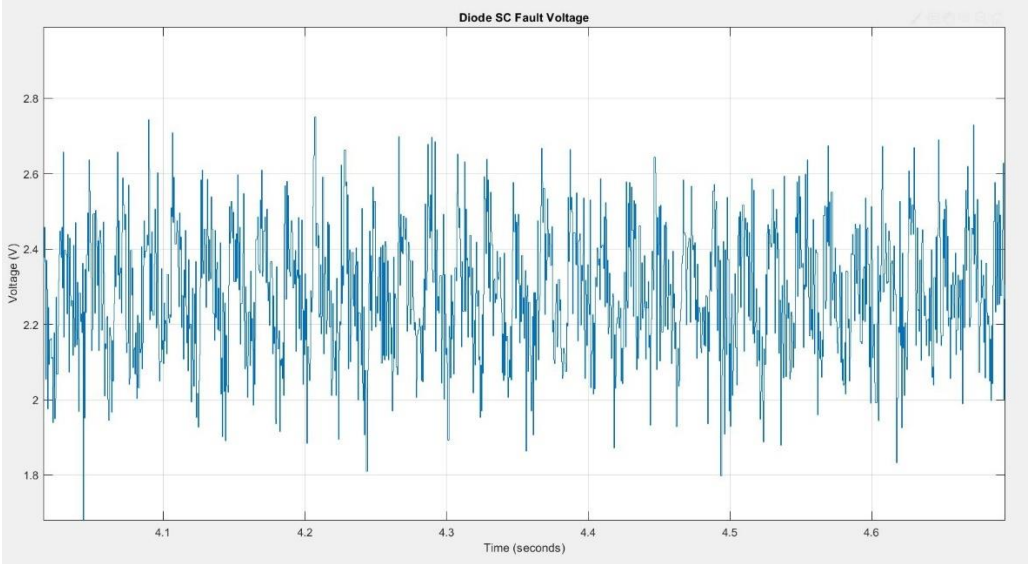


Figure 42 Diode open circuit voltage

OVERHEATING FAULT

When a diode overheats, the temperature effects a lot of its characteristics, which in turn would change the measured voltage and current, to implement this fault in the simulation, first the Simulink model of the diode must be examined.

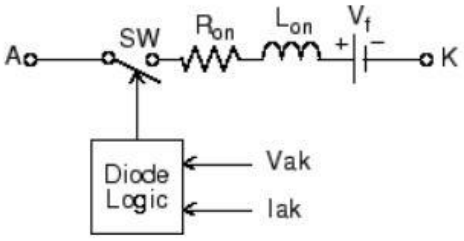
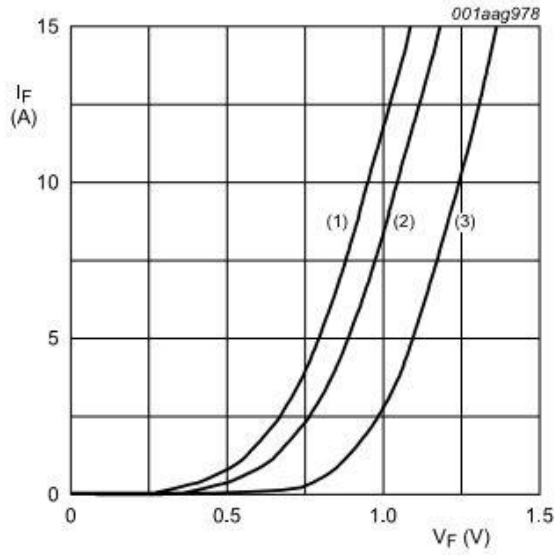


Figure 43 Model of a diode in simulink

In the diagram, it can be seen that there are 3 physical components in the model of the diode, the inductance can be ignored since it's value is so small it can be considered negligible. So the 2 other elements than can be effected are the on resistance and the forward voltage. The effect of temperature on both of these can be seen in the below graph from the datasheet of the diode BYQ28E-200E.



(1) T_j = 150 °C; typical values
(2) T_j = 150 °C; maximum values
(3) T_j = 25 °C; maximum values

Figure 44 Output characteristics of a diode[15]

At room temperature the forward voltage of the diode is approximately 0.8V and its on resistance is 0.033 Ohms, and if the temperature of the junction reaches 150 Celsius then the value of the forward voltage and on resistance changes to 0.4V and 0.05 Ohms[15], this change was implemented in the model to simulate the fault and the measurements were taken, where it can be seen that the amplitude of the current has dropped but the voltage looks unchanged.

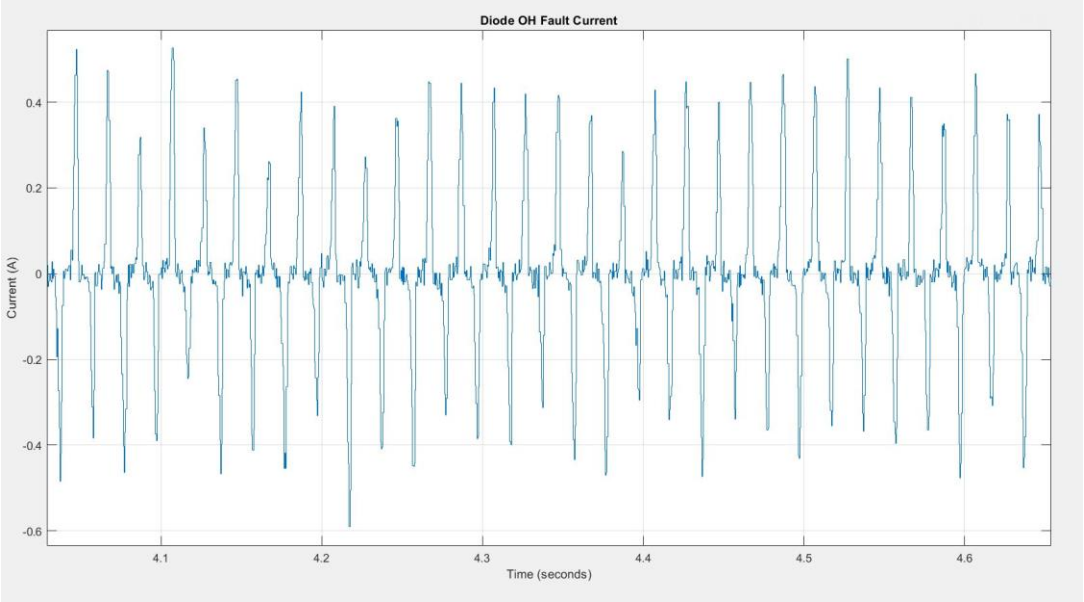


Figure 45 Diode overheating current

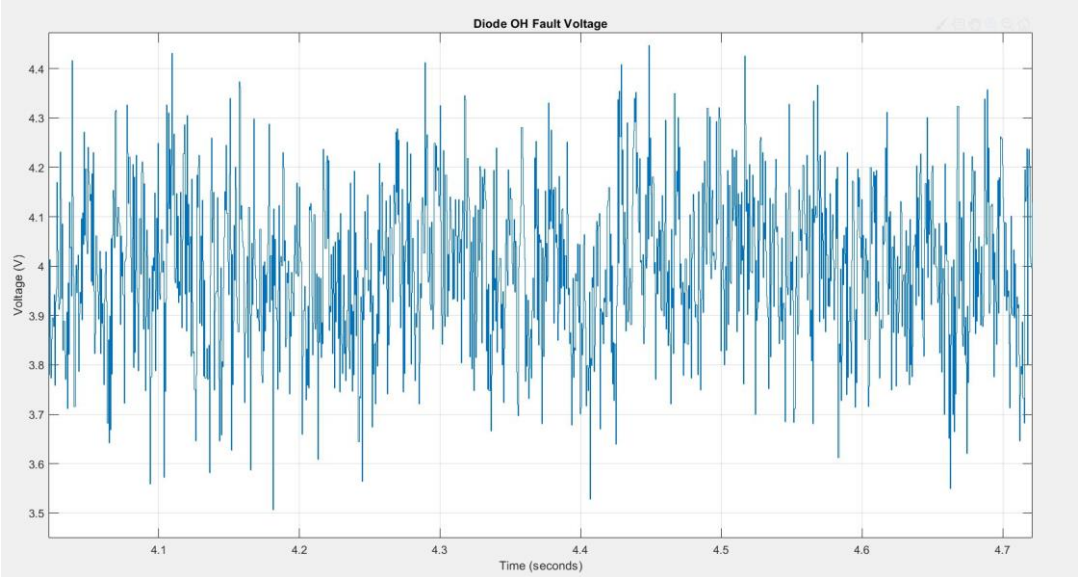


Figure 46 Diode overheating voltage

4.2.4 CAPACITOR FAULTS

The capacitor is instrumental in maintaining the output voltage to be smooth and steady, the main 2 faults chosen to simulate on the capacitor are:

- Open circuit capacitor
- Change in ESR due to heat

Both of these fault have effects on not just the output voltage but also the input current of the system.

OPEN CIRCUIT CAPACITOR

Sometimes due to a loose connection or harsh environment conditions, the capacitor may not be properly connected before the load, this can lead to an open circuit fault. In the simulation the capacitor was removed and the connections were left open. This can lead to no smoothing of voltage creating large variations in the DC voltage.

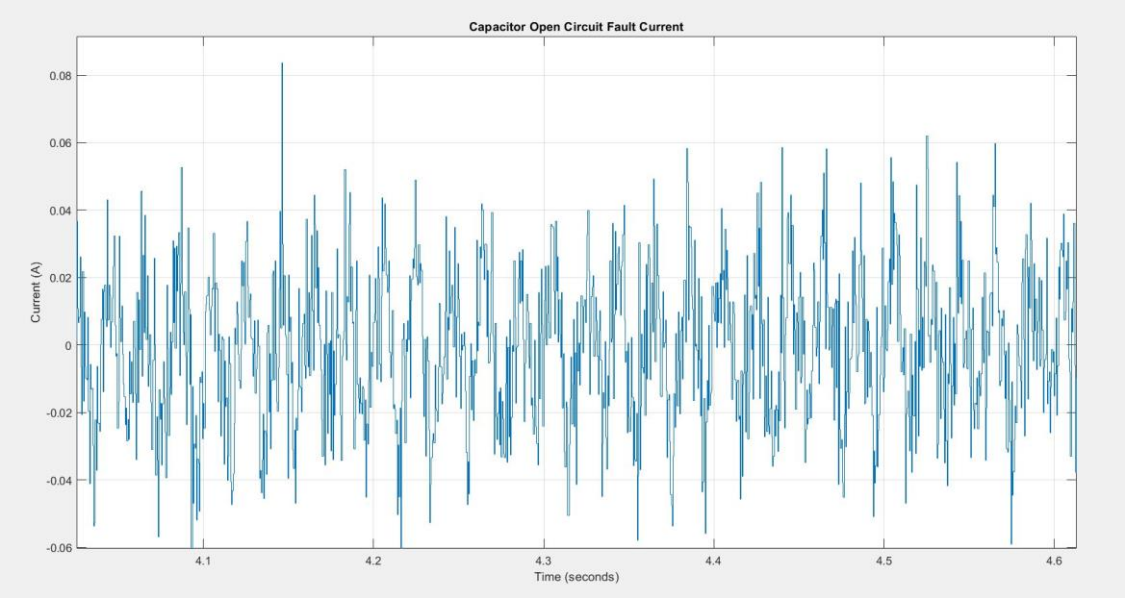


Figure 47 Capacitor open circuit current

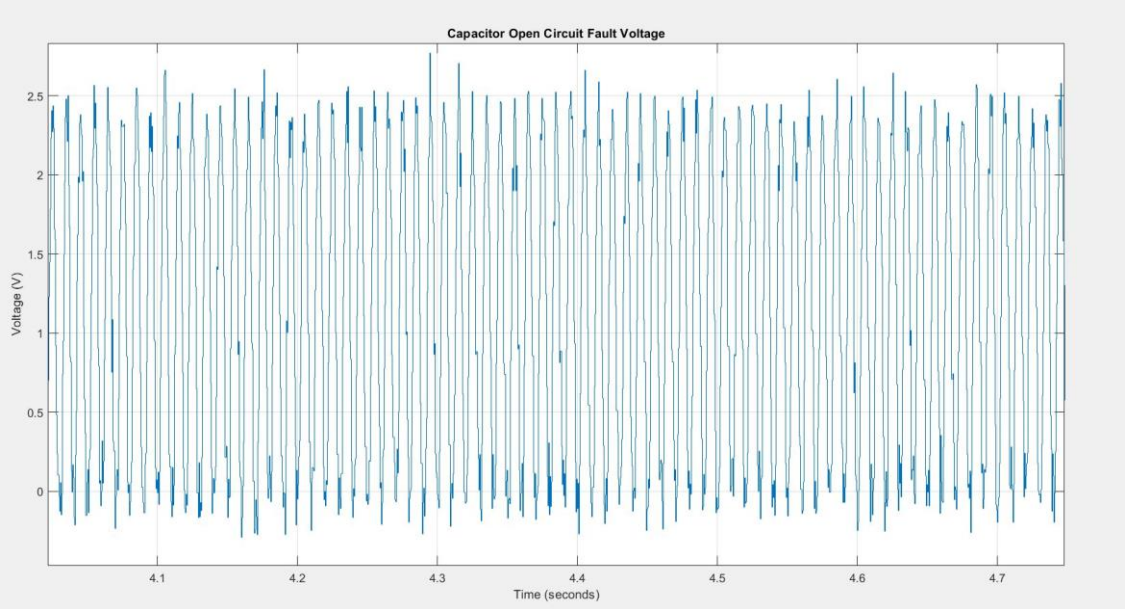


Figure 48 Capacitor open circuit voltage

CHANGE IN ESR OF CAPACITOR

A capacitor is usually never just a pure capacitance value, it will always have resistance and inductance values, albeit very small in magnitude. So a capacitance is usually modelled as a large capacitance value with small resistance and inductance values, however the inductance values are usually so small that their effect can be neglected. The capacitance used in the simulation is simulated as shown in the figure below.

Figure 49 Model of capacitance

As it was with the other components overheating, the ESR of the capacitor also increases, changing various characteristics about it such as leakage current, impedance and power dissipation.

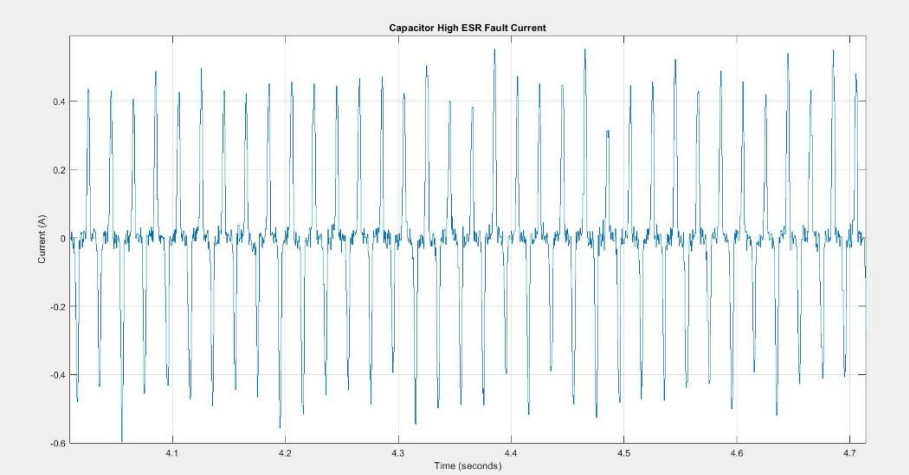


Figure 50 Capacitor high ESR current

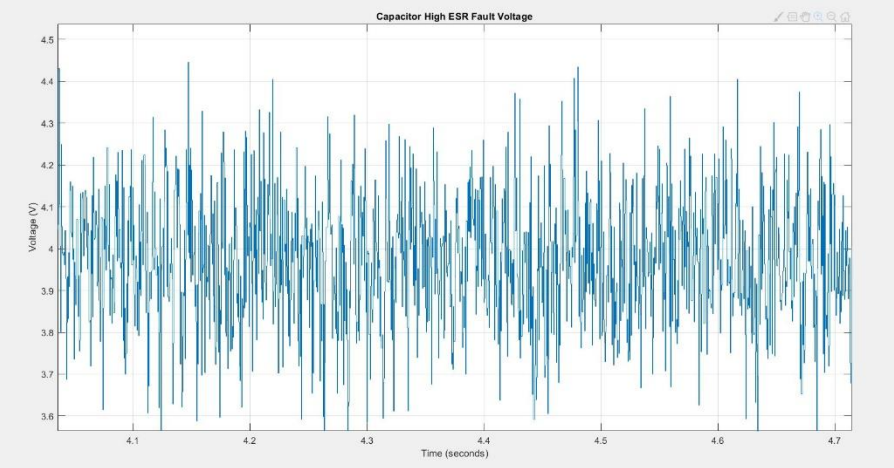


Figure 51 Capacitor high ESR voltage

4.2.5 EXTERNAL FAULTS

The external faults are those that do not affect any particular component but the system as a whole or the load, which is not a part of the rectifier. The external faults introduced are as follows:

- DC offset in input voltage
- Open circuit load
- Short circuit load

These will affect the current and the voltage measurements.

DC OFFSET IN INPUT VOLTAGE

In this fault, it is assumed that the power supply feeding the rectifier is faulty which leads to a DC offset voltage, the measurements with this fault can be seen in the figures below.

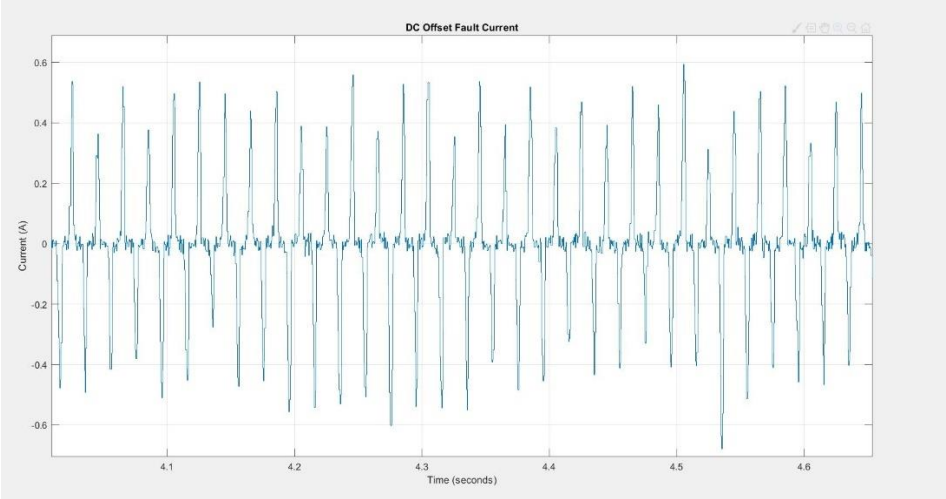


Figure 52 DC offset fault current

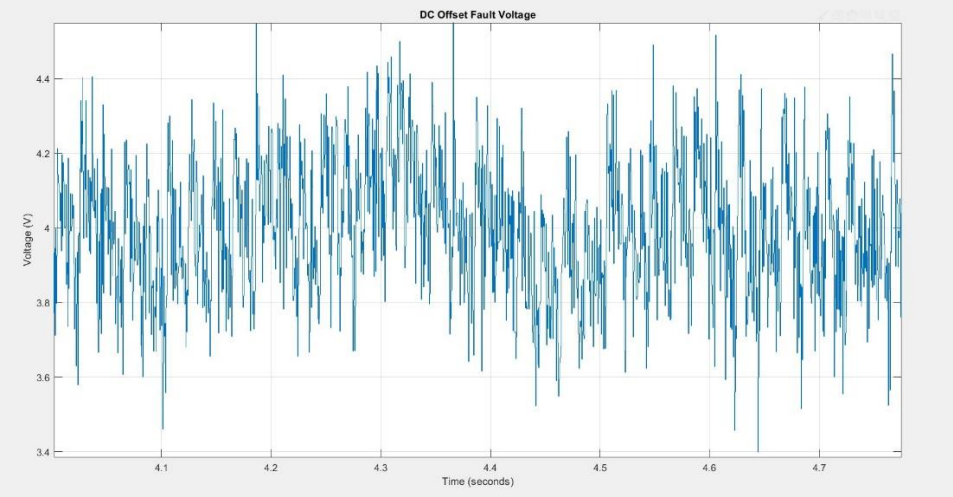


Figure 53 DC offset fault voltage

OPEN CIRCUIT LOAD

An open circuit at the load side can happen, leading to essentially no load connected to the rectifier, this leads to the voltage being achieved but the input current being too low.

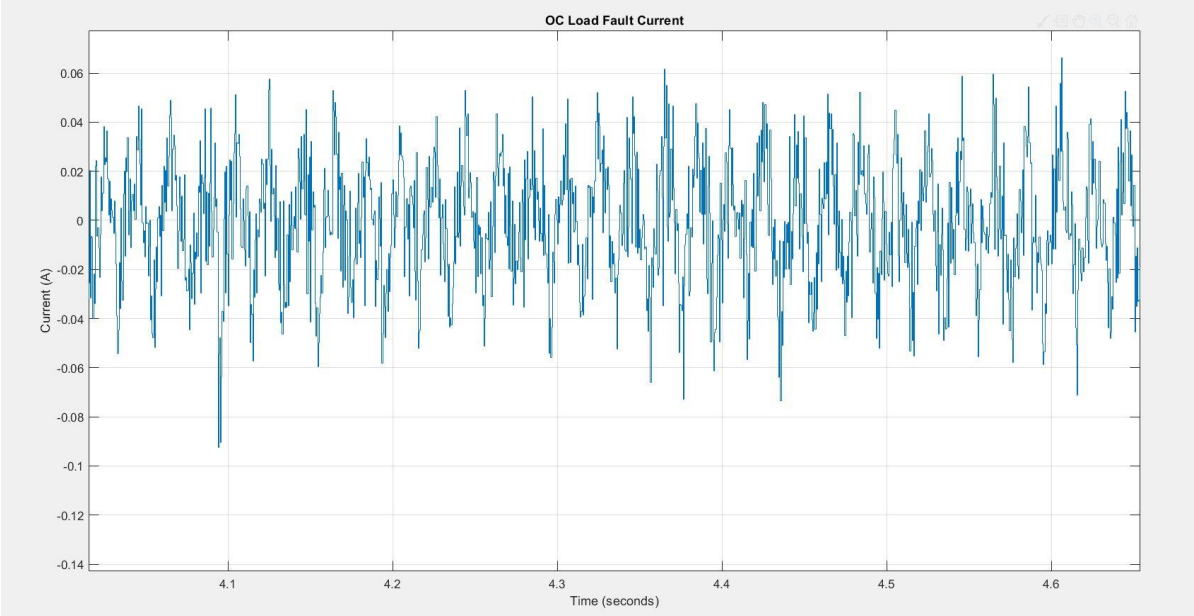


Figure 54 Open circuit load fault current

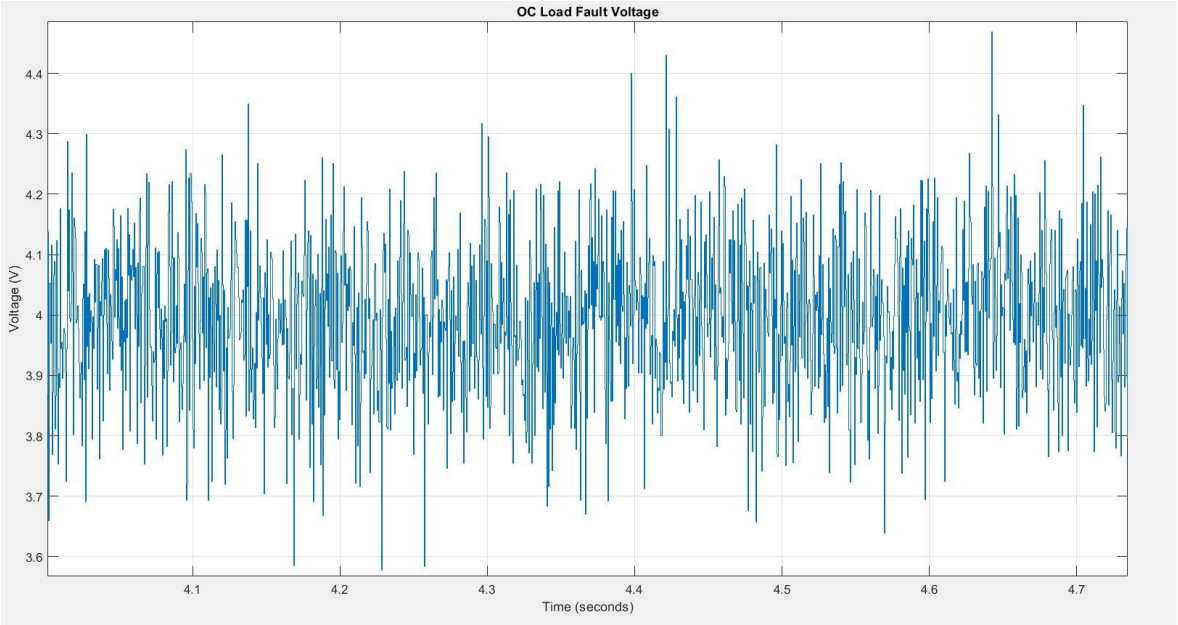


Figure 55 Open circuit load fault voltage

SHORT CIRCUIT LOAD

A short circuit can occur at the load side if there is a malfunction the connected load leading to a short circuit between the terminal, this leads to dangerously high currents and the system cannot operate like this.

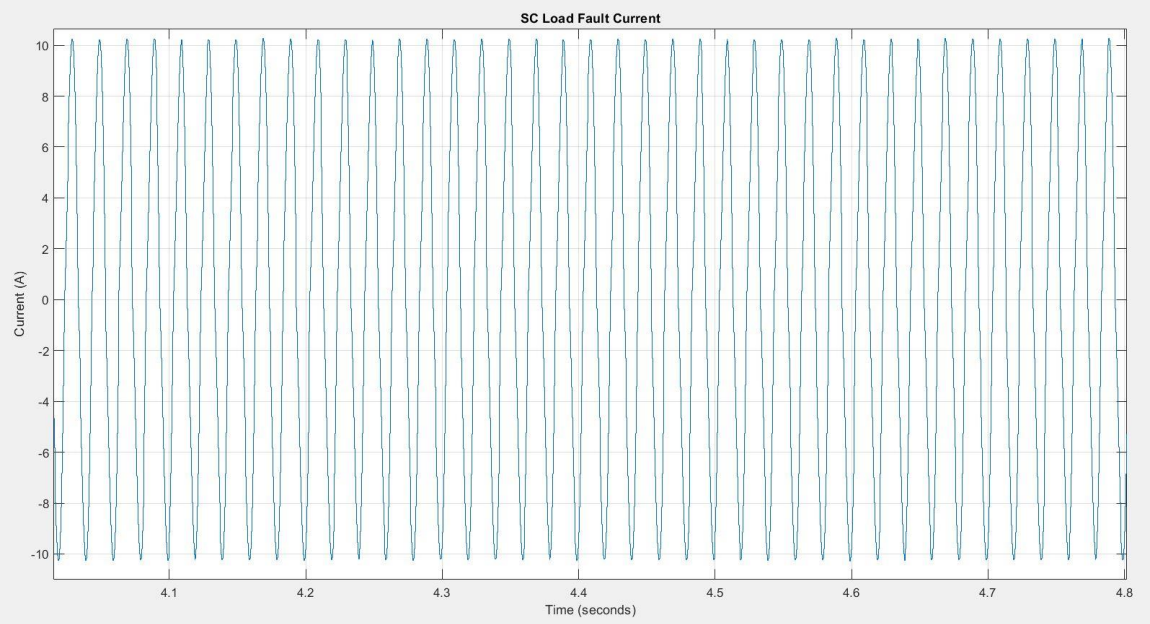


Figure 56 Short circuit fault current

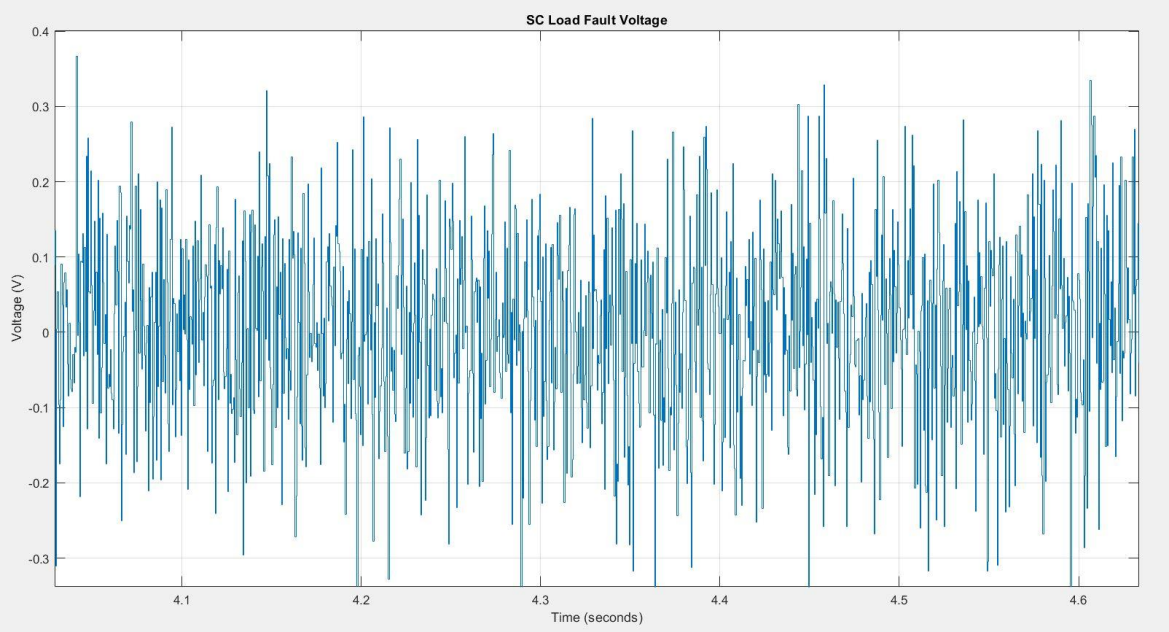


Figure 57 Short circuit load voltage

4.3 FAULT TREE ANALYSIS

A fault tree analysis is now conducted with the faults chosen to implement to obtain a better understanding of what the faults are affecting and how. Because the faults effect the voltage and current in many different ways, the 2 main undesirable events are defined as, change in the current, compared to a healthy system and a change in both current and voltage, compared to a healthy system. The change in voltage is not defined as an event because it does not happen when a fault occurs. After this broad classification, the sublevel events are defined as what exactly is happening to the voltage or current, and below that is the base level event that occurs, which is the fault. The fault tree can be seen in the diagram below.

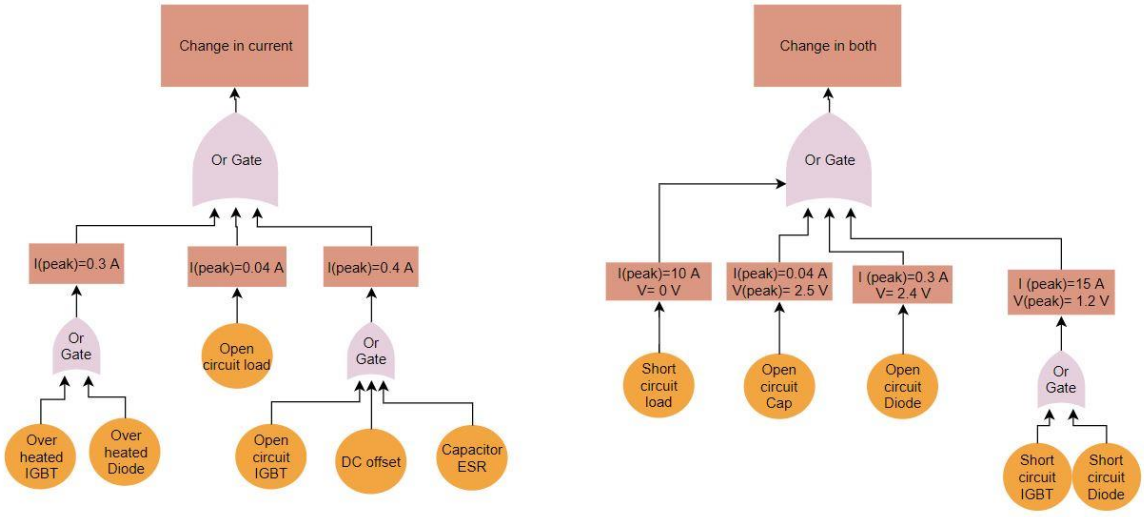


Figure 58 Fault tree analysis of the rectifier

The fault tree analysis was conducted based on the data collected, the substates were defined by examining the data to see how it differentiated from the data of the healthy system, these are differences that are apparent, however the neural network may not classify them based on these differences but some others which by not be as apparent. From the fault tree analysis, it can be seen that multiple faults (denoted by the orange circles) can have the same effect on the voltage or current, this might make detection of those faults difficult or maybe even not possible, this will have to be seen when the algorithm is trained on the data.

4.4 FAILURE MODES AND EFFECTS ANALYSIS

A failure modes and effects analysis(FMEA) is conducted on the rectifier based on the fault tree analysis and the data collected. The sublevel events from the fault tree in figure 61 are defined as possible effects in the FMEA with the faults defined as the root cause. Each root cause is then assigned a score from 1 to 10 in the categories of severity(S), occurrence(O) and detection(D), with 1 being low severity, occurrence or detection and 10 being high severity, occurrence or detection. The numbers assigned to each category are multiplied which gives the risk priority number(RPN).

Possible effects	Root cause	S	O	D	RPN
I(peak)= 0.3A	Overheated IGBT	8	6	5	120
	Overheated Diode	8	6	5	120
I(peak)= 0.04A	Open circuit load	6	2	10	120
I(peak)= 0.4A	Open circuit IGBT	4	4	5	80
	DC offset	3	2	5	30
	High capacitor ESR	3	4	5	60
I(peak)= 10A V= 0V	Short circuit load	10	2	10	200
I(peak)= 0.04A V(peak)= 2.5V	Open circuit capacitor	8	2	10	160
I(peak)= 0.3A V(peak)= 2.4V	Open circuit diode	6	4	10	240
I(peak)= 15A V(peak)= 1.2V	Short circuit IGBT	10	2	1	20
	Short circuit Diode	10	2	1	20

Table 6 FMEA for the rectifier

From the above table, it is clear that the causes with the highest risk priority number are the open circuit diode and the short circuit load faults, both of these have an adverse effect on the system which is represented by their high severity numbers. Another thing to note is the final effect with short circuit IGBT and Diode, these have high severity numbers but they have very low detectability because the voltage and current measurements look identical, although the neural network may be able to detect the faults by looking for some features not clearly visible.

5. DATA PROCESSING AND WORKING OF ALGORITHM

The data collected for the purpose of fault detection is the AC side current and the DC side voltage of the converter. This data needs to be processed before being used for the algorithm for training and validation.

5.1 DATA PROCESSING

The data was collected in MATLAB as timeseries data with a sampling rate of 2000 Hz, where the simulation is run for ten seconds and the fault is introduced at one second, as can be seen in the figure below.

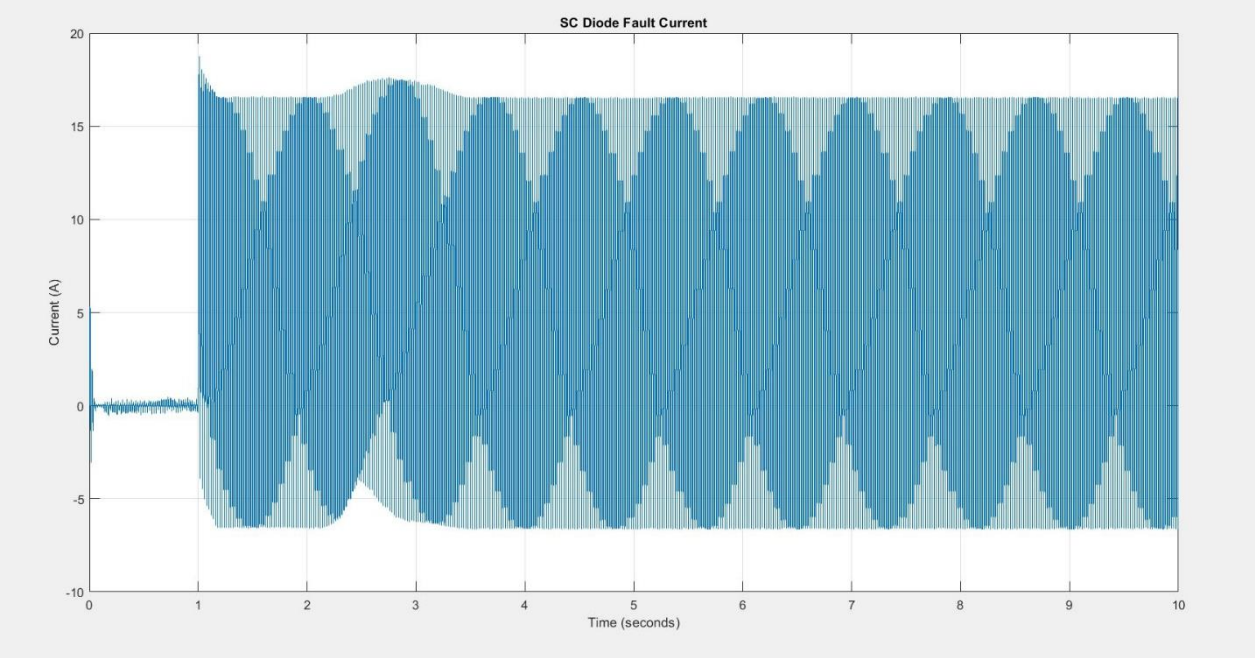


Figure 59 Short circuit diode current

The data before the fault occurs is not used, so only the data after one second is extracted. The total number of useable samples or data points in one measurement of current or voltage is 18000, and five simulations are carried out with noise for each fault. There are 18000 data points for voltage and current each, which can be seen visually represented below.

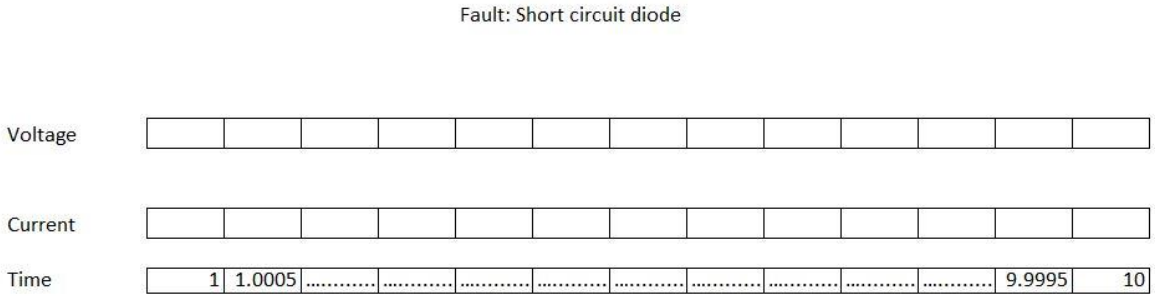


Figure 60 Visual representation of data collected

The machine learning algorithm needs the data to be labelled and processed before, so this data cannot be given to the algorithm in is present state. Each of the cells in the figure 54 represents 200 data points, this data is processed as shown in the diagram below.

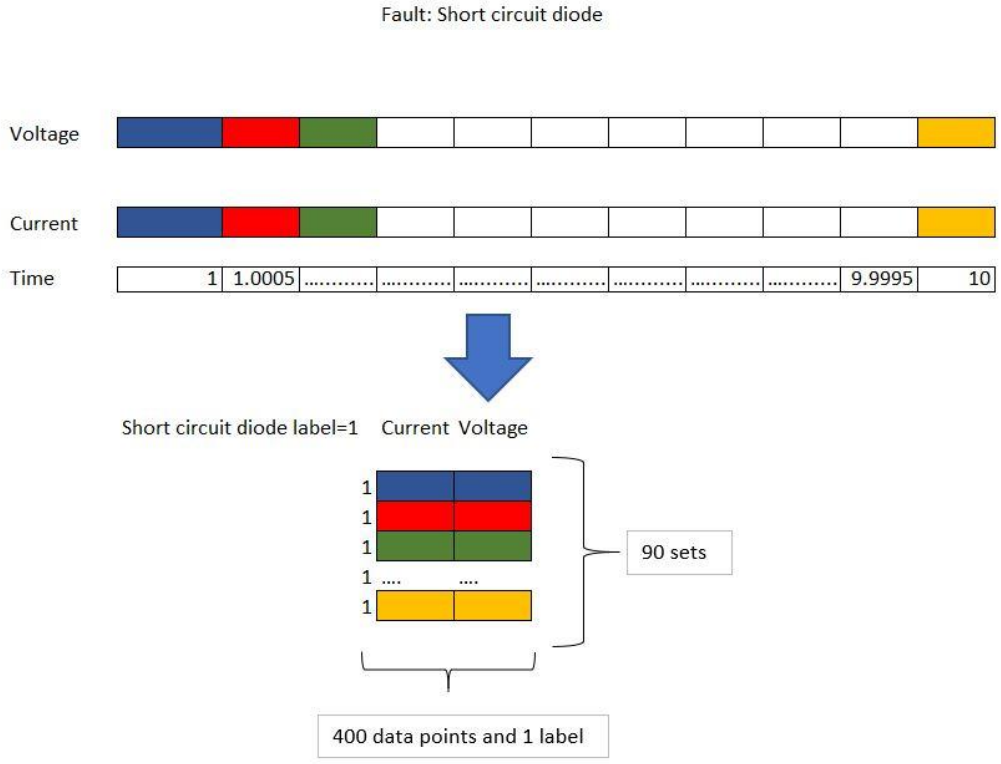


Figure 61 Restructuring of collected data

The data is restructured as shown and the resulting data set is the data from one simulation, this is done again 4 more times so the structure of the dataset collected for one fault is a 401 x 450 data set, with each row starting with the label and then 200 samples of the current followed by 200 samples of the voltage from the same time.

This process is repeated for the other faults and all the data sets are assembled into one master data set containing the measurements and labels for all the faults. This will be split into training and testing data as explained previously in chapter 3.

5.2 STRUCTURE OF THE NEURAL NETWORK

The structure of the neural network chosen to implement has various parameters that need to be determined, either arbitrarily or according to what best suits the purpose of this project. The parameters of the neural network are explained in chapter 3 and are as follows:

- Number of layers
- Type of each layer
- Activation function of each layer
- Number of neurons in each layer
- Filters, padding and strides
- Other parameters not related to structure (Epochs, optimization functions etc.)

The following is a summary of the neural network model in python, most of the parameters mentioned above can be seen.

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 201, 2)	0
conv1d (Conv1D)	(None, 192, 300)	6300
max_pooling1d (MaxPooling1D)	(None, 96, 300)	0
conv1d_1 (Conv1D)	(None, 92, 200)	300200
max_pooling1d_1 (MaxPooling1	(None, 46, 200)	0
conv1d_2 (Conv1D)	(None, 44, 200)	120200
max_pooling1d_2 (MaxPooling1	(None, 22, 200)	0
dropout (Dropout)	(None, 22, 200)	0
flatten (Flatten)	(None, 4400)	0
dense (Dense)	(None, 200)	880200
dense_1 (Dense)	(None, 100)	20100
dense_2 (Dense)	(None, 12)	1212

Figure 62 Summary of the model in python

From the summary of the model, the neural network has a total of 7 layers, the first of which is the input layer, which is mentioned as a reshape layer, so basically the data which is processed as a 402 x 1 vector is just rearranged to a 201 x 2 matrix and that is the input layer of the neural network. The input layer is then followed by 3 one dimensional convolution layers with pooling layers in between them, and then 3 densely connected layers, the last of which is the output layer.

The activation functions for each layer is the same and that is the relu function, it was chosen because of its better performance and faster training in deep neural networks. However, the output layer is the only layer that differs in the type of activation function used, it uses the softmax layer, which was further explained in chapter 3, it basically gives the probability of each fault based on the current data.

The number of filters in each convolutional layer can be determined by looking at the last dimension of each of those layers, so the first convolutional layer has 300 filters with a kernel size of 10 and stride of a default value of 1, the second and the third layers have 200 filters each with kernel sizes of 5 and 3 respectively, the flatten layer is there just to flatten the matrix of 22 x 200 into a 1 x 4400 vector for inputting into the next layer which is a densely connected layer.

The dropout layer is present to prevent the problem of overfitting which was discussed in chapter 3. During training some of the layer outputs are ignored, set to zero or "dropped out". This has the effect of making the layer look like it has a different number of nodes, so each update to a training layer is performed with a different "view" of the layer. This has the effect of forcing certain nodes to take on more or less responsibility for the inputs, which makes the training process a little more intensive but it ultimately makes for a model less prone to overfitting and a lot more robust.

There are some other parameters not related to the neural network directly, such as the number of epochs. A neural network cannot learn all the features from a data set in just one pass over the data, it needs to be fed the data multiple times to extract all the features and accurately predict using unseen data, epochs is the number of times the neural network "sees" the training data.

The final aspect to discuss is the optimization function, this function determines how the gradient descent is done, there are various functions with different advantages, but for this neural network the adam optimization function was chosen, which combines the best parts of the 2 other stochastic gradient descent algorithms, it calculated the exponential moving average and the square of the gradient, and assigns 2 parameters to control the decay rates of these.

5.3 TRAINING OF THE NEURAL NETWORK

Before the training of the neural network is started, the data collected must be shuffled and split into training data and testing data, because if the network is tested on the training data or data that it has "seen" before there is a risk it can just "recall" what the output is and not that it has actually learned any features, so 20% of the data is kept aside for testing and the remaining data is shuffled and used for training the neural network.

As the neural network goes through the data in each epoch, it is using the adam optimization function to adjust the weights according to check what lowers the gradient and eventually after 10 epochs the network is trained and is ready to be tested on testing data that it has not seen.

So essentially the first 3 convolutional layers are responsible for feature extraction from the signals, and the last 3 densely connected layers are responsible for the classification of what kind of fault has occurred.

6. VALIDATION AND RESULTS

6.1 VALIDATION

The validation of the neural network is done using the 20% of the data that was set aside for this purpose. This data has not been seen by the neural network before, the prediction of the neural network is compared to the actual label of the data and the accuracy of the detection is determined by dividing the total number of correct guesses by the network with the total number of data batches of voltage and current.

After the total accuracy is determined the accuracy of detection for each of the faults is also determined to see where the neural network has its limitations and after using the validation data to test the algorithm, the following results were obtained. The model was used to predict the first 20 results from the testing data and the results were compared to the actual labels.

```
Test accuracy: 91.11111164093018 %
Prediction:  SC IGBT  Actual:  SC Diode
Prediction:  OC Load  Actual:  OC Load
Prediction:  OC Diode  Actual:  OC Diode
Prediction:  SC IGBT  Actual:  SC Diode
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  OC IGBT  Actual:  OC IGBT
Prediction:  Cap ESR  Actual:  Cap ESR
Prediction:  OC Load  Actual:  OC Load
Prediction:  OC Load  Actual:  OC Load
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  SC IGBT  Actual:  SC IGBT
Prediction:  SC IGBT  Actual:  SC Diode
Prediction:  Healthy  Actual:  Healthy
Prediction:  OC Diode  Actual:  OC Diode
Prediction:  SC IGBT  Actual:  SC Diode
Prediction:  SC IGBT  Actual:  SC IGBT
Prediction:  DC Offset  Actual:  DC Offset
Prediction:  OC IGBT  Actual:  OC IGBT
Prediction:  SC IGBT  Actual:  SC IGBT
Prediction:  SC IGBT  Actual:  SC Diode
```

Figure 63 Accuracy of the neural network and it's predictions

The neural network has an accuracy of 91.1% in the detection of faults, although it is not ideal, the neural network still does a good job at classifying most of the faults as can be seen in the predictions, however it seems have a difficult time differentiating between short circuited IGBTs and diodes. The accuracy of the neural network in the detection of each state is then examined to see where exactly the network is falling short.

```
Healthy accuracy: 98.66369962692261 %

Open Circuit IGBT accuracy: 100.0 %

Short Circuit IGBT accuracy: 100.0 %

Overheated IGBT accuracy: 97.32739329338074 %

Open Circuit Diode accuracy: 100.0 %

Short Circuit Diode accuracy: 0.0 %

Overheated Diode accuracy: 100.0 %

DC Offset accuracy: 99.55456852912903 %

Capacitor ESR accuracy: 100.0 %

Open Circuit Capacitor accuracy: 100.0 %

Short Circuit Load accuracy: 100.0 %
```

Figure 64 Accuracy of detection of each state

From the accuracy results, the neural network is perfect at classification of IGBT and capacitor faults, but it doesn't seem to recognise the short circuit diode fault state at all, this is very odd behaviour, for the network not recognise even one data batch of that state. To determine what went wrong, the data of the short circuit diode and short circuit IGBT are examined.

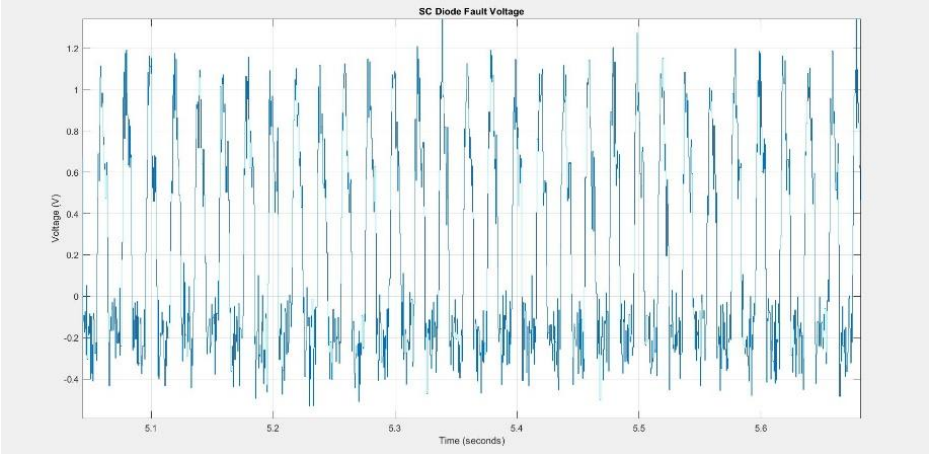


Figure 65 Short circuit diode voltage

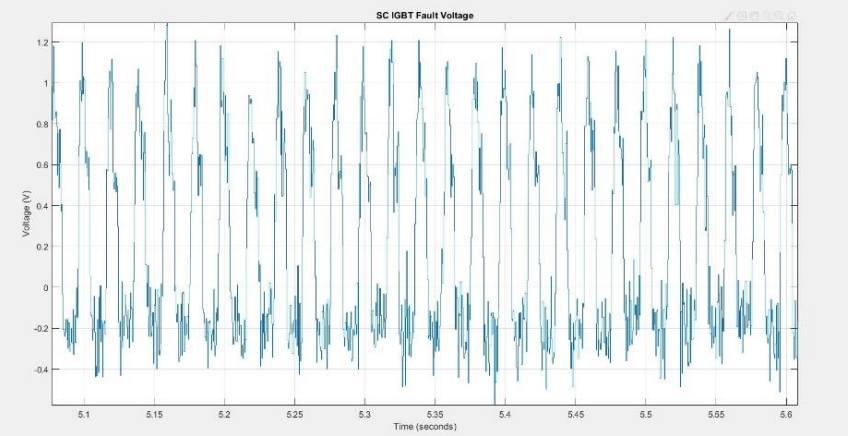


Figure 66 Short circuit IGBT voltage

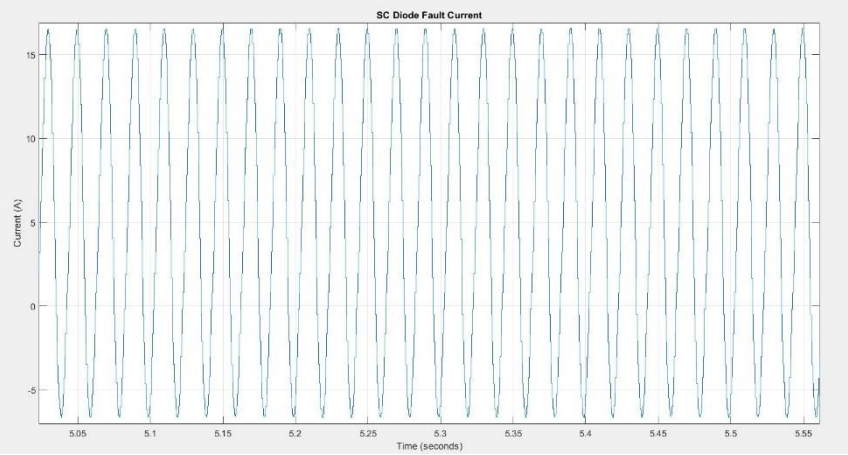


Figure 67 Short circuit diode current

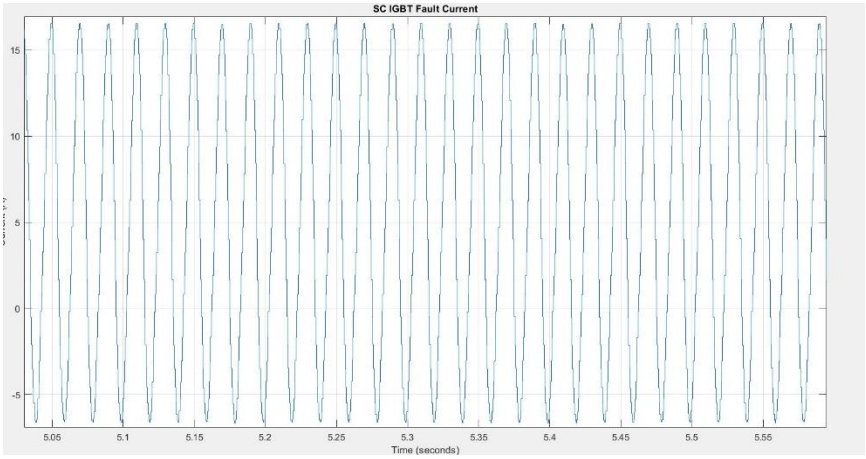


Figure 68 Short circuit IGBT current

The currents and the voltages of both the short circuit faults for the diode and IGBT seem to be matching in amplitude and frequency, they do not even seem to be time shifted, the currents and the voltages seem to be the same. This happens because when either the IGBT or the diode short circuit, there is only one path for the current

to take on that leg of the rectifier, which is the short circuit, this can be seen in the diagrams below.

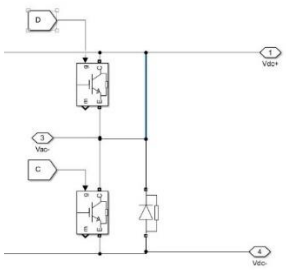


Figure 69 Short circuit diode in the rectifier

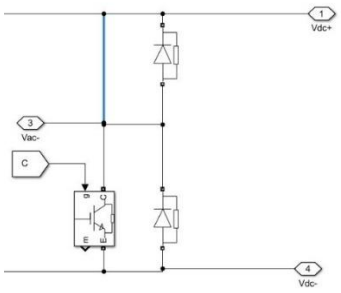


Figure 70 Short circuit IGBT in the rectifier

In both of the diagrams the path of least resistance is always the short circuit and that will be the path that the current will follow, and that is why both sets of data collected for short circuit diode and short circuit IGBT are very similar and why the neural network cannot tell them apart. So to remedy this the 2 faults are clubbed together as one fault state defined as "Short circuit rectifier leg", using this slight modification to the dataset, the neural network is trained and tested again and the accuracy of the network is now expected to be better.

```
Test accuracy: 99.59595799446106 %
Prediction:  Healthy  Actual:  Healthy
Prediction:  OH IGBTS  Actual:  OH IGBTS
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  OH Diodes  Actual:  OH Diodes
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  OC Diode  Actual:  OC Diode
Prediction:  OC IGBT  Actual:  OC IGBT
Prediction:  DC Offset  Actual:  DC Offset
Prediction:  OH IGBTS  Actual:  OH IGBTS
Prediction:  Cap ESR  Actual:  Cap ESR
Prediction:  Cap ESR  Actual:  Cap ESR
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  OC Capacitor  Actual:  OC Capacitor
Prediction:  OH IGBTS  Actual:  OH IGBTS
Prediction:  OC Load  Actual:  OC Load
Prediction:  OC Diode  Actual:  OC Diode
Prediction:  OC Load  Actual:  OC Load
Prediction:  SC Load  Actual:  SC Load
Prediction:  SC Load  Actual:  SC Load
Prediction:  OH IGBTS  Actual:  OH IGBTS
```

Figure 71 New neural network accuracy

The accuracy of the network in the detection of faults has gone up significantly and for the testing data it can detect 99.5% of all the faults, the accuracy of detection of each of the states is examined to see what the neural network is unable to detect.

```
Healthy accuracy: 97.99554347991943 %

Short Circuit Rectifier Leg accuracy: 100.0 %

Open Circuit IGBT accuracy: 100.0 %

Overheated IGBT accuracy: 99.3318498134613 %

Open Circuit Diode accuracy: 100.0 %

Overheated Diode accuracy: 100.0 %

DC Offset accuracy: 100.0 %

Capacitor ESR accuracy: 100.0 %

Open Circuit Capacitor accuracy: 100.0 %

Short Circuit Load accuracy: 100.0 %
```

Figure 72 Accuracy of detection of each state

Here it can be seen that every state has a detection accuracy greater than 99% except for the detection of the healthy state, however it is also more important that the algorithm, detects the faults and it is better if the algorithm detects the faults and give a false positive for healthy states than if the algorithm doesn't detect the fault at some times.

7. CONCLUSION

The purpose of this project was to develop a fault detection method for a controlled rectifier based on machine learning algorithms. The data used for training the neural network was the AC side current and the DC side voltage. The lab work done in this project was mainly related to testing of the control algorithm and initially incorporated additional lab work for the collection of the data for the faults, but unfortunately due to the COVID-19 pandemic the measurements used were from the simulation with added noise. The rectifier's constituent parts were closely examined and some of the faults that could affect each component were determined. From these faults, 11 were chosen to be implemented, within the constraints of the simulation software Simulink. A steady state operating condition for the rectifier was set with the input voltage at 4 V 50 Hz AC and the DC output voltage controlled to 4 V, the measurements taken at this state were labelled as "healthy" with the rest of the faults implemented in the simulation occurring suddenly at 1 second in simulation time.

The data collected was split into training and validation batches, and the neural network was trained on the training data and was tested using the validation data. Initially the neural network was only giving a detection accuracy of 91.1%, then on closer examination of the accuracy of detection of each fault, it was seen that the short circuit diode state was not being detected at all. The data collected for that fault was very similar to the data collected for the short circuit IGBT state, these 2 states were then clubbed into 1 state defined as "short circuit rectifier leg" and the neural network was trained and tested again. After clubbing the 2 faults together the accuracy of the neural network improved significantly to 99.5%, where it was having a little trouble recognising the healthy state of the rectifier but was detecting each fault with an accuracy of greater than 99.5%.

From the results of this project it is evident that machine learning algorithms are a viable and accurate method of fault detection with very few false positives and high accuracy, moreover they excel at pattern recognition if given enough data. However machine learning algorithms do come with their drawbacks as well, one of the main drawbacks is that a large amount of data is required for the algorithm to be trained, and if the size of the dataset is not large enough it may lead to overfitting and it may not be able to detect some faults. But nowadays, large amounts of data is starting become publicly available, which may be reliably used to make better algorithms.

7.1 FUTURE WORK

Further work can be done to improve the results of the project, the measurements could be collected on a real controlled rectifier with the faults implemented on the components and the data collected in real time, this would give a much more accurate picture of how the algorithm could be used for real devices. Also another aspect that could be further explored is the machine learning side. Different topologies of neural networks could be tested to see if the accuracy can be further improved such as recurrent neural networks or unsupervised learning, or training algorithms could be used with control to develop a method of fault tolerant control.

References

[1] E. Wiggelinkhuizen, T. Verbruggen, H. Braam, "Condition monitoring for offshore windfarms", 2003

[2] J. Helsen, C. Devriendt, W. Weijtjens, P. Guillaume, "Condition monitoring by means of SCADA analysis", EWEA 2015 Annu. Event, Nov. 2015.

[3] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review", Renewable Energy an international journal, 2018.

[4] F. Fuchs, "Some diagnosis methods for voltage source inverters in variable speed drives with induction machines-a survey," Ind. Electron. Soc. 2003. IECON'03, pp. 1378–1385, 2003.

[5] S. S. Manohar, A. Sahoo, A. Subramaniam and S. K. Panda, "Condition monitoring of power electronic converters in power plants — A review," 2017

[6] I. Silva, D. Spatti, R. Flauzino. L. Liboni. S. Alves, "Artificial Neural Networks", Springer, 2017.

[7] A.Krizhevsky, I.Sutskever, G.Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", 2017

[8] S. Chavan, M. Chavan, "Power Switch Faults Diagnosis and Tolerant Schemes in Converters of Photovoltaic System", 2014

[9] Z. Sang, C. Mao, J. Lu, D. Wang, "Analysis and Simulation of Fault Characteristics of Power Switch Failures in Distribution Electronic Power Transformers", Energies 2013, vol. 6, pp. 4246-4268, 2013

[10] A. Tch, "The mostly complete chart of Neural Networks, explained", https://towardsdatascience.com, Aug. 4, 2017

[11] H. Schulz, S. Behnke, "Deep Learning", KI - Künstliche Intelligenz, 2012

[12] S. R. Rath, "Convolutional Neural Network Architectures and Variants", https://debuggercafe.com, June 23, 2019

[13] H. Nakao, Y. Yonezawa, Y. Nakashima and F. Kurokawa, "Failure prediction using low stability phenomenon of digitally controlled SMPS by electrolytic capacitor ESR degradation," 2017 IEEE Applied Power Electronics Conference and Exposition (APEC), Tampa, FL, 2017, pp. 2323-232

[14] International Rectifier, "INSULATED GATE BIPOLAR TRANSISTOR", IRGB6B60KPbF datasheet,2003

[15] WeEn Semiconductors, "Dual ultrafast power diodes", BYQ28E-200E datasheet, 2018

APPENDIX A

MATLAB CODE FOR DATA PROCESSING

```
function B =
combiner(i1,i2,i3,i4,i5,v1,v2,v3,v4,v5,label)
I1=i1.Data;
I2=i2.Data;
I3=i3.Data;
I4=i4.Data;
I5=i5.Data;
I=(cat(2,I1,I2,I3,I4,I5))';
V1=v1.Data;
V2=v2.Data;
V3=v3.Data;
V4=v4.Data;
V5=v5.Data;
V=(cat(2,V1,V2,V3,V4,V5))';
r=1;
j=2001;
for i=1:5
    j=2001;
    while j<=19801
        T1=I(i,j:j+200);
        T2=V(i,j:j+200);
        T=horzcat(T1,T2);
        comb(r,:)=T;
        r=r+1;
        j=j+200;
    end
end
comb=[ones(450,1)*label comb];
B=comb;
```

APPENDIX B

PYTHON CODE NEURAL NETWORK

```
import numpy as np
import sklearn
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from pandas.plotting import
register_matplotlib_converters
a=np.loadtxt('data11.txt', delimiter=',')
print(a)
a = np.expand_dims(a, axis=2)
labels=(a[0:4949, 0])-1
labels_names = ['Healthy', 'DC Offset', 'OC Load', 'OC
IGBT', 'OC Capacitor','SC Rectifier','OC Diode','SC
Load', 'OH Diodes', 'Cap ESR', 'OH IGBTS']
print(labels)
data=a[0:4949, 1:403]
print(data)
print(a[0:4949, 0])
data_train, data_test, labels_train, labels_test =
train_test_split(data, labels, test_size=0.20,
random_state=42)

model = keras.Sequential([
    keras.layers.Reshape((201,2), input_shape=(402,1)),
    keras.layers.Conv1D(filters=300, kernel_size=10,
activation='relu',padding='valid', input_shape=(201, 2)),
    keras.layers.MaxPooling1D(),
    keras.layers.Conv1D(filters=200, kernel_size=5,
activation='relu',padding='valid'),
    keras.layers.MaxPooling1D(),
    keras.layers.Conv1D(filters=200, kernel_size=3,
activation='relu',padding='valid'),
    keras.layers.MaxPooling1D(),
    keras.layers.Dropout(0.5),
    keras.layers.Flatten(),
    keras.layers.Dense(200, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(11, activation="softmax")
```

```
    ])

##Defining other aspects of NN
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
##Training NN
model.fit(data_train, labels_train, epochs=10)
##Testing NN
test_loss, test_acc = model.evaluate(data_test,
labels_test)
nofault_loss, nofault_acc =
model.evaluate(data[0:449,0:402],labels[0:449])
DCO_loss, DCO_acc =
model.evaluate(data[450:899,0:402],labels[450:899])
OCL_loss, OCL_acc =
model.evaluate(data[900:1349,0:402],labels[900:1349])
OCI_loss, OCI_acc =
model.evaluate(data[1350:1799,0:402],labels[1350:1799])
OCC_loss, OCC_acc =
model.evaluate(data[1800:2249,0:402],labels[1800:2249])
SCD_loss, SCD_acc =
model.evaluate(data[2250:2699,0:402],labels[2250:2699])
OCD_loss, OCD_acc  =
model.evaluate(data[2700:3149,0:402],labels[2700:3149])
SCL_loss, SCL_acc =
model.evaluate(data[3150:3599,0:402],labels[3150:3599])
OHD_loss, OHD_acc =
model.evaluate(data[3600:4049,0:402],labels[3600:4049])
ESR_loss, ESR_acc =
model.evaluate(data[4050:4499,0:402],labels[4050:4499])
OHI_loss, OHI_acc =
model.evaluate(data[4500:4949,0:402],labels[4500:4949])
print('\nTest accuracy:', test_acc*100, '%')
prediction=model.predict(data_test)
for i in range(20):
    print("Prediction: ",
labels_names[np.argmax(prediction[i])], " Actual: ",
labels_names[int(labels_test[i])])
print('\nHealthy accuracy:', nofault_acc*100, '%')
print('\nShort Circuit Rectifier Leg accuracy:',
SCD_acc*100, '%')
print('\nOpen Circuit IGBT accuracy:', OCI_acc*100, '%')
print('\nOverheated IGBT accuracy:', OHI_acc*100, '%')
print('\nOpen Circuit Diode accuracy:', OCD_acc*100, '%')
print('\nOverheated Diode accuracy:', OHD_acc*100, '%')
print('\nDC Offset accuracy:', DCO_acc*100, '%')
print('\nCapacitor ESR accuracy:', ESR_acc*100, '%')
```

```
print('\nOpen Circuit Capacitor accuracy:', OCC_acc*100,
'%')
print('\nShort Circuit Load accuracy:', SCL_acc*100, '%')
print('\n')
```