



**VIT**<sup>®</sup>  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)



**UNIVERSITI  
TEKNOLOGI  
PETRONAS**

# **Employment of ANN for Predictive Motor Maintenance and Bearing Fault Detection using Park's Vector Analysis**

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

*in*

**ECE with Specialization in IoT and Sensors**

*by*

**Aadhi Aadhavan. B (18BIS0132)**

**Valli Meenaa. V (18BIS0091)**

**Under the guidance of**

**Dr. Sujatha R**

**SENSE, VIT, Vellore**

**Dr. Madiyah Omar**

**UTP, Malaysia**

**May, 2022**

## **DECLARATION**

I hereby declare that the thesis entitled “Employment of ANN for Predictive Motor Maintenance and Bearing Fault Detection using Park's Vector Analysis” submitted by me, for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering with Specialization in Internet of Things and Sensors to VIT is a record of bonafide work carried out by me under the supervision of Dr. Sujatha R and Dr. Madiyah Omar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Hyderabad

Date: 03-05-2022

Valli Meenaa V.

**Signature of the Candidate**

## **CERTIFICATE**

This is to certify that the thesis entitled “Thesis title” submitted by Valli Meenaa V. & 18BIS0091, SENSE, VIT, for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering with Specialization in Internet of Things and Sensors, is a record of bonafide work carried out by her under my supervision during the period, 01. 12. 2021 to 30.04.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Hyderabad

Date: 03-05-2022

**Signature of the Guide**

**Internal Examiner**

**External Examiner**

HOD: Sasikumar P

ECE with Specialization in IoT and Sensors

## **ACKNOWLEDGEMENTS**

I, Valli Meenaa V., hereby acknowledge that I underwent a wonderful learning experience in the collaboration between VIT Vellore and UTP Malaysia. I am grateful for having a chance to communicate with and learn from wonderful people and professionals who lead me through this period.

I convey my sincere gratitude to my supervisors Dr. Sujatha R. and Dr. Madiyah Omar. Without their kind direction and proper guidance, this Research Project would have been futile. I would like to thank them for guiding me and giving me practical enlightenment of topics, I only knew theoretically, prior to this venture. I am especially thankful to VIT Vellore for presenting me with this opportunity.

Last but not the least, I am grateful to my teammate, Aadhi Aadhavan B. for the wonderful teamwork, much needed to fulfil the needs of the Capstone Project.

**Valli Meenaa V.**

## **EXECUTIVE SUMMARY**

One of the most effective ways to understand a motors health and efficiency is through Motor Current Signature Analysis (MCSA). The proposed model strives to predict instantaneous motor health my employing MCSA by plotting the Park Vector analysis graph. This graph would then be fed as the input to an Artificial Neural Network (ANN), which would classify the motor as healthy or unhealthy based on the Park Vector plot. The trained model has managed to acquire an accuracy >99% in the test set, thus triumphing over other methods of predictive maintenance. The proposed model is built of python, and it is designed in such a way that the computational complexity of the proposed model is low enough to run on edge computing devices so as to implement an efficient industrial automation system.

The industrial application of the proposed project has huge implications in the maintenance industry, as it enables cost efficient non-invasive and highly accurate implementation of predictive maintenance and analysis.

## **Table of contents:**

Declaration.....	2
Certificate.....	3
Acknowledgement.....	4
Executive Summary.....	5
Table of contents.....	6
List of figures.....	7
List of Tables.....	8
<b>1. Introduction.....</b>	<b>9</b>
1.1. Research motivation.....	9
1.2. Objective.....	9
1.3. Background.....	10
1.3.1. Literature Survey	
<b>2. Project description and goals.....</b>	<b>14</b>
2.1. Project description.....	14
2.2. Goals.....	14
<b>3. Technical Specifications.....</b>	<b>15</b>
3.1. Hardware Specifications.....	15
3.2. Software Specifications.....	15
<b>4. Software design approach.....</b>	<b>15</b>
4.1. Introduction.....	15
4.2. Development flowchart.....	17
4.3. Data collection.....	17
4.4. Example of data.....	18
4.5. Development of ANN.....	20
4.6. Deployment of ANN in Rpi.....	24
<b>5. Hardware Design Approach.....</b>	<b>28</b>
5.1. Design architecture.....	28
5.2. Design of power source.....	28
5.3. Design of external casing.....	29
<b>6. Schedule, Tasks and Milestones.....</b>	<b>30</b>
6.1. Constraints faced.....	31
<b>7. Results and Discussions.....</b>	<b>32</b>
7.1. ANN Accuracy.....	32
7.2. Deployment of ANN in Rpi.....	32
<b>8. Summary.....</b>	<b>34</b>
<b>9. References.....</b>	<b>35</b>
<b>10. Appendix A.....</b>	<b>38</b>

## List Of Figures

<b>Illustration number</b>	<b>Title</b>	<b>Page number</b>
1	Portrays the Park Vector Analysis graph of a Healthy Motor	15
2	Portrays the Park Vector Analysis graph of a Un-Healthy Motor	15
3	Project development flowchart	16
4	Data collection process	16
5	Examples of healthy Park Vector Images	17
6	Examples of un-healthy Park Vector Images	18
7	Visualization of the ANN developed for image classification	19
8	ANN development flowchart	20
9	ReLu activation function	21
10	Sigmoid activation function	21
11	Raspberry Pi Landing page	23
12	OS selection	23
13	Version check	25
14	AI-PMUL accompanied with Rogowski coils	27
15	3D Printed case for the MOBIT	28
16	MOBIT hardware set-up	28
17	Initially planned Gantt Chart	29
18	Implemented Gantt Chart	29
19	RaspberryPI model B	32

## List of Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
1	Testing Result Analysis of Proposed Method in terms of distinct measures	32
2	Confusion matrix	32



# **I. Introduction:**

## **1.1. Research Motivation:**

Predictive maintenance is an emerging concept that is gaining mainstream popularity in the field of industrial automation. It involves constant monitoring of the machinery to understand the factors which are involved in the deteriorating health of a certain machine or system. This allows the industry to know the instantaneous health of important machinery and conduct maintenance and shut-downs as and when necessary, without causing extensive damage to said machinery or the entire subsystem.

## **1.2. Objective:**

- Industrial 3-phase Induction motors are one of the most widely used and important pieces of equipment in most systems. Thus, the tracking of motor health becomes ineluctable.
- Bearings tend to be the primary component of any machinery that involved rotational mechanics like the motor.
- Motors are prone to extensive wear and tear especially because of the variable load that they incur and this wear and tear is reflected directly on the bearing causing bearing faults.
- Current methods of preventive maintenance lack the non-invasive factor proposed in this model and further, they lack the level of accuracy attained by the proposed model.

## **1.3. Background:**

Even though induction motors (IMs) can be extremely robust and reliable machines, they are highly susceptible to wear and tear and a plethora of different practical

issues such as bearing fault, fluctuating current input, and ambiguous change in the motor load. Bearings are the most critical component of any instrument which involves rotational mechanics. Needless to say, they play a major role in the working of induction motors. But, as prominent as the use of bearings may be, they are susceptible to faults because bearings are cheap and aren't built to last. This is why, industries often undergo scheduled maintenance where the health of all these motors/bearings is checked and replaced if necessary. Studies have stated that, more than 40% of the IMs failures are to be associated with bearing faults. Thus, the instantaneous monitoring and prediction of IM health is ineluctable.

As bearing fault analysis is a significantly increasing concern in the industry, early prediction of bearing faults would pave way too much forgiving maintenance costs. This would directly lead to the smoother operation of the industry. The proposed model uses the concept of MCSA [2][3][4] by means of park vector analysis to diagnose the IM. This Park Vector analysis graph is then fed as the input of an ANN image classifier which would classify the motor as healthy or unhealthy.

Even though vibration-based analysis, or temperature-based analysis of motor health is more well known, MCSA is the best suited for motor health analysis especially when the motors are in a highly in-accessible or remote areas, which is usually the case with industrial IMs. MCSA is chiefly used in applications like condition monitoring [5] of IMs breakage in rotor bar and end ring and also in detection of bearing faults. In the proposed model, the MCSA is conducted through Park Vector Analysis. This method transforms the IMs 3 phase stator current into two individual orthogonal phases and the finds the bearing fault based on the spatial variation of Park Vector Current locus.

Further, the proposed and designed model is compatible with linux OS, and can thus be deployed in a RaspberryPi (RPI), which is the preferred edge device in most industries for the purpose of industrial automation. The deployed model has proved to be robust and stable and has outperformed the accuracy of the existing predictive maintenance models. The usage of RPI enables the industry to conduct predictive maintenance at a at a very blow cost factor and at a miniscule energy footprint thus

enabling industries to reduce full scale maintenance and run the plants in a smooth and effective way

### 1.3.1 Literature Review:

Park's Vector analysis for detecting a wide variety of bearing faults was first introduced in [6]. This was proposed as an alternative for the stator current analysis and instantaneous power analysis methods as these models aren't effective for the detection of bearing faults. This literature proposes a non-invasive, efficient method for condition monitoring for bearing fault by employing the use of Park's Vector Analysis. It also aims to localize and segregate the detected and diagnosed faults for easy and effective maintenance. The extensive theoretical and empirical analysis provided by hardware implementation in this literature concludes that, the proposed methodology, can not only diagnose either kind of bearing faults, but also classify them, based on their Park's Vector Approach's features.

A novel method for fault detection, namely, hierarchical multitask convolutional neural networks (HMCNN) was introduced in [7]. It is an amalgamation of hierarchical classifiers, multitask learning and a robust CNN architecture. The proposed model only has the necessity to train with the data acquired through the various proposed data acquisition methods to detect and build a multi-classification task, where the shared layer is used to reduce training parameters which in-turn reduce the computational load, enabling us to deploy such models on the edge so as to implement industrial automation. Further the HMCNN model improves network generalization ability.

A systematic summary of the existing literature on the usage of machine learning and deep learning models for bearing fault diagnostics was provided in [8]. The said literature speculates that, even though various algorithms like ANN, SVM and various other machine learning models have been successfully implemented for the detection and diagnosis of bearing faults, the development of newer DL algorithms have piqued interest and enabled many to revisit this pressing issue. Deep Learning methods have been specifically chosen for bearing fault analysis as they hold

superiority in fault feature extraction and fault classification as compared to conventional statistical models. [9] proposes a novel method to construct a dual dimensional fault diagnosis representation from the existing mono dimensional acoustic emission signals. A state-of-the-art strategy for the deployment of the continuous wavelet transform with damage frequency band information to render the defect signature wavelet image (DSWI). This DSWI, is then used as the input of a Deep Convolutional Neural Network (DCNN) which is employed to detect and classify the bearing fault in the selected motor.

An intelligent fault diagnosis system has been proposed in [10]. A Deep Convolutional Neural Network has been employed to act on the raw vibrational data, considering the fact that, vibrational data is the most commonly used feature for bearing fault detection and it is compatible with the predictive maintenance schemes of most existing industries. The proposed DCNN has a huge scalability factor due to the presence of its novel multi-scale convolutional layers, while also reducing the parameters required for robust training, reducing the computational cost.

The detection of faults in an inter turn stator which is housed in a double fed induction motor which uses pulse width modulation is explored in [11]. The proposed method uses the Park's Vector Approach, coupling it with a neural network for automating the detection and diagnosis process. The problem with the proposed model is that, the detection model works with really high accuracy only when the motor is being operated in a steady state. Another problem with a single Park Vector is that, while it is relatively easy to detect faults, the method fails to classify the faults. This is overcome by the usage of multiple Park Vectors (MVPs). This method is explored in [12]. The characteristic fault frequency component (CFFC) from a variety of features such as stator and rotor winding faults, bearing faults etc. are extrapolated from the 3-phase stator current (MCSA). It has been noticed that under healthy conditions, the frequency component of the MCSA remains unbalanced. Upon adding the correction coefficient, the parks vector model appears in a circular shape for the healthy condition. But, according to the fault condition, the amplitude changes, thus changing the shape of the Park's Vector chart to be more elliptical. By monitoring the variations and anomalies in the

various Park's Vector charts, we are able to differentiate and classify the faults and also have an idea about the severity of the said fault. The experiments in [20] were verified on a 7.5hp p three phase wound rotor induction motor (WRIM).

The limitation of the MCSA, namely its disability to detect and diagnose small faults was discussed in [13]. A new Motor square Current Signature Analysis (MSCSA) was proposed in the literature in question. The proposed methodology is divided into 3 different steps. The IM current is first recorded with various data acquisition techniques, the recorded values are squared and the squared values are used to plot a frequency analysis of the square current. This allows richer and more refined information to be analysed, thus allowing the detection of minuscule faults in the IM motors, successfully evading the shortcomings of traditional Park's Vector Approach

As stated in [14] it is self-explanatory that increasing the accuracy of fault detection in industries gives rise to improved system safety and economic performance. The above work directly challenges the problem of roller bearing with race faults. It accomplishes this by creating training data through the employment of high-resolution simulations, which takes into account the physics and dynamics of roller bearings. This data is then used to train a novel machine learning algorithm which is validated against other experimental data sets. The above literature, compares a variety of other algorithms to conduct a study which postulates the efficiency and the accuracy of different models, taking into account the computational cost at every step. In [15] a novel application of dynamic time warping (DTW) to bearing fault classification has been proposed. It was concluded that the proposed model was robust, and was considered a parameter free method of race fault detection.

## **II. Project Description and Goals**

### **2.1. Project Description:**

- The proposed model employs MCSA by plotting Park's vector analysis graph to predict motor health.
- The Park vector Analysis gets its input from the 3 phases of the motor, and it plots it as a graph. The shape of the graph represents the health of the motor. The Park vector graphs are then fed into the ANN model.
- The trained model has managed to acquire an accuracy >99% in the test set, thus triumphing over other predictive maintenance methods.
- It is designed so that the proposed model's computational complexity is low enough to run on edge computing devices to implement an efficient industrial automation system.

### **2.2. Goals:**

- Allow prediction of motor bearing faults in a preemptive fashion, which reduces the need for full-fledged shutdowns and plant closure.
- Design apparatus to measure and detect IM current ( $< 1000\text{ A}$ ) and map it from 0-5 V so as to deploy MCSA (AI based) via RPi.
- Allow instantaneous monitoring of motor health.
- Due to the predictive nature of fault detection, motors don't go awry and pull down the efficiency of the industry.
- Reduce the frequency of motor replacement.
- Development of a fully autonomous Motor Bearing Non-Invasive Fault Testing Rig (MOBIT).
- Helps the industry economically by reducing full-scale shutdowns and ensuring a constant supply of petroleum products.

### **III. Technical Specifications**

#### **3.1. Hardware Specifications:**

- Industrial 3-phase Induction Motor
- Clamp Transducers (LEM RT-2000)
- Rogowski Coils
- BNC Female Connector
- Integrator/Signal Conditioner (LEM AI-PMUL)
- Raspberry Pi Debian x64

#### **3.2. Software Specifications:**

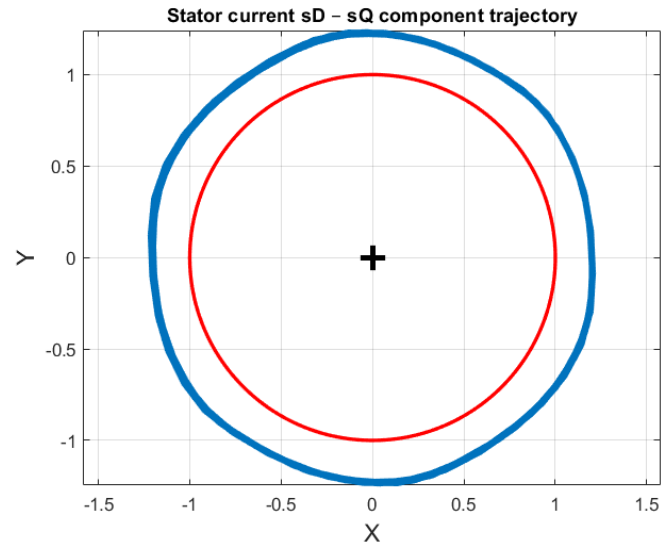
- Python (SciKit Learn)
- Keras
- TensorFlow
- Matplotlib

### **IV. Software Design Approach**

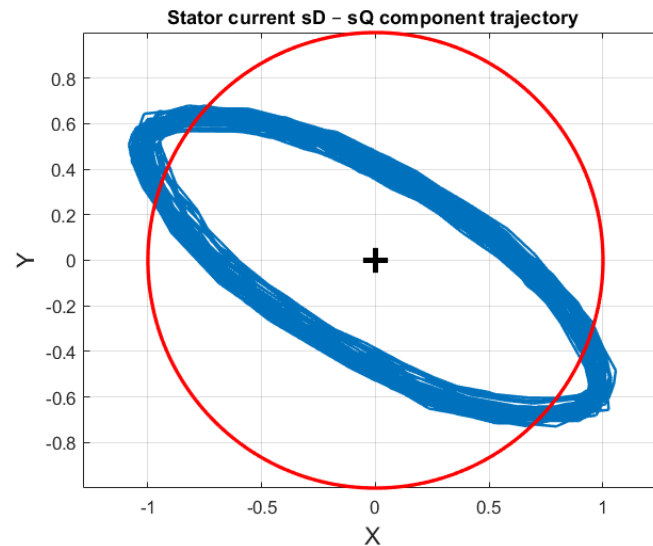
#### **4.1. Introduction:**

The objective of the project is to develop and deploy a python based artificial intelligence model which can monitor and predict the health of Industrial Induction Motors (IIM). The parameters or the features that are required by the AI model to learn and make predictions are provided by a MCSA method called Park Vector Analysis. The Park vector Analysis gets its input from the 3 phases of the motor and it plots a graph. The shape of the graph represents the health of the motor. A, well rounded circular graph represents a healthy motor whereas an elliptical graph represents a motor that is in need of maintenance.

Few examples of Park vector graphs are as shown below in illustration 1 and 2



**Illustration 1:** Portrays the Park Vector Analysis graph of a Healthy Motor



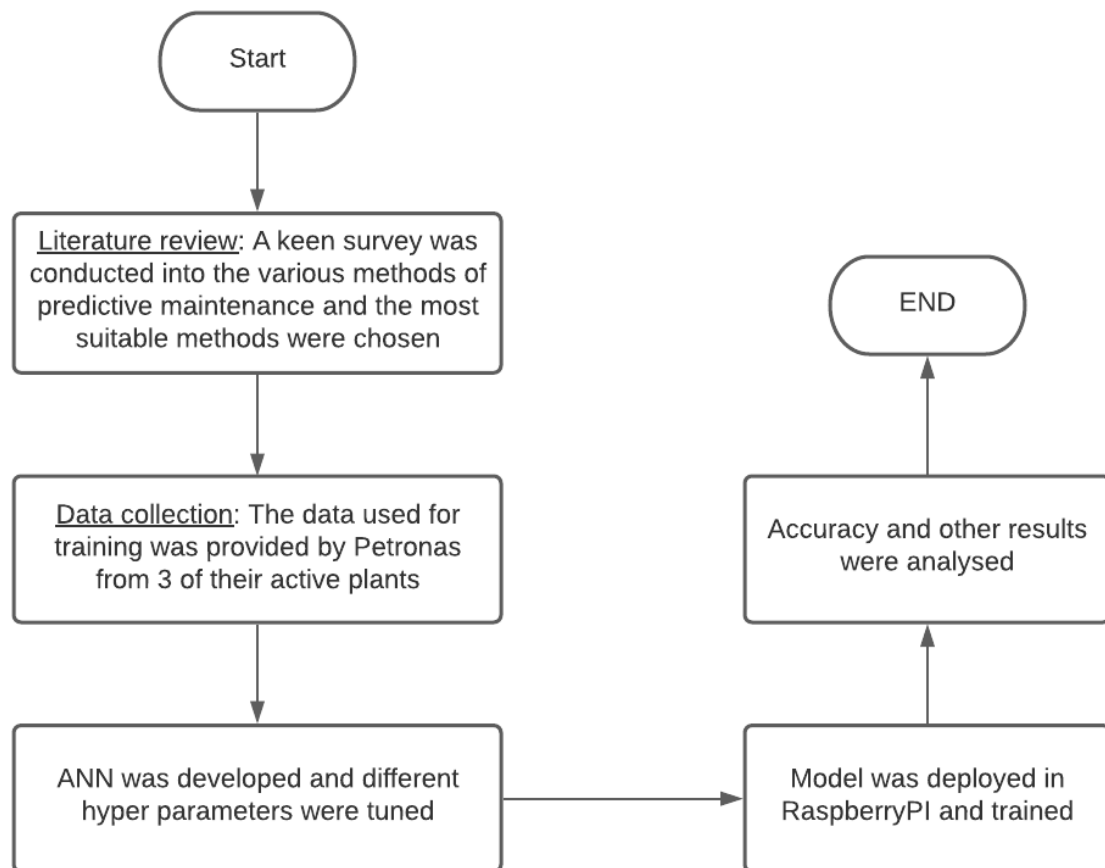
**Illustration 2:** Portrays the Park Vector Analysis graph of a Un-Healthy Motor

As we can see, the stator current sD-sQ component trajectory for a healthy motor is circular, whereas for an unhealthy motor, it assumes an elliptical shape.

Thus, the development of an image classification algorithm became a necessity to automatically classify these images for predictive maintenance which is a huge part of industrial automation.



## 4.2. Development Flowchart



**Illustration 3:** Project development flowchart

## 4.3. Data Collection / generation:

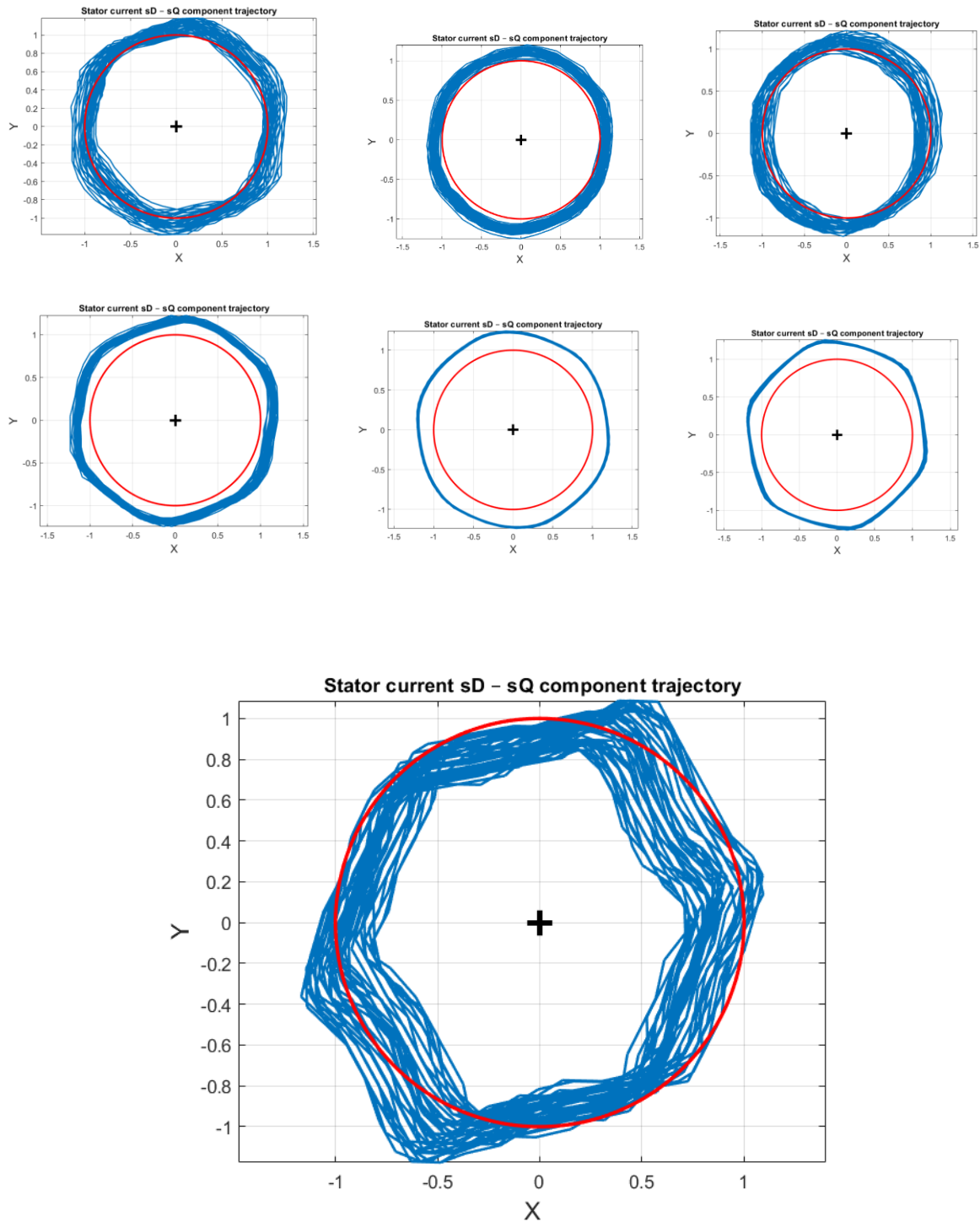
The data was provided by Petronas for their industry 4.0 programme. The data was provided from 3 of their currently active plants (Illustration 4). This data was processed with the use of Park Vector analysis to arrive at the graphs shown in illustration 1 and 2.



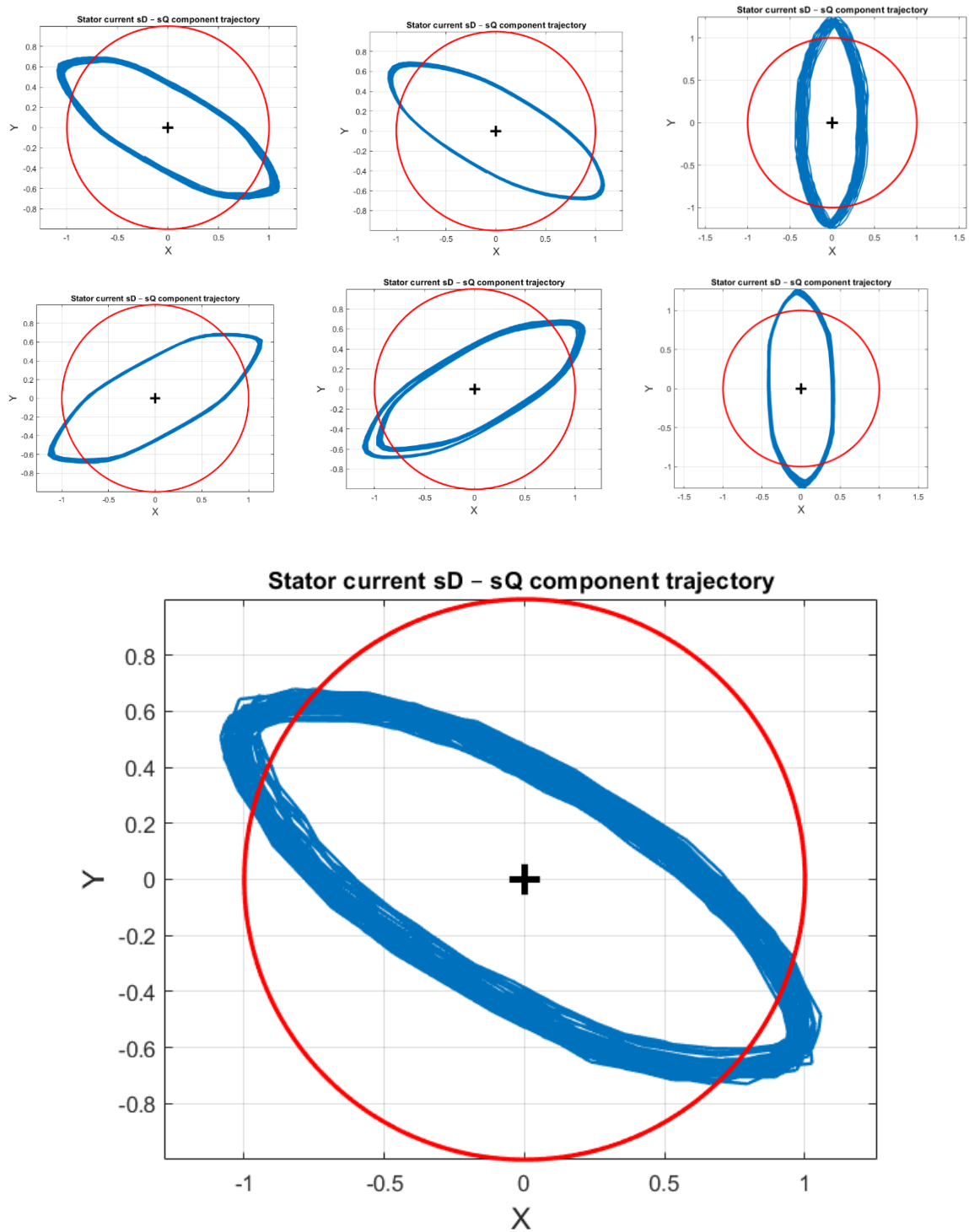
**Illustration 4:** Data collection process

#### 4.4. Example of data fed to the ANN:

Healthy:

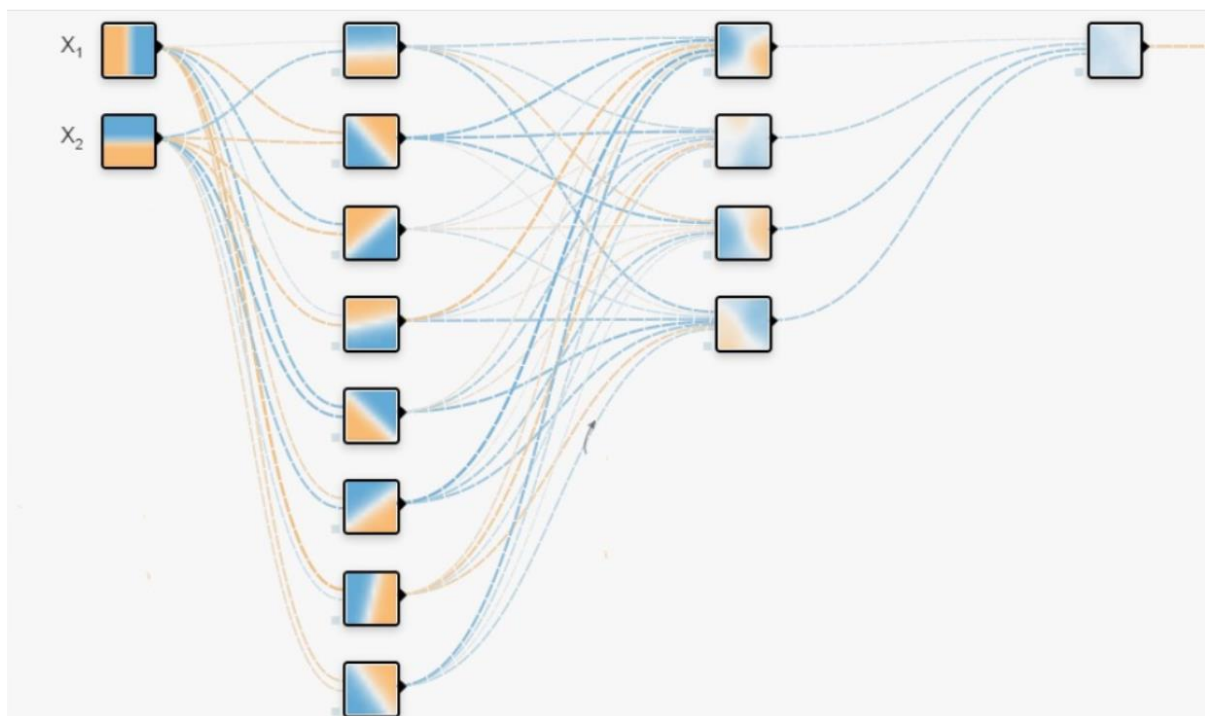


**Illustration 5:** Examples of healthy Park Vector Images

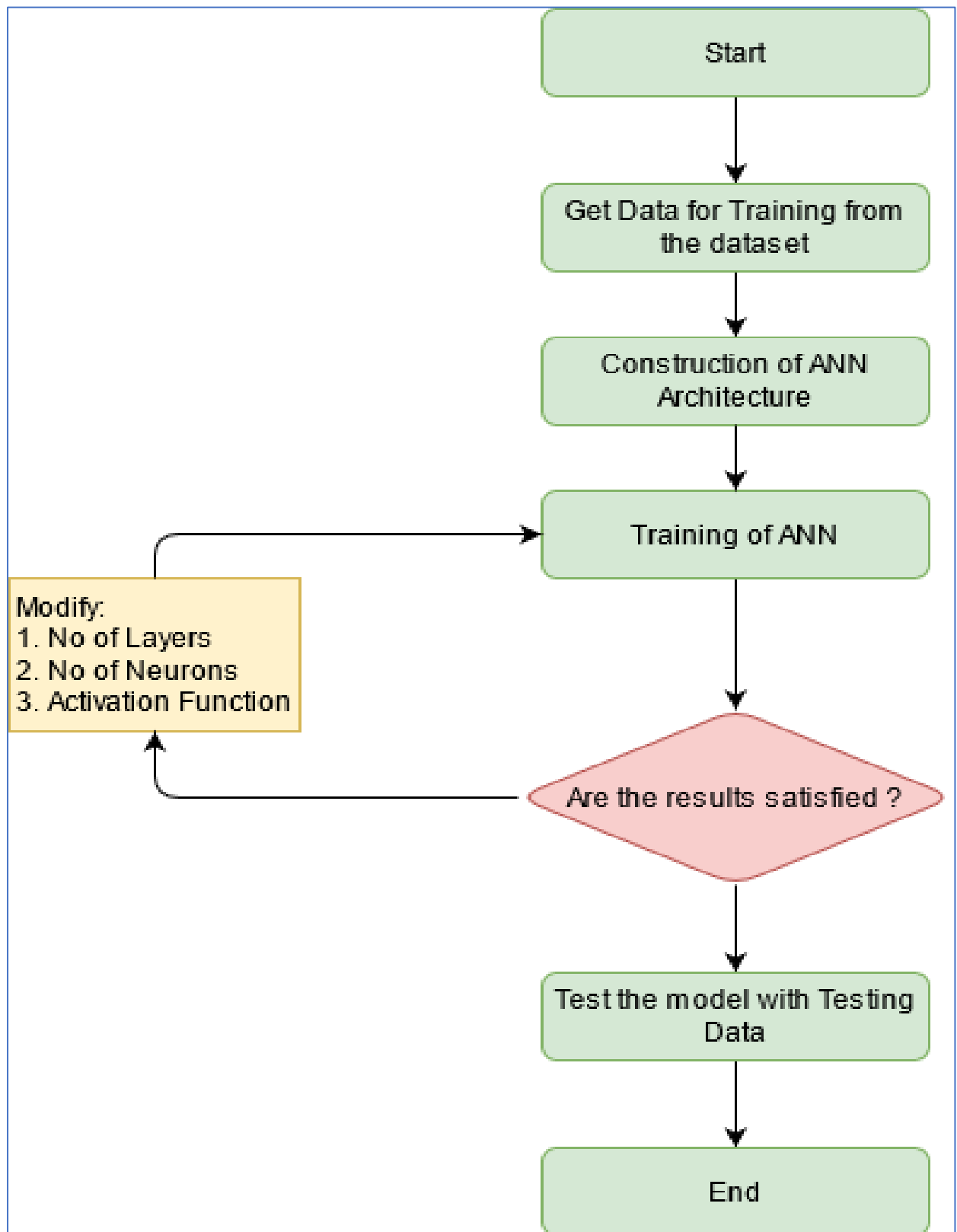
UnHealthy:**Illustration 6:** Examples of un-healthy Park Vector Images

#### 4.5. Development of ANN

- The first step in the development of the assigned project is the development of the image classifier as shown in illustration 8.
- Initial thoughts were focussed on the development of a CNN model.
- But CNNs are extremely computation driven and a SoC like RaspberryPI could not handle the computation of a complex and extremely iterative algorithm like CNN.
- Further analysis into the provided data showed the simplistic nature of the park vector graphs.
- Thus, focus shifted to a generic ANN model which are much simpler and comparatively, computationally light.
- Thus a 4 Layer deep ANN was designed as shown below in illustration 7



**Illustration 7:** Visualization of the ANN developed for image classification



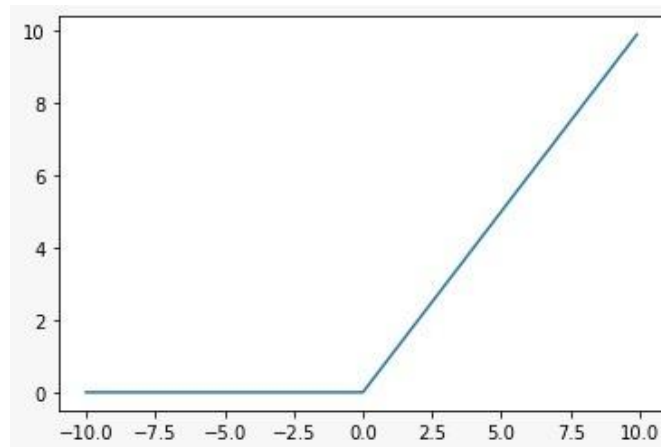
**Illustration 8:** ANN development flowchart

#### 4.5.1. Activation Functions

A combination of 2 different activation functions were used in the training of the deep learning model. These are the ReLu activation function and the sigmoid activation function. These are visualized as shown in illustration 9 and 10.

The ReLu function is defined as the following equation:

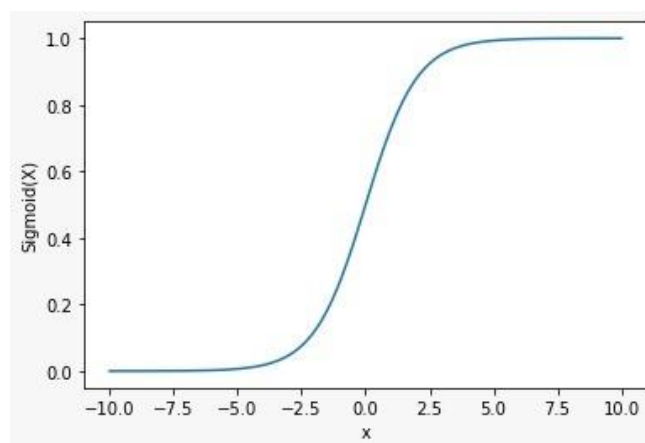
$$y = \max(0, x)$$



**Illustration 9:** ReLu activation function

The sigmoid activation function is as defined in the equation below:

$$S(x) = \frac{1}{1 + e^{-x}}$$



**Illustration 10:** Sigmoid activation function

**4.5.2. Hyper parameters:**

Number of neurons in layer 1: 8 neurons

Number of neurons in layer 1: 4 neurons

Number of neurons in layer 1: 1 neuron

Optimizer: Adam optimizer

Loss function: Binary Cross Entropy

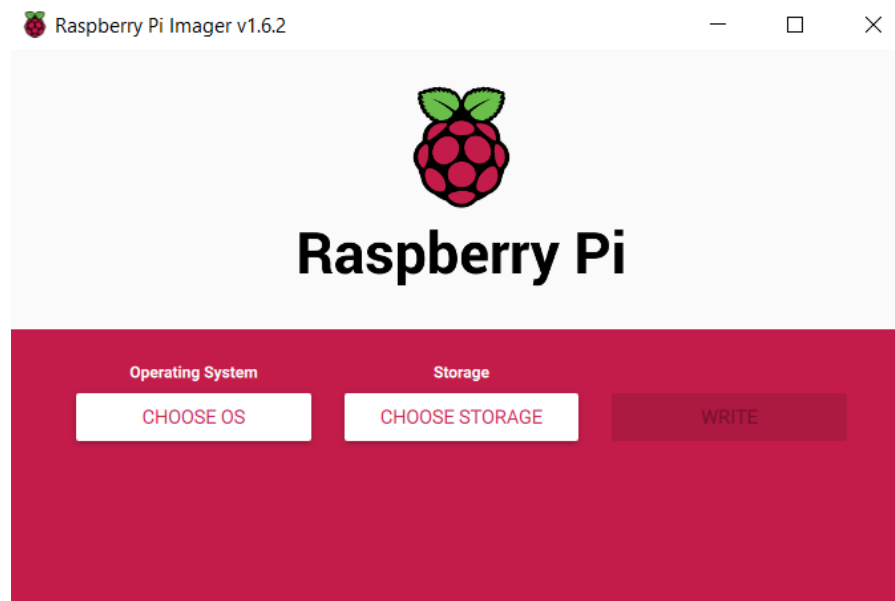
Epochs:250

## 4.6. Deployment of ANN in RaspberryPI

1. Make sure to install the 64-bit Raspbian OS from the RaspberryPi imager software from windows (custom file install). This is because, the ANN uses TensorFlow and Keras which is supported only by 64Bit operating systems.

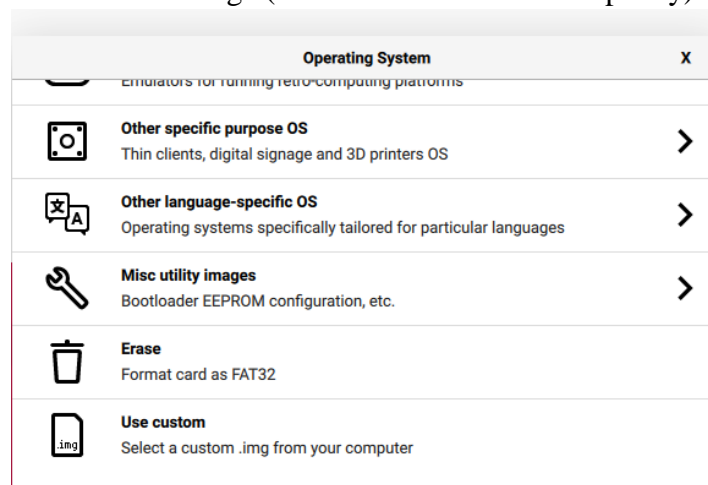
**Download 64-Bit OS from this link:**  
<https://www.raspberrypi.org/forums/viewtopic.php?t=275370>

- a. Select Choose OS



**Illustration 11:** Raspberry Pi Landing page

- b. Scroll down and select Choose Custom → Select the OS file that we downloaded from the link --> Choose storage (SD Card with 16GB > Capacity) → Write



**Illustration 12:** OS selection



2. Now Connect RaspberryPI to preferred monitor and assemble required peripherals and boot the raspberryPi.
3. Install the Python IDE of choice. For the purpose of documentation, we choose Spyder. This is done by implementing the following code in the terminal of the RPI

**sudo apt-get install spyder3**

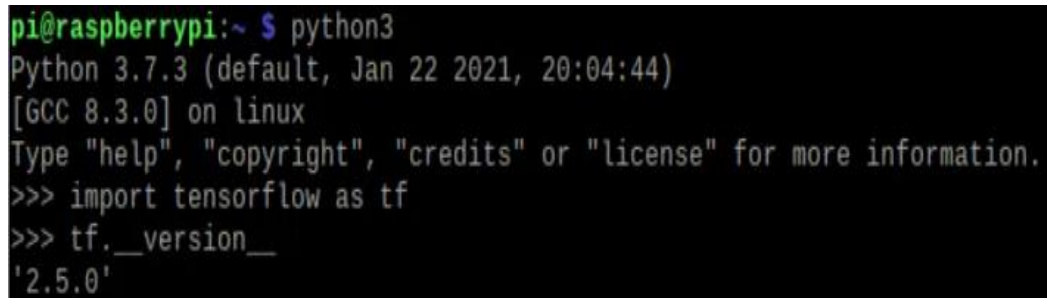
This will install the Synder IDE in RaspberryPi. This can be opened through the "Programming" tab

4. Installation of TensorFlow 2.5.x in raspberry Pi
  - a. Implement the below commands one by one in the RaspberryPI terminal (\$ represents terminal commands. Should not be typed in the terminal)

```
# get a fresh start (remember, the 64-bit OS is still under development)
$ sudo apt-get update
$ sudo apt-get upgrade
# Install pip and pip3
$ sudo apt-get install python-pip python3-pip
# Remove old versions, if not placed in a virtual environment (let pip search for them)
$ sudo pip uninstall tensorflow
$ sudo pip3 uninstall tensorflow
# utmost important: use only numpy version 1.19.5
# check the version first
$ pip3 list | grep numpy
# if not version 1.19.5, update!
$ sudo -H pip3 install numpy==1.19.5
# install the dependencies (if not already onboard)
$ sudo apt-get install gfortran
$ sudo apt-get install libhdf5-dev libc-ares-dev libeigen3-dev
$ sudo apt-get install libatlas-base-dev libopenblas-dev libblas-dev
$ sudo apt-get install liblapack-dev
# upgrade setuptools 40.8.1 -> 57.0.0
$ sudo -H pip3 install --upgrade setuptools
$ sudo -H pip3 install pybind11
$ sudo -H pip3 install Cython
# install h5py with Cython version 0.29.23 (± 15 min @1500 MHz)
$ sudo -H pip3 install h5py==3.1.0
# install gdown to download from Google drive
$ pip3 install gdown
# copy binary
```

```
$ sudo cp ~/.local/bin/gdown /usr/local/bin/gdown
# download the wheel
$gdown
https://drive.google.com/uc?id=158xXoPWOyfNswDTaapyqpREq_CBk1O_G
# install TensorFlow 2.5.0 (± 68 min @1500 MHz)
$ sudo -H pip3 install tensorflow-2.5.0-cp37-cp37m-linux_aarch64.whl
```

- b. After installation, check the version as follows.



```
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.5.0'
```

**Illustration 13:** Version check

## 5. Install Keras in raspberry PI

- a. Input the following commands in Terminal

```
$ sudo apt-get install libblas-dev
$ sudo apt-get install liblapack-dev
$ sudo apt-get install python3-dev
$ sudo apt-get install libatlas-base-dev
$ sudo apt-get install gfortran
$ sudo apt-get install python3-setuptools
$ sudo apt-get install python3-scipy
$ sudo apt-get update
$ sudo apt-get install python3-h5py
```

```
pip3 install keras
pip install keras
```

## 6. Install ScikitLearn / Sklearn

- a. Input the following commands in Terminal

```
$ sudo apt-get install python3-sklearn python3-sklearn-lib python3-sklearn-doc
```

**7. Install Matplot.lib**

- a. Input the following commands in Terminal

```
$ sudo apt update
```

```
$ sudo apt install python3-matplotlib
```

**8. Install Pandas**

- a. Input the following commands in Terminal

```
$ sudo apt-get install python-pandas
```

## V. Hardware Design Approach

### 5.1. Design architecture for measuring IM current and providing that as input to Rpi:

- This was achieved by employment of Clamp Transducers, Rogowski coils, and AI-PMUL Rogowski Integrator (Illustration 14).
- The AI-PMUL integrator reads the value provided by the Rogowski coils and maps it into a 0-5 V range, which would then be provided as an analog input to the Rpi.



**Illustration 14:** AI-PMUL accompanied with Rogowski coils

### 5.2. Design Power Source:

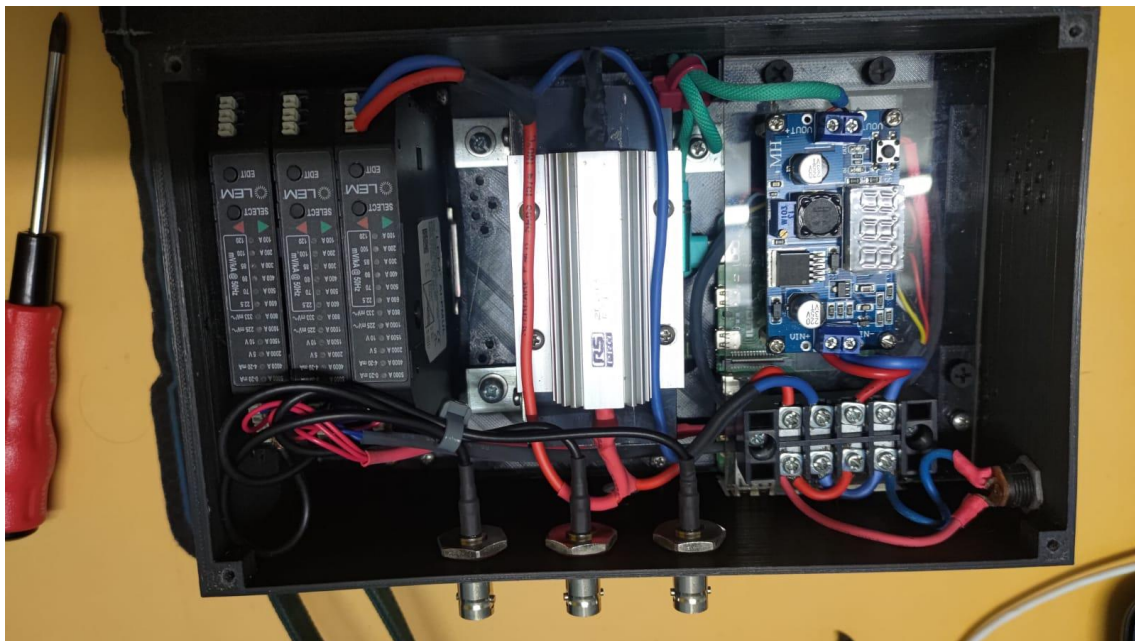
- The AI-PMUL demands a 24 V, 0.1 A power supply, whereas Rpi requires a 5 V input.
- Currently, the MOBIT is powered by a 24 V adaptor whose voltage is regulated to 5 V to power the Rpi.
- Further, the adaptor is connected to a power resistor, so as to fulfil the 0.1 A power requirement of the AI-PMUL.

### 5.3. Design of External Casing:

- An external case was 3D printed to house the electronics, along with a touch screen display for the raspberry PI



**Illustration 15:** 3D Printed case for the MOBIT



**Illustration 16:** MOBIT hardware set-up

## VI. Schedule, Tasks and Milestones

The Gantt chart portrayed in illustration 17 and 18 below shows the initial timeline designed for the development of the project. But, due to a huge success in the development of the model, the Gantt chart was later edited to dedicate more time towards the deployment of the project in a Linux system based RaspberryPI development board.

	Week 1	Week 2	Week 3	Week 4	Week 5
<i>Background study and understanding data</i>					
<i>Development of ANN model</i>					
<i>Deployment in RaspberryPI</i>					
<i>Implementing corrections and suggestions</i>					
<i>Drafting Report/Research article</i>					

**Illustration 17:** Initially planned Gantt Chart

	January	February	March	April
<i>Background study and understanding data</i>				
<i>Development and deployment of ANN and design of hardware</i>				
<i>Construction and testing of hardware</i>				
<i>Implementing corrections and suggestions</i>				
<i>Drafting Report/Research article</i>				

**Illustration 18:** Implemented Gantt Chart

## **6.1. Constraints faced:**

### **Hardware:**

- The power requirements of the AI-PMUL transducer requires an external power source, thus, making a handheld device impossible.
- Design of a common power source for both RPi and AI-PMUL

### **Software:**

- Restricted size of dataset provided by Petronas (protected by NDA).

## VII.Results and Discussions

### 7.1. ANN Accuracy:

The accuracy of the ANN model on the validation and the test set was >99.99%. This is shown in table 1 below.

F1 Score	Accuracy	Precision	Recall
100	100	100	100

**Table 1.** Testing Result Analysis of Proposed Method in terms of distinct measures

The confusion matrix for the designed model is as shown in table 2

16	0
0	20

**Table 2:** Confusion matrix

In a confusion matrix, the cell, (0,0) shows the number of true positives, (0,1) shows false positives, (1,) shows false negatives and (1,1) shows true negatives. Thus, from the above inference, we can deduce that our model has a 100% classification accuracy

### 7.2. Deployment Of ANN in RaspberryPI:

The ANN model was deployed in a RaspberryPI 4 Model B with 8GB RAM, coupled with a 16GB SD card. It was running the 64bit version of the Raspbian OS which is developed on Debian. This is illustrated in illustration 19





**Illustration 19:** RaspberryPI model B

The model was successfully deployed. Even though it was built on tensorflow and keras which is computationally intensive, the model was built to be light on the computation.

**Training time on the RaspberryPI SoC: 1.3 minutes**

**Training time on conventional desktop: 50 seconds**

**Training time on cloud compiler (colab): 36 seconds**

Even though the RaspberryPi takes a longer time to train the model as compared to other devices, the small physical footprint of the RaspberryPi allows us to compute on the edge, and also implement the proposed model in remote locations without the employment of expensive data transmission systems.

## **VIII. Summary:**

In the proposed work, we have developed an ANN (artificial neural networks) model for the various bearing fault segregation classification based on Park vector analysis of three-phase

stator currents. The experimental results indicate that the developed ANN model rendering 100% classification results for the dataset. It can be foreseen that the proposed method will enhance the reliability and accuracy of the methods used for the online detection and diagnosis of various bearing faults. Further this model was deployed on the RaspberryPI SoC thus qualifying it to be employed in industrial automation and related tasks.

## IX. References:

- [1] B. Rao, Handbook of condition monitoring: Elsevier, 1996.
  
- [2] Sukhjeet Singh, Amit Kumar, Navin Kumar,  
Motor Current Signature Analysis for Bearing Fault Detection in Mechanical Systems, Procedia Materials Science, Volume 6, 2014, Pages 171-177, ISSN 2211-8128,
  
- [3] S. Vigneshkumar, V. K. Shankar, P. N. Krishna and P. Supriya, "Fault Detection in Gearbox Using Motor Electrical Signature Analysis," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1-6, doi: 10.1109/ICCCNT.2018.8494153.
  
- [4] Miljković, Dubravko. (2015). Brief Review of Motor Current Signature Analysis. CrSNDT Journal. 5. 14-26.
  
- [5] L. Collamati, F. Filippetti, G. Franceschini, S. Pirani and C. Tassoni, "Induction machine stator fault on-line diagnosis based on LabVIEW environment," Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96), 1996, pp. 495-498 vol.1, doi: 10.1109/MELCON.1996.551587.
  
- [6] Irfan, M., Saad, N., Ibrahim, R., Asirvadam, V. S., & Alwadie, A. (2017). Analysis of distributed faults in inner and outer race of bearing via Park vector analysis method. Neural Computing and Applications. doi:10.1007/s00521-017-3038-0
  
- [7] Liu, Y.-Z., Zou, Y.-S., Jiang, Y.-L., Yu, H., & Ding, G.-F. (2020). A Novel Method for Diagnosis of Bearing Fault Using Hierarchical Multitasks Convolutional Neural Networks. Shock and Vibration, 2020, 1–14. doi:10.1155/2020/8846822

- [8] S. Zhang, S. Zhang, B. Wang and T. G. Habetler, "Deep Learning Algorithms for Bearing Fault Diagnostics—A Comprehensive Review," in *IEEE Access*, vol. 8, pp. 29857-29881, 2020, doi: 10.1109/ACCESS.2020.2972859.
- [[9] Duong, Bach Phi & Kim, Jaeyoung & Jeong, Inkyu & Im, Kichang & Kim, Cheol-Hong & Kim, Jongmyon. (2020). A Deep-Learning-Based Bearing Fault Diagnosis Using Defect Signature Wavelet Image Visualization. *Applied Sciences*. 10. 8800. 10.3390/app10248800.
- [10] Z. Zilong and Q. Wei, "Intelligent fault diagnosis of rolling bearing using one-dimensional multi-scale deep convolutional neural network based health state classification," 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, pp. 1-6, doi: 10.1109/ICNSC.2018.8361296.
- [11] Ourici, Amel & Ouari, Ahmed. (2012). Park's Vector Approach to detect an inter turn stator fault in a doubly fed induction machine by a neural network. *Computer Science & Information Technology*. 2. 10.5121/csit.2012.2503.
- [12] Vilhekar, Tushar & Ballal, Makarand & Suryawanshi, Hiralal. (2017). Application of multiple parks vector approach for detection of multiple faults in induction motors. *Journal of Power Electronics*. 17. 972-982. 10.6113/JPE.2017.17.4.972.
- [13] Pires, V. & Kadivonga, Manuel & Martins, J.F. & Pires, A. J.. (2013). Motor square current signature analysis for induction motor rotor diagnosis. *Measurement*. 46. 942–948. 10.1016/j.measurement.2012.10.008.
- [14] Mohd Faridz Mohd Yunoh, Shahrum Abdullah and S. S. K. Singh, "Artificial Neural Network Classification for Fatigue Feature Extraction Parameters Based on Road Surface Response," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, pp. 1480-1485, 2018. [Online]. Available: <http://dx.doi.org/10.18517/ijaseit.8.4-2.6805>.

[15] Sobie, C., Freitas, C., & Nicolai, M. (2018). Simulation-driven machine learning: Bearing fault classification. *Mechanical Systems and Signal Processing*, 99, 403–419. doi:10.1016/j.ymssp.2017.06.025

## X. Appendix A:

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[4]:
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as img
```

```
import numpy as np
```

```
import pandas as pd
```

```
#import seaborn as sns
```

```
import tensorflow as tf
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
#from tqdm import tqdm
```

```
import os
```

```
from sklearn.utils import shuffle
```

```
from sklearn.model_selection import train_test_split
```

```
from tensorflow.keras.applications import EfficientNetB0
```

```
from tensorflow.keras.models import Sequential,Model
```

```
from tensorflow.keras.applications.vgg16 import preprocess_input,VGG16
```

```
from tensorflow.keras.layers import
```

```
MaxPooling2D,Conv2D,Dense,BatchNormalization,Dropout,GlobalAveragePooling2D,F
```

```
latten,Input
```

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,  
TensorBoard, ModelCheckpoint
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
#from tensorflow.keras.utils.vis_utils import plot_model
```

```
#import ipywidgets as widgets
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
#import io
```

```
from PIL import Image
```

```
from IPython.display import display, clear_output
```

```
#import cv2
```

```
#from warnings import filterwarnings
```

```
# from glob import glob
```

```
# In[6]:
```

```
len(os.listdir('/home/pi/Desktop/UTP'))
```

```
# In[7]:
```

```
# Prepere data
```

```
healthy = os.listdir('/home/pi/Desktop/UTP/Healthy')
```

```
unhealthy = os.listdir('/home/pi/Desktop/UTP/Unhealthy')
```

```
# In[9]:
```

```
# Prepere input data
```

```
X_data =[]
```

```
for file in healthy:
```

```
    image=img.imread("/home/pi/Desktop/UTP/Healthy/"+file)
```

```
    #img = cv2.imread('/home/pi/Desktop/UTP/Healthy'+file)
```

```
    #face = cv2.resize(img, (224, 224) )
```

```
    #(b, g, r)=cv2.split(face)
```

```
    #img=cv2.merge([r,g,b])
```

```
    X_data.append(image)
```

```
for file in unhealthy:
```

```
    image=img.imread("/home/pi/Desktop/UTP/Unhealthy/"+file)
```

```
    #img = cv2.imread('/home/pi/Desktop/UTP/Unhealthy'+file)
```

```
    #face = cv2.resize(img, (224, 224) )
```

```
    #(b, g, r)=cv2.split(face)
```

```
    #img=cv2.merge([r,g,b])
```

```
    X_data.append(image)
```

```
# In[10]:
```

```
X = np.squeeze(X_data)
```



X.shape

# In[11]:

#show one training sample

from matplotlib import pyplot as plt

plt.imshow(X[5], interpolation='nearest')

plt.show()

# In[12]:

# normalize data

X = X.astype('float32')

X /= 255

# In[13]:

# Prepare outputs:

target\_yes=np.full(len(healthy),1)

target\_no=np.full(len(unhealthy),0)

Y=np.concatenate([target\_yes,target\_no])

Y

# In[14]:

```
len(Y)
```

```
# In[15]:
```

```
# creation of x_train, y_train, x_test, y_test arrays
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
number_of_train = X_train.shape[0]
```

```
number_of_test = X_test.shape[0]
```

```
print('number_of_train:', number_of_train)
```

```
print('number_of_test:', number_of_test)
```

```
# In[16]:
```

```
# Modify input and output for processing in Keras
```

```
X_train = X_train.reshape(number_of_train,X_train.shape[1]*X_train.shape[2]*3)
```

```
X_test = X_test.reshape(number_of_test,X_test.shape[1]*X_test.shape[2]*3)
```

```
print("X train flatten",X_train.shape)
```

```
print("X test flatten",X_test.shape)
```

```
# In[17]:
```

```
# Evaluating the ANN
```

```
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier

from sklearn.model_selection import cross_val_score

from tensorflow.keras.models import Sequential # initialize neural network library

from tensorflow.keras.layers import Dense # build our layers library

def build_classifier():

    classifier = Sequential() # initialize neural network

    classifier.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu',
input_dim = X_train.shape[1]))

    classifier.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu'))

    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])

    return classifier

classifier = KerasClassifier(build_fn = build_classifier, epochs = 250)

accuracies = cross_val_score(estimator = classifier, X = X_train, y = Y_train, cv = 3)

mean = accuracies.mean()

variance = accuracies.std()

print("Accuracy mean: "+ str(mean))

print("Accuracy variance: "+ str(variance))

# In[18]:

# Predictions on Test Datasets using ANN model

classifier.fit(X_train, Y_train)

y_pred = classifier.predict(X_test)
```

```
y_pred = y_pred>0.5
```

```
# In[19]:
```

```
X0 = np.reshape(X_test[0], (525, 700, 3))
```

```
X1 = np.reshape(X_test[1], (525, 700, 3))
```

```
X2 = np.reshape(X_test[2], (525, 700, 3))
```

```
X3 = np.reshape(X_test[3], (525, 700, 3))
```

```
# In[20]:
```

```
print(y_pred)
```

```
op=[]
```

```
for i in range(0,4):
```

```
    if y_pred[i] == 1:
```

```
        op.append("Healthy")
```

```
    else:
```

```
        op.append("UnHealthy")
```

```
print(op)
```

```
# In[21]:
```

```
plt.imshow(X0)
```

```
print ("Prediction:", op[0])
```

```
# In[22]:
```

```
plt.imshow(X1)
```

```
print ("Prediction:", op[1])
```

```
# In[23]:
```

```
plt.imshow(X2)
```

```
print ("Prediction:", op[2])
```

```
# In[24]:
```

```
plt.imshow(X3)
```

```
print ("Prediction:", op[3])
```