# INDORE

**INTERNSHIP REPORT 2023- 24**



*A report submitted in partial fulfilment of the requirements for the Award of Degree of*

#### BACHELOR OF TECHNOLOGY

**in**

***Computer Science Engineering***

#### by

**Mr.Pavas Bhatnagar**

**EN20CS301292**

**Under the Supervision of**

**Smt.Bhavna N. Merh**

**SOG , ACSD**

**Raja Ramanna Centre for Advanced**

**Technology (RRCAT) , Indore**

**(Duration: 12th June 2023 to 12th August 2023)**

***Department of Computer Science Engineering***

# MEDI-CAPS UNIVERSITY

AB Rd, Pigdamber, Rau, Indore, Madhya Pradesh 453331



**2022 – 2023**

**DECLARATION**

I hereby declare that work, which is being presented in the Internship Report as the partial fulfilment for the award of degree of **Bachelor of Technology** in **Computer Science Engineering** in the **Department of Computer Science Engineering** at **Medi-Caps University**, **Indore**, is an authentic record of my work carried out under the Mentorship of Industry Mentor Smt. Bhavna N. Merh **, SOG, ACSD** and The matter embodied in this internship report has not been submitted for the award of any degree.

Enrollment Number: EN20CS301292 Student Signature Date:



**2022 – 2023**

**INTERNSHIP APPROVAL SHEET**

This is to certify that **Mr. Pavas Bhatnagar** Enrollment Number EN20CS301292 has successfully completed his industrial internship starting from 12th June to 12th August 2023 and has submitted the final report. His work has been found satisfactory and it is recommended to accept it as a partial fulfilment for the award of degree of **Bachelor of Technology, Computer Science Engineering** of the **Department of Computer Science Engineering** at **Medi-Caps University**, **Indore.**

|  |  |
| --- | --- |
| **Internal Examiner**  Date: | **External Examiner**  Date: |



**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

# Acknowledgement

First, I would like to extend my sincere gratitude towards my **Sh. R.K. Agrawal, Head, Software & Embedded Systems Section,ACSD** of RRCAT for giving me an opportunity at their esteemed organization. Industry Mentor of RRCAT for giving me an opportunity at their esteemed organization.

Secondly , I would like to thank **Smt. Bhavna N. Merh, Scientific Officer ‘G’, ACSD and Dr. Surendra Yadav, Scientific Officer ‘G’** who helped me and guided me in my work at RRCAT. With their patience and openness, they created an enjoyable working environment. It is indeed with a great sense of pleasure and immense sense of gratitude, that I acknowledge the help of these individuals.

**Mr. Abhisek Nayak** deserves special gratitude for his invaluable guidance and support, which were crucial to the successful completion of my work. Without hisgenerous help and expert advice, I would have struggled to accomplish my goals.

*Name: Pavas Bhatnagar*

*Enrolment Number: EN20CS301292 Branch: Computer Science Engineering*

**EXECUTIVE SUMMARY**

I have Completed my internship at RRCAT. The domain of my internship was Machine Learning which is most trending technology in today’s IT field. After joining this internship. I have highlighted the experience and skills that we gained, challenges that we faced and actions taken in solving the problem during our training.

The project that I worked upon was “**Mapping of Magnet Power Supply behaviour during Indus 2 Beam Energy Ramping Operation”**.

This project mainly aims to create a Mapping function of the Magnet power supply during the ramping operation. The project involves the designing of the architecture of Feedfoward Neural Network using python programming language and its machine learning libraries mainly Tensorflow and Keras.

\

**TABLE OF CONTENT**

Declaration I Internship Approval Sheet II Certificate III

Acknowledgement V Executive Summary VI

**CHAPTER 1 − A Brief introduction of the organization’s Research areas--- 1**

1.1 Brief Introduction to RRCAT 1

#### CHAPTER 2 − Overview of the Organization 2- 6

* 1. A brief history 2
  2. Research Area 2-4
  3. Competitors \_ 5
  4. Brief summary of all the departments 5-6

#### CHAPTER 3 – Plan of Internship Program 7-8

* 1. Brief introduction of the branch and department 7
  2. Starting and Ending date of the Internship 7
  3. Names of the department in which you obtained Training 7

#### CHAPTER 4 – Brief Overview on Technologies and Tools 8-10

* 1. Tools 8
  2. Hardware specification 10

[CHAPTER 5 – Problem Identification and Solution 11](#_TOC_250004)

5.1 Problem Identification 11 5.2 Consequences of the current problem 11

5.3 Solutions 11

[CHAPTER 6 – Introdcution 12-1](#_TOC_250003)4

* 1. [Whats are neural networks 1](#_TOC_250002)2
  2. [Feedforward 1](#_TOC_250001)2
  3. Backpropagation 12
  4. Model Architecture 14

CHAPTER 7 – Conclusion 15-16

[CHAPTER 8 - References and Sources Used 1](#_TOC_250000)7

**Appendices 18-23**

# A Brief Introduction of the Organization’s Research areas



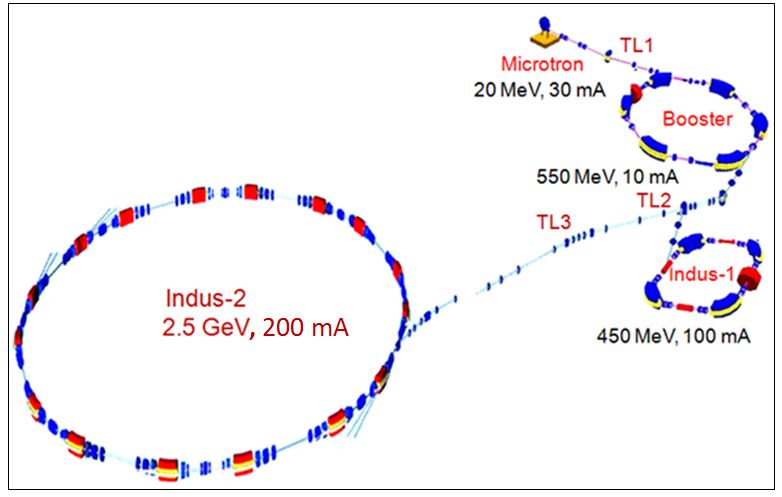
## 1.1 Brief Introduction to RRCAT

A R&D institute of India's Department of Atomic Energy, the **Raja Ramanna Centre for Advanced Technology** conducts research and development in the non-nuclear front-line fields of lasers, particle accelerators, and associated technologies.

The Indian Department of Atomic Energy founded RRCAT to increase activity at the Bhabha Atomic Research Centre (BARC), Mumbai, in two cutting-edge scientific and technological fields: Lasers and Accelerators.

### Accelerator

RRCAT has developed and commissioned two synchrotron radiation sources, Indus-1 and Indus-2, which serve as a national facility. In addition, the centre is pursuing various other accelerator activities, including the development of a high energy proton accelerator for a spallation neutron source, electron accelerators for food irradiation and industrial applications, and advanced technologies such as superconducting radio-frequency cavities and micromodules.

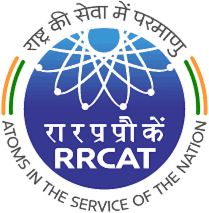


# Chapter 2 Overview of the Organization



## Brief History

Gyani Zail Singh, the then-president of India, laid the centre’s cornerstone on February 19, 1984. In May 1984, work on the laboratories and homes began. The first group of scientists from the Mumbai-based Bhabha Atomic Research Centre (BARC) relocated to RRCAT in June 1986, and scientific work got underway. Since that time, the centre has quickly developed into a top research and development facility for lasers, accelerators, and their applications. Originally known as the Centre for Advanced Technology, it was renamed Raja Ramanna Centre for Advanced Technology in December 2005 in honour of eminent Indian Physicist Raja Ramanna.



#### Figure 2.1 RRCAT Logo

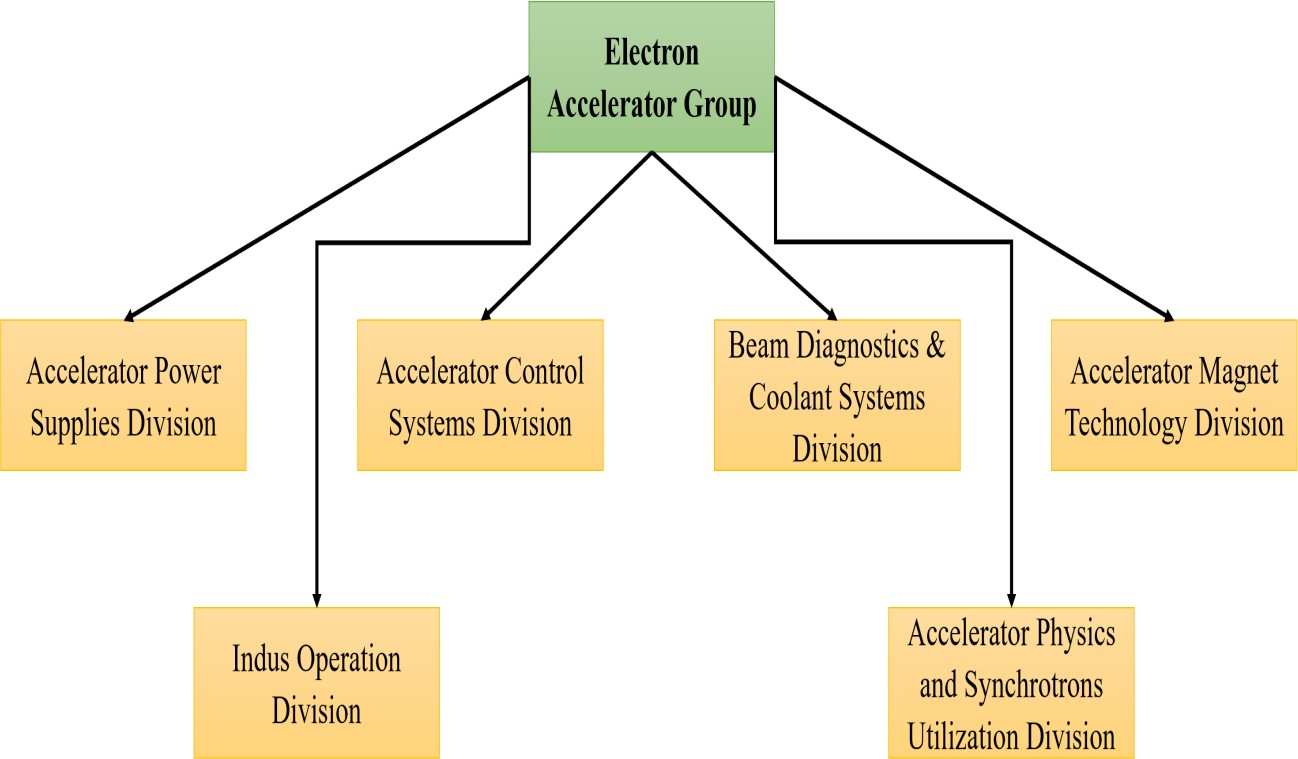
RRCAT motto is “Photons in the Service of Nation” and true to its motto, RRCAT has always inspired the students and young researchers to learn about sources, behaviours and applications of photons (light particles).

## Internal organization structure of RRCAT

The Raja Ramanna Centre for Advanced Technology (RRCAT) is a premier research and development institution located in India. It is involved in a wide range of research activities, mainly focused on advanced technologies related to accelerators, lasers, and materials science.

RRCAT has several groups dedicated to accelerator technology, including the Proton Accelerator Group, Electron Accelerator Group, and Industrial Accelerators Division. These groups work on developing and improving particle accelerators, which are used in a variety of fields including medicine, industry, and basic research.

In addition to accelerator and laser technology, RRCAT has a Materials Science and Advanced Technology Group that works on developing and characterizing new materials for various applications.



#### Figure 2.2 Introduction of the Departments Under Which I have done Internship

During my internship with the Electron Accelerator Group at RRCAT, I worked in the Accelerator Control Systems Division. My work involved development of application based on machine learning algorithms.

**Accelerator Control Systems Division**

The Accelerator Control Systems Division is responsible for the design, development, commissioning, maintenance, and upgrades of control systems for various particle accelerator projects. This involves implementing complex control systems for subsystems such as RF, vacuum, and MPS, which have diverse function requirements and are distributed over large areas with a high I/O count.

The division has expertise in various areas such as data acquisition, instrumentation, testing and synchronization, data logging, machine interlocks, and software applications using technologies such as VME bus, embedded systems, Lab View, EPICS, WinCCOA, Java, and Visual C++. The division has successfully implemented control systems for various projects, including the Microtron Injector, transfer lines, and Indus-1 and Indus-2 storage rings.

### Projects

The activities span development, implementations and operation of control systems of accelerator for R & D activities and industrial applications. The scope encompasses:

* + 1. Indus-1 SRS Control System
       - Indus-1 SRS Subsystems
    2. Indus-2 SRS Control System
       - Indus-2 SRS Subsystems
    3. Control System for Electron Linac & Electron Beam Radiation Processing Facility (EBRPF)
    4. Control System for Test Stands for Super Conducting Cavity Testing
       - Vertical Test Stand (VTS)
       - Horizontal Test Stand (HTS)
    5. Control System for Proton Linac Sub-systems

### Developments

Accelerator Controls System Division (ACSD) is involved in development of control systems for various particle accelerators. These control systems are heterogeneous, distributed and involve large number of I/OS. These systems require state of art technology and continuous upgradations. Accordingly, a large number of indigenous developments, and R & D activities are taken which are finally deployed in actual systems.

1. Hardware Developments
   * VME Bus Based Boards
     + I/O Boards
     + CPU Boards
     + Timing Boards
     + Communication Boards
   * Embedded Control Modules (ECMs)
     + FPGA with Controller Based ECM (For Indus-1 RF Control System)
     + FPGA Based ECM (For Microtron Control System)
     + DSP Based ECM (For Indus-1 MPS Control System)
     + Programmable Control Module (For Linac Scanning Magnet Power Supply)
   * Beam current display and distribution system
2. Software Developments
3. VLSI Based Developments

## Brief Summary of all the Departments

Raja Ramanna Centre for Advanced Technology (RRCAT) is a premier research and development institution located in Indore, India. The centre is involved in cutting-edge research in areas such as accelerator and laser technologies, materials science, and related fields. RRCAT has several departments, including:

* + 1. Materials Science and Advanced Technology Group
    2. Proton Accelerator Group
    3. Electron Accelerator Group
    4. Laser Group
    5. Technology Development and Support Group

These departments are further subdivided into various divisions, such as the Laser Materials Processing Division, RF Systems Division, Accelerator Power Supplies Division, and more. Each division is responsible for carrying out specific research and development activities related to their respective areas of expertise. RRCAT diverse range of departments and

divisions allows it to undertake multidisciplinary projects and advance scientific knowledge in a wide range of fields.

# Chapter 3 Plan of Internship Program



## A brief introduction of the branch or department where you performed your internship

During my internship with the Electron Accelerator Group's Accelerator Control Systems Division (ACSD) at RRCAT, I had the opportunity to gain hands-on experience in developing the Neural Networks.

## Starting and Ending dates of Internship

The internship at Raja Ramanna Center for Advanced Technology (RRCAT), Indore, was for a duration of 8 weeks from 12th June to 12nd August, 2023.

## The names of the departments in which you obtained training and the duration of your training in these departments

I have completed my internship with the Electron Accelerator Group's Accelerator Control Systems Division (ACSD) at RRCAT. Accelerator Control Systems Division is involved in design, development, commissioning, maintenance and up-gradation of control systems for various particle accelerators housed in Indus complex. The Indus control system is heterogeneous, distributed and involves large number of I/Os.The system requires state of art technology and continuous up-gradations.

Accordingly, VME based systems are ingeniously developed to work at various layers of control system architecture. Similarly, software developments viz. SCADA based, Web Based, Databases etc. have been done which are finally deployed in actual systems.

# Chapter 4 Brief Overview on Technologies and Tools



## Tools

### Visual Studio Code:

Visual Studio Code (also known as VS Code) is a tool. Generally, most machine learning projects are developed as ‘.ipynb’ in Jupyter notebook or Google Collaboratory. However, Visual Studio Code is powerful among programming code editors, and also possesses the facility to run ML or Data Science codes. VS Code is available as a free and open- source tool and can be used on Windows, macOS, and Linux operating systems.

The Visual Studio Code’s marketplace is full of extensions for basically all programming purposes, whether for auto-completing code snippets or enhancing the readability of the code, it contains a variety of options or services as extensions.

### Google Tensorflow

TensorFlow is one of the most popular open-source libraries used to train and build both machine learning and deep learning models. It offers a powerful library, tools, and resources for numerical computation, specifically for large scale machine learning and deep learning projects. It enables data scientists/ML developers to build and deploy machine learning applications efficiently. For training and building the ML models, TensorFlow provides a high-level Keras API, which lets users easily start with TensorFlow and machine learning.

### Keras

Keras is the high-level API of the TensorFlow platform. It provides an approachable, highly-productive interface for solving machine learning (ML) problems. It provides complete framework to create any type of neural networks. Keras is easy to learn and it supports simple neural network to very large and complex neural network model. It basically uses the Sequential model for creating the neural network where it is the linear composition of Keras layers. It is easy, minimal and has abitilty to represent nearly all available neural networks

### NumPy

NumPy (Numerical Python) is an open source Python library that’s used in almost every field of science and engineering. It’s the universal standard for working with numerical data in Python, The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

It contains multidimensional array and matrix data structures. It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays.

### 

### Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library (histograms, scatter plots, bar charts, etc) for Python and its numerical extension NumPy. Matplotlib is a multi-platform data visualization library built on NumPy arrays. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.

### Scikit-learn (sklearn)

It is an open source library. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library is largely written in Python and is built upon NumPy, SciPy and Matplotlib.

### Pandas

Pandas is built on top of two core Python libraries—[matplotlib](https://mode.com/python-tutorial/libraries/matplotlib/) for data visualization and [NumPy](https://mode.com/python-tutorial/libraries/numpy/) for mathematical operations. Pandas acts as a wrapper over these libraries, allowing you to access many of matplotlib's and NumPy's methods with less code. For instance, pandas' .plot() combines multiple matplotlib methods into a single method, enabling you to plot a chart in a few lines.

## Hardware Specifications

**Processor (CPU)**: I was using Intel® Core(TM) i5-4460 which is the 4th generation processor and have turbo boost to 3.4Ghz, the processor had 4 cores and 4 threads.

**Random Access Memory (RAM)**: The RAM installed in the Desktop was 8 GB DDR3 , which was suitable for the multitasking.

**Storage**: The device had 1TB capacity of Hard Disk Drives (HDDs) which had three partitions namely C, D and F

**Graphics Processing Unit (GPU)**: The device had Intel® UD Graphics 4600 which was more than enough for my project work as it didn’t require any graphical processing.

**Operating System**: The operating system installed in the PC was Windows 10 updated version.

**Internet Connectivity**: High Speed Internet was used for my research purpose.

.

# Chapter 5 Problem Identification and Solution

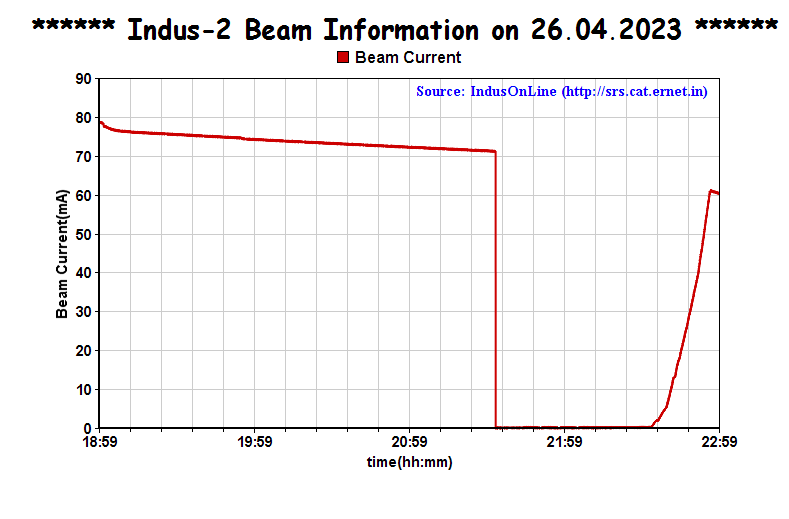
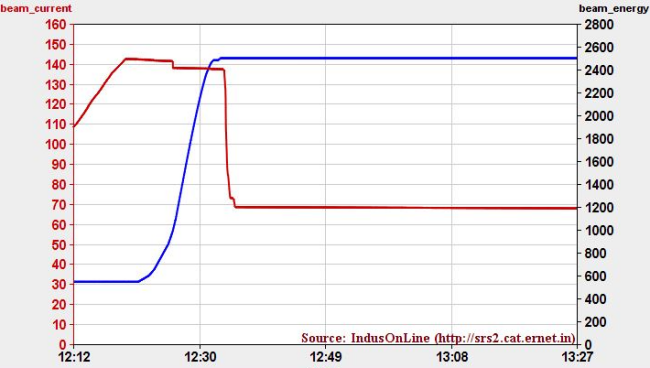


## Problem Identification

Indus-2 synchrotron radiation source operates 24x7 to deliver synchrotron beam to users. The beam lifetime is defined as the time required by beam current to reach 1/e of its original value with a standard exponential decay. Beam faults occur as sudden drops of beam current and are of two types. Partial loss (Fig 1.(a)) which is when the beam current drops by more than 2 mA from its previous value and the other is, Complete loss (Fig 1. (b)) which happens when the beam current suddenly drops and the beam is killed by the accelerator subsystems hence current effectively goes to 0 mA. Both faults have to be avoided.

## Consequences of the current problem

Once the beam is lost, Indus-2 beam filling has to be done again with energy ramped to to 2.5 Ge V. This ramping process takes time and quite a lot of energy.



(a) (b)

    Figure 1: Partial loss (a) and Complete loss (b) w.r.t beam current deviation in time

These beam faults actually happen due to anomalies or faults which occur in subsystems that make up the accelerator like RF system, Vacuum system, Magnet systems, etc. Most of the times it has been noted that a fault in MPS can directly affect a beam and can be then held responsible for beam faults. So an expert naturally first looks at MPS faults to eliminate them from the picture but this is also tedious since there are 117 individual supplies each with its own unique signatures, operational ranges, cycles etc. Currently operators use web based applications to check raised error flags and find each and every faulty MPS. On top of this, Indus-2 data lake has collected terabytes of raw operational data over many years of successful run. We can use this huge mine of data to develop and understand the various patterns and characteristics of these systems.

## Solution

In this study, we suggest a deep learning feedforward Neural Network model to map various MPS currents in-order to predict the beam loss or find out quickly if any powers supply current has deviated. The dipole power supply current is mapped to various steering magnet coils currents. The proposed method can predict the currents of steering coils as a function of dipole and quadrupole power supply currents, thus finding deviations in settings given to the machine. This will allow the

accelerator's scientists to delve deeper into correlations between MPS and beam and gain valuable insights.

Our proposed solution helps the scientists to gain quick and powerful insights by analyzing large amounts of operational data collected in the Indus-2 data lake and also serves as a litmus test to see how capable AI systems can be for such applications. We leverage the power of AI to train high accuracy deep learning models to detect faults which are otherwise difficult to detect by conventional methods like template matching or normal logic based computer programming, since these methods already expect end user to have complete understanding of underlying complex patterns and correlations which is not always possible when handling data at large scale. We want to make it easy to analyse large amounts of data rapidly and reliably.

# Chapter 6

# Introduction



**What are Neural Networks?**

Neural networks are artificial systems that were inspired by biological neural networks that works like neurons works in human brain. These systems learn to perform tasks by being exposed to various input datasets and examples without any task-specific rules.

The idea is that the system generates identifying characteristics from the data that they have been passed without being programmed with a pre-programmed understanding of the input datasets. Neural networks are based on computational models for threshold logic. Threshold logic is a combination of algorithms and maths. Neural networks are based either on the study of the brain or on the application of neural networks to artificial intelligence.

**Feedforward**

The process of receiving an input to produce some kind of output to make some kind of prediction is known as Feed Forward

There can be multiple hidden layers which depend on what kind of data you are dealing with. The number of hidden layers is known as the depth of the neural network. The deep neural network can learn from more functions. Input layer first provides the neural network with data and the output layer then make predictions on that data which is based on a series of functions. ReLU Function is the most commonly used activation function in the deep neural network.

**Backpropagation**

The [Backpropagation](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "backpropagation) neural network is a [multilayered](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "multi-layer Perceptron), [feedforward](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "feedforward) neural network and is by far the most commonly used. It is also considered one of the simplest and most general methods used for [supervised training](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "supervised learning) of multilayered NN. Backpropagation works by approximating the non-linear relationship between the [input](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "input) and the [output](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "output) by adjusting the [weight](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "weight) values internally. It can further be generalized for the input that is not included in the [training patterns](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "training patterns) (predictive abilities).

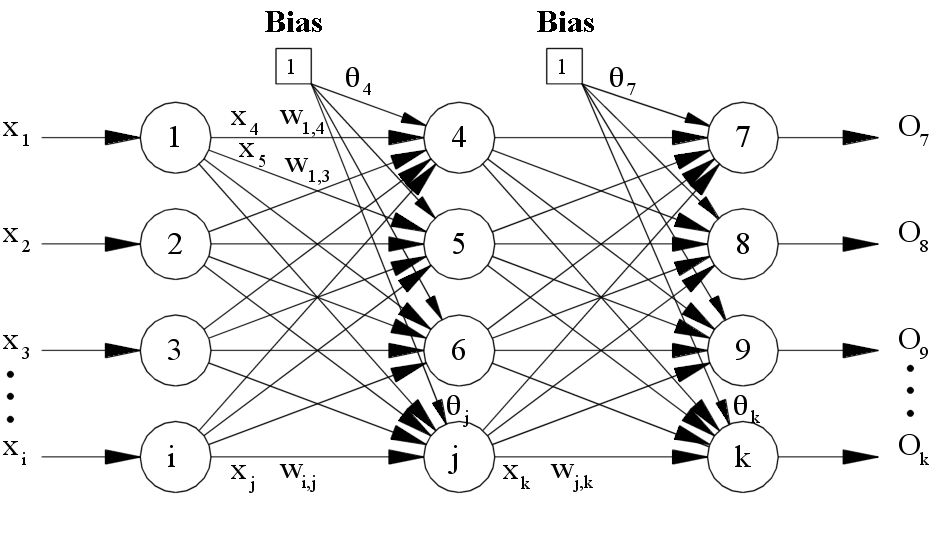
Generally, the Backpropagation network has two stages, training and [testing](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "testing). During the training phase, the network is "given" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of that person.

For a single training example, Backpropagation algorithm calculates the gradient of the error function. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a group of methods used to efficiently to train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates and remembers the updated weight to improve the network until it is not able to perform the task for which it was being trained. Derivatives of the activation function to be known at network design time is required to use Backpropagation algorithm.

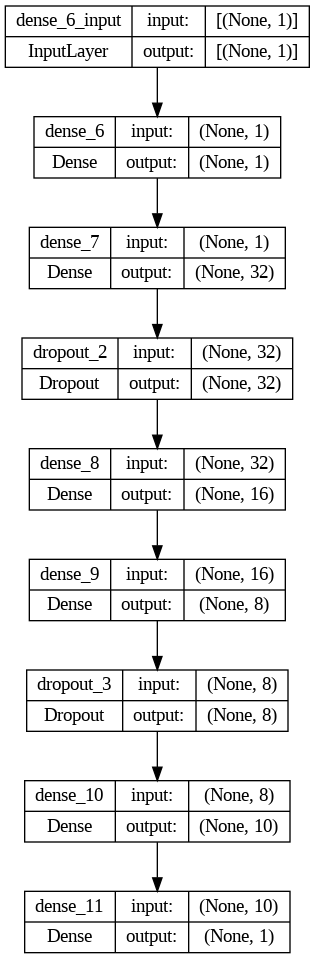
More on encoding information - a neural network, as most [learning algorithms](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "learning algorithm), needs to have the inputs and outputs encoded according to the user defined scheme. The scheme will define the network architecture so that once a network is trained, then the scheme cannot be changed without creating a totally new network. Similarly there are many types of encoding the network response.

The following [figure](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/BackPropagation.html" \l "Figure 5) shows the how the Backpropagation works in a neural network, that includes and [input layer](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "input layer), one [hidden layer](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "hidden layer) and an [output layer](https://www.cse.unsw.edu.au/~cs9417ml/MLP2/Glossary.html" \l "output layer). It should be noted that Backpropagation neural networks can have more than one hidden layers.



**Model Architecture**

The keras function API will help us in creating models that are more flexible than models created using the sequential API. The functional API will work with a model that has a nonlinear topology, sharing layers and handling multiple outputs and inputs. The deep learning model has multiple layers and is directed. The keras functional API helps us in the creation of graph layers



# Chapter 7 Conclusion



The internship was an amazing experience all around. The knowledge I've gained from this internship will be very beneficial as my career develops. Besides the technical aspects, the most significant lessons I took away from this internship were time management, the value of working hard to succeed in any field, critical thinking, and communication skills.

As a newcomer to the field, I had to develop the habit of working on projects while sitting down for 6 to 8 hours at a time, which included attending meetings, writing code, having discussions, and much more. Additionally, this internship taught me more about current technological trends and my ability to pick them up quickly and easily. I also learned about the many technologies and techniques the company employs to advance and adapt to the environment.

I learned a lot about the software industry, its culture, the workplace, and everything related to software development thanks to the training programme. I have gained the confidence to fix the situation by holding onto hope by attending team meetings with the team members, leader, and project manager. The internship training also taught me that you can't just build things by thinking about the requirements of the moment; you also need to keep the project flexible so that the developer's team can easily modify or improve it without interfering with earlier work.

I firstly build a neural network from scratch that means I made the neural network without using the libraries (keras and tensorflow). The code included mainly 8 files, which was used to build the neural network, this code mainly followed the steps that are:

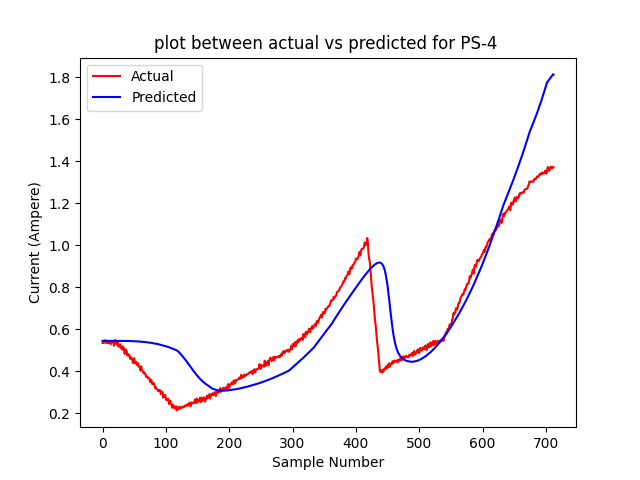
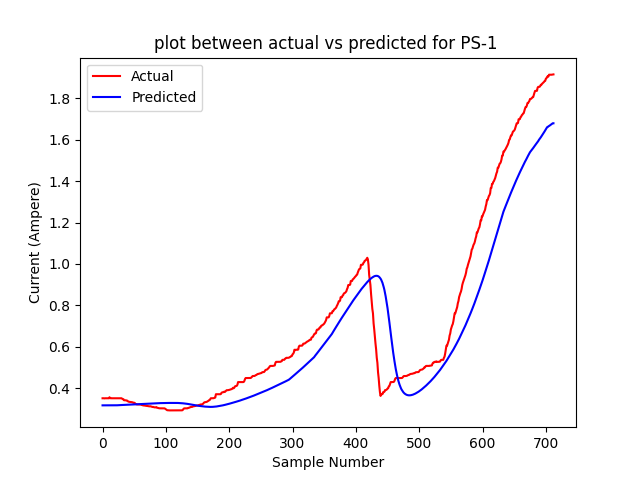
1. Initialization of the network
2. Forward Propagation
3. Finding the error of Backpropagation
4. Training of the network
5. Prediction

Then my main focus was on how to build a neural network using the libraries like keras.

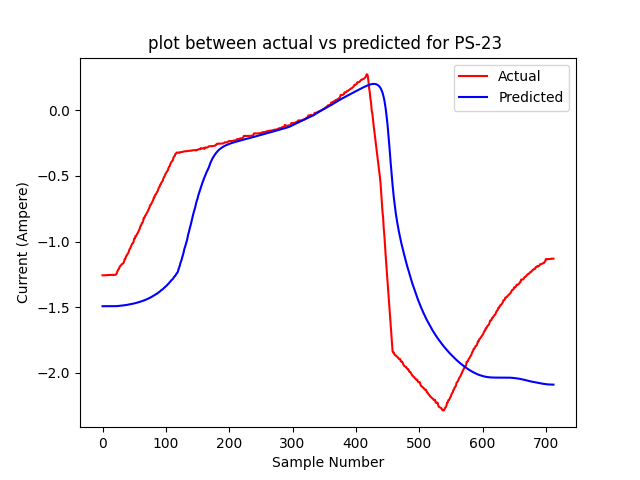
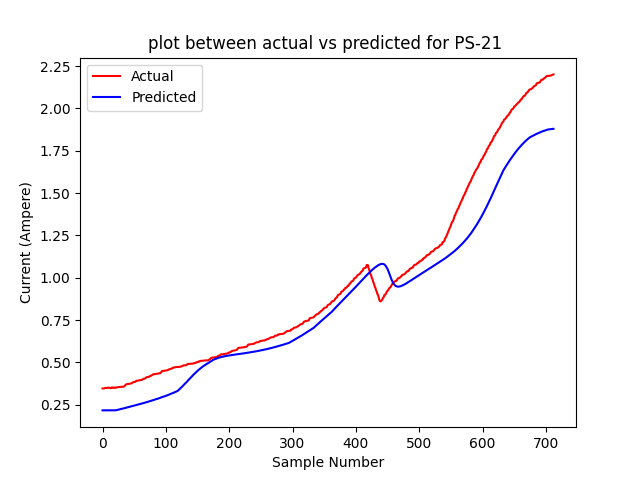
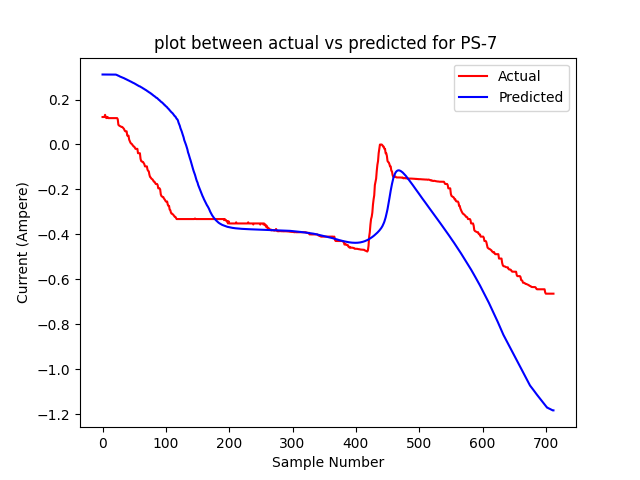
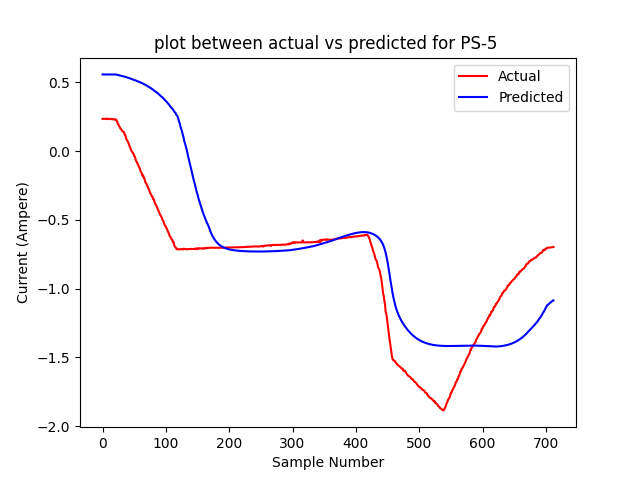
I researched everything from the keras documentation and Machine learning mastery.

After that applied this knowledge on the Indus data that was given by my mentor and made the architecture using Tan hyperbolic activation function. Finally,made a 2 hidden layered feedforward backpropagation neural network which was basically a multi single-input and multi-output model

These mainly were the plots of prediction of the 5 power supplies



# 



# Chapter 8

# References and Sources Used



1. *Keras: The high-level API for TensorFlow. https://www.tensorflow.org/guide/keras*
2. *Pandas: <https://mode.com/python-tutorial/libraries/pandas/>*
3. *10 minutes to pandas https://pandas.pydata.org/docs/user\_guide/10min.html*
4. *NumPy: the absolute basics for beginners. https://numpy.org/doc/stable/user/absolute\_beginners.html*
5. *Scikit Learn Tutorial : <https://www.tutorialspoint.com/scikit_learn/index.htm>*
6. *Machine Learning Mastery :<https://machinelearningmastery.com/>*
7. *Stack Overflow : <https://stackoverflow.com/>*
8. **Company Website:** *Atomic Energy of India Government, D. O. (n.d.). Raja Ramanna Centre for Advanced Technology. Raja Ramanna Centre for Advanced Technology. <https://www.rrcat.gov.in/>*

# Appendices



**Code for the main architecture of Neural Network (saving model by model)**

import tensorflow as tf

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt

import time

from pickle import dump

start\_time = time.time()

scaler=MinMaxScaler()

data = pd.read\_csv(r'C:\Users\Acer\Desktop\pavas\_indus\DATA\combined-csv-files.csv');

inputs = data.iloc[:,0:1]

print(inputs)

output=[]

for output\_column in range(27, 115):

    outputs = data.iloc[:, output\_column:output\_column+1]

    outputs = scaler.fit\_transform(outputs)

    output.append(outputs)

    print(outputs)

#scaler = StandardScaler()

scaled\_inputs = scaler.fit\_transform(inputs)

#scaled\_outputs = scaler.fit\_transform(outputs)

dump(scaler, open('scaler.pkl', 'wb'))

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(1, activation='tanh', input\_shape=(1,)))

model.add(tf.keras.layers.Dense(32, activation='tanh'))

#model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Dense(10, activation='tanh'))

#model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Dense(1))

#model.add(Dense(500, input\_dim=27, activation='sigmoid'))

#model.add(Dense(88, activation='tanh'))

#early\_stopping = tf.keras.callbacks.EarlyStopping(

    #monitor='val\_loss',  patience=10, restore\_best\_weights=True)

sgd\_optimizer=tf.keras.optimizers.SGD(learning\_rate=0.01, momentum=0.9)

model.compile(optimizer=sgd\_optimizer, loss='mse')

for i in range (0,1):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled\_inputs, output[i], test\_size=0.2, random\_state=42)

    model.fit(X\_train, y\_train, epochs=100, batch\_size=16, validation\_split=0.2)

#es = EarlyStopping(monitor='val\_loss', mode    ='min', verbose=1, patience=10)

#model.fit(X\_train, y\_train,validation\_data=(X\_test, y\_test), epochs=200, batch\_size=32, callbacks=[es])

    y\_test = np.array(y\_test)

    loss = model.evaluate(X\_test, y\_test)

#scaled\_predictions = model.predict(X\_test)

    predictions = model.predict(X\_test)

#predictions = scaler.inverse\_transform(scaled\_predictions)

#predictions = scaler.inverse\_transform(predictions)

    print("Predictions:")

    print(predictions)

    print(y\_test)

    print(predictions.shape)

    print("\nActual Outputs:")

    print(y\_test)

    print(y\_train.shape)

model\_path=r'C:\Users\Acer\Desktop\pavas indus\models\model\_column\_{}.keras'.format(100)

    model.save(model\_path)

    print("Model for Output Column {} saved.".format(i))

'''plt.plot(y\_test)

plt.plot(predictions)

plt.legend(["y\_test", "predictions"], loc ="lower right")

plt.show()'''

#print("\nAccuracy: {:.2f}%".format(accuracy \* 100))

print("All models saved")

print("--- %s seconds ---" % (time.time() - start\_time))

**Code for prediction of new test data**

import tensorflow as tf

import pandas as pd

import numpy as np

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

import csv

from pickle import load

new\_data= pd.read\_csv

(r'C:\Users\Acer\Desktop\dummydata\mps\_PL\_6jun23.csv')

new\_inputs = new\_data.iloc[:, 5:6]

scaler = load(open('scaler.pkl', 'rb'))

scaled\_new\_inputs = scaler.fit\_transform(new\_inputs)

diff=[]

for i in range(0, 88):

model=tf.keras.models.load\_model(r'C:\Users\Acer\Desktop\pavas indus\models\model\_column\_{}.keras'.format(i))

    #print("model loaded successfully")

    new\_predictions = model.predict(scaled\_new\_inputs)

# Print the predictions

    print("New Predictions:")

    print(new\_predictions)

    #print(new\_predictions.shape)

predictions\_df = pd.DataFrame(new\_predictions)

predictions\_df.to\_csv(r'C:\Users\Acer\Desktop\pavas

indus\outputs\output\_{}.csv'.format(i)

,index=False,header=None)

    print(r"File saved\_{}".format(i))

    predictions\_df = pd.read\_csv(r'C:\Users\Acer\Desktop\pavas indus\outputs\output\_{}.csv'.format(i),header=None)

real\_output\_df=pd.read\_csv(r'C:\Users\Acer\Desktop\final\_pavas\mps\_PL\_6jun23.csv',header=None).iloc[:, 34+i:34+i+1]

    scaler.inverse\_transform(predictions\_df)

    temp=[]

    for i in range(0,743):

        temp.append(real\_output\_df.iat[i,0]-predictions\_df.iat[i,0])

    diff.append(temp)

with open(r"C:\Users\Acer\Desktop\pavas indus\error1.csv","w") as my\_csv:

    newarray = csv.writer(my\_csv,delimiter=',')

    newarray.writerows(diff)

print("Error file saved")

#error\_df = real\_output\_df.subtract(predictions\_df)

#print(error\_df)

**Code to find predicted values above threshold value**

import csv

def find(file\_path, threshold):

    row\_column\_indices = []

    with open(file\_path, newline='') as csvfile:

        reader = csv.reader(csvfile)

        for row\_idx, row in enumerate(reader):

            for col\_idx, value in enumerate(row):

                if float(value) > threshold:

                    row\_column\_indices.append((row\_idx + 1, col\_idx + 1))

    return row\_column\_indices

if \_\_name\_\_ == "\_\_main\_\_":

    csv\_file\_path = r'C:\Users\Acer\Desktop\pavas indus\error1.csv'

    threshold\_value = 0.5

    result = find(csv\_file\_path, threshold\_value)

    if result:

        output\_filename = r'C:\Users\Acer\Desktop\pavas indus\new2.txt'

        with open(output\_filename, "w") as output\_file:

            output\_file.write("Cells with values greater than 0.5 are:\n")

            for row\_idx, col\_idx in result:

                output\_file.write(f"Row: {row\_idx}, Column: {col\_idx}\n")

        print("Output saved")

    else:

        print("No cells found.")

**Code for the main architecture of Neural Network (saving only one model)**

import tensorflow as tf

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

data = pd.read\_csv(r'C:\Users\Acer\Desktop\pavas\_indus\DATA\combined-csv-files.csv');

inputs = data.iloc[:,0:1]

outputs = data.iloc[ :,27:]

#print(outputs)

scaler = StandardScaler()

scaled\_inputs = scaler.fit\_transform(inputs)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled\_inputs, outputs, test\_size=0.2, random\_state=42)

model = tf.keras.models.Sequential([

    tf.keras.layers.Dense(64, activation='tanh', input\_shape=(1,)),

    tf.keras.layers.Dense(32, activation='tanh'),

    tf.keras.layers.Dense(32, activation='tanh'),

    tf.keras.layers.Dense(88)

])

#learning\_rate = 0.001

#optimizer = tf.keras.optimizers.SGD(learning\_rate)

adam\_optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001)

model.compile(optimizer=adam\_optimizer, loss='mse' , metrics='accuracy')

#model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

y\_test = np.array(y\_test)

loss, accuracy = model.evaluate(X\_test, y\_test)

# Make predictions

#scaled\_predictions = model.predict(X\_test)

predictions = model.predict(X\_test)

#predictions = scaler.inverse\_transform(scaled\_predictions)

print("Predictions:")

print(predictions)

print(predictions.shape)

print("\nActual Outputs:")

print(y\_test)

print(y\_train.shape)

model\_path = r'C:\Users\Acer\Desktop\final\_pavas\model100.keras'

model.save(model\_path)

#print("\nAccuracy: {:.2f}%".format(accuracy \* 100))

model = tf.keras.models.load\_model

(r'C:\Users\Acer\Desktop\final\_pavas\model100.keras')

new\_data = pd.read\_csv(r'C:\Users\Acer\Desktop\new\_output\mps\_PL\_5aug23.csv',header=None)

new\_inputs = new\_data.iloc[:, 4:5]

print(new\_inputs)

scaled\_new\_inputs = scaler.fit\_transform(new\_inputs)

#print(new\_inputs)

predictions = model.predict(scaled\_new\_inputs)

# Print the predictions

print("New Predictions:")

print(predictions)

print(predictions.shape)

predictions\_df = pd.DataFrame(predictions)

predictions\_df.to\_csv(r'C:\Users\Acer\Desktop\final\_pavas\9aug\9aug.csv', index=False,header=None)

print("File saved.")

#error\_df = real\_output\_df - predictions\_df.iloc[:, 0:88]

#error\_df.to\_csv('path/to/error\_output.csv', index=False)

#real\_output\_df = pd.read\_csv(r'C:\Users\Acer\Desktop\final\_pavas\mps\_PL\_6jun23.csv', usecols=range(33, 123))

# Load the second CSV file (predicted output)

#predicted\_output\_df = pd.read\_csv(r'C:\Users\Acer\Desktop\final\_pavas\output.csv', usecols=range(0, 88))

#error\_df = real\_output\_df.subtract(predicted\_output\_df, axis=0)

#error\_df.to\_csv(r'C:\Users\Acer\Desktop\final\_pavas\error.csv', index=False)

#print("Error saved and calculated")

old\_output = pd.read\_csv(r'C:\Users\Acer\Desktop\new\_output\mps\_PL\_5aug23.csv',header=None)

new\_output = pd.read\_csv(r'C:\Users\Acer\Desktop\final\_pavas\9aug\9aug.csv',header=None)

plt.xlabel('Sample Number')

plt.ylabel('Current (Ampere)')

for i in range (0,88):

    plt.plot(range(713), old\_output.iloc[:713,33+i:34+i].to\_numpy(), color="red", linestyle="solid")

plt.plot(range(713), new\_output.iloc[:713,0+i:1+i].to\_numpy(), color="blue", linestyle="solid")

plt.xlabel('Sample Number')

plt.ylabel('Current (Ampere)')

plt.legend(['Actual','Predicted'])

    plt.title(f"plot between actual vs predicted for PS-{i+1} ")

    plt.show()