National College of Ireland

Configuration Manual

MSc Research Project
Research Project/Internship

Abiodun Ali
Student ID: 19209347

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ..... Abiodun Ali .....

Student ID: .....19209347.....

Programme: ..... MSCCYBE..... **Year:** 2021-2022

Module: ..... Research Project/Internship
.....

Lecturer: ..... Vikas Sahni .....
Submission Due Date: ..... August 15, 2022.....

Project Title: ... Anomalies Detecting network intrusion using a software-defined network and Deep Learning.

Word Count: ..... **Page Count:** .....13.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....Abiodun Ali.....

Date: .....14/08/2022.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Abiodun Ali
Student ID: 19209347

1 Introduction

The steps and processes taken in the development of this project for an SDN based Deep Learning approach to combat cyber-threats in networks is presented in this Configuration Manual. It describes all necessary settings and software tools needed to replicate the experimental setup for the project.

2 System Specification

The system configuration used in the project are:

- Operating System: Windows 11
- Processor: Intel Core i7 7th Gen
- GPU: 1070 GTX
- SSD:256
- Hard Drive: 1TB
- RAM: 16GB.

3 Software Tools

Some of the software tools used to implement this project are:

- Python
- Spyder
- Excel
- Pandas
- Numby
- Keras
- Tensorflow

3.1 Python Libraries

A software library consists of the collection of pre-written codes which have been utilized by the developers for resolving the common tasks of programming. The programmers utilize the python libraries as well as frameworks for reducing the development time. Python contains a wide range of set of in-built libraries including Matplotlib, NumPy, TensorFlow, Seaborn, Pandas etc. various ML libraries which are utilized in the implementation of the proposed scheme include TensorFlow, SciKitLearn, Keras, and Pandas.

3.1.1 TensorFlow

This is an open-source machine learning framework used to carry out high performance numerical computations. It offers great architectural support, allowing for simple computation deployment over a wide range of platforms, from edge devices to mobiles, servers, and desktops. It carries out high level tasks needed to build advanced neural network models and provides flexibility such that functionalities for your model can be defined. TensorFlow was used to build the SDN based NTD model.

3.1.2 Keras

It is a Python-based deep learning (DL) API which executes over the top of TensorFlow machine learning (ML) platform. The major goal of the developing Keras is enabling the fast experimentation, and it lessens the number of the essential actions of the user for typical use cases. It deals with consistent as well as simple APIs and provides an easy way to execute the neural networks as well as makes deep learning (DL) easily accessible that aids the developers learn the complicated features sequentially from the input data. It has been utilized as an API for the TensorFlow to construct the SDN based NTD model.

3.1.3 SKLearn

This python library has been utilized in the implementation of the machine learning (ML) models for classification, clustering, regression, and statistical tools to analyse them. It contains the functionalities for the inbuilt datasets, feature extraction, assembling techniques, feature selection, as well as dimensionality reduction. This has been utilized for building K-NN, Naïve Bayes, and Logistic Regression classifier models.

3.1.4 Numpy

It is a python library that has been utilized for working with arrays and functions for operating the domain of linear algebra, matrices, as well as Fourier transform. It was created by Travis Oliphant in 2005 which is an open-source project, and it can be used freely. It stands for Numerical Python.

3.1.5 Matplotlib

A low-level graph plotting python library which functions as a visualization utility and John D. Hunter created it. It is an open-source project and could be used freely which is mostly written in python, few of the segments are written in C, JavaScript, and Objective-C for the Platform compatibility.

3.1.6 OS and Pandas

The OS module in Python performs the functionality for creation and removal of directory (folder), gathering its contents, changing as well as identifying the current directory, and so on. The OS module must be firstly imported for interacting with underlying operating system (OS). Pandas which is a python library for the analysis of the data and was created in 2008 by Wes McKinney out of a requirement for a scalable and powerful quantitative analysis tool. Pandas has become one of the most popular libraries of python which contains an extremely active contributors' community.

4 Software Installation

To install TensorFlow, it is important to have “Python” installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation.

Consider the following steps to install TensorFlow in Windows operating system.

Step 1 − Verify the python version being installed.

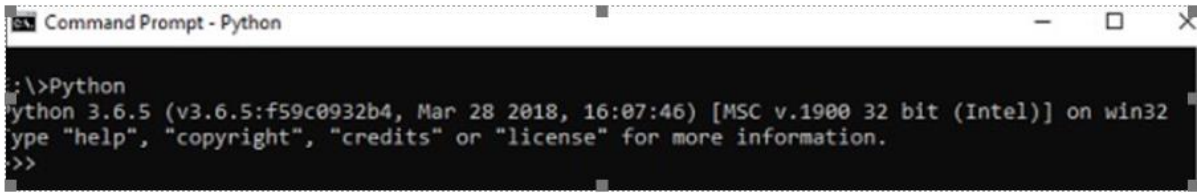

Fig 4.1: Verify of python version

Step 2 − A user can pick up any mechanism to install TensorFlow in the system. Recommend “pip” and “Anaconda”. Pip is a command used for executing and installing modules in Python.

Before install TensorFlow, need to install Anaconda framework in our system.

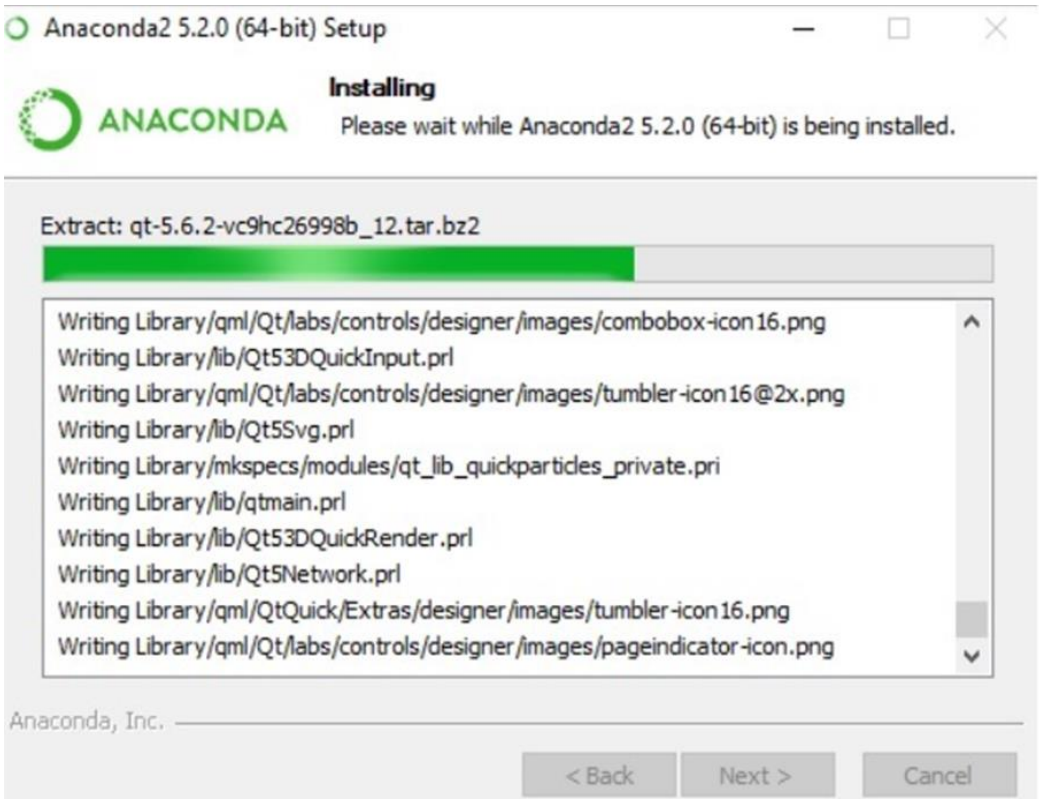

Fig 4.2: install Anaconda framework

After successful installation, check in command prompt through “conda” command. The execution of command is displayed below –

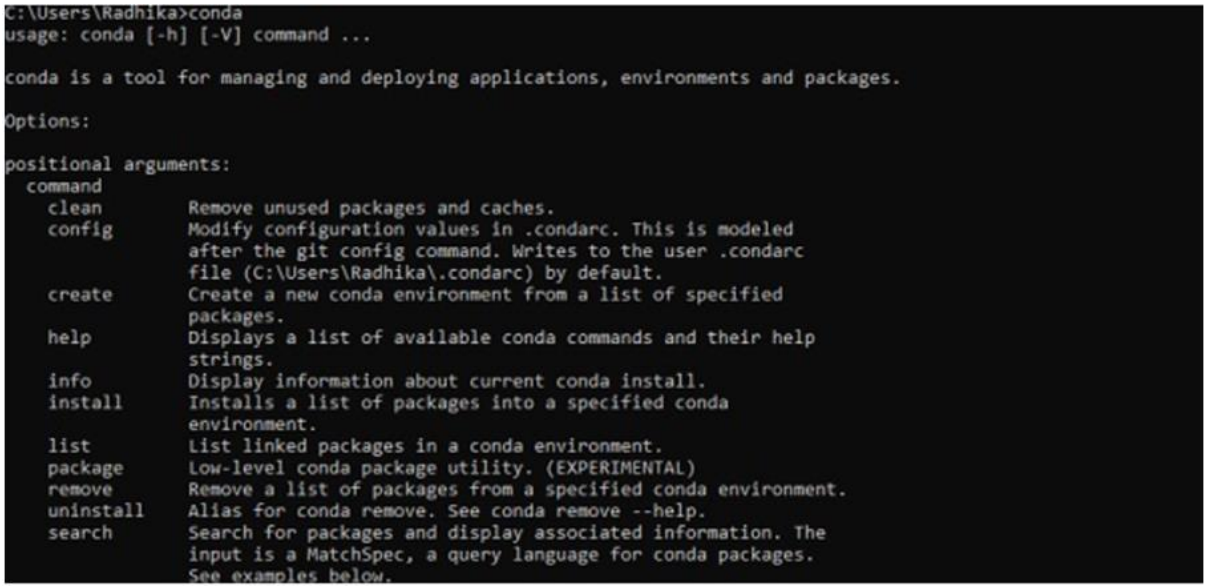

Fig 4.3: Confirmation of conda

Step 3 − Execute the following command to initialize the installation of TensorFlow −
conda create --name tensorflow python = 3.5

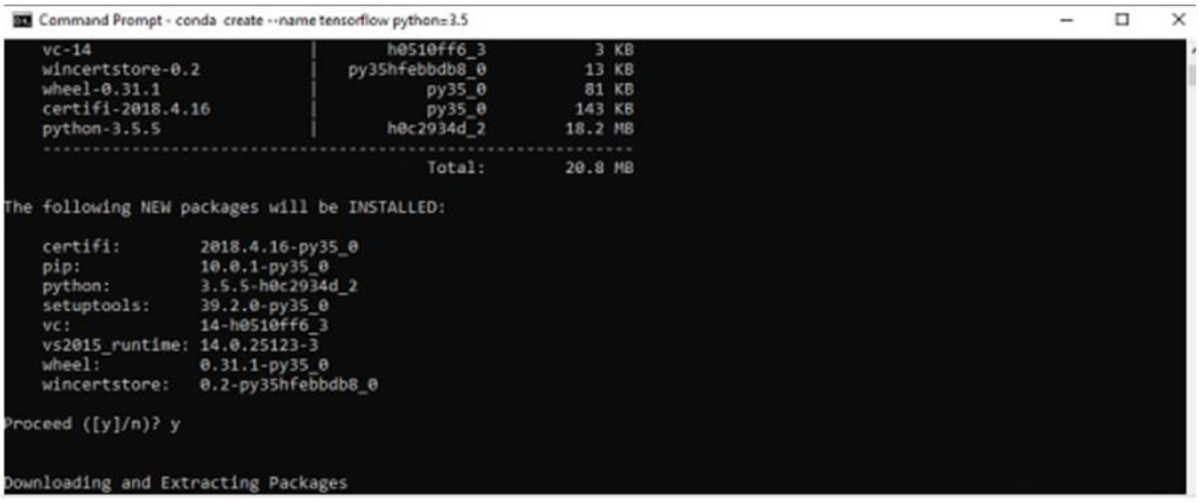

Fig 4.4: installation of TensorFlow

It downloads the necessary packages needed for TensorFlow setup.
Step 4 − After successful environmental setup, it is important to activate TensorFlow module.
activate tensorflow

Step 5 − Use pip to install “Tensorflow” in the system. The command used for installation is mentioned as below −
pip install tensorflow
And,
pip install tensorflow-gpu

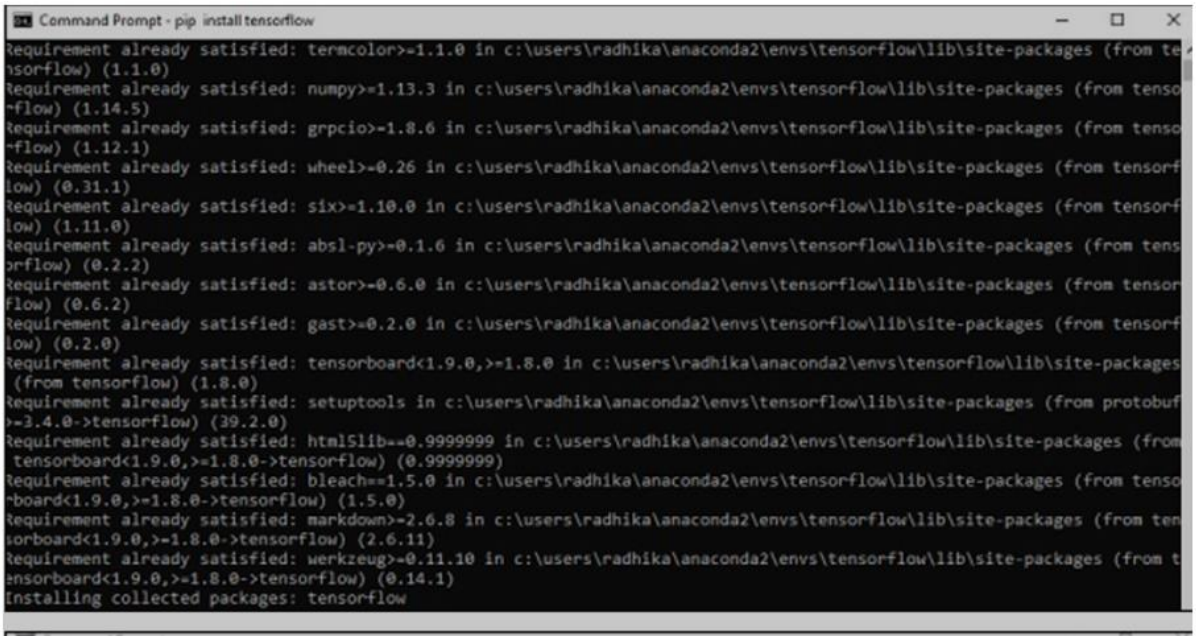

Fig 4.5: Installation of tensorflow-gpu

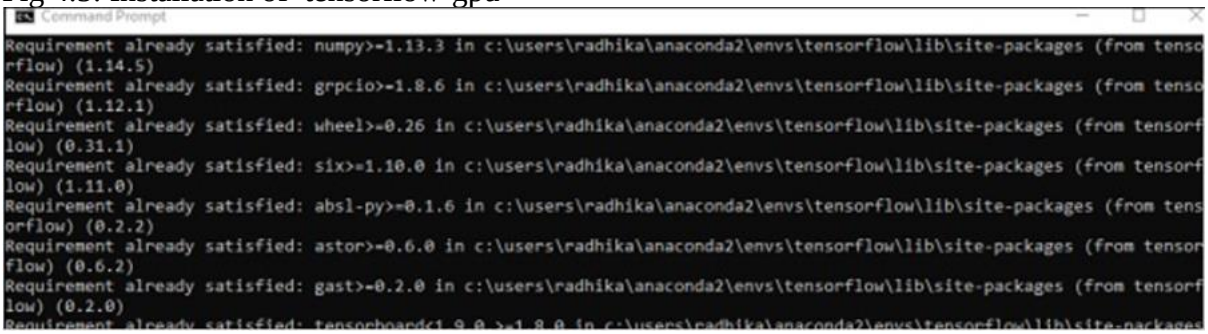

Fig 4.6: Installation of tensorflow-gpu

After successful installation, it is important to know the sample program execution of TensorFlow.
Following example helps us understand the basic program creation “Hello World” in TensorFlow.

Fig 4.7: Creation “Hello World” in TensorFlow.

The code for first program implementation is mentioned below −
>> activate tensorflow

>> python (activating python shell)
>> import tensorflow as tf
>> hello = tf.constant('Hello, Tensorflow!')
>> sess = tf.Session()
>> print(sess.run(hello))

5 Implementation:

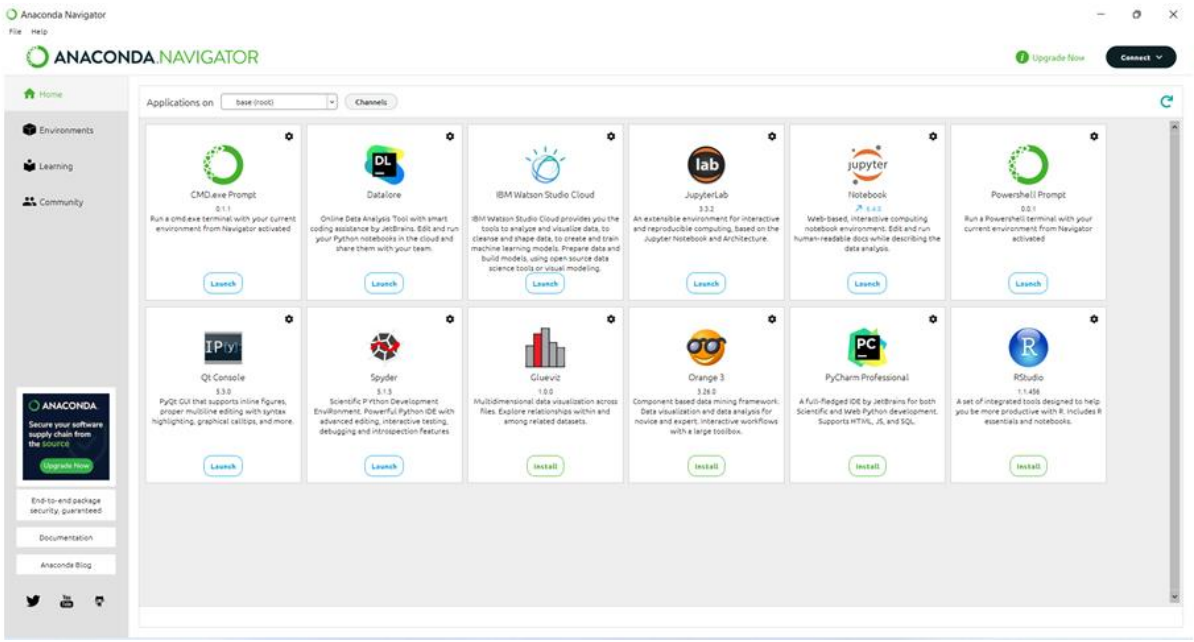

Fig 5.1: Anaconda Navigation Phase

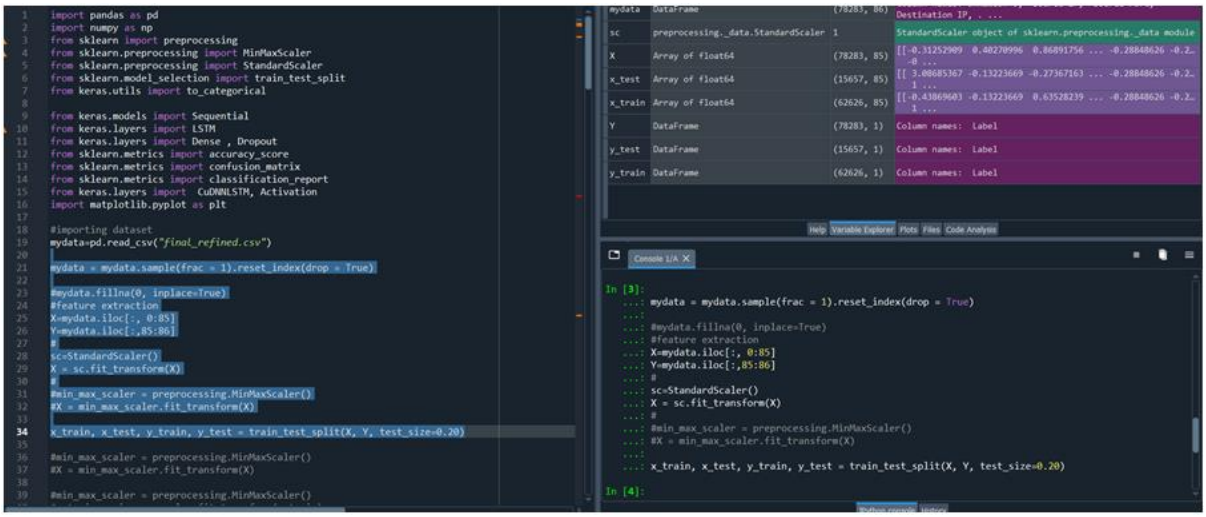

Fig 5.2: Importation of the libraries

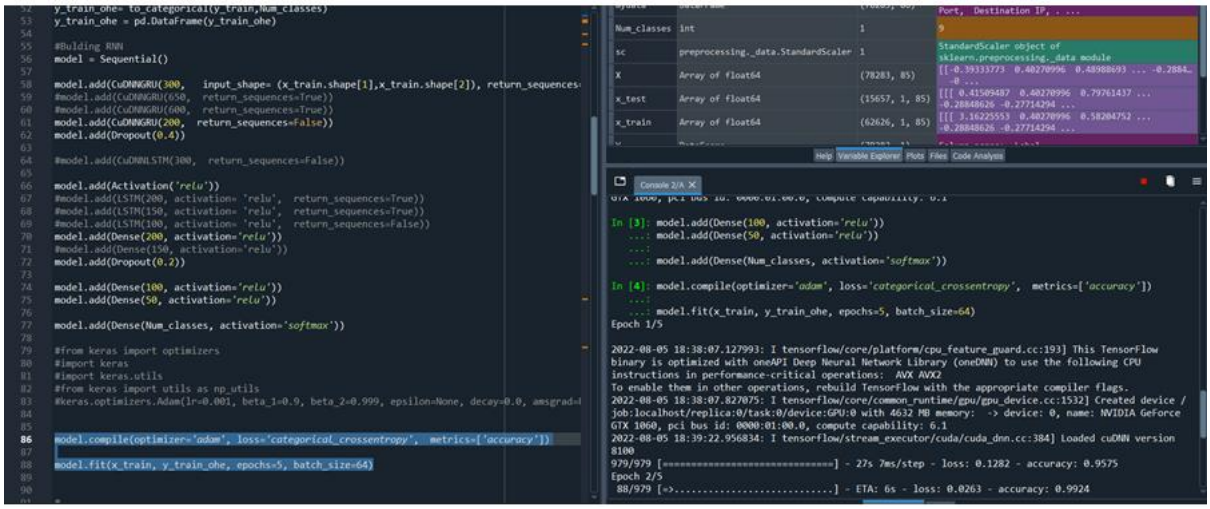

Fig 5.3: Importation of the libraries

6 GRU

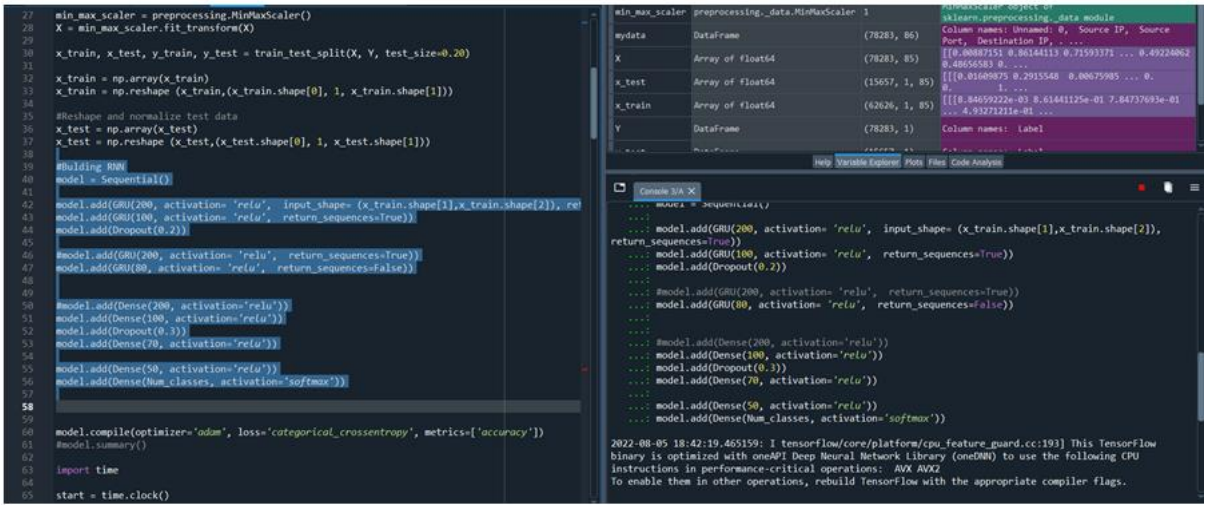

Fig 6.1: Implementation of GRU

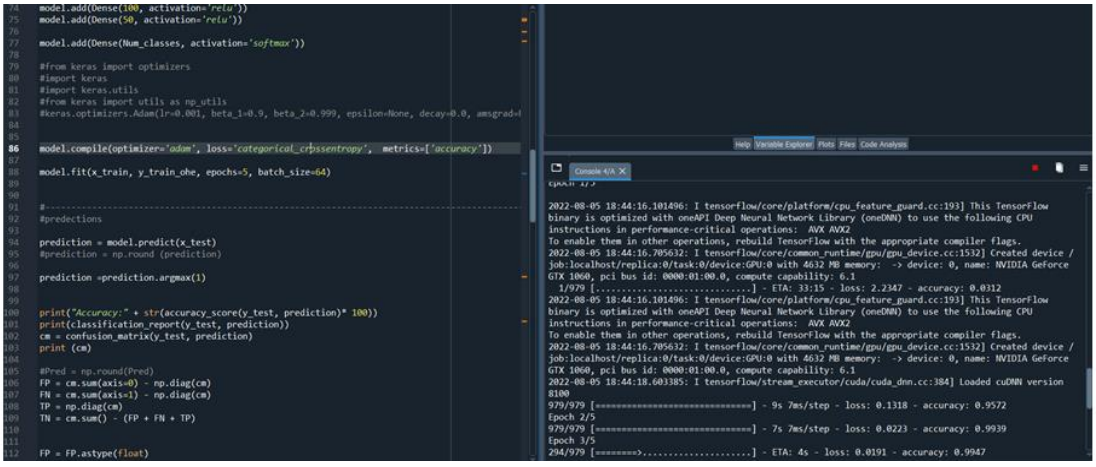

Fig 6.2: Implementation of GRU

7 DNN

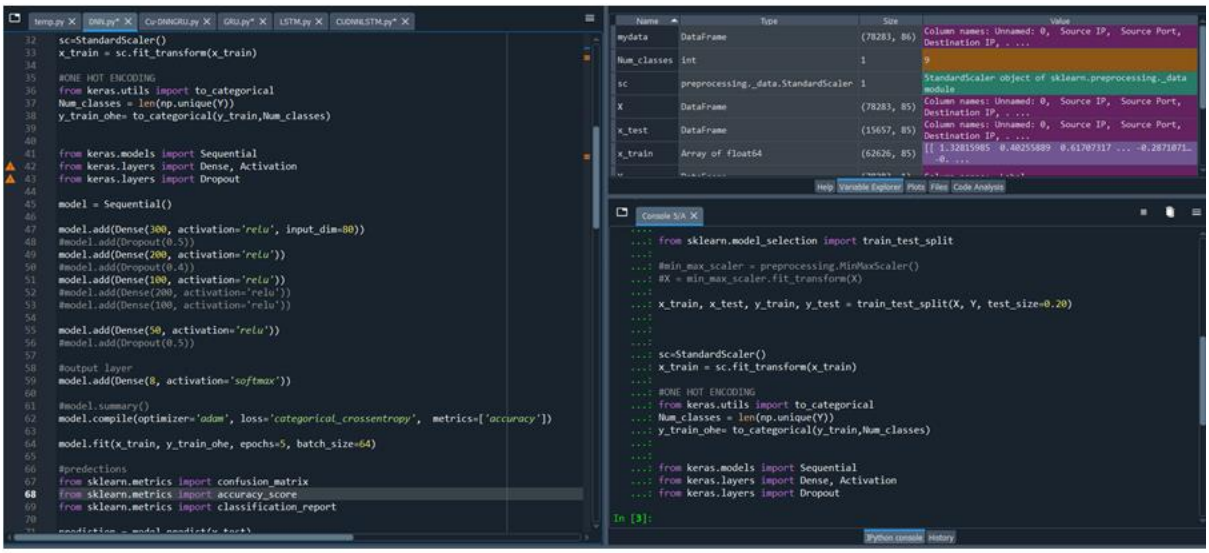

Fig 7.1: Implementation of DNN

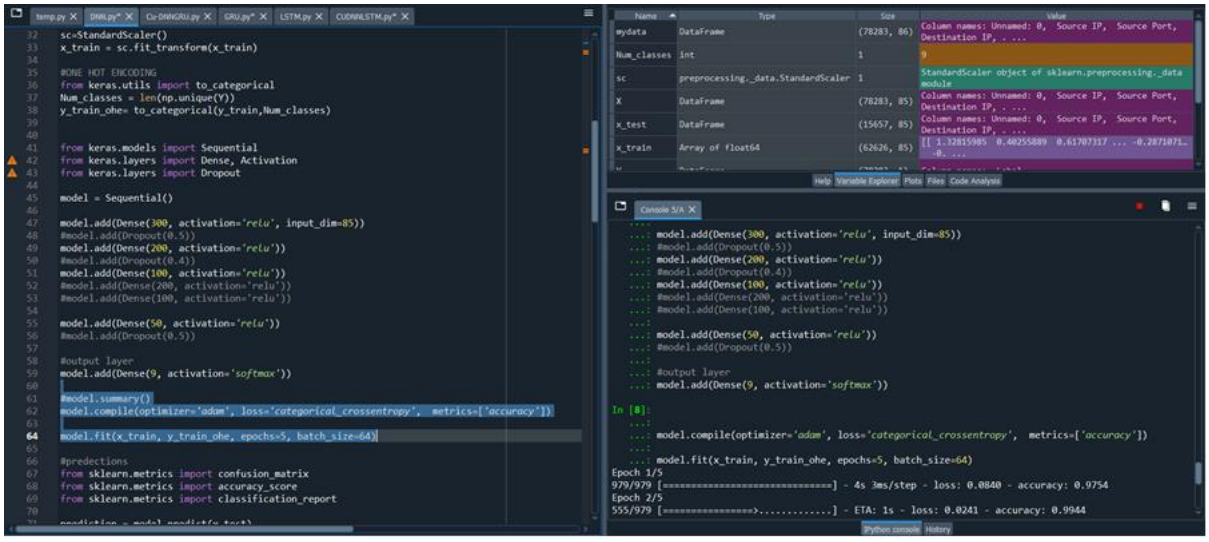

Fig 7.2: Implementation of DNN

8 Confusion Matrix

```
#predections
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

prediction = model.predict(x_test)
#
prediction =prediction.argmax(1)

print("Accuracy:" + str(accuracy_score(y_test, prediction)* 100))
print(classification_report(y_test, prediction))
cm = confusion_matrix(y_test, prediction)
print (cm)
```

Fig 8.1: Predection of Confusion Matrix

9 Results:

```
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)


FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
```

Fig 9.1: Output

10 Graphs:

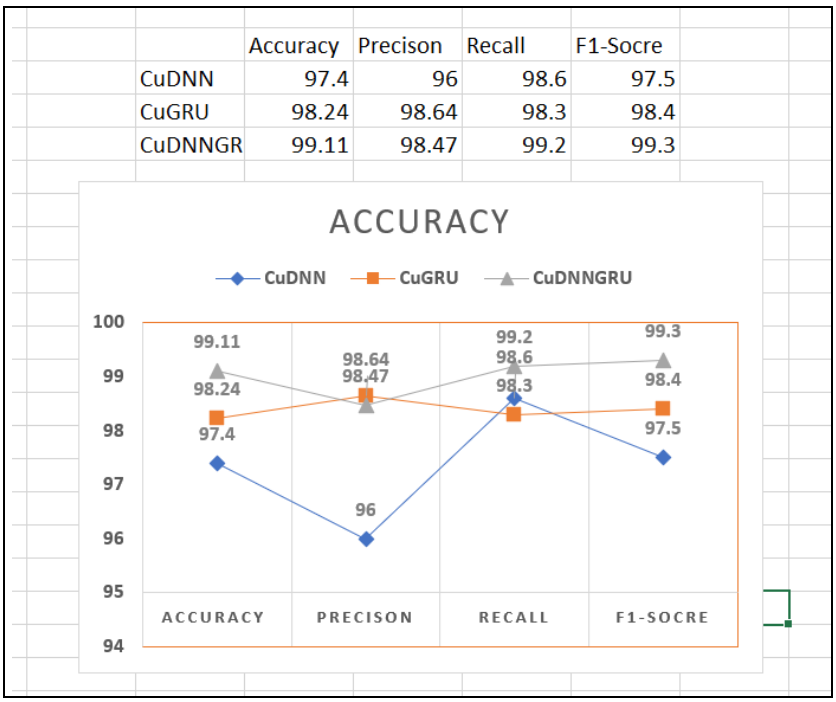

Fig 10.1: Accuracy Graph Output

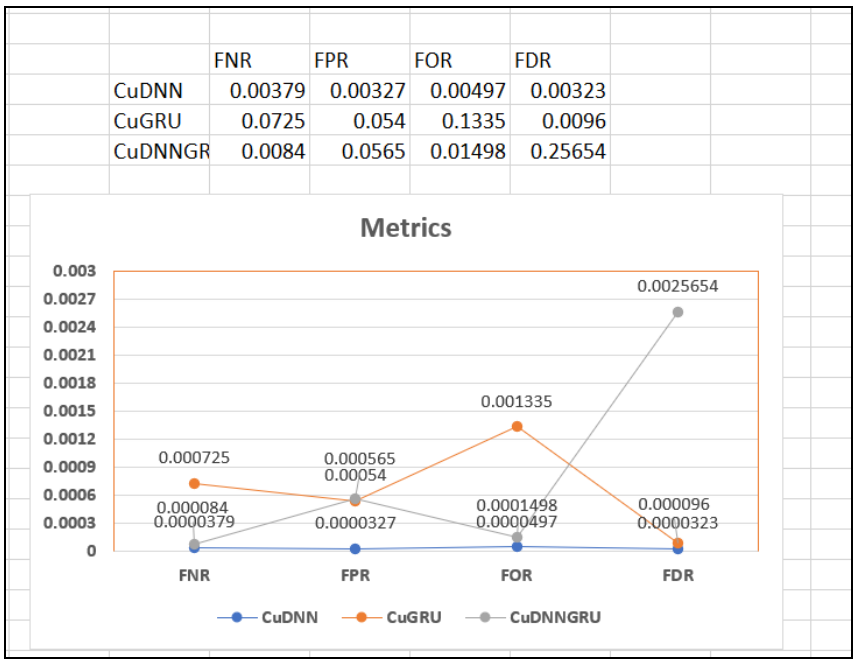

Fig 10.2: FPR Metrics Graph Output

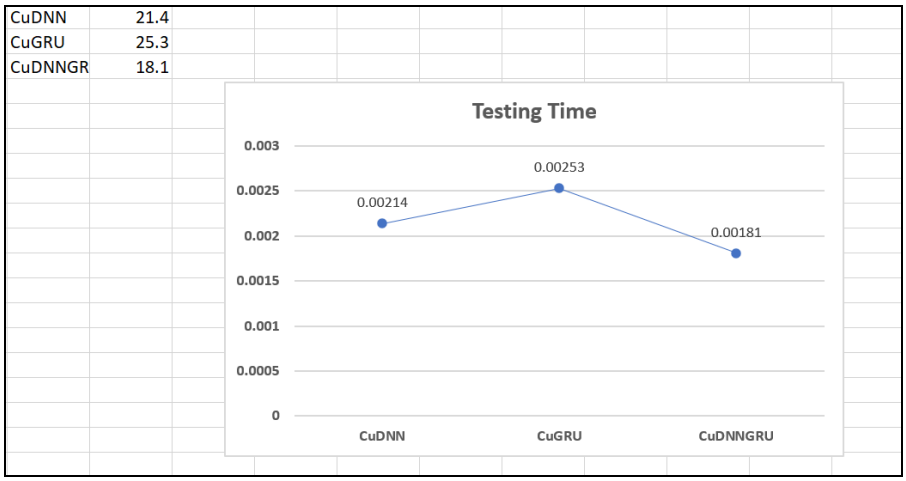

Fig 10.3: Testing Model Training Graph Output

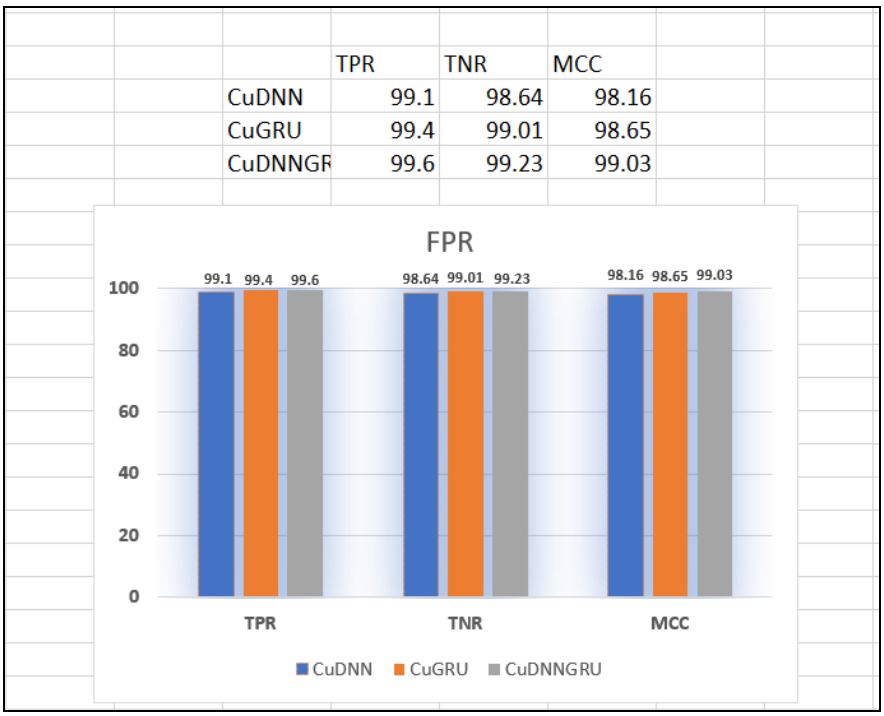

Fig 10.4: TNR Graph Output