

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)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

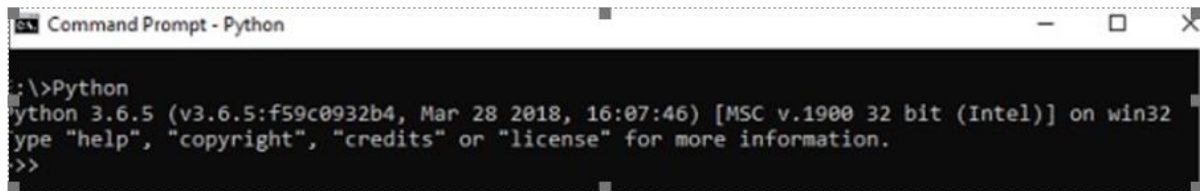
The OS module in Python performs the functionality for creation and removal of directory (folder), gathering its contents, changing as well as identifying the current directory, and so on. The OS module must be firstly imported for interacting with underlying operating system (OS). Pandas which is a 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.



```
Command Prompt - Python
:\>Python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
type "help", "copyright", "credits" or "license" for more information.
>>
```

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.

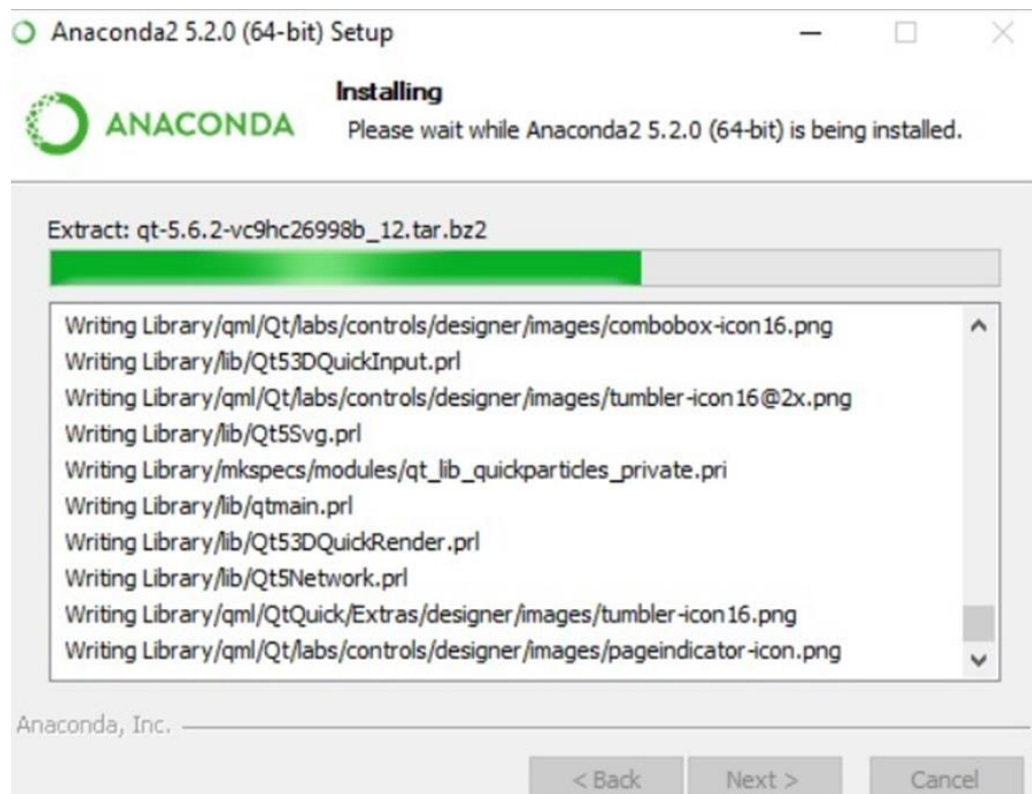


Fig 4.2: install Anaconda framework

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

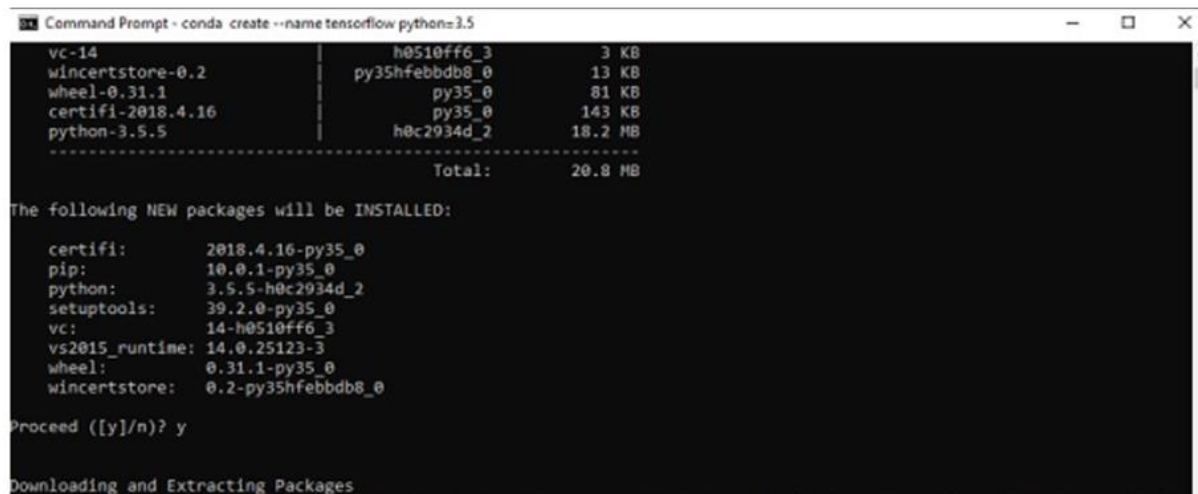
```
C:\Users\Radhika>conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:
positional arguments:
  command
  clean                Remove unused packages and caches.
  config              Modify configuration values in .condarc. This is modeled
                    after the git config command. Writes to the user .condarc
                    file (C:\Users\Radhika\condarc) by default.
  create              Create a new conda environment from a list of specified
                    packages.
  help                Displays a list of available conda commands and their help
                    strings.
  info                Display information about current conda install.
  install             Installs a list of packages into a specified conda
                    environment.
  list                List linked packages in a conda environment.
  package             Low-level conda package utility. (EXPERIMENTAL)
  remove              Remove a list of packages from a specified conda environment.
  uninstall           Alias for conda remove. See conda remove --help.
  search              Search for packages and display associated information. The
                    input is a MatchSpec, a query language for conda packages.
                    See examples 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



```
Command Prompt - conda create --name tensorflow python=3.5
vc-14                | h0510ff6_3          | 3 KB
wincertstore-0.2     | py35hfbbdb8_0      | 13 KB
wheel-0.31.1         | py35_0              | 81 KB
certifi-2018.4.16   | py35_0              | 143 KB
python-3.5.5         | h0c2934d_2          | 18.2 MB
-----
Total:                |                    | 20.8 MB

The following NEW packages will be INSTALLED:

certifi: 2018.4.16-py35_0
pip: 10.0.1-py35_0
python: 3.5.5-h0c2934d_2
setuptools: 39.2.0-py35_0
vc: 14-h0510ff6_3
vs2015_runtime: 14.0.25123-3
wheel: 0.31.1-py35_0
wincertstore: 0.2-py35hfbbdb8_0

Proceed ([y]/n)? y
Downloading and Extracting Packages
```

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

```

Command Prompt - pip install tensorflow
Requirement already satisfied: termcolor>=1.1.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.8.0)
Requirement already satisfied: setuptools in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from protobuf>=3.4.0->tensorflow) (39.2.0)
Requirement already satisfied: html5lib==0.9999999 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow) (0.9999999)
Requirement already satisfied: bleach==1.5.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow) (1.5.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorboard<1.9.0,>=1.8.0->tensorflow) (0.14.1)
Installing collected packages: tensorflow

```

Fig 4.5: Installation of tensorflow-gpu

```

Command Prompt
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages

```

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.

```

Python 3.5.5 [Anaconda, Inc.] (default, Apr 7 2018, 04:52:34) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, Tensorflow!')
>>> sess = tf.session()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'tensorflow' has no attribute 'session'
>>> sess = tf.Session()
2018-06-28 11:12:04.586763: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
>>> print(sess.run(hello))
b'Hello, 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:

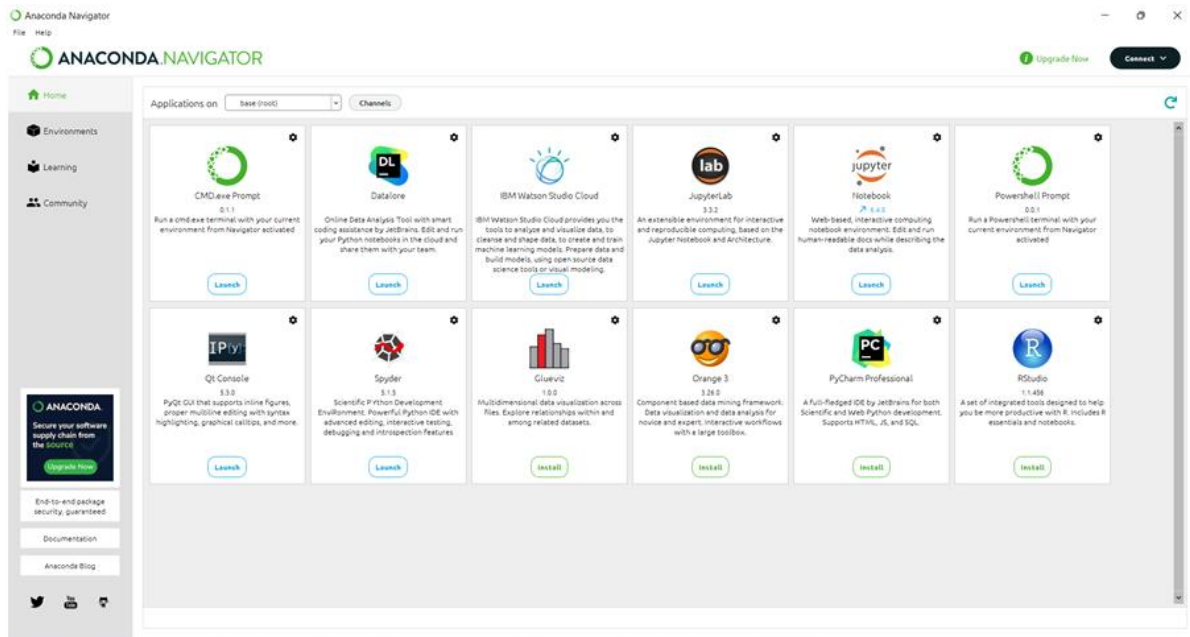


Fig 5.1: Anaconda Navigation Phase

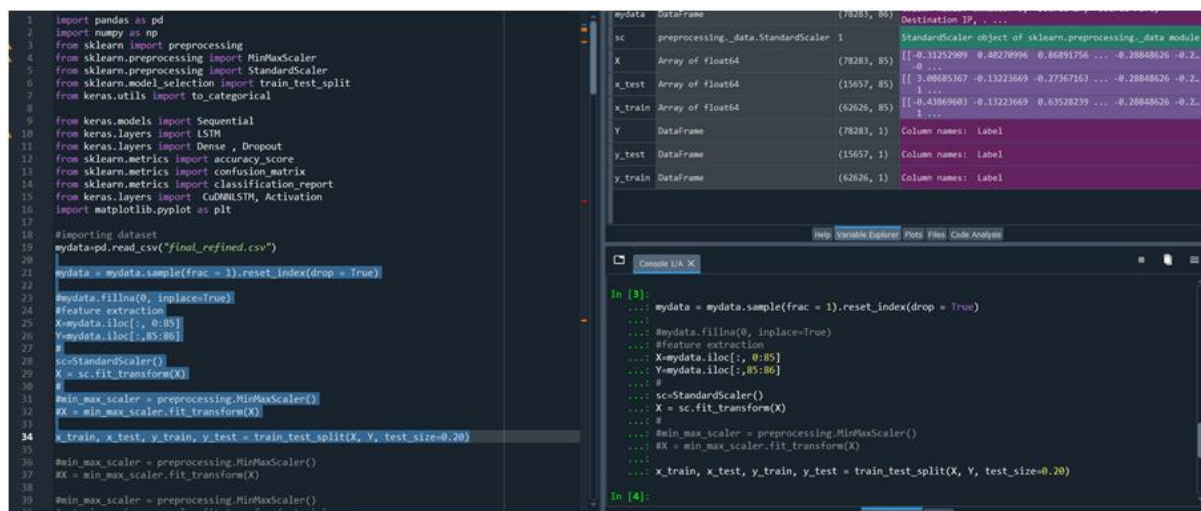


Fig 5.2: Importation of the libraries


```

53 y_train_ohc = to_categorical(y_train, num_classes)
54 y_train_ohc = pd.DataFrame(y_train_ohc)
55
56 #Building RNN
57 model = Sequential()
58 model.add(CuDNNGRU(300, input_shape=(x_train.shape[1],x_train.shape[2]), return_sequences=True))
59 model.add(CuDNNGRU(600, return_sequences=True))
60 model.add(CuDNNGRU(600, return_sequences=False))
61 model.add(Dropout(0.4))
62
63 #model.add(CuDNNGRU(300, return_sequences=False))
64
65 model.add(Activation('relu'))
66 #model.add(LSTM(200, activation='relu', return_sequences=True))
67 #model.add(LSTM(150, activation='relu', return_sequences=True))
68 #model.add(LSTM(100, activation='relu', return_sequences=False))
69 model.add(Dense(200, activation='relu'))
70 #model.add(Dense(150, activation='relu'))
71 #model.add(Dense(100, activation='relu'))
72 model.add(Dropout(0.2))
73
74 model.add(Dense(100, activation='relu'))
75 model.add(Dense(50, activation='relu'))
76
77 model.add(Dense(num_classes, activation='softmax'))
78
79 #from keras import optimizers
80 #import keras
81 #import keras.utils
82 #from keras import utils as np_utils
83 #keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
84
85 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
86
87 model.fit(x_train, y_train_ohc, epochs=5, batch_size=64)
88
89
90

```

Fig 5.3: Importation of the libraries

6 GRU

```

27 min_max_scaler = preprocessing.MinMaxScaler()
28 X = min_max_scaler.fit_transform(X)
29
30 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
31
32 x_train = np.array(x_train)
33 x_train = np.reshape(x_train, (x_train.shape[0], 1, x_train.shape[1]))
34
35 #Reshape and normalize test data
36 x_test = np.array(x_test)
37 x_test = np.reshape(x_test, (x_test.shape[0], 1, x_test.shape[1]))
38
39 #Building RNN
40 model = Sequential()
41
42 model.add(GRU(200, activation='relu', input_shape=(x_train.shape[1],x_train.shape[2]), return_sequences=True))
43 model.add(GRU(100, activation='relu', return_sequences=True))
44 model.add(Dropout(0.2))
45
46 #model.add(GRU(200, activation='relu', return_sequences=True))
47 model.add(GRU(80, activation='relu', return_sequences=False))
48
49
50 #model.add(Dense(200, activation='relu'))
51 model.add(Dense(100, activation='relu'))
52 model.add(Dropout(0.3))
53 model.add(Dense(70, activation='relu'))
54
55 #model.add(Dense(50, activation='relu'))
56 model.add(Dense(num_classes, activation='softmax'))
57
58
59 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
60 model.summary()
61
62 import time
63 start = time.clock()
64
65

```

Fig 6.1: Implementation of GRU

```

94 model.add(Dense(100, activation='relu'))
95 model.add(Dense(50, activation='relu'))
96
97 model.add(Dense(num_classes, activation='softmax'))
98
99 #from keras import optimizers
100 #import keras
101 #import keras.utils
102 #from keras import utils as np_utils
103 #keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
104
105 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
106
107 model.fit(x_train, y_train_ohc, epochs=5, batch_size=64)
108
109
110 #-----
111 #predictions
112 prediction = model.predict(x_test)
113 prediction = np.round(prediction)
114
115 prediction = prediction.argmax(1)
116
117
118 print("Accuracy: " + str(accuracy_score(y_test, prediction)* 100))
119 print(classification_report(y_test, prediction))
120 cm = confusion_matrix(y_test, prediction)
121 print(cm)
122
123 #Pred = np.round(Pred)
124 FP = cm.sum(axis=0) - np.diag(cm)
125 FN = cm.sum(axis=1) - np.diag(cm)
126 TP = np.diag(cm)
127 TN = cm.sum() - (FP + FN + TP)
128
129 FP = FP.astype(float)

```

Fig 6.2: Implementation of GRU

7 DNN

```
32 sc=StandardScaler()
33 x_train = sc.fit_transform(x_train)
34
35 #ONE HOT ENCODING
36 from keras.utils import to_categorical
37 Num_classes = len(np.unique(Y))
38 y_train_oh = to_categorical(y_train, Num_classes)
39
40
41 from keras.models import Sequential
42 from keras.layers import Dense, Activation
43 from keras.layers import Dropout
44
45 model = Sequential()
46
47 model.add(Dense(300, activation='relu', input_dim=80))
48 #model.add(Dropout(0.5))
49 model.add(Dense(200, activation='relu'))
50 #model.add(Dropout(0.4))
51 model.add(Dense(100, activation='relu'))
52 #model.add(Dense(200, activation='relu'))
53 #model.add(Dense(100, activation='relu'))
54
55 model.add(Dense(50, activation='relu'))
56 #model.add(Dropout(0.5))
57
58 #output layer
59 model.add(Dense(8, activation='softmax'))
60
61 #model.summary()
62 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
63
64 model.fit(x_train, y_train_oh, epochs=5, batch_size=64)
65
66 #predictions
67 from sklearn.metrics import confusion_matrix
68 from sklearn.metrics import accuracy_score
69 from sklearn.metrics import classification_report
70
71 prediction = model.predict(x_test)
```

Name	Type	Size	Value
mydata	DataFrame	(7828, 86)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
Num_classes	int	1	9
sc	preprocessing_data.StandardScaler	1	StandardScaler object of sklearn.preprocessing_data module
X	DataFrame	(7828, 85)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
x_test	DataFrame	(15657, 85)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
x_train	Array of float64	(62626, 85)	[[1.32815985 0.40255889 0.61707317 ... -0.28710711 ...

```
...
... from sklearn.model_selection import train_test_split
...
... #main_max_scaler = preprocessing.MinMaxScaler()
... #X = min_max_scaler.fit_transform(X)
...
... x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
...
...
... sc=StandardScaler()
... x_train = sc.fit_transform(x_train)
...
... #ONE HOT ENCODING
... from keras.utils import to_categorical
... Num_classes = len(np.unique(Y))
... y_train_oh = to_categorical(y_train, Num_classes)
...
...
... from keras.models import Sequential
... from keras.layers import Dense, Activation
... from keras.layers import Dropout
...
In [3]:
```

Fig 7.1: Implementation of DNN

```
32 sc=StandardScaler()
33 x_train = sc.fit_transform(x_train)
34
35 #ONE HOT ENCODING
36 from keras.utils import to_categorical
37 Num_classes = len(np.unique(Y))
38 y_train_oh = to_categorical(y_train, Num_classes)
39
40
41 from keras.models import Sequential
42 from keras.layers import Dense, Activation
43 from keras.layers import Dropout
44
45 model = Sequential()
46
47 model.add(Dense(300, activation='relu', input_dim=85))
48 #model.add(Dropout(0.5))
49 model.add(Dense(200, activation='relu'))
50 #model.add(Dropout(0.4))
51 model.add(Dense(100, activation='relu'))
52 #model.add(Dense(200, activation='relu'))
53 #model.add(Dense(100, activation='relu'))
54
55 model.add(Dense(50, activation='relu'))
56 #model.add(Dropout(0.5))
57
58 #output layer
59 model.add(Dense(9, activation='softmax'))
60
61 #model.summary()
62 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
63
64 model.fit(x_train, y_train_oh, epochs=5, batch_size=64)
65
66 #predictions
67 from sklearn.metrics import confusion_matrix
68 from sklearn.metrics import accuracy_score
69 from sklearn.metrics import classification_report
70
71 prediction = model.predict(x_test)
```

Name	Type	Size	Value
mydata	DataFrame	(7828, 86)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
Num_classes	int	1	9
sc	preprocessing_data.StandardScaler	1	StandardScaler object of sklearn.preprocessing_data module
X	DataFrame	(7828, 85)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
x_test	DataFrame	(15657, 85)	Column names: Unnamed: 0, Source IP, Source Port, Destination IP, ...
x_train	Array of float64	(62626, 85)	[[1.32815985 0.40255889 0.61707317 ... -0.28710711 ...

```
...
... model.add(Dense(300, activation='relu', input_dim=85))
... #model.add(Dropout(0.5))
... model.add(Dense(200, activation='relu'))
... #model.add(Dropout(0.4))
... model.add(Dense(100, activation='relu'))
... #model.add(Dense(200, activation='relu'))
... #model.add(Dense(100, activation='relu'))
...
... model.add(Dense(50, activation='relu'))
... #model.add(Dropout(0.5))
...
... #output layer
... model.add(Dense(9, activation='softmax'))
...
In [8]:
...
... model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
...
... model.fit(x_train, y_train_oh, epochs=5, batch_size=64)
Epoch 1/5
979/979 [=====] - 4s 3ms/step - loss: 0.0840 - accuracy: 0.9754
Epoch 2/5
555/979 [=====] - ETA: 1s - loss: 0.0241 - accuracy: 0.9944
```

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: Prededion 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:

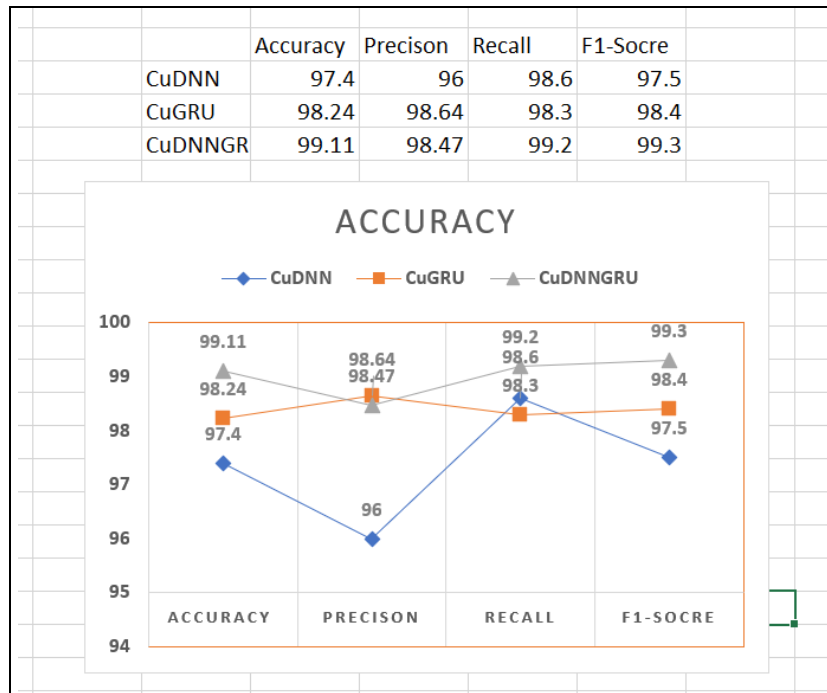


Fig 10.1: Accuracy Graph Output

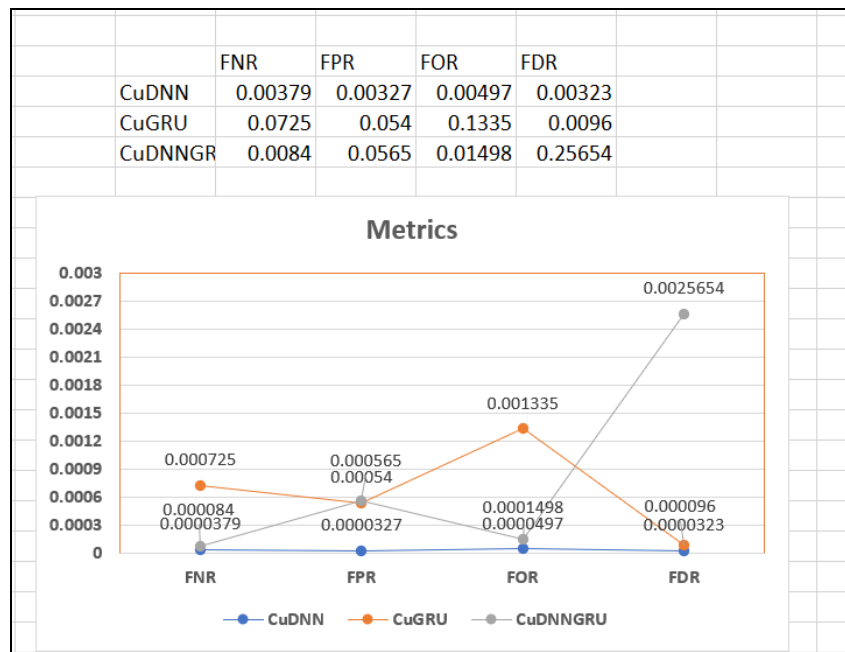


Fig 10.2: FPR Metrics Graph Output

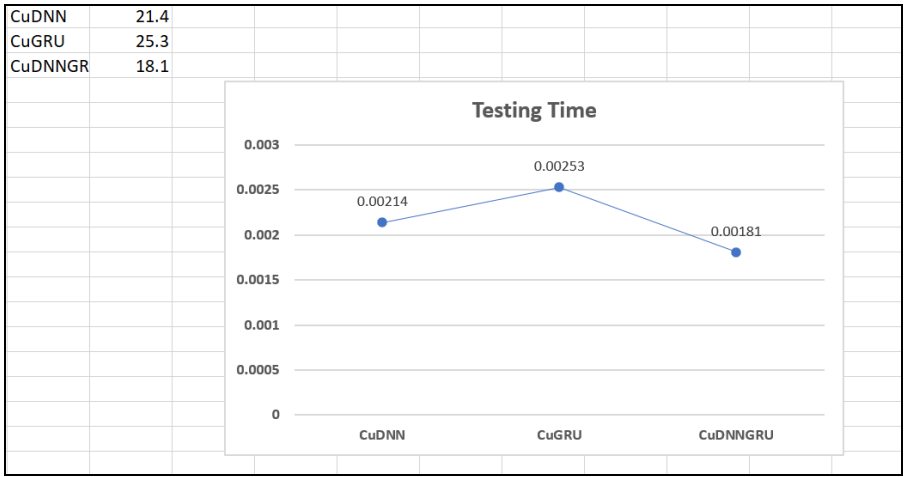


Fig 10.3: Testing Model Training Graph Output

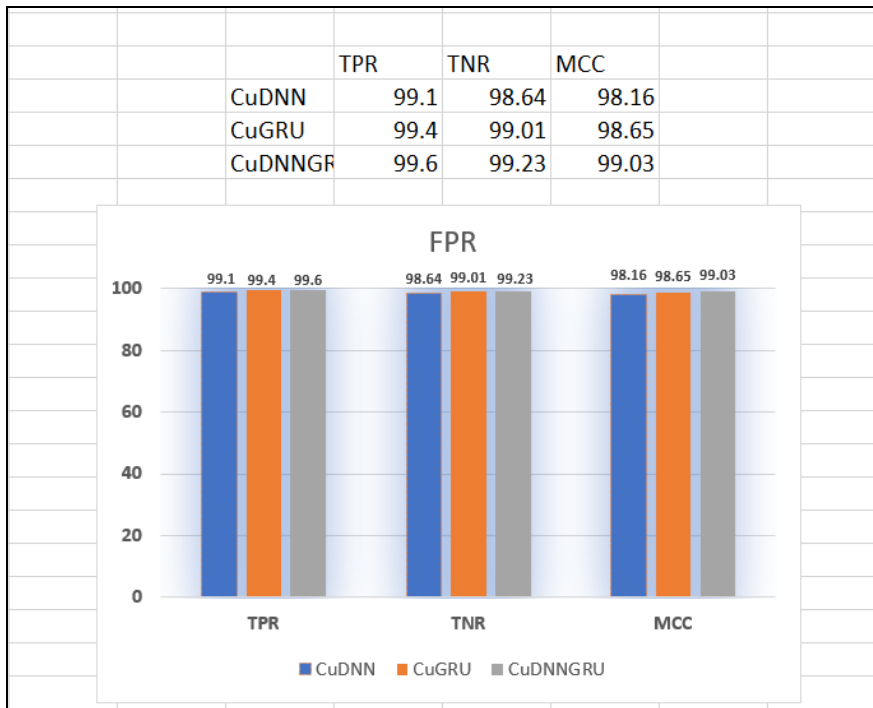


Fig 10.4: TNR Graph Output