National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Akshay Wakhare
Student ID: X19208103

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Akshay Ashok Wakhare
Student ID:	X19208103
Programme:	MSc in Cyber Security **Year:** 2020-2021
Module:	MSc Internship
Lecturer:	Prof. Vikas Sahni
Submission Due Date:	06/09/2021
Project Title:	Malware Detection in Android platform using DNN
Word Count:	……865…… **Page Count:** ……8……..…….……

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: Akshay Ashok Wakhare

Date: ……05/09/2021……

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

Malware Detection in Android platform using DNN

Akshay Wakhare
Student ID: X19208103

1 Introduction

The configuration manual document gives an overview and insights of the research carried out as the part of the Industry Internship. This manual will provide the details of the system configuration, Tools utilized while performing the research and the implementation of the project. In this project two deep learning models were developed as part of the research. The implementation section will guide through the process carried out in the development phase along with the final results of the research. The internship task report is also mentioned in this manual.

2 System Configuration

The system used while performing the activity was personal as the internship was Remote. The configuration of the system is as follows:

2.1 Hardware Configuration

- Operating system: Windows 10
- Processor: Intel i5-10th gen
- System Compatibility: 64-bit
- Hard Disk: Hybrid (256GB SSD + 1 TB HDD)
- RAM: 8GB

2.2 Software Configurations:

Prior to start the model building phase following software, tools and libraries were installed in the system.

Software/Tools	Version	Information
Python	3.8.5	To develop the model python is used in this project.
Anaconda	-	It is windows suitable platform that allows users computations, package management and model deployments. (Anaconda | The World's Most Popular Data Science Platform, 2021)
TensorFlow	2.5.0	For running deep neural

		networks the TensorFlow is the important library. (TensorFlow, 2021)
Keras	2.5.0	It is used to provide powerful deep learning APIs to boost the performance and for scaling. (Team, 2021)
NumPy	1.19.5	It is an open-source tools used to perform complex mathematical problems in data. (NumPy, 2021)
Sci-Kit Learn	0.24.1	It is the library which is utilized for problems such as Classification, Regression as well as for data pre-processing. (scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation, 2021)

3 Implementation

In this section the step-by-step guide is mentioned to run the project in any windows system.

1. Download and Install Anaconda Software in windows system. (https://www.anaconda.com/products/individual)

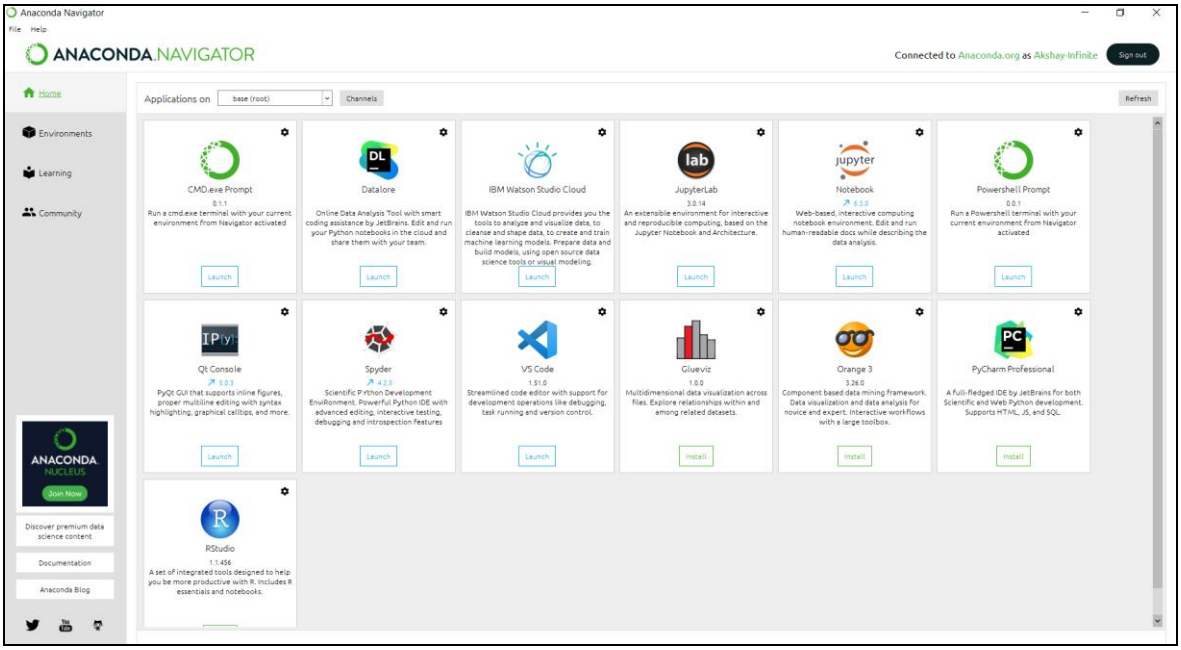


Fig.1 Anaconda Navigator

2. Open the Jupyter Notebook from Anaconda.

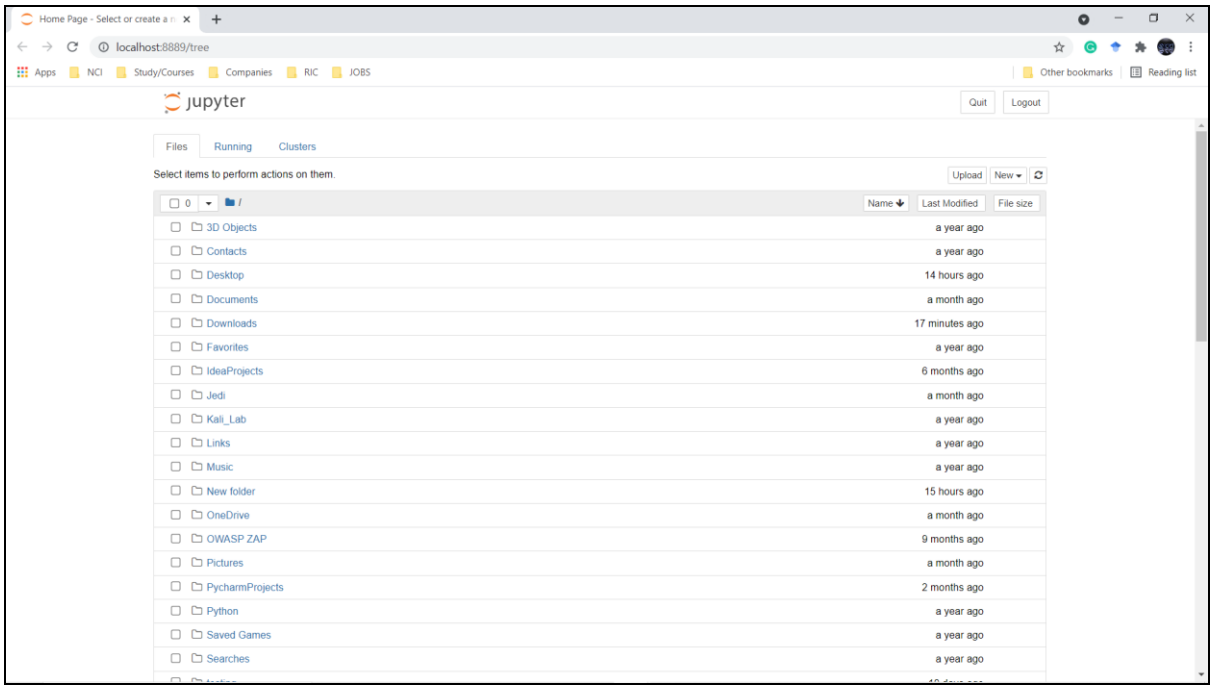


Fig. 2 Jupyter Notebook

3. After opening jupyter notebook click on new notebook (python 3) in which the development part for model will be covered.

4. In new notebook first import all the required libraries.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from scipy import stats
```

5. After that import the provided dataset.

```
dynamic = pd.read_csv('D:/Thesis/dynamic1.csv')
```

6. From this the data pre-processing will be done using following code.

```
In [ ]: # Checking Null and Empty Values
df = pd.concat([dynamic.isnull().sum(),dynamic.eq('').sum()],keys=['Nulls','Empty'],axis=1)
df.head(500)

In [ ]: # Encoding the Target Labels
le = LabelEncoder()
resencode = le.fit_transform(dynamic["Class"])
resencode
dynamic['Class'] = resencode
dynamic

In [ ]: # Checking data skewness
pd.set_option('display.max_rows', 16000)
dynamic.skew()

In [ ]: # Applying Cube Root to normalise the skewness
for keys in dynamic.columns:
    print (keys)
    if dynamic[keys].skew() > 1 or dynamic[keys].skew() < -1:
        dynamic[keys] = np.cbrt(dynamic[keys])

In [ ]: pd.set_option('display.max_rows', 16000)
dynamic.skew()

In [ ]: # Importing the dataset values into target variable
Features = dynamic.iloc[:, [296,469,53,428,266,68,323,8,0,107,294,290,468,332,187,244,213,143,38,301]].values
Result = dynamic.iloc[:, -1].values

In [ ]: # Applying train and test Split
from sklearn.model_selection import train_test_split
Features_train, Features_test, Result_train, Result_test = train_test_split(Features, Result, test_size = 0.2, random_state = 42)
```

7. After data pre-processing the model is defined and trained.

```
In [ ]: # Reshaping 2-dimentional input train data
sampled_Features_train = sampled_Features_train.reshape(len(sampled_Features_train), 1, sampled_Features_train.shape[1])

In [ ]: # Reshaping 2-dimentional input test data
Features_test = Features_test.reshape(len(Features_test), 1,  Features_test.shape[1])

In [ ]: #Defining model
DLSTM = Sequential()
# First Layer
DLSTM.add(LSTM(units = 32, activation = 'relu',  return_sequences = True, input_shape=(sampled_Features_train.shape[1], sampled_F
DLSTM.add(Dropout(0.2))
# Second Layer
DLSTM.add(LSTM(units = 32, activation = 'relu'))
DLSTM.add(Dropout(0.2))
# Output Layer
DLSTM.add(Dense(units=1, activation = 'sigmoid'))
DLSTM.add(Dropout(0.2))

In [ ]: DLSTM.compile(optimizer = 'adam', loss = 'mse', metrics = ['accuracy'])

In [ ]: # history = DLSTM.fit(sampled_Features_train, sampled_Result_train, epochs =100, batch_size = 32)

history = DLSTM.fit(sampled_Features_train, sampled_Result_train, validation_data = (sampled_Features_train, sampled_Result_trair
```

8. The Accuracy and Loss graphs are calculated after model training.

```
# Plotting Accuracy Graph
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper right')
plt.show()
```

```
# Plotting Loss Graph
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper right')
plt.show()
```

9. In this step the model is tested using test data. The confusion matrix and roc & auc score are calculated using following code.

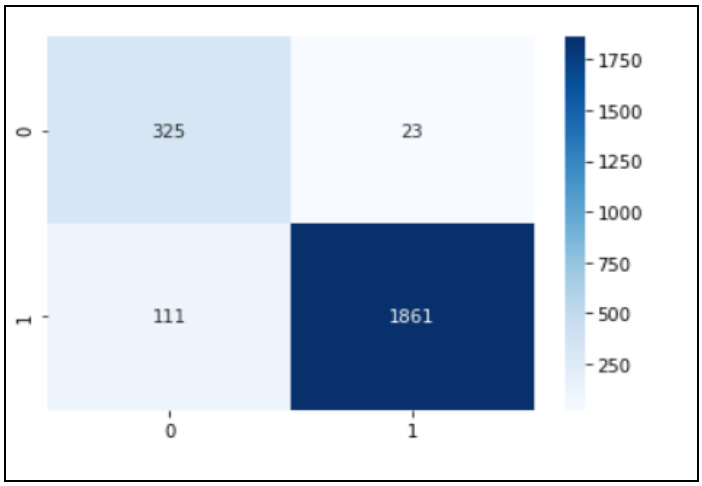
```
# Prediction of the train model with testing data
Result_pred = DLSTM.predict(Features_test)
Result_pred = (Result_pred > 0.5)
print(np.concatenate((Result_pred.reshape(len(Result_pred),1), Result_test.reshape(len(Result_test),1)),1))
```

```
# Calculating the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Result_test, Result_pred)
print(cm)
accuracy_score(Result_test, Result_pred)
```

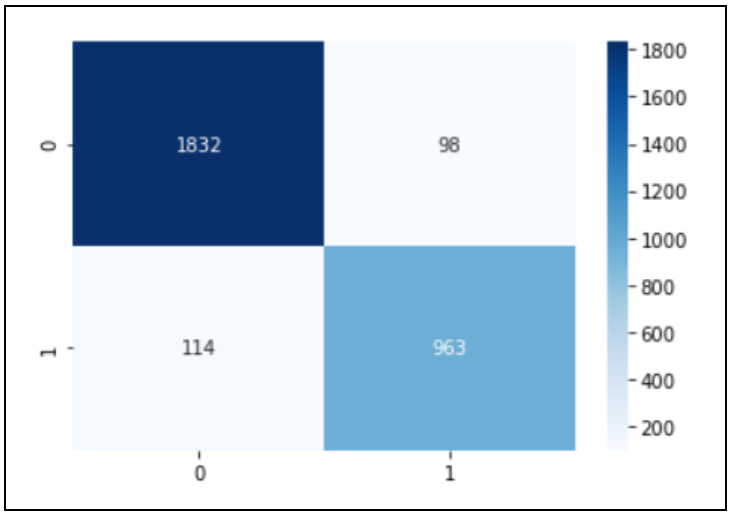
```
from sklearn.metrics import roc_auc_score

Accuracy = cm.diagonal().sum() / cm.sum()
print("Accuracy: " + str(Accuracy))

Precision = cm[1,1] / (cm[0,1] + (cm[1,1]))
print("Precision: " + str(Precision))

Sensitivity = cm[1,1] / (cm[1,0] + (cm[1,1]))
print("Sensitivity: " + str(Sensitivity))

Specificity = cm[0,0] / (cm[0,0] + (cm[0,1]))
print("Specificity: " + str(Specificity))

# rf_roc_auc_score = roc_auc_score(y_test, y_pred)
# print("ROC AUC Score: " + str(rf_roc_auc_score))

F1_score = 2 * (Precision * Sensitivity) / (Precision + Sensitivity)
print('F1 Score: ' + str(F1_score))
```

```
sns.heatmap(cm,cmap='Blues',annot=True, fmt='g')
```

10. The confusion matrix i.e., the final output of the model is plotted for both static and dynamic model. Also, the roc & auc scores along with testing time are calculated as model efficiency parameters.



Confusion Matrix for Dynamic Model



Confusion Matrix for Static Model

ROC & AUC Scores:

```
Accuracy: 0.9422413793103448
Precision: 0.9877919320594479
Sensitivity: 0.9437119675456389
Specificity: 0.9339080459770115
F1 Score: 0.9652489626556016
```

ROC & AUC scores for Dynamic model

```
Accuracy: 0.9294978383771201
Precision: 0.9076343072573044
Sensitivity: 0.8941504178272981
Specificity: 0.9492227979274611
F1 Score: 0.9008419083255378
```

ROC & AUC scores for Static model

Model Execution Time:

- The total time required to train the dynamic model with 100 epochs were 292.06 Seconds whereas the Testing time required to test the data on trained model was 0.21 milliseconds.
- For Static model total time required to train the model with 100 epochs were 393.67 Seconds whereas the Testing time required to test the data on trained model was 0.26 milliseconds.

In this research the feature importance is obtained using XGBoost for that the code is attached in the file named feature_importance.ipynb. The final code files are attached with project files named as Dynamic_RNN.ipynb and Static_RNN.ipynb respectively.

4 Internship Task Report

The Internship Activity Report is a 1-page monthly summary of the activities performed by you and what you have learned during that month. The Internship Activity Report must be signed off by your Company and included in the configuration manual as part of the portfolio submission.

Student Name: Akshay Ashok Wakhare . Company: Uniken India

Student number: x19208103 . Month Commencing: June 2021- August2021

Role Description:

The aim of the internship was to study and understand the Mobile Threat Detection Model. Perform research to suggest a solution for malware detection in android application by using deep learning technology. The task performed are:

- Studied and analysed the current Mobile Threat Detection model documentation.
- Carried out the research for the malware detection in android OS.
- Performed development activity for proposed solution.
- Developed and evaluated the models.
- Performed the manual testing on the current MTD model for various threat detection scenarios.
- Prepared the documentation for the activity performed.

Employer comments

- Akshay carried out the assigned tasks in time and demonstrated a good understanding about the product. The research carried by him in the area of malware detection in android has shown good results and can be used in current product.
- Akshay also performed the testing of the current MTD product to perform and detect various mobile threats. He was dedicated towards his assigned work and managed to perform the activity remotely.
- Given a proper guidance he can learn and understand delivering the artifacts to the clients. And he will be good asset to any organization.

Student Signature:  Date: September 5th 2021

Industry Supervisor Signature: 

Date: September 3rd 2021

References

Anaconda. 2021. *Anaconda | The World's Most Popular Data Science Platform*. [online] Available at: <https://www.anaconda.com/>.

TensorFlow. 2021. *TensorFlow*. [online] Available at: <https://www.tensorflow.org/>.

Team, K., 2021. *Keras: the Python deep learning API*. [online] Keras.io. Available at: <https://keras.io/>.

Numpy.org. 2021. *NumPy*. [online] Available at: <https://numpy.org/>.

Scikit-learn.org. 2021. *scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation*. [online] Available at: <https://scikit-learn.org/stable/>.