# Configuration Manual

MSc Research Project
Cybersecurity

Simon Onyebuchi Obetta
Student ID: x19152272

School of Computing
National College of Ireland

Supervisor: Arghir Nicolae Moldovan

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | | | |
|---|---|---|---|
| **Student Name:** | Simon Onyebuchi Obetta | | |
| **Student ID:** | x19152272 | | |
| **Programme:** | Cybersecurity | **Year:** | 2021 |
| **Module:** | MSc. Research Project | | |
| **Lecturer:** | Arghir Nicolae Moldovan | | |
| **Submission Due Date:** | 16th December 2021 | | |
| **Project Title:** | Detection of DDoS attacks in the IoT devices using Machine Learning Models on Urban IoT Dataset | | |
| **Word Count:** | 940 | **Page Count:** | 4 |

**Signature:** ……………………………………………

**Date:** 16th December 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

Simon Onyebuchi Obetta
x19152272

## 1 Introduction

This manual describes the software tools that were utilized to complete the project implementation. This manual also offers process instructions for installing essential software, loading code into systems, and completing a project to obtain the desired results.

## 2 Host Environment Requirement

The simulation was performed on a Lenovo Windows 10 laptop with the specifications listed below.
System Make: Lenovo
System Type: 64-bit operating system, x64-based processor
RAM: 8.00 GB
Window Edition: Window 10 Pro
Available Local Disk Storage: 400 GB
I recommend utilizing a Windows PC with more capabilities because the machine learning models were run on a large dataset (Urban IoT) with a size of 300MB to get a faster execution performance. The codes were developed in Windows operating system and contain some window commands that may be compatible with other operating systems.

## 3 Software Requirement

### 3.1 Development and Implementation Tools

The project codes are written in Python 3, and the implementation and configuration of each piece of code to reproduce the original result will be explained below. To build and run the programs, I used Spyder IDE 5.2.0 and Anaconda Prompt which are part of the Anaconda Navigator [2].

**Python 3:** The python 3 open-source can be downloaded for free via the Python website [1]. Because Python 3 is new and supports newer features and makes debugging easier, that is why I used this version.

**Anaconda Navigator** [2]: I used the two components of Anaconda Navigator which include Spyer IDE, mostly for the code's development. Used this because it is easier to debug code errors than other components like Jupyter Notebook. In addition, I used Anaconda Prompt for execution because it is faster to run my code.

## 3.2 Machine Learning Tools

After Python and Anaconda tools have been installed, the next step is to create a python virtue environment in the host machine and link the path to the Spyder IDE. This is because one of the machine learning tools 'TensorFlow' can only be installed in a python virtual environment. When the virtual environment has been installed and activated, then proceed with the below installation using the Anaconda Prompt.

**Table 1: Machine Learning Tools**

| No | Tools | Version |
|---|---|---|
| 1 | Numpy | 1.19.5 |
| 2 | tensorflow-gpu | 2.6.0 |
| 3 | pandas | 1.3.2 |
| 4 | matplotlib | 3.4.3 |
| 5 | sklearn | 0.0 |
| 6 | scikit-learn | 0.24.2 |
| 7 | geopandas | 0.9.0 |
| 8 | Shapely | 1.7.1 |
| 9 | haversine | 2.3.1 |

# 4. Design Specification

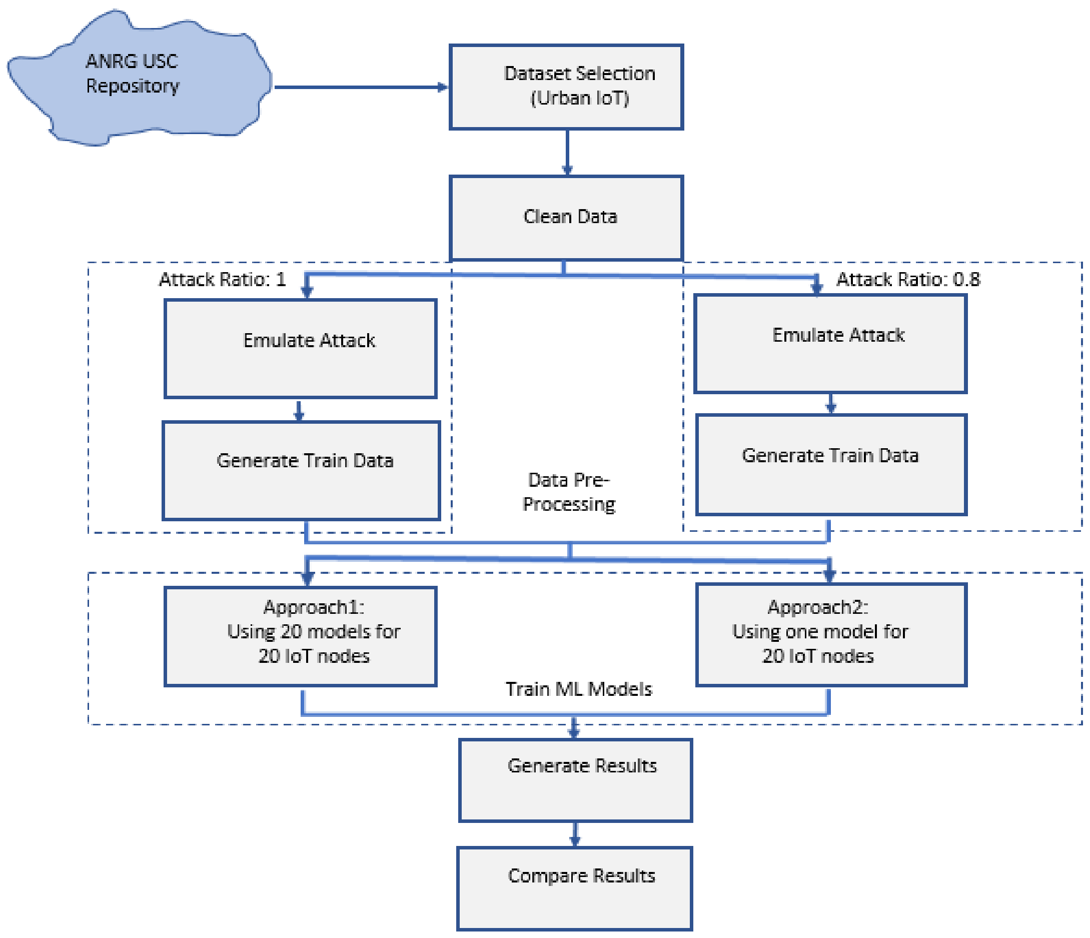

**Figure 1: Codes implementation design**

The design was carried out by employing two attack ratio approaches during attack emulation. The first approach is by using an attack ratio of 1 while the second approach has an attack ratio of, 0.8 as shown in figure 1 above. Therefore, the script folder for python scripts folder for attack ratio 1 and 0.8 are separated.

The below will describe the guide for running the codes and generating results for attack ratio 1 only. This is because the same steps can be replicated for the attack ratio of 0.8.

### 4.1. Dataset Used

The dataset used in this project is Urban IoT downloaded in the ANRG USC GitHub repository [3]. The dataset needs to be pre-processed and to fit the algorithm. The dataset is part of the zipped configuration folder and can be found in *Urban_IoT_DDoS_Data-main\dataset*.

## 5. Implementation

### 5.1. Clean Data

This is the first implementation and the code to clean data is available at:
*Urban_IoT_DDoS_Data main\source_code_ratio_1\clean_dataset\clean_data.py*.
Run the code and a benign dataset will be generated in the clean output path. This process normally takes 5 minutes to complete.

### 5.2. Emulate Attack

The script generates a DDoS attack on the dataset based on the attacked ratio set. The generate_attack.py script is available at
*Urban_IoT_DDoS_Data-main\source_code_ratio_1\attack_emulation\generate_attack.py*
The output generated six train and test datasets each. This process normally takes 20 minutes to complete.

### 5.3. Generate Training Data

This generates the training data by averaging the occupancy rates on the nodes over various time frames. The generate_training_data.py script is available at:
*Urban_IoT_DDoS_Data-main\ source_code_ratio_1\nn_training\generate_training_data.py*
The output generates train and test dataset. This process normally takes 5 days to complete.

### 5.4. Training Machine Learning Models and Generate Results

Note that models training also has two approaches, the first approach is creating 20 models for 20 nodes. The second approach is creating one model for 20 nodes. Therefore, there are eight scripts for creating model scripts available. Four of the scripts with the name '*train_<model>_20model.py'* create 20 models while the other four with the name *'train_<model>_onemodel.py*' create only one model as proposed.

The models are Feedforward Neural networks (FNN), Deep Neural Network (DNN), Autoencoder, and Random Forest (RF) that train on the training dataset for detecting the

DDoS attackers. The scripts save the final model and also the epochs logs and weights in the Output\saved_20_model or Output\saved_1_model depending on the approach.

Whenever each model is run, then run generate_result_20model.py or generate_result_onemodel.py to generate a result for the model before running another one so as the subsequent model outputs do not overwrite it.

The results are saved in the path; "*Urban_IoT_DDoS_Data-main\source_code_ratio_1\nn_training\Output\report* "

After each round, rename the result generated "*general_report_binary_accuracy_max*" and the plots for the model so that the file will not be overridden by the next round.

The below is the step of executing the code for the model train and result generation.

Round 1. train_fnn_20model.py → generate_result_20model.py → rename general_report_binary_accuracy_max

Round 2. train_dnn_20model.py → generate_result_20model.py → rename general_report_binary_accuracy_max

Round 3. train_autoencoder_20model.py → generate_result_20model.py → rename general_report_binary_accuracy_max

Round 4. train_rf_20model.py → generate_result_20model.py → rename general_report_binary_accuracy_max

**Repeat step 5.4 for the one model approach.**

## 5.5. Compare Results

The mean result metrics (Accuracy, Recall, and Precision) for the 20 IoT nodes will be manually calculated and compared against other models.

**Repeat the full implementation processes for the attack ratio of 0.8**

# References

[1] python.org "Python Releases for Windows". [Online] Available at: https://www.python.org/downloads/windows/ [Accessed December 7, 2021].

[2] Anaconda.com, "Anaconda Navigator". [Online] Available at: https://docs.anaconda.com/anaconda/navigator/index.html [Accessed December 7, 2021].

[3] github.com, "ANRGUSC/Urban_IoT_DDoS_Data". [Online] Available at: https://github.com/ANRGUSC/Urban_IoT_DDoS_Data/tree/main/dataset [Accessed December 7, 2021].