Configuration Manual

MSc Research Project
Cybersecurity

Sabareesan Mohandass
Student ID: X22215786

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sabareesan Mohandass …………………………………………………………………
Student ID:	X22215786………………………………………………………….……………………………
Programme:	Cybersecurity……………………………………………… **Year:** 2023-2024
Module:	MSc Research Project………………………………………………………………………
Lecturer:	Michael Prior……………………………………………………………………………………
Submission Due Date:	12/08/2024…………………………………………………………………………….………
Project Title:	Addressing Cloud Security Challenges using AI-Driven IoT Intrusion Detection Systems with UQ-IDS Dataset …………………………………….………
Word Count:	1879……………………………… **Page Count:** 9…………………………….……….………

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: Sabareesan ………………………………………………………………………………………………………

Date: 12/08/2024 ………………………………………………………………………………………………………

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

Sabareesan Mohandass
X22215786

1 Introduction

The purpose of this document is to outline the steps necessary to execute the research project and to specify the configuration required to run the models and access the webpage. The key stages involve accessing the online platform and installing the essential software, libraries and packages, as well as the minimal setup needed for the project to function properly.

2 Experimental Setup

2.1 System Configuration

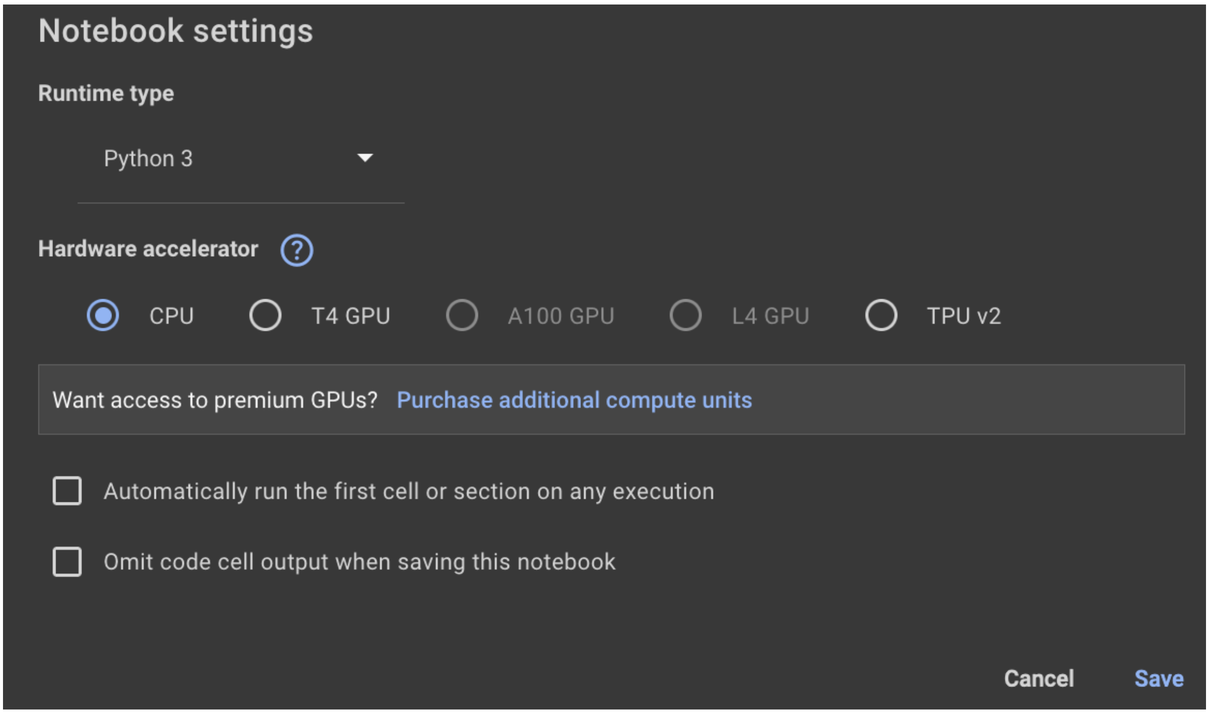
Hardware Used in this Experiment	Version	Purpose
MacBook Pro Processor: 2.3 GHz 8-Core Intel Core i9 Memory: 16 GB 2667 MHz DDR4	macOS Ventura Version 13.5.2 (22G91)	Workstation

2.2 Software Used in this Experiment

2.2.1 **Google Colab** is a free cloud service hosted by Google for machine learning education and research. Since my research work is based on ML, I took an advantage of it. It provides a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud, but code requires the following libraries:

- NumPy 1.22.4
- Pandas 1.5.3
- Plotly 5.13.1
- Matplotlib 3.7.1
- Seaborn 0.12.1
- Scikit-learn 1.2.2
- TensorFlow 2.12.0
- Keras 2.12.0

2.2.2 To run this code, will need a Google account and a web browser, open the code in Google Colab and run it by clicking the "Run" button.



2.3 Dataset

The newly published dataset NF-UQ-NIDS 2023 includes flows from various network configurations and attack parameters. The NF-UQ-NIDS dataset has a total of 11,994,893 records, out of which 9,208,048 (76.77%) are benign flows and 2,786,845 (23.23%) attacks.

Source: https://staff.itee.uq.edu.au/marius/NIDS_datasets/

2.3.1 Steps to upload dataset into Google Collab

1) Upload CSV file to Google Drive:
2) Mount Google Drive in Google Colab:
3) In Colab notebook, execute the following code:

```
from google.colab import drive
drive.mount('/content/drive')
```

4) Access the CSV file in Colab:

```
file_path = '/content/drive/My Drive/data.csv'
```

2.4 Web Framework

Flask is a lightweight and flexible web framework for Python. It's designed to be easy to learn and use, making it a popular choice for building web applications, APIs, and microservices.

Here, to enhance the usability and accessibility of IDS, developed a web application using Python Flask, provides a user-friendly interface to interact with the IDS, initiate and stop analysis, and view real-time predictions. Flask's API endpoints enable seamless communication between the frontend and backend, allowing for efficient data processing and visualization. By integrating Flask with my Conv-LSTM model, monitors the network traffic and receive immediate alerts on potential security threats."

2.4.1 Configuration for Python Flask in Google Colab:

1. Installation: Install Flask and Flask-Ngrok using pip:

```
!pip install flask-ngrok
!pip install Flask==3.0.0 pyngrok==7.1.2
```

2. Ngrok Setup: Use Ngrok to expose your Flask application running on Colab to the public internet. You'll need an Ngrok auth token (obtain one from their website).

```
ngrok_key = "YOUR AUTH KEY"
port = 5000
from pyngrok import ngrok
ngrok.set_auth_token(ngrok_key)
ngrok.connect(port).public_url
```

3. Flask **App:** Create a basic Flask application

```
# Initialize Flask app
app = Flask(__name__, template_folder=template_folder, static_folder=static_folder)
```

4. Create API to connect between the app to the endpoints

/start-prediction to initiate the IDS analysis.
/stop-prediction to halt the analysis.
/get-predictions to retrieve the latest predictions.
/traffic-summary to provide an overview of detected attacks.

```
@app.route('/start-prediction', methods=['GET'])
def start_prediction():
    thread = threading.Thread(target=predict_rows)
    thread.start()
    return jsonify({'message': 'Prediction started.
Monitoring network packets...'})
```

5. Run the App

```
if __name__ == '__main__':
    app.run(port=5000)  # Specify a port if necessary
```

3 Implementation Steps

This code has two sections

- Section 1 of notebook contains code to train and evaluate a two different deep learning models and one hybrid model for classifying network traffic.
- Section 2 has a code to set up a Flask web application to provide a user interface for interacting with trained intrusion detection model.

3.1 Pre-requisites to run the code in Section 1

- Google Colab Environment with python libraries and packages as shown in 3.1.1

- File (Code): uq-ids-Thesis.ipynb
- Data Files:
    - NetFlow_v1_Features.csv': Contains a mapping or description of network flow features.
    - 'NF-UQ-NIDS.csv': Contains the raw network traffic data.
    - 'UQ_IDS_Final_Data.csv': Contains a sampled version of the raw data (created in a previous run).
    - 'final_sampled_data_rough.csv': Contains processed version of the data.

3.1.1 Importing necessary python libraries

Importing Important Libraries

```
import numpy as np         # importing numpy for numerical, array manipulation
import pandas as pd          # importing pandas for data manipulation
import sys                    # importing sys library
import plotly.graph_objects as go    # importing different visualisation libraries
import plotly.io as pio                # library for plotting graphs
import plotly.offline as pyo
import plotly.express as px
import plotly.colors as pc
import matplotlib.pyplot as plt
from plotly.subplots import make_subplots
import seaborn as sns
from sklearn import preprocessing            # importing preprocessing from sklearn
from sklearn.model_selection import train_test_split    # importing library for data split
from sklearn.preprocessing import MinMaxScaler       # importing min max scalar for data normalisation7
from sklearn.preprocessing import LabelEncoder,OneHotEncoder  # importing encoders for data encoding
from collections import Counter                  # importing counter library for counting purpose
import gc

# importing libraries for model building
import tensorflow as tf    # importing tensorflow
import keras      # import keras
from tensorflow.keras.models import Sequential      # importing different required modules from keras and tensorflow
from tensorflow.keras.layers import Dense, Dropout, Activation,Embedding,Flatten,TimeDistributed
from tensorflow.keras.layers import LSTM, Bidirectional,GRU,SimpleRNN
from tensorflow.keras.layers import Embedding,Flatten,Conv1D,MaxPooling1D
from tensorflow.keras.models import load_model, save_model
from tensorflow.keras.models import Model
from keras.models import Sequential
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve,confusion_matrix,precision_score,recall_score,roc_auc_score,accuracy_score,classification_report

#setting some variables and seeding conditions
pd.set_option('display.max_columns', 500)
import warnings                                # importing warnings
warnings.filterwarnings('ignore')
%matplotlib inline
from plotly.offline import init_notebook_mode, iplot # Importing offline plugin of plotly
init_notebook_mode(connected=True)
```

3.2 Loading raw data from file

The raw data contains many records which can be computationally expensive to process. To address this, the code performs random sampling to reduce the dataset size while preserving its statistical properties for efficient analysis and model building.

```
shape of raw data is: (11994893, 15)
```

This line of code is for randomly sampling 2% of the data from the raw_data DataFrame and storing it in a new DataFrame called final_data. Since the original dataset was very large, this was done to reduce the size of the data, which in turn reduces the computational resources required for analysis and model building.

Lodaing Raw Data

```
[53] data_features = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/NetFlow_v1_Features.csv')   # Reading mapping file of the data
     raw_data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/NF-UQ-NIDS.csv')     # reading raw data from file

     print('Data Features are :')    # printing different features in data along with their meaning
     data_features

[55] raw_data.head(10)          # checking the first ten rows of the dataset

[56] print('shape of raw data is:', raw_data.shape )      # Reading shape of raw data file

     shape of raw data is: (11994893, 15)
```

The raw data contains a very large of number of records which is directly proportional to the high resource requirement, thus requires randomly sampling of data to perform further analysis and model building.

```
[57] # final_data = raw_data.sample(frac=0.02)       # sampling 200K record from raw data
     # final_data.to_csv('UQ_IDS_Final_Data.csv', index = False)     # saving sampled into a new csv file
     final_data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/UQ_IDS_Final_Data.csv')     # reading new sampled data file

     final_data.head()    # visualising first five rows of data

[59] final_data['Attack'].unique()      # Checking unique values in target variable

     array(['Benign', 'password', 'DoS', 'DDoS', 'Reconnaissance',
            'Infilteration', 'Brute Force', 'Exploits', 'xss', 'injection',
            'Generic', 'Analysis', 'Bot', 'Backdoor', 'Fuzzers', 'scanning',
            'mitm', 'Theft', 'Shellcode', 'ransomware', 'Worms'], dtype=object)
```

3.3 Data Preprocessing

The original dataset has many unique attack types, leading to a multi-class classification problem. Training a model on such a large number of classes would require a lot of data and computational resources.

```
class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 16 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   IPV4_SRC_ADDR                100000 non-null  object
 1   L4_SRC_PORT                  100000 non-null  int64
 2   IPV4_DST_ADDR                100000 non-null  object
 3   L4_DST_PORT                  100000 non-null  int64
 4   PROTOCOL                     100000 non-null  int64
 5   L7_PROTO                     100000 non-null  float64
 6   IN_BYTES                     100000 non-null  int64
 7   OUT_BYTES                    100000 non-null  int64
 8   IN_PKTS                      100000 non-null  int64
 9   OUT_PKTS                     100000 non-null  int64
 10  TCP_FLAGS                    100000 non-null  int64
 11  FLOW_DURATION_MILLISECONDS   100000 non-null  int64
 12  Label                        100000 non-null  int64
 13  Attack                       100000 non-null  object
 14  Dataset                      100000 non-null  object
 15  Attack_Category              100000 non-null  object
dtypes: float64(1), int64(10), object(5)
memory usage: 12.2+ MB
```

To deal with this, the code maps the different attack types to broader domain categories, such as "Network Attacks" and "Exploitation Attacks". This reduces the number of classes and simplifies the problem, making it more manageable with the available resources. This is achieved using the categorize attack function, which maps each attack type to its corresponding category. For example, attacks like 'DDoS', 'DoS' are categorized as 'Network Attacks', and attacks like 'injection', 'Exploits' are categorized as 'Exploitation Attacks'.

```
Numerical Features Count 11
['L4_SRC_PORT', 'L4_DST_PORT', 'PROTOCOL', 'L7_PROTO', 'IN_BYTES', 'OUT_BYTES', 'IN_PKTS',
'OUT_PKTS', 'TCP_FLAGS', 'FLOW_DURATION_MILLISECONDS', 'Label']
Categorical Features Count 3
['IPV4_SRC_ADDR', 'IPV4_DST_ADDR', 'Attack_Category']
```

- Conversion of IP addresses to decimal values.

3.4 Data Analysis and Visualisations

The data analysis section aims to visually explore the distribution of individual features and their relationship with the target variable (Attack_Category) is generated through the code.

a. Histogram plot of IPV4_SRC_ADDR vs Attack Category
b. Histogram Plot of IPV4_SRC_ADDR vs Attack Category
c. Bar plot of TCP Flag Column
d. Sunburst plot of TCP Flag Column vs Attack Category
e. Pie plot of Network Protocols
f. Sunburst plot of Network Protocols vs Attack Category
g. Pie plot of IN_BYTES and OUT_BYTES distribution vs Attack Category

3.5 Feature Engineering

3.5.1 Feature Selection

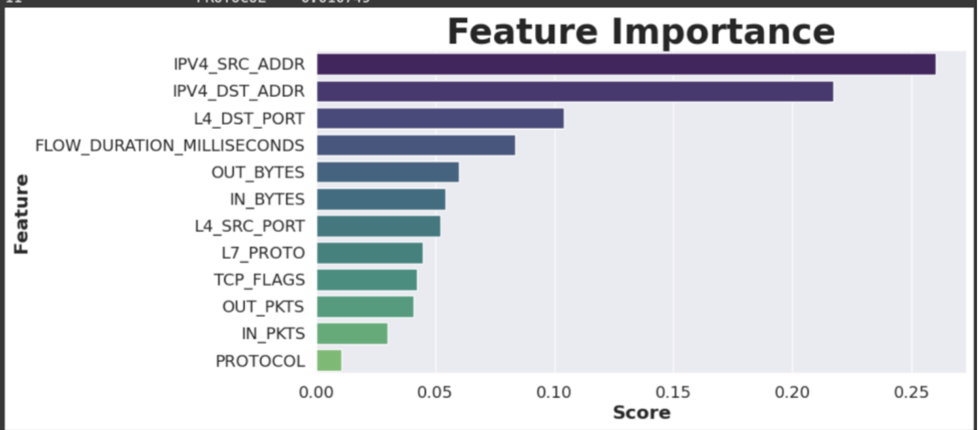
- Balance the dataset using SMOTE oversampling.
- Apply OneHotEncoder to the target variable.
- Scale numerical features using MinMaxScaler.

```
The number of classes before fit Counter({0: 44238, 2: 32029, 1: 22579})
The number of classes after fit Counter({0: 44238, 2: 44238, 1: 44238})
```

3.5.2 Feature Extraction

- Uses Random Forest model to determine feature importances.
- Select the top 8 features based on importance.
- Splitting the data into training and testing sets
- Uses a test_size of 0.2, 20% of the data will be used for testing and remaining 80% of the data will be used for training.

```
                      Feature  Importance
0               IPV4_SRC_ADDR    0.260217
1               IPV4_DST_ADDR    0.217328
2                 L4_DST_PORT    0.103950
3   FLOW_DURATION_MILLISECONDS    0.083553
4                   OUT_BYTES    0.059991
5                    IN_BYTES    0.054143
6                 L4_SRC_PORT    0.052076
7                    L7_PROTO    0.044942
8                   TCP_FLAGS    0.042279
9                    OUT_PKTS    0.040860
10                    IN_PKTS    0.029911
11                   PROTOCOL    0.010749
```



3.6 Model Implementation

3.6.1 CNN Model

With 10 filters, a kernel size of 1, and ReLU activation. A dropout layer is then added to prevent overfitting.

```
CNN Model Accuracy: 83.46%
CNN Model Precision: 0.8334
CNN Model Recall: 0.8346
CNN Model F1_score: 0.8304
```

3.6.2 LSTM Model

A batch normalization layer is included to stabilize training, followed by an LSTM layer with 12 units, configured to return sequences for potential stacking with more layers.

```
LSTM Model Accuracy: 81.49%
LSTM Model Precision: 0.8414
LSTM Model Recall: 0.8149
LSTM Model F1_score: 0.8105
```

3.6.3 Conv-LSTM Model

Firstly, reshapes the data to have the following dimensions: (samples, time steps, rows, cols, features). and LSTM layer with 12 units, configured to return sequences for potential stacking with more layers.

```
Conv-GRU Model Accuracy: 86.76%
Conv-GRU Model Precision: 0.8659
Conv-GRU Model Recall: 0.8676
Conv-GRU Model F1_score: 0.8665
```

3.7 Web Framework with a Best Model Chosen

Based on the above metrics (accuracy, precision, recall, F1-score), the Conv-LSTM model generally outperforms the CNN and LSTM models suggests that the combination of local feature extraction (CNN) and temporal modelling (LSTM) is most effective for this intrusion detection task. The Conv-LSTM likely captures both the subtle patterns within individual flows and the broader temporal dependencies across multiple flows that characterize different attack types.

3.7.1 Pre-requisites to run the code

- Google Colab Environment with desired libraries and packages
- File (Code): uq-ids-Thesis.ipynb
- Webpage file: index.html
- Data Files:
  - Trained Model: Ensure you have a trained deep learning model saved as a HDF5 file (/content/drive/MyDrive/Colab Notebooks/Best_Model_Conv_LSTM.h5) in the specified location.
  - Data for Prediction: A CSV file (/content/drive/MyDrive/Colab Notebooks/Pred_data.csv) containing network traffic data that want to analyse for intrusions.

1. Install Necessary Libraries:

```
!pip install flask-ngrok
!pip install Flask==3.0.0 pyngrok==7.1.2
```

2. Set Up Ngrok Account and Authentication Token:

```
ngrok_key = "2[redacted]Y8m"
port = 5000
from pyngrok import ngrok
ngrok.set_auth_token(ngrok_key)
ngrok.connect(port).public_url
```

```
'https://6382-34-132-5-187.ngrok-free.app'
```

3. Run the Flask App

Execute the second Flask code block in notebook which will start the Flask app and make it accessible through a public URL provided by ngrok.

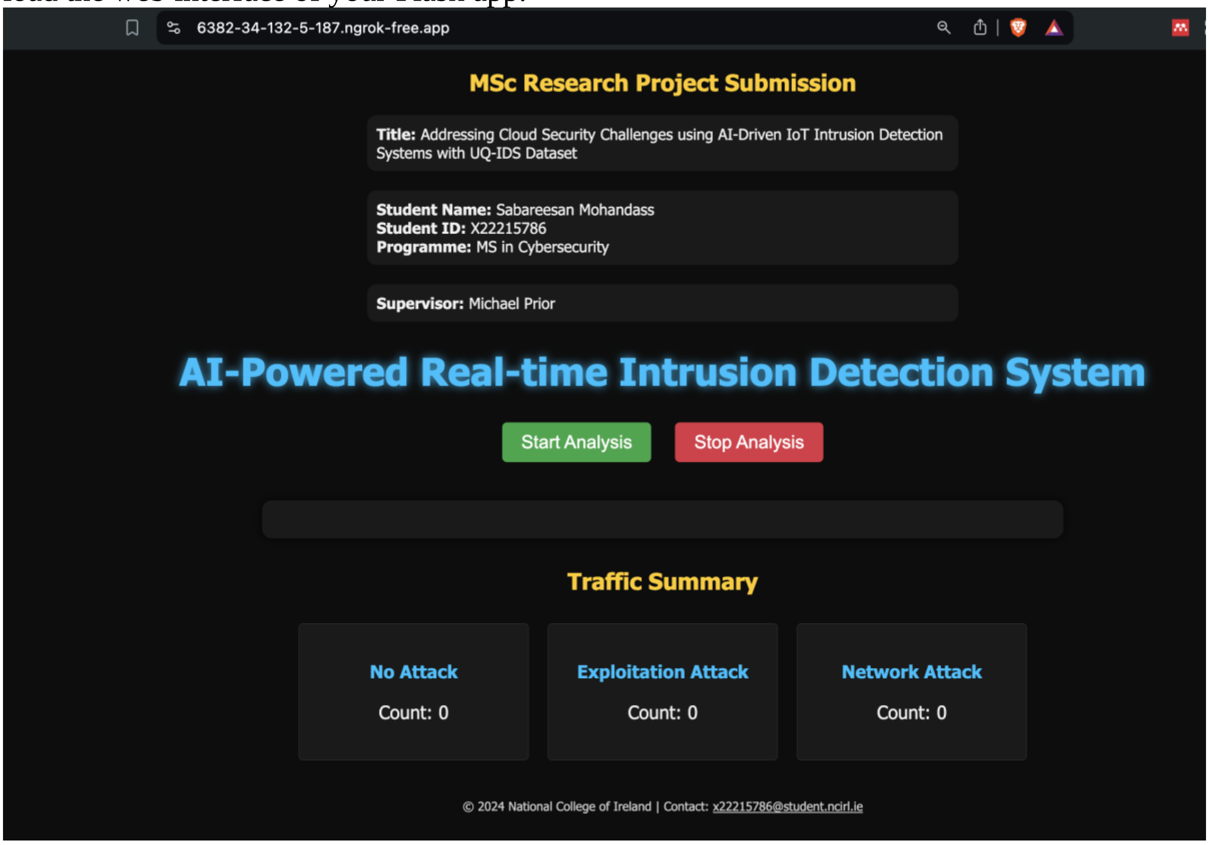
```python
from flask import Flask, render_template, jsonify
from flask_ngrok import run_with_ngrok
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
import threading
import time
from datetime import datetime
import random
from IPython.display import display, HTML
import json
import os

template_folder = '/content/drive/MyDrive/Colab Notebooks/templates'
static_folder = '/content/drive/MyDrive/Colab Notebooks/static'


# Initialize Flask app
app = Flask(__name__, template_folder=template_folder, static_folder=static_folder)
```

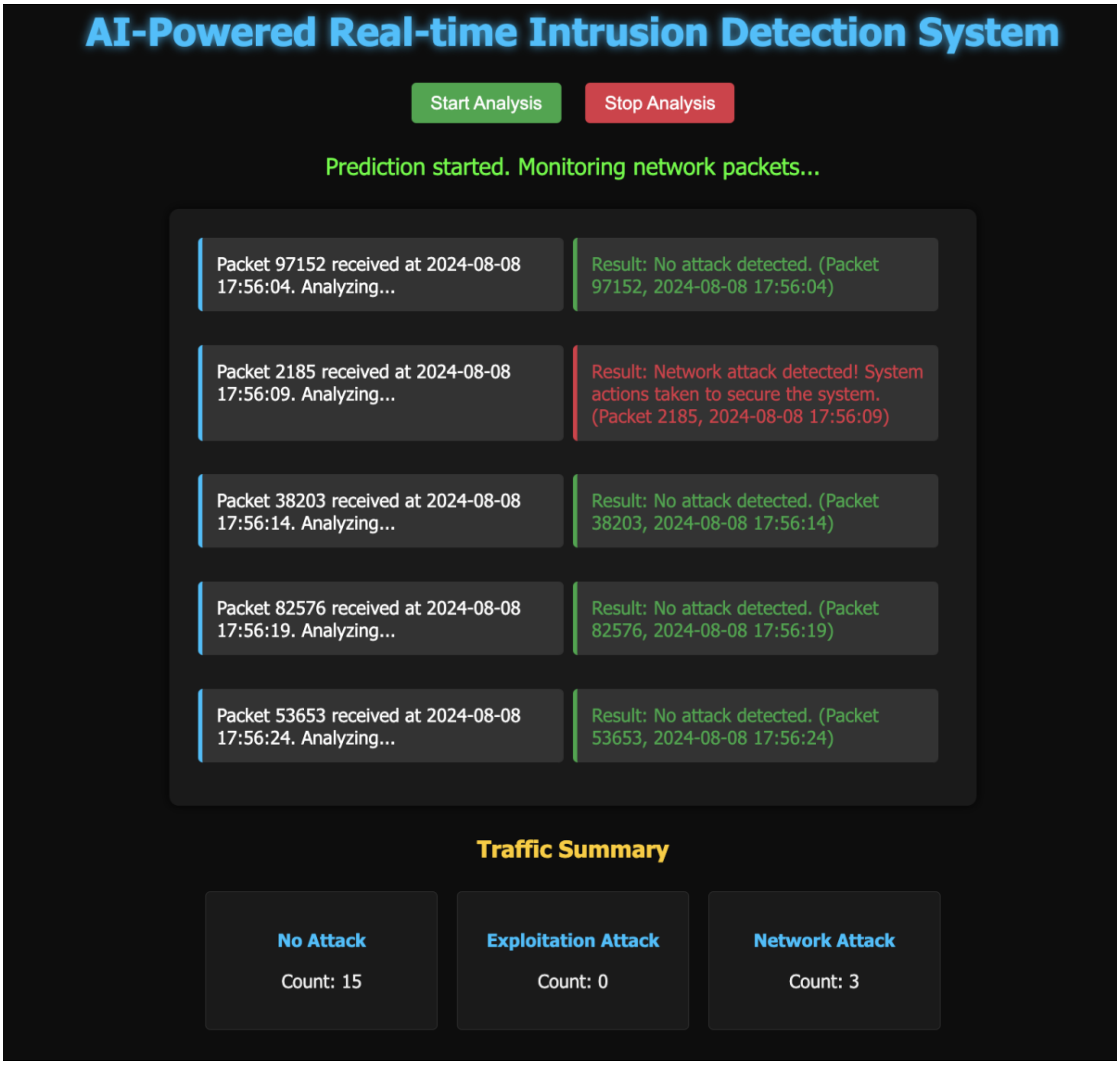
4. Access the Web Interface:

Open a web browser and paste the public URL generated by ngrok above in step 2 and this will load the web interface of your Flask app.

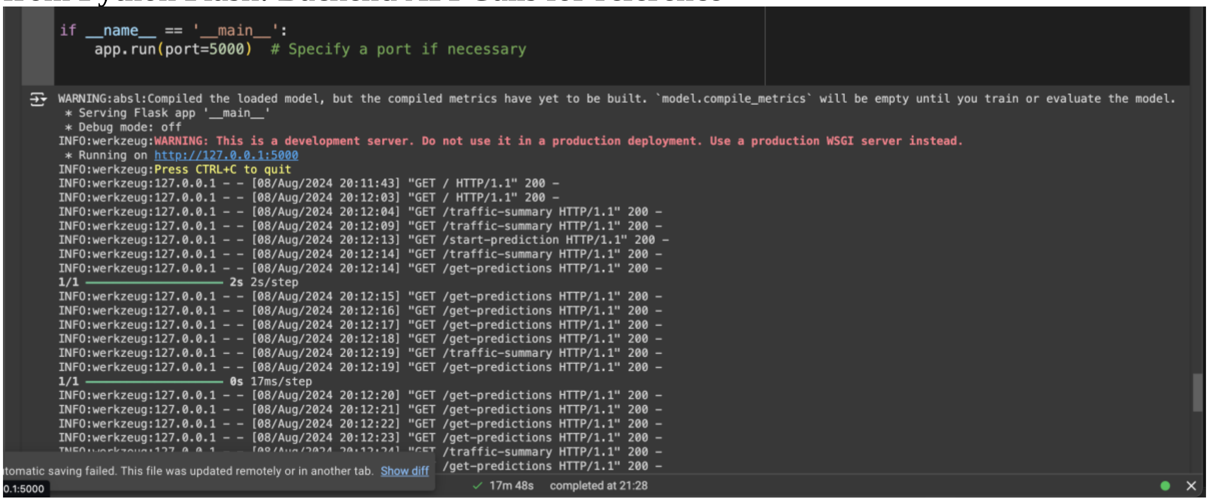


5. Interact with the App:

Use the buttons and visualizations in the web interface to start/stop predictions, view real-time results, and analyze the network traffic summary.



from Python Flask: Backend API Calls for reference




```
End*****************
```