National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Jisha Joy
Student ID: 21240868

School of Computing
National College of Ireland

Supervisor: Dr. Shivani Jaswal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jisha Joy
Student ID:	21240868
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Shivani Jaswal
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	649
Page Count:	6

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:	Jisha Joy
Date:	13th December 2023

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

Jisha Joy
21240868

1 Introduction

This document describes the requirements and steps to be performed for replicating the CNN-LSTM with Attention Mechanism(CLAM) model for DNS-based data exfiltration detection in cloud networks. It specifies the required tools, data processing, configuration steps, and specific packages required in the model development.

2 Requirements

The main requirements for the model development are:

- **Jupyter Notebook:** Jupyter Notebook is an interactive platform used for writing machine-learning codes and also used as a visualization platform for the developed models. Jupyter Notebook is used as a local platform for model development.
- **Anaconda:** Anaconda is an open-source package environment for Python with different packages and tools required to work on data. The Jupyter Notebook is accessed from within the Anaconda Navigator.
- **Google Colab:** Google Colab is a cloud-hosted platform that allows the development and visualization of a machine-learning model like Jupyter Notebook. It provides computing resources like T4 GPU for running the machine learning code in the cloud. Also, it reduces the effort of setting up the environment and has the libraries and packages for machine learning models preinstalled.

3 Specification

The specifications of the system required for developing the model are given below:

3.1 Hardware

The model is developed locally on a Windows 10 machine. The hardware specification of the system used for the development of the model is shown in figure 1

Figure 1: Hardware Specifications for model development

The specification of the Google Colab platform used for hosting the model in the cloud platform is given in figure 2

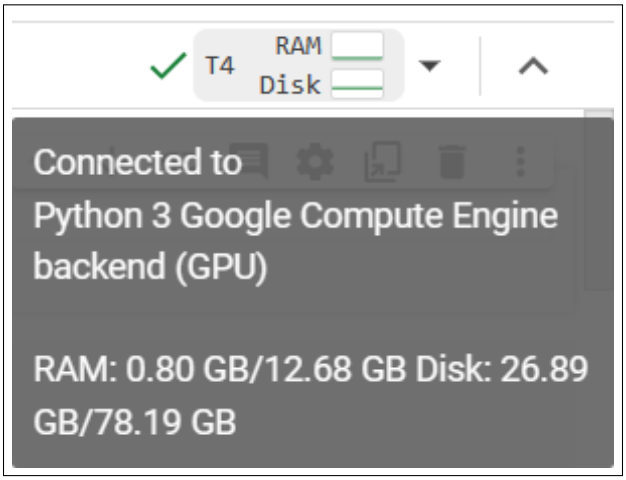

Figure 2: Hardware Specifications of Google Colab platform

3.2 Software Specification

1. Anaconda Navigator
2. Python 3
3. Jupyter Notebook 6.5.2
4. Google Colab with T4 GPU

4 Python Libraries Used

1. **Pandas:** Pandas is an open-source library in Python used for analyzing and performing manipulation on datasets.
2. **Matplotlib:** Matplotlib is a Python visualization library used for data visualization. This enables the plotting of graphs to represent the results of the machine-learning model.
3. **Tensorflow:** Tensorflow is an open-source library used for performing numerical computations on the dataset.
4. **Numpy:** Numpy is also an open-source Python library used for numerical calculations especially array and matrix-based tasks.
5. **Seaborn:** Seaborn is also a visualization tool used for plotting statistical graphs representing the results of the machine learning model.

5 Installation

The local and cloud-based installation steps are mentioned below:

5.1 Local Installation

1. Install Anaconda Navigator for Windows 10
2. Open Jupyter Notebook from the different tools available in Anaconda Navigator
3. Select Python kernel for running the notebook
4. Install the required packages and libraries
    - *pip install pandas*
    - *pip install matplotlib*
    - *pip install scikit-learn*
    - *pip install –upgrade scikit-learn*
    - *pip install scikit-learn==1.0*
5. Upload the dataset

5.2 Cloud Installation

1. Access Google Colab website
2. Open a new Jupyter Notebook
3. Connect to T4 GPU runtime
4. Upload the dataset to Google Drive
5. Grant permission to Colab for accessing Google Drive

6 Data Preprocessing

Data required for the research is obtained from the UNB website as an NSL-KDD dataset.
`https://www.unb.ca/cic/datasets/nsl.html`

Python code for preprocessing the dataset is shown in figure 3

```
[ ] from sklearn.preprocessing import LabelEncoder

    le_list = [LabelEncoder() for i in range(0,full_data.shape[1])]

    for i in range(0,full_data.shape[1]):
        if full_data.dtypes[i]=='object':
            full_data[i] = full_data[i].astype('string')
            le_list[i].fit(full_data[full_data.columns[i]])
            full_data[full_data.columns[i]] = le_list[i].transform(full_data[full_data.columns[i]])
```

Figure 3: Data Preprocessing for model development

7 Model Development

The code snippet for the CLAM training model development with 10 epochs is shown in figure 4

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv1D, LSTM, concatenate, Dense, Attention, GlobalAveragePooling1D, Dropout, MaxPooling1D, GRU

# Input layer
input_layer = Input(shape=(41, 1))

# 1D CNN branch
cnn_branch = Conv1D(64, kernel_size=3, activation='relu')(input_layer)
cnn_branch = MaxPooling1D()(cnn_branch)
cnn_branch = GlobalAveragePooling1D()(cnn_branch)

# LSTM branch
lstm_branch = LSTM(128, activation='tanh', return_sequences=True)(input_layer)
lstm_branch = Attention()([lstm_branch, lstm_branch])
lstm_branch = GlobalAveragePooling1D()(lstm_branch)
lstm_branch = tf.keras.layers.Reshape((128,))(lstm_branch)

# Concatenate the outputs from both branches
merged = concatenate([cnn_branch, lstm_branch])
```

```
# Dense layers with attention mechanism
attention = Dense(128, activation='tanh')(merged)
attention = tf.keras.layers.Reshape((1, 128))(attention)
attention = tf.keras.layers.Dense(1, activation='softmax')(attention)
attention = tf.keras.layers.Flatten()(attention)
attention = tf.keras.layers.RepeatVector(128)(attention)
attention = tf.keras.layers.Permute([2, 1])(attention)
attention = tf.keras.layers.Reshape((128,))(attention)
# Apply attention to merged features
merged_attention = concatenate([merged, attention])
dense_1 = Dense(64, activation='relu')(merged)
drop_1 = Dropout(0.1)(dense_1)
dense_2 = Dense(32, activation='relu')(drop_1)
drop_2 = Dropout(0.1)(dense_2)
dense_3 = Dense(32, activation='relu')(drop_2)
drop_3 = Dropout(0.1)(dense_3)
# Output layer for classification
output_layer = Dense(n_outputs, activation='softmax')(dense_3)

# Create the model
model = tf.keras.Model(inputs=input_layer, outputs=output_layer)

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 4: CLAM training model with 10 epochs

8 Comparison

The developed model was evaluated using the following models:

8.1 Basic Classification Models

The training accuracy of the classification models with the dataset was evaluated using AutoML technique as shown in figure 5.

```
# import ClassificationExperiment and init the class
from pycaret.classification import ClassificationExperiment
exp = ClassificationExperiment()
```

```
# compare baseline models
best = compare_models(include=['rf', 'dt', 'gbc', 'knn', 'xgboost', 'nb', 'lr', 'svm'])
```

```
evaluate_model(best)
```

Figure 5: Training basic classification models

8.2 Deep Learning Model

The training accuracy of a Convolutional Neural Network(CNN) and Long Short Term Memory(LSTM) with the dataset is evaluated as represented in figure 6.

```
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout,Conv1D,MaxPooling1D,Flatten
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=4, activation='relu', input_shape=(41,1)))
model.add(Conv1D(filters=32, kernel_size=4, activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout,LSTM,MaxPooling1D,Flatten
model = Sequential()
model.add(LSTM(6,return_sequences=True,input_shape=(41, 1)))
model.add(Dropout(0.5))
model.add(LSTM(4))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Figure 6: Training CNN and LSTM models

References

https://www.tensorflow.org/tutorials/images/cn

https://www.geeksforgeeks.org/ml-handling-missing-values/?ref=lbp