Kai Kit Lok (Brian) - 301108938
Kevin Estrada - 216179224
(Team Lead)Cody Wuco -301090621

CSC 180.02 - Intelligent System

Project2 : Network Intrusion Detection System

Due Date: October 14, 2021

Problem Statement

In this project, we aim to build an AI-based Network Intrusion Detection System, a predictive model distinguishing between attack and normal connections.

Methodology

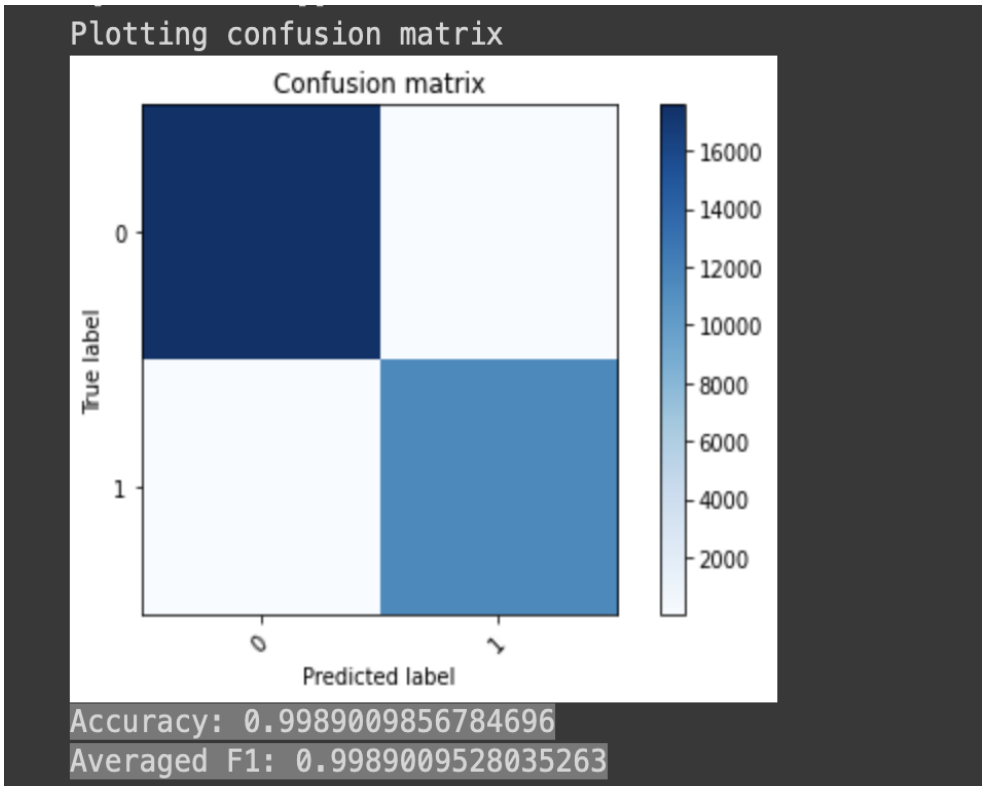
A dataset contains a wide variety of intrusion simulated in a military network environment. It has the connection detail from a source IP address to a target IP address under some well-defined protocol.

We use the classified connection from the dataset to train the AI model. To accomplish that, we encode all the numeric and categorical features. We used Fully-Connected Neural Networks and Convolutional Neural Networks networks and early stopping for the training of the data. We also tried different amounts of neurons, learning methods and different optimizers to get accurate scores. We processed the data encoding good connections as ‘0’ and attacks as ‘1’.

Tools: pandas, numpy, sci-learn, tensorflow, Jupyter notebook, google colab

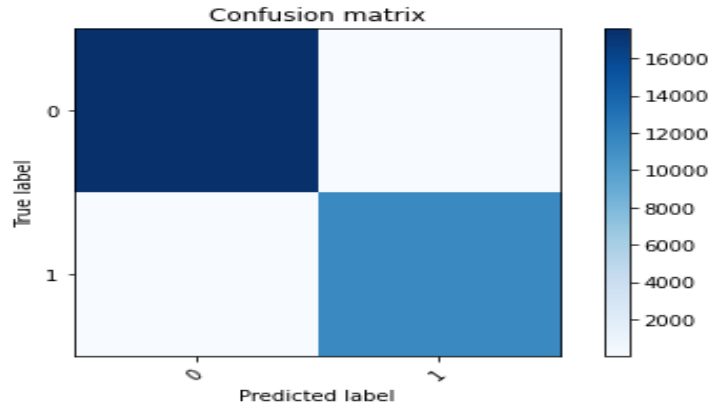
Experimental Results and Analysis

Convolutional Neural model



Fully-Connected model

```
Plotting confusion matrix
```




```
Accuracy: 0.9987979530858262
Averaged F1: 0.9987979081315314
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     17589
           1       1.00      1.00      1.00     11528

    accuracy                           1.00     29117
   macro avg       1.00      1.00      1.00     29117
weighted avg       1.00      1.00      1.00     29117
```

The prediction was very accurate, as you can see from the confusion matrix. The precision, recall, and f1-score are the highest possible numbers.

The difference between the 2 model types accuracy was negligible. The only differing fact seemed to be the speed in which the model finished training. The convolution model seemed to perform the tasks at a noticeable slower pace.

Hyper Parameter tuning

Convolutional Neural Network

Each run had a 5 model loop.

Relu, Adam, 2 Kernels (32, 64 neurons), Kernel size 3, 1 Layer 1000 neurons

HH:MM:SS	Adam	SGD
Sigmoid	5m 47s	N/A

Tanh	1m 34s	N/A
Relu	1m 36s	8m 15s

Summary:

Relu, Adam, 2 Kernels (32, 64 neurons), Kernel size 3

HH:MM:SS	1 Hidden Layers	2 Hidden Layers
10 Neurons per layer	1 m 37s	1 m 56 s
500 Neurons per layer	1 m 38 s	1 m 26s
1000 Neurons per layer	1m 36s	1 m 42s

Relu, Adam, Kernel size 3, 1 Layer 1000 neurons

HH:MM:SS	1 Kernel	2 Kernels
32, 64 Neurons	1m 16s	1m 36s
128, 64 Neurons	1m 57s	1m 54s
500, 250 Neurons	2m 11s	7m 51s

Relu, Adam, 2 Kernels (32, 64 neurons), 1 Layer 1000 neurons

HH:MM:SS	2 Kernels
Size 2	1m 56s
Size 3	1m 36s
Size 5	1m 37s

Fully-Connected Neural Network

Each run had a 5 model loop.

100 neurons in the first layer, 500 in the second layer

HH:MM:SS	Adam	SGD
Sigmoid	1 m	2 m

Tanh	47 s	1 m
Relu	41 s	1 m

:Sigmoid seems has poorer efficiency than other activations and optimizers

Relu, Adam, and 100 Neurons on first layer, 500 on second layer

HH:MM:SS	1 Hidden Layers	2 Hidden Layers
10 Neurons per layer	41 s	57 s
100 Neurons per layer	43 s	31 s
500 Neurons per layer	41 s	54 s

The efficiency seems to change almost randomly with changes in neurons and layers.

Task Division and Project Reflection

(Brian) Was in charge of building and tabulating the Fully connected model
(Kevin) Was in charge of tabulating the CCN model and building the confusion matrix
(Cody) Was in charge of building the CNN model and data processing

All other responsibilities were pair processed and passed around as we found that others had more time or a better grasp on the ideas.

We were having issues on training the model, because we did not handle the data that is missing value after encoded columns properly. We did not realize that we weren't updating the dataframe to retain the columns that we were dropping, so when we moved forward, we noticed that we were getting NAN value losses and the accuracy did not improve, which caused us to believe that our normalizations functions weren't working, however it was the problem mentioned previously.

To make our model more analyzable, we can know what feature has more effect on the attack connection by regularizing the numeric data to a logistic regression model. The coefficient for each feature can be observed from the model. The higher coefficient is, the more that feature has an effect on it.

The prediction accuracy is surprisingly high. One of the reasons might be because this is a binary classification problem. The result either is one or another. Another reason could be the features to determine an attack connection are very specific.

Since it is our second time using google collabs it was easier to communicate and run things together. Due to having the knowledge of using the tool previously. Communication was improved and helped when trying to schedule things easier.