CSC 180

Project 2

Team Members:

Santiago Bermudez, ID: 301118090

Amad Shah, ID: 301753101

John Kieren, ID: 301144467

Due: 10/14/22

(1) Problem Statement

We will need to be able to detect network intrusions in order to protect a computer network from unauthorized users, including insiders. We will also need to be able to create a predictive model that can distinguish between bad connections, called intrusions or attacks, and good connections.

(2) Methodology

To begin, we would start off our project by creating a shared Google Colab so that we could all work the programming for this project together. We would obviously have to download the 10% subset of the network intrusion dataset and upload that into our colab. We would then go through an identical process as the last project for the first few steps. We would first label all of the columns before working with our dataset. This time around, we would also add headers to split our work into sections so that we don't have to scroll around as much. We would then run some useful functions, make a copy of our data frame and then check it for issues. We then go through data preprocessing by removing duplicate values, processing any missing values, and then perform some encoding and normalization on our dataframe as needed. We would encode categorical features and then normalize numeric features. We would also encode good connections as 0 and attacks as 1 for our outputs for future testing. After normalization, we would double check for missing values and get rid of columns that are empty as a result of the normalization. We would then split our data into training and testing subsets. After that, we would start testing with fully-connected neural networks. The process is similar to the last project, but this time we have to do binary classification. We would vary hyperparameters and do early stopping. Rather than use RMSE, we would capture the accuracy, precision, recall, and F1 scores of our predictions. We would also get the confusion matrix and use that to evaluate our fully-connected neural network models. We would also follow a similar process for our convolutional neural networks, but also play around with kernel numbers and sizes apart from the other usual hyperparameters.

(3) Experimental Results and Analysis

For our training/testing, we decided to compare the scores of fully-connected neural networks and convolutional neural networks within their own respective groups.m Interestingly enough, the best model we got was from convolutional neural networks, although performance between the two models seemed to be similar with slight variances depending on which hyperparameter we altered.

For Fully-Connected Neural Networks:

Activation	RMSE	Accuracy	Precision	Recall	F1

Relu (*Standard)	0.04375985637307167	0.9976921174822101	0.9976920440758239	0.9976921174822101	0.9976919266402202
Sigmoid	0.04273755103349686	0.9978020166497239	0.997803541628824	0.9978020166497239	0.9978023265096002
Tanh	0.9503727555274963	0.6025771354781988	0.36325842098652755	0.6025771354781988	0.4532680895650783

Layers Count	RMSE	Accuracy	Precision	Recall	F1
Added Layers	0.14751118421554565	0.9775256202434267	0.9780955621301518	0.9775256202434267	0.9774226840274512
Reduced Layers	0.04744195193052292	0.9975547435228178	0.9975582503804307	0.9975547435228178	0.9975553024658789

Neurons Count	RMSE	Accuracy	Precision	Recall	F1
Added Neurons	0.05380035191774368	0.9965381762233151	0.996543737940787	0.9965381762233151	0.9965366499166715
Reduced Neurons	0.04507247358560562	0.997499793939061	0.9975008514340395	0.997499793939061	0.9974992457138985

Optimizers	RMSE	Accuracy	Precision	Recall	F1
sgd	0.05307013913989067	0.9967854493502212	0.9967863698559878	0.9967854493502212	0.9967857329534472
adam	0.04273755103349686	0.9978020166497239	0.997803541628824	0.9978020166497239	0.9978023265096002

The **best RMSE score** we got for fully-connected neural networks was from the model:

Model	RMSE	Accuracy	Precision	Recall	F1
Sigmoid, normal layers and neurons count, and adam optimizer	0.04273755103349686	0.9978020166497239	0.997803541628824	0.9978020166497239	0.9978023265096002

For Convolutional Neural Networks:

Activation	RMSE	Accuracy	Precision	Recall	F1
Relu (*Standard)	0.037512030 45248985	0.998152282 1719271	0.998153647 4264716	0.998152282 1719271	0.998151871 8685713
Sigmoid	0.048625554 889440536	0.997231857 6776453	0.997234680 8714332	0.997231857 6776453	0.997232400 7893386
Tanh	0.059291720 390319824	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202

Layers Count	RMSE	Accuracy	Precision	Recall	F1
Added Layers	0.050973318 5172081	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202
Reduced Layers	0.037348534 911870956	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202

Neurons Count	RMSE	Accuracy	Precision	Recall	F1
Added Neurons	0.040407374 50122833	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202
Reduced Neurons	0.044604625 552892685	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202

Optimizers	RMSE	Accuracy	Precision	Recall	F1
sgd	0.050973318 5172081	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202
adam	0.037512030 45248985	0.998152282 1719271	0.998153647 4264716	0.998152282 1719271	0.998151871 8685713

Kernel Count	RMSE	Accuracy	Precision	Recall	F1
Added kernels	0.037234861 40370369	0.995178074 6642854	0.995185044 1692354	0.995178074 6642854	0.995175541 2205202

Reduced kernels	0.04670027643442154	0.9951780746642854	0.9951850441692354	0.9951780746642854	0.9951755412205202

Kernel Size	RMSE	Accuracy	Precision	Recall	F1
Bigger kernels	0.04301976040005684	0.9951780746642854	0.9951850441692354	0.9951780746642854	0.9951755412205202
Smaller kernels	0.0479968786239624	0.9951780746642854	0.9951850441692354	0.9951780746642854	0.9951755412205202

The **best RMSE score** we got for convolutional neural networks was from the model:

Model	RMSE	Accuracy	Precision	Recall	F1
Big kernel count, relu, adam, big layer count, normal neuron count, and normal kernel size	0.03723486140370369	0.9951780746642854	0.9951850441692354	0.9951780746642854	0.9951755412205202

(4) Task Division and Project Reflection

-Who is responsible for which part:

Task:	Name:
(5 pts) Split data to training and test.	Amad Shah
(20 pts) Use the following models to detect bad connections (intrusions). Compare their recall, precision and F1-score. PLOT the confusion matrix for each model. - Fully-Connected Neural Networks - Convolutional Neural Networks (CNN)	John Kieren, Santiago Bermudez

(5 pts) Drop redundant rows. Remove all records with missing values.	Amad Shah
(10 pts) Encode categorical features and normalize numeric features.	Amad Shah
(5 pts) Some columns may have missing values after you normalize them. Handle those columns appropriately.	Amad Shah
(5 pts) Use EarlyStopping	John Kieren, Santiago Bermudez
(20 pts) Vary the following hyperparameters to record how they affect model performance in your report. Tabulate your findings. - Activation: relu, sigmoid, tanh - Layers and neuron counts - Optimizer: adam and sgd - Kernel number and kernel size (for CNN only)	John Kieren, Santiago Bermudez
(10 pts) Your report includes the following sections: - Problem Statement - Methodology - Experimental Results and Analysis	Everyone
(10 pts) Your report includes the following section: - Task Division and Project Reflection	Everyone
(10 pts)Your report should have at least more than one page.	Everyone

-Challenges your group encountered and how you solved them:

- Our first real challenge was switching from regression to classification for this project. It really was just a minor inconvenience and caused a small delay in our project.

- Another issue we had was working with convolutional neural networks. For starters, we had a hard time figuring out how to convert non-image data into a form of image data that we can work with for this model.
- We also had been stuck for some time on understanding the parameters for convolutional neural networks, what it was we were supposed to enter for them, and how to calculate the right values.

-What you have learned from the project as a team:

This project went by rather smoothly compared to the first one. For this project we had an easier time handling data preprocessing such as eliminating redundant rows and columns, encoding certain features, and normalizing other features. When it came to splitting our data into training and test sets, that went easier as well. For fully-connected neural networks, we already had experience with that, but we had to adjust a bit to get familiar with classification and switch over from regression. Apart from learning and working with classification for the first time, we had to learn how to use convolutional neural networks, which is a type of network most well suited for working with images. Obviously, our challenge was learning how to represent non-image data as a form of image data in order to be able to work with this model. This was probably the hardest part of the project and a hurdle that we are glad we overcame.