Mini-Project 2: Network Intrusion Detection

Ronngguang ou (219817313)
John Leonardo (216649967)
Manuel Herrera (219721880)
CSC 180 Intelligent Systems
Mini-Project 2
Friday, October 11, 2019

Problem Statement

Using the KDD Cup 1999 Data of a wide variety of intrusions simulated in a military network environment, are we able to to build a network intrusion detector? The goal is to make a detector model that is able to distinguish between regular and 22 attack type connections.

Methodology

We imported an external library containing necessary functions that were both provided and created to preprocess and display our data.

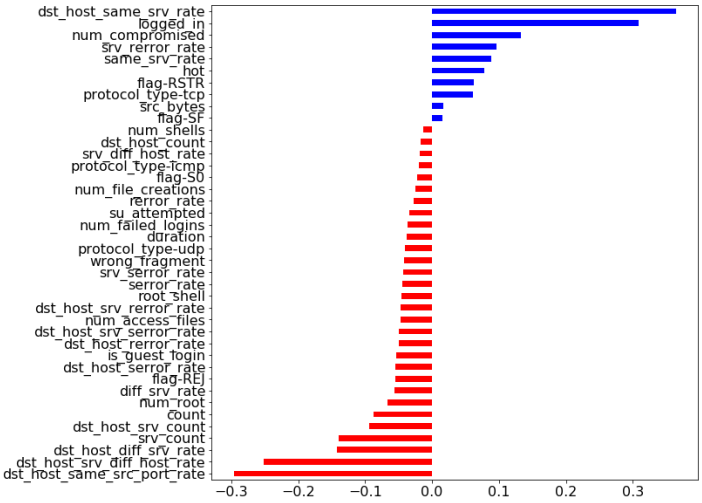
We retrieved the network intrusion data from the KDD Cup 1999 dataset provided by the University of California, Irvine (UCI). For computation reasoning, we are using the 10% subset data provided by UCI. We converted our csv file into a dataframe for preprocessing while simultaneously assigning columns names to each feature. Due to the existence of a large number of redundant records within the dataset, it was imperative we removed duplicate records, otherwise, our model would have been biased towards frequent records. Doing so decreased our record counts from 494,021 to 145,586 unique ones.

Thereafter, our grab_important_feature function call handled a variety of tasks. It dropped the “service” feature because we decided it had no prediction power. Next, it encoded the categorical features “flag” and “protocol_type” and the remaining data was passed to a logistic regression through the L2 regularizer to compute the coefficients per feature. The regularizer returned a dataframe with 34 features greater than or equal to 0.015. At last, it prepared both “x” and “y” from our outcomes feature and split it into training and testing sets.

We created a function called balance_it that balanced our data by removing attack types with lower than 100 counts and reducing any connections to be within a reasonable range of each other. The decision was made for our final model implementation and testing to use a range within 100-1000.

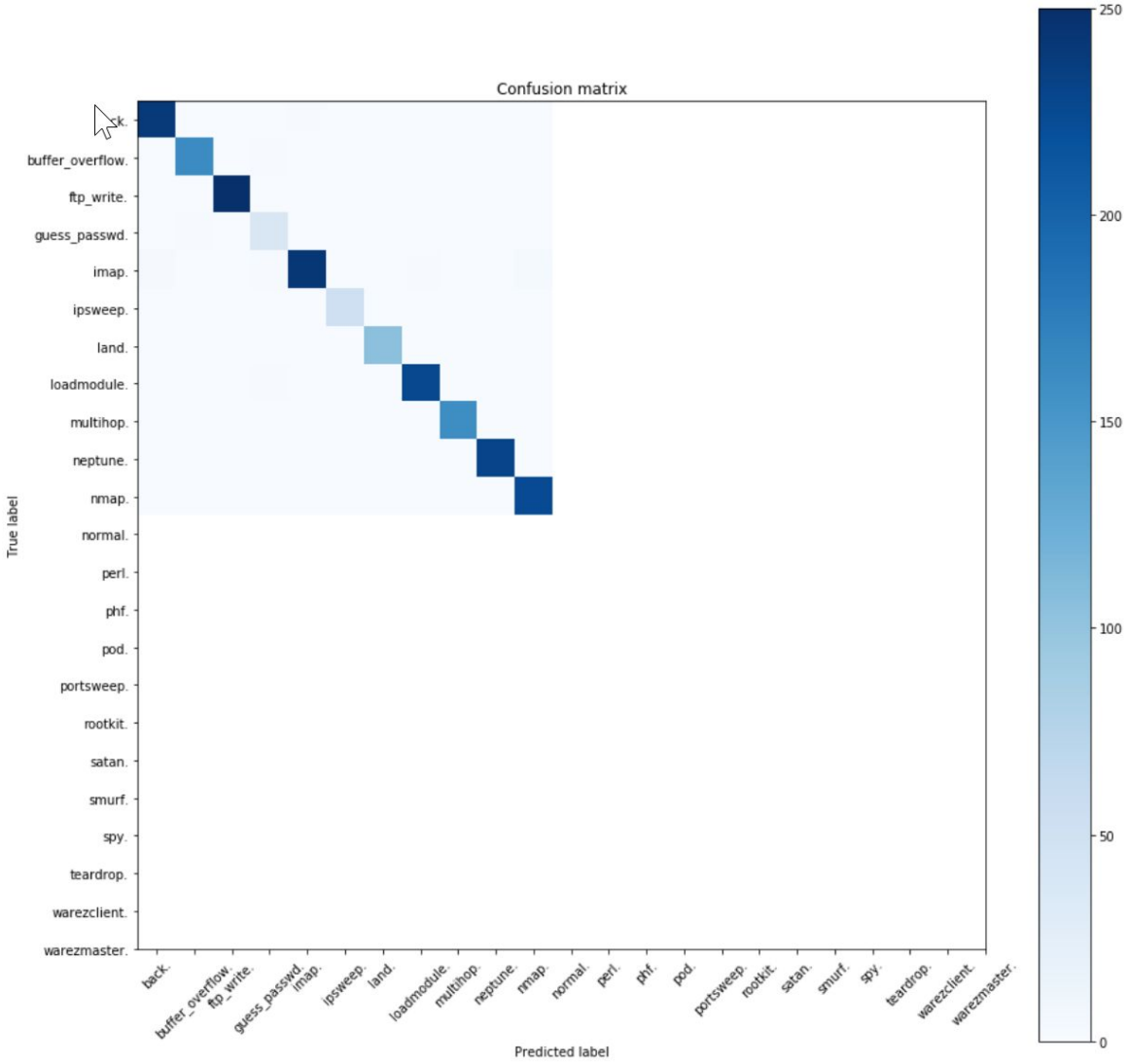
Experimental Results and Analysis

Using the logistic regression function through the L2 regularizer, we analyzed the features which met our set threshold, excluding the outcome feature. After experimenting with the threshold, we deemed a value that returned a dataframe with 34 features for further testing. We ran counts on the outcome feature to determine which were deemed unnecessary because testing them would return f1-scores of 0.00. Furthermore, our experiments were unbalanced and therefore we truncated a single type of attack and a normal connection to keep our values within a reasonable range. We used this final outcome collection to run our model using relu as our activation parameter when adding layers and used categorical crossentropy for this multiclass classification when compiling our model with the sgd optimizer. After training our model through 20 iterations, our neural network came resulted with the following values for precision, recall, and f1-score:




```
outcome
0     968
5     651
9     1000
10    158
11    1000
14    206
15    416
17    906
18    641
20    918
21    893
Name: outcome, dtype: int64
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	242
1	1.00	0.96	0.98	163
2	1.00	1.00	1.00	250
3	0.86	0.95	0.90	39
4	0.97	0.98	0.98	250
5	0.98	0.98	0.98	52
6	0.99	1.00	1.00	104
7	0.99	0.99	0.99	227
8	0.99	1.00	0.99	160
9	1.00	1.00	1.00	230
10	0.99	0.99	0.99	223
accuracy			0.99	1940
macro avg	0.98	0.98	0.98	1940
weighted avg	0.99	0.99	0.99	1940



In order to attempt the convolutional neural network, we had to create an image of arrays, labeled as x. We assigned x as a 4D array with proper depth declaration of 1. Our model summary is as follows:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 1, 8, 32)	128
conv2d_1 (Conv2D)	(None, 1, 6, 64)	6208
max_pooling2d (MaxPooling2D)	(None, 1, 3, 64)	0
dropout (Dropout)	(None, 1, 3, 64)	0
flatten (Flatten)	(None, 192)	0
dense_3 (Dense)	(None, 128)	24704
dropout_1 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 2)	258

Total params: 31,298
Trainable params: 31,298
Non-trainable params: 0

After tuning our parameters for our CNN, it was apparent increasing both the kernel size and stride would decrease performance.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	251
1	0.75	0.86	0.80	7
2	0.00	0.00	0.00	1
3	0.92	1.00	0.96	11
4	1.00	0.75	0.86	4
5	0.95	0.97	0.96	152
6	1.00	1.00	1.00	4
7	0.00	0.00	0.00	2
8	1.00	0.33	0.50	3
9	1.00	1.00	1.00	13026
10	0.89	0.80	0.84	40
11	1.00	1.00	1.00	21914
12	0.00	0.00	0.00	1
14	1.00	0.98	0.99	48
15	0.99	0.99	0.99	92
16	0.00	0.00	0.00	3
17	0.99	0.99	0.99	220
18	1.00	0.99	1.00	160
19	0.00	0.00	0.00	2
20	1.00	1.00	1.00	231
21	0.89	0.95	0.92	220
22	0.62	1.00	0.77	5
accuracy			1.00	36397
macro avg	0.73	0.71	0.71	36397
weighted avg	1.00	1.00	1.00	36397

Unbalanced

	precision	recall	f1-score	support
0	1.00	1.00	1.00	247
1	0.88	0.78	0.82	9
2	1.00	0.50	0.67	2
3	0.92	0.92	0.92	12
4	1.00	1.00	1.00	3
5	0.97	1.00	0.98	163
6	1.00	0.75	0.86	4
7	0.00	0.00	0.00	3
8	1.00	0.67	0.80	3
9	1.00	1.00	1.00	261
10	0.95	0.87	0.91	46
11	0.96	0.95	0.96	253
12	1.00	1.00	1.00	2
13	1.00	1.00	1.00	1
14	1.00	1.00	1.00	52
15	1.00	1.00	1.00	110
16	1.00	0.50	0.67	2
17	0.99	0.99	0.99	222
18	0.99	1.00	0.99	158
20	1.00	1.00	1.00	221
21	0.97	0.98	0.97	204
22	0.75	1.00	0.86	6
accuracy			0.98	1984
macro avg	0.93	0.86	0.88	1984
weighted avg	0.98	0.98	0.98	1984

Balanced + Adam

	precision	recall	f1-score	support
0	1.00	1.00	1.00	247
1	0.70	0.78	0.74	9
2	1.00	0.50	0.67	2
3	0.92	0.92	0.92	12
4	1.00	0.67	0.80	3
5	0.95	1.00	0.98	163
6	1.00	0.75	0.86	4
7	1.00	0.33	0.50	3
8	1.00	0.33	0.50	3
9	0.99	1.00	1.00	261
10	0.97	0.78	0.87	46
11	0.96	0.95	0.96	253
12	1.00	0.50	0.67	2
13	1.00	1.00	1.00	1
14	1.00	1.00	1.00	52
15	1.00	1.00	1.00	110
16	0.00	0.00	0.00	2
17	0.99	1.00	0.99	222
18	0.99	1.00	1.00	158
20	1.00	1.00	1.00	221
21	0.96	0.98	0.97	204
22	0.75	1.00	0.86	6
accuracy			0.98	1984
macro avg	0.92	0.79	0.83	1984
weighted avg	0.98	0.98	0.98	1984

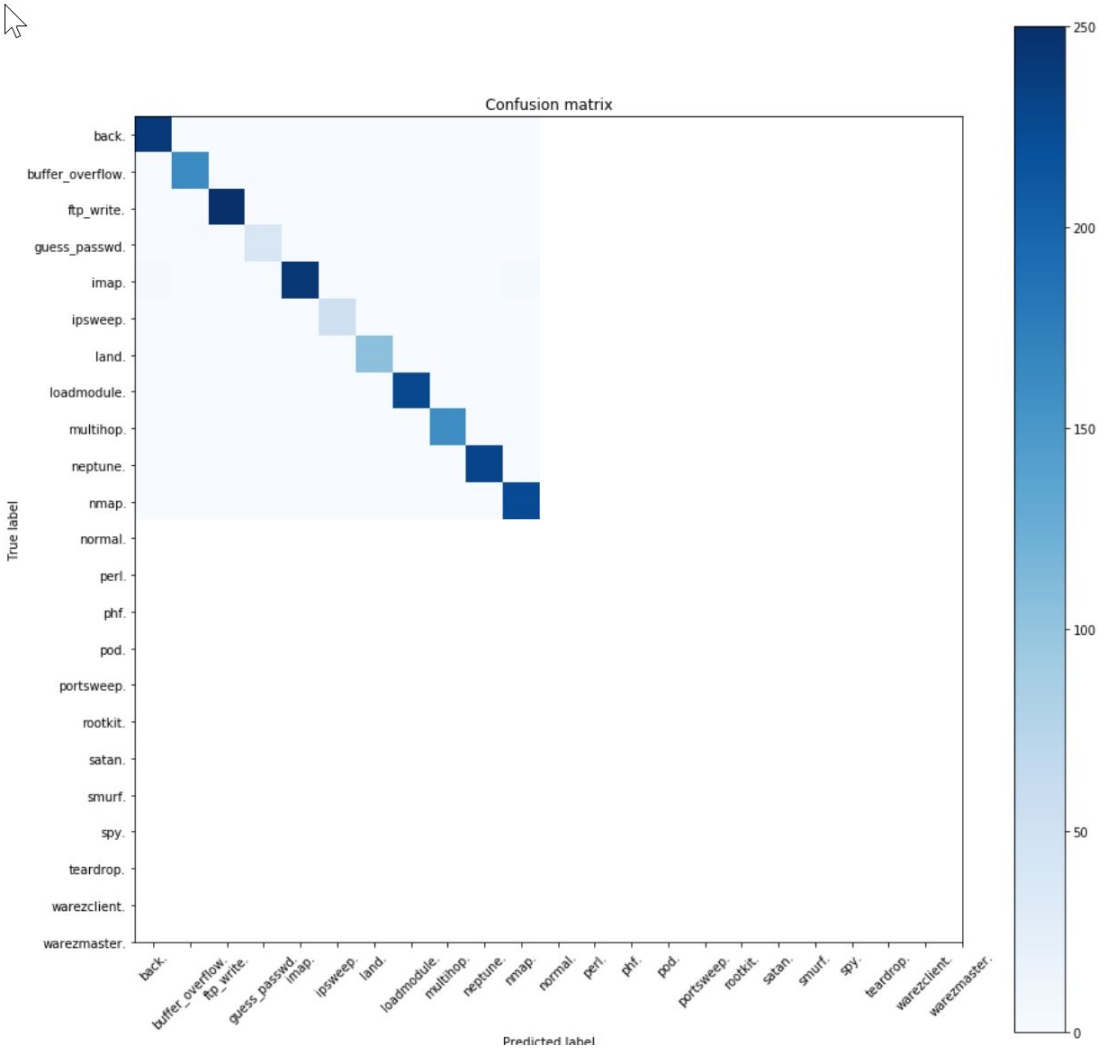
Balanced + SGD

In addition, using relu as our activation proved to be the most effective compared to sigmoid and tanh and thus produced our final results.

```
Accuracy: 0.9917525773195877
Averaged F1: 0.9917659459232241
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       242
           1       0.99      0.99      0.99       163
           2       1.00      1.00      1.00       250
           3       0.93      0.95      0.94        39
           4       0.98      0.98      0.98       250
           5       1.00      1.00      1.00        52
           6       1.00      0.98      0.99       104
           7       0.99      0.98      0.99       227
           8       0.99      1.00      0.99       160
           9       1.00      1.00      1.00       230
          10       1.00      1.00      1.00       223

    accuracy                           0.99      1940
   macro avg       0.99      0.99      0.99      1940
weighted avg       0.99      0.99      0.99      1940
```



Task Division and Project Reflection

Task division for this project were assigned as follows, Manuel handled preprocessing and truncating the data for the model. John handled the fully-connected neural network and Jason was in charge of convolutional neural networks (CNN) and extraction of the most important features.

Challenges encountered throughout the project included our model returning NaN in our values when we initially forgot to remove NaN values but we fixed it after calling an inline NaN removal feature when inserting our data into a dataframe. Applying the L2 regularization when retrieving important features proved to be an issue. We had to read further into the logistic regression functions documentation page to discover the correct format our data needed to be used as a parameter. Furthermore, our convolutional neural network returning 0 f1-scores in different attack types which we solved by simple removing attack types that didn't meet a threshold.

Overall, we learned the correct approach when solving a classification problem using neural networks and CNN.