Applying FNNs in Network Traffic Analysis

Tarun Chinmai Sekar
1213181929

Intrusion Detection and Prevention systems need to be aware of traffic that flow through the network and quickly categorize them as malicious or benign. It is essential to apply machine learning in this process as it is impossible to model all kinds of traffic and label them as malicious or benign. In today's evolving threat landscape, it is more important than ever to detect and block malicious traffic on the fly, and for defenses to adapt to the changing tools of attackers. This project presents a Feed-Forward Neural Networks (FNN) based solution that can detect malicious traffic flowing through the network. FNNs are neural networks which operate without any "memory". As a result their ability to "retain" memory is less. However, they give better performance than statically built signature-based systems that can detect only existing attacks. The aim of this project is to present a Neural Network built with Keras and TensorFlow that is trained with a few varieties of attacks and can be applied to detect a number of new or unknown attacks.

Tools Used:

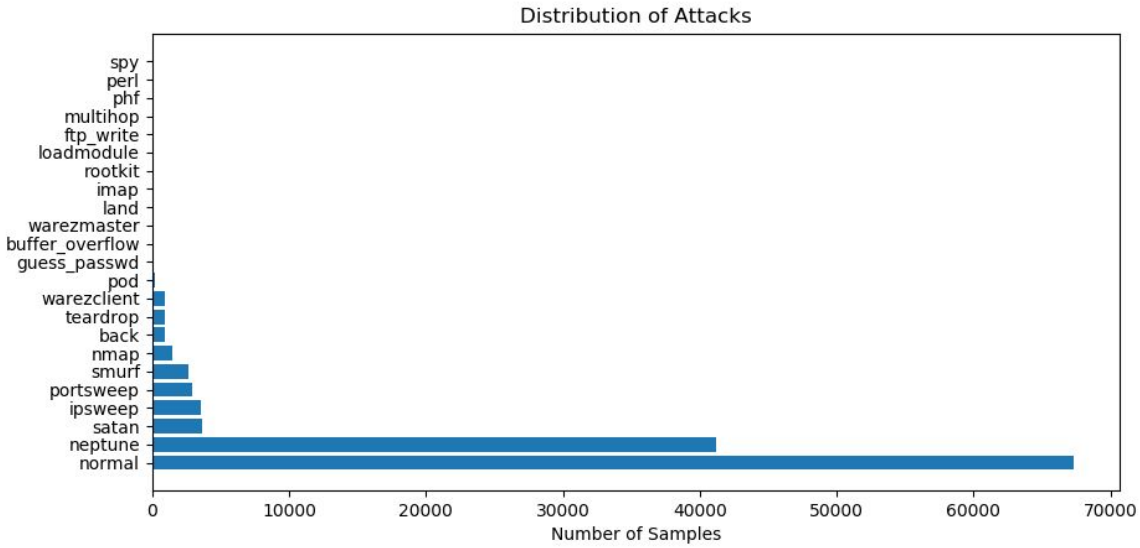
- Python3.6
- Keras
- Tensorflow
- Jupyter notebook & IPython
- Matplotlib

Dataset:

The NSL-KDD Dataset was used to perform the analysis. The dataset is available as two different files. KDDTrain+.txt with 125972 records and KDDTest+.txt 22543 records. Both the files have 43 features. The features of the dataset are as follows

```
'duration', 'protocol_type','service','flag', 'src_bytes', 'dst_bytes','land',
'wrong_fragment', 'urgent', 'hot', 'num_failed_logins',
'logged_in','num_compromised', 'root_shell', 'su_attempted', 'num_root',
'num_file_creations','num_shells', 'num_access_files', 'num_outbound_cmds',
'is_host_login', 'is_guest_login', 'count', 'srv_count','serror_rate',
'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
'diff_srv_rate','srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
'dst_host_same_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate',
'type', 'difficulty'
```

Of these, the first 41 columns are taken as the features, and the 42nd column - type is taken as the label. The difficulty column is discarded as it is irrelevant to the current system. Looking at the distribution of the attack types in the type column, it is clear that some attacks are more prevalent than the others. The following figure shows the distribution.



Experimental Method:

The experiment was carried out by splitting the training dataset into four distinct subsets.

A1 - normal, neptune, satan

A2 - normal, warezmaster, portsweep

A3 - normal, nmap, buffer_overflow

A4 - normal, teardrop, multihop

The test dataset was prepared by removing the attacks from the test dataset. I.e, for A1, the corresponding test data will not have neptune and satan attacks. Moreover, the data is multi-class, and is converted to binary data by renaming all attack types to ‘attack’ in both the training and testing set.

The data is then transformed through LabelEncoding and OneHotEncoding to make training faster. The model is constructed as a three layer neural network, with 12 nodes in the first two layers, and one in the last layer. The first two layers use ‘pRelu’ activation function and the last layer uses the ‘sigmoid’ activation function. ‘Binary_crossentropy’ loss is considered, and ‘adam’ is used for gradient descent. The model is trained for 100 epochs with a batch size of 100, and a validation split of 15%. The accuracy,

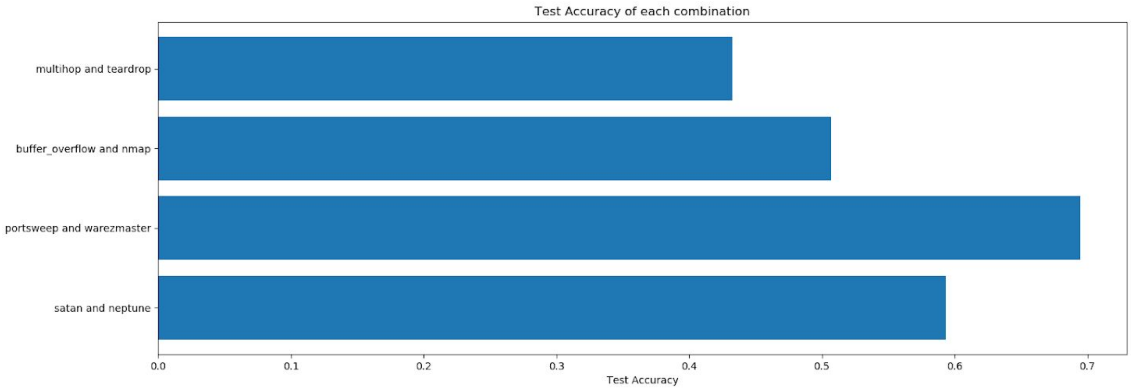
test_accuracy, validation_accuracy, loss, validation_loss, time taken to train are the metrics measured for the model and the results are compared.

Experimental Results:

1. When data representing new type of attack is passed, does your model predict it to be normal or attack ?
   Looking at the confusion matrices for the model when trained for the combination of attacks, it is clear that the performance of the model suffers when the training data is completely removed from the test set. In particular, it can be seen from the table below that the model has a large number of false positives for “normal” regardless of the dataset. So when the new type of attack is passed, the model predicts it as normal for most subsets except that of A2.

Subsets	True Negative	False Positive	False Negative	True Positive
A1	1432	6009	686	9025
A2	6311	5420	52	9659
A3	1357	11382	41	9670
A4	533	12269	42	9669

2. What is the average accuracy that it detects the new type of attack to be as normal or attack ?
   The average accuracy of the model is pretty low, as the model is tested with completely new types of attacks. However, subset A2 still manages to get an accuracy of above 70%.



3. How are the untrained subset of attacks different from the trained ones?

Untrained subset of attacks yield poorer results compared to the trained ones, with the trained ones giving around 80% accuracy for attack type A2 as opposed to 70% for untrained ones. The attacks are not drastically different, but there are some nuanced variations amongst them that cause the variations in the accuracies.

4. Observe your model's prediction capability wrt change in the attack types it was trained on
   The model's prediction capabilities are predominantly dependant on the attack type it was trained on. The model A2 with the warezmaster and portsweep, despite having less number of records for training performs the best because the attack's characteristics are closely related with other different attacks, and share some of the common features. Despite having a large number of records, the combination of neptune and satan do not yield good results, thereby proving that it is not the quantity of records but the correlation between them that yields the most benefit for the prediction.
5. Does the prediction accuracy relate to the attacks being similar? If so, what is the similarity ?
   The prediction accuracies are dependant on the features of the attack, and as seen from the above experiments, the A2 set with warezmaster and portsweep attacks which are predominantly tcp based attacks. Moreover, they also cover a wide range of services impacted and hence the model is able to accurately predict the class for other attack types.

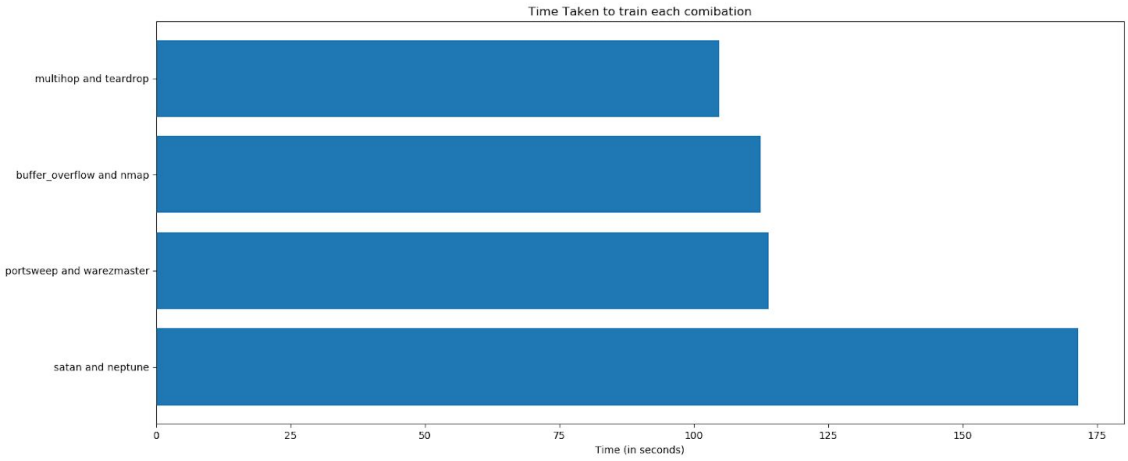
In addition to the above conclusions, an additional experiment was performed to find the pair of attacks that yield the maximum accuracy. The top 10 pairs are as follows

Attack1	Attack2	Accuracy
warezmaster	portsweep	0.743587352
portsweep	guess_passwd	0.731694635
back	portsweep	0.720116221
portsweep	phf	0.711445675
portsweep	land	0.709146968
portsweep	imap	0.707706053
portsweep	ftp_write	0.706920431

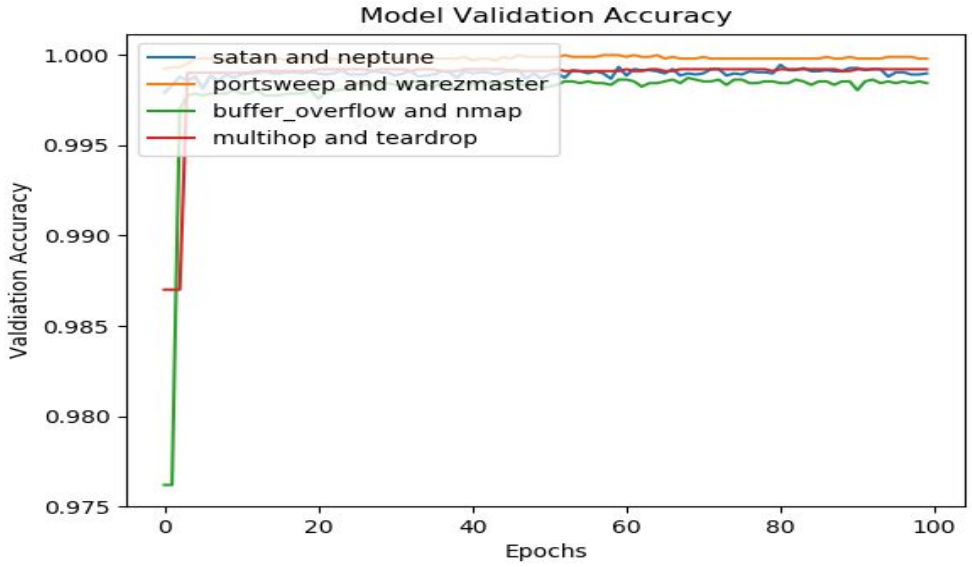
portsweep	multihop	0.706723891
perl	portsweep	0.70586133
teardrop	portsweep	0.705595781

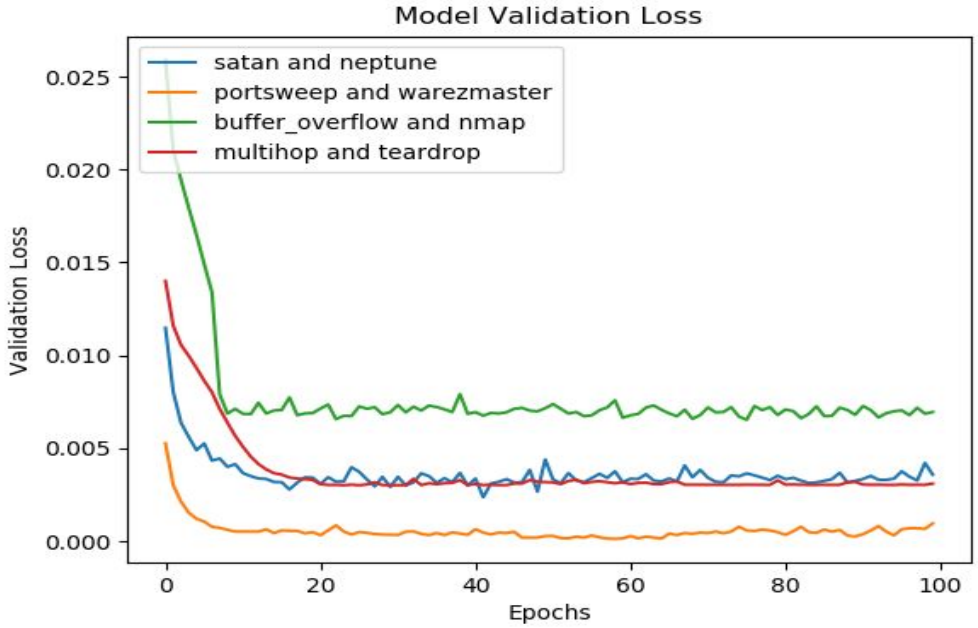
It is abundantly clear from the above table that the portsweep attack type is a tcp based attack that captures most of the features in the other attacks. For instance, the full dataset has the following distinct values for the feature - ‘flag’
Full-dataset = ['OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH']
Portsweep = ['OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'SF']
This shows that the portsweep dataset is missing just 4 values. As a result, the model is able to recognize attacks that have this values, and can identify it as attack. The same is observed for the features ‘service’ as well. The entire dataset has 70 unique values, and portsweep has 56. By just considering the categorical values, one can arrive at the conclusion that since portsweep data is the most general of the datasets, models trained with it have better performance.

Additionally, analyzing the time taken to train each model, it is seen that the model with the largest number of records took the longest to train.



The validation loss and validation accuracies of the model are plotted in the below two graphs.


Model Validation Accuracy
satan and neptune
portsweep and warezmaster
buffer_overflow and nmap
multihop and teardrop
Valdiation Accuracy
1.000
0.995
0.990
0.985
0.980
0.975
0
20
40
60
80
100
Epochs


Model Validation Loss
satan and neptune
portsweep and warezmaster
buffer_overflow and nmap
multihop and teardrop
Validation Loss
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
Epochs