Executive Summary:

In this report, an alternate method for intelligently detecting intrusions using neural networks was used. We will be implementing the Intrusion Detection System using R. The artificial neural networks are implemented using the 'neuralnet' package. The results obtained had high accuracy and very low false positive and negative ratios. The neural network implementation was fairly quick in terms of competition time and had moderate memory consumption.

Specifications:

Our aim in this project was to determine the performance of Neural Networks in several different settings as follows:

1. Understanding their ability to differentiate a particular type of attack from other attacks and normal conditions.

2. Understanding how efficient Neural Networks are in distinguishing attacks from normal usage (anomaly detection).

3. Understanding how well Neural Networks perform at differentiating a given sets of attacks from other situations.

Preprocessing:

To boost the running efficiency, we decided to reduce the number of attributes that serve as inputs to the NNs. There was a lot of scope for attribute reduction in the data sets as a lot of attributes had either negligible variation or no variation at all. Reducing the attributes has a lot of advantages and the more attributes one can reduce, the more efficient and simple the model becomes. However, attribute reduction also affects the accuracy of the model. So we had to find a balance between speed and accuracy. We decided on the final number of attributes by the process of trial and error.

The basis for checking the quality of an attribute was the variance in the attribute samples. The variance for each attribute was calculated and summed to obtain the total variation in the data. The total variation was used to calculate the contribution of every attribute to the total variation in the data.The criterion we used for reduction was that the attribute should contribute to at least 1 percent of the total variation in the data set.

Using this measure, the number of attributes were reduced to 16 , from the original 41 which was a reduction of around 60%.Thus, the runtime of implementation was significantly improved.

Except for the final case of having to differentiate between multiple attacks, the approach mentioned above worked well for almost all the configurations (giving us around 99% accuracy). For the final case however, the accuracy was not satisfactory (around 94%). So for that particular case, we decide to consider all attributes that had a variation greater than zero, which improved the accuracy significantly (98%). Another interesting observation was that changing the number of attributes did not significantly affect training and testing speed but did affect the memory consumption of the NN objects.

Methods and Techniques:

The columns were filtered based on their variation and a dataset was made. The data was then split into training set (70%) and test set (30%).Neural network was trained using the training set. The structure of our NN consisted of a single hidden layer of 8 to 10 neurons and input and output layers. Each attribute had a corresponding input. The output was defined by a single neuron. This output was the weighted sum of the output of the hidden layers. Since the NNs dealt only with numbers, the names of the attacks were replaced with numbers.

After training the neural network, the test set was used to predict responses. After getting the responses, some post-processing had to be done. Since the predictions were numerical, some numbers were outside the range of consideration. This was dealt by replacing the numbers outside the range with the corresponding extremities of the range.

E.g. All predictions below zero were replaced by zero.

The numbers were converted back to their names. The confusion matrix using the correct output from the test set was made to predict the output. The accuracy, false positive and negative ratios, running time and memory consumption of the neural networks was calculated.

Note: The running time includes both the training time and the prediction time.

Implementation Details:

Separate R scripts were written to sample data from the data sets. This data set was stored in separate .CSV files. These .CSV files were then read into the main R scripts. The results were displayed on the command line in a text form. Plots of the neural networks were also generated for better visualization.

The 'neuralnet' package, available in R, was used for our neural network computations. The package provides functions to both create the neural network and to predict using the created neural network object. The provided function to create neural networks is highly configurable. One can provide the entire structure of the network in the form of a list. The list size indicates the number of hidden layers and each element in the list represents the number of neurons in the corresponding layer.

One can also decide the threshold error value below which the training stops. These stored values were then used to create formula that would be fed to the 'neuralnet' function from the neuralnet package (Details about the function and package we used for creating neural networks are given in the next section). The neuralnet function returned an object that encapsulated all the properties of the neural network like the weights and the network structure. The neural network object was used to predict the output for the test set.

Results:

Result with 16 attributes:

Attributes Considered:

[1] "duration" "service"

[3] "dst_bytes" "hot"

[5] "logged_in" "num_root"

[7] "count" "srv_count"

[9] "rerror_rate" "srv_rerror_rate"

[11] "same_srv_rate" "dst_host_count"

[13] "dst_host_srv_count" "dst_host_same_srv_rate"

[15] "dst_host_same_src_port_rate" "dst_host_srv_rerror_rate"

Total Variation Captured: 91.8976 %

No of Attributes considered: 16

Results:

Misuse Detection:

Attack Type: Neptune

	obtainedOutput	
expectedOutput	Neptune	Other
Neptune	49	13
Other	3	835

Neural Network Memory Consumption: 1619528 bytes

Test Accuracy: 0.982

False Positives ratio: 0.3%

False Negatives ratio: 1.4%

Execution time for Neptune attack detection model: 0.742 seconds

Attack Type: Satan

	obtainedOutput	
expectedOutput	Other	Satan
Other	837	3
Satan	5	55

Neural Network Memory Consumption: 1619528 bytes

Test Accuracy: 0.991

False Positives ratio: 0.5%

False Negatives ratio: 0.3%

Execution time for Satan attack detection model: 2.775 seconds

Attack Type: Smurf

	obtainedOutput	
expectedOutput	Other	Smurf
Other	833	1
Smurf	11	55

Neural Network Memory Consumption: 1619528 bytes

Test Accuracy: 0.987

False Positives ratio: 1.2%

False Negatives ratio: 0.1%

Execution time for Smurf attack detection model: 0.574 seconds

Attack Type: PortSweep

	obtainedOutput	
expectedOutput	Other	PortSweep
Other	836	0
PortSweep	4	60

Neural Network Memory Consumption: 1619528 bytes

Test Accuracy: 0.996

False Positives ratio: 0.4%

False Negatives ratio: 0%

Execution time for PortSweep attack detection model: 3.668 seconds

Attack Type: NMap

	obtainedOutput	
expectedOutput	NMap	Other
NMap	48	10
Other	7	835

Neural Network Memory Consumption: 1619528 bytes

Test Accuracy: 0.981

False Positives ratio: 0.7%

False Negatives ratio: 1.1%

Execution time for NMap attack detection model: 0.721 seconds

Anomaly Detection:

	obtainedOutput	
expectedOutput	Attack	Normal
Attack	294	4
Normal	4	598

1594328 bytes

Neural Network Memory Consumption: 1594328 bytes

Test Accuracy: 0.991

False Positives ratio: 0.4%

False Negatives ratio: 0.4%

Execution time for Anomaly Detection: 5.941 seconds

Five and Other:

	obtainedOutput					
expectedOutput	Neptune	NMap	Other	PortSweep	Satan	Smurf
Neptune	52	0	0	0	4	0
NMap	0	41	21	2	0	0

Other	0	18	693	2	0	1
PortSweep	0	5	1	45	0	0
Satan	8	1	3	0	49	8
Smurf	0	2	1	5	1	57

Neural Network Memory Consumption: 1829288 bytes

Test Accuracy: 0.919

False Positives ratio: 4.4%

False Negatives ratio: 3.75%

Execution time for Detecting Five Specified Attacks: 24.782 seconds

Result with 35 attributes:

Attributes Considered:

[1] "duration" "protocol_type"

[3] "service" "flag"

[5] "src_bytes" "dst_bytes"

[7] "hot" "num_failed_logins"

[9] "logged_in" "num_compromised"

[11] "root_shell" "num_root"

[13] "num_file_creations" "num_shells"

[15] "num_access_files" "is_guest_login"

[17] "count" "srv_count"

[19] "serror_rate" "srv_error_rate"

[21] "rerror_rate" "srv_rerror_rate"

[23] "same_srv_rate" "diff_srv_rate"

[25] "srv_diff_host_rate" "dst_host_count"

[27] "dst_host_srv_count" "dst_host_same_srv_rate"

[29] "dst_host_diff_srv_rate" "dst_host_same_src_port_rate"

[31] "dst_host_srv_diff_host_rate" "dst_host_serror_rate"

[33] "dst_host_srv_serror_rate" "dst_host_rerror_rate"

[35] "dst_host_srv_rerror_rate"

Total Variation Captured: 100 %

No of Attributes considered: 35

Results:

Misuse Detection:

Attack Type: Neptune

expectedOutput	obtainedOutput Neptune	Other
Neptune	60	0
Other	0	840

Neural Network Memory Consumption: 2275728 bytes

Test Accuracy: 1

Execution time for Neptune attack detection model: 0.697 seconds

Attack Type: Satan

obtainedOutput

expectedOutput Other Satan

Other 839 3

Satan 0 58

Neural Network Memory Consumption: 2275728 bytes

Test Accuracy: 0.997

Execution time for Satan attack detection model: 5.244 seconds

Attack Type: Smurf

obtainedOutput

expectedOutput Other Smurf

Other 835 1

Smurf 8 56

Neural Network Memory Consumption: 2275728 bytes

Test Accuracy: 0.99

Execution time for Smurf attack detection model: 0.576 seconds

Attack Type: PortSweep

obtainedOutput

expectedOutput Other PortSweep

Other 842 0

PortSweep 1 57

Neural Network Memory Consumption: 2275728 bytes

Test Accuracy: 0.999

Execution time for PortSweep attack detection model: 1.346 seconds

Attack Type: NMap

obtainedOutput

expectedOutput NMap Other

NMap 58 16

Other 2 824

Neural Network Memory Consumption: 2275728 bytes

Test Accuracy: 0.98

Execution time for NMap attack detection model: 0.66 seconds

Anomaly Detection:

obtainedOutput

expectedOutput Attack Normal

Attack 281 2

Normal 2 615

2250528 bytes

Neural Network Memory Consumption: 2250528 bytes

Test Accuracy: 0.996

Execution time for Anomaly Detection: 2.198 seconds

Five and Other:

	obtainedOutput					
expectedOutput	Neptune	NMap	Other	PortSweep	Satan	Smurf
Neptune	66	0	0	0	1	0
NMap	0	55	1	0	0	0
Other	0	4	706	1	0	0
PortSweep	0	0	1	51	0	1
Satan	1	1	2	0	54	2
Smurf	0	0	0	7	0	66

Neural Network Memory Consumption: 2578832 bytes

Test Accuracy: 0.978

Execution time for Detecting Five Specified Attacks: 10.472 seconds

Description of the results:

The results above are for 2 separate runs over the datasets one with 35 attrbutes and another with 16 attributes. Out of the initial 41 attributes, we came across 6 attributes which had no variation at all ie all their values were same for all the instances, and as a result those attributes were removed from the dataset resulting in the 35 atribute dataset. Later, after running a calculation check over this dataset we tried to find out the columns which have more than 1% variance as compared to other attributes and this is when we narrowed it down to a 16 attributes dataset. Apart from the number of attributes in each of these datasets, they are similar in all other respects and so is the structure of the results of the scripts over them.

Now, lets go through the outputs shown above to understand them better. The very first Misuse Detection talks about the Attack type Neptune. The output is expressed in the form of a confusion matrix, memory consumption by the Neural Network, Test accuracy and Execution time. Confusion

Matrix is a square matrix which explains in this case, how many Neptune attacks were classified as Neptune(49) and how many were classified as Others(13) by mistake. Similarly, the next row shows how many Other attacks were classified as Neptune(3) by mistake and how many were correctly classified as Others(835). Based on the confusion matrix we calculate the other terms such as Test accuracy, False Positives and False Negatives.

Test Accuracy - Sum of the figures in the matrix diagonal divided by the sum of figures in the entire matrix. This decimal figure can actually be seen as the percent ie 0.982 ~ 98.2%

False Positives - Other attacks incorrectly classified as Neptune.(in this case 13)

False Negative - Neptune attacks incorrectly classified as Others.(in this case 3)

Work Load Distributions:

The group 8 consists of 3 project-mates. The most of the work was divided in one-third for convenience purposes. The work was modularized, such that it could be split independently amongst allowing us to work on things in parallel. All the datasets used in our report have been generated by R scripts. The basic R code of reading in instances of dataset files was with us. But making specific dataset files like the ones mentioned below:

i. One type of attack and not that attack (normal & others)
ii. Anomaly detection with Normal and not-Normal
iii. Misuse detection with 5 types of attacks and not one of those 5 attacks (normal and others)

The generation of these 3 different types of dataset files was split amongst the 3 of us. Once the dataset files were generated separately from each of our R programs, we coupled them together into one R script. Also, the same was done with building the NN model and testing the network over these datasets. The NNs were built separately since their building and execution of NNs is an independent process. Later, once the results were achieved, the codes were also combined in a single R script. The main reason behind combining all our separate codes into a single R script file was to carry out all the actions right from the extraction of the data from the dataset files to modelling, building and executing the NN such that we could start this script execution and it would take care of all the action from scratch.
The reports for the Project Part 1 & 2 have also been written collaboratively by all of us. The various sections of the report such as the Methods & Techniques, Implementation, Specifications, Executive Summary and Results amongst each other so we could finish the report in parallel and get a unique feedback from each one of us in the reports. Once all the sections were completed individually, the were combined into one report document.