**CHAPTER 4**

**Implementation and results**

# **Overview**

This chapter list out all implementation process, and results of the proposed prevention system for DDoS attack in cloud server through HIPDS in hypervisor. As shown in figure 4.1, how will the implementation process staring from implement the cloud server in windows environment and put the normal and anomaly IPs that came from combination of two datasets inside folder that belong to cloud database storage.

Thus, the Host based integrated IDPS will detect and prevent three types of DDoS attack which are TCP flooding, TCP Sync and UDP attack. The hypervisor then will profile the results either in white list, or black list that reported in excel sheet. Therefore, in chapter will divided in two major sections one for the implementation phase and second one is for the results which will achieve the thesis objective of the implementation and results.

The Cooperative Association for Internet Data Analysis (CAIDA)is a collaborative undertaking among organizations in the commercial, government and research sectors aimed at promoting greater cooperation in the engineering and maintenance of a robust, scalable global Internet infrastructure. CAIDA datasets are primarily used by researchers for scientific analysis of Internet traffic, topology, routing, performance and security related events.

Caida dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007 (20:50:08 UTC to 21:56:16 UTC) which last modified 2013-05-08. This type of denial-of-service attack attempts to block access to the targeted server by consuming computing resources on the server and by consuming all of the bandwidth of the network connecting the server to the Internet.

The CAIDA "DDoS Attack 2007" Dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007 and updated in 2013 .Flooding type DDoS attack was launched to block access to the targeted server by consuming computing resources on the server and by consuming all of t the one-hour trace comprises only of traffic to the victim machine and the responses to the attack from the victim machine. Traffic towards other hosts and payload has been removed. The IP addresses of the hosts were prefix anonymized the network address was kept the same while the host addresses were changed. he bandwidth of the network connecting the server to the Internet.

traffic traces store only attack traffic to the victim and response from the victim; non-attack traffic has been removed as much as possible. Per Moore et al, it is a high-rate attack if there are more than 10,000 packets per second over the network, with 1000 attack packets per second covering 60% of the attack traffic(Moore et al. 2006). Thus, this is low-rate attack traffic We consider real-time low-rate and high-rate DDoS attack scenarios for both datasets during our experiments. However, low-rate attack does not consume all the computing resources on the server or all bandwidth of the network connecting the server to the Internet. So, a low-rate DDoS attack scenario not only contains attack traffic but also contains attack free traffic. During our experiment, we mix low-rate attack traffic and legitimate traffic to prepare the low-rate DDoS attack scenarios in the UCLA DDoS dataset.

For the UCLA dataset



Figure 4.1: The overall implementation and results phases in chapter 4

**4.2 Implementation phase**

Converting your MATLAB code to cloud server, and deploying smart algorithms in same time made it possible that this code is used anywhere on the world. The first phase in implementation phase will have two phases -which are how to implement the cloud server -which have the datasets, and second phase will show how will implement the Host based intergraded IDPS .

**4.2.1 Framework of cloud server implementation**

Cloud computing has implementing by deployed at Tomcat 9 Software as a cloud webserver in a Windows environment – which might see simple enough, given that all cloud storage future in feasible hardware implements(Tomcat 2016). Cloud server that have designed which have all the IPs traffic -which have generated from the dataset. While, in table 4.1 shows the specification of hardware that the cloud and the rest of the proposed model in chapter 3.

**Table 4.1: The hardware specification for the implementation phase**

|  |  |
| --- | --- |
| Hardware | Type |
| |  | | --- | | Operating System | | Windows 10 Education (x64) Version 1607 |
|  |  |
| |  | | --- | | Processor | | 3.10 gigahertz Intel Xeon E5-1607 v3 |
|  |  |
| Drive | 500.00 Gigabytes Usable Hard Drive Capacity |
| System Model | Dell Inc. Precision Tower 5810 |
| |  | | --- | | Memory Modules | | 8116 Megabytes Usable Installed Memory |

* + 1. **Dataset**

Two DDoS datasets are used for evaluating the proposed attack detection technique. One of these datasets represents a network layer flooding attack while the other two represent different types of application level attacks. Each of these attack datasets are simulated for a period of 5 minutes: first by two minutes of normal activity, followed by three minutes of flooding. Each of these datasets is now described.

**Table 4.1 level Statistics of the real time CAIDA and UCLA DDoS dataset.**

|  |  |  |
| --- | --- | --- |
| Characteristic Features | CAIDA Dataset | UCLA Dataset |
| Activity type | DDoS Attack | DDoS Attack |
| Duration | approximately one hour of anonymized traffic |  |
| Packets type | ICMP ECHO |  |
| Number of target(s) | 1 |  |
| Number of packets sen | 359,655,826 | 10213 |
| File format | pcap | Text tiles |
| Size (uncompressed) | 42 data files (11.1 GiB) | 11.1 GB |

* + 1. **Dataset Processing**

A subset of CAIDA and UCLA as used for the experiments described in this section. The dataset is a six-hour network trace around the1st semi-final match of the CAIDA. The original dataset was first filtered to remove all traffic except HTTP GET requests. Secondly, only packets containing 200, 206 and 400 HTTP status codes were extracted and used as the input trace (processed dataset) for the experiments. Table 4.2 shows the high-level characteristics of the original dataset and the processed dataset used in the simulations.

**Table 4.2: Characteristic features of the dataset used for simulations**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Original Dataset** | **Preprocessed Dataset** |
| Number of requests | 29,662,465 | 19,615,109 |
| Number of unique source IPs | 79,033 | 78,646 |
| Number of unique source IPs | 11,885 | 11,839 |
| Number of unique source IPs | 4,089 | 2,376 |
| Number of unique source IPs | GET, POST, HEAD | GET |
| HTTP status co des | all | 200, 206 and 400 |

**4.4.4 Features of datasets for All attacks**

In implementation phase have used CAIDA and UCLA datasets (for traffic), and the dataset has customized and imported the to MATLAB and save the entire dataset as a mat file. The mat file example: udp\_Feat.mat is used by udp\_Feat\_Tbl (feature table generation script). This file(udp\_Feat\_Tbl) calculates all features and populates them in a structure (myStr) and also in an array (X). XIP is a structure containing Source IP’s and Destination IP’s, however, in table 4.3 shown the features for All attacks.

Table 4.2 features for All attacks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CAIDA | | | UCLA |  |
| **No** | **UDP flooding** | | **TCP sync** | **TCP push** | **TCP flooding** |
| 1 | No of packets | No of packets | No of packets | No of packets | No of packets |
| 2 | Average packet size | Average packet size | Average packet size | Average packet size | Average packet size |
| 3 | No of bytes | No of bytes | No of bytes | No of bytes | No of bytes |
| 4 | Packet rate | Packet rate | Packet rate | Packet rate | Packet rate |
| 5 | Bit rate | Bit rate | Bit rate | Bit rate | Bit rate |
| 6 | Ratio udp | Ratio udp | Ratio udp | Ratio udp | Ratio udp |
| 7 | Ratio tcp | Ratio tcp | Ratio tcp | Ratio tcp | Ratio tcp |
| 8 |  |
| 10 | Source count | Source count | Sync Count | Sync Count | Sync Count |
| 11 | Destination count | Destination count | Push count | Push count | Push count |
| 12 | - | - | Fin count | Fin count | Fin count |
| 13 | - | - | Ack count | Ack count | Ack count |

While , X contains the following features as listed in the code below which taken from udp\_Feat\_Tbl.m file as an example :   
    X(ji,1)=myStr(ji).Nopckts;%#ok  
    X(ji,2)=myStr(ji).AvgPckSz;%#ok  
    X(ji,3)=myStr(ji).Nobytes;%#ok  
    X(ji,4)=myStr(ji).Pckrate;%#ok  
    X(ji,5)=myStr(ji).Bitrate;%#ok  
    X(ji,6)=myStr(ji).RatioUDP;%#ok  
    X(ji,7)=myStr(ji).RatioTCP;%#ok  
       X(ji,9)=myStr(ji).SrcCnt;%#ok  
    X(ji,10)=myStr(ji).DstCnt;%#ok

These (X myStr XIP No\_Pckts) are saved in udp\_Feat.mat file to be used by PCA, LDA and proposed algorithm for data reduction respectively. so, the so PCA and LDA will work in dot mat files will analyzing features column wise will become easier. Next step will create a folder called TestDoc in webapps folder of cloud apache tomcat server -which located at: C:\cloud apachetomcat9\webapps\TestDoc. And it have three databases, as show in table 4.3 -which execute 3attacks, and run 4files from MATLAB to udp203set.m, TCP Push-Ack attack and tcp203set.m”.

**Table 4.3: the dot mat converted data**

|  |  |
| --- | --- |
| Mat files types | Testing type |
| caida1tcpsync.mat | test tcp sync database |
| tcp2tracePsh.mat | test udp database |
| trace1udp.mat | test udp database |
|  |  |

## Implementation of UDP Flood

Form UCLA data set we have download trace1 from their website and convert it all dot mat file contains 105237 records with 10000 records of UDP attacks and the rest 5237 records are normal. Both are combined to form trace1 intro. mat file as shown as a sample of record in figure 4.2. Where in original travel file have UDP attack records from 1 – 99999 , and from 100000 – 105237 contain Normal records

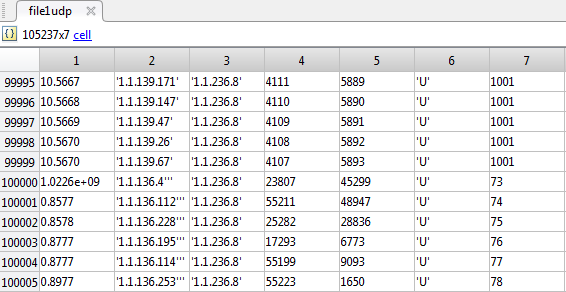
****

Figure 4.2: Sample of records

There are 7 fields in this DB in UDP UCLA dataset as shown in table 4.4

|  |  |
| --- | --- |
| Mat files types | Testing type |
| packet Time | is time when packet was sent |
| Source IP | is number masking the IP address of packet source |
| Destination IP | is number masking the IP address of packet destination |
|  |  |
| Source Port Number | is the original source port |
| Destination Port Number | is the original destination port |
| Packet Length in bytes | is length of packet (without header) in Bytes |

**4.5.1 PCA ,LDA and ANTLION over UDP flood attack**

In some situations, the dimension of the input vector is large, but the components of the vectors are highly correlated (redundant). It is useful in this situation to reduce the dimension of the input vectors. An effective procedure for performing this operation are PCA and LDA. This technique has three effects: it orthogonalizes the components of the input vectors (so that they are uncorrelated with each other); it orders the resulting orthogonal components at PCA and LDA, therefore, those with the largest variation come first; and it eliminates those components that contribute the least to the variation in the data set.

The performance of a trained network can be measured to some extent by the errors on the training, validation, and test sets, but it is often useful to investigate the network response in more detail. One option is to perform regression analysis between the network response and the corresponding targets.

Neural networks have been applied very successfully in the identification and control of dynamic systems. The universal approximation capabilities of the multilayer perceptron make it a popular choice for modeling nonlinear systems and for implementing general-purpose nonlinear controllers. This chapter introduces three popular neural network architectures for prediction and control that have been implemented in the Neural Network Toolbox

To implement the UDP flooding attack, it has several steps where will stored at **udp\_Feat.mat** as shown as:

1. Starting from load Database with both normal and UDP attack records from Cloud
2. Extract the IP’s, Ports and Packet size
3. Find uniqueness of IP addresses, Ports, and packets
4. Feature extraction and find the correct number of Destination IPs
5. Determine the source and destination IP’s
6. Store Feature values in a structure
7. Copy calculated features into a feature array
8. sort the number of packets in descending order.

After preparation the all procedures for the UDP attack based on selected datasets it come the next part for reduce data size by PCA and LDA. PCA will start working in flowing steps in MATALB code udp\_Feat.mat :

1. Load the “udp\_Feat.mat”
2. Calculate the PCA
3. Remove redundant information - eigen values are less or equal to zero
4. Create Attack matrix
5. Plotting Inputs vs reduced PCA output into original data and transformed data
6. ALO Optimizer
7. Classification using FeedForward Networks
8. Create a Pattern Recognition Network
   1. Set up Division of Data for Training, Validation, Testing
   2. Train the Network
   3. Test the Network
   4. Plotting
      1. plot confusion
      2. Plot Errors
9. Metrics calculation
   1. Plot the attack and normal traffic with reduced dataset
10. Filter the system and white list the attack IP sources
11. Create a Pattern Recognition Network
12. Save the Black Listed SourceIP Addresses to the cloud

## Implementation of TCP attack

TCP sync attack, a continuous umpteen number of Sync without any acknowledgement from another computer (communicating computer) results in only sync packets being flooded. This will cause the stack overflow (buffer overflow). This results in non-availability of services to all.

**TCP Flood Attack**

|  |  |
| --- | --- |
| Mat files types | Testing type |
| packet Time | is time when packet was sent |
| Source IP | is number masking the IP address of packet source |
| Destination IP | is number masking the IP address of packet destination |
|  |  |
| Source Port Number | is the original source port |
| Destination Port Number | is the original destination port |
| Packet Length in bytes | is length of packet (without header) in Bytes |
| FLAG | is TCP flag (as defined in tcpdump) |
| SEQ\_from and SEQ\_to. | are sequence numbers of first and last byte of packet data |
| ACK | is the sequence number acknowledged by the packet |
| WIN | is the window size |

* 1. **TCP Sync attack**

In Ciada dataset TCP sync attack, have a continuous umpteen number of Sync without any acknowledgement from another computer (communicating computer) results in only synchronize packets being flooded. This will cause the stack overflow (buffer overflow). This results in non-availability of services to all. Wireshark have used to convert the pcap files and export it to text file. Then, it has imported in Matlab to applying all the hypothesis mechanism

The procedures have applied for the proposed method in above section 4.5.1 .

|  |  |
| --- | --- |
| Mat files types | Testing type |
| packet Time | is time when packet was sent |
| Source IP | is number masking the IP address of packet source |
| Destination IP | is number masking the IP address of packet destination |
|  |  |
| Source Port Number | is the original source port |
| Destination Port Number | is the original destination port |
| Packet Length in bytes | is length of packet (without header) in Bytes |
| FLAG | is TCP flag (as defined in tcpdump) |
| SEQ\_from and SEQ\_to. | are sequence numbers of first and last byte of packet data |
| ACK | is the sequence number acknowledged by the packet |
| WIN | is the window size |

**4.7 IDPS Metrics and parameters**

Five features are calculated and stored as part of database. The same is reduced with PCA. Which are: No of packets, Average Packet Size, No of bytes, Packet rate and Bit rate .

focuses on solution development with limited effort on systematically characterizing the reliability (precision) of the solution under different conditions. It is especially important to identify conditions where the algorithm's performance can be optimized and conditions where the algorithm may fail. Performance evaluation can achieve this objective.

Therefore, performance evaluation of a machine learning system is a fundamental issue in machine learning and receives increasing attention from machine learning researchers. In this thesis, we will mainly use the well-established ROC analysis to assess the performance of different methods compared in our work. For over a decade, ROC analysis is gaining popularity more intensely also in the field of machine learning.

Since predicted classes of given instances are not necessarily same as true classes, a matrix is used to keep a record of the number of prediction errors. This matrix is called ***a contingency table*** or ***a confusion matrix*** (since it represents the confusion between classes). There are four possible outputs for a classification of each instance, each individual test that is performed can yield one of four possible results, as **True positive, True negative. False positive**. **False negative.**

**4.8 IDPS DDoS attack preventing model Result phase**

For the solid results carried out in this thesis, validation and test data sets are each set to 15% of the original data. With these settings, the input network feature vectors and output target vectors will be randomly divided into three sets as follows:

* 70% are used for training.
* 15% are used to validate that the network is generalizing and to stop training before overfitting.
* The last 15% are used as a completely independent test of network generalization.
  + 1. **UDP flooding Result phase of proposed model (PCA+LDA+ALO)**

Taken to machine learning domain, PCA performs unsupervised transformation, while LDA is supervised. However, PCA is a more generic dimensionality reduction technique. LDA is a more specialized generative method, because we assume a certain statistical model (latent Dirichlet allocation) that generates the text. PCA is mainly used for feature extraction. It finds the features with the highest variation in your data. It is also a nice way to reduce dimensionality if there is a redundancy in your data. Mathematically, you try to find the best projection where the variation is maximised, on the other hand, is mainly used for classification. The space where between class variance is maximized but within class variances are minimized. The input features are extracted from in total 504 records of UDP packages after PCA operation. Therefore, the input matrix is a 5x504 matrix defining 5 attributes of 504 UDP data. The target input is a 504 matrix where each column indicates a correct category with a one in either element 1 (normal) or element 2 (attack).

For PCA and LDA only method, the input feature set is used directly to train a standard feedforward neural network that can be used to detect abnormal network traffic patterns in IDPS. The standard neural network that is used is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The number of hidden neurons is set to 10. The number of output neurons is set to 2, which is equal to the number of categories in the target vector (i.e. the normal network traffic and the attack network traffic).

One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.To address the overfitting problem, the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases.

The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over fit the data, the error on the validation set typically begins to rise. The network weights and biases are saved at the minimum of the validation set error.

The third subset is the test set. The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set. It can be seen clearly, that the feedforward network consists of a series of three layers in the setup. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

The diagram also reveals some other important characteristics of the neural network in question. For example, the total input data set is divided randomly divided into the three subsets using the division parameters specified in the above, i.e. with the ratios for training, testing and validation are 0.7, 0.15 and 0.15, respectively. The Scaled Conjugate Gradient schema is used to train for the optimal values of network weights and bias. The training performance is evaluated during the training process using Cross Entropy metric calculated based on network responses.

The summary diagram also indicates that it took in total 11 iterations for the neural network to converge after the validation errors have increased for 6 consecutive iterations (Validation Checks).

The classification results using trained neural network based on PCA method is shown in Figure 2 in the form of confusion matrix. A Confusion Matrix is a visual performance assessment of a classification algorithm in the form of a table layout or matrix. Each column of the matrix represents predicted classifications and each row represents actual defined classifications. The diagonal cells show for how many (and what percentage) of the examples the trained network correctly estimates the classes of observations. That is, it shows what percentage of the true and predicted classes match.

The off diagonal cells show where the classifier has made mistakes. The column on the far right of the plot shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the bottom right of the plot shows the overall accuracy.The confusion matrix represents the metric scores in a compact way so that it also makes easy to calculate other representative evaluation measures such as Recall, Precision, Accuracy and F-Measure etc.

Figure 2 shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 352 network records (70%) used for neural network training purpose, 76 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Training Confusion Matrix as example, 175 network traffic packages are correctly classified as normal.

This corresponds to 49.7% of all 352 training samples. Similarly, 173 network traffic packages are correctly classified as attack. This corresponds to 50% of the total training samples. Overall, 98.9% of the predictions on the training samples are correct and 1.1% are wrong classifications. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies. Thus, the trained model made in total 497 correction predictions, including 252 normal traffic and 245 attack attempts. There are in total 7 instances where the model misclassifies the real attacks to be normal traffic (false negative). This means the overall classification result has achieve 98.6% of accuracy, with the overall error rate to be 1.4%.

****

Figure 4.3

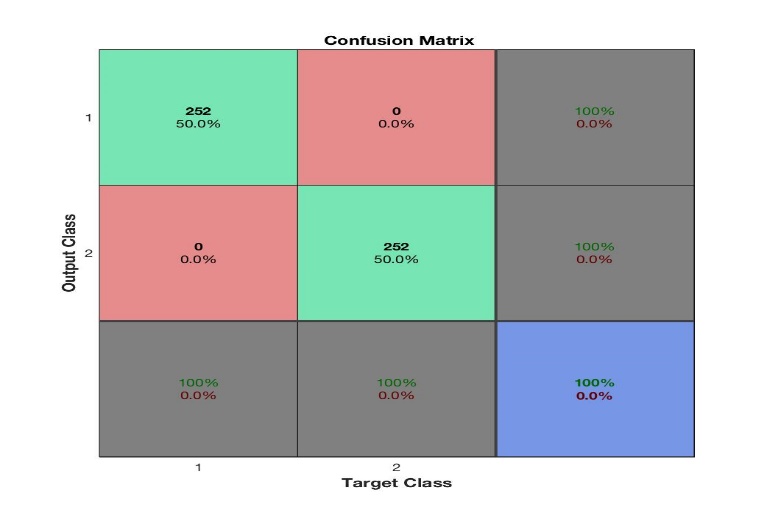
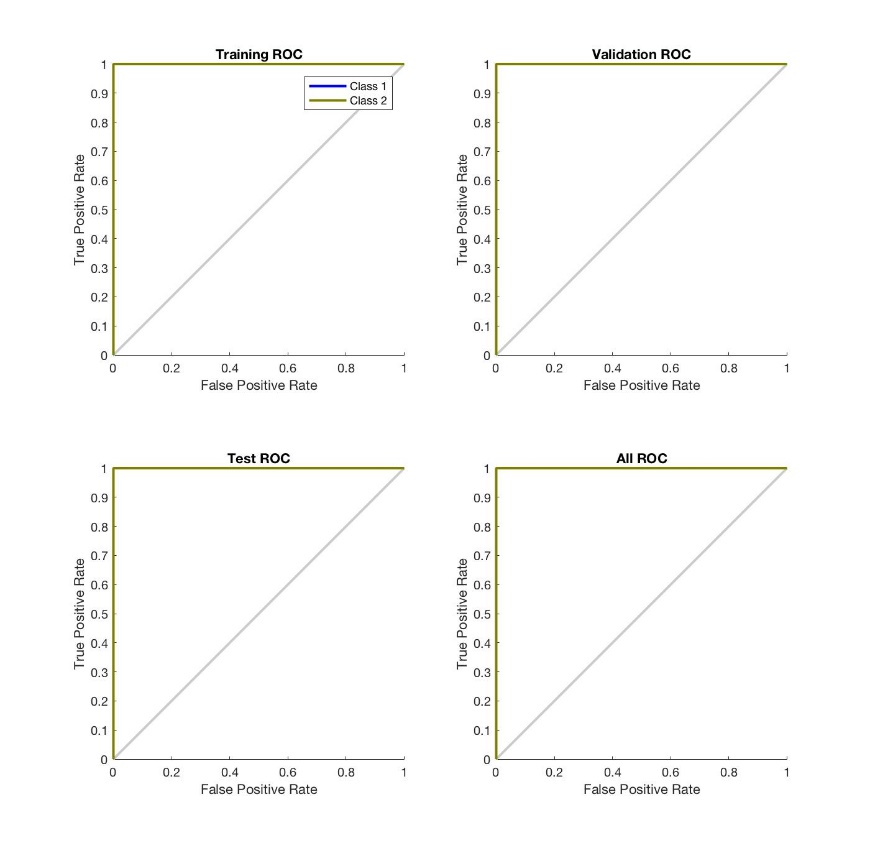
****

Figure 4.4

As one can easily observed from the plotted ROC curves for Training ROC, Validation ROC, Test ROC and All ROC, the classified outputs of this method is as expected. One can see the curve is hugged close the top-left corner.

****

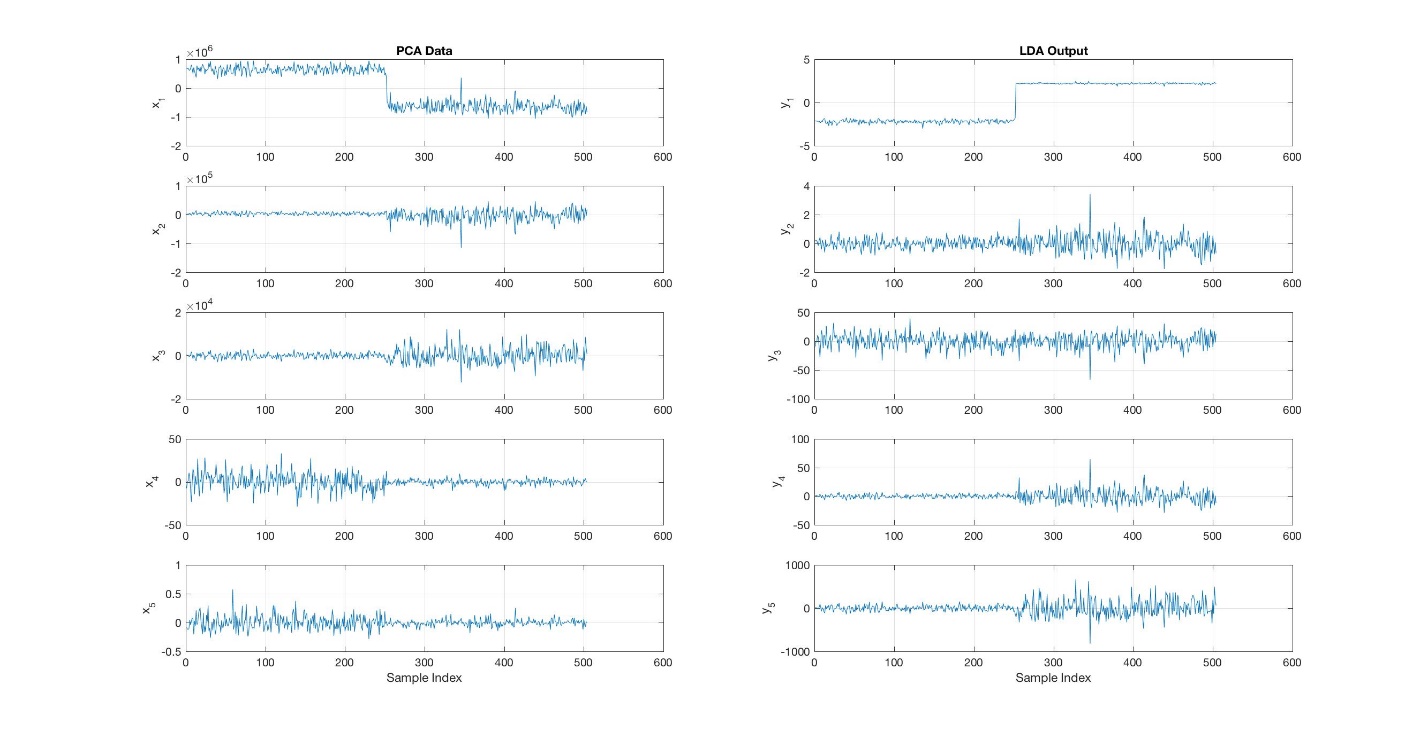
****

Figure 4.8

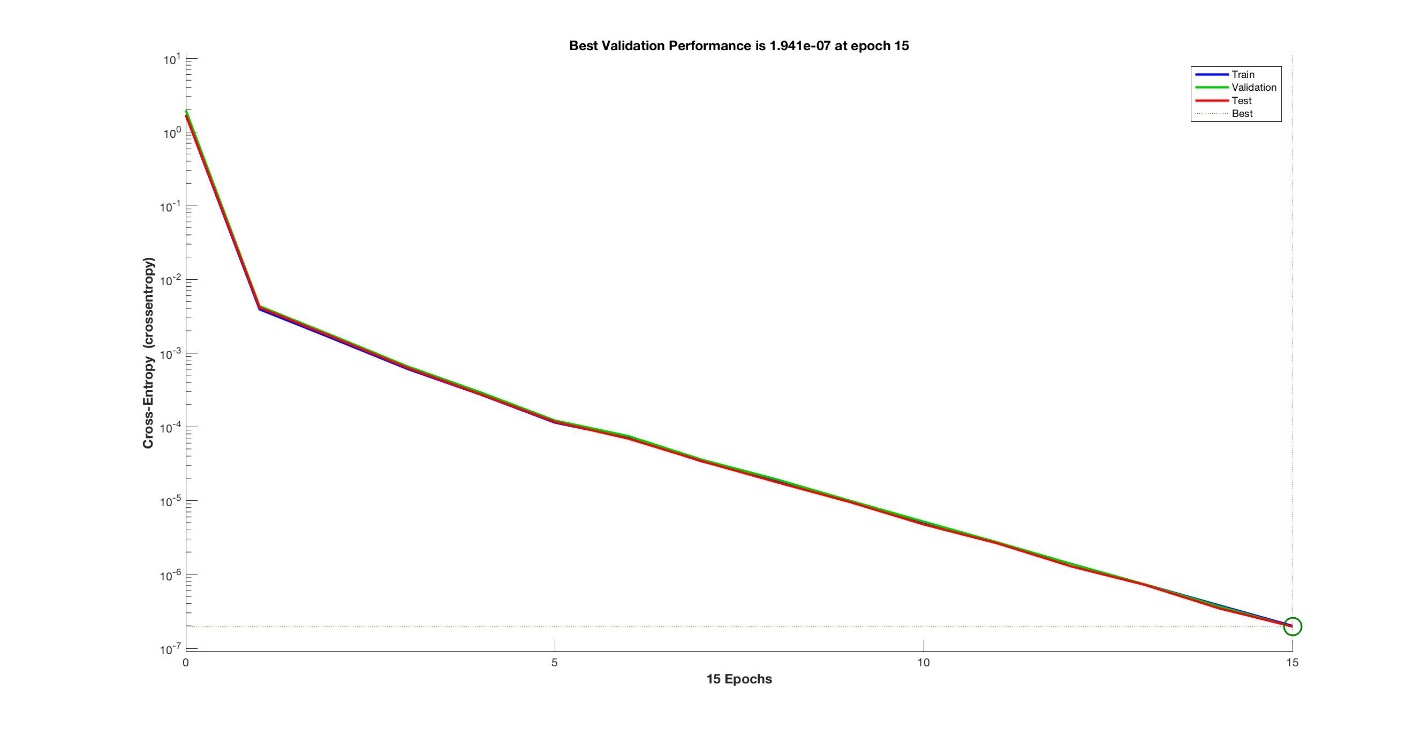
****

Figure 4.10

Figure 4.11

Figure xxx shows the neural network training state plot. It also shows validation check at epoch xxx and highlights that there is no validation failure up to this epoch.

* 1. **TCP flooding Result phase of proposed model (PCA+LDA+ALO)**

For PCA and LDA only method, the input feature set is used directly to train a standard feedforward neural network that can be used to detect abnormal network traffic patterns in IDPS. The standard neural network that is used is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The number of hidden neurons is set to 10. The number of output neurons is set to 2, which is equal to the number of categories in the target vector (i.e. the normal network traffic and the attack network traffic).

One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.To address the overfitting problem, the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases.

The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over fit the data, the error on the validation set typically begins to rise. The network weights and biases are saved at the minimum of the validation set error.

The third subset is the test set. The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set. It can be seen clearly, that the feedforward network consists of a series of three layers in the setup. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

The diagram also reveals some other important characteristics of the neural network in question. For example, the total input data set is divided randomly divided into the three subsets using the division parameters specified in the above, i.e. with the ratios for training, testing and validation are 0.7, 0.15 and 0.15, respectively. The Scaled Conjugate Gradient schema is used to train for the optimal values of network weights and bias. The training performance is evaluated during the training process using Cross Entropy metric calculated based on network responses.

The summary diagram also indicates that it took in total 11 iterations for the neural network to converge after the validation errors have increased for 6 consecutive iterations (Validation Checks).

The classification results using trained neural network based on PCA method is shown in Figure 2 in the form of confusion matrix. A Confusion Matrix is a visual performance assessment of a classification algorithm in the form of a table layout or matrix. Each column of the matrix represents predicted classifications and each row represents actual defined classifications. The diagonal cells show for how many (and what percentage) of the examples the trained network correctly estimates the classes of observations. That is, it shows what percentage of the true and predicted classes match.

The off diagonal cells show where the classifier has made mistakes. The column on the far right of the plot shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the bottom right of the plot shows the overall accuracy.The confusion matrix represents the metric scores in a compact way so that it also makes easy to calculate other representative evaluation measures such as Recall, Precision, Accuracy and F-Measure etc.

Figure 2 shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 352 network records (70%) used for neural network training purpose, 76 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Training Confusion Matrix as example, 175 network traffic packages are correctly classified as normal.

This corresponds to 49.7% of all 352 training samples. Similarly, 173 network traffic packages are correctly classified as attack. This corresponds to 50% of the total training samples. Overall, 98.9% of the predictions on the training samples are correct and 1.1% are wrong classifications. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies. Thus, the trained model made in total 497 correction predictions, including 252 normal traffic and 245 attack attempts. There are in total 7 instances where the model misclassifies the real attacks to be normal traffic (false negative). This means the overall classification result has achieve 98.6% of accuracy, with the overall error rate to be 1.4%.

****

Figure 4.12

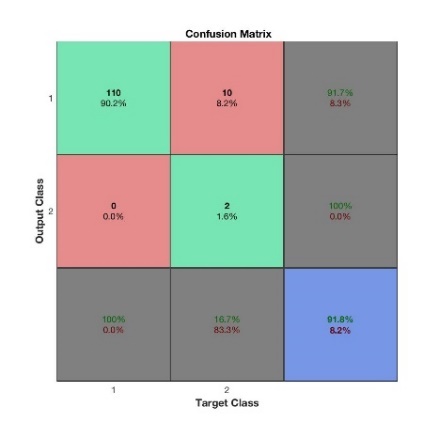
****

Figure 4.13

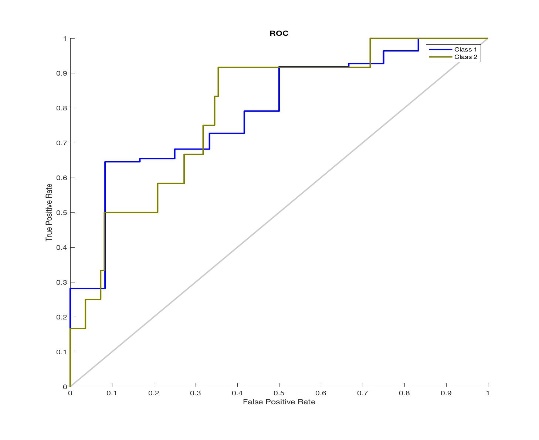
****

Figure 4.14

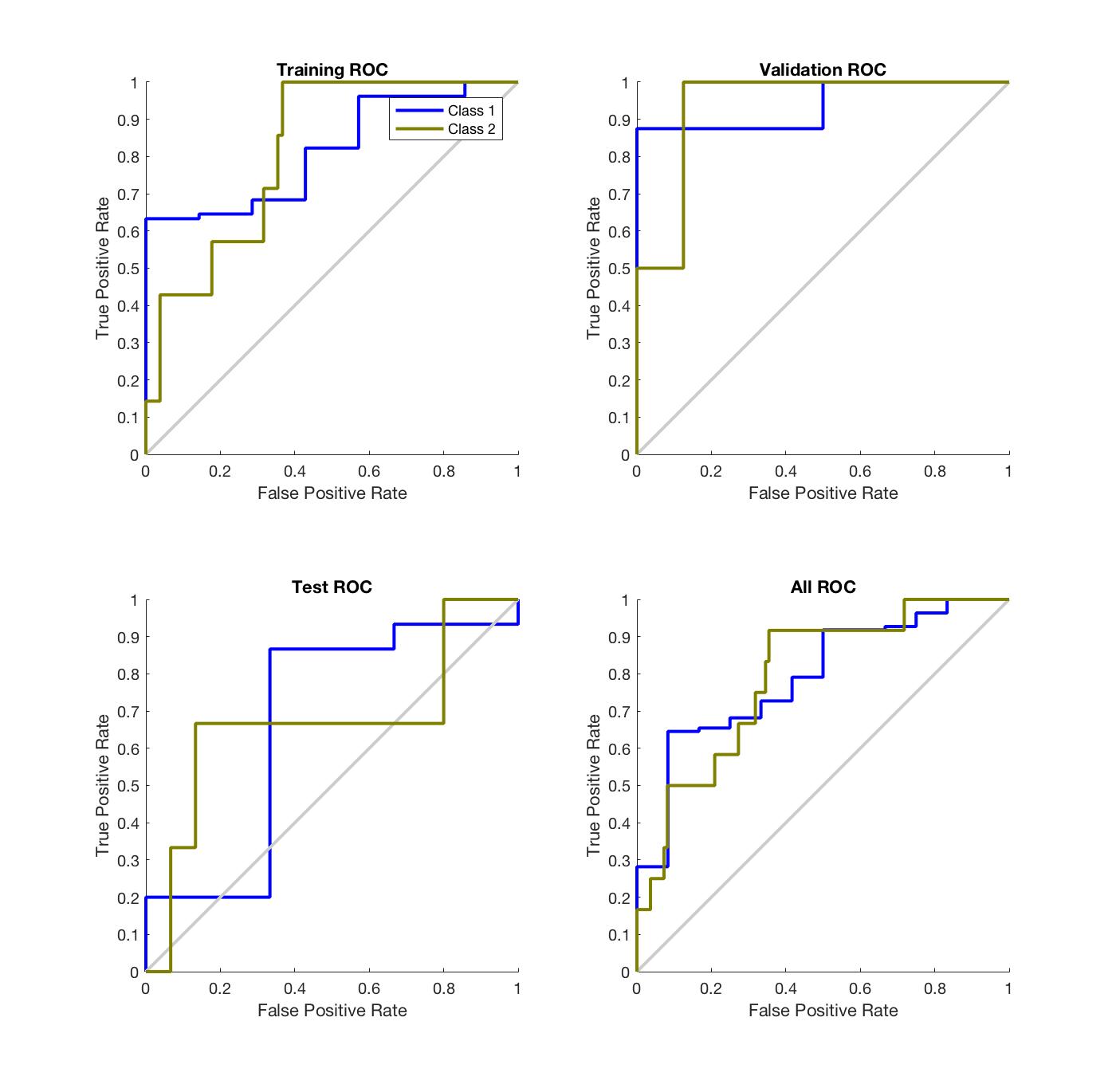
****

Figure 4.15

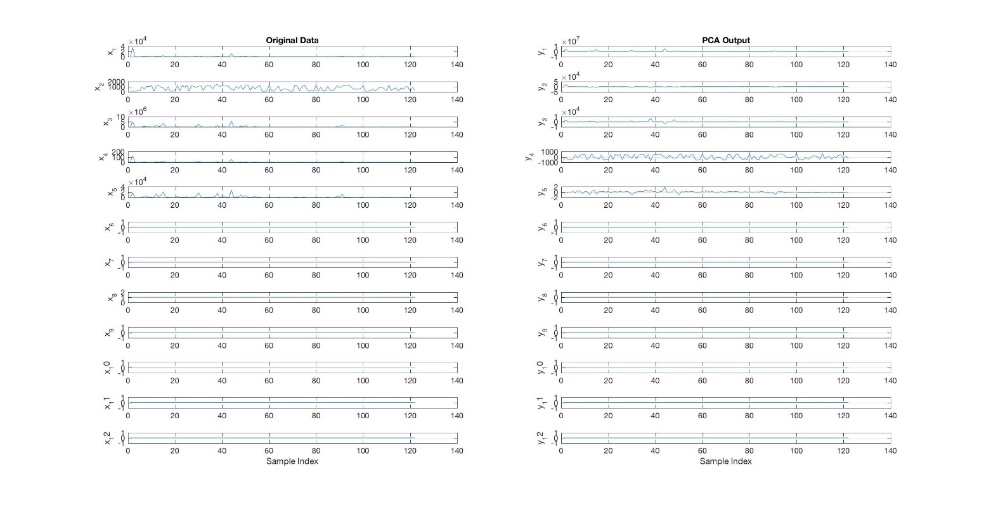
****

Figure 4.16

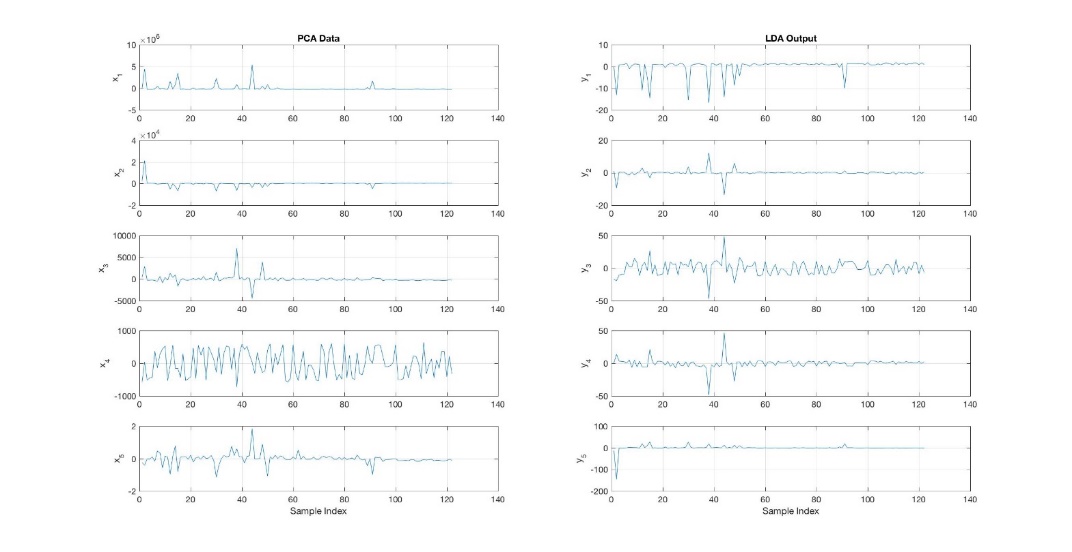
****

Figure 4.18

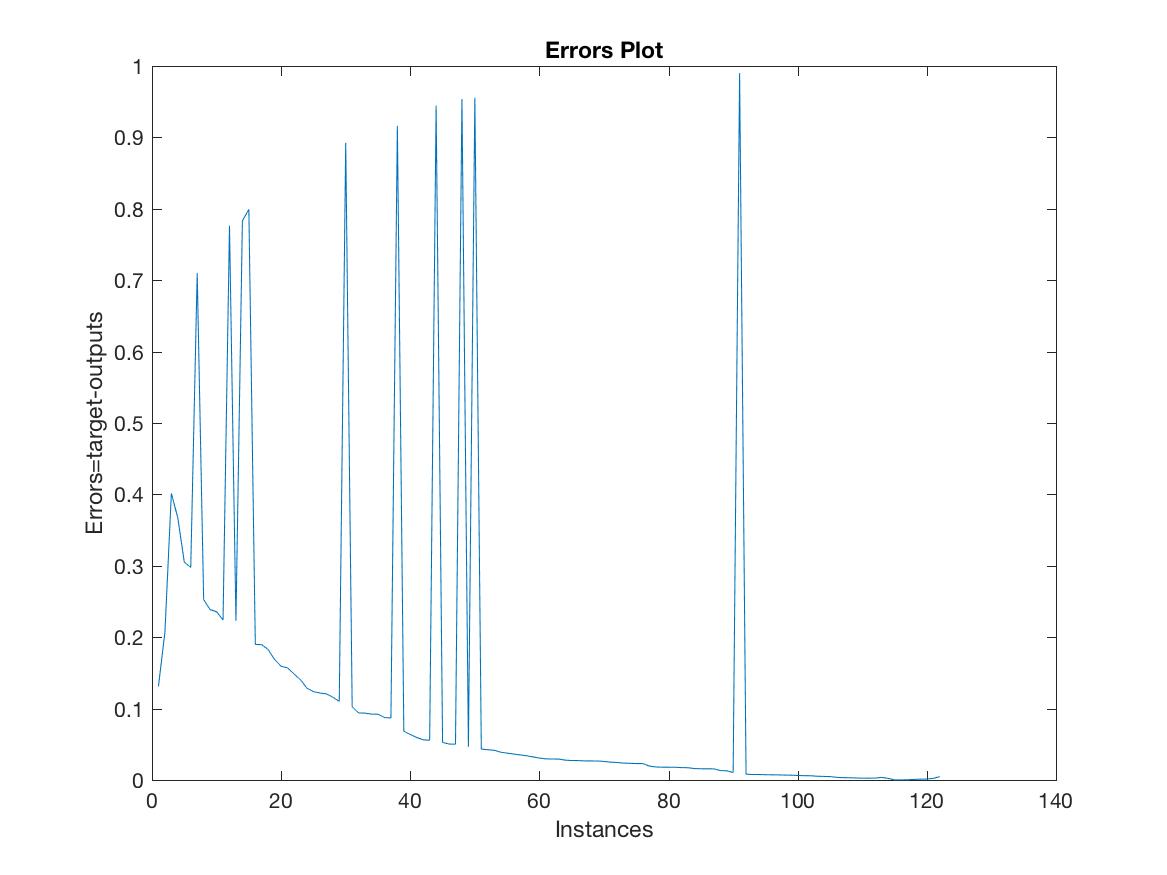
****

Figure 4.19

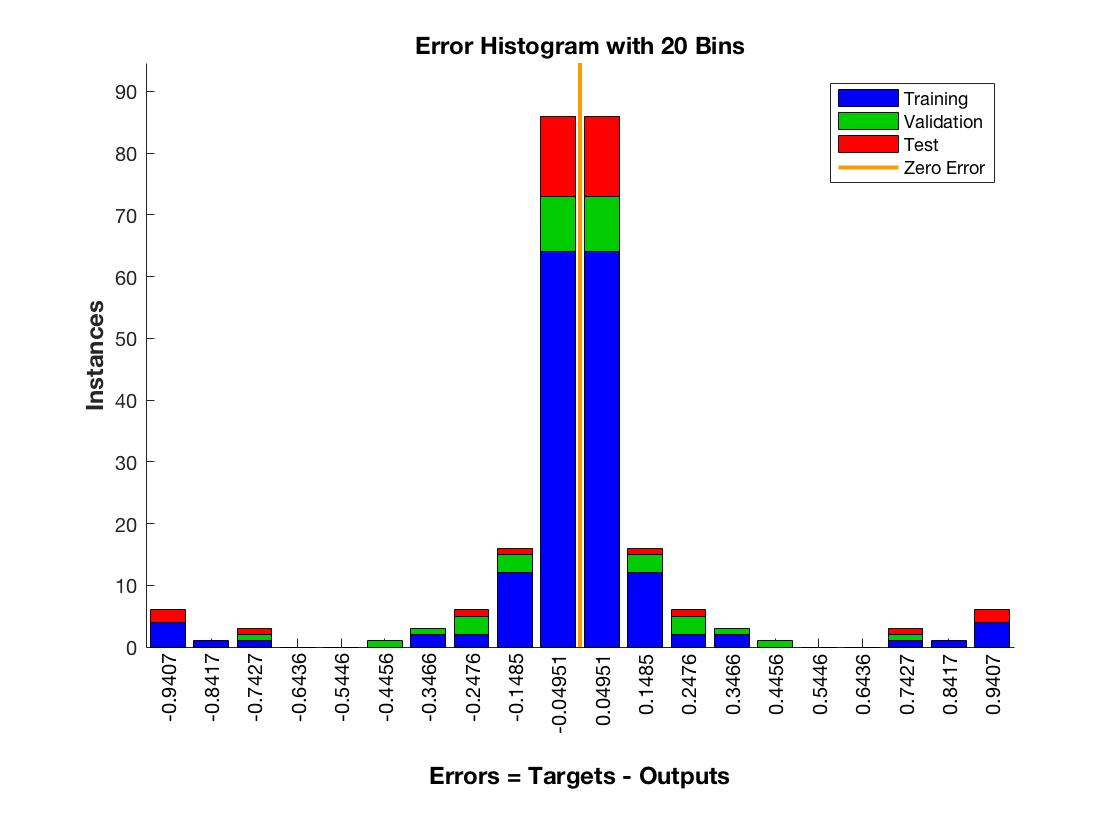
****

Figure 4.20

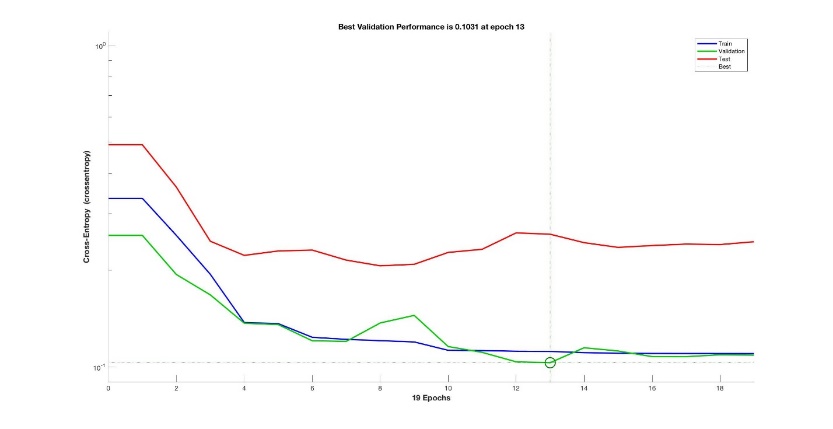
****

Figure 4.21

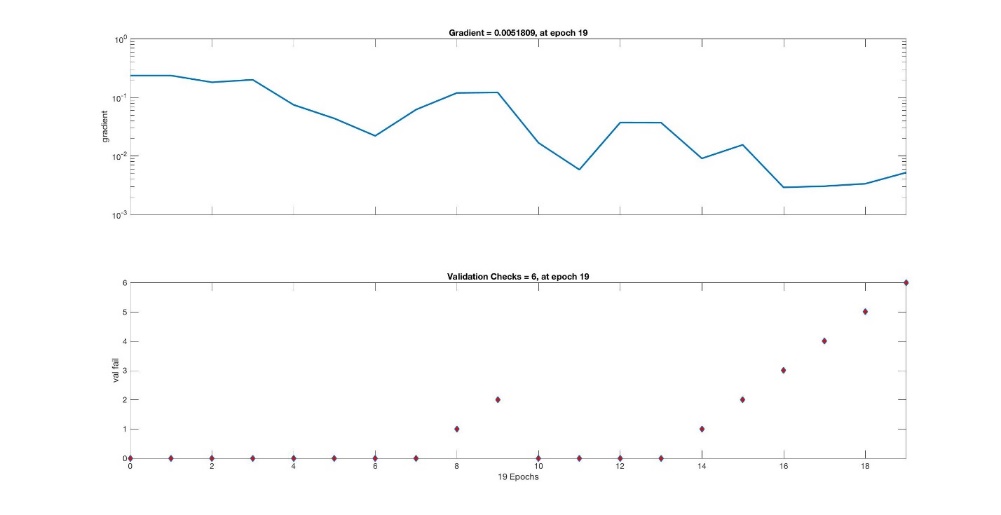
****

Figure 4.22

* + 1. **UDP flooding Result phase of proposed model (PCA+LDA+ALO)**

For PCA and LDA only method, the input feature set is used directly to train a standard feedforward neural network that can be used to detect abnormal network traffic patterns in IDPS. The standard neural network that is used is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer. The number of hidden neurons is set to 10. The number of output neurons is set to 2, which is equal to the number of categories in the target vector (i.e. the normal network traffic and the attack network traffic).

One of the problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.To address the overfitting problem, the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases.

The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over fit the data, the error on the validation set typically begins to rise. The network weights and biases are saved at the minimum of the validation set error.

The third subset is the test set. The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set. It can be seen clearly, that the feedforward network consists of a series of three layers in the setup. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

The diagram also reveals some other important characteristics of the neural network in question. For example, the total input data set is divided randomly divided into the three subsets using the division parameters specified in the above, i.e. with the ratios for training, testing and validation are 0.7, 0.15 and 0.15, respectively. The Scaled Conjugate Gradient schema is used to train for the optimal values of network weights and bias. The training performance is evaluated during the training process using Cross Entropy metric calculated based on network responses.

The summary diagram also indicates that it took in total 11 iterations for the neural network to converge after the validation errors have increased for 6 consecutive iterations (Validation Checks).

The classification results using trained neural network based on PCA method is shown in Figure 2 in the form of confusion matrix. A Confusion Matrix is a visual performance assessment of a classification algorithm in the form of a table layout or matrix. Each column of the matrix represents predicted classifications and each row represents actual defined classifications. The diagonal cells show for how many (and what percentage) of the examples the trained network correctly estimates the classes of observations. That is, it shows what percentage of the true and predicted classes match.

The off diagonal cells show where the classifier has made mistakes. The column on the far right of the plot shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the bottom right of the plot shows the overall accuracy.The confusion matrix represents the metric scores in a compact way so that it also makes easy to calculate other representative evaluation measures such as Recall, Precision, Accuracy and F-Measure etc.

Figure 2 shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. As mentioned before, there are 352 network records (70%) used for neural network training purpose, 76 network records (15%) used for validation and test respectively. For each confusion matrix, the first two diagonal cells show the number and percentage of correct classifications by the trained network. Take Training Confusion Matrix as example, 175 network traffic packages are correctly classified as normal.

This corresponds to 49.7% of all 352 training samples. Similarly, 173 network traffic packages are correctly classified as attack. This corresponds to 50% of the total training samples. Overall, 98.9% of the predictions on the training samples are correct and 1.1% are wrong classifications. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies. Thus, the trained model made in total 497 correction predictions, including 252 normal traffic and 245 attack attempts. There are in total 7 instances where the model misclassifies the real attacks to be normal traffic (false negative). This means the overall classification result has achieve 98.6% of accuracy, with the overall error rate to be 1.4%.

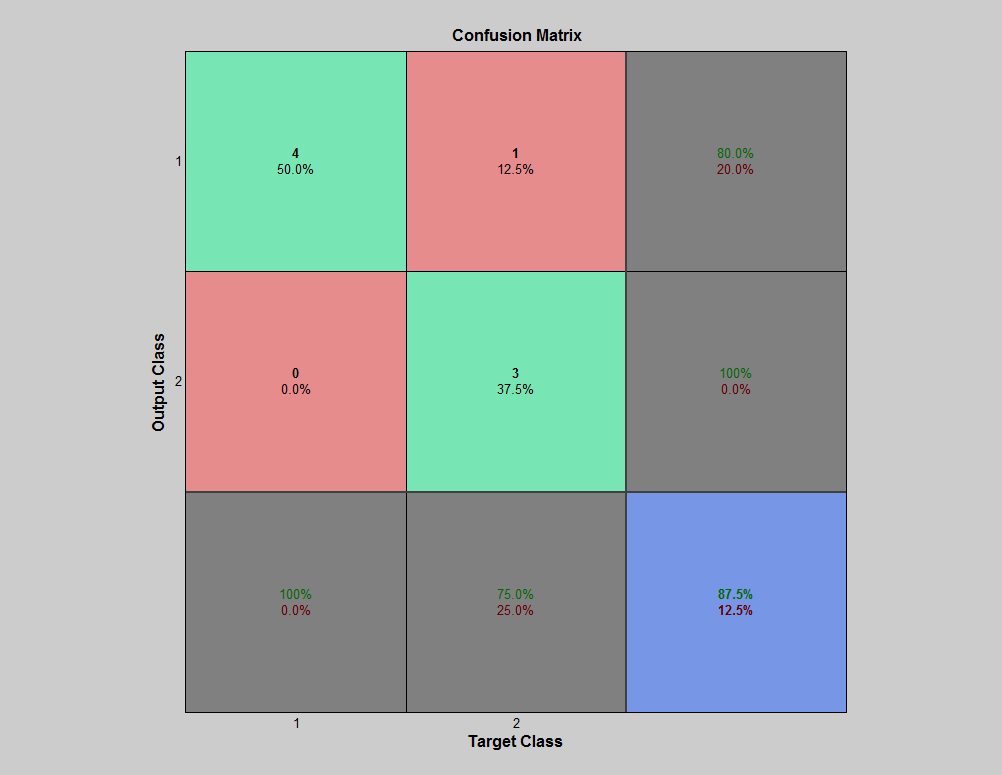
****

Figure 4.23

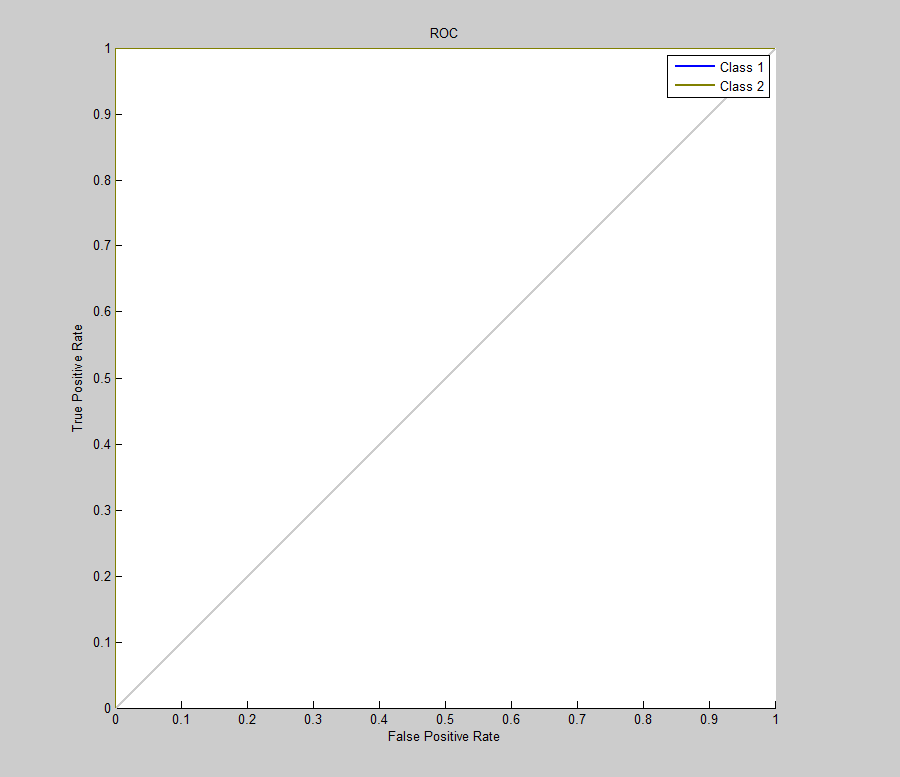
****

Figure 4.24

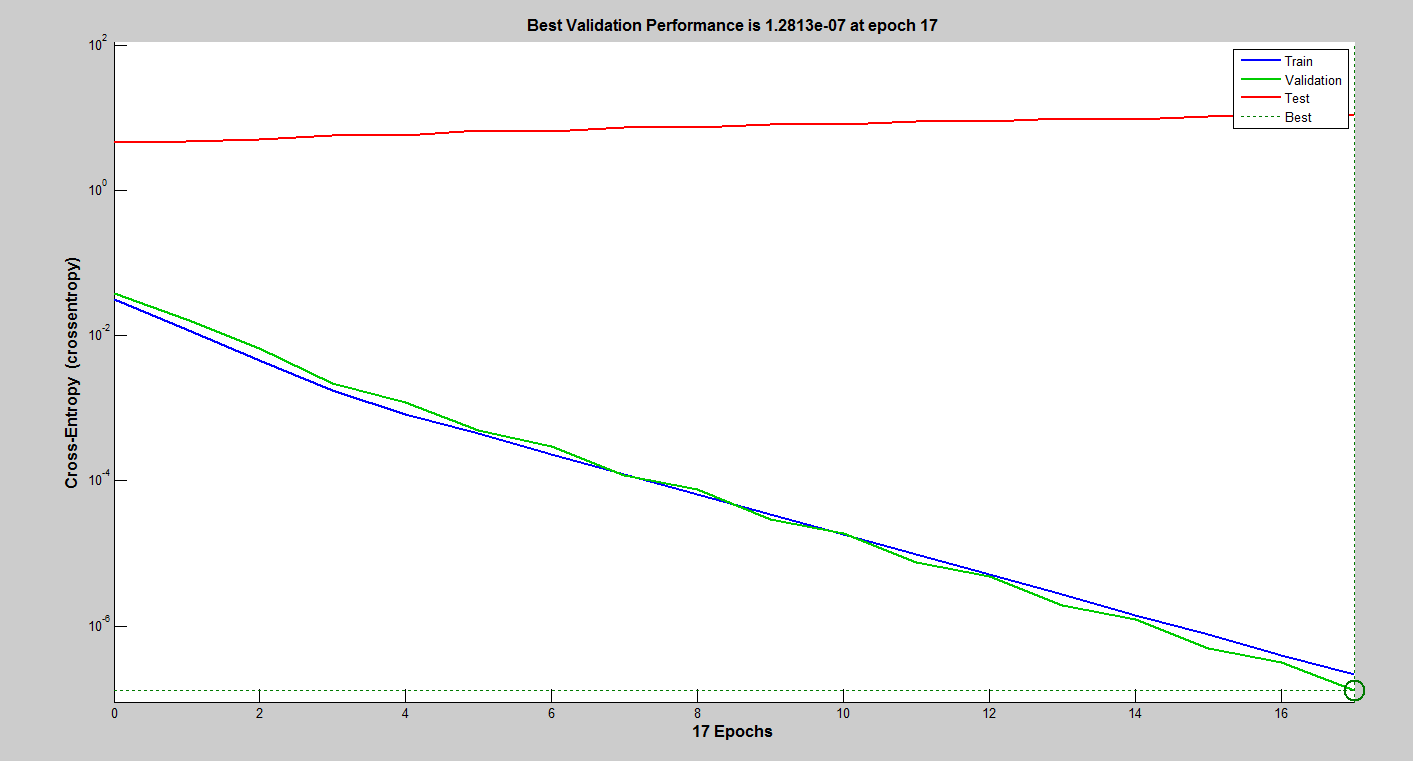
****

Figure 4.4

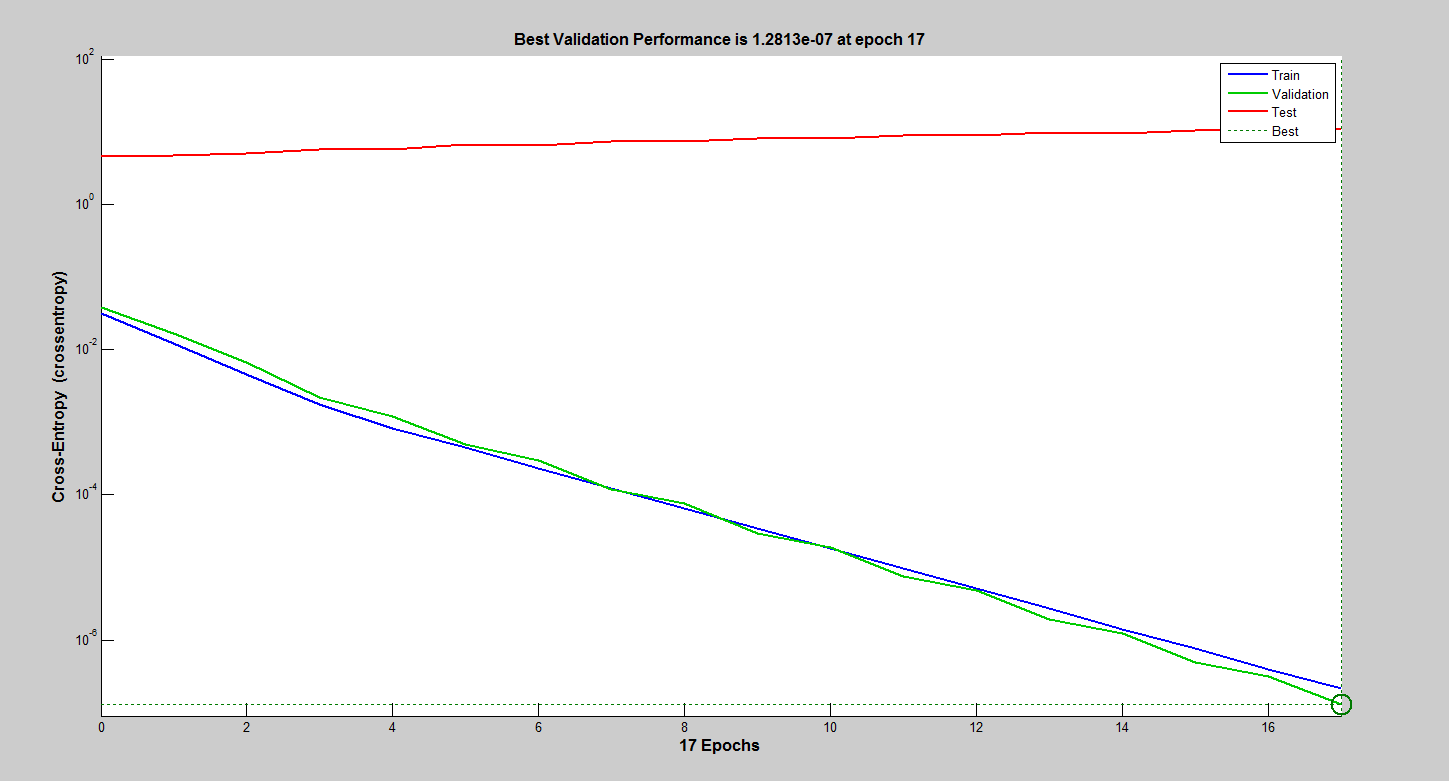
****

Figure 4.4

Figure 4.4

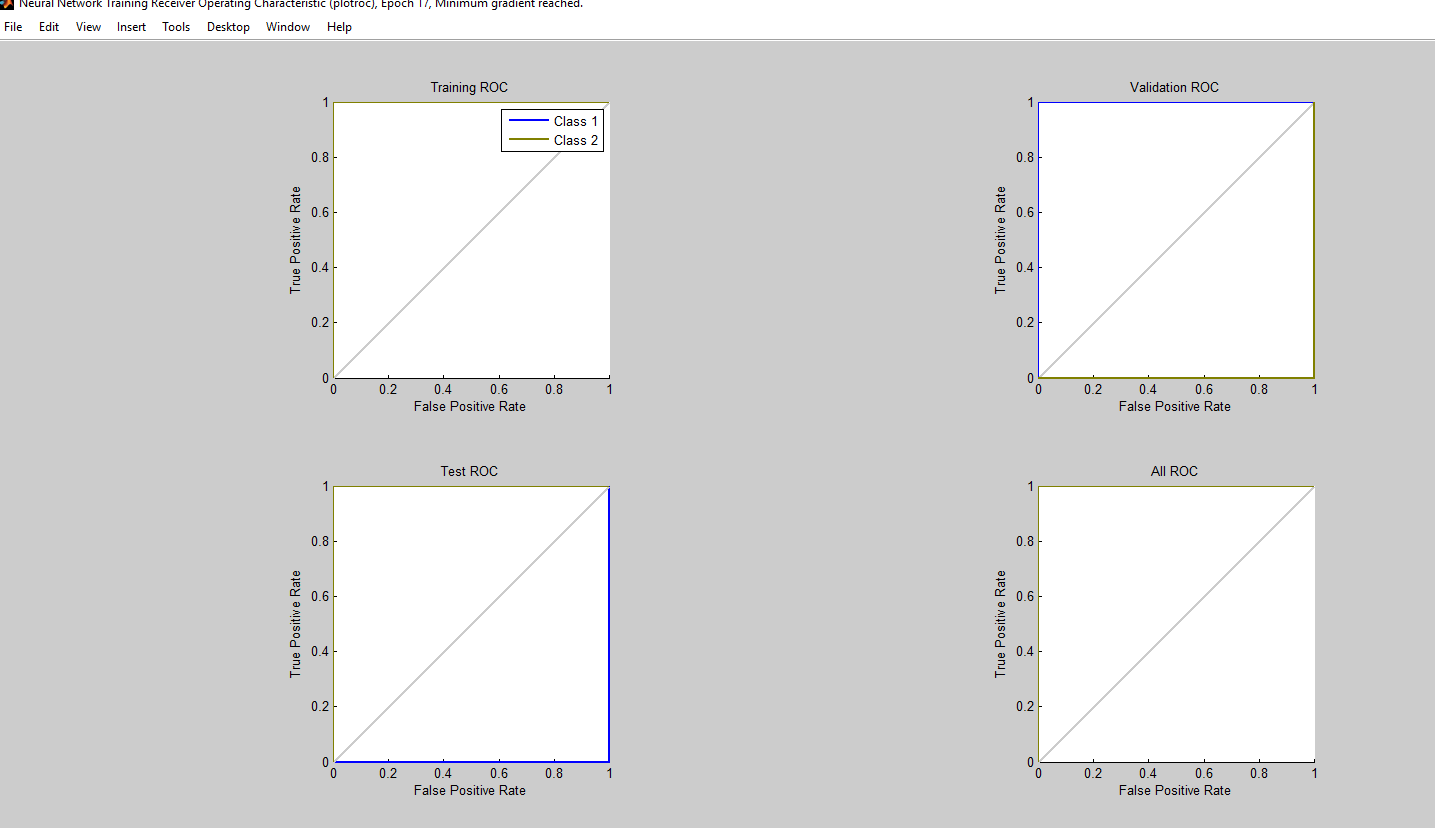
****

Figure 4.4

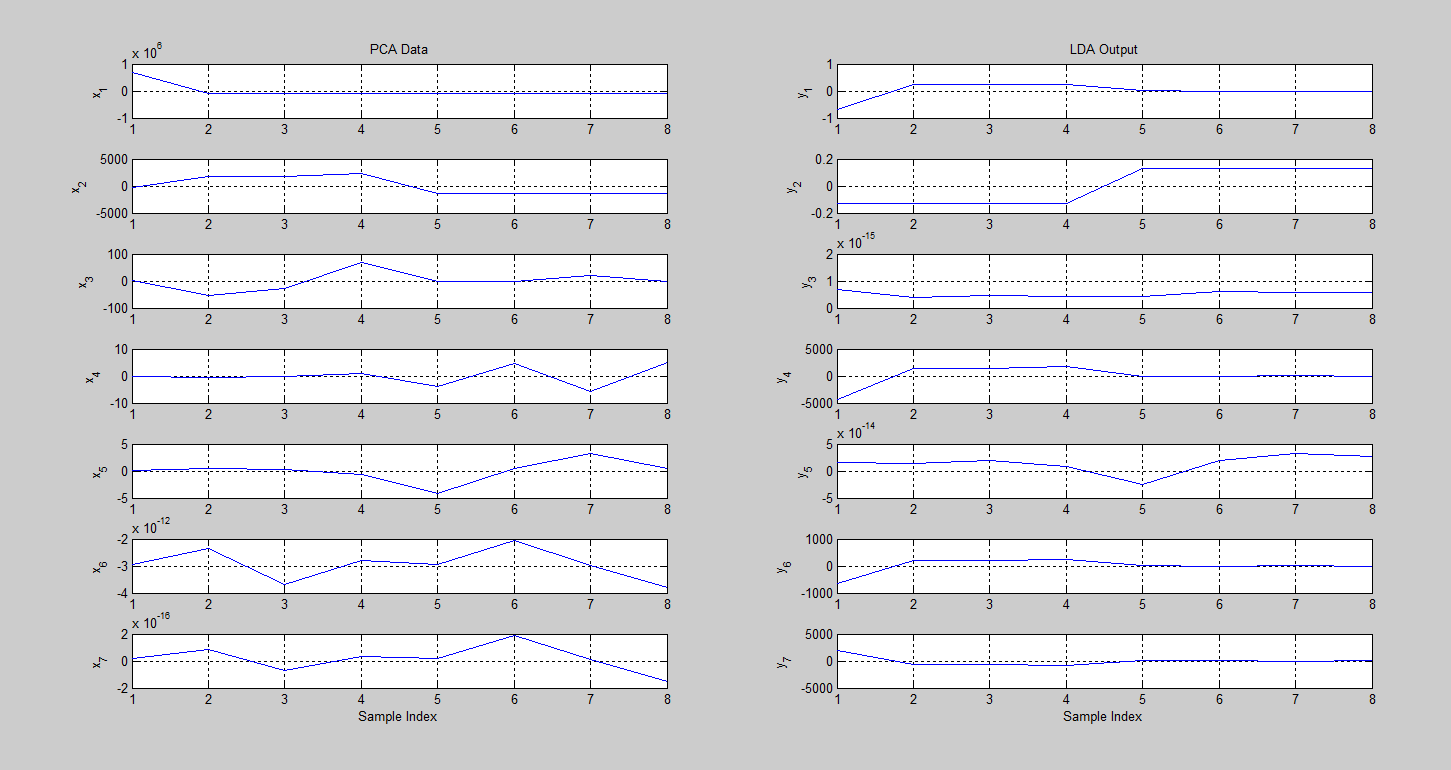
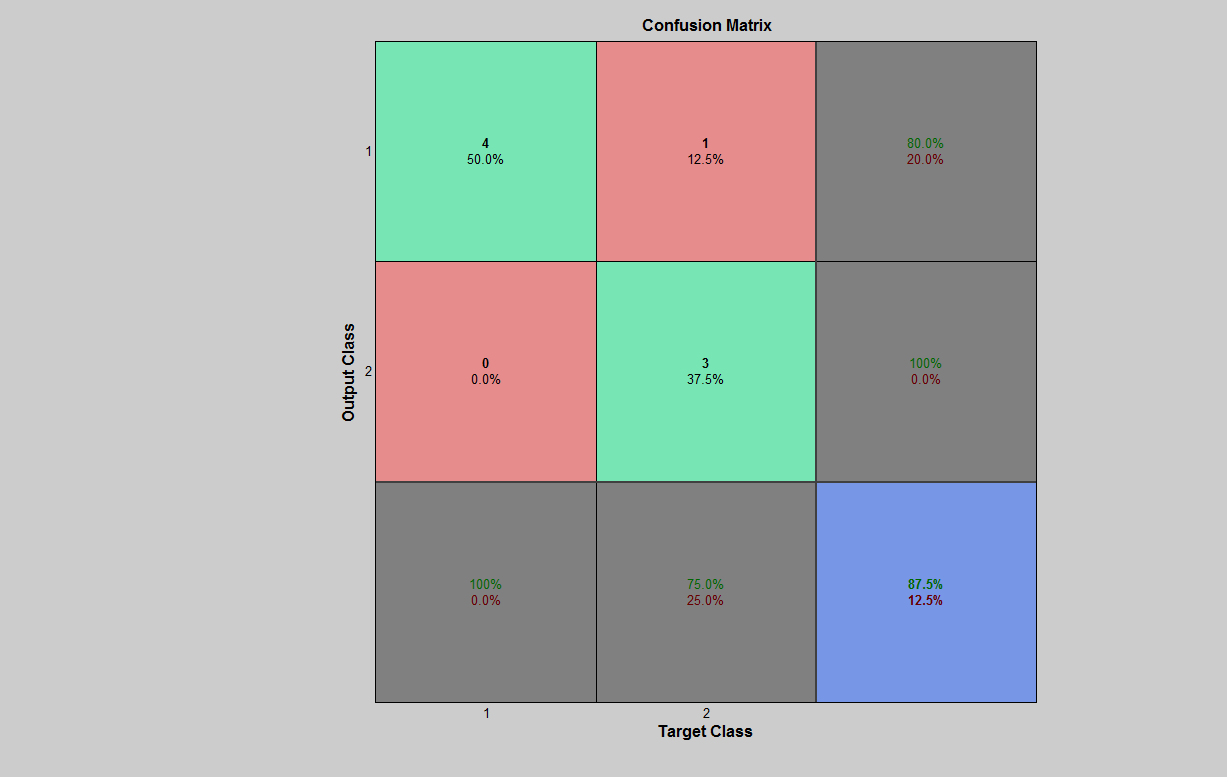
****

Figure 4.4

**4.8 Black list on Hypervisor results**

**As shown in Appendix x, x2,x2**

**For UDP black list have 252 blocked IP’s , for TCP flood have xxx IP’s and for TCP SYNC have xxx IP’S**

**Figure 4.x summarized the blocked IPS**

**4.11 Chapter summary**

This chapter has achieved the designed model into phases: implementation phase, and results. The most important

Moore, David, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. 2006. "Inferring internet denial-of-service activity." *ACM Transactions on Computer Systems (TOCS)* 24 (2):115-139.

Tomcat, Apache. 2016. "Tomcat 9 Software Downloads."