TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

THESIS NO.: 072/MSCS/667

NETWORK INTRUSION DETECTION USING RESILIENT BACKPROPAGATION

by

Shyam Dahal

A THESIS

SUBMITTED TO DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

NOVEMBER, 2017

NETWORK INTRUSION DETECTION USING RESILIENT BACKPROPAGATION

BY:

SHYAM DAHAL

072/MSCS/667

SUPERVISED BY:

Er. BABU RAM DAWADI

ATHESIS SUBMITTED TO DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PULCHOWK CAMPUS

INSTITUTE OF ENGINEERING

TRIBHUVAN UNIVERSITY

LALITPUR, NEPAL

NOVEMBER, 2017

COPYRIGHT

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

APPROVAL PAGE

The undersigned certify that it has been read and recommended to the Department of Electronics and Computer Engineering for acceptance, a report of thesis entitled **"Network Intrusion Detection using Resilient Backpropagation"**, submitted by **Mr. Shyam Dahal** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"**.

_____

Er. Babu Ram Dawadi
Supervisor,
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering.

_____

Er. Manoj Ghimire
External Examiner,
Chief Executive Officer,
nLocate Pvt. Ltd**.**

_____

Prof. Dr. Subarna Shakya
Committee Chairperson,
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering.

Date of Approval: 10th November, 2017

DEPARTMENTAL ACCEPTANCE

The thesis entitled **"Network Intrusion Detection using Resilient Backpropagation"**, submitted by **Mr. Shyam Dahal** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"** has been accepted as a bonafide record of work independently carried out by him in the department.

_____

Dr. Dibakar Raj Pant
Head of Department,
Department of Electronics and Computer Engineering,
Pulchowk Campus,
Institute of Engineering,
Tribhuvan University,
Pulchowk, Nepal.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor **Er. Babu Ram Dawadi** for providing precious knowledge and suggestions about the thesis. I would also like to thank **Dr. Aman Shakya**, Coordinator, MSCSKE, for his constant guidance on research activity.

I am highly indebted to **Prof. Dr. Subarna Shakya, Prof. Dr. Shashidhar Ram Joshi, Dr. Dibakar Raj Pant, Dr. Sanjeeb Pandey, Er. Dinesh Baniya Kshatri, Er. Sharad Chandra Joshi** for teaching and assisting me with different resources and skills required for my thesis work. Last but not least, I am always grateful to everyone for their direct and indirect support to complete this thesis successfully.

ABSTRACT

Information is one of the most valuable possessions today. As the Internet expands both in number of hosts connected and number of services provided, security has become a key issue for the technology developers. With the growth of smart devices and internet technologies, anomalous traffic detection has become a major concern. This thesis is focused on the detection of attacks in a network by using Multilayer Perceptron (MLP) trained with Resilient Backpropagation. NSL-KDD dataset, an intrusion detection attacks database is used as an input dataset for network intrusion detection. In each record of the NSL-KDD dataset, there are 42 attributes, 41 attributes unfolding different features and a label assigned to each either as an attack type or as normal. The 2^{nd} (Protocol_type), 3^{rd} (Services), and 4^{th} (Flag) attributes are converted into numerical format. The 42^{nd} attribute is first classified under different category of attacks (DoS, Probe, R2L, U2R) or normal, then assigned a numerical value and finally, 5 bit code are assigned to each of them. Architecture of MLP is determined to have 41 neurons in the input layer, 30 neurons in the hidden layer, and 5 neurons in the output layer. The number of neurons in the hidden layer was fixed selecting that value for which Performance (MSE) was best. In this thesis, a Multilayer Perceptron is trained with Resilient Backpropagation algorithm and the research evaluates the performance of the algorithm. Out of the 47735 records, 40% of it is used for training neural network, 50% of it is used for validation, and 10% is used for testing. Different parameters (TP, FP, FN) are noted from the confusion matrix and recall rate, precision rate were calculated for each. The result showed overall recall rate and precision rate to be 99.8% and 99.83%. The overall detection rate of the system is found to be 96.7% which is better than any other existing backpropagation algorithms.

Keywords: *Resilient, Backpropagation, Dataset, Intrusion detection, NSL-KDD, Multilayer Perceptron, Performance, MSE*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BFGS	Broyden-fletcher-Goldfarb-Shanno
BP	Backpropagation
DOS	Denial of Service
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
KDD	Knowledge Discovery in Database
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NIDS	Network based IDS
NSL	Network Socket Layer
R2L	Remote to Local
SOM	Self Organizing Map
TCP	Transmission Control Protocol
U2R	User to Root
UDP	User Datagram Protocol

1. INTRODUCTION

1.1 Background

With the growth of smart devices and internet technologies, anomalous traffic detection has become a major concern. Attacks on computer infrastructure are becoming an increasingly serious problem nowadays, therefore several information security techniques are available today to protect information systems against unauthorized use, duplication, alteration, destruction, and viruses attacks. The number of intrusions into computer systems is growing. The reason is that new automated hacking tools are appearing every day, and these tools with variety of system vulnerability information are easily available on the web [1].

The word “intrusion” means the act of wrongfully entering upon, seizing, or taking possession of the property of another [2]. The attack detection tools are very important for providing safety in computer and network system. These tools fully depend on accuracy of attack detection. Moreover, the detection is mandatory for prevention of any attack. Therefore, accurate detection of an attack is very important. Several attempts have been done in the field of attack detection but they suffered many limitations such as time consuming statistical analysis, regular updating, non adaptivity, lack of accuracy, and flexibility. Therefore, an Artificial Neural Network (ANN) supports an ideal specification of an attack detection system and is a solution to the problems of previous systems. As a result, an Artificial Neural Network inspired by nervous system has become an interesting tool in the applications of attack detection systems due to its promising features. Attack detection by artificial neural networks is an ongoing area and thus interest in this field has increased among the researchers. Neural networks have the ability to classify patterns, and thus can be used in other aspects of intrusion detection systems such as attack classification and alert validation [1].

An unauthorized user who tries to enter in network or computer system is known as intruder. A system that detects and logs inappropriate activities is called as intrusion detection system (IDS). The intrusion detection systems can be classified into three categories: host based, network based and vulnerability assessment based. A host

based intrusion detection system evaluates information found on a single or multiple host systems, including contents of operating systems, system files and application files. While network based intrusion detection system evaluates information captured from network communications, analyzing the stream of packets traveling across the network. Packets are captured through a set of sensors. Vulnerability assessment based intrusion detection system detects vulnerabilities on internal networks and firewall [2].

There are two general methods of detecting intrusions into computer and network systems: anomaly detection and signature recognition. Anomaly detection techniques establish a profile of the subject's normal behavior (norm profile), compare the observed behavior of the subject with its norm profile, and signal intrusions when the subject's observed behavior differs significantly from its norm profile. Signature recognition techniques recognize signatures of known attacks, match the observed behavior with those known signatures, and signal intrusions when there is a match [2].

Attacks can be classified into four main categories**:**

1) Denial of Service Attack (DoS): In computing, a denial of service attack is an attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled. There are many varieties of denial of service (DoS) attacks. Some DoS attacks (like a mailbomb, Neptune, or smurf attack) abuse a perfectly legitimate feature. Others (teardrop, Ping of Death) create malformed packets that confuse the TCP/IP stack of the machine that is trying to reconstruct the packet. Still others (apache2, back, syslogd) take advantage of bugs in a particular network detection.

2) User to Root Attack (U2R): User to root exploits are a class of exploits in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, from social engineering) and is able to exploit some vulnerability to gain root access to the system. There are several types of user to root attacks. The most common is the buffer overflow attack. Buffer overflows occur when a program copies too much

data into a static buffer without checking to make sure that the data will fit. Another example is the loadmodule attack which exploits programs that make assumptions about the environment in which they are running.

3) Remote to Local Attack (R2L): A remote to local attack occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine. There are many possible ways an attacker can gain unauthorized access to a local account on a machine. Some of the attacks exploit buffer overflows in network server software (imap, named, sendmail). The Dictionary, Ftp-Write, Guest, and Xsnoop attacks all attempt to exploit weak or misconfigured system security policies. In the Xlock attack, a remote attacker gains local access by fooling a legitimate user who has left their X console unprotected, into revealing their password.

4) Probing Attack: In recent years, a growing number of programs have been distributed that can automatically scan a network of computers to gather information or find known vulnerabilities. An attacker with a map of which machines and services are available on a network can use this information to look for weak points. Some of these scanning tools (satan, saint, mscan) enable even a very unskilled attacker to very quickly check hundreds or thousands of machines on a network for known vulnerabilities. Other examples of probing attack includes inside sniffer, Ipsweep, resetscan, etc.

1.1.1 Artificial Neural Networks

Artificial Neural Networks or simply “Neural Nets” go by many names such as connectionist models, parallel distributed processing models, and neuromorphic systems [3]. Whatever the name, all these models attempt to achieve good performance via dense interconnection of simple computational elements. In this respect, artificial neural net structure is based on our understanding of biological nervous system. Neural net models have greatest potential in areas such as speech and image recognition. Instead of programming with sequential instructions, neural net models explore many competing hypotheses simultaneously using massively parallel

nets composed of many computational elements connected by links with variable weights [3].

Artificial Neural Networks (ANN) offer a different approach for analyzing data, and for recognizing patterns within that data, than traditional computing methods, therefore Artificial Neural Networks have been used in many applications such as: classification (Medical diagnosis, target recognition, character recognition, fraud detection, intruder detection and speech recognition), Function Approximation (machine diagnostics), and Data Mining (Clustering, data visualization and data extraction). Neural network is a universal classifier and with the proper choosing of its architecture it can solve any, even very complicated, classification task [3].

Feed forward neural network is an artificial neural network where connections between the units or nodes do not form a directed cycle [3]. The feed forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from input nodes, through the hidden node (if any) and to the output nodes.

Multilayer Perceptron (MLP) is a feed forward neural network. Figure 1.1 below shows the input layer, hidden layer(s) and output layer of Multilayer Perceptron.

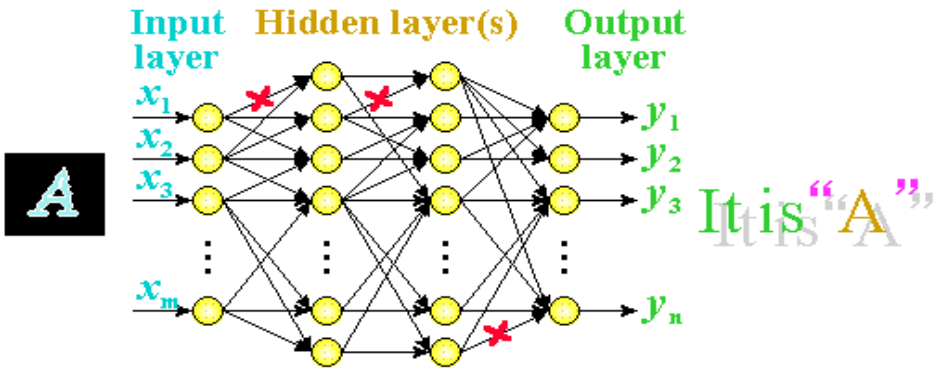

Figure 1.1: Multilayer Backpropagation.

(Source: Copied form [3])

1.1.2 Learning in Neural Networks

One of the powerful features of neural networks is learning. Learning in neural networks is carried out by adjusting the connection weights among neurons. It is

similar to a biological nervous system in which learning is carried out by changing synapses connection strengths among cells. The operation of a neural network is determined by the values of the interconnection weights. There is no algorithm that determines how the weights should be assigned in order to solve specific problems. Hence, the weights are assigned by a learning process.

Learning may be classified into two categories: Supervised Learning and Unsupervised Learning [3]. Supervised learning is the method in which every input pattern that is used to train the network is associated with an output pattern, which is the target or desired pattern. In simple word, supervised learning is the process where network is trained to produce/recognize the output given that it is fed with inputs along with the same output [3]. In unsupervised learning, in the other hand, the target output is not presented to the network. It is as if there is no teacher to present the desired output and hence, the system learns of its own by discovering and adapting to features of the input vectors/pattern [3].

1.1.3 Training and Testing of MLP

In ANN, generally initial weights and biases are set randomly with small values. Once, these values are set, the network becomes ready to be trained. Training a network generally means feeding the network with the training sequence. The training sequences are simply the vectors of input combinations along with the required output. The network processes the input vector, changes its internal weights and biases to give the result near to the output. After certain epochs of the training, the weights and biases are kept constant and real environment data is fed to the network to test the network. There are quite a few training algorithms developed during the years of time which provides good result in terms of how fast network converges to problem, how much memory does the network uses to produce the output etc. In this thesis, resilient backpropagation algorithm is used to train multilayer perceptron.

1.1.4 Backpropagation (BP)

Backpropagation is the most widely used neural network system. The MLP network is usually learned using the Backpropagation algorithm (BP). It uses the backpropagation rule for training. The backpropagation training algorithm is an

iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward perceptron and the desired output.

Figure 1.2 represents the flowchart of backpropagation algorithm.

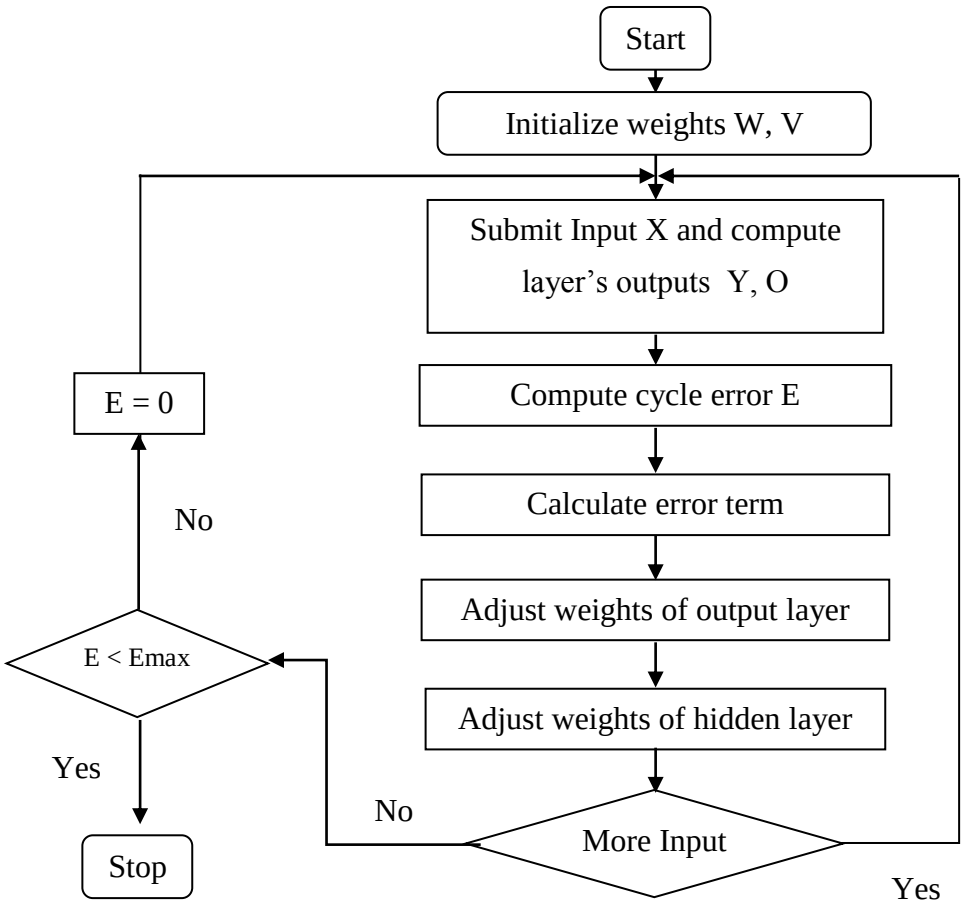

Figure 1.2: Backpropagation Algorithm

Backpropagation Algorithm Steps:

Step 0: Initialize the weights to small random values.

Step 1: Feed the training sample through the network and determine the final output.

Step 2: Compute the error for each output unit, for unit k it is:

$$\delta_k = (t_k - y_k) f'(y_{ink}) \quad \text{..... (1.1)}$$

Step 3: Calculate the weight correction term for each output unit, for unit k if it is:

$$\Delta W_{jk} = \alpha \delta_k Z_j \quad \text{..... (1.2)}$$

Step 4: Propagate the delta terms (error) back through the weights of the hidden units where the delta input for the j^{th} hidden unit is:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k W_{jk} \quad \text{.....(1.3)}$$

The delta term for the j^{th} hidden unit is:

$$\delta_j = \delta_{inj} f'(z_{inj}) \quad \text{..... (1.4)}$$

Step 5: Calculate the weight correction term for the hidden units:

$$\Delta V_{ij} = \alpha \delta_j Xi \quad \text{..... (1.5)}$$

Step 6: Update the weights:

$$W_{JK}(new) = W_{jk}(old) + \Delta W_{jk}(for\_output\_layer) \quad \text{..... (1.6)}$$

$$V_{ij}(new) = V_{ij}(old) + \Delta V_{ij}(for\_hidden\_layer) \quad \text{..... (1.7)}$$

Step 7: Test for stopping (maximum cycles, small changes, etc).

Resilient backpropagation algorithm is used for training of multilayer perceptron.

1.1.5 Transfer (Activation) Function

A neuron may sum its inputs, or average them, or something entirely more complicated. Each of these behaviors can be represented mathematically, and that representation is called the transfer (activation) function [3].

MLP networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, because they compress an infinite input range into a finite output range.

The bipolar sigmoid function: $f(x) = -1 + \frac{2}{1+e^{-x}}$ ..... (1.8)

which has derivative of: $f'(x) = 0.5 \times [1 + f(x)] \times [1 - f(x)]$ ..... (1.9)

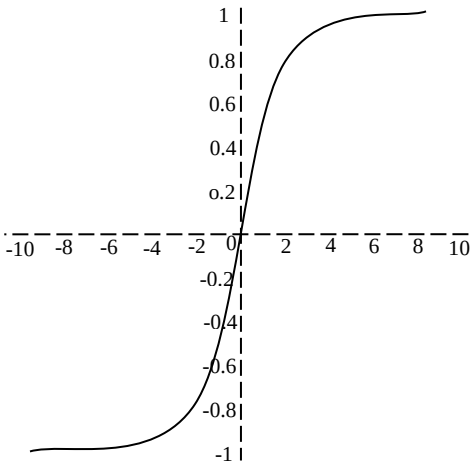

Figure 1.3: Bipolar sigmoid function

1.1.6 Resilient Backpropagation

Resilient backpropagation method is a learning heuristic for supervised learning in feedforward artificial neural networks. This is a first-order optimization algorithm. This algorithm was created by Martin Riedmiller and Heinrich Braun in 1992.

Resilient back propagation (Rprop), an algorithm that can be used to train a neural network, is similar to the more common (regular) back-propagation. But it has two main advantages over back propagation: First, training with Rprop is often faster than training with back propagation. Second, Rprop doesn't require us to specify any free parameter values, as opposed to back propagation which needs values for the learning rate (and usually an optional momentum term). The main disadvantage of Rprop is that it's a more complex algorithm to implement than back propagation.

Understanding Gradients and the Rprop Algorithm

Many machine learning algorithms, including Rprop, are based on a mathematical concept called the gradient. In order to understand gradient, its wise to observe the graph in Figure 1.4 in which the curve plots error vs. the value of a single weight. The idea here is that we must have some measure of error (there are several), and that the value of the error will change as the value of one weight changes, assuming we hold

the values of the other weights and biases the same. For a neural network with many weights and biases, there'd be graphs like the one in Figure 1.4 for every weight and bias.

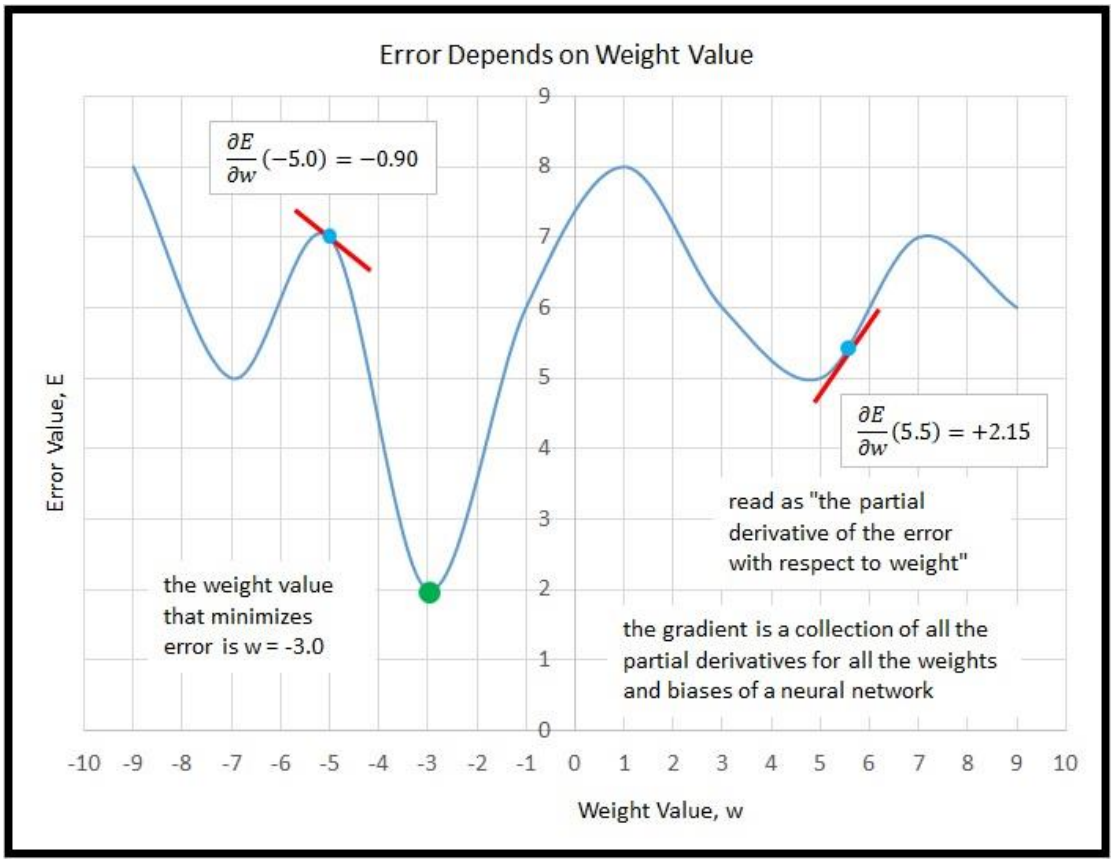

Figure 1.4: Partial derivatives and the gradient

(Source:https://visualstudiomagazine.com/articles/2015/03/01/resilientbackpropagation.aspx)

A gradient is made up of several "partial derivatives". A partial derivative for a weight can be thought of as the slope of the tangent line (the slope, not the tangent line itself) to the error function for some value of the weight. For example, in the figure, the "partial derivative of error with respect to weight" at weight = -5.0 is -0.90. The sign of the slope/partial derivative indicates which direction to go in order to get to a smaller error. A negative slope means go in the positive weight direction, and vice-versa. The steepness (magnitude) of the slope indicates how rapidly the error is changing and gives a hint at how far to move to get to a smaller error.

Partial derivatives are called partial because they only take one weight into account; the other weights are assumed to be constant. A gradient is just a collection of the all-partial derivatives for all the weights and biases. Note that although the word gradient is singular, it has several components. Also, the terms gradient and partial derivative (or just "the partial," for brevity) are often used interchangeably when the meaning is clear from the context.

During training, regular back propagation uses the magnitudes of the partial derivatives to determine how much to adjust a weight value. This seems very reasonable, but if we look at Figure 1.4 we can see a drawback to this approach. Suppose a weight has a current value of -5.0 and regular back propagation sees a fairly steep gradient and calculates a weight delta of +7.0. The new weight value will be -5.0 + 7.0 = 2.0 and so the weight has gone well past the optimum value at -3.0. On the next iteration of training, the weight could swing wildly back and overshoot again but in the other direction. This oscillation could continue and the weight for the minimum error would never be found.

With regular back-propagation algorithm, normally a small learning rate (0.0001 to 1) is used, which, along with the magnitude of the gradient, determines the weight delta in a training iteration. This means we likely won't overshoot an optimal answer, but it means training will be very slow as we creep closer and closer to a weight that gives minimum error.

The Rprop algorithm makes two significant changes to the back-propagation algorithm. First, Rprop doesn't use the magnitude of the gradient to determine a weight delta; instead, it uses only the sign of the gradient. Second, instead of using a single learning rate for all weights and biases, Rprop maintains separate weight deltas for each weight and bias, and adapts these deltas during training.

Similarly to the Manhattan update rule, Resilient backpropagation takes into account only the sign of the partial derivative over all patterns (not the magnitude), and acts independently on each "weight". For each weight, if there is a sign change of the partial derivative of the total error function compared to the last iteration, the update value for that weight is multiplied by a factor $\eta-$, where $\eta- < 1$. If the last iteration produces the same sign, the update value is multiplied by a factor of $\eta+$, where

$\eta+ > 1$. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, so as to minimize the total error function. η^+ is empirically set to 1.2 and η^- to 0.5.

$W_{k+1} = \eta^-. W_k$ if sign change. …….. (1.10)

$W_{k+1} = \eta^+. W_k$ if sign do not change. ……. (1.11)

where η^- and η^+ are the learningrate factors. These determine how much of a jump the update values will take in either the positive or negative direction.

Resilient Backpropagation Algorithm

```
while epoch < maxEpochs loop
  calculate gradient over all training items
  for each weight (and bias) loop
    if prev and curr partials have same sign
      increase the previously used delta
      update weight using  new delta
    else if prev and curr partials have different signs
      decrease the previously used delta
      revert weight to prev value
    end if
    prev delta = new delta
    prev gradient = curr gradient
  end-for
  ++epoch
end-while
return curr weights and bias values
```

Actually, training a neural network is the process of finding values for the weights and biases so that, for a set of training data with known input and output values, the computed outputs of the network closely match the known outputs. The most common technique used to train neural networks is the back-propagation algorithm. Back propagation requires a value for a parameter called the learning rate. The effectiveness of back propagation is highly sensitive to the value of the learning rate.

Rprop was developed by researchers in 1993 in an attempt to improve upon the back-propagation algorithm.

1.2 Problem Statement

IDS is Rule Based Monitoring and Controlling System, therefore, selection of algorithm used to define standard rule base is a major challenge. The selection of improper algorithm and model can maximize the occurrence of false alarm rate, high resource consumption, and low intrusion detection rate and may result inefficiency to entire system and may even lead to security vulnerabilities. The proper selection of classifier algorithm leads to increase in efficiency of IDS being implemented.

One of the most commonly used approaches in Intrusion Detection System's expert system is rule-based analysis. Rule-based analysis relies on sets of predefined rules that are provided by an administrator or created by the system.

Unfortunately, expert systems require frequent updates to remain sync to knowledge of different intrusions. This design approach usually results in an inflexible detection system that is unable to detect an attack if the sequence of events is even slightly different from the predefined profile. The problem may lie in the fact that the intruder is an intelligent and flexible agent while the rule based IDSs obey fixed rules. This problem can be tackled by the application of soft computing techniques (for example, Artificial Neural Network) in IDSs. Accuracy of detecting intrusion is also another major problem since a little deviation from the detection would affect the confidentiality, integrity, and availability of information.

1.3 Objectives

Objectives of the thesis are as follows:

- To detect and classify intrusion using Resilient Backpropagation algorithm and evaluate its performance.

2 LITERATURE REVIEW

Several research works have already been carried out and many research papers have been published regarding improvement on intrusion detection system (IDS) [1, 2, 4-13]. Each of the papers has focused on different algorithmic techniques being implemented in IDS with their resulted output in simulation tools as well.

The research work done by XiaoHang Yao [1] put forward an IDS combining with genetic algorithm and backpropagation. The intrusion detection system model presented in this paper adopts anomaly detection and misuse detection means. The system is composed of eight different modules. Five kinds of Neural Network technologies are described in this paper. The result of experiment shows that combining genetic algorithm with backpropagation efficiently enhances the learning speed of backpropagation neural network and improves the detection accuracy rate of IDS. Finally, a discussion of the future neural network technologies, which guarantee to enhance the detection efficiency of IDS is provided.

There is another research work performed by Farah Jemili, Montaceur Zaghdoud and Mohamad Ben Ahmed [2], in which Bayesian Network was used to build automatic intrusion detection system based on signature recognition. The goal of this work is to propose a method to propagate both the stochastic and the epistemic uncertainties, coming respectively from the uncertain and imprecise character of information, through the Bayesian model, in an intrusion detection context. However, some challenges in attack plan recognition were pointed and to minimize the risk, they have planned to apply their algorithms to alert streams collected from live networks and to integrate an expert system which can provide recommendations based on attack scenarios prediction.

Fred Cohen noted in 1984 that it is impossible to detect an intrusion in every case, and that the resources needed to detect intrusions grow with the amount of usage [4].The research work performed by Anderson, James P. [4] put forward the research on computer security threat monitoring and surveillance. The balance of this report outlines the considerations and general design of a system which provides an initial set of tools to computer system security officers for use in their jobs. The discussion does not suggest the elimination of any existing security audit data collection and

distribution. Rather it suggests augmenting any such schemes with information for the security personnel directly involved.

A system that can detect network intrusion at the time of attack is called a real-time Intrusion Detection System. A real-time Intrusion Detection System (IDS) captures the present network traffic data which is on-line data. There are only few papers on on-line (real-time) network IDS which are discussed below.

The research work performed in [6] put forward the research on an analysis of Intrusion Detection System using backpropagation neural network. In this paper, authors proposed a new learning methodology towards developing a novel intrusion detection system by backpropagation neural networks. The main function of Intrusion Detection System is to protect the resources from threats. It analyzes and predicts the behaviors of users, and then these behaviors will be considered an attack or a normal behavior.

There are several techniques which exist at present to provide more security to the network, but most of these techniques are static. We can test the proposed method by a benchmark intrusion dataset such as NSL-KDD to verify its feasibility and effectiveness. Results show that choosing good attributes and samples will not only have impact on the performance, but also on the overall execution efficiency. The proposed method can significantly reduce the training time required. Additionally, the training results are good. It provides a powerful tool to help supervisors analyze, model and understand the complex attack behavior of electronic crime.

Labib and Vemuri [7] developed a real-time IDS using Self Organizing Maps (SOM) to detect normal network activity and DoS attack. They preprocessed their dataset to have 10 features for each data record. Each record contained information of 50 packets. The system uses a structured SOM to classify real-time Ethernet network data. The authors were able to classify simulated DoS network attacks graphically as opposed to normal traffic by showing that the clustering of neurons was very different between the two. Puttini et al. [8] used a Bayesian classification model for anomaly detection to classify normal network activity and attack using a 3-month training dataset and a 1-month test dataset. In this paper, design and development of the IDS are considered in 3 main stages: normal behavior construction, anomaly detection, and model update. Detection and update algorithms for the special case of Gaussian

parametrical model are designed and evaluated with respect to their real-time features in a PC-like platform without any special hardware requirements. Amini et al. [9] designed a real-time IDS using two unsupervised neural network algorithms which are Adaptive Resonance Theory (ART) and Self-Organizing Map (SOM). They classified two attack types plus normal data during a 4-day experiment with a 27-feature dataset, where each feature captures number of occurrences of an event in each time interval. The detection results showed that the ART-2 gave higher detection speed and detection rate than the SOM. However, the attacks were not classified into types or categories. Su et al. [10] created a real-time network IDS using fuzzy association rules and conducted their experiments by using four computers with DoS attack types in WIN32. They could separate the normal network activity from network attacks but they did not identify the attack type.

In the research work carried by [15], the highest detection rate was found to be 99.794% but with reduction of features from 41 to 22. In the research paper [16], the detection rate was found to be 79.9% using BFGS quasi-Newton Backpropagation taking number of hidden layers 21. The detection rate from [17] using NSL-KDD is found to be 85.7% using Decision Tree (DT) based CART (Classification and Regression Tree) algorithm.

An intrusion detection tool called SNORT is described in [11]. SNORT has now become a commercial tool. Its attack signature rules are available only to their registered customers. The signature rules or patches have to be frequently updated and installed in order to detect current attack types.

3 METHODOLOGY

3.1 Conceptual Design

The proposed system is under study in two phases. Training, validation, and testing of multilayer perceptron is carried out using NSL-KDD Dataset (2009). Also, testing is performed using real-time data captured from an organization. These two major parts can also be explained with the help of block diagram that follows.

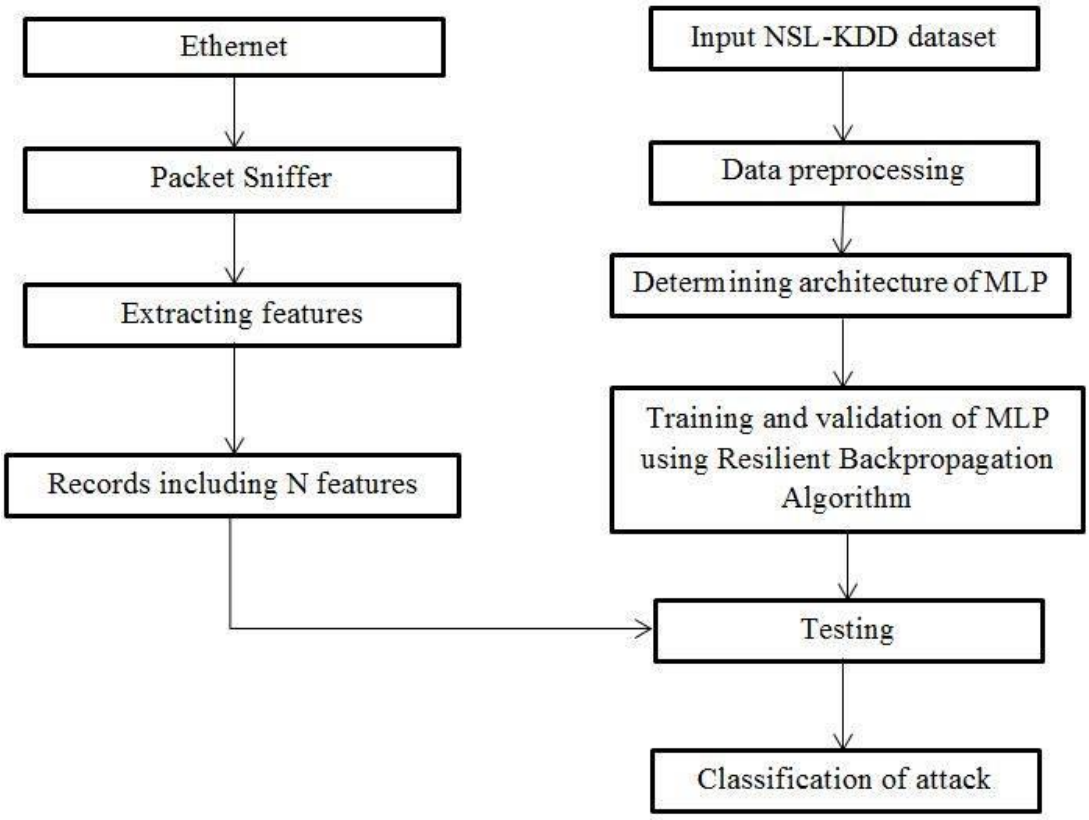

Figure 3.1: Block diagram of the proposed system

3.2 Input Dataset Collection

In this thesis, the source of data is NSL-KDD (2009). NSL-KDD is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset. Although, this new version of the KDD dataset still suffers from some of the problems and may not be a perfect representative of existing real networks, because of the lack of public

datasets for network-based IDSs, it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

The NSL-KDD data set has the following advantages over the original KDD data set:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

In each record of the NSL-KDD dataset, there are 41 attributes unfolding different features and a label assigned to each either as an attack type or as normal.

Table 3.1: Different features and its types of NSL-KDD dataset

Type	**Features**
Nominal	Protocol_type(2), service(3), flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numeric	Duration (1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18),

	num_access_files(19), num_outbound_cmds(20), count(23), srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29), diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

The details of the attributes namely the attribute name and their description are listed in the Table 5.2.

Table 3.2: Description of different types of attributes in NSL-KDD dataset.

Attribute No.	**Attribute Name**	**Description**
1	Duration	Length of time duration of the connection
2	Protocol_type	Protocol used in the connection
3	Service	Destination network service used
4	Flag	Status of the connection– Normal or error
5	Src_bytes	Number of data bytes transferred from source to destination in single connection
6	Dst_bytes	Number of data bytes transferred from destination to source in single connection
7	Land	If source and destination IP addresses and port numbers are equal, then this variable takes value 1 else 0
8	Wrong_fragment	Total number of wrong fragments

		in this connection
9	Urgent	Number of urgent packets in this connection (Urgent packets are packets with the urgent bit activated
10	Hot	Number of ‘hot’ indicators in the content such as entering a system directory, creating programs and executing programs
11	Num_failed_logins	Count of failed login attempts
12	Logged_in	Login status– 1 if successfully logged in otherwise 0
13	Num_compromised	Number of ‘compromised’ conditions
14	Root_shell	This variable takes value 1 if root shell is obtained otherwise 0
15	Su_attempted	1 if “su root” command is attempted or used otherwise 0
16	Num_root	Number of ‘root’ accesses or number of operations performed as a root in the connection
17	Num_file_creations	Number of file creation operations in the connection
18	Num_shells	Number of shell prompts
19	Num_access_files	Number of operations on access control files
20	Num_outbound_cmds	Number of outbound commands in an ftp session
21	Is_hot_login	1 if the login belongs to the ‘hot’ list i.e., root or admin, else 0
22	Is_guest_login	1 if the login is a ‘guest’ login otherwise 0

23	Count	Number of connections to the same destination host as the current connection in the past two seconds
24	Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds
25	Serror_rate	The percentage of connections that have activated the flag (4) among the connections aggregated in count (23)
26	Srv_serror_rate	The percentage of connections that have activated the flag (4) among the connections aggregated in srv_count (24)
27	Rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
28	Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)
29	Same_srv_rate	The percentage of connections that were to the same service among the connections aggregated in count (23)
30	Diff_srv_rate	The percentage of connections that were to different services among the connections aggregated in count (23)
31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections

		aggregated in srv_count (24)
32	Dst_host_count	Number of connections having the same destination host IP addresses
33	Dst_host_srv_count	Number of connections having the same port number
34	Dst_host_same_srv_rate	The percentage of connecctions that were to the same service among the connections aggregated in dst_host_count (32)
35	Dst_host_diff_srv_rate	The percentage of connections that were to different services among the connections aggregated in dst_host_count (32)
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port among the connections aggregated in dst_host_srv_count (33)
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in dst_host_srv_count (33)
38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4) among the connections aggregated in dst_host_count (32)
39	Dst_host_srv_serror_rate	The percentage of connections that have activated the flag (4) among the connections aggregated in dst_host_srv_count (33)
40	Dst_host_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated

		in dst_host_count (32)
41	Dst_host_srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)

3.3 Preprocessing

Data preprocessing is the most time consuming and complex task of preparing for subsequent analysis as per requirement for IDS model. The objectives of data preprocessing is to transform the raw input data into an appropriate format for subsequent analysis. The various steps involved in data preprocessing include merging data from data repositories, cleaning data to remove noise and duplicate observations and then selecting relevant observations as per the requirement at hand [12].

The 2nd, 3rd, and 4th attributes are only needed to be converted into numerical format as they are nominal in nature while other attributes are already numerical in nature. The 2nd attribute (protocol_type) are converted to numeric values as listed in the Table 5.3.

Table 3.3:Protocol types and corresponding numeric values assigned.

Protocol_type	**Numeric Values**
TCP	1
UDP	2
ICMP	3

Similarly, the 3rd attribute (service) and 4th attribute (flag) are converted to numeric values as shown in the Table 5.4 and Table 5.5 respectively.

Table 3.4: Numerical values assigned to different types of services

Service	**Numeric Values**	**Service**	**Numeric Values**
http	1	supdup	35
http_443	2	systat	36
http_8001	3	telnet	37
imap4	4	tim_i	38
IRC	5	time	39
iso_tsap	6	urh_i	40
Klogin	7	urp_i	41
Kshell	8	uucp	42
Ldap	9	uucp_path	43
Link	10	vmnet	44
Login	11	whois	45
Mtp	12	X11	46
name	13	Z39_50	47
netbios_dgm	14	auth	48
netbios_ns	15	bgp	49
netbios_ssn	16	courier	50
netstat	17	csnet_ns	51
Nnsp	18	ctf	52
nntp	19	day39	53

ntp_u	20	discard	54
other	21	domain	55
pm_dump	22	domain_u	56
pop_2	23	echo	57
pop_3	24	eco_i	58
printer	25	ecr_i	59
private	26	efs	60
red_i	27	exec	61
remote_job	28	finger	62
rje	29	ftp	63
Shell	30	ftp_data	64
Smtp	31	gopher	65
sql_net	32	host13S	66
Ssh	33	S12	67
Sunrpc	34		

Table 3.5:Numeric values assigned to different types of flags

Flag	**Numeric Values**
OTH	1
REJ	2
RSTO	3

RSTOS0	4
RSTR	5
S0	6
S1	7
S2	8
S3	9
SF	10
SH	11

The 42nd attribute contains data that are categorized as normal or one of the four attack types. The Table 5.6 shows this detail.

Table 3.6: Different types of attacks in NSL-KDD dataset

Attack Class	**Attack Type**
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm, Mailbomb
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_passwd, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, *Spy*, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps, Snmpguess, Worm

Normal is assigned a value of ‘1’ ; DoS, Probe, R2L, and U2R are replaced with ‘2’, ‘3’, ‘4’, and ‘5’ respectively. Since these outputs are of five types, they are further assigned values as shown in the Table 5.7.

Table 3.7 Five digits binary code assigned to normal and four attacks.

Attack Class	**Code Assignment**
Normal	00001
DoS	00010
Probe	00100
R2L	01000
U2R	10000

3.4 Determining Architecture of MLP

A fully connected neural network with m inputs, h hidden nodes, and n outputs has (m × h) + h + (h × n) + n weights and biases. For example, a neural network with 4 inputs, 5 hidden nodes, and 3 outputs has (4 * 5) + 5 + (5 * 3) + 3 = 43 weights and biases.

As there are 41 different attributes of each record, we have 41 neurons in the input layer. Similarly, we require 5 neurons in the output layer because there are 5 different categories of attacks including normal.

The number of hidden layers is nearly always one. There is a lot of empirical weight behind this presumption--in practice very few problems that cannot be solved with a single hidden layer become soluble by adding another hidden layer. Likewise, there is a consensus in the performance difference from adding additional hidden layers: the situations in which performance improves with a second (or third, etc.) hidden layer are very small. One hidden layer is sufficient for the large majority of problems. The number of hidden neurons is based on a complex relationship between

1. Number of input and output nodes
2. Amount of training data available
3. Complexity of the function that is trying to be learned
4. The training algorithm

To minimize the error and have a trained network that generalizes well, we need to pick an optimal number of hidden layers, as well as nodes in each hidden layer.

- Too few nodes will lead to high error for our system as the predictive factors might be too complex for a small number of nodes to capture
- Too many nodes will overfit to our training data and not generalize well

Overfiitting means that the error (number of incorrectly classified patterns) on the training set is driven to a very small value, however, when new data is presented, the error is large. In these cases, the ANN has memorized the training examples; however, it has not learnt to generalize the solution to new situations.

In most situations, there is no way to determine the best number of hidden units without training several networks and estimating the generalization error of each. If too few hidden units are chosen, there will be high training error and high generalization error due to underfitting and high statistical bias. If too many hidden units are chosen, there may be low training error but still have high generalization error due to overfitting and high variance.

So, performance (mean squared error, in this case) of multilayer perceptron with different number of hidden layers is first calculated and then the optimal value of hidden neurons is selected.

3.5 Tools Used

MATLAB 2013

MATLAB (Matrix Laboratory) is a programming environment for algorithm development, data analysis, visualization, and numerical computation. MATLAB can solve technical computing problems faster than with traditional programming languages such as C, C++, and FORTRAN. MATLAB can be used in a wide range of applications including signal and image processing, communications, control design,

test and measurement, financial modeling and analysis, and neural networks. For a million engineers and scientists in industry and academia, MATLAB is the language of technical computing (MathWorks Matlab Help, 2013).

Neural Network Toolbox supports supervised learning with feedforward, radial basis, and dynamic networks. It also supports unsupervised learning with self-organizing maps and competitive layers. With the toolbox, we can design, train, visualize, and simulate neural networks. Simulation is done using Neural Network toolbox in Matlab 2013.

4 RESULTS ANALYSIS

To access the results of the proposed intrusion detection approach, the simulation is performed in Matlab 2013. The totoal number of records used for system evaluation using NSL-KDD is 47735 out of which, 40% of total is used for training, 50% of total is used for testing, and remaining 10% is used for validation.

Following parameters are calculated after training and testing of MLP.

True Positive (TP): Situation in which a signature is fired properly when an attack is detected and an alarm is generated.

False Positive (FP): Situation in which normal traffic causes the signature to raise an alarm.

True Negative (TN): Situation in which normal traffic does not cause the signature to raise an alarm.

False Negative (FN): Situation in which a signature is not fired when an attack is detected.

Recall Rate: Recall rate measures the proportion of actual positives which are correctly identified.

$$\text{Recall Rate} = \frac{TP}{TP + FN}$$

Precision Rate: Precision rate is the ratio of true positives to combined true and false positives.

$$\text{Precision Rate} = \frac{TP}{TP + FP}$$

Table 4.1 Mean squared error values for different number of hidden neurons. For 30 hidden neurons, performance is best.

No. of Neurons in the Hidden Layer	**Performance (MSE)**
10	0.00175
16	0.000346
23	0.00101
30	0.000268

The final architecture of neural network will be as shown in the figure below.

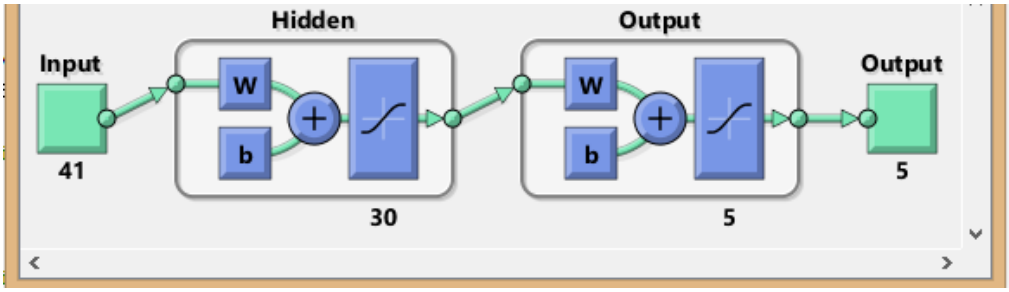

Figure 4.1 Architecture of MLP

The figure below shows the number of epochs required for training the neural network with Resilient Backpropagation algorithm, time required, gradient and performance.

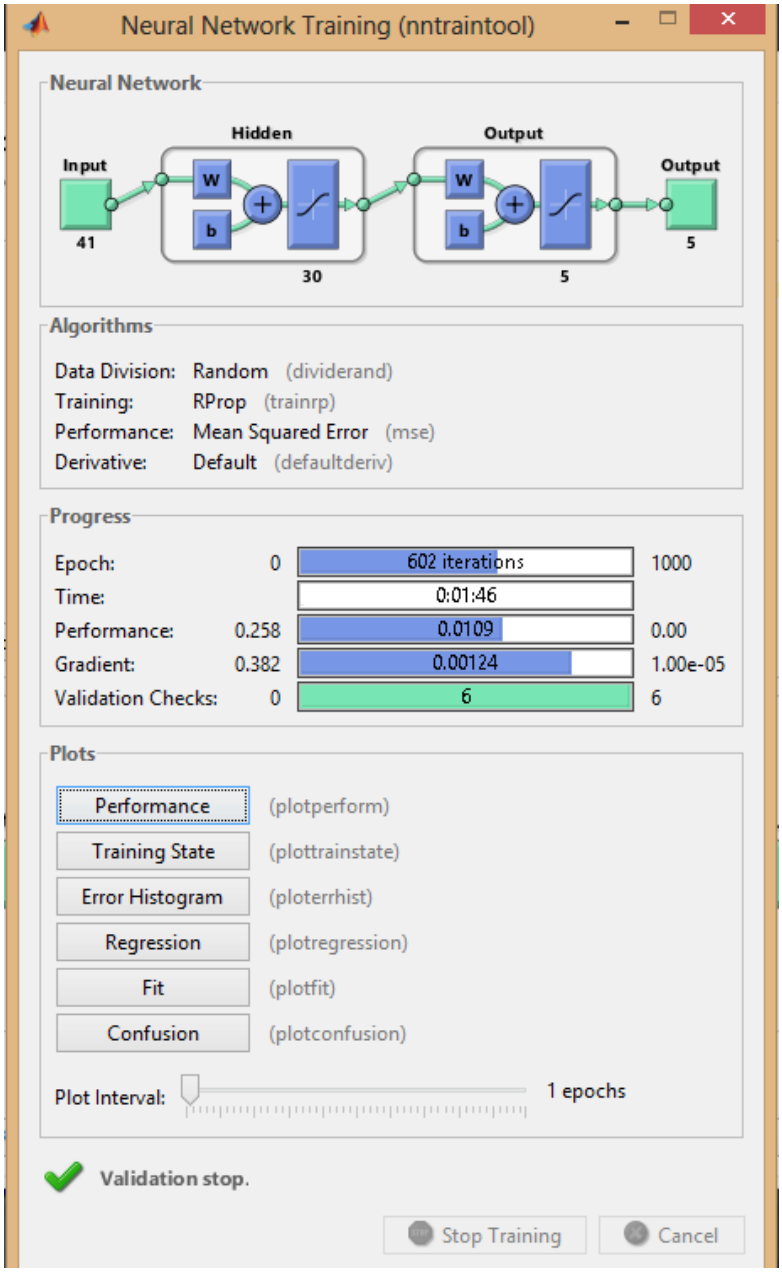

Figure 4.2: Summary of execution of Resilient Backpropagation

The training stopped when the validation error increased for six iterations, which occurred at iteration 602.

A plot of the training errors, validation errors, and test errors is shown in figure 4.3, the best validation is achieved at 596 epoch with validation value 0.011833. The result is reasonable because of the following considerations:

- The final mean squared error is small.
- The test set error and the validation set error have similar characteristics.

- No significant overfitting has occurred by iteration 596 (where the best validation performance occurs).

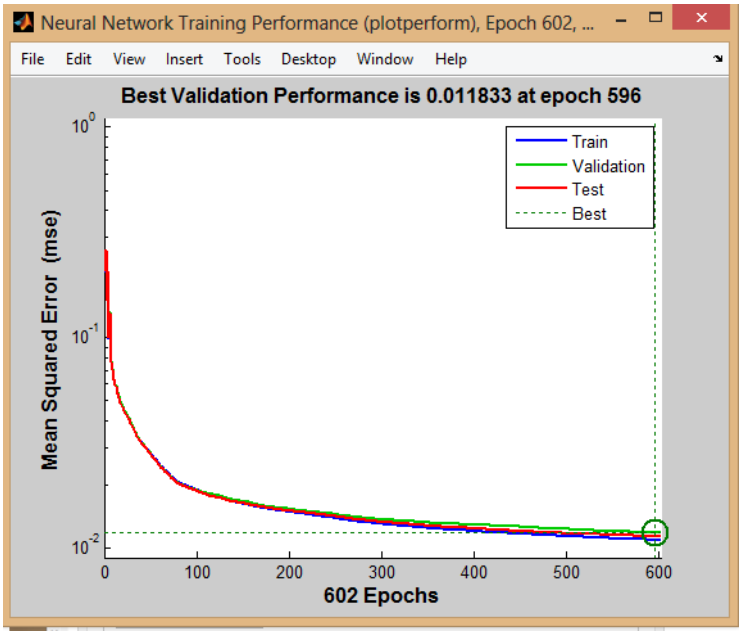

Figure 4.3: Mean square error versus no. of epochs

Similarly, training state of neural network the gradient valued decreases with increasing the number of epoch as shown in figure below.

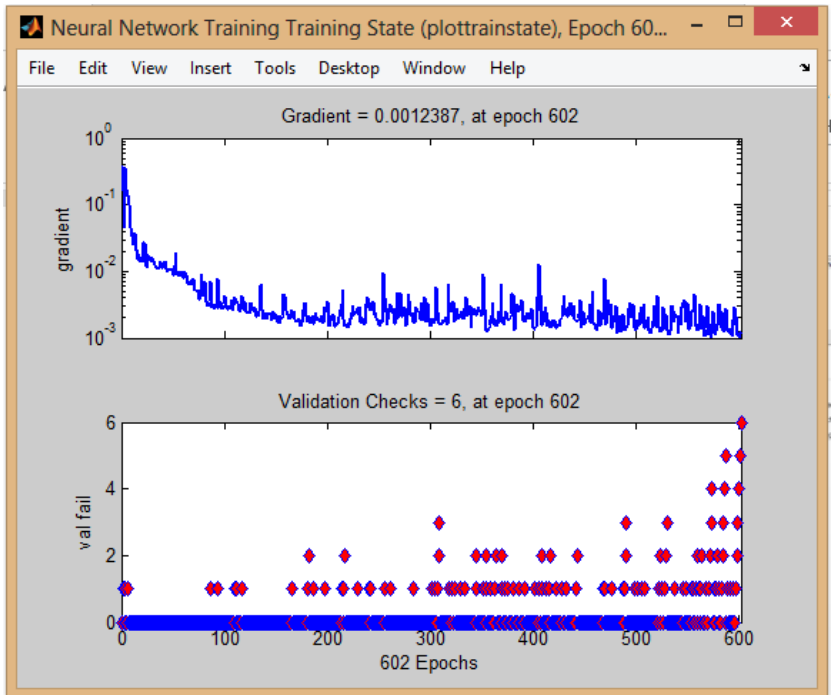

Figure 4.4 Graph demonstrating training states

Similarly, the figure below shows the error histogram in training, validation, and testing stages. In error histogram, 20 bins are taken in consideration. To "bin" the range of values means to divide the entire range of values into a series of intervals – and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but are not required to be) of equal size.

The blue bars represent training data, the green bars represent validation data, and the red bars represent testing data.

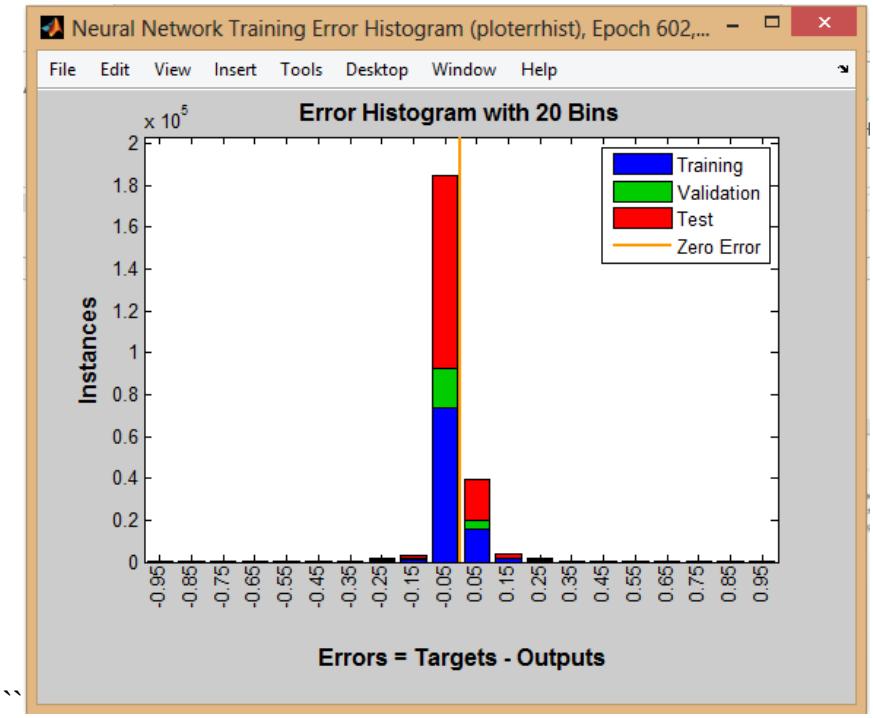

Figure 4.5 Histogram showing errors

The graph that follows demonstrates regression curve for training, validation, and testing for different values of R. In all cases, plot is between output and target. The regression plots display the network outputs with respect to targets for training, validation, and test set. Thus, regression is used to validate the network performance.

For a perfect fit, the data should fall along a 45° line, where the network outputs are equal to the targets. For this problem, the fit is reasonably good for all data sets, with R values in each case of 0.96 or above.

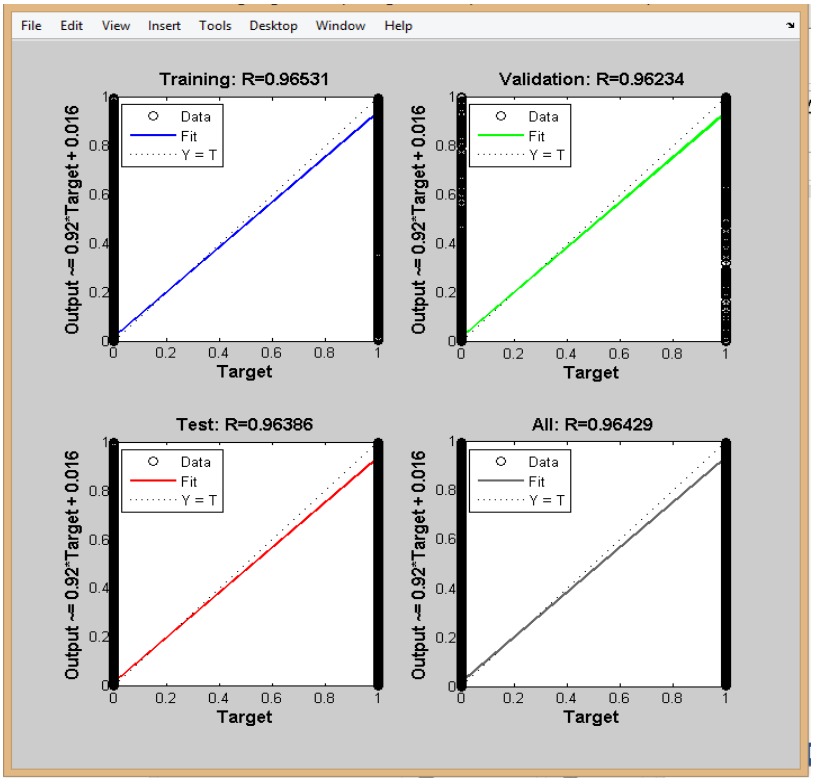

Figure 4.6 Regression

The predicted output by the neural network is shown below in the figure.

output2 <5x47735 double>

	1	2	3	4	5	6	7	8	9	
1	4.7914e-05	5.6355e-05	0.0013	0.0032	8.5405e-06	0.0016	4.3950e-04	2.0038e-04	0.0018	9.
2	1.0098e-05	3.1487e-05	0.1447	0.0055	0.0331	0.0030	8.1689e-04	1.0000	4.6533e-04	
3	0.0074	0.0143	8.0998e-04	0.9143	0.9883	2.7623e-09	6.4221e-04	1.6987e-05	1.7421e-09	3.
4	0.9999	0.9998	7.6016e-05	0.0026	2.9095e-04	2.1374e-04	4.4717e-04	0.0227	1.1126e-04	4.
5	4.8359e-09	6.2749e-09	0.9011	0.0102	2.5393e-06	0.9997	0.9932	1.2744e-05	1.0000	6.
6										
7										
8										
9										
10										
11										
12										
13										

Figure 4.7: Predicted output by neural network

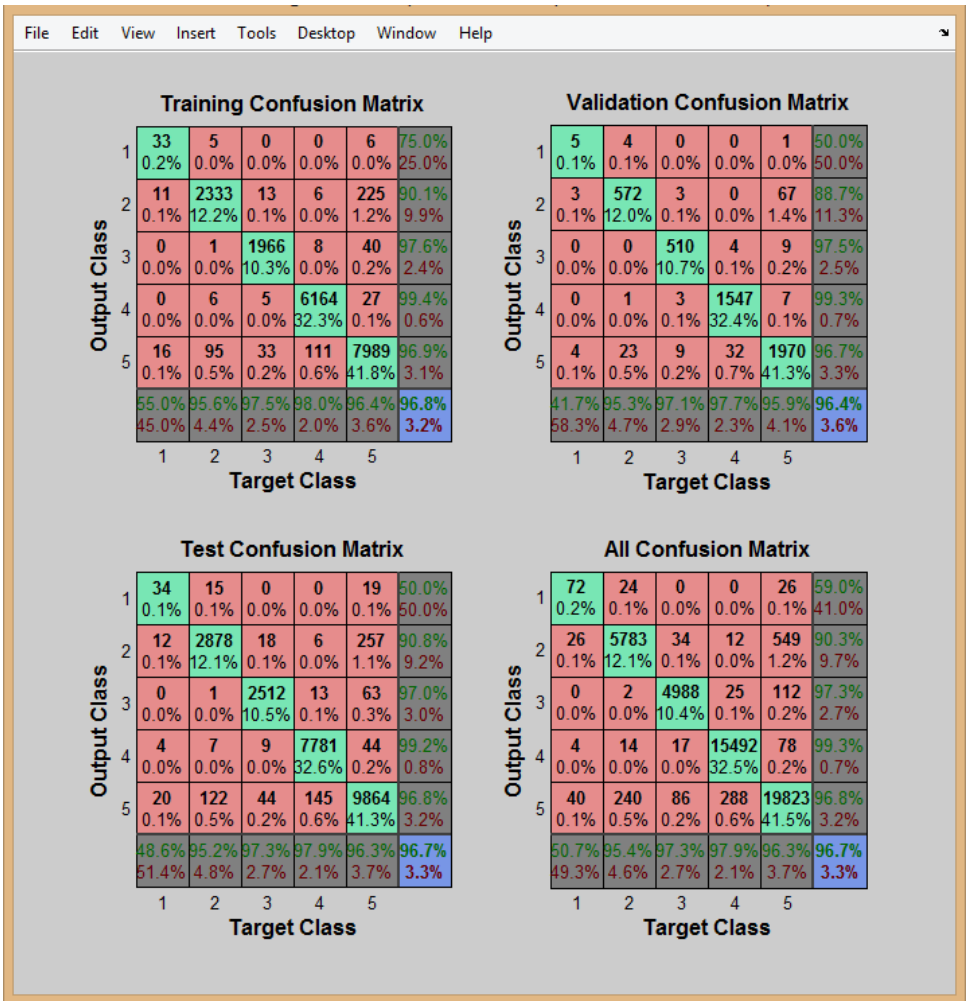

Figure 4.8 Confusion matrix.

From the confusion matrix, following parameters are calculated. Observing 'all confusion matrix', it is noted that the diagonal elements other than first and last diagonal elements represent TP of DoS, Probe, R2L, and U2R. To enlist the values of FP for different attacks, the values from first column (other than first and last elements of first column) are noted. Now, the values of first row beside first and last elements of first row are noted for FN for different attacks namely DoS, Probe, R2L, and U2R.

Table 4.2 Evaluation results of each attack classes

Attack	TP	FP	FN	Recall Rate	Precision Rate
DoS	5783	26	24	99.59%	99.55%
Probe	4988	0	0	100%	100%
R2L	15492	4	0	100%	99.97%
U2R	19823	40	26	99.87%	99.79%
Total	46086	70	50	99.86%	99.83%

The table below shows simulation results of NSL-KDD dataset.

Table 4.3 Simulation results.

SN	Performance (MSE)	Epoch	Recall Rate	Precision Rate
1	0.0109	602	99.86%	99.83%

The overall detection rate achieved in this research classifying the attack is found to be 96.7% using 30 number of hidden layers.

5 CONCLUSION

The NSL-KDD dataset (47735 records) was preprocessed and all attributes were converted into numerical format. Out of the total sample records taken, 40% of it was used to train neural network using Resilient Backpropagation algorithm, 50% was used for testing, and 10% was used for validation. The simulation results showed recall rate to be 99.8% and precision rate to be 99.83%. The overall detection rate in detecting intrusion was found to be 96.7%. The result was also nearly consistent when 200 number of datas from an organization was tested.

Network intrusion detection system model can be further enhanced to deal with online real-time traffic so that immediate classification of attack can be done and real-time preventive response can be carried out to prevent confidentiality, integrity, and availability of data.

REFERENCES

[1] XiaoHang Yao, "A Network Intrusion Detection Approach combined with Genetic Algorithm and Back Propagation Neural Network", IEEE-International Conference on E-Health Networking, Digital Ecosystems and Technologies, 2010.

[2] Farah Jemili, Montaceur Zaghdoud and Mohamad Ben Ahmed, "Intrusion Detection based on Hybrid Propagation in Bayesian Networks", IEEE ISE 2009, Richardson, TX, USA, June 8-11, 2009.

[3] Dr. Richard Spillman, "Artificial Intelligence". http://www.cs.plu.edu/courses/csce330/notes.html

[4] Anderson, James P., "Computer Security Threat Monitoring and Surveillance", Washing, PA, James P. Anderson Co., 1980.

[5] Reyadh Shaker Naoum, Namh Abdula Abid, Zainab Namh Al-Sultani, " An Enhanced Resilient Backpropagation Artificial Neural Network", IJCSNS International Journal of Computer Science and Network Security, VOL. 12 No. 3, March 2012.

[6] V. Jaiganesh, Dr. P. Sumathi and S. Mangayarkarasi, " An Analysis of Intrusion Detection System using Back Propagation Neural Network", Dr. N.G.P. Arts and Science College, Coimbatore, India.

[7] K. Labib, R. Vemuri, "NSOM: A Real-time Network-based Intrusion Detection System using Self-organizing Maps, Networks and Security", 2002.

[8] R. S. Puttini, Z. Marrakchi, L. Me, " A Bayesian Classification Model for Real-time Intrusion Detection, vol. 659, 2003.

[9] M. Amini, A. Jalili, H. Reza Shahriari, "RT-UNNID: A Practical Solution to Real-time Network-based Intrusion Detection using Unsupervised Neural Networks, Computer and Security", 2005.

[10] M-Y. Su, G-J. Yu, C-Y. Lin, "A Real-time Network Intrusion Detection System for Large-scale Attacks based on an Incremental Mining Approach, 2009.

[11] S. Chakrabarti, M. Chakraborty, I. Mukhopadhyay, “Study of Snort-based IDS”, IEEE International Advance Computing Conference, 2010.

[12] S. Sahu, S. Sarangi, S. Kumar Jena, “A Detail Analysis on Intrusion Detection Datasets”, 2014.

[13] L. Dhanabal, Dr. S. P. Shantharajah, “A Study on NSL-KDD Dataset for Intrusion Detection System based on Classification Algorithms”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 6, June 2015.

[14] NSL-KDD dataset (2009) available on:

http://www.unb.ca/cic/research/datasets/nsl.html

[15] Hee-Su Chae, Byung –Oh Jo, Sang Hyun Choi, Twae-Kung Park; "Feature Selection for Intrusion Detection using NSL-KDD " Department of Information Security Management, Department of Management Information System, Chugbuk National University in Korea, 2014.

[16] Bhupendra Ingre, Anamika Yadav ; "Performance Analysis of NSL-KDD Dataset using ANN" Department of Electrical Engineering,National Institute of Technology, Raipur; 2015.

[17] Bhupendra Ingre, Anamika Yadav, and Atul Kumar Soni; "Detection Tree based Intrusion Detection System for NSL-KDD Dataset" Department of Electrical Engineering, National Institute of Technology, Raipur; March, 2017.