**CHAPTER 1**

**INTRODUCTION**

In this chapter, the purpose of the project, outcomes and organization of report are discussed which gives an overview of the project.

* 1. **PURPOSE OF THE PROJECT**

One of the major challenges in network security is the provision of a robust and effective Network Intrusion Detection System (NIDS). Despite the significant advances in NIDS technology, the majority of solutions still operate using less-capable signature-based techniques, as opposed to anomaly detection techniques. There are several reasons for this reluctance to switch, including the high false error rate (and associated costs), difficulty in obtaining reliable training data, longevity of training data and behavioral dynamics of the system. The current situation will reach a point whereby reliance on such techniques leads to ineffective and inaccurate detection. The specifics of this challenge are to create a widely-accepted anomaly detection technique capable of overcoming limitations induced by the ongoing changes occurring in modern networks.

* 1. **OBJECTIVE**

The main objective of our proposed work is to device a better model using deep learning techniques in network intrusion detection and to predict anomaly in network and identify type of network attack.

* 1. **Function of Intrusion Detection**

Intrusion Detection function includes,

* Monitoring and analyzing both user and system activities,
* Analyzing system configuration and vulnerabilities assessing system and file integrity,
* Ability to recognize patterns typical of attacks,
* Analysis of abnormal patterns,
* Tracking user policy violations.
  1. **INTRUSION**

Intrusion is a set of actions aimed to compromise the security of computer and network components in terms of confidentiality, integrity and availability.

* 1. **APPLICATIONS OF ANOMALY DETECTION**

Data mining application includes,

* Fraud detection for credit cards
* Intrusion detection for cyber security
* Military surveillance for enemy activities
  1. **CLASSES OF COMPUTER ATTACKS**

**Virus -** It is a self-replicating program that infects the system without any knowledge or permission from user

**Worm -** A self-replicating program that propagates through network services on computer systems without user intervention. It can highly harm network by consuming network bandwidth

**Trojan:-** A malicious program that cannot replicate itself but can cause serious security problems in the computer system.

**Network Attack:-**It does not allow the authorized users for accessing network services and resources.

**Denial of Service (DOS):-**Attacker tries to prevent legitimate users from using a service.

**Remote to Local (R2L):-** Attacker does not have an account on the victim machine, hence tries to gain access.

**User to Root (U2R):-** Attacker has local access to the victim machine and tries to gain super user privileges.

**Probe:-**Attacker tries to gain information about the target host.

**Password Attack:-** Aims to gain a password within a short period of time, and is usually indicated by a series of login failures.

**Information Gathering Attack:-** Gathers information or finds known vulnerabilities by scanning or probing computers or networks.

**Physical Attack:-** An attempt to damage the physical components of networks or computers.

* 1. **TYPES OF IDS**

Intrusion Detection System is classified into two categories,

* Based on Deployment in real time
  + Host based IDS
  + Network based IDS
* Based on Detection Mechanism
  + Misuse based
  + Anomaly based
  + Hybrid
    1. **BASED ON DEPLOYMENT IN REAL TIME**

**Host-Based IDS (HIDS)**

A HIDS monitors and analyzes the internals of a computing system rather than its external interfaces. A HIDS might detect internal activity such as which program accesses what resources and attempts illegitimate access. An example is a word processor that suddenly and inexplicably starts modifying the system password database. Similarly, a HIDS might look at the state of system and its stored information whether it is in RAM or in the file system or in log files or elsewhere. One can think of HIDS as an agent that monitors whether anything or anyone internal or external has circumvented the security policy that the operating system tries to enforce.

**Network Based IDS (NIDS)**

A NIDS deals with detecting intrusions in network data. Intrusions typically occur as anomalous patterns though certain techniques model the data in sequential fashion and detect anomalous subsequences. The Primary reason for these anomalies is attacks launched by outside attackers who want to gain unauthorized access to the network to seal information or to disrupt the network. In typical setting, a network is connected to rest of the world through the Internet. The NIDS reads all incoming packets or flows, trying to find suspicious patterns. NIDS Dynamically monitors logs of network traffic in real time to identify the potential intrusions in a network using intrusion detection algorithms.

* + 1. **BASED ON DETECTION MECHANISM**

**Misuse based Detection**

Misuse-based intrusion detection normally searches forknown intrusive patterns. Detection is based on a set of rules or signatures for known attacks.It can detect all known attack patterns based on the reference data. How to write a signature that encompasses all possible variations of the pertinent attack is a challenging task.

**Anomaly based Detection**

Anomaly-based intrusion detection tries to identify unusual patterns.

**Principal Assumption:**

All intrusive activities are necessarily anomalous. Such a method builds a normal activity profile and checks whether the system state varies from the established profile by a statistically significant amount to report intrusion attempts.

Anomalous activities that are not intrusive may be flagged as intrusive. These are false positives. One should select threshold levels so that neither of the above two problems is unreasonably magnified nor the selection of features to monitor is optimized. Computationally expensive because of overhead and possibly updating several system profile matrices.

**Hybrid Detection**

It exploits benefits of both misuse and anomaly-based detection techniques. It Attempts to detect known as well as unknown attacks.

Today, researchers mostly concentrate on **Anomaly based network intrusion detection** because it can detect known as well as unknown attacks. All kinds of IDS use the **Data mining (DM)** techniques for detecting intrusions.

* 1. **OUTCOMES**

The outcomes of the proposed work are,

* Accuracy of Attack classification is improved compared to the existing system.
* Real time Detection of Network Anomaly.
  1. **ORGANIZATION OF THE PROJECT**

**Chapter: 2** describes about the various existing methodologies used for detecting the anomaly network and to classify the type of attack in the network.

**Chapter: 3** discusses about the system study.

**Chapter: 4** outlines the design and explain the methodology of the proposed system.

**Chapter: 5** describes about the implementation methodology.

**Chapter: 6** discusses about the coding and results.

**Chapter: 7** discusses about the conclusion and future enhancements.

**SUMMARY**

This chapter had given the brief description about the purpose of the project, objectives, outcomes and organization of the report.

**CHAPTER 2**

**LITERATURE SURVEY**

* 1. **OVERVIEW**

This section discusses the methodologies learnt from previous works for our proposed system.

* 1. **REDUCING THE DIMENSIONALITY OF DATA WITH NEURAL NETWORKS**

(G. E. Hinton and R. R. Salakhutdinov ,”Reducing the dimensionality of data with neural networks”, Vol 313, Issue 5786, pp. 504-507, Science Journal- 28 July 2006).

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such ‘‘auto encoder’’ networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep auto encoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

**MERITS:**

* It helps in data compression, and hence reduced storage space.
* It reduces computation time.
* It also helps remove redundant features, if any.

**DEMERITS:**

* It may lead to some amount of data loss.
* PCA tends to find linear correlations between variables, which is sometimes undesirable.
* PCA fails in cases where mean and covariance are not enough to define datasets.
* We may not know how many principal components to keep- in practice, some thumb rules are applied
  1. **LONG SHORT TERM MEMORY RECURRENT NEURAL NETWORK CLASSIFIER TO INTRUSION DETECTION**

([Jihyun Kim](https://ieeexplore.ieee.org/author/37085771478); [Jaehyun Kim](https://ieeexplore.ieee.org/author/37085771469); Huong Le Thi Thu ; [HowonKim](https://ieeexplore.ieee.org/author/37085770474)**,”**Long short term memory recurrent neural network classifier to intrusion detection**”**[2016 International Conference on Platform Technology and Service](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7445576),15-17 Feb. 2016)

Recurrent Neural Network is the most famous model to training the sequence data. The conventional RNN has trouble when it is used to train with a long step size. Recurrent Neural Network (RNN) is extension of a convention feed-forward neural network. Unlike feed forward neural networks, RNN have cyclic connections making them powerful for modelling sequences

**MERITS:**

* LSTM is great tool for anything that has a sequence. Since the meaning of a word depends on the ones that preceded it. This paved the way for NLP and narrative analysis to leverage Neural Networks.
* Sequence-to-Sequence LSTM models are the state of the technique for translations. They also have a wide array of applications like time series forecasting.

**DEMERITS:**

* Potentially time consuming and technically complex.
  1. **TOWARD AN ONLINE ANOMALY INTRUSION DETECTION SYSTEM BASED ON DEEP LEARNING**

([Khaled Alrawashdeh](https://ieeexplore.ieee.org/author/37086173198); [Carla Purdy](https://ieeexplore.ieee.org/author/37282122500),”Toward an online anomaly intrusion detection system based on deep learning”,[2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7835817),18-20 Dec. 2016)

A deep learning approach for anomaly detection using a Restricted Boltzmann Machine(RBM) and a deep belief network are implemented. This method uses a one-hidden layer RBM to perform unsupervised feature reduction. The resultant weights from this RBM are passed to another RBM producing a deep belief network.

**MERITS:**

* Low alarm rates: All it has to do is to look up the list of known signatures of attacks and if it finds a match report it.
* Signature based NID are very accurate.
* Speed: The systems are fast since they are only doing a comparison between what they are seeing and a predetermined rule.

**DEMERITS:**

* If someone develops a new attack, there will be no protection.
* “only as strong as its rule set.”
* Attacks can be masked by splitting up the messages.
  1. **A NEW INTRUSION DETECTION SYSTEM BASED ON FAST LEARNING NETWORK AND PARTICLE SWARM OPTIMIZATION**

(Mohammed Hasan Ali, Bahaa Abbas Dawood Al Mohammed, AlyaniIsmail, And Mohamad FadliZolkipli, ”A new intrusion detection system based on fast learning network and particle swarm optimization”, [IEEE Access](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639) ( Volume: 6 ), 27 March 2018)

Supervised intrusion detection system is a system that has the capability of learning from examples about the previous attacks to detect new attacks. Using artificial neural network (ANN)-based intrusion detection is promising for reducing the number of false negative or false positives, because ANN has the capability of learning from actual examples. In this paper, a developed learning model for fast learning network (FLN) based on particle swarm optimization (PSO) has been proposed and named as PSO-FLN. The model has been applied to the problem of intrusion detection and validated based on the famous dataset KDD99.

PSO-based optimization of FLN is based on designing a particle that represents one candidate solution of FLN weights. One specific problem in performing the optimization is requiring to select both the weight’s values as well as the number of neurons that are needed in the hidden layer of accomplish better accuracy. This means a variable length through the solution according to the number of the hidden neurons in FLN, and to overcome this problem, the maximum number of neurons in considered in assigning a length for the particle. For activation function, tanging has been used for the output of the hidden layer neurons.

**MERITS:**

* It is more efficient where traffic is more because it can run parallel computation.

**DEMERITS:**

* It is more efficient only when the hidden neurons is increased therefore increase in complexity
  1. **NETWORK ANOMALY DETECTION: METHODS,SYSTEMS AND TOOLS**

(Monowar H. Bhuyan, Bhattacharyya D.K., KalitaJ.K., ”Network anomaly detection: Methods, Systems and Tools [IEEE Communications Surveys & Tutorials](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=9739) ( Volume: 16 , [Issue: 1](https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=6734841) , First Quarter 2014 ),06 June 2013).

Bhuyan described the existing Network anomaly detection methods, systems and tools. Network anomaly detection is an important and dynamic research area. Many network intrusion detection methods and systems (NIDS) have been proposed in the literature. They have provided a structured and comprehensive overview of various facets of network anomaly detection so that a researcher can become quickly familiar with every aspect of network anomaly detection.

They have categorized the existing network anomaly detection methods and systems based on the underlying computational techniques used. They have briefly described and compared a large number of network anomaly detection methods and systems. In addition, they have discussed about tools that can be used by network defenders and datasets that researchers in network anomaly detection can use. It also highlights research directions in network anomaly detection.

Methods and systems for network anomaly detection includes,

* Statistical methods and systems
* Classification-based methods and systems
* Clustering and Outlier-based methods and systems
* Soft computing methods and systems
* Knowledge-based methods and systems
* Combination learner methods and systems

**Relevant feature identification:**

Feature selection plays an important role in detecting network anomalies. Feature selection methods are used in the intrusion detection domain for eliminating unimportant or irrelevant features. Feature selection reduces computational complexity, removes information redundancy, increases the accuracy of the detection algorithm, facilitates data understanding and improves generalization.

**The feature selection process includes three major steps:**

1. Subset generation
2. Subset evaluation
3. Validation.

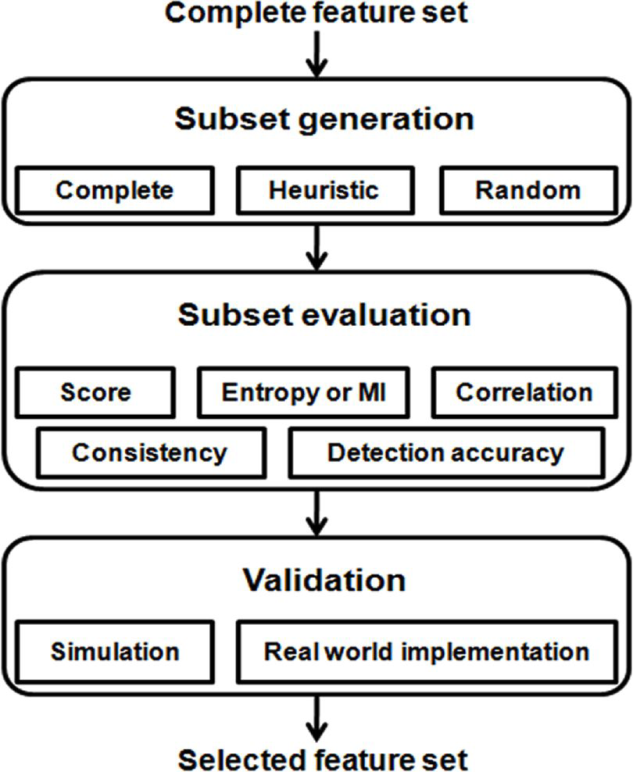
**Three different approaches for subset generation are:**

1. Complete
2. Heuristic
3. Random.

**Evaluation functions are categorized into five distinct categories:**

1. Score-based,
2. Entropy or mutual information-based,
3. Correlation-based,
4. Consistency-based,
5. Detection accuracy-based.

Simulation and real world implementation are the two ways to validate the evaluated subset. A conceptual framework of the feature selection process is shown in Figure. 2.1



**Figure. 2.1** Framework of feature selection process

* 1. **FLAG, ATTRIBUTE DESCRIPTION OF KDD CUP 99 DATASET**

This paper showed the flag description and attribute description of KDD CUP 99 dataset. This paper discussed about description of flag in KDD CUP 99. Figure. 2.2 shows the schematic diagram depicting the observation of the traffics (packet Pi) over a time window to construct the 41 features. Table 2.1 shows the flag description and Table 2.3 shows the attribute description of KDD CUP 99 dataset.

---------------------

P2

P1

Pn

41 features collected from the packet data observed over the time window of 2 seconds

**Figure. 2.2 Extraction of features from the network connections for detecting Intrusion**

**Some of the terminologies associated with the data set are:**

* The term **‘same host’** refers to the connections in the past two seconds that have the same destination host as the current connection, `and is attached to the features like protocol activities, service etc.
* The term **‘same service’** refers to the connections in the past two seconds that have the same service as the current connection.
* The features based on **‘same host’** and **‘same service’** is collectively called as time-based traffic features of the connection records.

**Table 2.1** Flag description of KDD CUP 99 dataset

|  |  |  |
| --- | --- | --- |
| **Attribute No** | **Flag Name** | **Description** |
| 1 | RSTOS0 | Originator sent a SYN followed by a RST, never see a SYNACK from the responder |
| 2 | RSTR | Established, responder aborted |
| 3 | RSTO | Connection established, originator aborted (sent a RST) |
| 4 | OTH | No SYN seen, just midstream traffic (a “partial connection” that was not later closed) |
| 5 | REJ | Connection attempt rejected |
| 6 | S0 | Connection attempt seen, no reply |
| 7 | S1 | Connection established, not terminated |
| 8 | S2 | Connection established and close attempt by originator seen (but no reply from responder) |
| 9 | S3 | Connection established and close attempt by responder seen (but no reply from originator) |
| 10 | SF | Normal establishment and termination |
| 11 | SH | Originator sent a SYN followed by a FIN (finish ‘flag’),never saw a SYN ACK from the responder (hence the connection was “half” open) |

**Table 2.2** Attribute description of KDD CUP 99 dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute No** | **Attribute Name** | **Description** | **Data Type** |
| 1 | Duration | Length of time durationof the connection | Numeric |
| 2 | Protocol\_type | Protocol used in the connection(TCP,UDP, ICMP) | Nominal |
| 3 | Service | Destination network service used(HTTP, FTP, Telnet etc) | Nominal |
| 4 | Flag | Status of the connection: Normal or Error | Nominal |
| 5 | Src\_bytes | Number of data bytes Transferred from source to destination in single connection | Numeric |
| 6 | Dst\_bytes | Number of data bytes Transferred from destination to source in single connection | Numeric |
| 7 | Land | if source and destination IP addresses and port numbers are equal then, this variable takes value 1  else 0 | Binary |
| 8 | Wrong\_fragment | Total number of wrong fragments in  this connection | Numeric |
| 9 | Urgent | Number of urgent packets in this connection.Urgent packets are packets with the urgent bit activated | Numeric |
| 10 | Hot | Number of “hot‟ indicators in the content such as: entering a systemdirectory,creating programs and executing programs | Numeric |
| 11 | Num\_failed\_logins | Count of failed login attempts | Numeric |
| 12 | Logged\_in | Login Status :  1 if successfully logged in;  0 otherwise | Binary |
| 13 | Num\_compromised | Number of “compromised'' conditions | Numeric |
| 14 | Root\_shell | 1 if root shell is obtained; 0  Otherwise | Numeric |
| 15 | Su\_attempted | 1 if ``su root'' command  attempted or used; 0 otherwise | Numeric |
| 16 | Num\_root | Number of “root'' accesses  or number of operations  performed as a root in the  connection | Binary |
| 18 | Num\_shells | Number of shell prompts | Numeric |
|  |  |  |  |
| 19 | Num\_access\_files | Number of operations on access control files | Numeric |
| 20 | Num\_outbound\_  cmds | Number of outbound commands in an ftp session | Numeric |
| 21 | Is\_hot\_login | 1 if the login belongs to the  ``hot'' list i.e.,root or admin;  else 0 | Binary |
| 22 | Is\_guest\_login | if the login is a ``guest'‘ lo1 if the login is a ``guest'‘ login; 0 otherwise | Binary |
| 23 | Count | Number of connections to the same destination host as the current connection in the past two seconds | Numeric |
| 24 | Srv\_count | Number of connections to the same service (port number) as the current connection in the past two seconds | Numeric |
| 25 | Serror\_rate | The percentage of connections  that have activated the flag (4) s0, s1,s2 or s3,among the Connections aggregated in count (23) | Numeric |
| 26 | Srv\_serror\_rate | The percentage of connections that have activated the flag (4) s0, s1,s2 or s3, among the connections aggregated in srv\_count (24) | Numeric |
| 27 | Rerror\_rate | The percentage of connections that have activated the flag (4) REJ,among the connections aggregated in count (23) | Numeric |
| 28 | Srv\_rerror\_rate | The percentage of connections that have activated the flag (4) REJ,among the connections aggregated in srv\_count (24) | Numeric |
| 29 | Same\_srv\_rate | The percentage of connections that were to the same service, among the connections aggregated in count (23) | Numeric |
| 30 | Diff\_srv\_rate | The percentage of connections that were to different services, among the connections aggregated in count (23) | Numeric |
| 31 | Srv\_diff\_host\_Rate | The percentage of connections that were to Different destination  Machines among the Connections aggregated in srv\_count (24) | Numeric |
| 32 | Dst\_host\_count | Number of connections having the same destination host IP address | Numeric |
| 33 | Dst\_host\_srv\_count | Number of connections having the same port number | Numeric |
| 34 | Dst\_host\_same\_srv\_rate | The percentage of connections that were to the same service, among the connections aggregated in dst\_host\_count(32) | Numeric |
| 35 | Dst\_host\_diff\_srv\_rate | The percentage of connections that were to Different services, among the connections aggregated in dst\_host\_count(32) | Numeric |
| 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) | Numeric |
| 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) | Numeric |
| 38 | Dst\_host\_serror\_  rate | The percentage of connections that have activated the flag (4) s0, s1,s2 or s3, among the connections aggregated in dst\_host\_count(32) | Numeric |
| 39 | Dst\_host\_srv\_  serror\_rate | The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst\_host\_srv\_ count(33) | Numeric |
| 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) | Numeric |
| 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) | Numeric |

**SUMMARY**

This chapter had given the description about the literature survey for various techniques involved in Network intrusion detection system.

**CHAPTER 3**

**SYSTEM STUDY**

* 1. **OVERVIEW**

In this chapter, overview of existing system and proposed system for Network Intrusion Detection System, various steps used for detecting and classifying the anomaly network are discussed.

* 1. **PROPOSED SYSTEM**

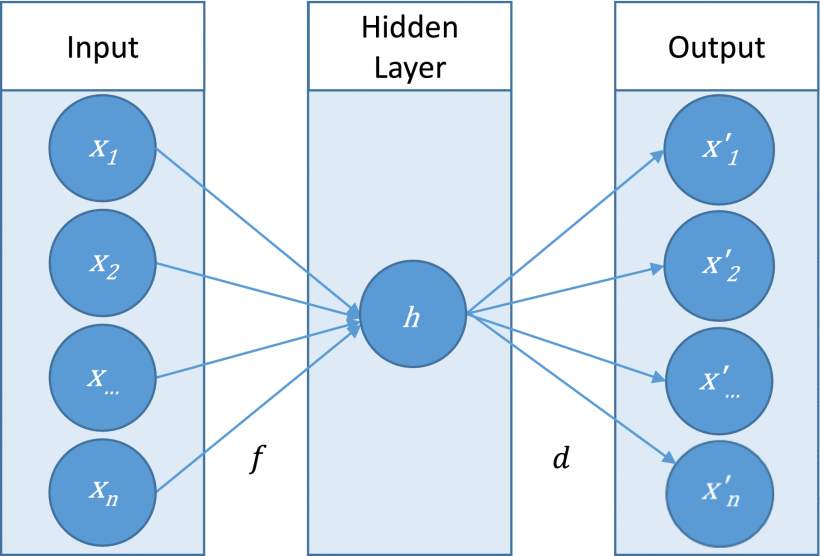
In this system, we are indented to reduce the number of features and for that purpose we are going to use the Deep Learning Techniques Non Symmetric Deep Auto encoders and Stacked Non Symmetric Deep Auto encoders.

* + 1. **Deep Learning**

Deep learning is an advanced sub-field of machine learning, which advances Machine Learning closer to Artificial Intelligence. It facilitates the modeling of complex relationships and concepts using multiple levels of representation. Supervised and unsupervised learning algorithms are used to construct successively higher levels of abstraction, defined using the output features from lower levels.

#### **Auto-Encoder**

A popular technique currently utilized within deep learning research is auto-encoders, which is utilized by our proposed solution. An auto-encoder is an unsupervised neural network-based feature extraction algorithm, which learns the best parameters required to reconstruct its output as close to its input as possible. One of it desirable characteristics is the capability to provide more a powerful and non-linear generalization than Principle Component Analysis (PCA). This is achieved by applying back propagation and setting the target values to be equal to the inputs. In other words, it is trying to learn an approximation to the identity function. An auto-encoder typically has an input layer, output layer (with the same dimension as the input layer) and a hidden layer. This hidden layer normally has a smaller dimension than that of the input (known as an under complete or sparse auto-encoder). An example of an auto-encoder is shown in Figure. 3.1.

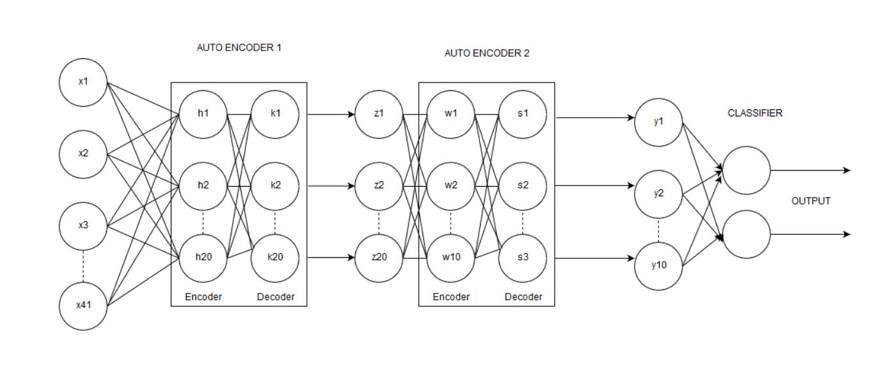


**Figure 3.1** Auto encoder

* + 1. **Stacked Auto-encoders**

Unlike a simple auto-encoder, a deep auto-encoder is composed of two symmetrical deep-belief networks, which typically have four or five shallow layers for encoding, and a second set of four or five layers for decoding. The work by Hinton and Salacukhudinov has produced promising results by implementing a deep learning algorithm to convert high dimensional data to low dimensional data by utilizing a deep auto-encoder.

Deep learning can be applied to auto-encoders, whereby the hidden layers are the simple concepts and multiple hidden layers are used to provide depth, in a technique known as a stacked auto-encoder. This increased depth can reduce computational costs and the amount of required training data, as well as yielding greater degrees of accuracy. The output from each hidden layer is used as the input for a progressively higher level. Hence, the first layer of a stacked auto-encoder usually learns first-order features in raw input. The second layer usually learns second-order features relating to patterns in the appearance of the first-order features. Subsequent higher layers learn higher-order features. An illustrative example of a stacked auto-encoder is shown in Figure. 3.2. Here, the superscript numbers refer to the hidden layer identity and the subscript numbers signify the dimension for that layer.

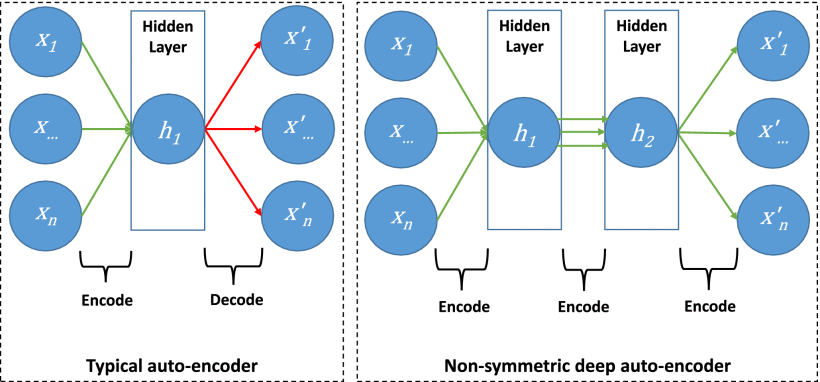


**Figure 3.2** Stacked Auto-encoders

* + 1. **Non Symmetric Deep Auto-encoders**

Decreasing the reliance on human operators is a crucial requirement for future-proofing NIDSs. Hence, our aim is to devise a technique capable of providing reliable unsupervised feature learning, which can improve upon the performance and accuracy of existing techniques.

NDAE, which is an auto-encoder featuring non-symmetrical multiple hidden layers. Fundamentally, this involves the proposed shift from the encoder-decoder paradigm (symmetric) and towards utilizing just the encoder phase (non-symmetric). The reasoning behind this is that given the correct learning structure, it is be possible to reduce both computational and time overheads, with minimal impact on accuracy and efficiency. NDAE can be used as a hierarchical unsupervised feature extractor that scales well to accommodate high-dimensional inputs. It learns non-trivial features using a similar training strategy to that of a typical auto-encoder. An illustrated example of this is presented in Figure.3.3



**Figure 3.3** Non Symmetric Deep Auto encoders

The hidden layer is used to create a lower dimensionality version of high dimensionality data (known as encoding). By reducing the dimensionality, the auto-encoder is forced to capture the most prominent features of the data distribution. In an ideal scenario, the data features generated by the auto-encoder will provide a better representation of the data points than the raw data itself.

The aim of the auto-encoder is to try and learn the function shown equation (1).

(1)

Here, h is a non-linear hypothesis using the parameters W (weighting) and b (bias), which can fit the given data (x).

Simply, it tries to learn an approximation to the identity of a function, where x′ is most similar to x. The learning process is described as a reconstruction error minimization function, as equation (2).

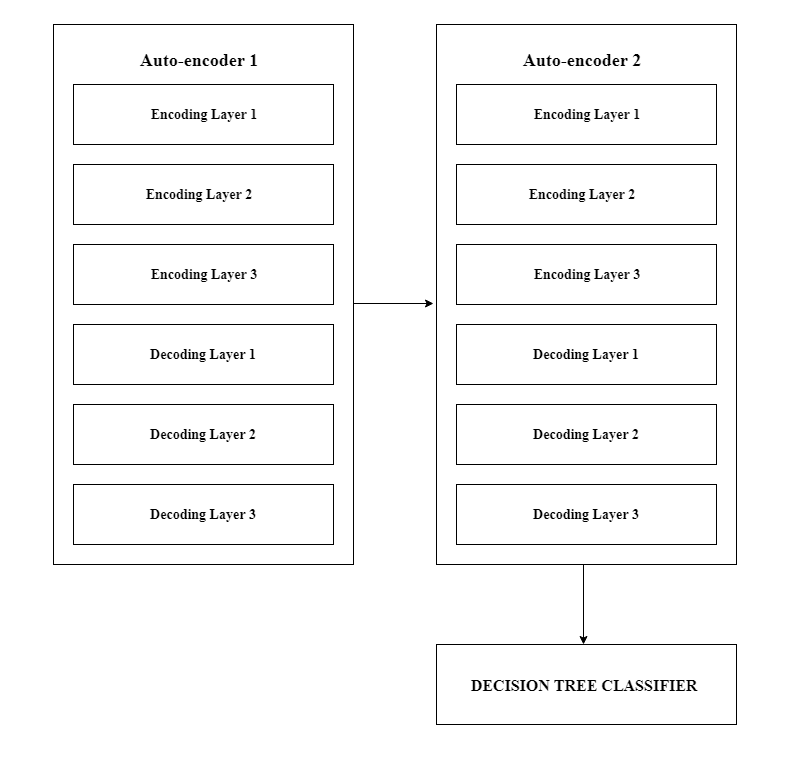
(2)

Here, L is a loss function penalizing d(f(x)) for being dissimilar to x, d is a decoding function and f is an encoding function.

* + 1. **Stacked Non-Symmetric Deep Auto-encoders**

Fundamentally, our model is based upon using our NDAE technique for deep learning. This is achieved by stacking our NDAEs to create a deep learning hierarchy. Stacking the NDAEs offers a layer-wise unsupervised representation learning algorithm, which will allow our model to learn the complex relationships between different features. It also has feature extraction capabilities, so it is able to refine the model by prioritizing the most descriptive features.

Due to the data that we envisage this model using, we have designed the model to handle large and complex datasets. Despite the 42 features present in the KDD Cup ’99 and NSL-KDD datasets being comparatively small, we maintain that it provides a benchmark indication as to the model's capability. However, the classification power of stacked auto-encoders with a typical soft-max layer is relatively weak compared to other discriminative models including Decision Tree , KNN and SVM. Hence, we have combined the deep learning power of our stacked NDAEs with a shallow learning classifier. For our shallow earning classifier, we have decided upon using Random Forest.



**Figure 3.4** Stacked NDAE Classification Model.

**SUMMARY**

This chapter had described about the overview, comparison between the existing system and proposed system for Network Intrusion Detection System and methods used in proposed system.

**CHAPTER 4**

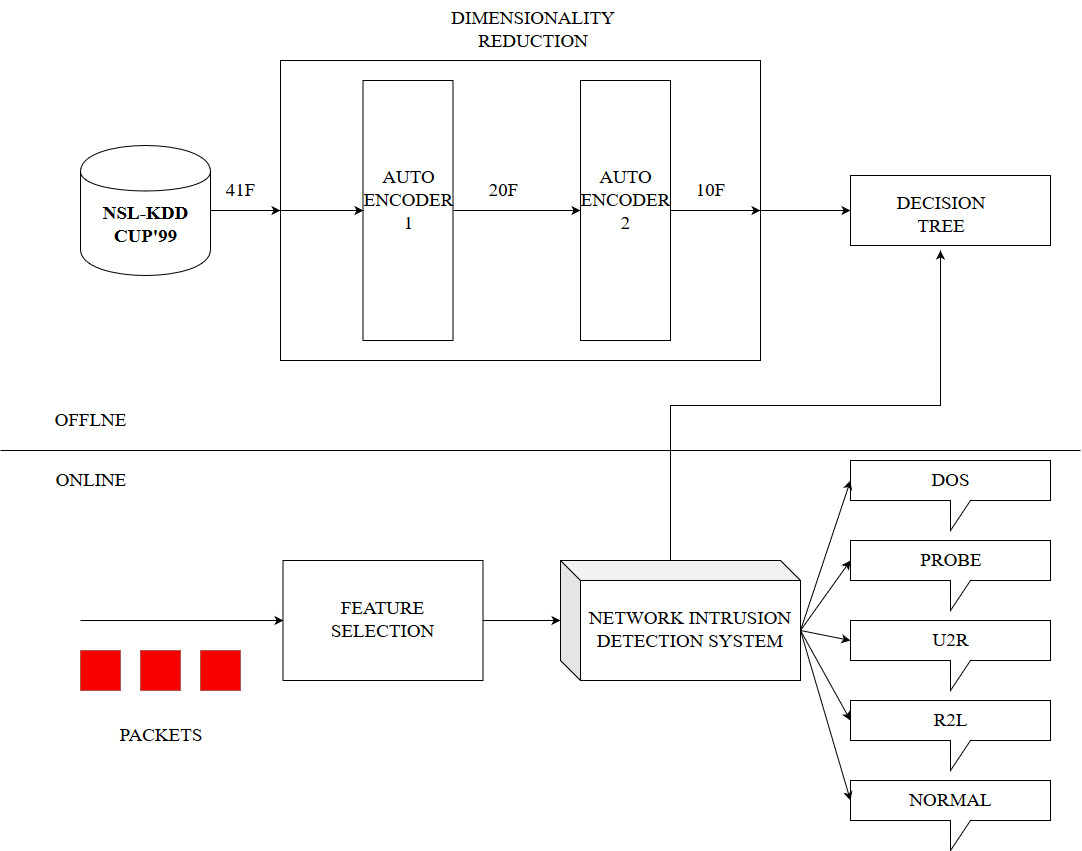
**SYSTEM DESIGN**

* 1. **OVERVIEW**

In this chapter system architectural design as well as the modules involved in the proposed system are discussed.

* 1. **SYSTEM ARCHITECTURAL DESIGN**

The architectural design for the Network Intrusion Detection System using Auto-encoders is shown in Figure 4.1

****

**Figure 4.1** Architectural design for the NIDS

In the above architectural design (**Figure 4.1**) KDD cup data from NSL-KDD dataset is used. The dataset contains network packets information like protocol, source, destination, payload etc up to 41 features and corresponding labels. The data is first preprocessed and then feed into the Stacked Non-Symmetric Deep Auto-encoders for Dimensionality Reduction and feature Extraction. Then Decision Tree is trained and tested using the modified dataset from our auto-encoder and results are evaluated.

* 1. **MODULES DESCRIPTION**

The various modules involved in the work are

* Data pre-processing
* Dimensionality Reduction
* Decision Tree Classifier
  + 1. **DATA-PREPROCESSING**

The Data set that we use in our proposed system is NSL-KDD data set which is a benchmark data set. The data should be pre processed because there is different variety of data in our data set.

**Label Encoding:**

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

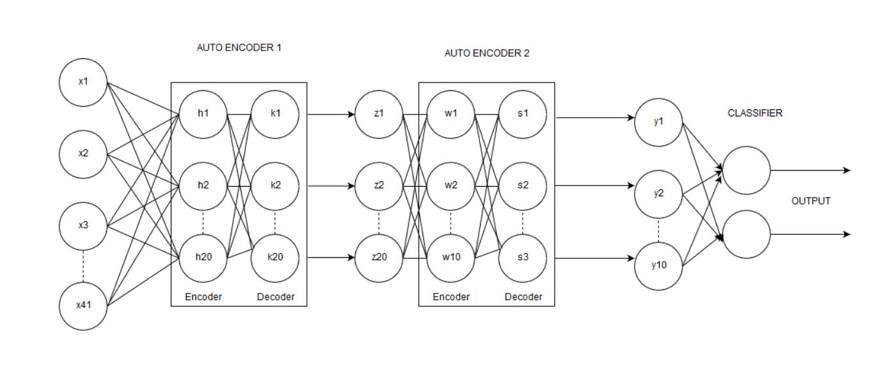
Using Label encoder the feature of type symbolic is converted into continuous value.

* Protocal\_type
* Service
* Flag
* Label

**Min-Max Normalization:**

Min max normalization is a normalization strategy which linearly transforms x to y= (x-min)/(max-min), where min and max are the minimum and maximum values in X, where X is the set of observed values of x. It can be easily seen that when x=min, then y=0, and when x=max, then y=1, This means, the min value in X is mapped to 0 and the maximum value in X is mapped to 1.So , the entire range of values of X from min to max are mapped to the range 0 to 1

* + 1. **DIMENSIONALITY REDUCTION**

****

**Figure 4.2** Dimensionality Reduction

In **Figure 4.2** represents the working of the stacked auto-encoders .The pre-processed dataset contains 41 feature attributes which is fed as input to our stacked auto encoders. The First auto-encoder coverts the 41 features into 20 features and it is fed into the auto encoder 2 which learns the data representation and outputs 10 features. These 10 features are used by NIDS for classifying type of attacks.

* + 1. **DECISION TREE CLASSIFIER**

The decision tree models are found to be very useful in the domain of data mining since they obtain reasonable accuracy and they are relatively inexpensive to compute. Decision tree classifiers are based on the “divide and conquer” strategy to construct an appropriate tree from a given learning set *S* containing a set of labeled instances. As a well-known and widely used algorithm, C4.5 algorithm developed by Quinlan generates accurate decision trees that can be used for effective classification.

C4.5 builds decision trees from a set of training data also with the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets.C4.5 examines the information gain ratio (can be regarded as normalized Information Gain) that results from choosing an attribute for splitting the data. The attribute with the highest information gain ratio is the one used to make the decision. Given a learning set *S* and a non-class attribute *X*, the *Information Gain Ratio* is defined as:



Where Si is the subset of S for which attribute X have a value and | S | is the number of samples in S. The decision trees are constructed as a set of rules during learning phase. Finally, it is used to predict the classes of new samples based on the rules.

**Decision Tree**

A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision. Decision tree are commonly used for gaining information for the purpose of decision -making.

Decision tree starts with a root node on which it is for users to take actions. From this node, users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

**Features of C4.5 Algorithm**

C4.5 is a popular decision tree based algorithm to solve data mining task. There is several features of C4.5.Some features of C4.5 algorithm are discussed below. Earlier versions of decision tree algorithms were unable to deal with continuous attributes. ‘An attribute must be categorical value’ was one of the preconditions for decision trees.

Another condition is ‘decision nodes of the tree must be categorical’ as well. Decision tree of C4.5 algorithm illuminates this problem by partitioning the continuous attribute value into discrete set of intervals which is widely known as ‘discretization’. For instance, if a continuous attribute C needs to be processed by C4.5 algorithm, then this algorithm creates a new Boolean attributes b so that it is true if C<b and false otherwise. Then it picks values by choosing a best suitable threshold.

**SUMMARY**

In this chapter architectural designs as well as different kinds of modules in developing the proposed system were discussed with detailed explanation.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

* 1. **DECISION TREE CLASSIFIER**

In this chapter various formulas involved in implementing the modules described in the proposed system are discussed.

* 1. **DATA PREPROCESSING**

The data should be pre processed because there is different variety of data in our NSL KDD data set. Using Label encoder the feature of type symbolic is converted into continuous value. Protocol type, Service, Flag, Label. The other data are having different ranging value so normalizing using Min-Max Normalization. Min max normalization strategy which linearly transforms x to y=(x-min)/(max-min), where min and max are the minimum and maximum values in X, where X is the set of observed values of x. It can be easily seen that when x=min, then y=0, and when x=max, then y=1, this means, the min value in X is mapped to 0 and the maximum value in X is mapped to 1. So , the entire range of values of X from min to max are mapped to the range 0 to 1. Another one is label encoding.

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. Label Encoder encode labels with a value between 0 and n\_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows. Depending on the data we have, we might run into situations where, after label encoding, we might confuse our model into thinking that a column has data with some kind of order or hierarchy when we clearly don’t have it. To avoid this, we ‘OneHotEncode’ that column. What one hot encoding does is, it takes a column which has categorical data, which has been label encoded and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

* 1. **AUTO ENCODER**

A popular technique currently utilized within deep learning research is auto-encoders, which is utilized by our proposed solution . An auto encoder is an unsupervised neural network-based feature extraction algorithm, which learns the best parameters required to reconstruct its output as close to its input as possible. One of it desirable characteristics is the capability to provide more a powerful and non-linear generalization than Principle Component Analysis (PCA). This is achieved by applying back propagation and setting the target values to be equal to the inputs. In other words, it is trying to learn an approximation to the identity function. An auto-encoder typically has an input layer, output layer (with the same dimension as the input layer) and a hidden layer. This hidden layer normally has a smaller dimension than that of the input (known as an under complete or sparse auto-encoder). An example of an auto-encoder.

NDAE, which is an auto-encoder featuring non-symmetrical multiple hidden layers. Fundamentally, this involves the proposed shift from the encoder-decoder paradigm (symmetric) and towards utilizing just the encoder phase (non-symmetric). The reasoning behind this is that given the correct learning structure, it is be possible to reduce both computational and time overheads, with minimal impact on accuracy and efficiency. NDAE can be used as a hierarchical unsupervised feature extractor that scales well to accommodate high-dimensional inputs. It learns non-trivial features using a similar training strategy to that of a typical auto-encoder. This is achieved by stacking our NDAEs to create a deep learning hierarchy .Stacking the NDAE offers a layer-wise unsupervised representation learning algorithm, which will allow our model to learn the complex relationships between different features. It also has feature extraction capabilities, so it is able to refine the model by prioritizing the most descriptive features. Due to the data that we envisage this model using, we have designed the model to handle large and complex datasets (further details on this are provided in Section VI). Despite the 42 features present in the KDDCup’99 and NSL-KDD data sets being comparatively small, we maintain that it provides a benchmark indication as to the model’s capability. However, the classification power of stacked auto-encoders with a typical soft-max layer is relatively weak compared to other discriminative models including RF, KNN and SVM

Most researchers use auto-encoders as a non-linear transformation to discover interesting data structures, by imposing other constraints on the network, and compare the results with those of PCA (linear transformation). These methods are based on the encoder-decoder paradigm. The input is first transformed into a typically lower-dimensional space(encoder), and then expanded to reproduce the initial data (decoder). Once a layer is trained, its code is fed to the next, to better model highly non-linear dependencies in the input. This paradigm focuses on reducing the dimensionality of input data. To achieve this, there is a special layer - the code layer, at the centre of the deep auto-encoder structure. This code layer is used as a compressed feature vector for classification or for combination within a stacked auto-encoder. The hidden layer is used to create a lower dimensionality version of high dimensionality data (known as encoding). By reducing the dimensionality, the auto-encoder is forced to capture the most prominent features of the data distribution. In an ideal scenario, the data features generated by the auto-encoder will provide a better representation of the data points than the raw data itself.

Deep learning can be applied to auto-encoders, whereby the hidden layers are the simple concepts and multiple hidden layers are used to provide depth, in a technique known as a stacked auto-encoder. This increased depth can reduce computational costs and the amount of required training data, as well as yielding. Greater degrees of accuracy. The output from each hidden layer is used as the input for a progressively higher level. Hence, the first layer of a stacked auto-encoder usually learns first order features in raw input. The second layer usually learns second-order features relating to patterns in the appearance of the first-order features.

* 1. **DECISION TREE CLASSIFIER**

**5-ClassClassification**:

By using the same 5 generic class labels as used in the KDD Cup ’99 dataset, we can compare the performance of the two models between the two datasets. It also aids comparability against similar works adopting this strategy. It is evident that our model offers increased accuracy, precision, recall, effectiveness (F-score) and the false alarm rate, when compared to the DBN approach.

**13-Class Classification**:

Our model is designed to work with larger and complex datasets. Therefore, we evaluate our model’s classification capabilities on a 13-class dataset. These 13 labels are those with more than the minimum 20 entries. The purpose of this analysis is to compare the stability of our model when the number of attack classes increases. Therefore, we do not compare these results

To perform our evaluations, we have used the KDD Cup ’99 and NSL-KDD datasets. Both of these datasets are considered as benchmarks within NIDS research. Furthermore, using these datasets assists in drawing comparisons with existing methods and research. In this, we will be using the metrics defined below:

1. True Positive (TP)-Attack data that is correctly classified as an attack.
2. False Positive (FP) - Normal data that is incorrectly classified as an attack.
3. True Negative (TN) - Normal data that is correctly classified as normal.
4. False Negative (FN) - Attack data that is incorrectly classified as normal.

We will be using the following measures to evaluate the performance of our proposed solution:

**Accuracy = TP + TN /TP + TN+ FP+ FN**

The accuracy measures the proportion of the total number of correct classification.

**Precision = TP /TP + FP**

The precision measures the number of correct classification. penalized by the number of incorrect classification..

**Recall = TP /TP + FN**

The recall measures the number of correct classification. Penalized by the number of missed entries

**F1-score =2\*((Precision\*Recall)/(Precision + Recall))**

The F-score measures the harmonic mean of precision and recall, which serves as a derived effectiveness measurement.

* + 1. **PERFORMANCE METRICS**

**Table 5.1** Confusion Matrix for 5 Classes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PREDICTED** | | | | | | |
| **ACUTAL** |  | **A** | **B** | **C** | **D** | **E** |
| **A** | **TPA** | EAB | EAC | EAD | EAE |
| **B** | EBA | **TPB** | EBC | EBD | EBE |
| **C** | ECA | ECB | **TPC** | ECD | ECE |
| **D** | EDA | EDB | EDC | **TPD** | EDE |
| **E** | EEA | EEB | EEC | EED | **TPE** |

In this, we will be using the metrics defined below:

1. The total number of test examples of any class would be the sum of the corresponding row (i.e. the TP+FN for that class).
2. The total number of FN’s for a class is the sum of values in the corresponding row (excluding the TP).
   * + **FNA= EAB+ EAC+ EAD+ EAE**
     + **FNB =EBA+ EBC+ EBD+ EBE**
3. The total number of FP’s for a class is the sum of values in the corresponding column(excluding the TP).

* **FPA=EBA+ ECA+ EDA+ EEA**
* **FPB=EAB+ ECB+ EDB+ EEB**

1. The total number of TN’s for certain will be the sum of all column and rows excluding that class’s column and row.

We will be using the following measures to evaluate the performance of our proposed solution:

**Accuracy**: Calculate as the sum of correct classifications divided by the total number of classifications.

**Precision=TP/(TP+FP):**

* + Precision A=**TPA/(TPA+ EBA+ ECA+ EDA+ EEA)**
  + Precision B=**TPB/(TPB+ EAB+ ECB+ EDB+ EEB)**

**Recall=Sensitivity=TP/(TP+FN):**

* + Recall A =Sensitivity A = **TPA/(TPA+ EAB+ EAC+ EAD+ EAE)**
  + Recall B =Sensitivity B = **TPB/(TPB+ EBA+ EBC+ EBD+ EBE)**

**Specificity = TN/(TN+FP):**

* + Specificity A = **TNA/( TNA+ EBA+ ECA+ EDA+ EEA)**
  + Specificity B = **TNB/( TNB+ EAB+ ECB+ EDB+ EEB)**
  1. **DATASET DETAILS**

The Data set that we use in our proposed system is NSL-KDD data set which is a benchmark data set.

**Table 5.2** Dataset details

|  |  |
| --- | --- |
| **NUMBER OF RECORDS** | * TRAINING-125974 * TESTING-22543 |
| **FEATURES** | * 41 |
| **ATTACKS OR CLASS** | * NORMAL * DOS * PROBE * U2L * R2L |

In (**Table 5.2)** shows the details of the dataset. It also describes the number of records present in the dataset, features and attacks.

**SUMMARY**

This chapter had described about the system implementation as well as various formulae used in the proposed system.

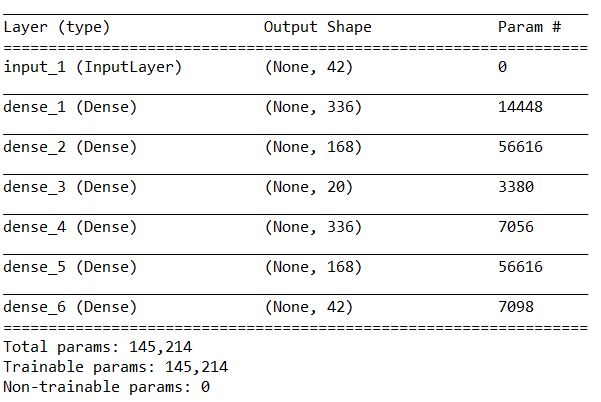
**CHAPTER 6**

**RESULTS AND DISCUSSIONS**

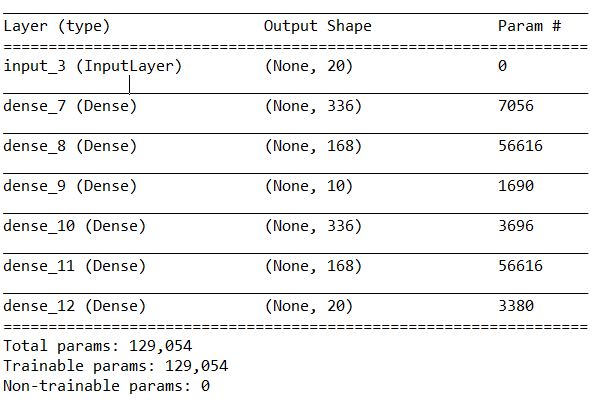
* 1. **INTRODUCTION**

This step contains results of various intermediate steps of the proposed system.

* 1. **IMPLEMENTATION RESULTS**



**Figure 6.1** First Auto Encoder



**Figure 6.2** Second Auto Encoder

**Table 6.3** Confusion Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PREDICATED** | | | | | | |
| **ACTUAL** |  | **NORMAL** | **DOS** | **PROBE** | **R2L** | **U2R** |
| **NORMAL** | 5842 | 1578 | 40 | 0 | 0 |
| **DOS** | 12 | 9630 | 68 | 0 | 0 |
| **PROBE** | 0 | 351 | 2060 | 10 | 0 |
| **R2L** | 0 | 76 | 401 | 2307 | 101 |
| **U2R** | 0 | 0 | 0 | 22 | 45 |

* 1. **PERFORMANCE ANALYSIS**
     1. **COMPARISON OF PERFORMANCE METRICS**

**Figure 6.3** Comparison of existing [2] method vs proposed method performance

Figure 6.3 represents the comparison of overall performance metrics of the existing [1] and proposed method

* + 1. **COMPARISON OF METRICS FOR EACH ATTACK OF EXISTING [1] AND PROPOSED METHOD**

**Figure 6.4** Accuracy Comparison

The Figure 6.4 represents Accuracy comparison of existing [1] and proposed method

**Figure 6.5** Precision Comparison

The Figure 6.5 represents Precision comparison of existing [1] and proposed method

**Figure 6.6** Recall Comparison

The Figure 6.6 represents Recall comparison of existing [1] and proposed method

**Figure 6.7** F-Score Comparison  
The Figure 6.7 represents F-Score comparison of existing [1] and proposed method

**SUMMARY**

This chapter had described the Implementation Results and Performance Analysis of the proposed work.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

* 1. **CONCLUSION**

In our work, we have discussed the problems faced by existing NIDS techniques. In response to this we have proposed our novel NDAE method for unsupervised feature learning. We have then built upon this by proposing a novel classiﬁcation model constructed from stacked NDAEs and the RF classiﬁcation algorithm. We have implemented our proposed model in Tensor Flow and performed extensive evaluations on its capabilities. For our evaluations we have utilized the benchmark KDD Cup ’99 and NSL-KDD datasets and achieved very promising results.

Our results have demonstrated that our approach offers high levels of accuracy, precision and recall together with reduced training time. Most notably, we have compared our stacked NDAE model against the mainstream DBN technique. These comparisons have demonstrated that our model offers up to a 5% improvement in accuracy and training time reduction of up to 98.81%. Unlike most previous work, we have evaluated the capabilities of our model based on both benchmark datasets, revealing a consistent level of classiﬁcation accuracy.

* 1. **FUTURE ENHANCEMENT**

In our future work, the ﬁrst avenue of exploration for improvement will be to assess and extend the capability of our model to handle zero-day attacks. We will then look to expand upon our existing evaluations by utilizing real-world backbone network trafﬁc to demonstrate the merits of the extended model.

**APPENDIX I**

**WORKING ENVIRONMENT**

**Hardware Requirement**

**Processor :** Intel core i7

**Hard Disk :** 1TB+252GB SSD

**RAM :** 16GB

**Graphics Card :** Nivida GTX 1050 Ti 2GB

**Software Requirement**

**Language Used :** Python and TensorFlow

**IDE Used :** Spyder 3.3.2

**Operating System :** Windows 10

**APPENDIX II**

**CODING**

**Data Pre processing of Train Dataset**

col\_names = np.array(["duration","protocol\_type","service","flag","src\_bytes",

"dst\_bytes","land","wrong\_fragment","urgent","hot","num\_failed\_logins",

"logged\_in","num\_compromised","root\_shell","su\_attempted","num\_root",

"num\_file\_creations","num\_shells","num\_access\_files","num\_outbound\_cmds",

"is\_host\_login","is\_guest\_login","count","srv\_count","serror\_rate",

"srv\_serror\_rate","rerror\_rate","srv\_rerror\_rate","same\_srv\_rate",

"diff\_srv\_rate","srv\_diff\_host\_rate","dst\_host\_count","dst\_host\_srv\_count",

"dst\_host\_same\_srv\_rate","dst\_host\_diff\_srv\_rate","dst\_host\_same\_src\_port\_rate",

"dst\_host\_srv\_diff\_host\_rate","dst\_host\_serror\_rate","dst\_host\_srv\_serror\_rate",

"dst\_host\_rerror\_rate","dst\_host\_srv\_rerror\_rate","label"])

train = pd.read\_csv('./KDDTrain.csv',names=col\_names)

classgroup\_map = {'back':'dos','buffer\_overflow':'u2r','ftp\_write':'r2l','guess\_passwd':'r2l','imap':'r2l', 'ipsweep':'probe','land':'dos','loadmodule':'u2r','multihop':'r2l','neptune':'dos','nmap':'probe', 'perl':'u2r','phf':'r2l','pod':'dos','portsweep':'probe','rootkit':'u2r','satan':'probe', 'smurf':'dos','spy':'r2l','teardrop':'dos','warezclient':'r2l','warezmaster':'r2l','normal':'normal', 'apache2':'dos','httptunnel':'r2l','mailbomb':'dos','mscan':'probe','named':'r2l','processtable':'dos', 'ps':'u2r','saint':'probe','sendmail':'r2l','snmpgetattack':'r2l','snmpguess':'r2l','sqlattack':'u2r' 'udpstorm':'dos','worm':'dos','xlock':'r2l','xsnoop':'r2l','xterm':'u2r'}

train['label'] = train['label'].map(classgroup\_map)

le=LabelEncoder()

train['protocol\_type'] = le.fit\_transform(train['protocol\_type'])

train['service'] = le.fit\_transform(train['service'])

train['flag'] = le.fit\_transform(train['flag'])

train['label'] = le.fit\_transform(train['label'])

train1 = train.iloc[:,0:42].values

test1 = test.iloc[:,0:42].values

for i in range(0,41):

if(max(train1[:,i])!=0):

train1[:,i] = ((train1[:,i]-(min(train1[:,i])))/(max(train1[:,i])- (min(train1[:,i])))).astype(np.float32)  
  
**Dimensionality Reduction using First AutoEncoder**

encoding\_dim = 20

input\_dim = Input(shape = (ncol, ))

encoded1 = Dense(42\*8, activation = 'relu')(input\_dim)

encoded2 = Dense(42\*4, activation = 'relu')(encoded1)

encoded3 = Dense(encoding\_dim, activation = 'relu')(encoded2)

decoded1 = Dense(42\*8, activation = 'relu')(encoded3)

decoded2 = Dense(42\*4, activation = 'relu')(decoded1)

decoded3 = Dense(ncol, activation = 'sigmoid')(decoded2)

autoencoder = Model(inputs = input\_dim, outputs = decoded3)

autoencoder.compile(optimizer = 'sgd', loss = 'mean\_squared\_error',metrics=['accuracy'])

autoencoder.summary()

autoencoder.fit(x=train1, y=train1, nb\_epoch =20, batch\_size = 32, shuffle = False, validation\_data = (test1, test1))

encoder = Model(inputs = input\_dim, outputs = encoded3)

encoded\_input = Input(shape = (encoding\_dim, ))

encoded\_train = pd.DataFrame(encoder.predict(train1))

encoded\_train = encoded\_train.add\_prefix('feature\_')

encoded\_train['label']=train\_label

encoded\_test = pd.DataFrame(encoder.predict(test1))

encoded\_test = encoded\_test.add\_prefix('feature\_')

encoded\_test['label']=test\_label

encoded\_train.to\_csv('train\_encoded2.csv', index=False)

encoded\_test.to\_csv('test\_encoded2.csv', index=False)

**Decision Tree**

dataset\_train = pd.read\_csv('train\_encoded3.csv')

X\_train = dataset\_train.iloc[:,0:10].values

y\_train = dataset\_train.iloc[:, 10].values

dataset\_test = pd.read\_csv('test\_encoded3.csv')

X\_test = dataset\_test.iloc[:, 0:10].values

y\_test = dataset\_test.iloc[:, 10].values

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

**REFERENCES**

1. Nathan Shone , Tran Nguyen Ngoc, Vu Dinh Phai , and Qi Shi, ”A Deep Learning Approach to Network Intrusion Detection”, IEEE Transactions On Emerging topics in Computational Intelligence, vol. 2, no. 1, February 2018
2. G. E. Hinton and R. R. Salakhutdinov ,”Reducing the dimensionality of data with neural networks”,Vol 313,Issue 5786,pp. 504-507 ,Science Journal- 28 July 2006
3. JihyunKim ;Jaehyun Kim ; Huong Le Thi Thu ; HowonKim,”Long short term memory recurrent neural network classifier to intrusion detection” 2016 International Conference on Platform Technology and Service, 15-17 Feb. 2016
4. Khaled Alrawashdeh ; Carla Purdy,” Toward an online anomaly intrusion detection system based on deep learning”, 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 18-20 Dec. 2016
5. Mohammed Hasan Ali,Bahaa Abbas Dawood Al Mohammed, AlyaniIsmail ,And Mohamad FadliZolkipli,”A new intrusion detection system based on fast learning network and particle swarm optimization”, IEEE Access ( Volume: 6 ), 27
6. Monowar H. Bhuyan, Bhattacharyya D.K., KalitaJ.K., ”Network anomaly detection: Methods, systems and tools IEEE Communications Surveys & Tutorials ( Volume: 16 , Issue: 1 , First Quarter 2014 ), 06 June 2013),March 2018

**WEB REFERNCES**

* + - <https://kb.juniper.net/InfoCenter/index?page=content&id=KB147373>
    - <http://arxiv.org/abs/1612.07640>
    - <https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>
    - http://ufldl.stanford.edu/wiki/index.php/Stacked\_Autoencoders