



***ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY***

**ANOMALY INTRUSTION DETECTION SYSTEM  
BASED ON RECURRENT NEURAL NETWORK**

**MASTERS THESIS**

**By**

**BELAYNEH BARIGA ABETU**

**Supervisor: SOLOMON ZEMENE(Ph.D.)**

**A Thesis Submitted as a Partial Fulfillment to the Requirements for the  
Award of the Degree of Master of Science in Electrical and Computer  
Engineering**

**(Computer Engineering)**

**to**


**COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING**

**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING**

**JANUARY 2021.**

## DECLARATION

I hereby declare that this thesis entitled “**Anomaly Intrusion Detection Systems Based on Recurrent Neural Network**” used to be prepared by me, with the guidance of my advisor. I declare that the thesis is my original work and has not been for a degree in any other university. That means, the work contained herein is my own without where explicitly mentioned in any other case in the text, and that this work has not been submitted, in whole or in part, for any other degree. Parts of this work have been published in [state previous publication].

Name: Beaarneth Bariza Signature:  Date: 10/2/21

## APPROVAL PAGE

This is to certify that the thesis prepared by Mr. Belayneh Bariga entitled “Anomaly Intrusion Detection System Based on Recurrent Neural Network” and submitted as partial fulfillment for the award of the Degree of Master of Science in Electrical and Computer Engineering (Computer Engineering) complies with the regulations of the university and meets the accepted standards with respect to originality, content and quality.

### Signed by Examining Board:

Advisor: Solomon Z Signature: [Signature] Date: 10/2/21  
External Examiner: Dr. Beakal G. Signature: [Signature] Date: Feb 10/2021  
Internal Examiner: Dr. Derge Y Signature: [Signature] Date: 10-Feb-21  
Chairperson: Yonas Tesfaye Mekecha Signature: \_\_\_\_\_ Date: \_\_\_\_\_  
Head of Computer Engineering Department  
DGC Chairperson: [Signature] Signature: [Signature] Date: 11-Feb-21  
College Dean/Associate Dean for GP: Muluneh Mekonnen Tulu (PhD) Signature: [Signature] Date: 11/02/21  
Associate Dean for College of Electrical and Mechanical Engineering



## **ACKNOWLEDGMENT**

First and foremost, my thanks would like to go to the Almighty God and His Mother Saint Merry, almighty custodian of Saint Gebriel and Saint Michael. Secondly, I would like to specific my straightforward gratitude to my advisor Dr. Solomon Zemene for the continuous help of my thesis study, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis, moreover his insightful comments and encouragement which added great value for my future career.

Besides my advisor, I would like to thank department of electrical and computer engineering those who assign research advisors and learning environment. And then, i am highly grateful to my parents, they taught me the right, encouraged me and gave me hope and unconditional love. I wish both of them happiness and nicely health. Especially, my mom you mean a lot to me, without you nothing will be completed successfully in my life and also my sisters. The final, however no longer the least thanks goes to my brothers, they are the motivating force at the back of me at all instances through both ups and downs of my academic life. Thank them for everything they give me.

**Thank you!**

## ABSTRACT

Despite the fast development in information technology, the problem of defending computer and network security become a main challenge for most organizations. Networks security need some tools and methods for protection due to the increase of security threats. Various ways have been developed to secure computer networks and communication over the Internet, such as firewall, authentication tools, and virtual private networks that create a protective shield almost always with vulnerabilities. This has produced Intrusion Detection Systems (IDS) to be settled that complement traditional approaches. However, with the expansion of computer technology, the behavior of intrusions has become complex, that makes the work of network security professionals hard to analyze and detect intrusions. Since, one of the challenge of an IDS is, conversion of patterns of class label into different attacks. This is mainly due to, the widespread number of vulnerabilities in computer systems and creativity of the attackers.

To overcome this issue, the thesis proposes anomaly intrusion detection system based on various recurrent neural networks, which were Simple Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) and Gate Recurrent Unit (GRU) used to identify and detect the attacks, Denial of Service (DoS), Probe, User to Root (U2R) and Root to Load (R2L) on the NSL\_KDD dataset. This helps to identify a crucial module that defend against the malicious traffics before the system is affected. Two experiments are performed on the proposed networks and dataset. These for using all and minimal features of the dataset. Overall classification accuracy, training time and accuracy are the evaluation methods of the systems work.

And the obtained results are as follow. In terms of overall classification accuracy, Simple RNN tended to lead among GRU and LSTM algorithms, which scored has 85.7115% on the test data. Although, LSTM and GRU offered good performance with accuracy 85.2775% and 85.6725% from test data, however they took a longer time for training compared to SimpleRNN, these is for using all input features. But for using minimal features of the dataset, SimpleRNN also scored good overall classification performance accuracy among GRU and LSTM on the unseen data. Moreover, for the classification and detection categories of attacks. U2R has achieved promised results for both experiments on these three algorithms among the other attacks.

**Keywords:** Computer Network Security, Anomaly Intrusion Detection System, NSL\_KDD Dataset, Variant Recurrent Neural Networks.

## TABLE OF CONTENTS

<b>Declaration.....</b>	<b>Error! Bookmark not defined.</b>
<b>Approval Page .....</b>	<b>Error! Bookmark not defined.</b>
<b>Acknowledgment.....</b>	<b>iv</b>
<b>Abstract.....</b>	<b>v</b>
<b>Table of Contents .....</b>	<b>vi</b>
<b>List of Figures.....</b>	<b>ix</b>
<b>List of Table.....</b>	<b>x</b>
<b>List of Abbreviations .....</b>	<b>xi</b>
<b>Chapter One: Introduction .....</b>	<b>1</b>
1.1 Overview .....	1
1.2 Motivation .....	2
1.3 Statement of the Problem .....	2
1.4 Research Questions .....	3
1.5 Objectives.....	3
1.5.1 General Objective .....	3
1.5.2 Specific Objectives .....	3
1.6 Scope and Limitation of the Study .....	3
1.7 Significance of the Study .....	4
1.8 Benefit of the research.....	4
1.9 Contributions.....	5
1.10 Research Methodology.....	5
1.10 Thesis Organization.....	6
<b>Chapter two: Background and Literature Review .....</b>	<b>7</b>
2.1 Overview .....	7
2.2 Computer System Security.....	7
2.3 Intrusion Detection System .....	7
2.4 Types of Intrusion Detection System .....	8
2.4.1 Anomaly Based Intrusion Detection System.....	8
2.4.2 Misuse Based Intrusion Detection System .....	9
2.4.3 Host and Network Based Intrusion Detection System .....	9
2.4.4 Passive and Active Intrusion Detection System .....	10
2.5 Machine Learning and Deep Learning .....	11

2.6 Literature Review .....	12
2.6.1 IDSs Based on Machine Learning Methods .....	12
2.6.2 IDSs Based on Deep Learning Methods.....	13
<b>Chapter Three: Methodology and Material Used.....</b>	<b>20</b>
3.1 Overview .....	20
3.2 A Overview of Systems Architecture.....	20
3.3 Data Collection and Preparation .....	22
3.5 Dataset Attack Types .....	24
3.5.1 Denial of Service (DoS) Attacks .....	24
3.5.2 User to Root (U2R) Attacks .....	25
3.5.3 Remote to Local (R2L) Attacks.....	25
3.5.4 Probe Attacks.....	26
3.6 Data Preprocessing.....	26
3.6.1 Data Cleaning .....	27
3.6.2 Label Encoding.....	28
3.6.3 One-Hot Encoding.....	29
3.6.4 Feature Scaling(Normalization) .....	29
3.6.5 Feature Selection .....	30
3.7 Algorithms Implementation .....	31
3.7.1 Basic Concept and Architecture of (RNN) Network.....	31
3.7.2 Basic Concept and Architecture of (LSTM) Network.....	32
3.7.3 Basic Concept and Architecture of (GRU) Network.....	34
3.8 Parameters Tuning Methods.....	35
3.8.1 Manual Tuning .....	35
3.8.2 Grid Search .....	35
3.9 Hyper Parameters .....	35
3.9.1 Hidden Layers.....	36
3.9.2 Hidden Units.....	36
3.9.3 Epochs .....	36
3.9.4 Activation Functions.....	36
3.9.5 Optimizers .....	37
3.9.6 Drop Out Function.....	38
3.9.7 Batch Sizes .....	39
3.10 Materials and Tools Used to Implement the System.....	39

3.10.1 The Hardware Material Requirement .....	39
3.10.2 The Software Tools Requirement.....	39
3.11 Performance Evaluation Metrics .....	39
3.11.1 Model Training and Testing Accuracy .....	39
3.11.2 Overall Classification Rate Accuracy.....	40
<b>Chapter Four: Result and Discussion .....</b>	<b>41</b>
4.1 Introduction .....	41
4.2 Preparation of the Experimental Dataset.....	41
4.3 Experimental Methodology .....	41
4.4 Experiments of Proposed System.....	42
4.3 Phase One and Two Experimental Result .....	43
4.4.1 SimpleRNN Experiments Using All and Minimal features .....	44
4.4.2 LSTM Experiment Using All and Minimal Features .....	46
4.4.3 GRU Experiment Using All and Minimal Features .....	49
4.5 Experimental Result Discussion.....	51
4.5.1 Comparison of Proposed Work .....	53
4.5.2 Comparison to Existing System .....	53
<b>Chapter Five: Conclusion and Future Work .....</b>	<b>55</b>
5.1 Conclusion.....	55
5.2 Future Work .....	55
<b>References.....</b>	<b>56</b>
<b>Appendices.....</b>	<b>63</b>
Appendix A: Full Description of NSL_KDD Dataset. ....	63
Appendix B: Sample Source Code.....	65
Appendix C: List of Selected Features for the NSL_KDD dataset.....	67
Appendix D: Capture of Sample Training and Testing.....	68



## LIST OF FIGURES

Figure 2. 1: Types of intrusion detection systems. ....	8
Figure 2. 2: Steps of anomaly intrusion detection system. ....	9
Figure 2. 3: Setup of a Network based Intrusion detection system. ....	10
Figure 2. 4: Steps of host base intrusion detection systems.....	10
Figure 3. 1: Overview of NAIDSs architecture.....	21
Figure 3. 2: Proposed system architecture. ....	23
Figure 3. 3: Basic loop structure in RNN [59].....	32
Figure 3. 4: Long short term memory architecture [62]. ....	33
Figure 3. 5 : Architecture of GRU cells[65]. ....	34
Figure 3. 6: Working principle of dropout function to neural network. ....	38
Figure 4. 1:Experimental system design methodology.....	43
Figure 4. 2: Capture of configuration of SimpleRNN model for all features. ....	44
Figure 4. 3: Capture of configuration of SimpleRNN model for minimal features.....	44
Figure 4. 4: Performance accuracy of simpleRNN for using 122 input features.....	45
Figure 4. 5: Performance accuracy SimpleRNN model for using 22 input features. ....	45
Figure 4. 6: Capture of configuration of the proposed LSTM model for using all features. ...	46
Figure 4. 7: Capture of configuration LSTM model for using minimal features. ....	47
Figure 4. 8: Accuracy predicted for all attack types with 122 features for LSTM classifiers. 48	
Figure 4. 9: Accuracy predicted for all attack types with 22 features for LSTM classifiers. ..	48
Figure 4. 10: Capture of configuration of the proposed GRU model using all features.....	49
Figure 4. 11: Capture of configuration of the proposed GRU model for minimal features. ...	49
Figure 4. 12: Accuracy predicted for all attack types with 122 features for GRU classifiers. 50	
Figure 4. 13: Accuracy predicted for all attack types with 22 features for GRU classifiers. ..	50

## LIST OF TABLE

Table 2. 1: Summary of different related works.....	18
Table 3. 1: NAN values in the NSL_KDD dataset.....	27
Table 3. 2: Before label Encoding. ....	28
Table 3. 3 : After label Encoding.....	28
Table 4. 1: List of implementation hyper parameters and its set of value.....	42
Table 4. 2 : SimpleRNN model using all and minimum features of the dataset.....	44
Table 4. 3: LSTM performance using all and minimum features of the dataset. ....	47
Table 4. 4 : GRU performance using all and minimum features of the dataset.....	50
Table 4. 5: Comparison with previous studies.....	54

## **LIST OF ABBREVIATIONS**

ANOVA:	Analysis of Variance
AMD:	Advanced Micro Devices
AIDS:	Anomaly Intrusion Detection Systems.
BPTT:	Back Propagation Through Time
CSV:	Common Separate Values
CFS:	Correlation Based Feature Selection
CICIDS2017:	Canadian Institute for Cyber Security 2017
DoS:	Denial-Of-Service
DNN:	Deep Neural Network
DARPA:	Defense Advanced Research Project Agency
FTP:	File Transfer Protocol
GRU:	Gated Recurrent Unit
HIDS:	Host Intrusion Detection Systems.
ICMP:	Internet Control Message Protocol
IP:	Internet Protocol
ISCXS IDS2012: systems 2012	Information Security Centre of Excellence Intrusion Detection
JSON:	Java Script Object Notation
KDD-CUP99:	Knowledge Discovery and Data Mining (KDD)-CUP 99

LSTM:	Long Short Term Memory.
NSL_KDD:	Network Simulation Language Knowledge Discovery in Database.
R2L:	Root to Load.
RNN:	Recurrent Neural Network.
TCP:	Transport Control Protocol.
U2R:	User to Root.
UDP:	User Datagram Protocol.
NAN:	Not A Number.
ICMP:	Internet Control Message Protocol.
HTTP:	Hypertext Transfer Protocol.
WEKA:	Waikato Environment for Knowledge Analysis.

# CHAPTER ONE: INTRODUCTION

## 1.1 Overview

Intrusion is a set of actions, the actions maybe cyber-attack incidents that are increasing with the increasing use of internet. Cyber-attack is the virtual life of the abuse in normal life, in this attack person encounters the situation of like, harassment, threats and blackmail. Intrusion detection systems(IDSs) are very significant software or hardware safety tools to eliminate threats that would then occur when carrying information, to stop unauthorized access or abuse, and to report attacks to those responsible for security [1]. A common method for intrusion detection is identifying and detecting anomalies in network traffic, and to protect infrastructure of network systems, which gather and analyze information from various areas inside a host or a network to detect likely security breaches [1].

Over the years, a number of machine learning approaches have been developed to detect network intrusions. However, today IDS on a deep learning is a key research area in network security, due to an increase of attacks on computers and networks [2]. Recurrent Neural Network (RNN) is the greatest current technique of carrying out classification and other analysis on sequences of data, it can access past information because of its loop like structure. In RNN, recurrent links can be designed in three techniques, among a neuron and a neuron itself or between a neuron and a neuron in the similar layer, or with the neuron and a neuron in the earlier layer of neural network architecture. These recurrent connections are formed with hidden and output neurons only and not bias neurons. This type of architecture makes it useful to persist past information in order to predict current information and to deal with different learning rates [3].

Long Short-Term Memory (LSTM) system is type of RNN. These networks has special memory cell structure, which is intended to hold long-term dependencies in data [4]. Much later, a decade and half after LSTM, gated recurrent unit (GRU) were introduced by [5]. They are like to LSTM networks, but with a simpler architecture, properly to work on long-term dependencies and sequential data. For this thesis to implement these three algorithm, for the classification and detecting of unknown attack on the NSL\_KDD dataset, python programing language with different parameters were used to implement the system work.

## **1.2 Motivation**

The motivation of this study work are the utility of intrusion detection devices for the one-of-a-kind organization, availability of open sources and weakness of currently on hand network protection equipment concerning detection of intrusion. Even though, intrusion detection is applicable in different organizations and used more than a decade, but there is still exist many troubles round IDS since computer systems is not secure 100%. And another motivation of this study is focused on, solving the problems in intrusion detection community that can help the administrator to make data pre-processing, classification and labeling of data and to mitigate the consequence of unique attacks, due to the crushing advance of attacks which makes the task hard. Attacks can be identified only after it happens. To overcome this situation, the usual updating of profiles is needed, the reduced workload of the officer growths the detection of attacks. Machine and deep learning consists of many different algorithms to accomplish the preferred tasks. All of these are aims to fit a model to the prescribed data and even analyze the data and simulate a model that is closest to the data being analyzed.

## **1.3 Statement of the Problem**

In the network security area, monitoring the whole network traffic in a network is a challenging issue, mainly because overlapping of the protocols “protocol layering” makes it difficult to extract features of packets quickly and monitor the network. Another task in IDS is a different of attack types including for Probe, Remote to Local (R2L), User to Root (U2R), and Denial of Service(DoS) should be identified well by IDS methods. Since, to convert the pattern of class labels into different attacks type is difficult.

However, one of the most problematic attacks to detect is R2L and U2R attack, for the reason that R2L attack is linked to the host stage features and network level. Besides, U2R attack is additionally challenging to detect at formerly stage due to the fact that it concerned the semantic elements such as content-based and target an application, this will come to be one problem in IDSs. And also, another problem announcement of this study addressed comparing the variant RNN themselves, these was to determine which of them yields better on the categories of attack, in terms of classification performance accuracy and execution of training and testing time.

## **1.4 Research Questions**

The following research questions (RQ) are derived from the intention of the thesis. The RQ1 is necessary for the first part of the objective of the thesis. The important of the RQ2 and RQ3 is to present how three methods works and gather results that are comparable with existing systems.

1. What methods for detecting DoS, Probe, R2L and U2L attacks are presented and How to enhance the performance of these attacks?
2. Does SimpleRNN, LSTM and GRU can detect the classification of anomaly attacks on the dataset?
3. What are the set of parameter that helps to achieve high accuracy with less training time for this study and how to set these parameters?

## **1.5 Objectives**

### **1.5.1 General Objective**

The general objective of this study is, to build Recurrent Neural Network approaches for intrusion detection system (IDS) that will enhance the computer network security system.

### **1.5.2 Specific Objectives**

The specific objectives of this research study are to:

- Explore the latest dataset that will be used for the design model training and testing.
- Preprocess the dataset and identify and classify the best features for model training and testing.
- Distinguishes normal and attack connection and categorized attack to their family using all features and using minimal features.
- Explore SimpleRNN, LSTM and GRU algorithms and train them on the prepared new training and testing data, and evaluate each of them.

## **1.6 Scope and Limitation of the Study**

Within the scope of this thesis, cyber-attack detection through a recurrent neural network learning based IDS model used to be targeted. Only three classes of RNNs approaches are to

be considered, Simple RNN, LSTM, and GRU will be used for developing the models. For the feature selection methods, filter-based feature selection was used. The proposed models are an instance of anomaly-based IDS, Signature-based IDS and hybrid-based IDS models are outside the scope of this thesis. Because of the data set, the cyber-attack of the NSL\_KDD offline mode dataset was once chosen for evaluation purposes of the models. Whereas, within the limitation of this study, due to time constraint this research has seemed at and targeted primarily on intrusion detection methods and techniques. This work did not look at other components of an intrusion detection system, like data collection and response for the reason that it is very huge area of research. And also, some other difficulty of this study is, very difficult to train and test the algorithms when certain amount of traffic data is not available on the dataset or when using minimum amount of traffic data performance accuracy of the design models not good. Another limitation of this study, only detect the unknown attack cannot prevent the intrusions and it is offline system.

### **1.7 Significance of the Study**

The advantage of this investigation is to sort and detect not normal behavior of network traffic, by using various categories of recurrent neural network algorithms. This system is considered by: - processing, classifying and analyzing network traffic data, to discover and accurately identify the anomaly attacks by increasing classification rate accuracy to the minimal, this means that detecting the intrusion by making use of unknown attacks pattern in the training phase and testing phase to increase classification detection accuracy. In general, main significance of this study is to evaluate and compare performance of different RNN algorithms on the NSL\_KDD dataset. Moreover, to solve the classification detection accuracy of the unknown attack of U2R on this dataset.

### **1.8 Benefit of the research**

Computer security has become important as the use of information technology has become part of our daily lives. Companies and home users keep sensitive information on their computers, there is a great necessity to protect that information from those who would exploit it. Intrusion detection system is main technique to overcome the problem of network security and minimize the impact of intruders on properly flow of organization work. But one of application result of this research are to safeguard businesses and organization networks from malicious activities, this is which can help organizations to protect their network from external and internal attacks.



Network security is a growing industry as more and more of the corporate workspace is converted to digital media. For example, today cyber criminals primarily focused on bank systems, great business area. So one way to help keep this was by using an Intrusion Detection System (IDS).

## **1.9 Contributions**

The main contributions of this thesis are summarized below:

- We present, design and implement variant recurrent neural networks methods to detect anomaly attacks on the dataset. Moreover, we study the performance of the models in the categories of U2R and R2L attacks, by enhancing classification accuracy with short training time.
- By contrast, we study the performance of SimpleRNN, LSTM and GRU algorithms themselves, for using all and minimal features of unknown attack on the dataset. And also compare the obtained results of all proposed works to other existing works in terms of overall classification accuracy and training time.

## **1.10 Research Methodology**

Different methodologies will be employed in this research in order to accomplish the objectives and problem declaration of this study. Therefore, the methodology that is followed to conduct this research is described as follows. The first mechanism to do is to conduct a comprehensive review of the literature to acquire a deeper understanding of the research area and its problem domains. This is found in chapter two deeply based on the methods when they used. Through this literature, we identify the significance of the previous works done in the part of anomaly intrusion detection based on a different machine and deep learning area. Existing works related to this research work assessed and point direction in providing a solution to identified problems. The second thing to do is to look for data collection and the datasets to learn about patterns that can identify different categories of attacks from normal on the dataset. Based on the proposed solution from the identified problem in the literature survey we have to select appropriate tools, techniques, algorithms and hyper-parameters. After identifying those requirements to design and implement the anomaly intrusion detection system using variant recurrent neural network approaches. Then finally, to evaluate the system in terms of, overall classification accuracy and classification accuracy of individual unknown attacks.

## **1.10 Thesis Organization**

The following are the reset of the thesis structured of the proposed works: the aim and scope of the thesis are stated in the first chapter. Chapter two introduces: background description of the proposed systems as well as concepts and theories about intrusion detection system and related work in the area of machine and deep learning methods. Chapter three discuss proposed implementation methodology, experimental setup and parameters used. Chapter four discuss and presents experimental results and discussions, chapter five concludes the overall work and futures work of this thesis.

## **CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW**

### **2.1 Overview**

This chapter presents and introduces the impact of the network security problem, key function and types of intrusion detection system, application of machine and deep learning. Afterwards, some conventional anomaly network intrusion detection methods are briefly discussed. In the next and final section, review of intrusion detection on machine and deep learning approaches are point-out and summary of more related work on the literature review are document in the form of table.

### **2.2 Computer System Security**

In [6], define a secure computer system is one that can be depended upon to act as likely. The reliability that is displayed between the likely behavior and the exhibited behavior is referred to as trust in the security of the computer system. They define the level of confidence as an indication of the assurance in the expected behavior of the computer system. Computer security also defined to confidentiality to mean information is available only those authorized to access it. Integrity assures that information remains unchanged by accident or malicious damaging and availability ensures that the computer system remains working when needed without deprivation of access to approved users [7].

### **2.3 Intrusion Detection System**

It is network security applications that monitor network for malicious activity. The aim of IDS is seemingly simple to detect intrusions. However, the task is hard and in fact detection of intrusion do not detect at all only identify sign of intrusions either while they are in progress or after the fact, such evidence is sometimes referred to as an attacks or manifestation. The attack may be in the form of capturing of the person's password. If there is no manifestation, if the manifestation lacks sufficient information, or if the information it contains is untrustworthy, then the system cannot detect the intrusion [8].

The reasons for the need for, intrusion detection system(IDS) are as follows:

- It detects attacks that cannot be stopped by other security tools.
- It answers to the analysis stage before the attack happens.

- It allows attack analysis, systems repair and the attacking issues to be corrected.

## 2.4 Types of Intrusion Detection System

It can be categorized in to various approaches, some of the approach are: misuse and anomaly detection, host and network, passive and active intrusion detection systems. Figure 2.1 shows various approaches to classify IDSs and the description of each type are discussed below [9].

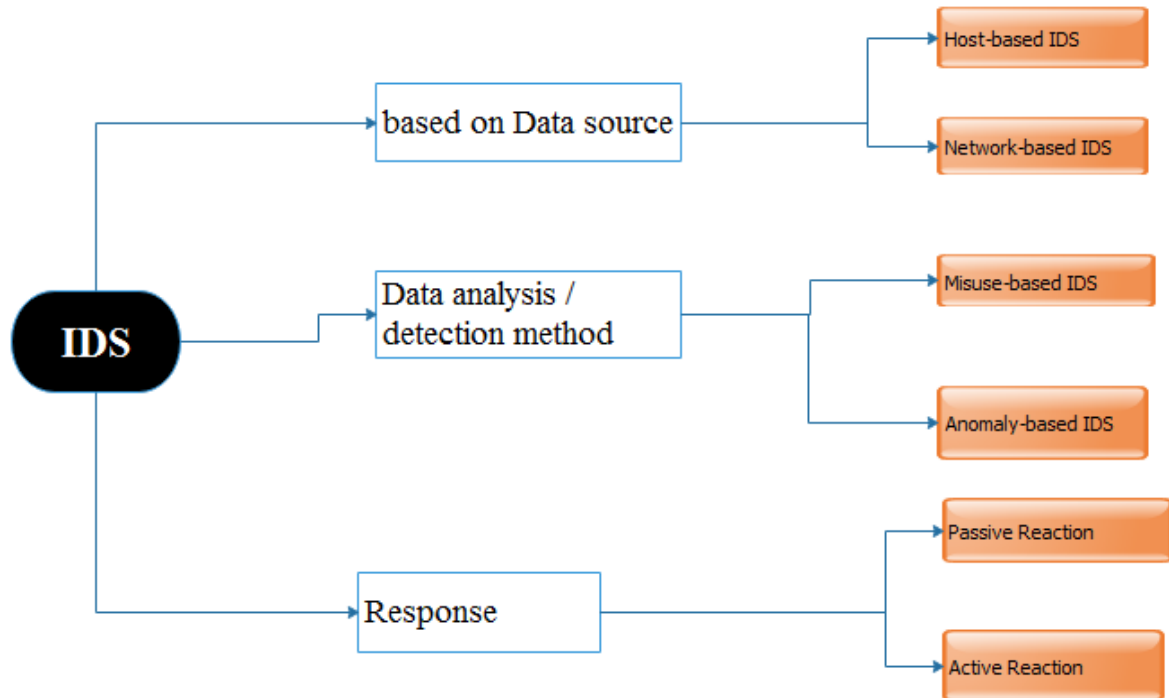


Figure 2. 1: Types of intrusion detection systems [10].

### 2.4.1 Anomaly Based Intrusion Detection System

Intrusion detection system, that expressions at network traffic and detects data that is incorrect (abnormal) is known as anomaly intrusion detection system. This system detects attacks based on irregularities in the pattern through the normal patterns of data on the network, that means it is constructed on the statically analysis of the network data, the flag of all the normal states is set to the normal activity profile of a system and the rest of the situations as anomalous activities [11]. They are detected in several ways, most frequently with machine and deep learning. Systems using these types of techniques have been used to great effect. One of the advantage of this systems is that, it can be very effective at detecting earlier unknown attacks [12]. Figure 2.2 shows, how the anomaly attack can be processed in network intrusion detection.

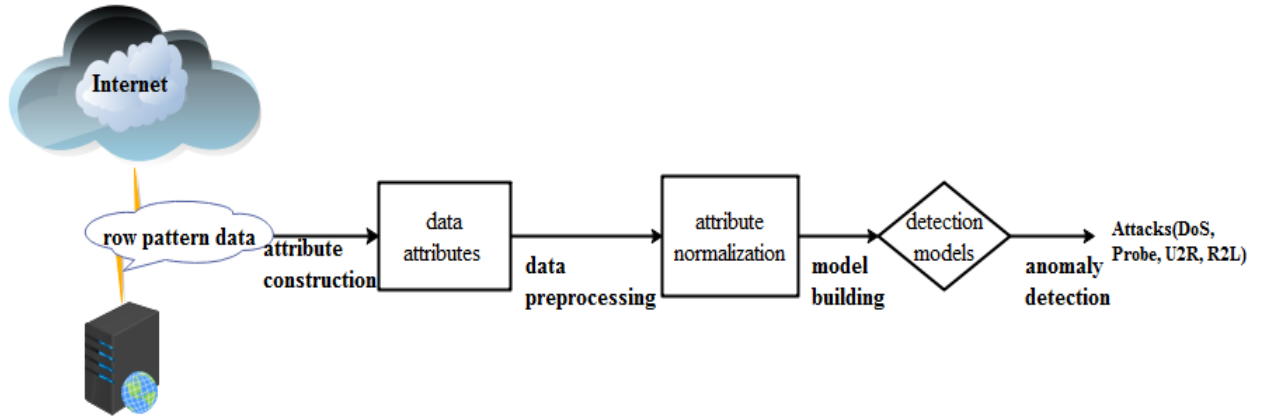


Figure 2. 2: Steps of anomaly intrusion detection system [13].

### 2.4.2 Misuse Based Intrusion Detection System

Misuse or knowledge-based (signature) intrusion detection, is a system based on rules, both preconfigured by way of the device or setup manually through the administrator. The essential idea is presenting the attack in the form of signatures and patterns, these patterns are then maintained as a database of regarded attack situations and signatures. The rules are looking for signatures on network and system operations attempting to trap a well-known attack that ought to be viewed as misuse. The attack signatures are the characteristics associated with successful known attacks.

The advantage is that the method possesses high accuracy in detecting acknowledged attacks. However, its detection capacity is limited by using the signature database. Unless new attacks are changed into signatures and added to the database, misuse-based IDS cannot become aware of any attack of this type, it cannot detect new forms of attack because their signatures are not yet available for pattern matching [14].

### 2.4.3 Host and Network Based Intrusion Detection System

In host based intrusion detection, the system detects intrusion action at host side. It evaluates network traffic and system-specific settings such as: - local safety rule, software calls and local log audits. This is necessary to be set up on each machine and involves configuration exact to that operating system and software. Whereas, network intrusion detection system(NIDS) detects intrusion activity at network side. Normally, NIDS sniffs packets from networks and passes it to detection engine. In comparison with HIDS, NIDS evaluates network traffic at all

layers, Open Systems Interconnection (OSI) model and made decision to purpose of the traffic, exploring for suspicious activity [12]. Figure 2.3 and Figure 2.4 show steps of network and host based intrusion detection systems.

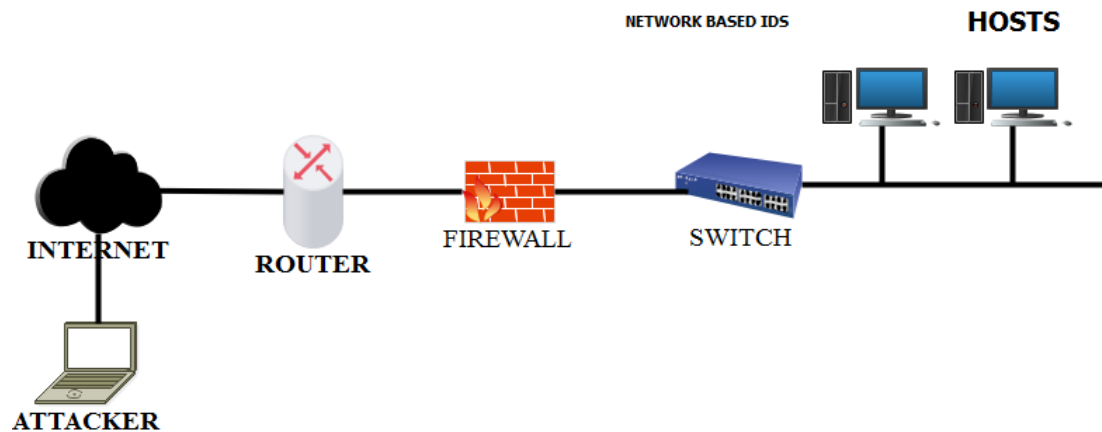


Figure 2. 3: Setup of a network based intrusion detection systems [15].

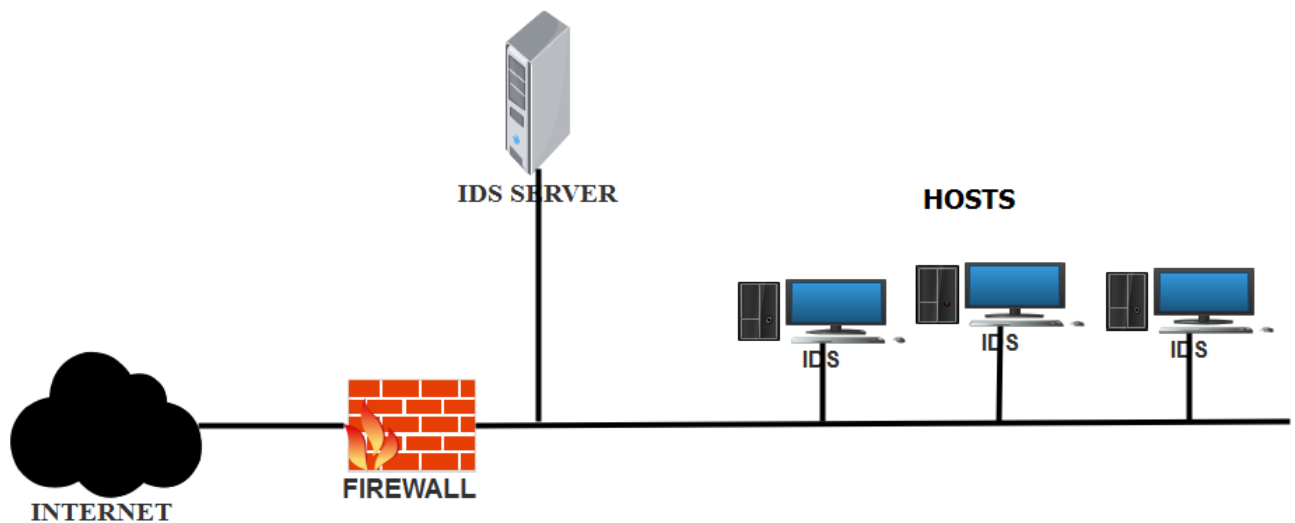


Figure 2. 4: Steps of host based intrusion detection systems [16].

#### 2.4.4 Passive and Active Intrusion Detection System

Active intrusion detection system, is additionally viewed as intrusion detection and avoidance systems. It is to configured automatically block distrusted attacks without any involvement required applying the way of an operator. It has the benefit of presenting real-time helpful action in answer to an attack.

Whereas, a passive IDS is a system is formed to only monitor and study network traffic exercise and alert a worker to possible exposures and attacks. A passive IDS is not successful in performing any defensive or corrective purposes on its own [12].

## **2.5 Machine Learning and Deep Learning**

Machine Learning is a branch of artificial intelligence, which deals with the evaluation and development of systems from the information from the data [17]. It is categorized broadly speaking into three sorts based on the use of labeled and unlabeled data: a) Supervised learning, b) Unsupervised learning, and c) Semi-supervised learning. Some of the extensive range applications of machine learning include regression, classification, prediction and others [18]. Whereas deep learning is a subsection of machine learning. One of the architectures of deep Learning is deep neural networks (DNNs). These DNNs are unknown but a class of artificial neural networks taking various hidden layers as related to standard neural networks. Deep studying algorithms run data by several “layers “of neural network algorithms, every of which passes a simplified illustration of the information to the next layer. Frequently used deep learning algorithms included as deep belief networks, auto-encoders, convolutional neural networks, recurrent neural networks, and recursive neural networks. This all are applied to research fields such as natural language processing, speech recognition, image processing, computer vision, audio recognition, machine translation and network filtering [19].

The main difference between machine and deep learning is the modification in the performance as the scale of the data increases. deep learning algorithms require a larger quantity of data to find the patterns in the network, whereas machine learning requires the less data but both of them are learning from data with different types of features. Moreover, one of the application of machine and deep learning on IDSs is that to filter a social network problem and can aid in answering the tasks with prediction (which is a task of forecasting the next value based on the preceding values) and classification (which is a task of splitting things into different classes). Therefore, in the area of maximum amount of record data and in the situation of cyber security shortage deep learning seems to be one of a solution for today [20].

## **2.6 Literature Review**

This section deal with the output of literature review of intrusion detection systems mainly focused on the machine and deep learning approaches. These approaches mainly focused on a detection case of misuse and anomaly detection.

### **2.6.1 IDSs Based on Machine Learning Methods**

This section presents review of some recent and most related work to the proposed systems, on various machine learning approaches. Saroj Kr, Biswas and Silchar [21] delivered IDS by the usage of machine learning classifiers for the network. Such classifiers are k-NN, DT, NN, SVM and NB, and they used CFS, IGR and PCA for feature selection techniques. Their experiment carried out for using 5-fold cross-validation and they used NSL\_KDD dataset to train and test for the classifiers models. Finally, their experimental results confirmed that K-NN classifier produces higher performance than others and amongst the feature selection methods, the IGR feature selection approach was better than others and CFS was the smallest one to others. Bisyron Wahyudi et. al. [22], address a machine learning-based model in real-time network traffic. This was a SVM model, and used 28 feature on the Knowledge Data Discovery (KDD) train and test dataset for evaluating this model. The experimental result on their proposed model showed that 93.4% average accuracy on the test records for the classification of two-class and 86.8% average accuracy for the classification of a multi-class. Additionally, the model was assessed on the NSL-KDD test dataset and the result of for overall classification accuracy of two-class has 82.6% and multi-class has 83.7% on the test data.

Sushant Kumar Pandey [23] constructed and designed various features selection techniques for anomaly based intrusion detection systems. Following classifier are used to implement the systems: Random Tree, REP Tree, Random Forrest, AdaBoostM1, Meta Tagging, Decision Stump, J48, Bagging and Naive Bayes. The performance of each model are evaluated on the use of NSL-KDDTrain+20% dataset, for binary and multi class classification applied with 10-fold cross-validation on the training data. Author used vote algorithm to function of data filter and information gain for a features selection. Each design models are achieved better accuracy, for both binary category and multiclass. But Naive Bayes are recorded less amount of accuracy among the other.



Nicholas J. Miller and Mehrdad Aliasgari [24], used different machine learning algorithms for evaluating the different attacks on the NSL\_KDD dataset. They used the following algorithms: Naive Bayes, Support Vector Machines, Neural Networks, and K-means Clustering. Whereas, all classifiers are evaluated on the categories of the four attack DoS, Probe, U2R and R2L on test dataset. Based on their experimental observations, Probe and DoS had high accuracy for all approaches. While the attacks of U2R and R2L were much more difficult to detect for all approaches.

Mouhammad Alkasassbeh and Mohammad Almseidin [25] used Knowledge Discovery and Data Mining (KDD) dataset for evaluating the performance of different methods of machine learning classifiers. These techniques are: J48, MLP and Bayes Network classifiers. All approaches achieved the following overall classification results on the training data, for MLP has 91.9017%, for Bayes Network has 90.7317% and for J48 has 93.1083%.

Prakash Chandra et. al. [26] proposed a network intrusion detection system, based on a method of the modified random forest classifier algorithm (MRFA) and CART (regression tree) with bagging approach. From features chosen on the dataset, MRFA as select the best features and built on the decision tree. The approach was implemented by using Java and simulated on the WEKA device and evaluated on KDD-Cup 99 and NSL-KDD data set. And their experimental study clearly showed that, the approach of MRFA performs outstandingly in terms of accuracy, detection rate and F-measure for both data sets, over the existing methods of Naive Bayes, J-48 and Random forest.

### **2.6.2 IDSs Based on Deep Learning Methods**

Many scholarly articles have been published on the topic of detecting intrusion using data and machine learning techniques that are listed in the above section. Although, following section are critical for evaluation of recent research efforts that are related to the proposed system. S. Mohammadi and A. Namadchian [27], proposed DNN-Auto-Encoder for the classification of anomaly intrusion detection system. This technique was to identify and detect attacks correctly to the best features of network connections. They used the memetic algorithm for classification purposes of the features of the attack. The model was evaluated on the NSL-KDD dataset. And they found that an average accuracy of 87.8% all the group of attacks.

Sheraz Naseer et. al. [28] Proposed a deep learning methods for processes of anomaly-based intrusion detection systems, such methods are: convolutional neural networks, auto-encoders, and recurrent neural networks. NSL\_KDD train and (NSL\_KDDTest+ and NSL\_KDDTest21) test data are used to evaluate the network performance. Jupyter development environment using keras 2.0 and Theano backend are used to implement systems models. Additionally, authors compared those models to conventional machine learning such as Decision Tree (J48), Naive Bayes and Random-Forest. They have done and recorded good accuracy performance of two class classification for both deep and conventional learning algorithms. But, CNN and RNN models achieved overall performance accuracy of 85% and 89% on training data, respectively. N. Shone et.al. [29]discussed a novel non-symmetric deep auto-encoder (NDAE), for the unsupervised feature learning classification model. Their proposed system classifier has been carried out on GPU-enabled tensor flow and evaluated with the use of KDD Cup '99 and NSL-KDD training datasets not included test data. For using KDD Cup'99, they have achieved good accuracy result, which has an average accuracy of 97.85% for two class classification. For using NSL\_KDD training dataset, the authors achieved an accuracy result of following attack types, for DoS has 94.58%, for Probe has 94.67% and for R2Land U2R are recorded very less accuracy compared to the others.

Ali H et. al. [30] has pointed out, sequential auto encoder long short term memory (LSTM) neural network for computer network intrusion detection. They used "ISCX IDS2012" public dataset for train and test purposes, in addition, used GRU, Bi-LSTM and feed-forward neural networks to detect the intrusion. Dimensionality reduction and feature extraction process are applied from dataset. They also assigned a threshold value based on cross-validation to classify whether or not the incoming network data sequence was anomalous or not.

Taief Alaa Alamiedy et. al. [31], proposed detection of abnormal intrusion detection for using multi-objective grey wolf optimization algorithm (GWO), The advantage of this algorithm was employed as used a feature selection mechanism to identify the most relevant features on the dataset. NSL-KDD dataset was used to demonstrate for the effectiveness of their approach through a different attack types. Their experimental result showed that and recorded following classification accuracy for the selected features of dataset. For the attack of DoS, Probe, R2L, and U2R has recorded 93.64%, 91.01%, 57.72%, 53.7%, classification accuracy from test data respectively.

M. Ponkarthika and V. R. Saraswathy [32] addressed a deep learning method, to implement a NIDS. The target of this technique was to become aware of the network behavior was normal or affected. Experimental implementation was carried out long short term memory (LSTM) which is applied to a Recurrent Neural Network, and used proper learning rate and hidden layer size. Weights are adjusted randomly trained and tested the model for the usage of KDD Cup dataset, so through the performance-tested, the learning network are fine for NIDS. Simone A. Ludwig [33] pointed out IDSs by using a different ensemble deep machine learning to classify and detect different kinds of attack. Such techniques are an auto-encoder, a deep belief neural network, a deep learning neural network and additionally to build extreme studying machine learning. They used the NSL\_KDD data and examined the models. Final experimental results showed that classification accuracy performed for all attacks were succeeded good accuracy on the deep learning networks.

Farrukh Aslam Khan et. al. [34] proposed a method of two-stage deep learning (TSDL) model and a stacked auto-encoder with a soft-max classifier, for effective network intrusion detection. This model comprised two decision stages. First stage was for classifying network traffic as usual or unusual and the second stage was using as an additional feature, for detecting the normal state and other classes of attacks. They used two public datasets which are KDD99 and UNSW-NB15 datasets. The simulation results validated that the systems meaningfully outperforms and achieved high detection rates of classification accuracy of normal and abnormal, up to 99.996% and 89.134%, for the KDD99 and UNSW-NB15 data sets, respectively.

R. Vinaya Kumar et. al. [35] address a deep learning model. This was to develop flexible and effective IDS to detect and classify cyber-attacks. They used KDDcup99 dataset and applied to the proposed system through different learning rates. The authors additionally used NSL-KDD, UNSW-NB15, Kyoto, WSN-DS and CICIDS 2017 for applied to the model. All the trials are run till 1,000 epochs with the learning rate amount changing in the range [0.01–0.5]. Through rigorous experimental testing, finally it is confirmed that DNNs achieved well in contrast with the classical machine learning classifiers.

Amir Andalib, Vahid Tabataba Vakili [36] proposed the detection of intrusion using ensemble of advanced learners. This system used three learning methods such as: - convolutional neural network, Random Forest as an ensemble technique and gated recurrent unit. NSL-KDD test dataset are used to verify the proficiency of for both models. Simulation results showed that on GRU was achieved overall classification accuracy of 83.17% with 91.73s (second) of training

time, CNN has 82.92% with 104.4s and RF has achieved an average accuracy of 80.14% for an execute time of 16.32s. Bhakti Nandurdikar and Prof. Rupesh Mahajan [37] discussed the difficulties handled by existing NIDS techniques. To improve this problem, they have proposed a novel NDAE (novel deep auto encoder) technique for unsupervised feature learning and to implement the intrusion avoidance system. The performance assessment of the model used was KDD CUP dataset. The result showed, high levels of accuracy, precision and recall together with reduced training time. This improved only 5% accuracy for the existing systems.

Félix Iglesias and Tanja Zseby [38] suggested and give a solution that one of the predominant issues of machine learning during training, which was the feature size in the records. Many machine learning models use thousands of features for training, this may yield slow training, greater over fitting and for the curse of dimensionality, feature selection acts to solve these problems. Removing pointless and trivial features reduces the training time and increases accuracy and other evaluation metrics. Thus, Ghadah Aldehim and Wenjia Wang [39] attempts to figure out, how to use the data set in feature selection methods. And Najafabadi et al [40] also address feature choice technique of ANOVA test for measuring the importance of feature selection process for data preprocessing.

Yanqing Yang et. al. [41], proposed an improved conditional variational auto-encoder (ICVAE) model with a deep neural network (DNN) for intrusion detection systems, one of the points of this model was the detection of different attacks on the IDS dataset. Which are NSL-KDD and UNSW-NB15 datasets. The experimental result showed that for the usage of NSL\_KDDTest+ dataset achieved an overall classification accuracy of 85.97% and for NSL\_KDD Test-21 has 75.43% and for using UNSW-NB15 data achieved an overall accuracy of 89.08%, the model also shows better accuracy for the attack categories on the NSL\_KDDTest+ dataset which has, 74.97% ,85.65% ,11.00% and 44.41% accuracy for the attack of Probe, DoS, U2R and R2L, respectively.

Divyasree T.H and Sherly K.K [42] proposed and used ensemble core vector machine(CVM), to classify and detect the attacks of U2R, R2L, Probe and DoS. KDDCup99 dataset was used for evaluation of the model. Chi-square algorithm was used for selecting the relevant features for each class of attacks on the dataset. Their experimental results achieved the classification accuracy, for DoS has 99.05%, for Probe has 94.50%, for R2L has 76.41%, and for U2R was recorded has 93.71%. Kunal Singh and Dr.K. James Matha [43] focused on Performance comparison of the detection of intrusion between existing DBN Algorithm and proposed state

preserving extreme learning machine (SPELM) Algorithm. For each models the NSL\_KDD dataset are used to training and testing of the models. Their experiment was performed by compared the Accuracy and Computational time of the dataset on the model. The finding performance of SPELM has better when compared to its existing system, an average accuracy of 93.20% as SPELM against 52.8% of the DBN algorithm with 90.8 seconds of computational time taken by SPELM as against 102 seconds DBN Algorithm. For each attacks their proposed model succeeded the following results, DoS has achieved 95%, probe 92%, R2L 92%, and U2R 93%.

Afreen bhumgara and Anand pitale [44], the author proposed hybrid approaches that combine some techniques like J48 Decision Tree, Support Vector Machine and Naïve Bayesian for detection of various varieties of attacks. These methods tests passed off on NSL-KDD training dataset and a classification accuracy according to the algorithms are followed. For using SVM for attacks U2R, R2L, Probe, and DoS have succeeded an accuracy of 93.4%, 93.7%, 97.5%, 97.1%, respectively. And also for using Naïve Bayesian the completed accuracy has 71.1% for U2R, 69.9% for R2L, 74.2% for Probe and 73.9 to the attack of DoS. However, for using J48 all attack groups are recorded a maximum accuracy.

Ritumbhra uikey and Manari cyanchandani [45] proposed different machine learning methods applied to intrusion detection systems. The different classifiers are: - deep convolution neural network (DCNN), random forest (RF), and naïve bayesian (NB), to the classification detection of different kinds of attacks on KDD99 and NSL-KDD dataset. And the results of each models are as follows: For the use of naïve bayesian, the achieved results are 45%, 31%, 61%, and 59 % to the attack categories of U2R, R2L, Probe, and DoS, respectively. For using DCNN the classification accuracy of U2R achieved 67%, R2L achieved 86.55%, Probe achieved 93.20% and DoS achieved high, which value has 97.44%. For RF algorithm their experimental result shows an accuracy of 95.48%, 52.17%, 99.98%, and 98.43% for the attack categories of U2R, R2L, Probe, and DoS, respectively.

Jin Kim et. al. [46] proposed deep neural network for intrusion detection system. KDD Cup 99 dataset was used for evaluation of the model. Using cross validation methods to separate the training data in to form of training and testing, this was for evaluation of the model. Data preprocessing is also applied for the dataset and to transform the processed data to the model. Training and testing classification accuracy are the evaluation method and used a four-layer DNN with 100 units. Finally, they reported very high accuracy values from the design model,

which was 99% accuracy. In table 2.1 below summarizes the previous works that are most related with the subject of this study.

Table 2. 1: Summary of different related works.

Authors name and year	Methods(algorithms used )	Dataset	Accuracy(acc%)
Divyasree T.H and Sherly K.K [38], 2018.	Core Vector machine(CVM) with Chi-square features selection	KDDCup99	DoS=99.05%,Probe = 94.50%, R2L=76.41%, U2R = 93.1%
Bisyron Wahyudi et. al. [18], 2018.	Support vector machine (SVM), used 28 input features on the dataset, and classify the attack in to multi class and two class classification.	>>	Obtained average acc of = 83.7 % of multiclass and 82.6% of two class classification
Yanqing Yang et. al.[37], 2019.	Improved Conditional Variational Auto-encoder (ICVAE) model with a deep neural network (DNN.	NSL_KDD	DoS = 85.65%, Probe = 74.97% R2L = 44.41%, U2R = 11.00%
Ritumbhra uikey and Manari cyanchandani [41], 2019.	Deep Convolution Neural Network (DCNN), Random Forest (RF), and Naïve Bayesian (NB) to classify and detect attack	>>	For DCNN:- DoS = 97.44%, Probe = 93.20%, U2R = 67%, R2L = 86.55%.
Afreen bhumgara and Anand pitale [40], 2019	J48 Decision Tree, SVM and NB, to detection the attacks of DoS, Probe, U2R and R2L.	>>	For SVM: - U2R = 93.4%, R2L = 93.7%, Probe = 93.7%, and DoS= 97.1%. For NB:- 71.1% for U2R, 69.9% for R2L, 74.2% for Probe and 73.9%to the attack of DoS
Amir Andalib, Vahid Tabataba Vakili [36] , 2020	CNN, Auto-encoders, GRU and RF, to detect the attack on the dataset.	>>	Overall performance accuracy for CNN= 82.92%, GRU = 83.17% and RF 80.14%
Jin Kim et. al. [46], 2017	Deep neural network (DNN). Across validation methods was used to detect intrusion attacks	KDD Cup 99 train	Achieved very high accuracy which is 99.99% on the dataset.

Problem domain from the review: Following are some concluded problems taken from various papers are shown here:

In[38]	Feature selection reducing the accuracy.
In[37]	Adjusting model parameters and trainable parameters are a problem of to decrease performance accuracy of attacks.
In [46]	Some of researcher used only training datasets for training and testing of their proposed models, and reported very high accuracy rates .
All	Less ability to detect the attacks (low Performance for detecting attacks, especially for U2R and R2L attacks).

Based on the research works reviewed we have summarized all IDSs to identify their gaps and to suggest a solution that enhance them. A lot of research works have been done which can be applied to intrusion detection using machine and deep learning methods. But the related works we reviewed showed the following issues. Lack of comparison of variant RNN learning models themselves for a multi class classification. Feature selection reducing the accuracy of U2R and R2L attacks. In addition, less accuracy of U2R and R2L attacks in IDS remains one of the major issues for both conventional and deep learning methods. Some of the researcher used only training datasets for training and testing of their proposed models, using cross-validation mechanisms. Most of the recent works followed this approach and reported very high accuracy rates, example, Kim et al. [42], used a 4-layer DNN and with 100 units for detection of intrusion on the KDD99 training dataset and, they informed 99% accuracy. We believe that this approach does not provide a reliable solution of anomaly detection issues, as given sufficient training, models can be over-fitted to achieve such high rates.

In addition, most of the works have been done that can to detect known attacks (classification of network traffic into normal or abnormal) which was signature based approach. Based on this observation, we select an anomaly-based network IDS using three types of recurrent neural network to enhance classification performance of anomaly intrusion detection, and also to solve classification accuracy of R2L and U2R attacks by using a feature selection mechanism. But to fill last issue, we chosen to train all models on training dataset without ever showing test dataset to the model during training and then tested/evaluated the models on testing dataset.

## **CHAPTER THREE: METHODOLOGY AND MATERIAL USED**

### **3.1 Overview**

The methodology of the proposed systems are as follows: first mechanism to do is to conduct a comprehensive review of the literature to acquire a deeper understanding of the research area and its problem domains this is described in chapter two. The second thing to do is to look for data collection and the datasets to learn about patterns that can identify different categories of attacks from normal on the dataset. Based on the proposed solution from the identified problem in literature review we have to select tools, techniques and hyper parameters. After identifying those requirements to implement systems. Then finally, to evaluate the system using metrics. Following section are briefly explain overall methodology is based on proposed systems including: Proposed NAIDSs, overview of systems architectures, type and structure of implementation algorithms, description and preparation of the dataset which is the NSL\_KDD+ train and test dataset, data preprocessing and feature selection are define. The final section of this chapter also explains the hyper parameters setting methods and the evaluation metrics of the research output of proposed work.

### **3.2 A Overview of Systems Architecture**

The systems architecture shown in Fig 3.1 presents basic architecture of an incremental network based abnormal detection methods. Most necessary phase of this architecture is the anomaly detection systems, since it detects abnormal behaviors and save the alerts in a computer as records. And copy of this alert record is send to the network administrator “network security analysts” thereafter, recognize actions are taken, via updating the database of the protection systems on the network. The system contains of the following five parts.

1. Router/ firewall, it is used for internet packets routing.
2. Switch-with promiscuous mode which is used for packets switching, and traffic sniffing for classification.
3. The NADS that are built to capture the packets, pre-processing, classify attacks and send alert reports.
4. A security analyst is to take suitable action in regards to the alerts, and additionally to check if the alert is proper or not and finally update the systems.
5. LAN, it contains four clients.



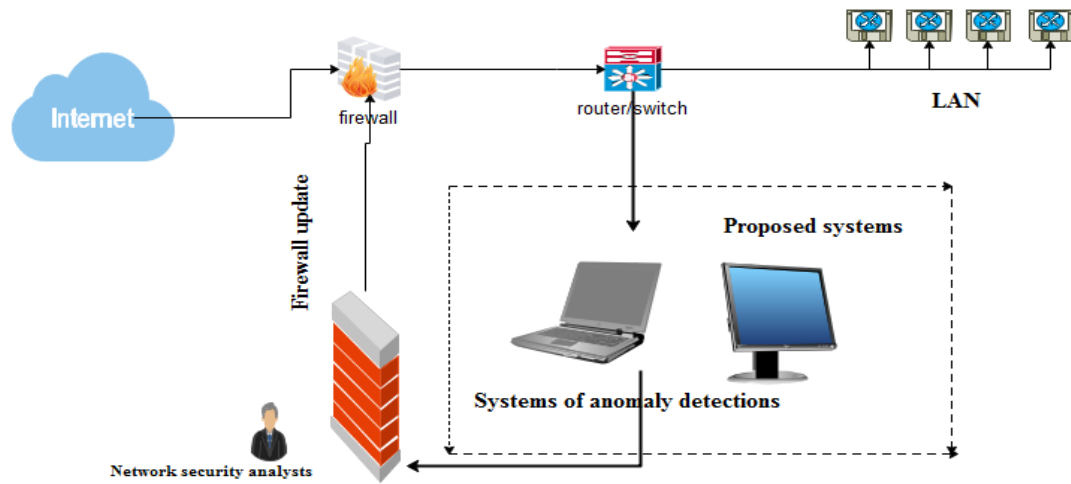


Figure 3. 1: Overview of NAIDSs architecture.

The system contains of the following five parts.

6. Router/ firewall, it is used for internet packets routing.
7. Switch-with promiscuous mode which is used for packets switching, and traffic sniffing for classification.
8. The NADS that are built to capture the packets, pre-processing, classify attacks and send alert reports.
9. A security analyst is to take suitable action in regards to the alerts, and additionally to check if the alert is proper or not and finally update the systems.
10. LAN, it contains four clients.

There for one of the main advantage of this study is to design the anomaly IDSs, this is how to classify and identify unknown attack to normal one on the intrusion detection dataset, thus the method of this system work is proposed in figure 3.2 as shown below, this proposed work has different types of components. Data collection (NSL\_KDD dataset), Data preprocessing, train-test split, model building (SimpleRNN, LSTM, GRU and hyper parameter choosing or setting), evaluation metrics. The data preprocessing phase which consists of sub-tasks: non-numeric data to convert to numeric form, label encoding and one hot encoding, in this case the dataset has nonnumeric values, so this values are must be to convert into 0s and 1s form, normalization for standardizing range of values, feature selection to select relevant features. In the train-test

split phase the whole preprocessed data is split into training set (for training machine learning algorithms) and testing set (for evaluating models performance). That means the training and testing data are classify in to group of attacks. The purpose of proposed systems is classifying and detecting the network behavior in the categories of attack (DoS, Probe, U2R and R2L) on the NSL\_KDD training and testing dataset. A detailed description of each component of the system architecture is presented in the following sections. A detailed description of each component of the system architecture is presented in the following sections.

### **3.3 Data Collection and Preparation**

To build an intrusion detection classification model, a huge amount of network traffic data is needed to provide enough data to the classifiers to train them properly. For this reason, a publicly available network intrusion detection dataset is collected from the Web [47]. Even though there are many publicly available datasets for simulating network intrusions, all of them may not be helpful for developing effective IDSs. For example, the KDD CUP 99 dataset which is available at [48] is the furthestmost widely used and freely available network. Furthermore, it can cause issues such as long training time, poor accuracy rate and low detection rate of a model. Even though this dataset is still used, for the assessment of network IDSs by many researchers, the evaluation results acquired by this dataset are less accurate. To overcome this, we used the latest dataset called NSL-KDD dataset. This dataset is prepared by the University of New Brunswick which is the revised version of the KDD CUP 99 dataset introduced for effective and robust evaluation of machine learning-based IDSs.

This dataset contains information corresponding to common types of attacks (denial of service attack, probe attack, a user to root attack, and remote to local attack) [49], the main reason for selecting this dataset are as follows: -

- They are standardized dataset and the experimental results can be achieved for IDS on deep learning approaches.
- It consists of 41 features and a wide variety of attacks which is difficult to find in any other dataset.
- Attributes are numeric and categorical so we have to figure out, how to load and handle data and fits in deep learning algorithm(memory).

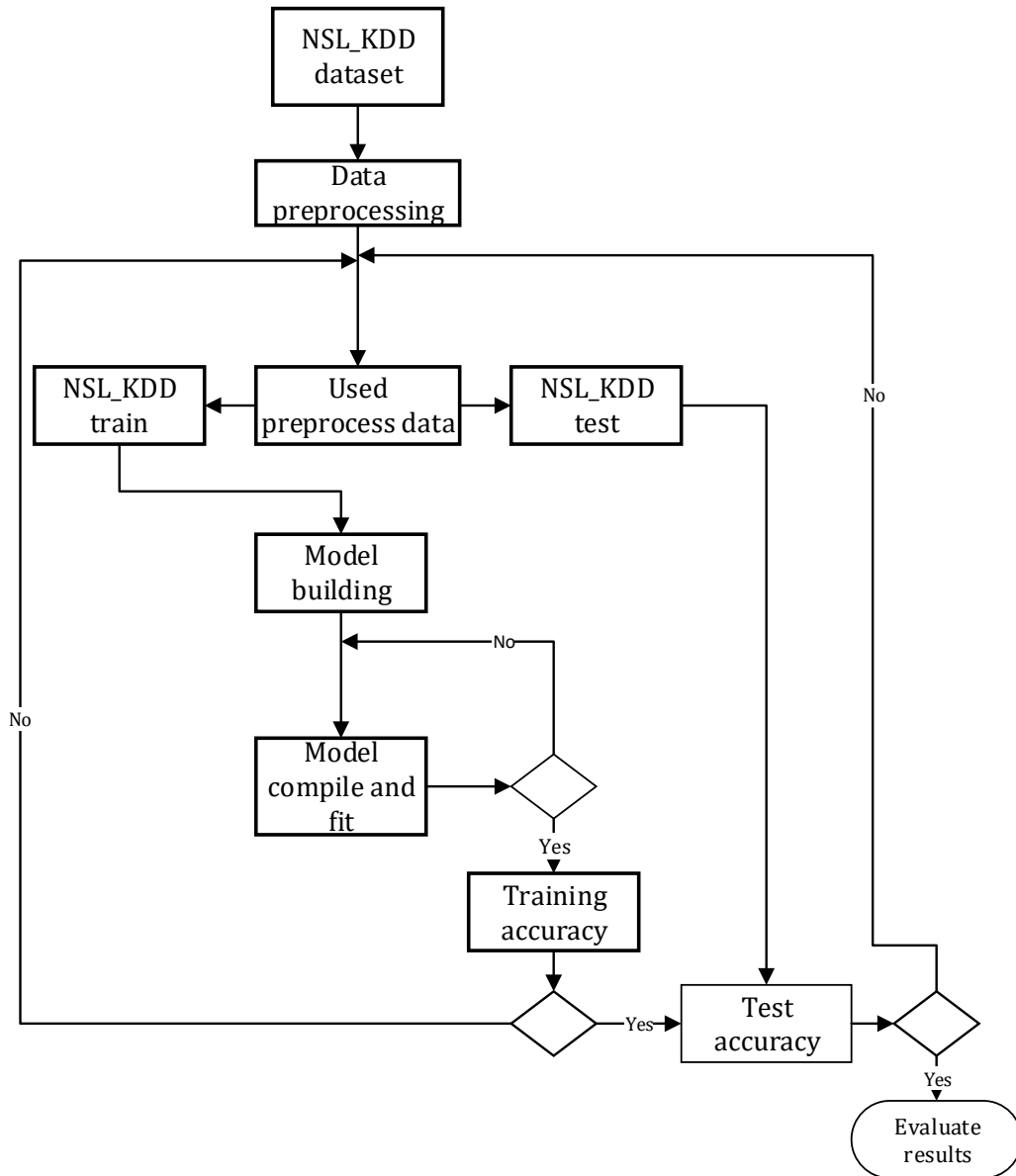


Figure 3. 2: Proposed system architecture.

- NSL\_KDD datasets are still the most comprehensive and widely used for researchers on the literature review, for comparison of different machine learning algorithms for the ability to detect and classification of the attacks.

However, the major limitation of this dataset is the presence of Not a number (NaN) (there are some values in the records that is not numbers) values and irrelevant records in the training and testing set and using it will reduce the quality of system performance accuracy.

The NSL-KDD dataset contains 42 features per record, with 41 of the features referring to the traffic input and the last feature (column) specifies the target class see Appendix A [50]. It is

available in 4 partitions, two partitions namely NSL\_KDDTrain20%+ and NSL\_KDDTrain+ serve as training data for models and supply 25,192 and 125,973 training records respectively. And other partitions are NSL\_KDDTest+ and NSLKDDTest-21 are available for performance assessment of trained models on unseen data and provide 22,544 and 11,850 records instances respectively. The target class has five categories that are one normal and four attack categories (DoS, Probe, R2L and U2R). In addition, the dataset holds a total of 24 training attack types and with an addition of 14 new attack types in the test data. Among the 41 attributes, 32 are numeric, 6 are binary and 3 are nominal data. Features 2, 3 and 4 are nominal(symbolic), features 7, 12, 14, 15, 21, and 22 are binary(Continuous) and the rest features are numeric types see Appendix A.

In this study we have chosen and used NSL\_KDD+ train and NSL\_KDD+ test data, since for evaluating deep learning models, maximum values of data must be used. So the chosen data have maximum recorded values compare to the other two and that is enough to train and test for our proposed approaches.

### 3.5 Dataset Attack Types

In multi-class classification is the problem of classifying instances into one of three or more classes. For this purpose, the feature attack-category has been introduced among NSL\_KDD dataset, features to categorize the records into specific classes and the description of each of them are the following:

#### 3.5.1 Denial of Service (DoS) Attacks

In this type of attack class, an attacker prevents the legitimate users from using services on the network, by overwhelming or flooding the systems with request and consuming the resources by exploiting system misconfigurations or by aiming the implementation bugs. Types of DoS attacks on the NSL\_KDD dataset are: *Back attack*, *Land attack*, *Neptune attack*, *Pod*, *Smurf attack*, *Teardrop attack*, *Apache2 attack*, *Udpstorm*, *Processtable*, and *Worm* [51], [52], [53].

- Land: - Access the server with same IP address and Port number, and beat down the server.
- Neptune: - Send only a SYN packet to the victim then do not establish 3-hand shaking. Beat down the waiting queue.
- Pod: - Send a lot of ping (ICMP) and beat down the server.

- Smurf: -Attackers use ICMP repeat application packets fixed to IP broadcast addresses from remote locations.
- Teardrop: - Exploits a flaw in the operation of elder TCP/IP stacks.

### 3.5.2 User to Root (U2R) Attacks

In this attacks class, an attacker first hack into a common user account on the network and then exploits vulnerabilities, so that he can increase source access of the systems. Regular buffer overflow is the most usual exploit in this attack class, and the reason may be programed mistakes or environmental assumptions. Types of this attacks are: *Buffer\_overflow attack*, *Loadmodule attack*, *Rootkit attack*, *Perl attack*, *Sql*, *Xterm*, and *Ps* [47, 48, and 49].

- Buffer\_overflow: -Pour the data more than buffer size. And modify the data in buffer.
- Loadmodule: -Load certain kernel driver module dynamically for acquiring root authority.
- Rootkit: - The program that hide the backdoor.

### 3.5.3 Remote to Local (R2L) Attacks

It is a classification of attacks, where as an attacker sends packets to a computer on the network, then exploits its vulnerabilities and illegally gains local access as a user. This means that unauthorized access from a remote computer, the attacker intrudes into a remote computer and gains local access to the victim machine or computer. Example of such attack can be password guessing and relevant features are: Network level features - “duration of connections” and “service requested” and host level features “number of failed login attempts”. The remote to local attacks types are: *Guess\_Password attack*, *Warezclient attack*, *Spy attack*, *Xlock attack*, *Xsnoop*, *Snmppguess*, *Snmppget attack*, *Ftp\_write*, *Imap*, *Phf*, *Multihop*, *Warezmaster*, *Httpunnel*, *Sendmail*, *Named* [47, 48, 49].

- Ftp\_write: - Overwrite or write another file to the ftp server in order to drive client to receive wrong file from the server.
- Guess\_password: -Guessing password by brute force or dictionary attack.
- Imap:-Injecting imap protocol command to e-mail server for sending commands.
- Phf: - Operating shell command by using phf cgi.

- Warezclicent: - Make users download the illegal warez software.
- Warezmaster: - Exploits a system bug associated with an FTP server. Etc...

### 3.5.4 Probe Attacks

In this class of attack, the attacker scans a network or host to gather known vulnerabilities and information about the host computer. The main purpose of this type of attacks is to improvement information about the remote victim. Example of such attack is port scanning. Relevant features of this type are: “duration of connection” and “source bytes”. Types of probe attacks are: *Satan attack*, *Ipsweep attack*, *Nmap attack*, *PortswEEP*, *Mscan* and *Saint* [47, 48, and 49]. Little of them are define below:

- Ipsweep: - Send ping to certain group of network and waiting for response. Distinguish host is alive.
- Nmap: -Network vulnerability scanning using nmap.
- PortswEEP: -Checking which port is open, or what kind of program is running on the system.
- Satan: - Network vulnerability scanning using satan tool.

## 3.6 Data Preprocessing

Among the crucial steps in the data-driven approach (machine learning) is data pre-processing. Data preprocessing should be used, which is applied to our data before feeding it to the system [54]. Usually, the collected data is not suitable or prepared to be used for a machine learning tasks. Hence, data pre-processing (also called data preparation) is preferable or sometimes is necessary to achieve better result. Since to achieve accurate and efficient results. Dataset should be free from unnecessary value from the collected data. In general, when the following proprieties are present in our data, we have to consider data pre-processing before doing a machine learning task such as:

- Missing values: happens when no data value is stored for the feature in a record.
- Nominal features: also known as categorical features. Typically, it means any feature which is definite in nature characterizes separate values which belong to exact finite set of classes (i.e. no numerical data).
- Non similar scale features: which means that we have to create sure that all features take the same ranges of values. For instance, the values range of the feature x1 is [0 –

10000] and the feature x2 takes on values between [0 – 5]. In this case we consider to normalize the data to make the features scale at the same range.

NSL-KDD network dataset has missing values records. And also it has some nominal feature and furthermore its features are not at the same values range. To solve these problems, we have to used following techniques called data cleaning, Numericalization (label encoding), One-Hot Encoding and Feature scaling (also called data normalization) respectively.

### 3.6.1 Data Cleaning

In this step imputation technique was applied on the dataset, that means to replace missing data with substituted values. Since, there are literally no values in the data set so that it will not have an effect, but makes it difficult for the algorithm to be train. NSL-KDD dataset has NAN values, so the NAN values cannot be processed by learning algorithms, these will either have to be converted into values or removed. Therefore, NAN values are substituted by 0 value of both training and testing data using the program of #df. fill nan (0) and #df\_test. fill nan (0) on the jupyter notebook.

Table 3. 1: NAN values in the NSL\_KDD dataset.

```
In [24]: df.head(5)
```

```
Out[24]:
```

		duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...
0	0.tcp,ftp_data,SF,491.0,0,0,0,0,0,0,0,0,0,0,0,...		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
1	0.udp,other,SF,146.0,0,0,0,0,0,0,0,0,0,0,0,0...		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
2	0.tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0...		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
3	0.tcp,http,SF,232.8153,0,0,0,0,0,1,0,0,0,0,0...		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
4	0.tcp,http,SF,199.420,0,0,0,0,0,1,0,0,0,0,0...		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

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	c
NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	
NaN	NaN	NaN	NaN	NaN	

dst_host_srv_diff_host_rate	dst_host_error_rate	dst_host_srv_error_rate	dst_host_error_rate	dst_host_srv_error_rate	label
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN

### 3.6.2 Label Encoding

It is a popular encoding procedure for control categorical features, in this technique, each label is allocated a unique integer created on alphabetical collection. That means the features are transformed from categorical to number [55].

For the NSL\_KDD dataset there are, 38 numeric features and three non-numeric features, so we should be converting the non-numeric features into number form, for example the features of 'protocol\_type' are: - ('tcp','udp', 'icmp') this is non-numeric form, so to encode into number form (1,2,3). We used the same method to encode the feature (service) that includes 70 service types, meaning this feature changed from 1 to 70. In the same way, the feature (flag) changed from 1 to 11. Therefore, the entire features of the dataset became 122 features (38 numeric + 3 protocol\_types + 70 services+ 11 flags = 122). Features of services and flags are documented in Appendix A. But following are sample label encoding of categorical features of these three nonnumeric features on the dataset.

Table 3. 2: Before label Encoding.

Out[42]:

	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0
3	tcp	http	SF
4	tcp	http	SF
5	tcp	private	REJ
6	tcp	private	S0
7	tcp	private	S0
8	tcp	remote_job	S0
9	tcp	private	S0

Table 3. 3 : After label Encoding.

	protocol_type	service	flag
0	1	20	9
1	2	44	9
2	1	49	5
3	1	24	9
4	1	24	9
5	1	49	1
6	1	49	5
7	1	49	5
8	1	51	5
9	1	49	5

Attack names, such as smurf, portsweep, guess\_passwd and Buffer\_overflow, were mapped to one of the classes: Normal to 0, DoS to 1, Probing to 2, U2R to 3 and R2L to 4 [56].



### 3.6.3 One-Hot Encoding

It is a technique used to change a categorical data into a numerical form. The problem with categorical data is that some machine learning algorithms cannot work with them such as logistic regression, support vector machine and neural network. . . etc. They only understand numerical or continuous data. One-hot encoding transforms each category into a feature where all new created features can be perceived as a vector of binary values. The record that has a such category, value one(1) is placed into the feature that represents the such category and the rest are 0 [57]. As mentioned above, NSL\_KDD dataset has 4 categorical features (protocol, service, flag and attack-category) that they have been encoded using one-hot encoding system. Therefore, the non-numeric values of protocol ('tcp','udp', 'icmp'), and its numeric values are encoded as binary vectors (1.0,0.0,0.0), (0.0,1.0,0.0) and (0.0,0.0,1.0). Similarly, the feature ‘service’ has 70 types of attributes, and the feature ‘flag’ has 11 types of attributes were mapped to integer values range from 0 to N-1, where N is the number of symbols in the recorded data. Following captured tables are representing the sample one hot encoding systems of all categorical features on the dataset.

Table 3.4: One- hot encoding of non-categorical features of NSL\_KDD dataset.

Out[158]:

	Protocol_type_icmp	Protocol_type_tcp	Protocol_type_udp	service_IRC	service_X11	service_Z39_50	service_aol	service_auth	service_bgp	service_courier
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

service_courier	...	flag_REJ	flag_RSTO	flag_RSTOS0	flag_RSTOR	flag_S0	flag_S1	flag_S2	flag_S3	flag_SF	flag_SH
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

### 3.6.4 Feature Scaling(Normalization)

Various machine learning systems achieve better or faster when features are on relatively similar scale. A dataset may contain many features that have values in different ranges. Some features may consist small floating-point values and others large integer values. Besides, some ML algorithms such as NNs, RNN cannot take large values (values which are larger than the initial values taken by the weights of a network). Therefore, to make the network learn easily,

the data fed to the algorithm should take small values (in the range 0-1), all features should take values in the same range, and should be normalized each feature independently to have a mean of 0 and, standard deviation of 1. In this research, a Standard-Scaler technique is used to standardize the features [58]. The reason why we select this normalization technique is because it normalizes most of the data to a small interval than the other standardizers such as Min-MaxScaler, Robust-Scaler, and Normalizer. The mean and standard deviation are calculated for the feature and then the feature is scaled, its general formula is as follow:

$$X_{\text{stand}} = \frac{x_i - \text{mean}(x)}{\text{stdev}(x)} \text{ or } \frac{X - \mu}{\sigma} \quad (3.1)$$

For using the above techniques following features are scaled like as:

- A symbolic feature (protocol, service and flag), to convert these features first to numerical value, and each of the mapped feature was linearly scaled to [0, 1] range.
- Features having lesser integer value like: - wrong fragment [0, 3], urgent [0, 14], hot [0, 101], num\_failed\_logins [0, 5], num compromised [0, 9] etc..., were also scaled linearly to [0, 1] range.
- Features having very large integer values, Logarithmic scaling (base 10) was realistic to two features has a very large integer range: - src\_bytes [0, 1.3 billion] and dts\_bytes [0, 1.3 billion], to reduce the range to [0, 9.14], then scaled to [0,1].
- All other features having values (0 or 1), no scaling was essential for these features.

### 3.6.5 Feature Selection

In machine learning, feature selection was very important things in the case where learning dataset is having too many features. Having too many inappropriate features in the data can reduce the performance of the models. Therefore, feature selection is a solution to solve these issues. Since, it decreases the number of features brings about palpable results on applications for: speeding up learning algorithm during training and reduces processing time as well as complexity of the model [59]. However, while in the selection of feature reduction technique, topmost care should be taken so that it does not reduce the accuracy of the model. So that in intrusion detection system, all of the features might not be useful for IDS to learn the models. In machine learning there are three general classes of feature selection mechanisms that can be used to learn and evaluate machine learning algorithms [60], such methods are called filter method, wrapper method and embedded method. Among this three, the filter method is chosen

and used for feature selection of this study. Since, in filter method features are selected independently from any machine algorithms. It uses a specific criterion, such as scores in statistical tests and variances, and to rank the importance of individual features from the data, it has also some advantages among the other methods [61]:

- Because of their independence in the selection of machine learning algorithms, they can be used as the input of any machine learning models.
- They have a very low computation time and will not over fit the data and they are very fast.

In this study, selected best features of the proposed datasets are used by using Scikit-learn python library. This is to select top most features of the data using SelectKBest and ANOVAF-score\_function value as a filter method [34,35,36].

- *score\_func*: the filter function that is used for feature selection.
- *k*: the number of top features to select.

### 3.7 Algorithms Implementation

To implement the classification and detection techniques of proposed work, three types of RNN algorithms are used, these are Simple RNN, LSTM, and GRU. The effectiveness of these three RNN algorithms are applied on the NSL\_KDD train and test dataset and then evaluated on the categories of attacks DoS, Probe, U2R and R2L. Following are brief explanation of these three algorithm with the help of corresponding network architectures and equations.

#### 3.7.1 Basic Concept and Architecture of (RNN) Network

According to [62], simple RNN is type of recurrent neural networks, which is not have any gates. They only have hidden layers and input at a specific time step. The input is increased by the hidden layer which is also the output from previous RNN cell. It has cyclic connections building them powerful for modeling sequences. Basic architecture and training of RNNs are a system of neuron-like nodes structured into consecutive "layers". Thus, each node in a given layer is linked with one-way linking, to each other node in the successive layer. Each neuron consumes a time-varying real-valued motivation. Each linking (synapse) has a adjustable real-valued weight. Nodes are either: -

- Input nodes: getting data from external of the network.
- Output nodes: yielding results.
- Hidden nodes: modify the information in way from input to output.

As it can be seen in Figure 3.3, simple RNNs only have a single vector that functions as the hidden state and the cell's output. **A** looks at some input and outputs of a value. **A** loop allows information to be delivered from one step of the network to the next. The vector of values at the output of the hidden layer that is observed at time  $t$ , is the currently hidden state vector which was output value,  $\mathbf{h}_t$  it is calculated by the following function as shown below and the vector of the previous hidden state  $\mathbf{h}_{t-1}$ , **W** (weight matrix for the cell's input), **U** (weight matrix for the hidden value is the input of this cell), and  $\sigma$  is a sigmoid(activation function) used to squash its inputs into the  $[0, 1]$  range:

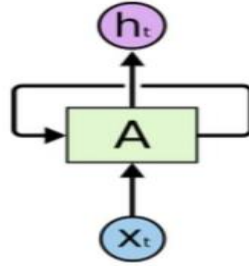


Figure 3. 3: Basic loop structure in RNN [63].

$$h_t = \sigma(\mathbf{W}x_t + \mathbf{U}h_{t-1}) \quad (3.2)$$

### 3.7.2 Basic Concept and Architecture of (LSTM) Network

Long Short Term Memory networks [64], are type of recurrent neural networks with memory cells, these memory cells are the essential part in handling long term dependences in the data, this is it capable of learning long-term dependencies and it explicitly designed to avoid the long-term dependency problem, the LSTM [65], contains three gates architectures unit, these architectures are composed of a cell(memory part) and three "regulators", usually called gates, these are: an input gate, an output gate and a forget gate. These gates are functionally used to manage how information flows into or out of the LSTM cell unit, the following diagram illustrates the transformations that are applied to the hidden state at time step  $t$ :

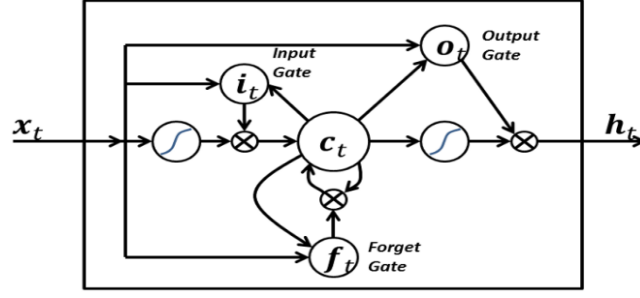


Figure 3. 4: Long short term memory architecture [66].

The figure looks complex let us look at it section by section.  $i$ ,  $f$ ,  $o$ , and  $c$  are gates in which the mechanism by which the LSTM works.

They are computed using the same equations but with different parameter matrices. During training, the LSTM learns the parameters for these gates. In mandate to gain a deeper understanding of how these gates modulate the LSTM's hidden state, let us consider the equations that shown below how it calculates the hidden state  $h_t$  at time  $t$  from the hidden state  $h_{t-1}$  at the previous time step: LSTM calculates a hidden states  $h_t$  as follows [67].

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \quad (3.3)$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \quad (3.4)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \quad (3.5)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \check{C}_t) \quad (3.6)$$

$$h_t = \sigma(C_t) * o_t \quad (3.7)$$

- $\sigma$  is modulating the output of these gates between zero and one.
- $W$  is the recurrent connection at, the previous and current hidden layer.
- $U$  (weight matrix) connecting the inputs to the current hidden layer.
- $C$  is the internal memory of the unit.
- $\check{C}$  is a “candidate” hidden state that is computed based on the present input and the previous hidden state.
- Forget gate( $f$ ): it's control on determining when, to remember what was figured within the previous time.

- Input gate (i): it's control on deciding, when to let the input enter the neuron.
- Output gate (o): it's control on determining when to let the output permit away to the next time.

### 3.7.3 Basic Concept and Architecture of (GRU) Network

GRU are another type of RNNs with memory cells. They real most like LSTM but with simpler cell architecture [68]. this is better than of LSTMs that share many of the same properties, instead of input, forget, and output gates within the LSTM cell. There are two cell or gates, an update gate(z), and a reset gate (r).

The update gate defined, how much previous memory to stay around and also the reset gate expresses a way to chain new input with the previous memory. For several applications, GRU has performance kind of like the LSTM, but being simpler means fewer weights, and faster execution train time [69]. The gates for a GRU cell is showed in Figure 3.5 and Equations leading the processes of GRU architecture are given below:

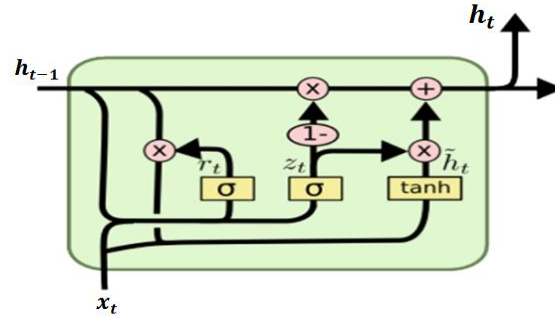


Figure 3. 5 : Architecture of GRU cells[65].

The following equations define the gating mechanism in a GRU. For GRU, the hidden state  $h_t$  is computed as follows:

$$z_t = \sigma(x_t U^z + h_{t-1} W^z) \quad (3.8)$$

$$r_t = \sigma(x_t U^r + h_{t-1} W^r) \quad (3.9)$$

$$\tilde{h}_t = \tanh(x_t U^h + (r_t * h_{t-1}) W^h) \quad (3.10)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (3.11)$$

### **3.8 Parameters Tuning Methods**

To implement the deep learning approach to IDS, knowing selection and set of the various hyper parameters were a difficult task? These include:

- Model building related parameters such as: - number of hidden layers and number of neuron in the hidden layer.
- Model training parameters such as: - optimizers, Regularization, Batch Size, Epochs, activation function etc.

The deep learning algorithms training process as well as excellence of the model is influenced relatively heavily, by the above mentioned hyper parameters, hence these are an important part, so in order to select or tuned such parameters, one of the following methods should be applied before to build the proper model, these are grid search and manual tuning methods.

#### **3.8.1 Manual Tuning**

It depends on the modeler's deep understanding of the learning neural model, and therefore, the training process to be able to choose the top appropriate hyper-parameters were tuned manually by trial and error and the major consideration of this was idea of the model capacity. This can be stilled commonly done, and experienced operators can guess parameter values which will achieve good accuracy for deep learning models [70].

#### **3.8.2 Grid Search**

This is a complete search within the parameter space during which all possible combinations are verified, in this method specify the grid of values (hyper parameters) that we want to try out, and optimize to get the best parameter combinations. Then we build models on each of those values (combination of multiple parameter values), we search the whole parameter grid to select best parameters using cross validation, and report the best parameters' combination in the whole grid. The output will be the model using, the best combination from the grid [71].

### **3.9 Hyper Parameters**

For training and testing of intrusion detection for a deep neural network, following parameters have a direct impact, for performance of a models on the dataset. Likewise, implementation of proposed RNN networks are parameterized functions, so that the optimal parameters that are

used to implement this this research were listed below and there set values are made in using one of the above mentioned method [72].

### **3.9.1 Hidden Layers**

The initial design of deep learning model consists of one hidden layer followed by a standard feed-forward output layer. A layer is where all training process takes place. Inside a layer there are countless amounts of weights (neurons). Usually, the amount of hidden layers to be used is based on scale of the dataset and therefore, the dimensions (input and output shape of training and testing dataset) [72].

### **3.9.2 Hidden Units**

A number of neurons within the RNNs, LSTM and GRU hidden layer, this were used to check how well-trained the neural network can be, this is what mean that the number of hidden neurons must be between the scale of the input layer and also the size of the output layer [68].

### **3.9.3 Epochs**

The amount of periods that the learning system will work through the whole training time from 1to100 epochs. For instance, one epoch within the learning algorithm means each sample within the training dataset has had a chance to update the inner model parameters. However, number of epochs is historically large, regularly hundreds or thousands, its permit the learning algorithm to run until the error from the model has been sufficiently minimized.

### **3.9.4 Activation Functions**

In machine learning activation function, is used to transform the activation level of input unit (neuron) into an output signal. During learning time of different RNN algorithms of proposed methods also used for activation functions. This suggests that, the input data of our approaches are numeric data point, this input value are fed into the neurons within the input layer, in this case separately neuron has a weight and multiplying input amount with neuron provides the output of neuron, which is moved to the next layer, so in this process an activation function should be applied for the learning model, which is called a sigmoid activation function [73].



The sigmoid activation function is sometimes stated logistic function having a characteristic “S”-shaped curve or squashing function. The sigmoid is a non-linear activation function, used mostly in feed forward neural networks. It performs in the output layers of the deep learning architectures, and used for predicting and has been applied successfully in binary classification problems, modeling logistic regression tasks as well as other deep neural network domains [69].

The logistic function is defined by the formula.

$$\sigma(x) = \frac{L}{1+e^{-k(x-x_o)}} \quad (3.12)$$

Where  $e$  is the natural logarithm base (also known as Euler’s number),  $x_o$  is the  $x$ -value of the sigmoid midpoint.  $L$  is the curve’s maximum value, and  $k$  is the steepness of the curve. By setting  $L=1$ ,  $k=1$ ,  $x_o = 0$ , we derive that.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (3.13)$$

This is so the equations of sigmoid function.

### 3.9.5 Optimizers

Learning a neural network, especially deep neural networks has been a long-lasting challenge in the machine learning community. Fundamentally, training neural network is a non -convex optimization problem, so to solve this issue an optimizer should be used, which is to re-adjust weights of the network after each iteration, until it reaches an optimal solution and its updates the gradients iteratively until the loss function gives a minimum value [74]. In machine learning there are so many types of optimizers to name a few are: SGD, RMSprop, AdaDelta/AdaGrad and Adam. Stochastic gradient decent is a natural choice, however training deep architecture using SGD is time consuming, due to the limited computation memory especially in CPU and GPU. AdaDelta/AdaGrad used for sparse data. Whereas RMSprop with momentum generates its parameter updates using a momentum on the rescaled gradient. Currently most commonly used method for training of deep leaning architecture is Adam [75].

Adam, which is method of efficient stochastic optimization, that only requires the first order gradients and which have small memory requirement. It uses past gradients to calculate present

gradients and update weights of network and it is derived from adaptive moment estimation for a combination of momentum and RMSprop [76]. It acts upon: -

- Gradient component using by the usage of V. this is an exponential moving ordinary of gradients (like momentum), and
- Learning rate component divisible by learning rate ( $\alpha$ ) by square root of S. this is for exponential moving ordinary of squared gradients (like in RMSprop). The following formula are compute the process of Adam.

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{\hat{S}} + \epsilon} \cdot \hat{V}_t \quad (3.14)$$

$$\text{Where } \hat{V}_t = \frac{V_t}{1-\beta_1^t}, \hat{S}_t = \frac{S_t}{1-\beta_2^t}, \text{ Thus}$$

$v_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t}$ ,  $S_t = \beta_2 V_{t-1} + (1 - \beta_2) \left[ \left( \frac{\partial L}{\partial w_t} \right)^2 \right]$  are the bias corrections, and with V and S initialized to 0. And  $\alpha$ ,  $\beta_1$  and  $\beta_2$  are configuration parameters of Adam.

### 3.9.6 Drop Out Function

This is used a deep neural network for regularization technique. The significant indication is randomly drops units (along with their connections) from neural network throughout training and testing. This were significantly reduces over fitting [77]. In general, dropout helps the network to generalize better and increase accuracy, by escaping training all neurons, on the complete learning data in one go. Figure 3.6 are defining how dropout function works with  $(1-p)$  probability for each of the specified layers.

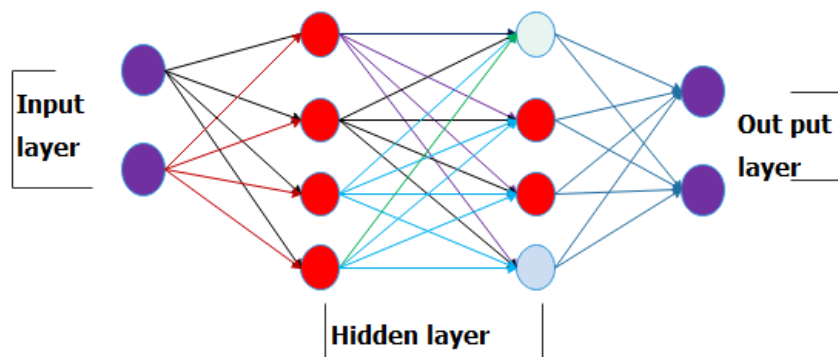


Figure 3. 6: Working principle of dropout function to neural network[77].

### **3.9.7 Batch Sizes**

Batch size is amount of samples processed before the model is updated. The dimension of a batch essential be more than (equal to one) and less than (identical to the amount of samples in the training data).

## **3.10 Materials and Tools Used to Implement the System**

To implement the proposed anomaly intrusion detection system, we used hardware and software requirements. This is where needed to be met the goal and outcome of this research.

### **3.10.1 The Hardware Material Requirement**

Listed below are the hardware material used within a laptop to carry out the experiment: AMD A6-4400M APU with Radeon (tm) HD Graphics 2.70 GHz processor, windows 10 version 1511, RAM 6.00GB, 64-bit operating system, x64-based processor and used 269 GB Local Disk.

### **3.10.2 The Software Tools Requirement**

Several tools are used in the development of the proposed anomaly intrusion detection system. Especially python programing language with different library frame work, have a great contribution for the accomplishment of this work [78]

## **3.11 Performance Evaluation Metrics**

A model can be evaluated during the training phase and testing phase. Therefore, to evaluate the performance of variant RNN models on a proposed dataset. We use following evaluation metrics: model training and testing classification accuracy, overall classification rate accuracy and the time for training and testing of a models.

### **3.11.1 Model Training and Testing Accuracy**

Machine learning model accuracy is the measurement, used to govern which model is great at finding relationships, and patterns among features in a data created on the input data. Which means performance accuracy of the training and predict unseen data.

### 3.11.2 Overall Classification Rate Accuracy

In machine learning, overall classification rate accuracy, is the most essential measurement the performance of a classifier. It determines the proportion of correctly classified samples in relation to the total number of samples of testing and training set. It can be obtained that the accuracy of each experiment was based on the percentage of successfully classification (PSC) on the training and testing set.

Thus, the overall (average) classification rate accuracy were obtained as follows: -

$OCRA = \frac{\text{sum of PSC of each attacks}}{\text{total number attack (ie.4)}}$ , since in this study the categories of anomaly attack on the training and test dataset are fours (DoS, Probe, U2R, R2L).

Where,  $PSC = \frac{\text{number of correctly classified instance}}{\text{number of instance in the train and test set}}$ .

## **CHAPTER FOUR: RESULT AND DISCUSSION**

### **4.1 Introduction**

This chapter described, detail experimental methodology and discuss the result of proposed systems, the chapter also compare three approaches with the proposed systems.

### **4.2 Preparation of the Experimental Dataset**

As already explained in chapter three in detailed, the dataset considered for this research was NSL\_KDD+ train and NSL\_KDD+ test dataset, where the values have 125973 pattern of train and the test dataset values has 22544 pattern records. The environment on which the application is built is called jupyter, which is a work bench provided by an anaconda plat form. The programing language used by this anomaly intrusion detection model is python. In order to make decided with the data, we considered three types of RNN algorithms such as SimpleRNN, LSTM and GRU to build this study. The performance accuracy of these models were evaluated on the dataset for the class of DoS, Probe, U2R and R2L attack types.

### **4.3 Experimental Methodology**

All experiment has two phases namely, learning model on the training data and then predicting the test data. Figure 4.1 show steps involved in proposed experimental model and the steps are shown below:

1. NSL-KDD+Train and NSL\_KDDTest are the input data set in the experiment.
2. Involves data processing and normalization.
3. Feature selection, it involves feature of data reduction, we reduce feature of data set.
4. Classify the instances of categories of the attack on the dataset into categories of DoS, R2L, U2R and Probe attacks.
5. SimpleRNN, LSTM and GRU algorithms are applied.
6. Adjust model parameters and select the model that were the best fit for the given data set and algorithms.
7. Compile and fit the input data to the model and then train and test the classifiers with these new training datasets.
8. Compare classification performance accuracy among all classifier for all and selected features of the data.

## 4.4 Experiments of Proposed System

The experiment has two major parts. The first part was experimentation that uses all features of the training and testing dataset, having 122 input features after preprocessing of the data. For the second experiment, the method is the same as for the first experiment, but filter based feature selection mechanism was applied on the dataset. In filter method to select the best feature (truncated unused data) on the dataset. So in this case we selected 22 best input features on the dataset and the experiment was carried out on the selected features. But before to implement these two experiment, first thing to do is to select the proper hyper parameters using parameters selection method, which is defined and explained in chapter four, which is called grid search method was chosen and used among the other, from this method the proper model trainable parameters were selected. Initially, we used reasonably sized SimpleRNN network which contains input layer, hidden layer and output layer. The input layers contain 122 neurons for the first experiment and 22 neurons for the second experiment and a hidden layers contain 4 contains 64 cell each and an output layers include 4 neurons and to find a suitable batch\_size, number of epochs, drop out function. A grid search method was applied to choice the optimal parameters between the following range of values. For batch\_size = [100, 500, 1000], epochs [50,100] and dropout rate [0.1,0.2,0.3,0.4, 0.5, ..., 0.9]. The experiments with higher batch\_size and epoch with lower dropout rate have showed superior performance in categorizing the connection records of attack DoS using the model, see in Appendix D. We decided to use 500 Batch\_size, 100 epochs and 0.1 dropout rate for the rest of the experiments. Table 4.1 shows the important parameters and its set values using a grid search method, these are to implement and experiment all the models having 122 and 22 input features of proposed dataset.

Table 4. 1: List of implementation hyper parameters and its set of value.

Parameters	Setting value
Number of hidden layers	4
Number of hidden units/ neurons	64
Batch size	500
Number of epochs	100
Optimizers	Adam
Activation function	Sigmoid
Loss function	Binary cross entropy
Regularization (dropout rate)	0.1

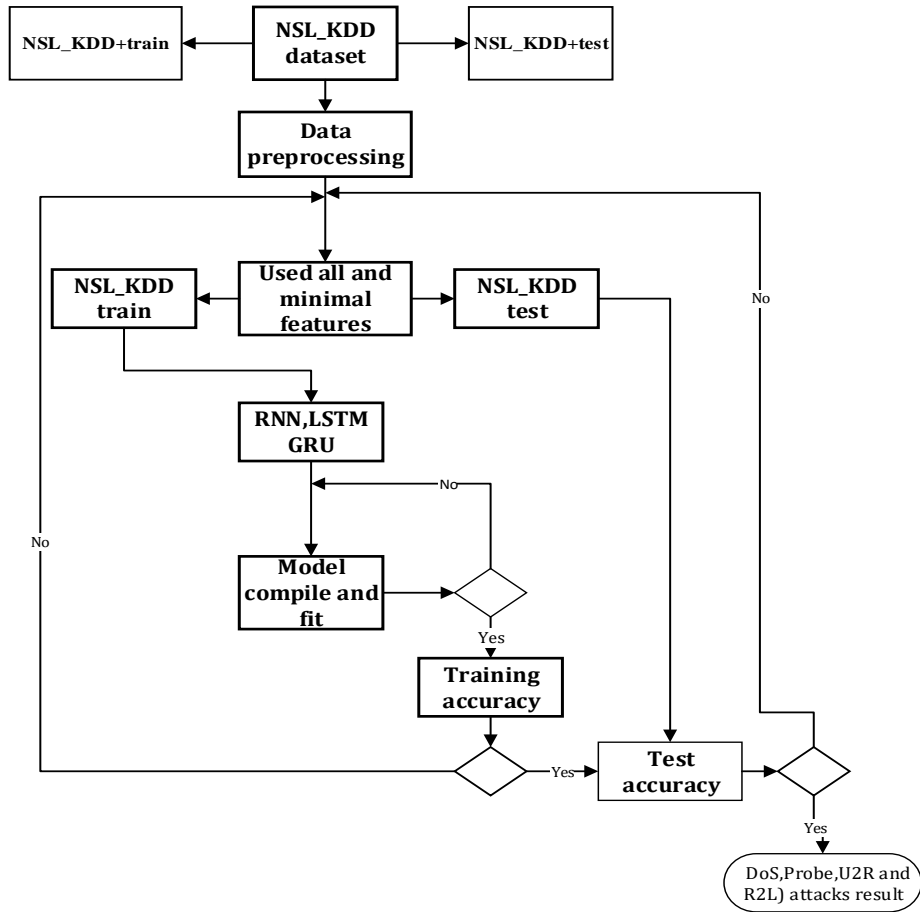


Figure 4. 1: Experimental system design methodology.

### 4.3 Phase One and Two Experimental Result

In these experiment, we used all and minimal features of the dataset, these for evaluating the performance of the design models. We used all features means that used 122 input attributes of the data for after preprocessing of the whole data. And we used minimal attributes of the data means to selected top best features of the data to be considered, so in this case a filter based feature selection technique is applied on the dataset and we selected top 22 features from dataset. Therefore, the classification performance accuracy of DoS, Probe, U2R and R2L attacks of these two experiment on the different models are shown below.

#### 4.4.1 SimpleRNN Experiments Using All and Minimal features

Layer (type)	Output Shape	Param #
simple_rnn_14 (SimpleRNN)	(None, 1, 64)	11968
dropout_18 (Dropout)	(None, 1, 64)	0
simple_rnn_15 (SimpleRNN)	(None, 1, 64)	8256
dropout_19 (Dropout)	(None, 1, 64)	0
simple_rnn_16 (SimpleRNN)	(None, 1, 64)	8256
dropout_20 (Dropout)	(None, 1, 64)	0
simple_rnn_17 (SimpleRNN)	(None, 64)	8256
dropout_21 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 1)	65
activation_5 (Activation)	(None, 1)	0
Total params: 36,801		
Trainable params: 36,801		
Non-trainable params: 0		

Figure 4. 2: Capture of configuration of SimpleRNN model for all features.

Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(None, 1, 64)	5568
dropout_1 (Dropout)	(None, 1, 64)	0
simple_rnn_2 (SimpleRNN)	(None, 1, 64)	8256
dropout_2 (Dropout)	(None, 1, 64)	0
simple_rnn_3 (SimpleRNN)	(None, 1, 64)	8256
dropout_3 (Dropout)	(None, 1, 64)	0
simple_rnn_4 (SimpleRNN)	(None, 64)	8256
dropout_4 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65
activation_1 (Activation)	(None, 1)	0
Total params: 30,401		
Trainable params: 30,401		
Non-trainable params: 0		

Figure 4. 3: Capture of configuration of SimpleRNN model for minimal features.

Table 4. 2 : SimpleRNN model using all and minimum features of the dataset.

Model	Attack	Using All features( accuracy in%)			Using minimal features( in%)		
		Train data	Test data	Run time(sec)	Train Data	Test data	Run time(sec)
Simple RNN	DoS	99.94%	<b>92.52%</b>	8s	99.73%	<b>93.13%</b>	6s
	Probe	85.08%	76.38%	7s	84.90%	<b>79.22%</b>	6s
	U2R	99.89%	<b>97.02%</b>	6s	<b>99.89%</b>	<b>97.11%</b>	5s
	R2L	98.41%	76.47%	5s	98.26%	76.49%	4s

Table 4.2 shows the training and testing classification accuracy results, in percent of different attacks for using SimpleRNN model with 100 epochs. This model is evaluated for using 122 and 22 input features, for the attacks of DoS, Probe, R2L and U2R on the NSL\_KDD dataset.



Bold values represent to the best result achieved during the experiments. In this model the classification accuracy of U2R attack was achieved maximum value among the other, for using minimal features of the unseen dataset. The sample training and testing of this experiments are documented in appendix D in this paper.

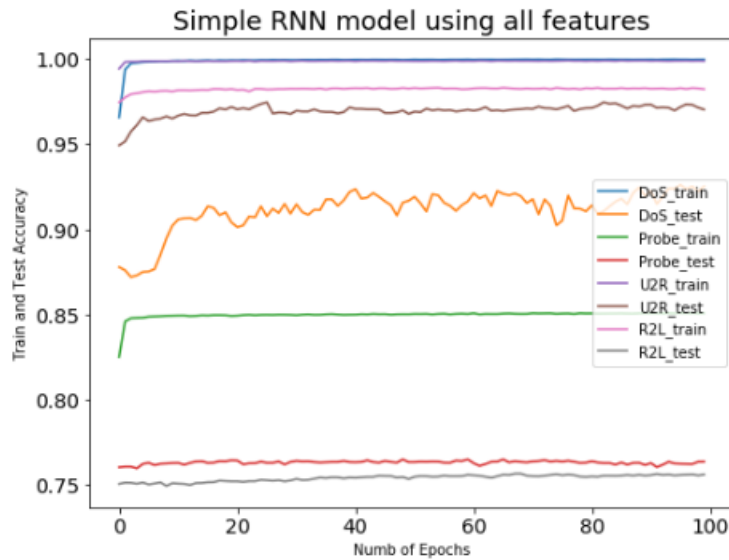


Figure 4. 4: Performance accuracy of simpleRNN for using 122 input features.

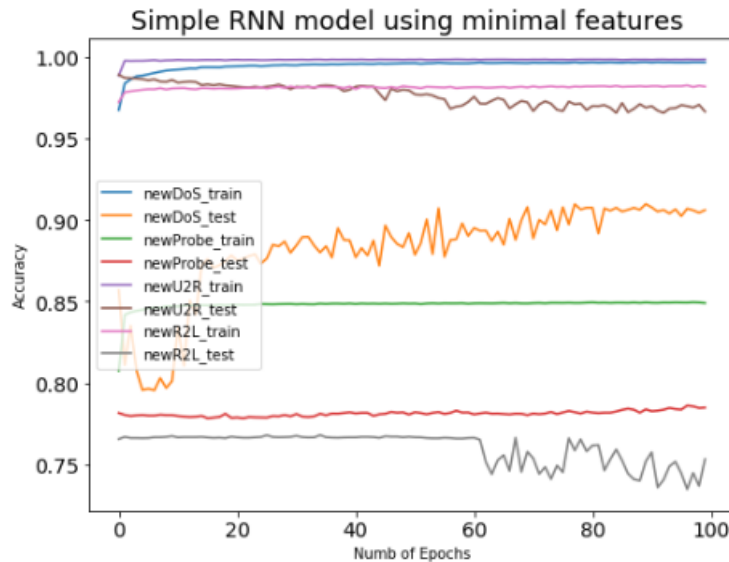


Figure 4. 5: Performance accuracy SimpleRNN model for using 22 input features.

The plot in the given Figure 4.4 and Figure 4.5 were indicated by the output values for the classification accuracy of four attacks, for using 122 and 22 input features of the dataset, that are evaluated on SimpleRNN model. In these experiments we used 100 epochs, 4 hidden layers, and 64 units of neuron for each hidden layer and 500 batch size, and also used other parameters

that were listed in table 4.1. The X-axis shows the time steps (number of epochs) during the training and testing case, this was going to epochs 1 to 100 made by the RNN algorithms. The Y-axis has the corresponding metric value which was model training and testing classification accuracy values of each attack classes. The tested and trained values of each category of attack are shown as different markers of colors on the figure, that were indicated in the middle line graph.

### Other Performance Measures

The overall classification performance accuracy of each categories of attack DoS, Probe, R2L and U2R for using all features of the data are:  $(\text{Dos}\% + \text{Probe}\% + \text{U2R}\% + \text{R2L}\%)/4$ . This is equal to  $99.96 + 85.09 + 99.89 + 98.41/4 = 95.835\%$  on the train data, whereas on the test data the OCP accuracy of each categories of attack are:  $91.57 + 76.23 + 98.59 + 76.47/4 = 85.715\%$ . Overall classification accuracy (using selected features) = 95.695% from training, 85.7375% from test data.

#### 4.4.2 LSTM Experiment Using All and Minimal Features

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 1, 64)	47872
dropout_22 (Dropout)	(None, 1, 64)	0
lstm_2 (LSTM)	(None, 1, 64)	33024
dropout_23 (Dropout)	(None, 1, 64)	0
lstm_3 (LSTM)	(None, 1, 64)	33024
dropout_24 (Dropout)	(None, 1, 64)	0
lstm_4 (LSTM)	(None, 64)	33024
dropout_25 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 1)	65
Total params: 147,009		
Trainable params: 147,009		
Non-trainable params: 0		

Figure 4. 6: Capture of configuration of the proposed LSTM model for using all features.

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 1, 64)	22272
dropout_9 (Dropout)	(None, 1, 64)	0
lstm_2 (LSTM)	(None, 1, 64)	33024
dropout_10 (Dropout)	(None, 1, 64)	0
lstm_3 (LSTM)	(None, 1, 64)	33024
dropout_11 (Dropout)	(None, 1, 64)	0
lstm_4 (LSTM)	(None, 64)	33024
dropout_12 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65
Total params: 121,409		
Trainable params: 121,409		
Non-trainable params: 0		

Figure 4. 7: Capture of configuration LSTM model for using minimal features.

Table 4. 3: LSTM performance using all and minimum features of the dataset.

Model	Attack	Using All features acc%			Using minimal features acc%		
		Train data	Test data	Run time(sec)	Train data	Test data	Run time(sec)
LSTM	DoS	<b>99.97%</b>	<b>93.22%</b>	30s	99.79%	88.42%	22s
	Probe	85.11%	76.27%	22s	84.87%	75.03%	17s
	U2R	99.86%	96.35%	19s	<b>99.89%</b>	<b>98.53%</b>	13s
	R2L	98.30%	75.27%	21s	98.25%	76.88%	16s

The training and testing performance of the classifier LSTM model is presented in Table 4.3. This table shows the classification accuracy of the attacks for DoS, Probe, U2R and R2L for the usage of all and minimal features of the proposed dataset. The results demonstrate that the model fit and performed very well in the classification accuracy for DoS and U2R attacks. Moreover, classification accuracy of U2R attack is obtained good accuracy for using minimum features of test data. Although all attack was achieved good result accuracy in both experiments of the model, the least performance result was found for the Probe attack, since the record value of this attack less on the dataset.

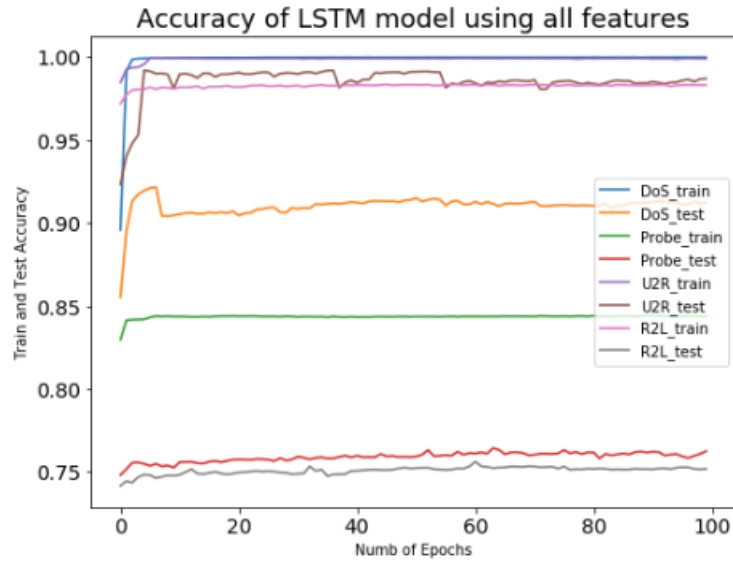


Figure 4. 8: Accuracy predicted for all attack types with 122 features for LSTM classifiers.

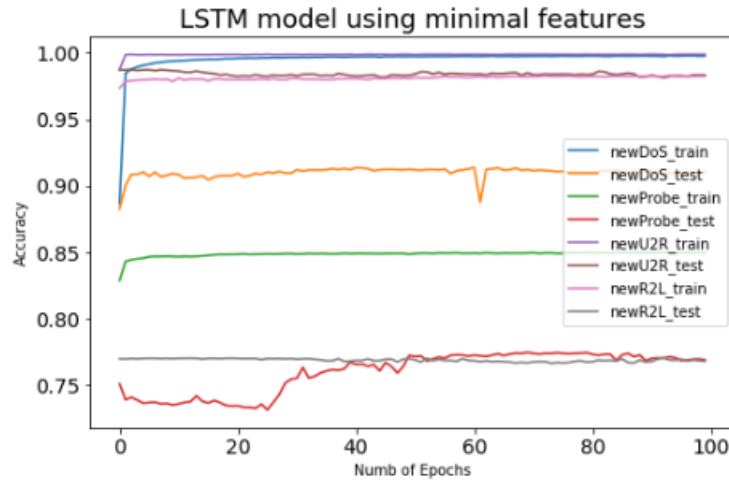


Figure 4. 9: Accuracy predicted for all attack types with 22 features for LSTM classifiers.

Figure 4.8 and Figure 4.9 described the training and testing classification accuracy of all four attacks on the LSTM model. To evaluate this model all and minimal features of the NSL\_KDD dataset was used. Those attacks are well detected and achieve promised classification accuracy of visibility result, in more than 80 percent on the training and more than 70 on the unseen data for both experiments. Moreover, the best classification accuracy was 99.89% and 98.53% achieved by the attack category of U2R for using minimum features of both the training and testing phase.

## Other Performance Measures

For using the LSTM algorithms OCA on the train data is 99.97% +85.11% +98.86% +99.30% /4 =95.81%, on the test data is = 85.2775%. And overall classification accuracy (for selected features) are = 95.7% from training data, 84.715% from test data .

### 4.4.3 GRU Experiment Using All and Minimal Features

Following captures figures and tables described the experimental result of GRU model for all and selected features of the NSL\_KDD dataset, the result were documented and the discussion are shown below.

Layer (type)	Output Shape	Param #
gru_1 (GRU)	(None, 1, 64)	35904
dropout_14 (Dropout)	(None, 1, 64)	0
gru_2 (GRU)	(None, 1, 64)	24768
dropout_15 (Dropout)	(None, 1, 64)	0
gru_3 (GRU)	(None, 1, 64)	24768
dropout_16 (Dropout)	(None, 1, 64)	0
gru_4 (GRU)	(None, 64)	24768
dropout_17 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 1)	65
activation_4 (Activation)	(None, 1)	0
Total params: 110,273		
Trainable params: 110,273		
Non-trainable params: 0		

Figure 4. 10: Capture of configuration of the proposed GRU model using all features.

Layer (type)	Output Shape	Param #
gru_1 (GRU)	(None, 1, 64)	16704
dropout_5 (Dropout)	(None, 1, 64)	0
gru_2 (GRU)	(None, 1, 64)	24768
dropout_6 (Dropout)	(None, 1, 64)	0
gru_3 (GRU)	(None, 1, 64)	24768
dropout_7 (Dropout)	(None, 1, 64)	0
gru_4 (GRU)	(None, 64)	24768
dropout_8 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
activation_2 (Activation)	(None, 1)	0
Total params: 91,073		
Trainable params: 91,073		
Non-trainable params: 0		

Figure 4. 11: Capture of configuration of the proposed GRU model for minimal features.

Table 4. 4 : GRU performance using all and minimum features of the dataset.

Model	Attack	Using All feature acc in%			Using minimal feature acc in%		
		Train data	Test data	Run time(sec)	Train data	Test data	Run time(sec)
GRU	DoS	<b>99.97%</b>	<b>93.31%</b>	21s	99.77%	90.79%	19s
	Probe	85.06%	76.42%	36s	84.89%	76.73%	11s
	U2R	99.89%	<b>98.20%</b>	15s	<b>99.90%</b>	<b>99.15%</b>	10s
	R2L	98.32%	75.56%	10s	98.29%	76.98%	8s

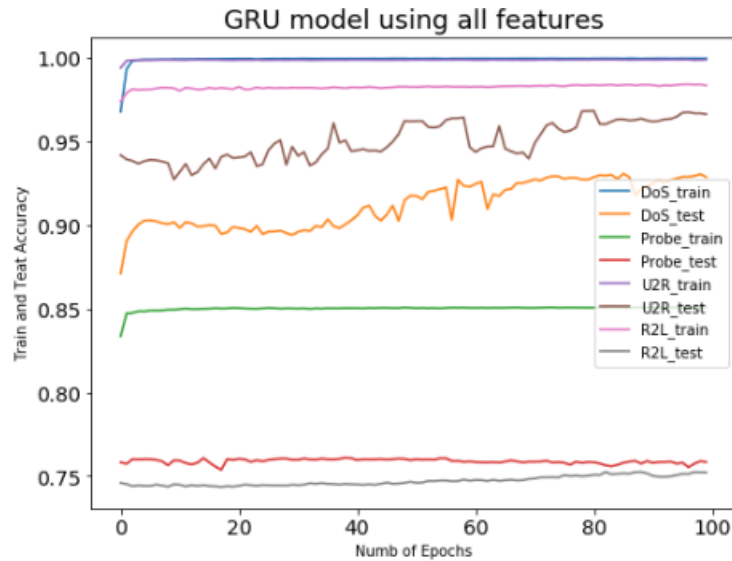


Figure 4. 12: Accuracy predicted for all attack types with 122 features for GRU classifiers.

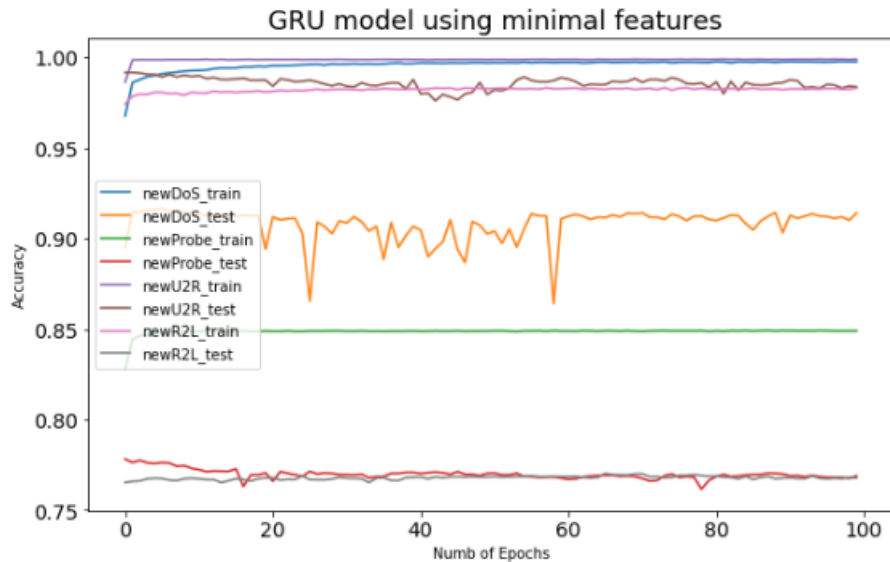


Figure 4. 13: Accuracy predicted for all attack types with 22 features for GRU classifiers.

Figure 4.12, Figure 4.13 and also Table 4.4 were described training and testing classification accuracy of four attacks on GRU model. This model was evaluated for using 122 and 22 input

features of the NSL\_KDD dataset, the best value of these two experiments are recorded on the categories of attack U2R and DoS, in both the training and testing data. However, for the usage of minimal features, U2R has achieved maximum value among the other attacks which value has: 99.90% on the training and 99.15% on the test data with short training time.

### **Other Performance Measures**

In this case overall classification performance accuracy, on GRU algorithms for using different experiments are documented. OCA for using: -

All features =95.81% from training data and 85.6725% from test.

Selected features =95.7125% from training data and 85.6625% from test.

## **4.5 Experimental Result Discussion**

All of the result from the experiments are presented and documented in the form of different types of graphs, tables and texts in this chapter. Moreover, the discussion should be give an overview of how the different experiments are performed and the results are so will mostly discussed. In this study we apply two phase of experiments. First phase was used all features of the dataset and the second one is selected top features of the dataset that was best contributed for design models. For all the experiments are designed and implemented using three types of RNN algorithms, and each implemented models were evaluated for the different types of attack on NSL\_KDD dataset in terms of: model training and testing classification accuracy, average or overall classification accuracy and an execution of training and testing time.

In the first phase of experiment, the train and test are performed on the categories of attacks (DoS, Probe, R2L, U2R) on the proposed algorithms and proposed dataset. Therefore, in the category of DoS, out of 125973 pattern total train data and 22544 pattern total test data, there are 113270 new DoS train attacks patterns on the training set, included all attacks and there are 17171new DoS test patterns include attack that is: apache2, mailbomb, processtable, and udpstorm. And in the category of Probe, there are about 78,999new train and 12,132new test patterns of attack data are performed. Thus, two new attacks are mscan and saint on the test data. In the category of R2L, from all train and tests of data, there are about 68,338 R2L attack data on the training, and also, there are about 12,596 R2L attack data on the test set, which include a new attack such as httptunnel, named, sendmail, snmpgetattack, snmpguess, worms,

xlock, and xsnoop. In the category U2R, U2R attack data from all trains and tests was 67,395 and 9,778 data. There is a new attack include ps, sqlattack, and xterm on the test data.

From this different label of attacks, the experimental results indicated that the proposed system achieved overall classification accuracy in percent of 95.8375%, 95.81% and 95.81% by using Simple RNN, LSTM and GRU algorithms. Besides the classification accuracy, for DoS attack is achieved from train data: simple RNN has 99.96%, LSTM has %, 99.97%, and GRU has 99.97% respectively, Simple RNN, LSTM, and GRU successfully achieved for Probe, U2R and R2L attack that are discussed and documented in table 4.2,4.3 and 4.4 of in this chapter.

Whereas the achieved overall classification accuracy on the test data were 85.66%, 85.2775% and 85.8725%. Additionally, accuracy of DoS for the use of simple RNN, LSTM, and GRU on the test, had 91.57%, 93.22% and 93.31% respectively. And for Probe on the simple RNN has 76.23%, LSTM has 76.27% and GRU has 76.42%, for U2R for using these three algorithms has 98.59%, 96.35%, and 98.20% respectively. And also, for R2L has 76.47%, 75.27% and 75.56% for using Simple RNN, LSTM and GRU algorithms.

In the second case of experiment, filter based features choice was useful on the training and testing data of NSL\_KDD dataset. This was to help in identifying and removing nonessential variables from the dataset. In this case of experiment, the selected features of the dataset were most contributed to the models for both training time and accuracy. In this case model accuracy after and before features selection almost the same as shown the result from above. However, the training time were much different, this is clearly show that how a feature selection can reduce the training time of the models. So, the proposed system carried out following overall classification accuracy rate on the use of selected 22 input features on the training and test data. These results are 95.695%, 95.7125% and 95.7125% for Simple RNN, LSTM and GRU in percent of the training data, respectively.

Whereas the achieved overall classification accuracy on the test data were 85.7375%, 84.715% and 85.6625%. In addition to that obtained results from all experiments, U2R attacks achieved a maximum value with short training time to the others. For the classification accuracy of Probe and R2L on the test data recorded less accuracy compare to others. The reason for this is that the class in balance given that there are only little sample of data in the test set for Probe and R2L, respectively, compared to the overall total 22544 samples or no more sequence of data



are recorded on the test data. In general, for all models training time of selected minimal input features were executed very short time compared to training time of all input features.

#### **4.5.1 Comparison of Proposed Work**

In this section discussed the comparison SimpleRNN, LSTM and GRU models themselves on proposed dataset for evaluating the attacks of DoS, Probe, U2R and R2L for in terms of best classification performance and training time. From these three models, Simple RNN models has a stable overall performance in classification among the other two, for both experiments in terms of training time, which was completed very short training and testing time.

#### **4.5.2 Comparison to Existing System**

We are comparing the performance of proposed SimpleRNN, LSTM, and GRU classifiers used to build our model with the performance of existing intrusion model for the intrusion dataset. Given below is the detail comparison:

In reference [21], three machine learning methods were proposed for classifying and detecting network anomalies attacks on the KDD training dataset, our model predicts and achieved for using Simple RNN classifier with 95.835% accuracy, LSTM has 95.81% accuracy and GRU has 95.810% accuracy achieving superior performance in comparison with the existing models MLP, J48 and Bayes network for in terms of overall classification accuracy.

In reference [32], author proposed an autonomous intrusion detection system by means of ensemble of advanced learners, author used two deep learning techniques called gated recurrent unit and convolutional neural network and also used one machine learning technique is random forest, for these three methods the author was to identify and classify normal and attack for the relevant features of the NSL-KDD test dataset. The obtained results of these algorithms were compared and obtained an overall classification accuracy of 83.17% for using GRU algorithm, for CNN has 82.92% and RF achieved an overall classification accuracy of 80.14%. But so our approach was 85.715% for using simpleRNN, 85.2775% for LSTM and GRU also achieved an overall classification accuracy of 85.6725%% on the unseen dataset.

In reference [38], network anomaly detection with chi-squared feature selection method was applied on the different attack of the KDD dataset. The feature selection method used in our proposed work has ease of selected features than the other method used in the referenced paper

[38]. The classification accuracy of the studies is also lower than our approach on the U2R and R2L attacks. Our models predict the U2R attack using Simple RNN classifier accuracy with 96.11% and R2L attack with 76.49% accuracy. For using LSTM U2R achieved 98.53% and for R2L 76.88%. And also for using GRU model U2R has recorded for 99.15% and R2L has 76.98% achieving superior performance in comparison with the existing models. In addition, Table 4.5 compares the overall classification accuracy and classification accuracy of individual attacks of previous studies with our proposed research work on the minimal features of NSL-KDD test data. Based on these comparison results, our approach outperforms are to examined the attack U2R and R2L with the NSL\_KDD dataset in terms of both accuracy and training time duration compared to the previous study works in references [18, 32, 38, 37] in network anomaly detection of attacks.

Table 4. 5: Comparison with previous studies

<b>Existing Approaches</b>	<b>Overall Classification accuracy and multi class classification acc in %</b>
SVM[18]	83.7% multiclass classification
CNN, GRU and RF[32]	RF= 80.14% CNN=82.92%, GRU= 83.17%
core vector machine with Chi-square[38]	R2L= 76.41% and U2R= 93.71%
Improved conditional variational auto-encoder (ICVAE) with DNN[37]	Overall classification accuracy = 85.97% DoS =85.65%, Probe= 74.97% U2R = 11.00%, R2L =44.11%
<b>Proposed approaches</b>	<b>Classification accuracy in %</b>
SimpleRNN with Filter based feature selection.	DoS= 91.13%, Probe= 79.22%, R2L= 76.49%, U2R= 96.11%.
LSTM with Filter based feature selection.	DoS= 88.42%, Probe= 75.03%, R2L= 76.88%, U2R= 98.53%.
GRU with Filter based feature selection.	DoS= 90.79%, Probe= 76.73%, R2L= 76.98%, U2R= 99.15%.

## **CHAPTER FIVE: CONCLUSION AND FUTURE WORK**

### **5.1 Conclusion**

The most important goal of this thesis was to implement anomaly intrusion detection system on variant recurrent neural network techniques. Which were called as Simple RNN, LSTM, and GRU. Two experiments were performed for using these three algorithms on the NSL\_KDD dataset. First one was using all features of the dataset and the second one was using minimal features of the dataset. The advantage of these experiments on the models are to give a proposed solution for, problem statement and intention objectives of proposed systems. And also to solve challenge of U2R and R2L attacks classification problem. This was what has been done and described in chapter four in detailed. The chosen hyper parameters are some of the factors that generate good accuracy for both experiments for each attack types, especially for the attack of U2R.

The classification performance of all models are good for each attacks for using all and minimal features of the proposed dataset. Overall, for the categories of U2R attack recorded high among the other, on minimal features of the dataset in comparison to that of all features, which values are for using SimpleRNN, GRU and LSTM has had 96.11%, 99.15% and 98.53% respectively. But in terms of time consumption, Simple RNN is consumed much less time than LSTM and GRU for both experiments of the proposed dataset. And also for comparison of all and minimal features during training time of all proposed methods, using minimal features consume much time than that of for using all features.

### **5.2 Future Work**

The future of network security technologies may be in the further integration of various deep learning tools to ensure network security, because the administration of computer safety is an increasingly complex and extensive task, which means computer systems still not secure 100%, while security needs are growing. However, proposed system used three algorithms of RNN types for detection of attacks types on the NSL\_KDD dataset. Hence, it can be analyzed using other RNN approaches. Since, there are so many different RNN algorithms that could be used to do classification and detection, in this thesis three of most detection algorithms were chosen to be compared.

## REFERENCES

- [1] A. G. Ali, L. Wei and T. Mahbod, Network Intrusion Detection and Prevention, Concepts and Techniques, s. c. f. s. information, Ed., New York: springer, 2010.
- [2] S. Sanjiban, M. Abhinav, G. Rishab, S. O. Mohammad and P. V. Krishna, "A deep learning based artifitil neural network approach for intrustion detection," in *International conference on mathematics and computing*, 2017.
- [3] C. S. Ralf and R. M. Eric, "Understanding LSTM," *Application of long short term memory(LSTM) neural network for flood forecasting*, Sep 2019.
- [4] A. Graves, A. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural network," in *IEEE international conference on acoustic, speech and signal processing*, 2013.
- [5] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Emperical evalution of gated recurrent neural network on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
- [6] S. Garfinkel and G. Spafford, Practical UNIX and Internet Security, Second ed., Tokiyo, Japan: in O'Reilly and Association, 1996.
- [7] D. Russell and G. Gangemi, Computer Security Basics, USA, Inc. California, : O'Reilly & Associates, 1991.
- [8] A. Richard and V. Giovanni, Intrusion Detection:A Brief History and Overview, California Santa Barbara.
- [9] B. K. Santos, S. ., R. T.Chandra, M.Ratnakar, B. Sk.Dawood and N.Sudhakar, "Intrusion Detection System- Types and Prevention," (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, vol. 4 (1), pp. 77 - 82, 2013.
- [10] O. Hamdan, M. N. Rafidah, B. B and A. Z. A, "Intrusion Detection System: Overview," *JOURNAL OF COMPUTING*, vol. 2, no. 2, pp. ISSN 2151-9617, FEBRUARY 2010.
- [11] [Online]. Available: [https://en.wikipedia.org/wiki/Anomaly\\_based\\_intrusion\\_detection\\_system#cite\\_note-Wang2004-1](https://en.wikipedia.org/wiki/Anomaly_based_intrusion_detection_system#cite_note-Wang2004-1).
- [12] D. ED, "An intrusion-detection model," *IEEE Transactions on Software*, vol. 13, no. 90, pp. 22-32, 1987.
- [13] B. M, S. E and K. M, "ANOMALY INTRUSION DETECTION DESIGN USING HYBRID OF UNSUPERVISED AND SUPERVISED NEURAL NETWORK,"

*International Journal of Computer Networks & Communications (IJCNC)*, vol. 1, no. 2, July 2009.

- [14] M. G. Mohamed, "Advances in Data Stream Mining," *wires data mining and knowledge discovery*, vol. 2, no. 1, pp. 79-85, 2012.
- [15] B. ALAN, P. CHANDRIKA, S. RASHEDA, S. BOLES LAW and E. MARK, "NETWORK-BASED INTRUSION DETECTION USING NEURAL NETWORKS," *proc.inteleigent engineering systems through artifitial neural network*, vol. 12, pp. 579-584, 2002.
- [16] J. Beale, J. Faster, C and J. Posluns, "SNORT 2.0 Intrusion Detection," *Syngress Publishing Inc*, 2003.
- [17] "A simple way to understand machine learning vs deep learning," [Online]. Available: <https://www.zendesk.com/blog/machine-learning-and-deep-learning/>.
- [18] M. K. Wakefield, "A guide to machine learning algorithms and their applications," [Online]. Available: [https://www.sas.com/en\\_ie/insights/articles/analytics/machine-learning-algorithms.html](https://www.sas.com/en_ie/insights/articles/analytics/machine-learning-algorithms.html).
- [19] Z. Shen, Z. Shibo, W. Bingnan and G. H. Thomas, "Machine Learning and Deep Learning Algorithms for Bearing Fault Diagnostics," *A Comprehensive Review*, 2019.
- [20] H. Nutan, R. Musharrat, O. Abdur, S. Faisal, H. Avishek and Dewan, "Application of Machine Learning Approaches in Intrusion Detection System :A Survey," *International Journal of Advanced Research in Artificial Intelligence(IJARAI)*, vol. 4, no. 3, 2015.
- [21] K. Saroj, Biswas and Silchar, "Intrusion Detection Using Machine Learning:A Comparison Study," *International Journal of Pure and Applied Mathematics* , vol. 118, no. 9, pp. 101-114, July 2018.
- [22] W. Bisyrn, R. Kalamullah and M. Hendri, "Implementation and Analysis of Combined Machine Learning Method for Intrusion Detection System," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 10, no. 2, pp. 295-304, 2018.
- [23] K. P. Sushant, "Design and performance analysis of various feature selection methods for anomaly-based techniques in intrusion detection system," *Wiley*, 2019.
- [24] J. Nicholas, Miller and A. Mehrdad, "Benchmarks For Evaluating Anomaly based Intrusion Detection Solutions," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 10, no. 5, September 2018.

- [25] A. Mouhammad and M. Almseidin, "Machine Learning Methods for Network Intrusion Detection," September 2018.
- [26] C. Prakash, K. L. Umesh and A. Prof. Nitin, "Network Intrusion Detection System Based On Modified Random Forest Classifiers For KDD Cup-99 and NSL-KDD Dataset," *International Research Journal of Engineering and Technology (IRJET)*, vol. 04, no. 08, pp. 2395-0072, Aug -2017.
- [27] M. Shahriar and N. Amin, "A New Deep Learning Approach for Anomaly Base IDS using Mimetic Classifier," *International Journal of Computers Communications & Control*, vol. 12(5), pp. 677-688, October 2017.
- [28] N. Sheraz, S. Yasir, K. Shehzad, K. B. Muhammad, H. Jihun, M. I. Muhammad and H. Kijun, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," *Special Section On Cyberthreats and Countermeasures In The Healthcare Sector Ieee Access*, vol. 6, pp. 48231-48246, 2018.
- [29] S. Nathan, N. N. Tran, D. P. Vu and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50 , FEBRUARY 2018.
- [30] H. Ali, Mirza and C. Selin, "Computer Network Intrusion Detection Using Sequential LSTM Neural Networks Autoencoders," *IEEE*, 2018.
- [31] A. Taief Alaa, A. Mohammed, M. A. Zakaria and A. Qusay M, "Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm," *Journal of Ambient Intelligence and Humanized Computing*, 29 October 2019.
- [32] M. Ponkarthika and V. R. Saraswathy, "Network Intrusion Detection Using Deep Neural Networks," *Asian Journal of Applied Science and Technology (AJAST)*, vol. 2, no. 2, pp. 665-673, 28 April 2018.
- [33] A. L. Simone, "Applying A Neural Network Ensemble To Intrusion Detection," *JAISCR*, vol. 9, no. 3, pp. 177-188, 2019.
- [34] A. K. Farrukh, G. Abdu, D. Abdelouahid and H. Amir, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," *Special Section On Artificial Intelligence And Cognitive Computing For Communication And Network* , vol. 7, pp. 30373-30385, March 20, 2019.
- [35] M. A. Vinayakumar, K. P. Soman, P. Prabakaran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *Digital Object Identifier IEEE ACCESS*, vol. 7, pp. 41525- 41550, April 11, 2019.

- [36] A. Amir and T. V. Vahid, "An Autonomous Intrusion Detection System Using Ensemble of Advanced Learners," *arXiv:2001.11936v1 [cs.LG]*, 31 Jan 2020.
- [37] N. Bhakti and M. Prof. Rupesh, "A Survey on Intelligent and Effective Intrusion Detection system using Machine Learning Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 01, p. 9, January 2020.
- [38] I. Félix and Z. Tanja, "Analysis of network traffic features for anomaly detection," vol. 101, pp. 59-84, 2015.
- [39] G. A. Wang and Wenjia, "Determining appropriate approaches for using data in feature selection," *Int. J Mach. Learn. & Cyber.* 8, pp. 915-928, 2017.
- [40] M. Maryam, M. K. Taghi and S. Naeem, "Evaluating Feature Selection Methods for Network Intrusion Detection with Kyoto Data," *International Journal of Reliability Quality and Safety Engineering*, vol. 23, no. 1, p. 22 pages, 2016.
- [41] Y. Yanqing, Z. Kangfeng, W. Chunhua and Y. Yixian, "Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network," *MDPI*, 2 June 2019.
- [42] H. Divyasree T and K. Sherly K, "A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach," *8th International Conference on Advances in Computing and Communication (ICACC)*, p. 442–449, 2018.
- [43] s. Kunal and K. J. M. Dr, "Performance Comparison of Intrusion Detection System Between Deep Belief Network (DBN)Algorithm and State Preserving Extreme Learning Machine (SPELM) Algorithm," 2019 (IEEE).
- [44] B. Afreen, Anand and Pitale, "Detection of Network Intrusions using Hybrid Intelligent Systems," in *2019 1st International Conference on Advances in Information Technology*, India, 2019 IEEE.
- [45] U. Ritumbhra, M. Dr and Gyanchandani, "Survey on Classification Techniques Applied to Intrusion Detection System and its Comparative Analysis," in *Proceedings of the Fourth International Conference on Communication and Electronics Systems IEEE Conference Record # 45898; IEEE Xplore ISBN: 978-1-7281-1261-9*, 2019.
- [46] K. Jin, S. Nara, Y. ., Seung and H. K. Sang, "Method of Intrusion Detection using Deep Neural Network," in *Proc. IEEE Int. Conf. Big Data Smart Comput.* (BigCompAvailable: <http://ieeexplore.ieee.org/abstract/document/7881684/>), p. 313–316, 2017.
- [47] [Online]. Available : <https://www.unb.ca/cic/datasets/ns1.html>..

- [48] S. F. B. D and Dennis, "Machine Learning Repository," UCI, [Online]. Available: <https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/>. [Accessed Last Accessed on 11 November 2019].
- [49] S. Gerry, "A Deeper Dive into the NSL-KDD Dataset," [Online]. Available: <https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657>. [Accessed Last Accessed on 23 December 2019].
- [50] L. Dhanabal and 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, no. 6, 2015.
- [51] M. Ahsan, G. Rahul and D. Anne, "SMOTE Implementation on Phishing Data to Enhance Cybersecurity," in *IEEE International Conference on Electro/Information Technology (EIT)*, IEEE, 2018.
- [52] T. Mahbod, B. Ebrahim, L. Wei and A. G. Ali, "A Detailed Analysis of the KDD CUP 99 Data Set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.
- [53] L. Dhanabal and 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*, pp. 446-452, 2015.
- [54] S. Kotsiantis, D. Kanellopoulos and P. Pintelas, "Data preprocessing for supervised learning," *International Journal Computer Science*, vol. 1, no. 1, pp. 111-117, 2006.
- [55] S. Tingkai and S. Chen, "Class label versus sample label-based cca," *Applied Mathematics and computation*, vol. 185(1) , p. 272–283, 2007.
- [56] C. Elkan, "Results of the KDD'99 Classifier Learning," *SIGKDD Explorations*, 2000.
- [57] P. Kedar, P. Chinmay and S. P. Taher, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," in *International Journal of Computer Applications*, vol. 175, no. 4, p. 0975 – 8887, October 2017.
- [58] A. Gökhan, O. G. Cem and T. E. Mehmet, "The Effect of the Normalization Method Used in Different Sample Sizes on the Success of Artificial Neural Network Model," *International Journal of Assessment Tools in Education* , vol. 6, no. 2, p. 170–192, 2019.
- [59] "Feature-Selection," [Online]. Available: <https://towardsdatascience.com/getting-data-ready-for-modelling-feature-engineering-feature-selection-dimension-reduction-39dfa267b95a>. [Accessed 2018].



- [60] k. Pavya and D. B. Srinivasan, "Feature Selection Techniques in Data Mining: A Study," *IJSDR*, vol. 2, no. 6, pp. 2455-2631, 2017.
- [61] M. W. Mwadulo, "A Review on Feature Selection Methods For Classification Tasks," *International Journal of Computer Applications Technology and Research*, vol. 5, no. 6, pp. 395 - 402, 2016.
- [62] [Online]. Available: [https://en.wikipedia.org/wiki/Recurrent\\_neural\\_network](https://en.wikipedia.org/wiki/Recurrent_neural_network).
- [63] [Online]. Available: <https://www.researchgate.net/figure/RNN-basic-structure-and-unfolding-in-time-according-to-the-number-of-sequence-elements>.
- [64] H. Seep and S. Jurgen, "Long Short- Term Memory," *Neural Computation*, 4 May 2016.
- [65] Vance and Ashlee, "Quote: These powers make LSTM arguably the most commercial AI achievement used for everything from predicting diseases to composing music," *Bloomberg Business Week*, 15 May 2018.
- [66] [Online]. Available: Understanding LSTM networks [http://colah.github.io/posts/2015-08-Understanding LSTM/](http://colah.github.io/posts/2015-08-Understanding-LSTM/).
- [67] O. C, "Understanding lstm networks," *GITHUB blog*, 27 August 2015.
- [68] [Online]. Available: [https://developer.ibm.com/articles/cc machine-learning and deep learning-architectures](https://developer.ibm.com/articles/cc-machine-learning-and-deep-learning-architectures).
- [69] P. Razvan, G. Caglar, C. Kyunghyun and B. Yoshua, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.
- [70] K. Greff, R. K. Srivastava, J. Koutník, R. Steunebrink and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, Oct 2017.
- [71] [Online]. Available: <https://pdf.co/blog/grid-search-for-hyper-parameter-selection-in-machine-learning>.
- [72] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *CoRR*, 2017.
- [73] E. N. Chigozie, I. Winifred, G. Anthony and M. Stephen, "Activation Functions :Comparison of Trends in Practice and Research for Deep Learning," *arxiv:1811.03378v1[cs.LG]*, 8 Nov, 2018.

- [74] K. Greff, K. Srivastava, K. I. J. R. Steunebrink B and J. Schmidhuber. [Online]. Available: [https://medium.com/datadriveninvestor/overview-of-different optimizers for-neural networks e0ed119440c3](https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3).
- [75] P. K. Diederik and L. B. Jimmy, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," in *Published as a conference paper at ICLR*, Amsterdam, 2017.
- [76] [Online]. Available: [https://towards datascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c](https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c).
- [77] N. rivastava, G. Hinton, E. Krizhevsky, I. A. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from over fitting," *Journal of Machine*, vol. 15(1), pp. 1929-1958, 2014.
- [78] K. V. Hanuman, "Python Library, Development Frameworks and Algorithms for Machine learning applications," *International journal of Engineering Research & Technolgy(IJERT)*, vol. 7, no. 04, 2018.

## APPENDICES

### Appendix A: Full Description of NSL\_KDD Dataset.

Table 5: Descriabtion and features of the NSL\_KDD dataset.

S/N	Feature Names	Description	Data types	Maximum value
1	Duration	Length of time duration of the connection		42908
2	Protocol_type	Protocol used in the connection	Symbolic	3 distinct values
3	Service	Destination network service used	Symbolic	70 distinct values
4	Flag	Status of the connection – Normal or Error	symbolic	11 distinct values
5	Src_bytes	Number of data bytes transferred from source to destination in single connection	Continuous	$1.3 \times 10^9(137996388)$
6	dst_bytes	Number of data bytes transferred from destination to source in single connection	Continuous	$1.3 \times 10^9(1309937401)$
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	Symbolic	
8	wrong_fragment	Total number of wrong fragments in this connection	Continuous	3
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	Continuous	3
10	Hot	Number of hot indicators in the content such as: entering a system directory, creating programs and executing programs	Continuous	77
11	num_failed_logins	Count of failed login attempts	Continuous	5
12	logged_in	1 if successfully logged in; 0 otherwise	Symbolic	5
13	num_compromised	Number of “compromised” conditions	Continuous	7479
14	root_shell	1 if root shell is obtained; 0 otherwise	Continuous	1
15	su_attempted	1 if “suroot” command attempted or used; 0 otherwise	Continuous	2
16	num_root	Number of <b>root</b> accesses or number of operations performed as a root in the connection	Continuous	7468
17	num_file_creations	Number of file creation operations in the connection	Continuous	43
18	num_shells	Number of shell prompts	Continuous	2
19	num_access_file	Number of operations on access control files	Continuous	9
20	num_outbound_cmds	Number of outbound commands in an ftp session	Continuous	0
21	is_host_login	1 if the login belongs to the “hot” list i.e. root or admin; else 0	Symbolic	1
22	is_guest_login	1 if the login is a “guest” login; 0 otherwise	Symbolic	1
23	Count	Number of connections to the same destination host as the current connection in the past two seconds.	Continuous	511
24	srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds	Continuous	511
25	Error_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	Continuous	1
26	error_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	Continuous	1
27	srv_error_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	Continuous	1
28	srv_error_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	Continuous	1
29	same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	Continuous	1

30	diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	Continuous	1
31	srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	Continuous	1
32	dst_host_count	Number of connections having the same destination host IP address	continuous	255
33	dst_host_srv_count	Number of connections having the same port number	Symbolic	255
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)	Continuous	1
35	dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	Continuous	1
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).	Continuous	1
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)	Continuous	1
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)	Continuous	1
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)	Continuous	1
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)	Continuous	1
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)	Continuous	1
42	Attack types/class	The attribute contains data about the various 5 classes of network connection vectors and they are categorized as <b>one</b> normal class and <b>four</b> attack class. The 4 anomaly attack classes are further grouped as DoS, Probe, R2L and U2R.	Labels (normal/anomaly)	Normal-67343 Anomaly-58630 on training data and 9711 normal and 12837 anomaly test data

### ➤ Three non-numeric attribute in the NSL-KDD dataset

@attribute 'protocol\_type' {'tcp','udp', 'icmp'}

@attribute 'service' {'aol', 'auth', 'bgp', 'courier', 'csnet\_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain\_u', 'echo', 'eco\_i', 'ecr\_i', 'efs', 'exec', 'finger', 'ftp', 'ftp\_data', 'gopher', 'harvest', 'hostnames', 'http', 'http\_2784', 'http\_443', 'http\_8001', 'imap4', 'IRC', 'iso\_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios\_dgm', 'netbios\_ns', 'netbios\_ssn', 'netstat', 'nnsp', 'nntp', 'ntp\_u', 'other', 'pm\_dump', 'pop\_2', 'pop\_3', 'printer', 'private', 'red\_i', 'remote\_job', 'rje', 'shell', 'smtp', 'sql\_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp\_u', 'tim\_i', 'time', 'urh\_i', 'urp\_i', 'uucp', 'uucp\_path', 'vmnet', 'whois', 'X11', 'Z39\_50'}

@attribute 'flag' {'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH'}

## Appendix B: Sample Source Code.

- Sample source code and important python library.

```
In [ ]: import os
import math
import sys
import operator
import requests
import warnings
import csv
import sklearn
%matplotlib inline
import keras as sk
import pandas as pd
import numpy as np
import seaborn as sn
import tensorflow as tf
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
from keras import regularizers
from sklearn import preprocessing
from keras.utils import np_utils
from keras.models import Sequential
from keras.utils import to_categorical
from keras.preprocessing import sequence
from sklearn.metrics import accuracy_score, make_scorer, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dense, Activation, LSTM, SimpleRNN, GRU, Dropout
from sklearn.feature_selection import f_classif, f_regression, SelectKBest, SelectPercentile
print(tf.__version__)
print(np.__version__)
print(pd.__version__)
print(sklearn.__version__)
print(sk.__version__)
print(sn.__version__)
```

```
#Import the dataset using pandas as reading a file
df = pd.read_csv("NSL_KDD_Dataset/csv/KDDTrain+.csv", header=None, names = col_names)
df_test = pd.read_csv("NSL_KDD_Dataset/csv/KDDTest+.csv", header=None, names = col_names)
# shape, this gives the dimensions of the dataset
print('Dimensions of the Training set:', df.shape)
print('Dimensions of the Test set:', df_test.shape)
##Handling Missing Data using imputation techniques
#checks the null data and not null data both the training and testing data of the NSL_KDD dataset
df.isnull()
df_test.isnull()
#We can fill NAN entries with a single value, such as zero:
#df.fillna(0)
#df_test.fillna(0)
#Transform categorical features into numbers using Label Encoder
df_categorical_values_enc=df_categorical_values.apply(Label Encoder().fit_transform)
print(df_categorical_values_enc.head())
```

```

# use One-Hot-Encoding systems to encode the data
enc = OneHotEncoder ()
df_categorical_values_encenc = enc.fit_transform(df_categorical_values_enc)
df_cat_data = pd. DataFrame (df_categorical_values_encenc.to array (), columns=dumcols)
df_cat_data. head ()
#Use StandardScaler () to scale the 4 types of anomaly attacks on the dataframes
from sklearn import preprocessing
scaler1 = preprocessing. StandardScaler (). fit(X_DoS_train)
X_DoS_train=scaler1.transform(X_DoS_train)
scaler2 = preprocessing. StandardScaler (). fit(X_Probe_train)
X_Probe_train=scaler2.transform(X_Probe_train)
scaler3 = preprocessing. StandardScaler (). fit(X_R2L_train)
X_R2L_train=scaler3.transform(X_R2L_train)
scaler4 = preprocessing. StandardScaler (). fit(X_U2R_train)
X_U2R_train=scaler4.transform(X_U2R_train)
#####
#Step 3: Feature Selection for 4 anomaly attack catogries (Dos, U2R, R2R AND PROBE)
## using different feature selection mechanisms, selectKBest, f_classify(percentile) np. steer
(divide='ignore', invalid='ignore');
selector=SelectPercentile (f_classif, percentile=10)
# Create a SelectKBest features
sub = SelectKBest (score_func=f_classif, # Set f_classif as our criteria to select features
                  k=22) # Select top 22 features based on the criteria

# reshape input to be the form [samples, time steps, features]
# reshape the training and test data(DoS) before starting to build the RNN, LSTM and GRU model
X_DoS_train = np. reshape (X_DoS_train, (113270, 1, 122))
X_DoS_test = np. reshape (X_DoS_test, (17171, 1, 122))
#crate the simpleRNN model to train and test the anomaly attacks (DoS, Probe, U2R, R2L)
##create and fit the network, optimizer=Adam, neurons, dropout
model = Sequential ()
model. add (SimpleRNN (64, return_sequences=True, input_shape= (1, 122))) # hidden layer 1
model.add (Dropout (0.1))
model.add (SimpleRNN (64, return_sequences=True)) #hidden layer 2
model.add (Dropout (0.1))
model.add (SimpleRNN (64, return_sequences=True)) # hidden layer 3
model.add (Dropout (0.1))
model.add (SimpleRNN (64, return_sequences=False)) #hidden layer4
model.add (Dropout (0.1))
model.add (Dense (1))
model.add(Activation('sigmoid'))
## model compile.....
model. compile (optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model. Summary ()
# fit the model...
history1 = model. Fit (X_DoS_train, Y_DoS_train, X_Probe_train, Y_Probe_train X_U2R_train, Y_U2R_train
X_R2L_train, Y_R2L_train batch_size=500, nb_epoch=100, verbose=1, Test= (X_U2R_test, Y_U2R_test))
## evaluate the model....
#print ("Accuracy: %.2f%%" % (scores [1] *100))
#print ("test_data_acc: %.2f%%" % (scores [1] *100))
plt. figure (0)

```

```

plt.plot(history1.history['acc']) #,r')
plt.label ("Numb of Epochs")
plt.label ("Train and Test accuracy")
plt.title (" simple RNN model U2R, DoS, Probe, R2L Attack")
plt.legend(['X_U2R_train','X_U2R_test','X_DoS_train','X_DoS_test','X_Probe_train','X_Probe_test','X_R2L_train','X_R2L_test']) #, loc='midel left')
plt.show ()

```

## Appendix C: List of Selected Features for the NSL\_KDD dataset.

Table 6: List of top most selected features on the dataset.

Attack types	Number of minimal (selected 22 features) on the dataset
DoS	- logged_in, - count, - serror_rate, - srv_serror_rate, - rerror_rate, - srv_rerror_rate, - same_srv_rate, - srv_diff_host_rate, - dst_host_count, - dst_host_srv_count, - dst_host_same_srv_rate, - dst_host_serror_rate, - dst_host_srv_serror_rate, - dst_host_rerror_rate, - dst_host_srv_rerror_rate, - Protocol_type_udp, - service_harvest, - service_pm_dump, - service_sql_net, - service_urp_i, - service_whois, - flag_RSTOS0.
Probe	- logged_in, - serror_rate, - srv_serror_rate, - rerror_rate, - srv_rerror_rate, - same_srv_rate, - diff_srv_rate, - srv_diff_host_rate, - dst_host_srv_count, - dst_host_same_srv_rate, - dst_host_diff_srv_rate, - dst_host_serror_rate, - dst_host_srv_serror_rate, - dst_host_rerror_rate, - dst_host_srv_rerror_rate, - service_harvest, - service_http, - service_netstat, - service_sql_net, - service_urp_i, - service_uucp, - flag_RSTOS0.
U2R	- urgent, - hot, - num_failed_logins, - num_compromised, - root_shell, - num_root - num_file_creations, - num_shells, - num_access_files, - is_host_login - dst_host_srv_count, - dst_host_same_srv_rate, - dst_host_diff_srv_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, - service_finger, - service_ftp - service_harvest, - service_sql_net.
R2L	- num_failed_logins, - logged_in, - is_guest_login, - count, - srv_count - rerror_rate, - srv_rerror_rate, - srv_diff_host_rate, - dst_host_count - dst_host_srv_count, - dst_host_same_srv_rate, - dst_host_rerror_rate, - dst_host_srv_rerror_rate, - service_domain, - service_finger, - service_ftp, - service_harvest, - service_netstat, - service_ntp_u, - service_sql_net, - service_urp_i, - flag_RSTOS0.

## Appendix D: Capture of Sample Training and Testing

- These are the first epoch (1) and last epochs (100) sample training and testing of attack on SimpleRNN model.

```
#crate the simpleRNN model to train and test the anomoly attacks(Dos,prope, u2r, r2l)
model = Sequential()
model.add(SimpleRNN(64, return_sequences=True, input_shape=(1, 122))) # hidden Layer 1
model.add(Dropout(0.1))
model.add(SimpleRNN(64,return_sequences=True)) #hidden Layer 2
model.add(Dropout(0.1))
model.add(SimpleRNN(64,return_sequences=True)) #hidden Layer 3
model.add(Dropout(0.1))
model.add(SimpleRNN(64, return_sequences=False)) #hidden Layer 4
model.add(Dropout(0.1))
model.add(Dense(1))
model.add(Activation('sigmoid'))

## model compile....
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.summary()
print("Train attacks using simpleRNN.")
print("Test attacks using simpleRNN.")
## fit the model...
print("Train and Test DoS attack using SimpleRNN.")
history1 = model.fit(X_DoS_train, Y_DoS_train, batch_size=500, nb_epoch=100, verbose=1, test_data=(X_DoS_test, Y_DoS_test))
print(" Train and Test Probe attack using SimpleRNN.")
history2 = model.fit(X_Probe_train, Y_Probe_train, batch_size=500, nb_epoch=100, verbose=1, test_data=(X_Probe_test, Y_Probe_test))
print(" Train and Test U2R attack using SimpleRNN.")
history3 = model.fit(X_U2R_train, Y_U2R_train, batch_size=500, nb_epoch=100, verbose=1, test_data=(X_U2R_test, Y_U2R_test))
print(" Train and Test R2L attack using SimpleRNN.")
history4 = model.fit(X_R2L_train, Y_R2L_train, batch_size=500, nb_epoch=100, verbose=1, test_data=(X_R2L_test, Y_R2L_test))
```

Epoch 1/100

113270/113270 [=====] - 19s 168us/step - loss: 0.0901 - acc: 0.9709 - val\_loss: 0.5624 - test\_acc: 0.8769

Epoch 100/100

113270/113270 [=====] - 10s 87us/step - loss: 9.1393e-04 - acc: 0.9994 - val\_loss: 0.6909 - test\_acc: 0.9157

Epoch 1/100

78999/78999 [=====] - 8s 101us/step - loss: -1.8839 - acc: 0.8286 - val\_loss: -1.3237 - test\_acc: 0.7602

Epoch 100/100

78999/78999 [=====] - 8s 99us/step - loss: -2.3220 - acc: 0.8508 - val\_loss: -1.4449 - test\_acc: 0.7623

Epoch 1/100

67395/67395 [=====] - 6s 84us/step - loss: 0.0835 - acc: 0.9932 - val\_loss: 0.3261 - test\_acc: 0.9642

Epoch 100/100

67395/67395 [=====] - 5s 77us/step - loss: -0.0250 - acc: 0.9989 - val\_loss: 0.1574 - test\_acc: 0.9859

Epoch 1/100

68338/68338 [=====] - 6s 82us/step - loss: -0.1505 - acc: 0.9750 - val\_loss: 4.9651 - test\_acc: 0.7490

Epoch 100/100

68338/68338 [=====] - 5s 79us/step - loss: -0.4189 - acc: 0.9841 - val\_loss: 7.7986 - test\_acc: 0.7647



```

from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
# Function to create model, required for KerasClassifier
def create_model():
    —# create model
    —model = Sequential()
    —model.add(SimpleRNN(64, input_dim=122, activation='sigmoid'))
    —model.add(Dense(1, activation='sigmoid'))
    —# Compile model
    —model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    —return model
# create model
model = KerasClassifier(build_fn=create_model, verbose=0)
# define the grid search parameters
batch_size = [100, 500, 1000]
epochs = [50, 100]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X_DoS_train, Y_DoS_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

Best: 0.999421 using {'batch_size': 500, 'epochs': 100}
0.999184 (0.000303) with: {'batch_size': 100, 'epochs': 50}
0.999050 (0.000236) with: {'batch_size': 100, 'epochs': 100}
0.999154 (0.000333) with: {'batch_size': 500, 'epochs': 50}
0.999421 (0.000145) with: {'batch_size': 500, 'epochs': 100}
0.999080 (0.000210) with: {'batch_size': 1000, 'epochs': 50}
0.999406 (0.000128) with: {'batch_size': 1000, 'epochs': 100}

```