Media Engineering and Technology Faculty

German University in Cairo

Machine Learning in Cloud Security I

Bachelor Thesis

Author: Yahia Mostafa Abbas

Supervisors: Dr.Eng Maggie Mashaly

Eng. Ahmed Abd El-Khaleq

Submission Date: 11 June, 2022

Media Engineering and Technology Faculty

German University in Cairo

Machine Learning in Cloud Security I

Bachelor Thesis

Author: Yahia Mostafa Abbas

Supervisors: Dr.Eng Maggie Mashaly

Eng. Ahmed Abd El-Khaleq

Submission Date: 11 June, 2022

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Yahia Mostafa Abbas

11 June, 2022

Acknowledgments

First and foremost, I'd want to thank my supervisor, Dr. Maggie, whose guidance and input helped me come up with new and better ideas. Your positive attitude and belief in our skills inspired me to succeed. I would also want to thank my co-supervisor, Eng. Ahmed Abd el-khaleq, for her patience and great effort to be of assistance to us at all times with her expertise and extremely good recommendations.

I'd want to thank Ahmed Tamer and Mosab Morgan from my study team for being so helpful, positive, with kind their ideas.

Abstract

Every day, attackers become more powerful than the day before because they develop more sophisticated and smarter methods of hacking and gathering information about the victim. Intelligent cybersecurity solutions have also been critical in improving defence against harmful attacks. One of the most important cybersecurity systems for protecting against attacks is intrusion detection systems (IDS). In this paper we will propose a different deep learning models in order to achiever a very high accuracy. We will propose two Multi-layer Perceptron (MLP) Models , two-dimensional Convolutional Neural Networks (CNN2D) and one-dimensional (CNN1D). As a result, we recommend training this model on the whole NSL-KDD benchmark dataset without utilising any feature selection techniques. We investigate the model's performance in binary classification and multi-class classification. We compare our models with various machine learning models such as NB tree, random tree and J48 and we also compare our models with different deep learning models such as GRU, RNN, CNN1D and BAT. The experimental results indicate that our models have a high accuracy compared to other models. CNN1D achieving a very high accuracy on KDDTest+ 91.4 % which is higher than any method used by difference 4.1 %, and 87.8 % in KDDTest-21 which is also higher than any method by 2.86 %. Our proposed MLP2 model achieving a very high accuracy on KDDTest+ 86.56 % which is higher than any method used by difference 2.31 %, and 80.09 % in KDDTest-21 which is also higher than any method by 10.67 %

Contents

List of Tables

List of Figures

List of Terms

AC Accuracy

AE Autoencoder

AF Activation Function

AI Artificial Intelligence

AIDS Anomaly Intrusion Detection System

ANN Artificial Neural Networks

BLSTM Bidirectional Long Short-Term Memory

CGATN Conditional Generative Adversarial Networ

Colab Colaboratory

ConvNets Convolutional Networks

DL Deep Learning

DQL Depp Q Learning

DR Detection Rate

DT Decision Tree

DoS Denial of Service attack

FC Fully Connected Layer

FN False Negative

FP False Positive

FPR False Positive Rate

GRU Gated Recurrent Unit

ICMP Internet Control Message Protocol

IDS Intrusion Detection System

IPS Intrusion Prevention System

MALWARE Malicious Software

ML Machine Learning

MLP MultiLayer Perceptron

NBC Naive Bayes Classifiers

NSL-KDD Network Security Laboratory Knowledge Discovery in Databases

OHE One-Hot Encoder

PCA Principle Component Analysis

Probe Proving attack

R2L Root to Local attack

RL Reinforcement Learning

RNG Random Number Generator

RNN Recurrent Neural Network

ReLU Rectified Linear Unit

SDN Software-Defined Networking

SVM Support Vector Machine

TCP Tranimission Control Protocol

TN True Negative

TP True Positive

U2R User to Root attack

UDP User Datagram Protocol

tanh Tangent Hyperbolic

Chapter 1

Introduction

1.1 Motivation

Everyday, the attackers become more stronger than the day before because they are developing more complex and intelligent ways to hack and gather information about the target. A bad hacker can cause a lot of damage stealing sensitive information, intercepting data and a lot more, so the network security is the most important aspect in banks, organizations and Government Departments. Also an intelligent cybersecurity systems have been very important to improve the protection against malicious threats. IDS is considered as one of the most important cybersecurity system to protect against threats. Hacker are inventing a new methods for attacking, so we cannot depend only on a static database to prevent the attacks. Here, IDS is playing an important role to detect the anomaly threats and provide security for the network.

We can categorize network traffic into two main classes (normal , anomaly). And we can divide the anomaly class into sub-classes such as DoS (Denial of Service attacks), Probe (Probing attacks), R2L (Root to Local attacks) and U2R (User to Root attacks), so IDS will be used to classify the traffic type either into the two main classes or to the five classes. We need to get a high accuracy in classification in order to make our system more robust to attacks.

1.2 Aim Of The Thesis

In this thesis, we aimed to build an anomaly intrusion detection system (AIDS) to detect the anomaly attacks. We proposed and test n deep learning models. We have two models of Multilayer Perceptron (MLP), two-dimensional Convolutional Neural Networks (CNN2D), one-dimensional (CNN1D). We ust the NSl-KDD dataset for training and testing our models. Then we tune our Hyperparameters to obtain the best performance in both binary and five-category classification.

1.3 Thesis Outline

This thesis is divided into five chapters, one of which is an introduction to the motivation for this study and the goal of the thesis. Chapter 2 background, in this chapter includes an introduction to the cloud computing, introduction to malware types, introduction to a machine learning and its different classes and an introduction to the NSL-KDD dataset. Chapter 3 methodology that describes the work phases in depth, including the data preprocessing step, model construction techniques, and performance measures used to assess the model. Chapter 4 Results will contain our model outputs as well as comparisons to the state of the art across several criteria. Chapter 5 conclusion covers the end of current work as well as a preview of future work.

Chapter 2

Background

This chapter will include an introduction to the cloud computing, introduction to malware types, introduction to a machine learning and its different classes and an introduction to the NSL-KDD dataset.

2.1 Cloud Computing

Cloud Computing is considered as one of the most important technologies which allow to store a large amount of data, backup, create applications etc... Cloud computing has a great impact on our lives as it improves the production efficiency in many areas.[1] Without cloud computing services each company should have a server room (routers, switches, firewalls, servers and engineers for maintenance) which cost a lot of money. Cloud computing has a lot of Properties such as:

- **Security** : Cloud service providers use the most efficient technology to protect data and prevent attacks.
- **Low cost** : Small companies don't need to have its own server room.

- **Multi-sharing** : Many users can share information with each other effectively using cloud computing.
- **Ease of access**: Anyone can simply access services or applications on the cloud using Application programming interface (API).
- **Agility** : Cloud share a lot of resources among users so it can run very fast (distributed computing environment).

There are four models of cloud computing that the organization can deploy according to its needs. And the four types are:

2.1.1 Public Cloud

Public cloud computing portrays computing that makes resources accessible to the common open or enormous industry groups over the Web, with the utilize of web applications/ web services. Google, Amazon and Microsoft are examples of public cloud service providers. The cloud service providers embrace full responsibility of establishment, administration and support.[2]

2.1.2 private Cloud

Private cloud, also called internal cloud, is within the internal structures of cloud computing infrastructure and provide cloud computing services to the internal users which gives more security than the public one. And it's used by organizations to control their own data centers.[3]

2.1.3 Hybrid Cloud

Hybrid cloud is a combination between public and private. It takes the economies and efficiencies from public and from the private it takes security and control.[4]

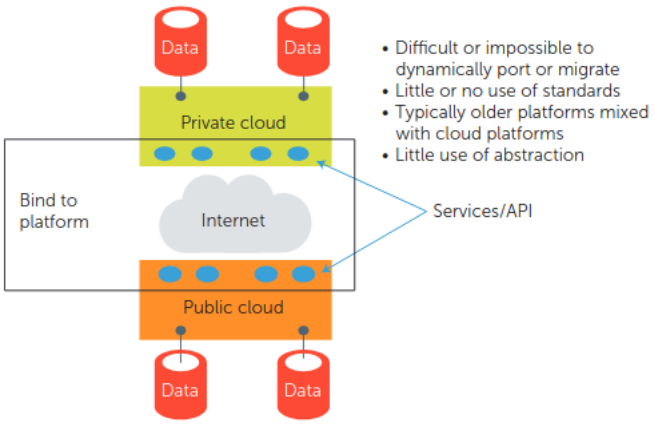


Figure 2.1: Hybrid cloud illustrative image

2.1.4 Community Cloud

Community Cloud is built collaboratively using the resources contributed by the users. The community cloud infrastructure are outlined with the necessities of the community of the clients they are to serve, so as to empower commitment and interest by the community members.[5]

2.2 MALWARE TYPES

Malicious software (or malware) is defined as software that carries out an attacker's malicious intent, and it is one of the most serious security risks that the Internet faces today. When it comes to malware analysis, you'll find that generating informed predictions about what the Malware is trying to do and then validating those theories may help you save time. So, In this section we will cover some Malware types.[6]

2.2.1 Backdoor Malicious

Code that installs itself on a computer to grant access to the attacker. Backdoor often allow an attacker to get access to a computer with little or no authentication and run commands on the local system.

2.2.2 Botnet

In that it gives the attacker access to the system, it's similar to a backdoor, but all machines infected with the same botnet get the same commands from a single command-and-control server.

2.2.3 Downloader Malicious code

That exists only for the purpose of downloading further malicious programs. When an attacker initially gains access to a system, they frequently install downloaders. Additional harmful malware will be downloaded and installed by the downloader tool.

2.2.4 Information-stealing malware

Malware that gathers data from the victim's computer and transfers it to the attacker. Sniffers, password hash grabbers, and key-loggers are just a few examples. This virus is commonly used to obtain access to internet accounts including email and banking. Launcher is a rogue software that allows other malicious programs to run. In order to assure stealth or broader access to a system, launchers typically employ innovative approaches to launch other malicious applications.

2.2.5 Rootkit Malicious code

The purpose of this code is to hide the existence of other code. Rootkits are frequently combined with additional malware, such as a backdoor, to provide the attacker remote access and make the code harder to detect by the victim.

2.2.6 Spam-sending malware

Malware that infects a user's computer and then sends spam from that computer. This software allows attackers to make money by allowing them to offer spam-sending services.

2.2.7 Scareware Malware

Intended to scare an infected user into purchasing something Its user interface is frequently designed to resemble that of an antivirus or other security tool. It notifies customers that their system is infected with malicious malware and that the only way to remove it is to purchase their ”software,” whereas the software it sells does nothing more than delete the scareware.

2.2.8 Virus

Malicious code that can replicate itself and infect other systems.

2.3 Network Attacks

Network attacks are unauthorized actions to perform a malicious actions such as steal information , destroy files , take the network down or gain access to internal systems. And there are different types of network attacks such as:

2.3.1 Denial of Service

A denial of service (DOS) attack is an attack where the attacker tries to overwhelm a machine by sending too much traffic (Making Computer resources or memory full) which the machine can’t handle or sending a small script when the machine executes this script it causes the machine to be out-of-service like XML bomb.

2.3.2 Probe Attack

A probe is an assault that is designed to be detected and reported by its target, leaving a distinctive ”fingerprint” in the report. From this information, the attacker then utilities the collaborative infrastructure to understand the detector’s position and defensive capabilities.

2.3.3 User to Root attack

User to root attacks (U2R) are exploitation attacks in which the hacker logs in with a normal user account and attempts to exploit system flaws in order to achieve super user privileges.

2.3.4 Remote to user attack

Remote to user (R2L) assaults are a form of computer network attack in which an intruder delivers a group of packets to another computer or server via a network that he or she does not have authority to access as a local user.

2.4 Network Defense Technologies

Network defense technologies are different types of mechanism to add an extra layer of security to protect your network against attacks such as:

2.4.1 Intrusion Detection System

The Intrusion Detection System (IDS) is a security tool that can monitor system intrusion activity in real time.IDS uses a combination of hardware and software to detect network threats and protect systems against unauthorised access.[7] It can identify and assess whether system users exceed their privileges and whether there are intruders who exploit system security weaknesses to conduct intrusion on the system by monitoring the status, behaviour, and usage of the network system.

2.4.2 Intrusion Prevention System

Intrusion Prevention System (IPS) is a security tool that can monitor system intrusion activity in real time and it has an advantage over intrusion detection system (IDS) which it can take an action towards the attacks by preventing and blocking it.[8]

2.4.3 Honeypot

Honeypots are a powerful network security concept. The goal of such a system is to gather information regarding intrusion attempts or resource intrusions. This information might vary greatly, for example, time, date, intruder IP address, intruder operating system, or used keywords, exploits, and instructions after penetration.[9]

2.5 Machine Learning

Machine learning (ML) is about to teach a computer to do a specific task and improve the performance without being explicitly programmed to do this task. Machine learning is considered a branch from Artificial intelligence (AI). Machine learning (ML) is used in IDS / IPS in order to detect network attacks. Machine learning has different methods such as:

2.5.1 Supervised Machine learning

According to Tom Mitchell supervised machine learning is ”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”.

- Decision Tree (DT): A tree where each node could represent a state or attribute to be tested and the leaves represent a classification. Classification started from the root and attribute testing happens at each node until reaching the leaf which provides the decision.[10]
- Naïve Bayes: A classifier uses Bayes theorem to calculate the incoming probability based on past experience. It’s important characteristic is its strong assumption of independence of each feature.[11]
- Support Vector Machine (SVM): Support vector machine (SVM) is a classification method used in machine learning to deal with both linear and nonlinear data. SVM is based on statistics theory and data analysis.[12]

2.5.2 Unsupervised Machine Learning

Unsupervised Machine learning algorithms are usually used with unlabeled date which doesn't has a correct answer and there isn't a previous data to be used as a guide. Those algorithms are trying to make a structure about its data. When applying the algorithm to a new data it makes a new structure based on the old structure. The main purpose of the unsupervised machine learning algorithms is to group similar data.

- K-Means : K-Means is considered the most easiest unsupervised machine learning algorithm and the most widely used. The key of this algorithm is to cluster data into K clusters where each cluster has many common features. The main idea is to choose K centroids away from each other because their initial position will determine the shape of the cluster.[13]
- Principal Component Analysis (PCA): It's considered as one of the most used algorithms for dimensions reduction with minimum mean squared error.[14]

2.5.3 Artificial Neural Network

Artificial neural network (ANN) is a series of algorithms try to mimic the human brain in how to make a decision , create pattern and process data. ANN is widely used because it has the ability to model a linear and non-linear model by using the earlier pattern collected about this system. ANN can achieve a high success if it has the optimal parameters which are activation function (AF) and weights. ANN is based on nodes called (Neurons) and it has layers , input layer , output layer and hidden layers in between.[15]

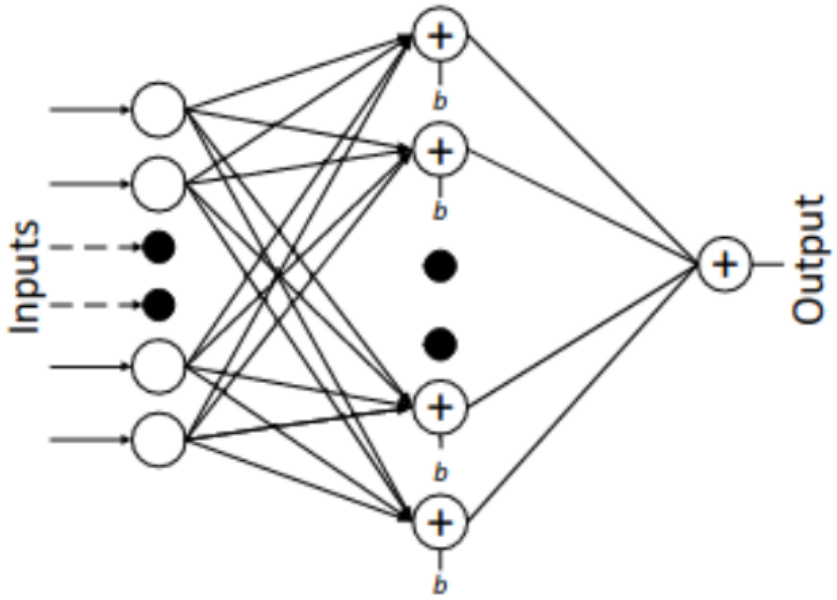


Figure 2.2: ANN components

2.5.4 Deep Learning

Deep learning (DL) is considered as a subset from machine learning which use a mathematical equation to get useful information or patterns from the data and try to link the input by the output. Deep learning can handle a huge amount of data which the human brain can't handle and DL has a performance and accuracy much butter than the human brain. DL takes place in many industries such as agriculture , business and network security.

2.6 NSL-KDD Dataset

NSL-KDD stands for(Network Security Laboratory Knowledge Discovery in Databases) is an offline (shared dataset which I can make some changes to it) dataset. Datasets play a great role on effecting the algorithm performance which leads to a very bad estimation, so NSL-KDD is considered as a solution for this problem. NSL-KDD is created from different datasets by choosing some records and it has some proprieties such as:

1. There are no redundant records in the training so the classifier isn't biased.

2. It has 41 features.

3. It has 21 different types of attacks can be grouped into five classes.

The main five classes are (Normal , DOS , probe , R2L , U2R) are discussed earlier.

Table 2.1: Mapping the 21 types into the five main classes.

Normal	Normal
Dos	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb ,Processtable, Udpstorm, Apache2, Worm
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

NSL-KDD dataset has three main files :

1. KDDTrain+.csv: the file we will use to train our algorithm.

2. KDDTest+.csv: the file used for testing our algorithm.

3. KDDTest-21.csv: A subset of the KDDTest+ which does not include records with difficulty level of 21 out of 21

Table 2.2: NSL-KDD datasets.

Dataset File	Normal	Dos	Probe	R2L	U2R	Total
KDDTrain+	67342	67342	11656	995	52	125972
KDDTest+	9711	7457	2421	2754	200	22542
KDDTest-21	2512	4242	2402	2754	200	11849

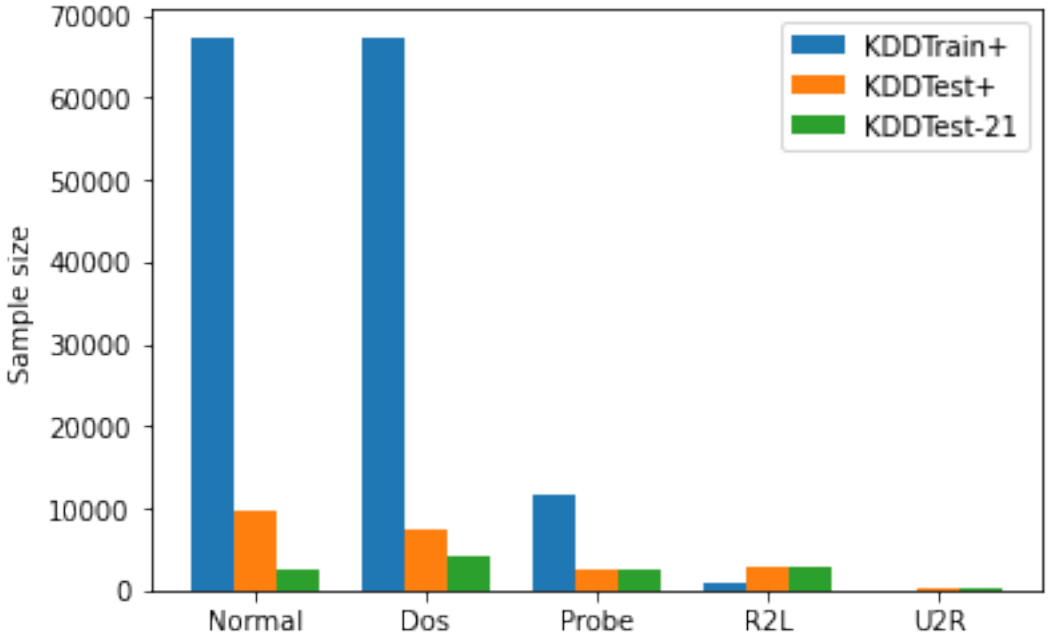


Figure 2.3: The distribution of Main classes in NSL-KDD datasets

Chapter 3

Related Work

In this chapter, We are going to discuss the work done on some research papers regarding IDS based on some deep learning models and machine learning models trained in the NSL-KDD dataset. We are going to compare our results with their results.

3.0.1 IDSs based on Machine Learning Algorithms

In [16], the authors tried different machine learning algorithms such as ANN, SVM, NBC, random forest usnig NSL-KDD dataset achieving 78.51 % in multi classification as the highest accuracy using ANN.

In [17], the authors introduce a new method to balance data and evaluate their influence on various ML classifiers, researchers used a synthetic data creation approach called Conditional Generative Adversarial Network (CTGAN). The method trained in NSL-KDD and achieved 77.62 % in multi classification as the highest accuracy..

In [18], the authors introduced a new generation of network intrusion detection methods in this research, which integrate Q-learning based reinforcement learning with a deep feed forward neural network technique for network intrusion detection. A Deep Q learning (DQL) trained in NSL-KDD dataset and achieved 78 % in multi classification as the highest accuracy.

In [19], the authors applied a comparison of traditional machine learning classification techniques was undertaken to categorise network traffic. Machine learning algorithms trained in NSL-KDD dataset and achieved 78.69 % in binary classification as the highest accuracy.

In [20], the authors performed a comprehensive study on NSL-KDD by visualizing patterns and utilizing distinctive learning-based models and used a SVM-SMOTE to solve the problem of imbalance issue. Autoencoder treat the routine binary classification issue as an anomaly detection issue. It learns the representation of normal samples and employments the recreation error as the anomaly score. Achieving 87.52 % in binary classification and 80.47 % in multi classification.

In [21], the authors proposed a novel contextual discounting model based on the differentiation between the normal and anomaly behavior. The model trained in NSL-KDD dataset and achieved accuracy 80.62 % on KDDTest+ and 81.84 % on KDDTest-21 in binary classification.

3.0.2 IDSs based on Machine Deep Algorithms

In [22], the authors proposed an IDS to extract optimized and more correlated features using big data visualization and statistical analysis methods followed by an autoencoder (AE) for threat detection. The authors also proposed a Multi-Layer Perception MLP model consisting of two layers. The proposed MLP achieved 81.6 % in multi classification and the proposed AE 87 % in multi classification.

In [19], the authors proposed a Multi-Layer Perception (MLP) model achieving maximum accuracy of 79.6 % in binary classification. In [23], the authors proposed a MLP model achieving maximum accuracy of 83.27 %.

In [24], the authors proposed a new approach o use three learning techniques in parallel: gated recurrent unit (GRU), convolutional neural network as deep techniques and random forest as an ensemble technique. Achieving 87.28 % accuracy on the KDDTest+ dataset and 76.61 % accuracy on KDDTest-21.

In [25], the authors proposed a model which consists of two concatenated ConvNets and is built on a two-stage learning process: learning a base dataset and transferring the learned knowledge to the learning of the target dataset. Achieving 87.30 % accuracy on the KDDTest+ dataset and 84.94 % accuracy on KDDTest-21.

In [26], the authors proposed a deep learning approach for flow-based anomaly detection in an SDN environment. The model build for intrusion detection system, trained the model with the NSL-KDD dataset and tested it using the test+ dataset. The author's claim gave an accuracy of 75.75 % using only six basic features.

in [27], the authors proposed a deep learning approach for intrusion detection system using recurrent neural networks (RNN-IDS). The model is trained and tested using NSL-KDD dataset. Achieving 83.28 % accuracy on the KDDTest+ dataset and 68.55 % accuracy on KDDTest-21 in binary classification and 81.29 % accuracy on the KDDTest+ dataset and 64.67 % accuracy on KDDTest-21 in multi classification.

In [28], the authors proposed to use the whole NSL-KDD dataset to train an IDS model based on Convolution Neural Networks (CNN), a common deep learning approach. Achieving 80.13 % accuracy on the KDDTest+ dataset and 62.32 % accuracy on KDDTest-21 in binary classification.

In [29], the authors proposed BLSTM (Bidirectional Long Short-Term Memory) and attention mechanisms are combined in the BAT model in order to address the issues of low detection accuracy and feature engineering in intrusion detection, Achieving 80.13 % accuracy on the KDDTest+ dataset and 62.32 % accuracy on KDDTest-21 in binary classification.

Chapter 4

Methodology

This chapter describes and illustrates the effort that went into building our deep learning-based IDS. Section one begins with procedures to prepare the dataset. The strategies employed in this thesis are discussed in depth in part two. The hyper-parameter tuning results are shown in section three. Finally, the measure applied in the model assessment procedure is shown in section four.

4.1 Data Preprocessing

We can't apply any model to NSL-KDD dataset without data preprocessing, so we should prepare our dataset to be used in the model. NSL-KDD dataset has three symbolic features out of the 41 features, so first we have to convert the symbolic features into a numerical ones by encoding and we should normalize our data for better performance and running time.[30]

Table 4.1: Features Description

features name	Feature type	description
Duration	numerical	connection duration.

Protocol_type	Symbolic	protocol used in the connection.
Service	Symbolic	service used by destination network .
Flag	Symbolic	status of the connection – normal or error.
Src-bytes	numerical	number of data bytes transferred from source to destination in single connection.
Dst-bytes	numerical	number of data bytes transferred from destination to source in single connection.
Land	numerical	if source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0.
Wrong-fragment	numerical	total number of wrong fragments in this connection.
Urgent	numerical	number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated.
Hot	numerical	number of hot indicators in the content such as entering a system directory, creating programs and executing programs.
Num-failed-logins	numerical	count of failed login attempts.
Logged-in	numerical	login Status : 1 if successfully logged in and 0 otherwise.
Num-compromised	numerical	number of compromised conditions.
Root-shell	numerical	1 if root shell is obtained and 0 otherwise .
Su-attempted	numerical	1 if “su root” command attempted or used and 0 otherwise.
Num-root	numerical	number of root accesses or number of operations performed as a root in the connection.
Num-file-creations	numerical	number of file creation operations in the connection.
Num-shells	numerical	number of shell prompts .
Num-access-files	numerical	number of operations on access control files .

Num-outbound-cmds	numerical	number of outbound commands in an ftp session.
Is-hot-login	numerical	1 if the login belongs to the hot list i.e., root or admin and 0 otherwise.
Is-guest-login	numerical	1 if the login is a “guest” login and 0 otherwise.
Count	numerical	number of connections to the same destination host as the current connection in the past two seconds .
Srv-count	numerical	number of connections to the same service (port number) as the current connection in the past two seconds .
Serror-rate	numerical	the percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23).
Srv-serror-rate	numerical	the percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv-count (24).
Rerror-rate	numerical	the percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23) .
Srv-rerror-rate	numerical	the percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv-count (24).
Same-srv-rate	numerical	the percentage of connections that were to the same service, among the connections aggregated in count (23) .
Diff-srv-rate	numerical	the percentage of connections that were to different services, among the connections aggregated in.
Srv-diff-host-rate	numerical	the percentage of connections that were to different destination machines among the connections aggregated in srv-count (24).
Dst-host-count	numerical	number of connections having the same destination host IP address .

Dst-host-srv-count	numerical	number of connections having the same port number .
Dst-host-same-srv-rate	numerical	the percentage of connections that were to the same service, among the connections aggregated in dst-host-count (32) .
Dst-host-diff-srv-rate	numerical	the percentage of connections that were to different services, among the connections aggregated in dst-host-count (32) .
Dst-host-same-src-port-rate	numerical	the percentage of connections that were to the same source port, among the connections aggregated in dst-host-srv-count (33) .
Dst-host-srv-diff-host-rate	numerical	the percentage of connections that were to different destination machines, among the connections aggregated in dst-host-srv-count (33).
Dst-host-serror-rate	numerical	the percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst-host-srv-count (32).
Dst-host-srv-serror-rate	numerical	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).
Dst-host-rerror-rate	numerical	the percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst-host-srv-count (32) .
Dst-host-srv-rerror-rate	numerical	the percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst-host-srv-count (33).

4.1.1 Binary Encoding

Binary encoding is an encoder to map a symbolic data into a numerical form. We have three symbolic data protocol_type , service and flag. And if we apply binary encoding to the categorical features , it will encode every column individually. For example in the protocol_type feature we have only three unique classes which are Tranimission control protocol (TCP) , User datagram protocol (UDP) and Internet control message protocol (ICMP), The encoder will give every class a unique decimal number so for example TCP will take 0 , UDP will take 1 and ICMP will take 2. then it will convert the decimal numbers given to the three classes to the binary. Because we have only three classes that will create two columns using this equation:

$$cols = \lceil log_2(sc) \rceil \tag{4.1}$$

where cols is the number of columns created for every symbolic feature after applying binary encoding and sc is the number of unique classes in this feature.

Table 4.2: Binary Encoding Example

Date Entry	Decimal Representation	Protocol_type1	Protocol_type0
TCP	0	0	0
UDP	1	0	1
ICMP	2	1	0

4.1.2 One-Hot Encoding

One-Hot encoder is an another way to encode symbolic features into numerical. Taking Protocol_type as an example, we have three classes each class will be mapped into a three-dimensional binary vector, service will be converted into 70 features and flag will be converted into 11 features, so after encoding all categorical features our dataset will have 122 features.

$$cols = unq \tag{4.2}$$

where col is the number of columns after encoding and unq is the number of unique classes in each feature.

Table 4.3: One-Hot Encoding Example

Date Entry	TCP	UDP	ICMP
TCP	1	0	0
UDP	0	1	0
ICMP	0	0	1

Using different encoder will result in different accuracy, For example if we apply binary encoder and one-hot encoder to the same result that will give us different accuracy between the two models, As shown in the figure below.

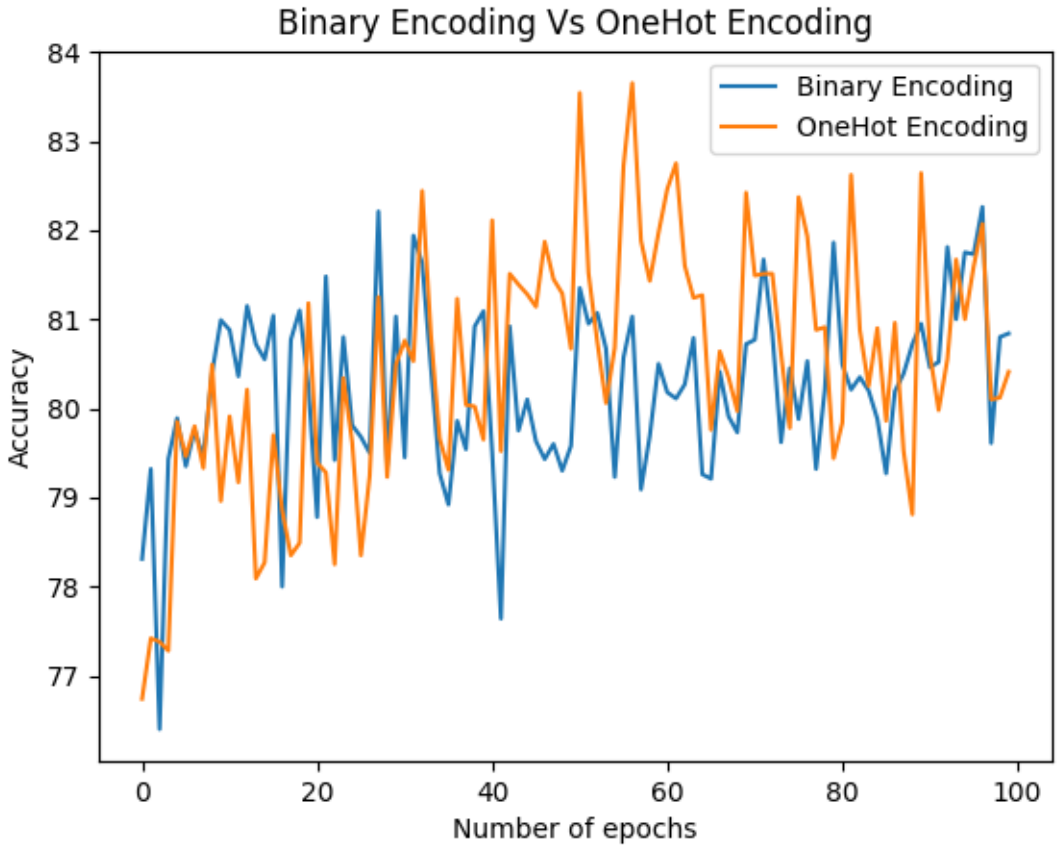


Figure 4.1: One-Hot Encoder vs Binary Encoder

4.1.3 Data Normalization

NSL-KDD dataset has some features whose values are very large such as 'duration' and 'src_bytes' and some features whose values are very small values srv_serror_rate which will effect the algorithm performance and it will make the running time slower, so data normalization is considered as a solution for those two issues. There are two main types of data normalization:

- Min-Max scaler: It's used to put all values in the dataset in range [0 , 1] using this equation :

$$x_{j,new} = \frac{x_j - x_{j,min}}{x_{j,max} - x_{j,min}} \tag{4.3}$$

  where x_j is the current j^{th} value , $x_{j,new}$ is the new j^{th} value after applying Min-Max scaler , $x_{j,min}$ is the minimum value in the column and $x_{j,max}$ is the max value in the column.

- Standard Scaler: It's used to obtain a standard distribution (Guassian) with a zero mean and a unit variance using this equation:

$$x_{j,new} = \frac{x_j - mean}{std} \tag{4.4}$$

  where x_j is the current j^{th} value , $x_{j,new}$ is the new j^{th} value after applying standard scaler , mean is the mean value of the column and std is the standard deviation of the current column.

4.2 Model Building

In this section, We will discuss the different deep learning methods used to build IDS on NSL-KDD dataset. We will build our model using multilayer perceptron (MLP), two dimensional Convolutional Neural Network (CNN2D) and multi-stage features CNN1D Each model will be described in terms of its components, architecture, and hyperparameters used to train it.

4.2.1 Multilayer Perceptron

MLP is one of ANN structures , MPL is consisting of a series of layers the first layer is the input layer , the last layer is the output and the layers in between are called hidden layers. Each layer contains a number of artificial neurons depend on the network architecture. And at each neuron we sum the inputs coming from the previous layer, then we will apply the activation Function (AF) and finally we will output the value to the next layer. MLP is one of the most widely used algorithms in classification problems and to build a mathematical model.[31]

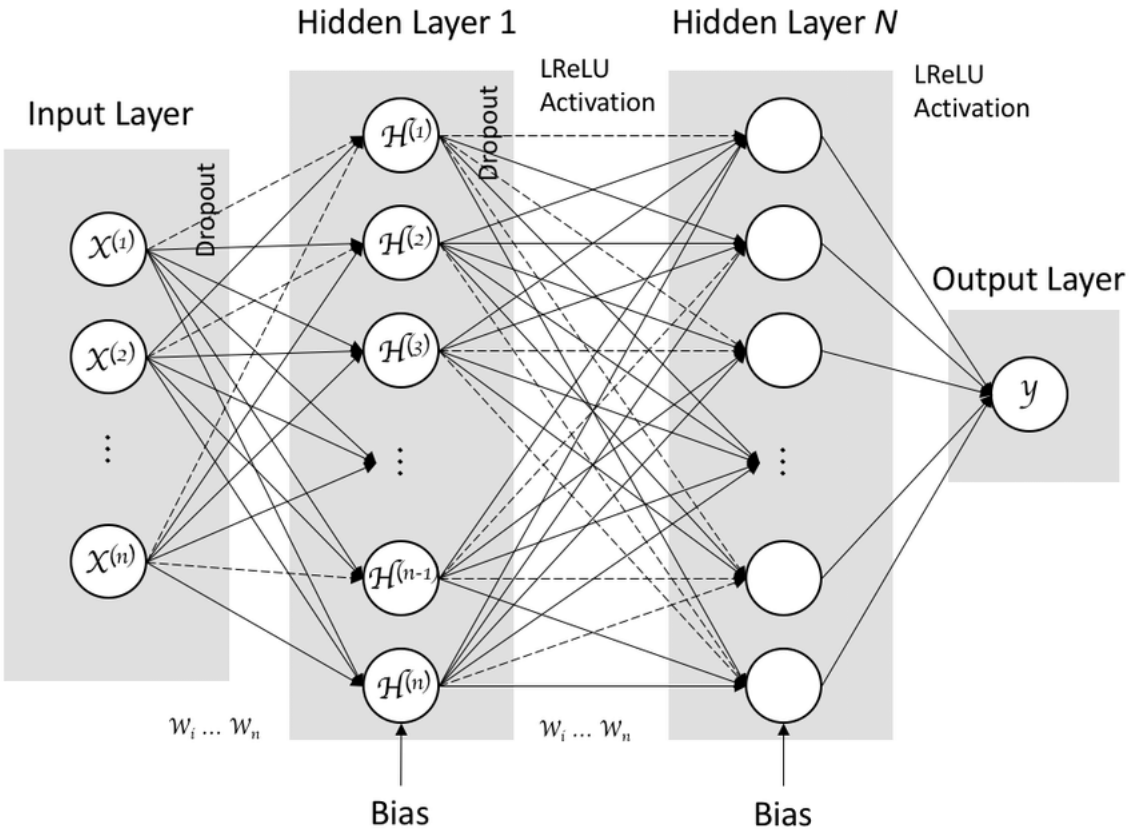


Figure 4.2: MLP structure

There are various activation functions (AF) that can be applied to each layer.

A. ReLU

ReLU stands for Rectified Linear unit and it's considered as the most popular activation function for deep neural network.[32] Its output is zero when the output is negative or a zero. And it

has a range from $[0, \infty[$. It has an outstanding performance and increasing running time speed. [33]

$$f(x) = max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{4.5}$$

where f(x) is the output and x is the input.

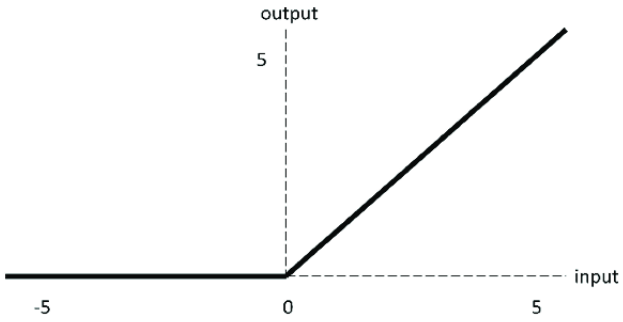


Figure 4.3: ReLU Activation function

B. Tanh

It's a hyperbolic tan function. It's another activation function, it's centred around the zero and its range is [-1 , 1].[34]

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.6}$$

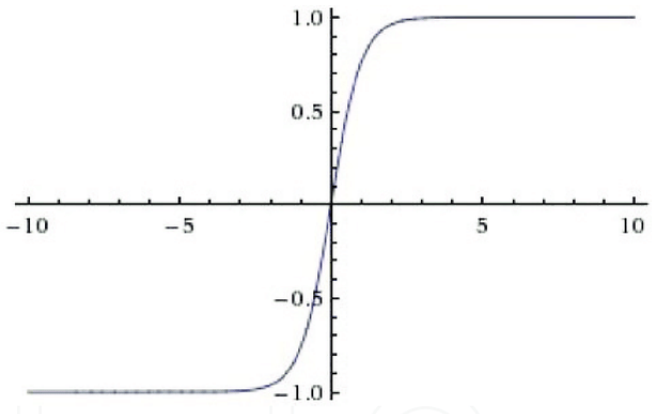


Figure 4.4: Tanh Activation function

C. Sigmoid

Sigmoid is another activation function, it's widely used in the output layer in binary classification problems. It always has a positive output value, so we can say its range is $[0, \infty[$. [35]

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.7}$$

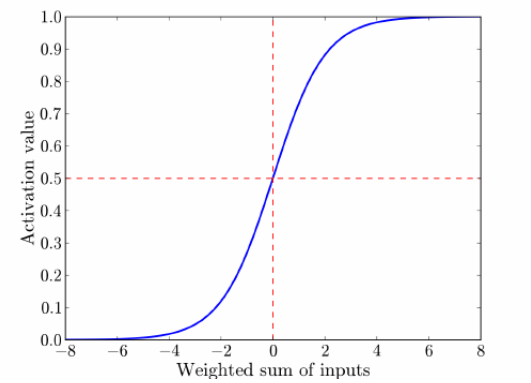


Figure 4.5: Sigmoid Activation function

D. Softmax

Softmax is another activation function, it's widely used in the output layer in Multi classification problems. For every class it will have a value from [0 , 1] and this value is representing the probability of object to be from this class, the sum of all probabilities must be one, and the class with the highest probability will be predicted as a result.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.8}$$

where $f(x_i)$ is the probability of the i^{th} class.

E. MLP Model training Our Model has a series of input layer , six hidden layers and output layer, and each layer has a number of neurons to achieve high performance and accuracy. We

use a ReLU as an activation function in the hidden layers, We use a sigmoid activation function for binary classification , softmax for multi class classification. Our objective is to minimize the loss function and maximize the accuracy, so we will use a cross-entropy cost function either binary or categorical based on the classification type.

$$J(W,b) = \frac{-1}{m} * \sum_{i=1}^{m} (y^{(i)} log a^{i} + (1 - y^{(i)} log(1 - a^{(i)})) \tag{4.9}$$

where J(W,b) is the cost function with weight W and bias b as inputs , m is the sample size , $y^{(i)}$ is the correct result and $a^{(i)}$ is the expected result.

We have built two model for MLP, each model has different hyperparamters such as different number of hidden units , learning rate and number of epochs.

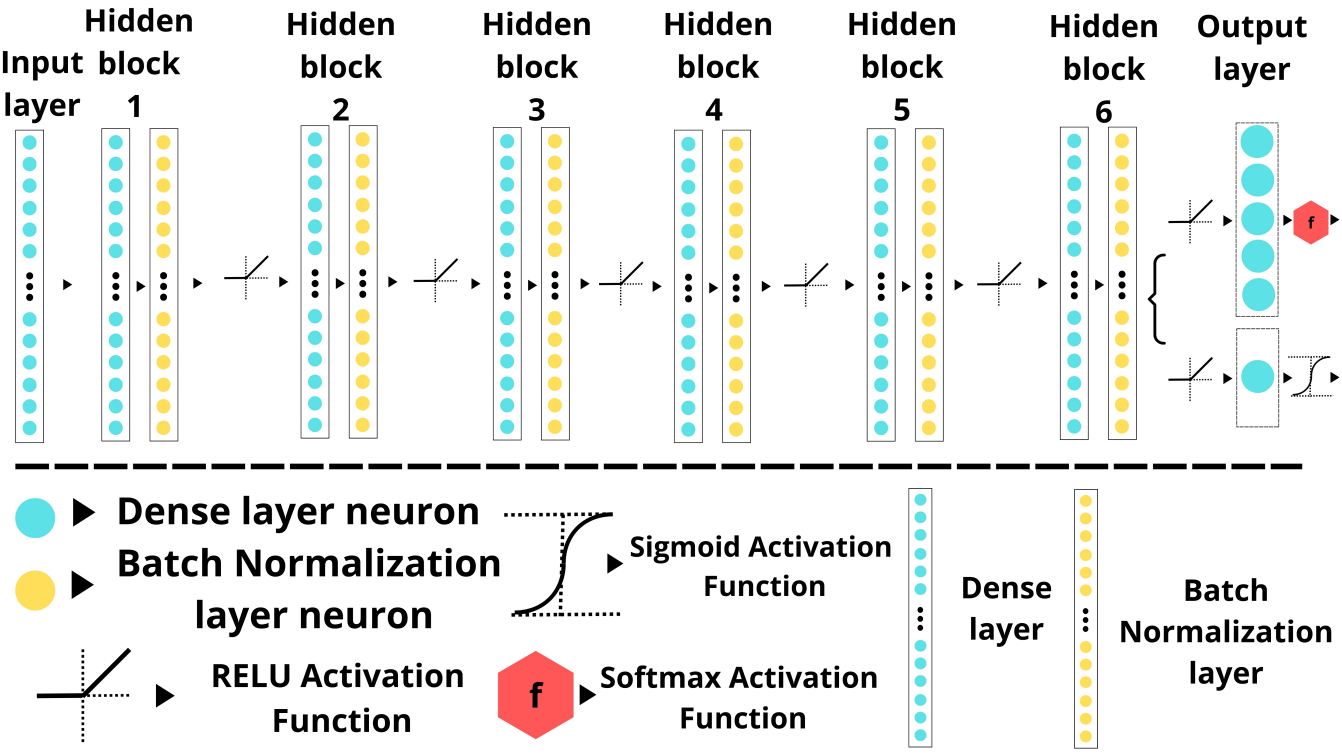


Figure 4.6: MLP model 1 Architecture

Table 4.4: MLP Binary Classification Model 1 Structure

input layer	122
Dense layer 1 size	128
Dense layer 2 size	256
Dense layer 3 size	512
Dense layer 4 size	1024
Dense layer 5 size	256
Dense layer 6 size	64
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	binary crossentropy
Optimizer	Adam
Learning rate	0.0001
Batch size	128
epochs	89
seed	100

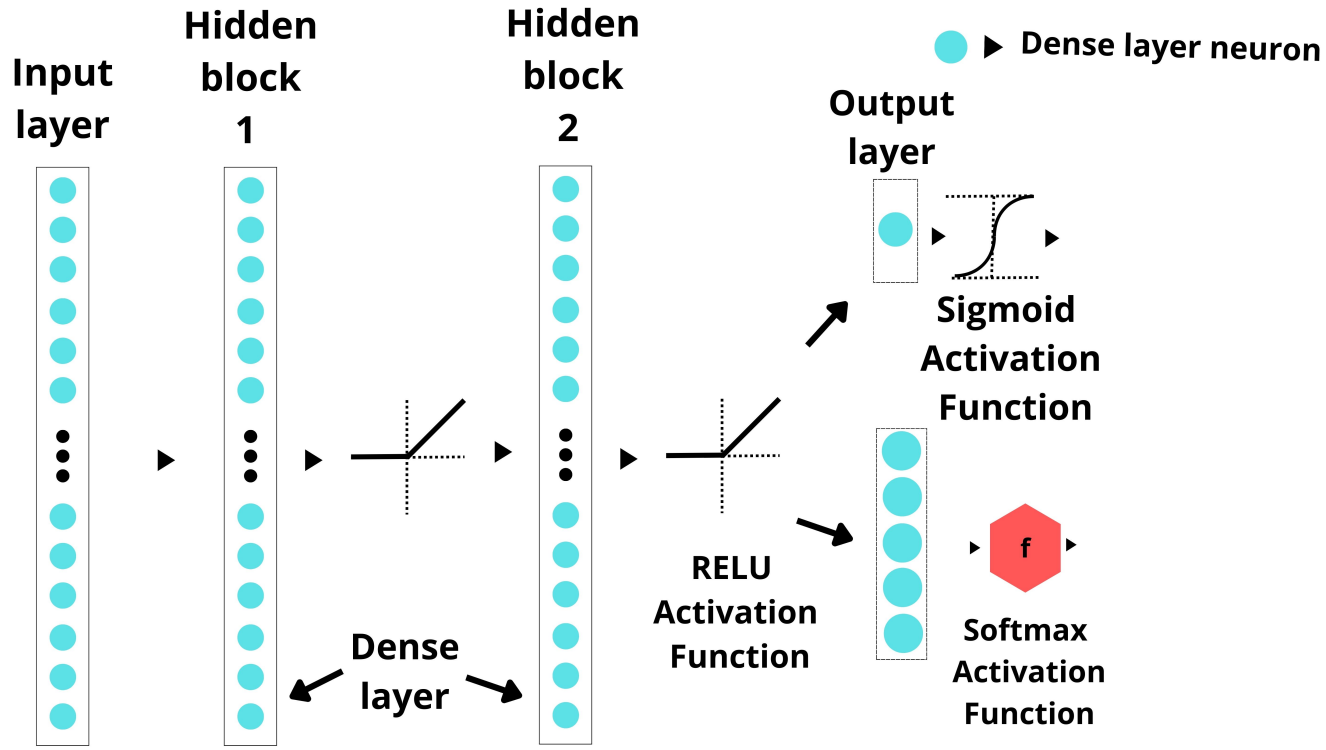


Figure 4.7: MLP model 2 Architecture

Table 4.5: MLP Binary Classification Model 2 Structure

input layer	122
Dense layer 1 size	102
Dense layer 2 size	50
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	binary crossentropy
Optimizer	Adam
Learning rate	0.1
Batch size	128
epochs	18
seed	100

Now we are going to show some algorithms used in this model and some hyperparameters

definitions.

Gradient Descent

is a first-order iterative optimization procedure for finding a local minimum/maximum of a function. This algorithm is used to minimize a cost/loss function, this approach is extensively used in machine learning (ML) and deep learning (DL).

Epochs number

An epoch means training the neural network with all the training data for one cycle. In an epoch, we use all of the data exactly once.

Learning rate α

The learning rate is a tuning parameter in an optimization algorithm that sets the step size at each iteration as it moves toward the loss function's minimum. As shown in figure 4.8, if we have the same model and we have changed only the learning rate, we will end up with different accuracies.

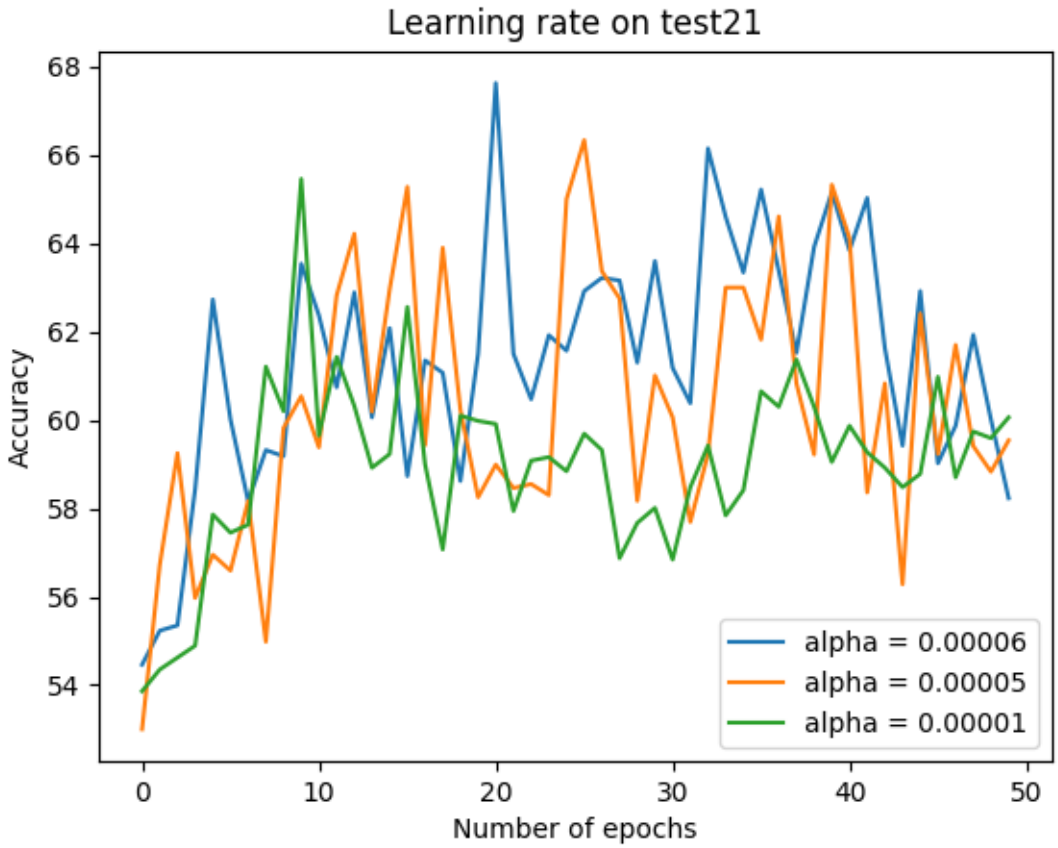


Figure 4.8: Learning rate α affects the model performance

Adam Optimizer

Adaptive Moment Estimation is a technique for optimizing gradient descent algorithms. When working with huge problems with a lot of data or parameters, the approach is quite efficient. It is efficient and takes minimal memory. It's essentially a hybrid of the 'gradient descent with momentum' and the 'RMSP' algorithms.

Batch size

After computing the gradient of error with respect to a portion of the training set, the parameters are updated.

Random Number Generator

Random number generator (RNG) is an algorithm to generate random numbers that can't be predicted. Random number generators are playing an important role in machine learning algorithms. For example weighs in deep learning is initiated by a random number. Random number can have a great effect on the algorithm performance. As shown in figure 4.9 if we have a model and we only change the seed value, it will effect the performance of our model. For example with seed 3 the first epoch might meet a local maximum point, so the derivative will be zero, so no improvement will happen in this case.

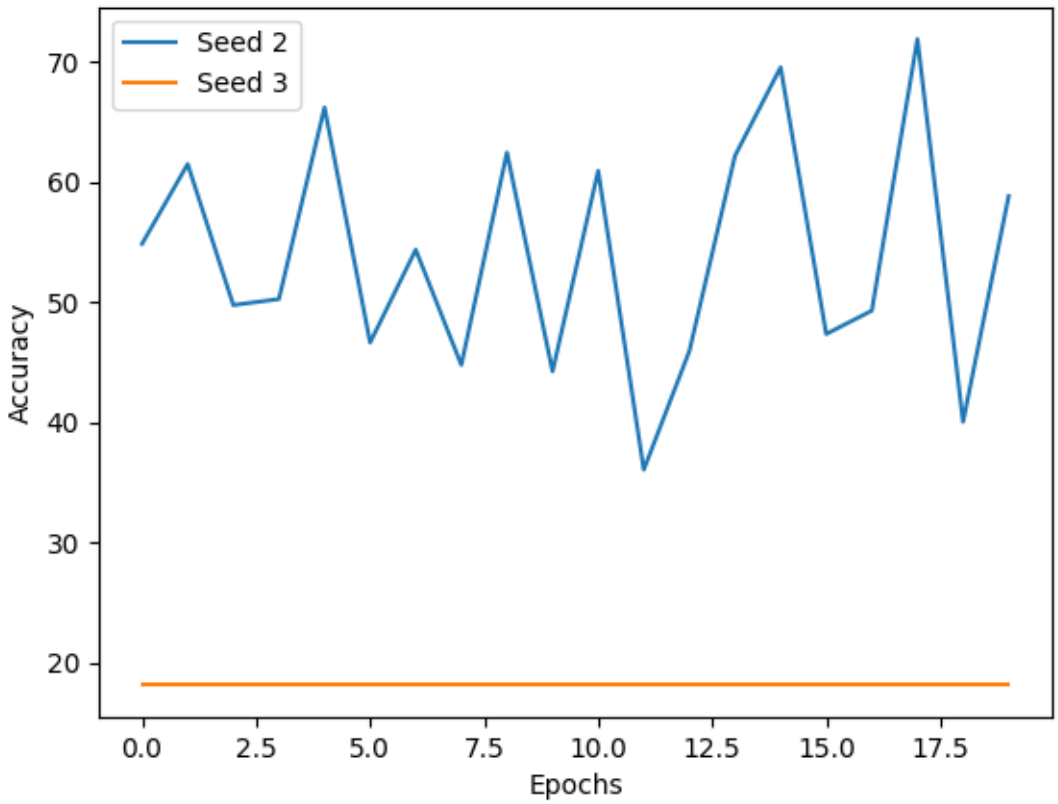


Figure 4.9: Different seeds lead to different performance

Table 4.6: MLP Multi Classification Model 1 Structure

input layer	122
Dense layer 1 size	128
Dense layer 2 size	256
Dense layer 3 size	512
Dense layer 4 size	1024
Dense layer 5 size	256
Dense layer 6 size	64
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	categorical crossentropy
Optimizer	Adam
Learning rate	0.0001
Batch size	128
epochs	13
seed	100

Table 4.7: MLP Multi Classification Model 2 Structure

input layer	122
Dense layer 1 size	102
Dense layer 2 size	50
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	categorical crossentropy
Optimizer	Adam
Learning rate	0.1
Batch size	128
epochs	34
seed	100

Class Weight

As shown in table 2.2, there is an obvious problem which is the classes distributions are unbalanced. For example normal and dos attack classes represent the majority of the dataset, on the other hand U2R attack has only 52 records which represent 0.041% of the KDDTrain+ dataset. So the algorithm is going to predict normal and dos attack most of the time because the algorithm is trained a lot on those classes. In order to solve this problem, we are going to use class weights. Where each class has its own class weight based on the number of samples in this class, so classes with small number of occurrences in the dataset will have a high weight.

$$W_i = \frac{total\ number\ of\ samples\ in\ the\ dataset}{i^{th} class\ samples\ * number\ of\ classes} \tag{4.10}$$

As shown in figure 4.10 after applying class weight to our training dataset the accuracy has improved.

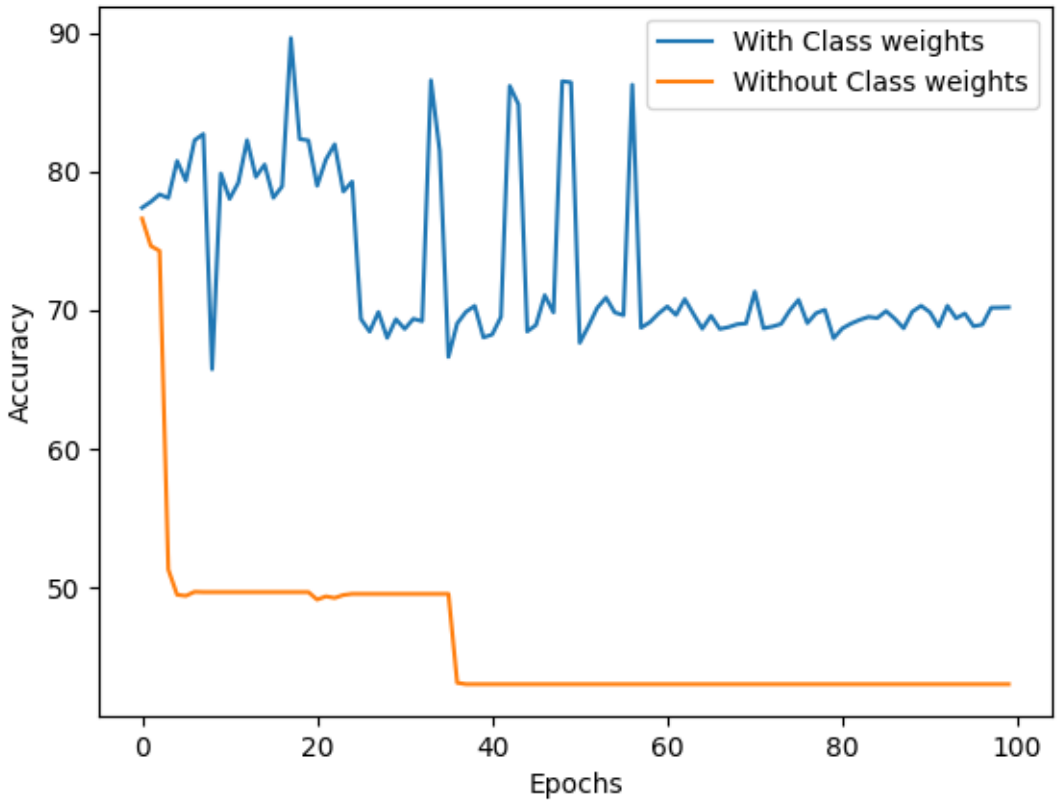


Figure 4.10: Class Weigh to improve imbalanced classes

4.2.2 CNN 2D

Two dimensional Convolutional Neural Network is one of the most important neural networks architectures which has been proved to be successfully applied in many fields such as image recognition , Localization , speech recognition and etc.[36] CNN2D is composed of different layers and stages. It takes an image as an input and usually the input is either a 3D vector (no batching) or 4D vectors if the batching is used. The 4 dimensions are (batch size , image width , image height , number of channels) the number of channel can be 3 if the photo is an RGB (colored) , or 1 of it's on a gray scale(black and white). After taking the input , it will go through different layers such as convolutional layer, pooling layer , fully connected layer and finally the output layer. [37]

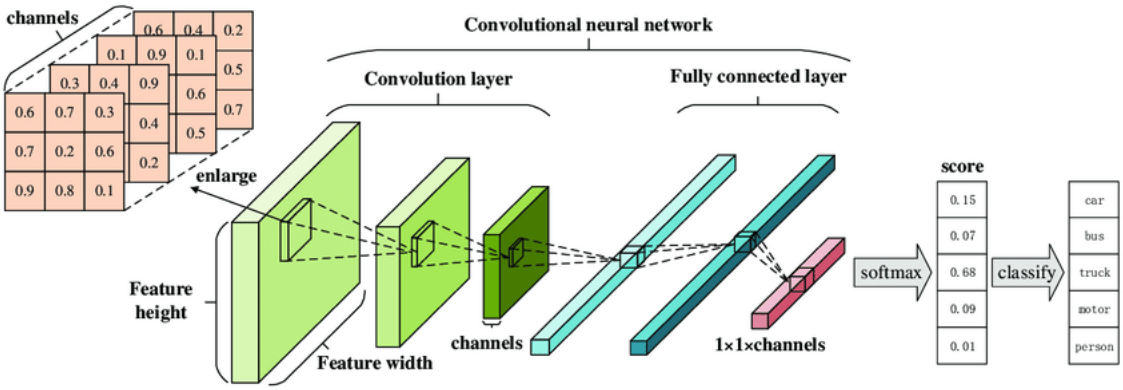


Figure 4.11: Architecture of Two dimensional Convolutional Neural Network

A. Convolutional Layer

Convolutional Layer is considered as the basic block of CNN2D. Most of calculations and extraction of important features take place in this layer. It has a kernel filters with size to perform the convolution. It has a stride (step) that will loop over the previous 2d layer, and padding (adding a frames of zeros) to keep the output size larger than the normal case.[38]

$$A_{l+1} = A_l * C \tag{4.11}$$

where A_{l+1} is the output layer resulting from the convolution operation between A_l the l^{th} layer and the C the convolution filter.

$$x_{i,k}^{l,j} = f(b_j + \sum_{a=1}^{m} w_{a,k}^{j} r_{i+(k-a)*s+a-1}^{l-1,j}) \tag{4.12}$$

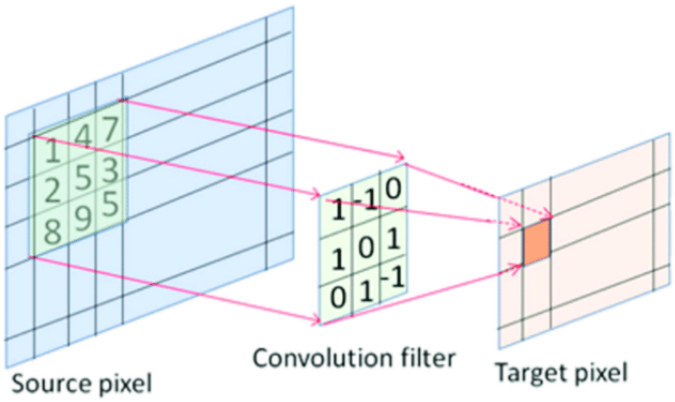


Figure 4.12: Depiction of the convolution layer with a filter in convolutional layer

To calculate the resolution of the next layer we use:

$$W_l, H_l = (\frac{W_{l-1} - f_w + 2p}{s} + 1), (\frac{H_{l-1} - f_h + 2p}{s} + 1) \tag{4.13}$$

where W_l, H_l are the width of the current layer and the height respectively , W_{l-1}, H_{l-1} are the width of the previous layer and the height respectively, f_w, f_h are the width of the filter and the height respectively, p is the padding and s is the stride.

B. Pooling layer

Pooling layer is usually applied after the convolutional layer to reduce the resolution (dimensions) of the next layer without any information loss.[39] There are two main types of pooling:

- Max Pooling : It's a pooling operating to calculate the maximum value for patches and uses it to create a pooled featured map. Nowadays, Max pooling is the most selected pooling mechanism.
- Average Pooling : It's a pooling operating to calculate the average value for each patch on the feature map.

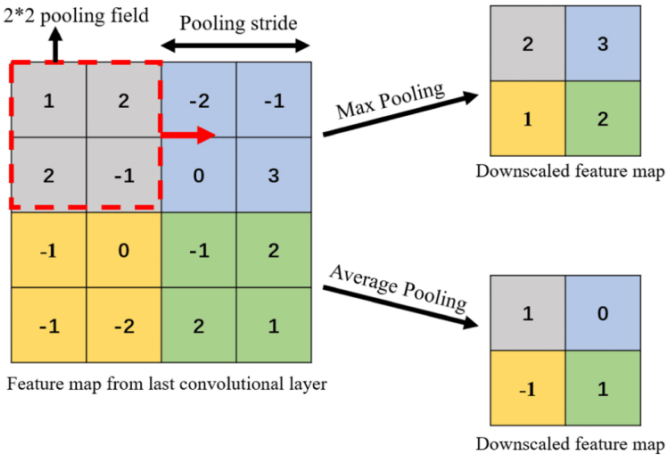


Figure 4.13: Max and average pooling layer example

C. Fully Connected layer and Flatten layer

Fully connected (FC) layer is similar to the hidden layer in the MLP model. Flatten layer is used to reshape the 2D layer into 1D layer, so we can use the Fully connected layer.

D. CNN2D Model training

This model consists of three convolutional layers, each convolutional layer is followed by a max pooling layer, but first we need to modify the input in order to be suitable for processing, so we need to delete one feature whose mean and variance is zero which is 'num_outbound_cmds'. Our input after applying One-Hot encoder will have 121 features, then we will reshape the input to (batch_size , 11 ,11 ,1). Each convolutional layer has a kernal size (2,2) and will use a ReLU as an activation function followed by a max pooling layer followed by a batch normalization layer. After the third convolutional layer we will add a flatten layer to convert it to 1D, then we will apply a three fully connected layers followed by a batch normalization layer, finally the output layer.

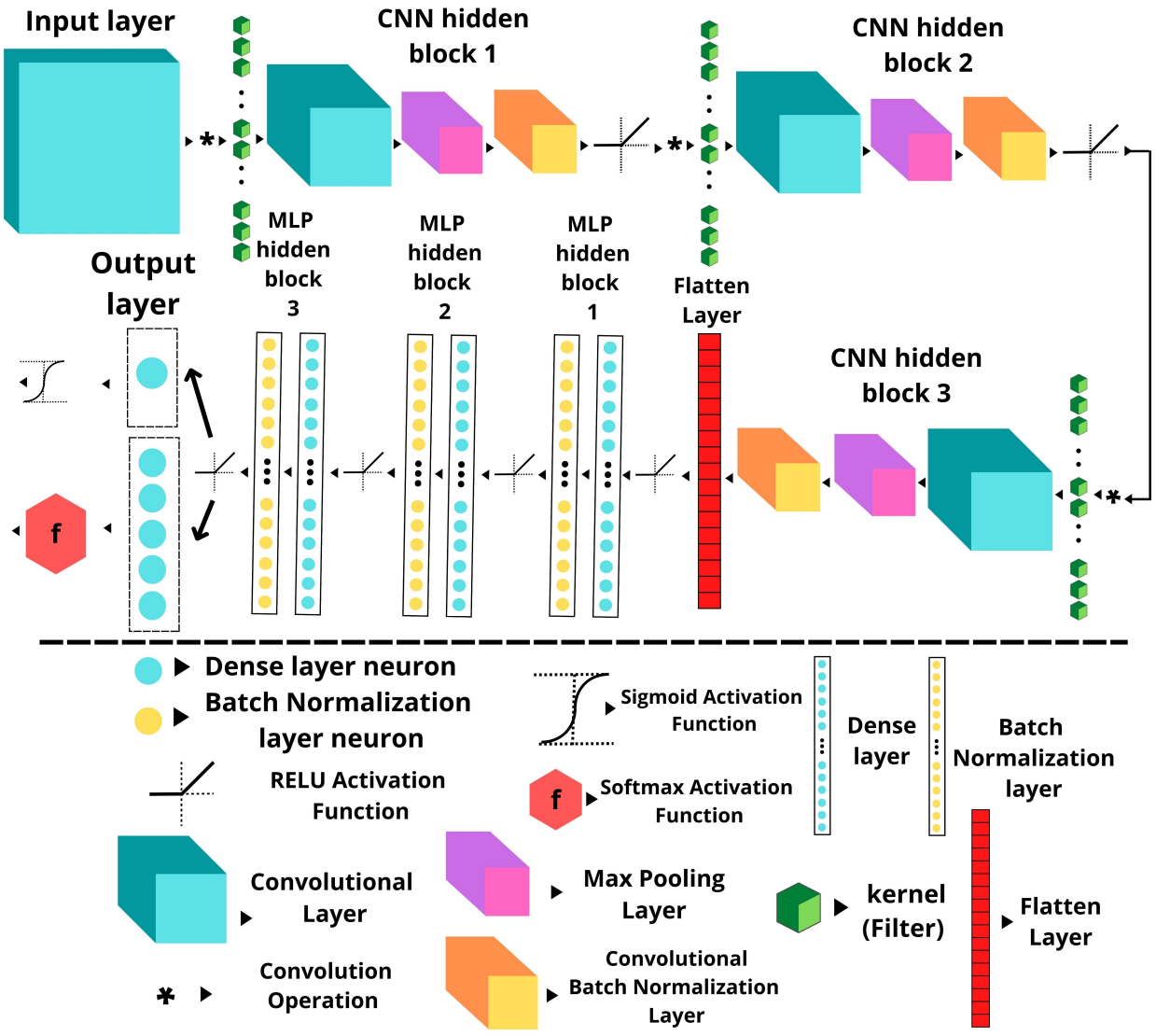


Figure 4.14: CNN2D model Architecture

Table 4.8: CNN2D Binary Classification Model Structure

input layer	(batch_size , 11 ,11 ,1)
Conv2d1+MaxPooling1	(batch_size,5,5,20)
Conv2d2+MaxPooling2	(batch_size,2,2,40)
Conv2d3+MaxPooling3	(batch_size,1,1,60)
Flatten	(batch_size,60)
FC layer 1	(batch_size,64)
FC layer 2	(batch_size,80)
FC layer 3	(batch_size,25)
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	binary crossentropy
Optimizer	Adam
Learning rate	0.009000000000000001
Batch size	128
epochs	6
seed	63

Table 4.9: CNN2D Multi Classification Model Structure

input layer	(batch_size , 11 ,11 ,1)
Conv2d1+MaxPooling1	(batch_size,5,5,20)
Conv2d2+MaxPooling2	(batch_size,2,2,40)
Conv2d3+MaxPooling3	(batch_size,1,1,60)
Flatten	(batch_size,60)
FC layer 1	(batch_size,64)
FC layer 2	(batch_size,80)
FC layer 3	(batch_size,25)
Output layer	5 , softmax
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	categorical crossentropy
Optimizer	Adam
Learning rate	0.0001
Batch size	128
epochs	13
seed	100

4.2.3 Multi-Stage features CNN1D

CNN1D is another type of CNN which is recently developed. These research have proven that 1D CNNs are beneficial and hence superior to their 2D counterparts in dealing with 1D signals for specific applications because of the following reasons:[40]

- The computational complexity of 1D CNNs is much lower than that of 2D CNNs because it uses simple arrays instead of matrix.
- According to recent research, 1D CNNs with relatively shallow architectures may learn difficult tasks requiring 1D signals. On the other hand, To perform such tasks, 2D CNNs often require more complex structures. Networks with shallow architectures are obviously more easier to train and deploy.

- Compact 1D CNNs are well-suited for real-time and low-cost applications, particularly on mobile or handheld devices, because to their minimal processing needs.

This network, like CNN2D, has convolutional layers, pooling layers, dropout layers, and fully connected layers, but it is one-dimensional.

A. One-dimensional convolutional layer

Convolution occurs in this layer utilising kernels that generate feature maps, similar to the idea of a two-dimensional convolutional layer.

B. CNN1D Model training

Our model is consisting of three convolutional layers, each layer is followed by a max pooling layer and a dropout layer. Each convolutional layer uses the ReLU as an activation function. Finally, the output layer is used to classify the output, a sigmoid is used for binary classifaction and a softmax for multi classification.

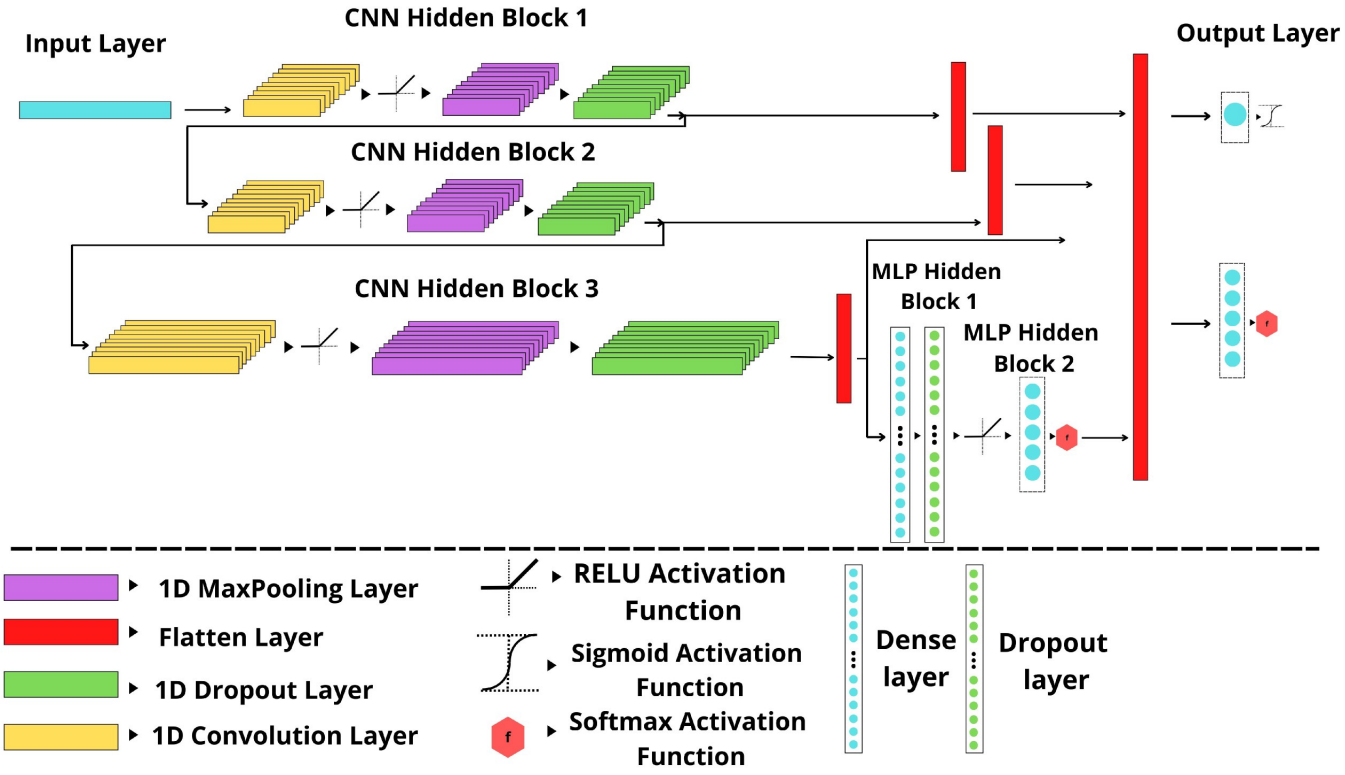


Figure 4.15: CNN1D model Architecture

Table 4.10: CNN1D Binary Classification Model Structure

input layer	(batch_size , 1 ,122)
Conv2d1+MaxPooling1+Dropout1	(batch_size,1,62)
Conv2d2+MaxPooling2+Dropout2	(batch_size,1,62)
Conv2d3+MaxPooling3+Dropout3	(batch_size,1,124)
FC layer 1	(batch_size,256)
FC layer 2	(batch_size,5)
Output layer	1 , sigmoid
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	binary crossentropy
Optimizer	Adam
Learning rate	0.5
Batch size	128
epochs	17
seed	13

Table 4.11: CNN1D Multi Classification Model Structure

input layer	(batch_size , 1 ,122)
Conv2d1+MaxPooling1+Dropout1	(batch_size,1,62)
Conv2d2+MaxPooling2+Dropout2	(batch_size,1,62)
Conv2d3+MaxPooling3+Dropout3	(batch_size,1,124)
FC layer 1	(batch_size,256)
FC layer 2	(batch_size,5)
Output layer	5 , softmax
Normalization	Min-Max scaler
Activation function	ReLU
Loss function	categorical crossentropy
Optimizer	Adam
Learning rate	0.05
Batch size	128
epochs	75
seed	13

A. Dropout Layer

Deep learning approaches include several hidden layers and numerous neurons in each layer, allowing them to learn complex correlations. In certain circumstances, intricate associations are the consequence of sampling noise, which causes overfitting. This indicates that the model will not generalise the problem and will perform poorly on test datasets. Dropout is a way for dealing with this issue. It operates by momentarily disconnecting a unit's incoming and outgoing connections. Choosing which unit to delete is a random procedure in which each unit is kept with a given probability p that is independent of other units.

B. Validation Dataset

The validation dataset is derived from the KDDTrain+ dataset in order to give an impartial evaluation of the model fit on the training dataset while changing model Hyperparameters. The

validation dataset allows you to check stability and avoid overfitting.

4.2.4 Experimental Settings

The software used in this experiment consists of keras on the backend of Tensorflow, which is one of the newest and simplest frameworks, was employed in this experiment. The experiment is performed on Google co-lab. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. Our experiment study the performance of our model on NSL-KDD dataset to detect binary (normal , attack) and five classification (normal , dos , U2R , probe , R2L) using the 41 features that can be divided into four group:

1. Basic features No.(1-10).
2. Content features No.(11-22).
3. Time-based network traffic statistics features No.(23-31).
4. Host-based network traffic statistics features No.(32-41)

4.2.5 Evaluation Metrics

We use three measurement for evaluating the model performance. We have Accuracy (AC) which can be applied to the binary model and the multi class classification, Detection Rate (DR) and False Positive Rate (FPR) applied for the binary model only. There are four values that we need to know before calculating the three metrics.

1. **True Positive (TP)** : is a result in which the model predicts the positive class properly.
2. **True Negative (TN)** : is a result in which the model predicts the negative class properly.
3. **False Positive (FP)** : is an outcome in which the model guesses the positive class inaccurately.

4. **False Negative (FN)** : is an outcome where the model incorrectly predicts the negative class.

Now we can calculate the three metrics:

- **Accuracy (AC)**: The percentage of properly categorised predictions by the model.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.14}$$

- **Detection Rate (DR)** : The rate of the number of accurately classified records over the overall number of peculiarity records.

$$DR = \frac{TP}{TP + FN} \tag{4.15}$$

- **False Positive Rate (FPR)** : The number of erroneously classified records divided by the whole number of normal records.

$$FPR = \frac{FP}{FP + TN} \tag{4.16}$$

Our objective is to maximize the accuracy and the detection rate and minimize the false positive rate.

Chapter 5

Testing and Results

5.1 Performance Analysis

The four models have been trained on the KDDTrain+ and tested using KDDTest+ and KDDTest-21 for multi classification and binary classification. The models achieve a high performance in binary and multi classification

Table 5.1: Comparison between our models in Binary classification accuracies

Model	KDDTrain+	KDDTest+	KDDTest-21
MLP Model 1	**99.60547**	84.78906	71.14768
MLP model 2	94.18	89.59	81.95
CNN2D	96.31349	90.57357	83.32489
CNN1D	92.15	**91.4**	**87.8**

Table 5.2: Comparison between our models in Multi classification accuracies

Model	KDDTrain+	KDDTest+	KDDTest-21
MLP Model 1	**99.87**	84.53	71.59
MLP model 2	93.49	**86.56**	**80.09**
CNN2D	96.12	81.81	66.70
CNN1D	92.15	84.7	71.24

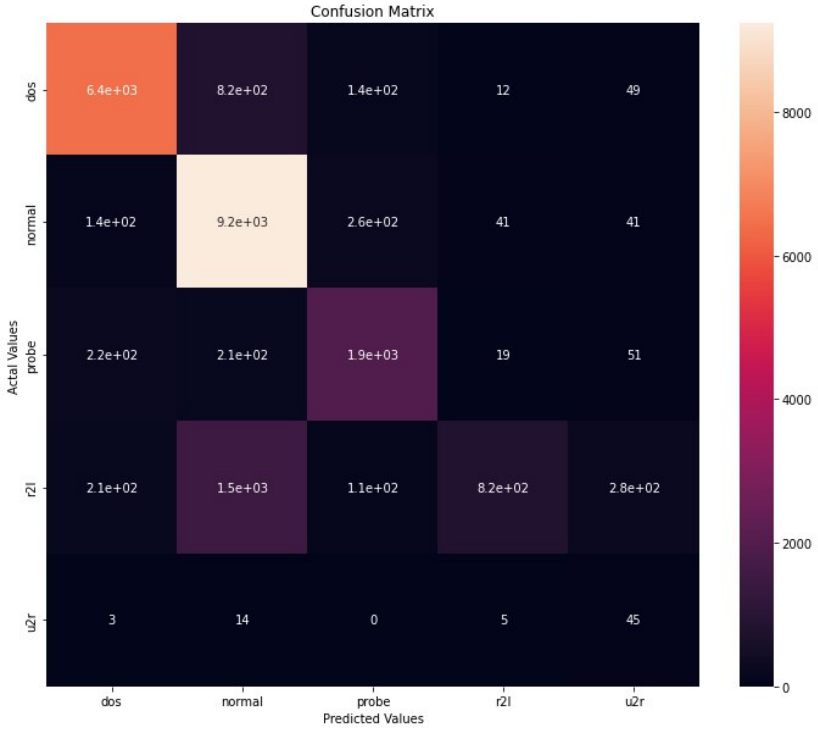


Figure 5.1: CNN2D model Confusion Matrix for Multi Classification

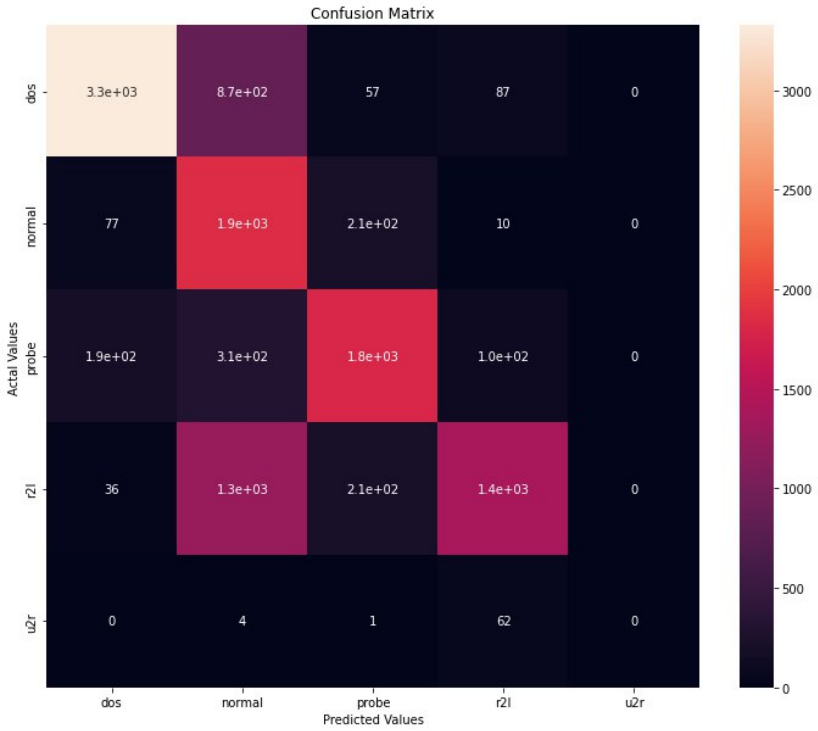


Figure 5.2: MLP1 model Confusion Matrix for Multi Classification

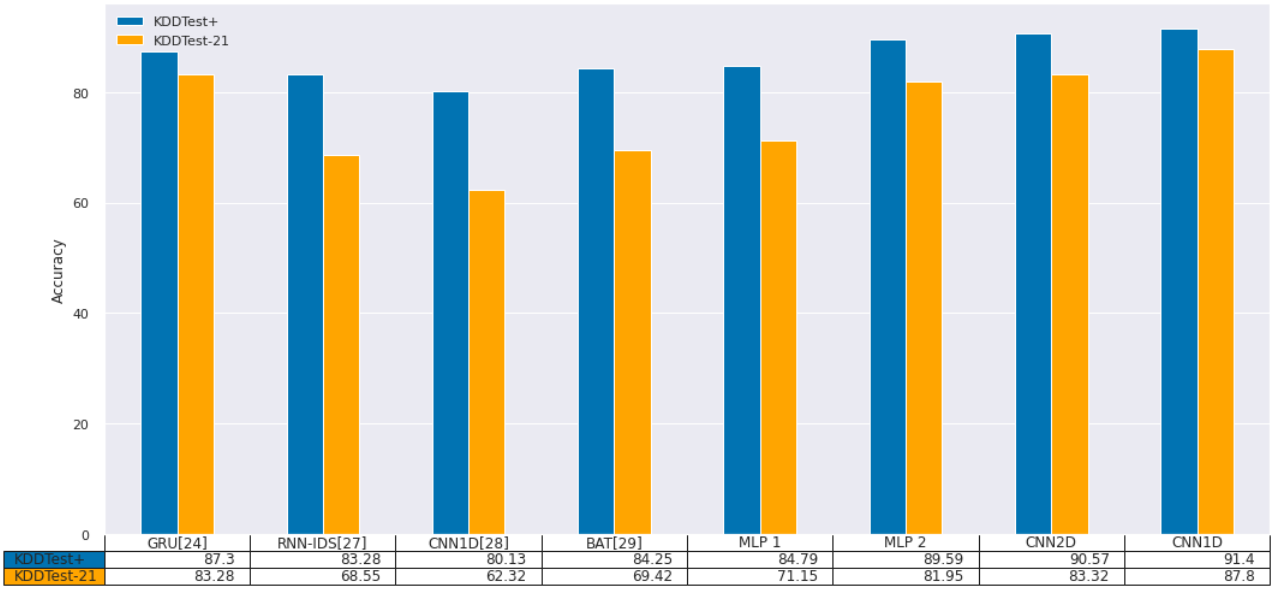
As shown in fig 5.1, the number of correct classification for U2R model is 45, while in 5.2 the number of correct classification for U2R model is 0. The reason is the MLP1 model is trained without performing the class weight to solve the problem of imbalanced classes and because the number of samples of the U2R attack in the train dataset wasn't enough to train the model in how to classify this attack we ended with zero correct classification. On the other hand, the CNN2D model is trained with performing the class weight, so the model can predict a good number of U2R attacks correctly.

5.2 Comparison to the state of art

In this section, we will compare our models to other deap and machine learning models in both binary and multi classification on the benchmark NSL-KDD dataset.

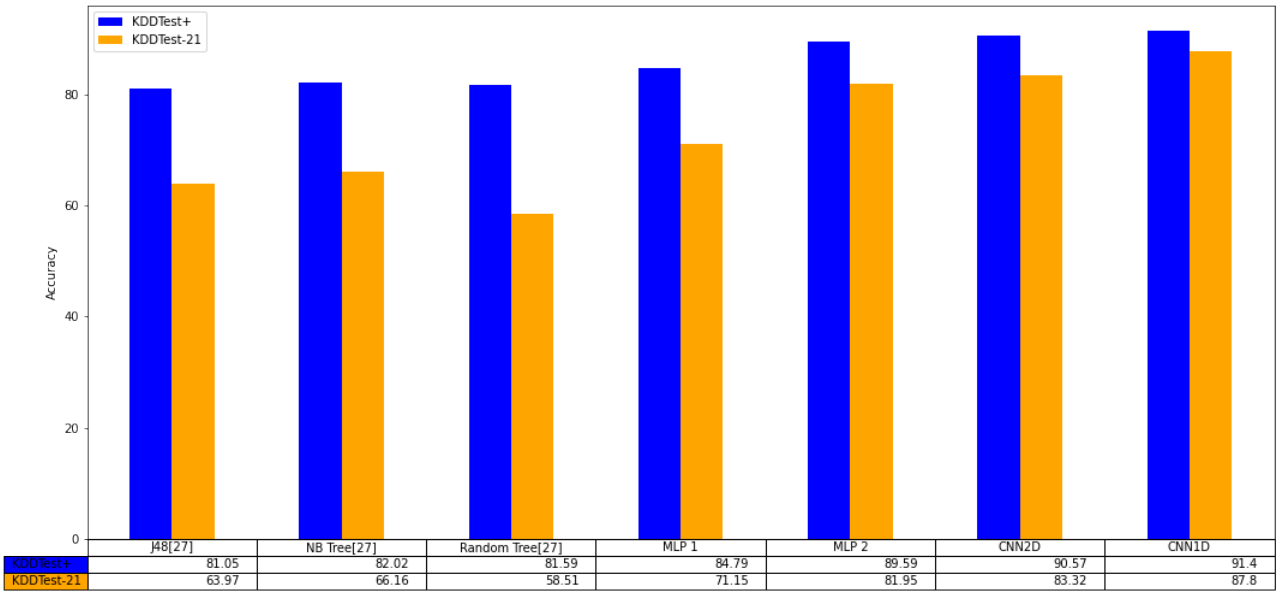
5.2.1 Binary classification

In [27] the results of binary classification are provided for machine learning techniques such as J48, Naïve Bayesian, NB Tree, Random Forest, Random Tree and SVM in the field of intrusion detection. Also the results of the deep learning methods used in intrusion detection system, such as GRU used in [24], RNN-IDS used in [27], CNN1D used in [28] or the BAT used in [29]. Our proposed CNN1D achieving a very high accuracy on KDDTest+ 91.4% which is higher than any method used by difference 4.1 %, and 87.8 % in KDDTest-21 which is also higher than any method by 2.86 %.



	GRU[24]	RNN-IDS[27]	CNN1D[28]	BAT[29]	MLP 1	MLP 2	CNN2D	CNN1D
KDDTest+	87.3	83.28	80.13	84.25	84.79	89.59	90.57	91.4
KDDTest-21	83.28	68.55	62.32	69.42	71.15	81.95	83.32	87.8

Figure 5.3: Comparison between Deep Learning methods and our proposed methods in Binary Classification

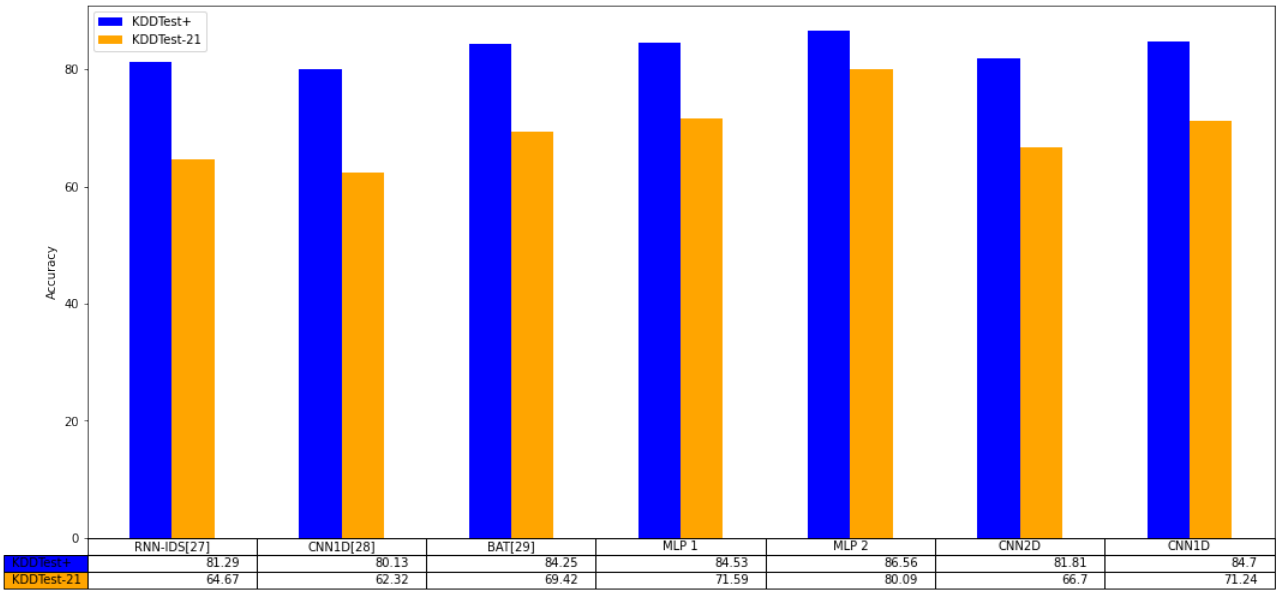


	J48[27]	NB Tree[27]	Random Tree[27]	MLP 1	MLP 2	CNN2D	CNN1D
KDDTest+	81.05	82.02	81.59	84.79	89.59	90.57	91.4
KDDTest-21	63.97	66.16	58.51	71.15	81.95	83.32	87.8

Figure 5.4: Comparison between Machine Learning methods and our proposed methods in Binary Classification

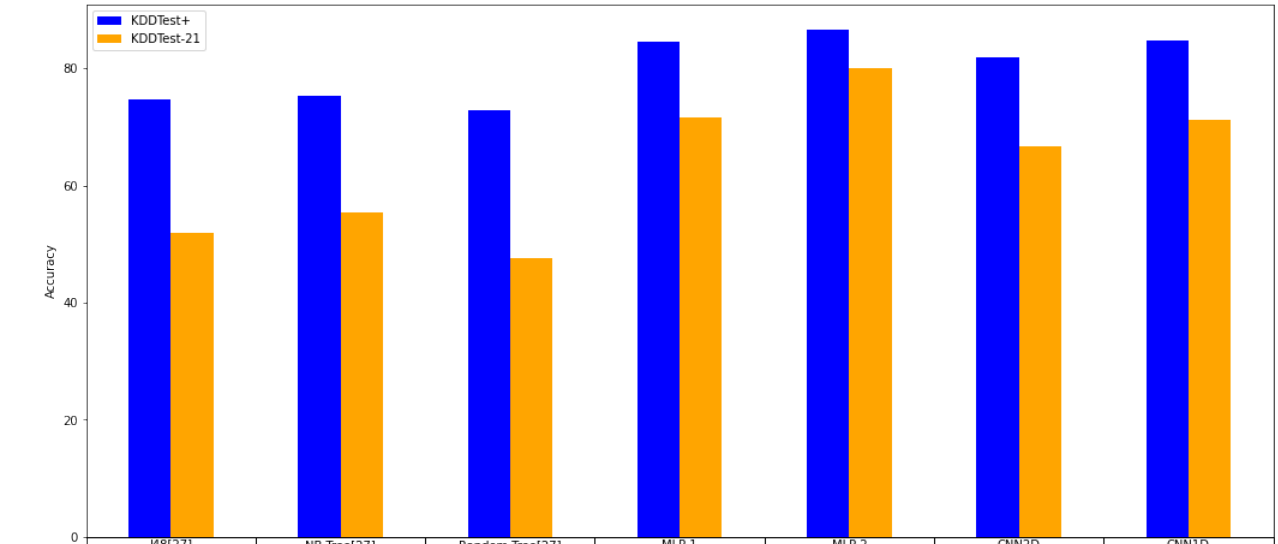
5.2.2 Multi classification

In [27] the results of multi classification are provided for machine learning techniques such as J48, Naïve Bayesian, NB Tree, Random Forest, Random Tree and SVM in the field of intrusion detection. Also the results of the deep learning methods used in intrusion detection system, such as GRU used in [24], RNN-IDS used in [27], CNN1D used in [28] or the BAT used in [29]. Our proposed MLP2 model achieving a very high accuracy on KDDTest+ 86.56% which is higher than any method used by difference 2.31 %, and 80.09 % in KDDTest-21 which is also higher than any method by 10.67 %.



	RNN-IDS[27]	CNN1D[28]	BAT[29]	MLP 1	MLP 2	CNN2D	CNN1D
KDDTest+	81.29	80.13	84.25	84.53	86.56	81.81	84.7
KDDTest-21	64.67	62.32	69.42	71.59	80.09	66.7	71.24

Figure 5.5: Comparison between Deep Learning methods and our proposed methods in Multi Classification



	J48[27]	NB Tree[27]	Random Tree[27]	MLP 1	MLP 2	CNN2D	CNN1D
KDDTest+	74.6	75.4	72.8	84.53	86.56	81.81	84.7
KDDTest-21	51.9	55.4	47.7	71.59	80.09	66.7	71.24

Figure 5.6: Comparison between Machine Learning methods and our proposed methods in Multi Classification

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we aim to build an efficient anomaly intrusion detection system (AIDS) which is very important to many organizations and governments sectors because malware development is increasing on a daily basis. We used deep learning algorithms in order to build an intrusion detection system with higher classification accuracies either in the binary classification or in the multi classification. In this thesis we have proposed different deep learning models namely MLP, CNN2D and CNN1D. CNN1D has the highest performance in binary classification achieving 91.4 % on KDDTest+ and 87.8 % on KDDTest-21 while MLP2 has the highest performance in multi classification achieving 86.56 % on KDDTest+ and 80.09 % on KDDTest-21. CNN1D overcomes deep learning and machine learning models in binary classification where it has a higher accuracy than other model by difference 4.1 % on KDDTest+ and 2.86 % on KDDTest-21. Also MLP2 overcomes deep learning and machine learning models in multi classification where it has a higher accuracy than other model by difference 2.31 % on KDDTest+ and 10.67 % on KDDTest-21. To sum up we believe our models are very strong and powerful to be used in an intrusion detection system.

6.2 Future Work and Limitations

In the future, we hope to analyse our model using other benchmark datasets, such as CICIDS2017 and UNSW NB15. We want to apply our models in a real life system. Furthermore, we want to see how applying features selection methods like PCA affects the NSL-KDD dataset.

Bibliography

[1] David S. Linthicum. Connecting fog and cloud computing. *IEEE Cloud Computing*, 4(2):18–20, 2017.

[2] Confidence M. Hlatshwayo and Tranos Zuva. Mobile public cloud computing, merits and open issues. In *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 128–132, 2016.

[3] Zong-pu Jia and Xiao Tian. A novel security private cloud solution based on ecryptfs. In *2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering*, volume 3, pages 38–41, 2013.

[4] David S. Linthicum. Emerging hybrid cloud patterns. *IEEE Cloud Computing*, 3(1):88–91, 2016.

[5] Amin M. Khan, Felix Freitag, and Luís Rodrigues. Current trends and future directions in community edge clouds. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 239–241, 2015.

[6] Ajeet Singh Poonia and Shivraj Singh. Malware detection by token counting. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pages 1285–1288, 2014.

[7] Mouhammad Alkasassbeh and Mohammad Almseidin. Machine learning methods for network intrusion detection, 2018.

[8] Xin Zhan, Huabing Yuan, and Xiaodong Wang. Research on block chain network intrusion detection system. In *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pages 191–196, 2019.

[9] Daniel Fraunholz, Marc Zimmermann, and Hans D. Schotten. An adaptive honeypot configuration, deployment and maintenance strategy. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 53–57, 2017.

[10] Sachin S. Gavankar and Sudhirkumar D. Sawarkar. Eager decision tree. In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pages 837–840, 2017.

[11] Eka Rahayu Setyaningsih and Indah Listiowarni. Categorization of exam questions based on bloom taxonomy using naïve bayes and laplace smoothing. In *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, pages 330–333, 2021.

[12] Ning Li, Li Zhao, Ai-Xia Chen, Qing-Wu Meng, and Guo-Fang Zhang. A new heuristic of the decision tree induction. In *2009 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1659–1664, 2009.

[13] Saurabh Shah and Manmohan Singh. Comparison of a time efficient modified k-mean algorithm with k-mean and k-medoid algorithm. In *2012 International Conference on Communication Systems and Network Technologies*, pages 435–437, 2012.

[14] Xingfu Zhang and Xiangmin Ren. Two dimensional principal component analysis based independent component analysis for face recognition. In *2011 International Conference on Multimedia Technology*, pages 934–936, 2011.

[15] Ömer Faruk Ertuğrul, Ramazan Tekin, and Yilmaz Kaya. Randomized feed-forward artificial neural networks in estimating short-term power load of a small house: A case study. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–5, 2017.

[16] Suchet Sapre, Pouyan Ahmadi, and Khondkar Islam. A robust comparison of the kdd-cup99 and nsl-kdd iot network intrusion detection datasets through various machine learning algorithms, 2019.

[17] Ayesha S. Dina, A. B. Siddique, and D. Manivannan. Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks, 2022.

[18] Hooman Alavizadeh, Julian Jang-Jaccard, and Hootan Alavizadeh. Deep q-learning based reinforcement learning approach for network intrusion detection, 2021.

[19] Poornima Mahadevappa, Syeda Mariam Muzammal, and Raja Kumar Murugesan. A comparative analysis of machine learning algorithms for intrusion detection in edge-enabled iot networks, 2021.

[20] Zachary Tauscher, Yushan Jiang, Kai Zhang, Jian Wang, and Houbing Song. Learning to detect: A data-driven approach for network intrusion detection, 2021.

[21] Islam Debicha, Thibault Debatty, Wim Mees, and Jean-Michel Dricot. Efficient intrusion detection using evidence theory. 2021.

[22] Cosimo Ieracitano, Ahsan Adeel, Mandar Gogate, Kia Dashtipour, Francesco Carlo Morabito, Hadi Larijani, Ali Raza, and Amir Hussain. Statistical analysis driven optimized deep learning system for intrusion detection, 2018.

[23] Elie Alhajjar, Paul Maxwell, and Nathaniel D. Bastian. Adversarial machine learning in network intrusion detection systems, 2020.

[24] Amir Andalib and Vahid Tabataba Vakili. An autonomous intrusion detection system using an ensemble of advanced learners. 2020.

[25] Peilun Wu, Hui Guo, and Richard Buckland. A transfer learning approach for network intrusion detection. 2019.

[26] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 258–263, 2016.

[27] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017.

[28] amp; Zhai Y Ding, Y. Intrusion detection system for nsl-kdd dataset using convolutional neural networks. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence - CSAI '18*.

[29] Tongtong Su, Huazhi Sun, Jinqi Zhu, Sheng Wang, and Yabo Li. Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset. *IEEE Access*, 8:29575–29585, 2020.

[30] L Dhanabal and SP 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*, 4(6):446–452, 2015.

[31] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access*, 6:3491–3508, 2018.

[32] Hossam H. Sultan, Nancy M. Salem, and Walid Al-Atabany. Multi-classification of brain tumor images using deep neural network. *IEEE Access*, 7:69215–69225, 2019.

[33] Zheng Hu, Yongping Li, and Zhiyong Yang. Improving convolutional neural network using pseudo derivative relu. In *2018 5th International Conference on Systems and Informatics (ICSAI)*, pages 283–287, 2018.

[34] Amer Zayegh and Nizar Albassam. Neural network principles and applications. *Digital Systems*.

[35] V. Morello, E. D. Barr, M. Bailes, C. M. Flynn, E. F. Keane, and W. van Straten. Spinn: a straightforward machine learning solution to the pulsar candidate selection problem. 2014.

[36] Xu Kang, Bin Song, and Fengyao Sun. A deep similarity metric method based on incomplete data for traffic anomaly detection in iot. *Applied Sciences*, 9(1), 2019.

[37] Mi-Young Lee, Joo-Hyun Lee, Jin-Kyu Kim, Byung-Jo Kim, and Ju-Yeob Kim. The sparsity and activation analysis of compressed cnn networks in a hw cnn accelerator model. In *2019 International SoC Design Conference (ISOCC)*, pages 255–256, 2019.

[38] Yoshinaga I. Sekijima K. Azechi I. amp; Baba D. Kimura, N. Convolutional neural network coupled with a transfer-learning approach for time-series flood predictions. *Water*, 12(1):96.

[39] Yidong G. Ze C. Zhi W. Erik S. Minfei, L. Microstructure-informed deep convolutional neural network for predicting short-term creep modulus of cement paste. *Cement and Concrete Research*, 152:106681.

[40] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey, 2019.