MITIGATING UNBALANCED AND OVERLAPPED PROBLEMS OF LARGE NETWORK INTRUSION DATA USING MULTIPLE-LEVEL DETECTION TECHNIQUES

HO YAN BING

MASTER OF SCIENCE

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN
JULY 2022

MITIGATING UNBALANCED AND OVERLAPPING PROBLEMS OF LARGE NETWORK INTRUSION DATA USING MULTIPLE-LEVEL DETECTION TECHNIQUES

By

HO YAN BING

A dissertation submitted to the Department of Internet Engineering and Computer Science,
Lee Kong Chian Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Science
July 2022

ABSTRACT

MITIGATING UNBALANCED AND OVERLAPPING PROBLEMS OF LARGE NETWORK INTRUSION DATA USING MULTIPLE-LEVEL DETECTION TECHNIQUES

Ho Yan Bing

Network intrusion data sets are usually unbalanced in class distribution because intrusions are rare occurrences in computer networks. Besides, data set classes may overlap because of their high similarity. These problems have caused a low detection rate for intrusions that are the minority in data sets because learning algorithms favour the majority class (normal traffic). Our study aims to design a multiple-level detection for detecting network intrusions by mitigating the unbalanced class distribution and overlapping class problems. We propose two two-level classifications (TLC) with different arrangements to improve the detection rate of intrusions. TLC type I detects only binary classes: one general intrusion and normal traffic at the first level. Then, detailly classifies the intrusion classes at the second level. On the other hand, TLC type II detects the intrusion classes and normal traffic at the first level and then passes the traffic that is classified as normal to the second level for further detection. To evaluate our proposed TLCs, we used two unbalanced and overlapped network intrusion data sets in this study: UNSW-NB15 and CICIDS2017. Our proposed TLC Type II achieved an overall accuracy of 0.9817 and 0.999 for UNSW-NB15 and CICIDS2017, respectively. The unbalanced and overlapped problems were mitigated using the proposed TLC Type II. The classifiers in TLC- Type II are arranged so that the occurrences of misclassified intrusions are minimised.

ACKNOWLEDGEMENT

Thanks to my supervisor and co-supervisor, Dr. Khor Kok Chin and Dr. Yap Wun She, for your patience, guidance, and support. I have benefited greatly from your wealth of knowledge and meticulous editing. I am extremely grateful that you took me on as a student and continued to have faith in me over the years.

I am honoured to be one of the recipients of the UTAR Research Fund (UTARRF), which supports my research throughout my postgraduate study.

I gratefully recognise the help of the MIMOS lab, University Tunku Abdul Rahman Sungai Long Campus, which provided the workstation and the Internet access environment for me to conduct the research.

Most importantly, I am grateful for my family's unconditional, unequivocal, and loving support.

APPROVAL SHEET

This dissertation/thesis entitled "**MITIGATING UNBALANCED AND OVERLAPPING PROBLEMS OF LARGE NETWORK INTRUSION DATA USING MULTIPLE-LEVEL DETECTION TECHNIQUES**" was prepared by HO YAN BING and submitted as partial fulfillment of the requirements for the degree of Master of Science at Universiti Tunku Abdul Rahman.

Approved by:

kckhor

(Dr. Khor Kok Chin)
Date: 22/7/2022
Supervisor
Department of Internet Engineering and Computer Science
Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

[illegible]

(Dr. Yap Wun She)
Date:……………………..
Co-supervisor
Department of Electrical and Electronic Engineering
Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE,

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 21/07/2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ***Ho Yan Bing*** (ID No: ***20UEM02159***) has completed this final year project/ dissertation/ thesis* entitled "**Mitigating Unbalanced And Overlapping Problems Of Large Network Intrusion Data Using Multiple-Level Detection Techniques**" under the supervision of Dr. Khor Kok Chin (Supervisor) from the Department of Internet Engineering and Computer Science, Lee Kong Chian Faculty of Engineering and Science, and Dr. Yap Wun She (Co-Supervisor) from the Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science.

I understand that University will upload softcopy of my dissertation in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

Bing

(*HO YAN BING*)

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name	HO YAN BING
Date	21/07/2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDS	Intrusion Detection System
NB	Naïve Bayes
GNB	Gaussian Naïve Bayes
CART	Classification and Regression Tree
ID3	Iterative Dichotomiser 3
KNN	K-Nearest Neighbours
ANN	Artificial Neural Network
MLP	Multi-Layer Perceptron
SVM	Support Vector Machine
RF	Random Forest
MCS	Multiple Classifiers Systems
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
TPR	True Positive Rate
FPR	False Positive Rate
TLC	Two-Level Classification

CHAPTER 1

INTRODUCTION

1.1 General Information

Nowadays, the Internet has become the primary communication tool for people to work and socialise. Not only that, the data traffic on the Internet has experienced exponential growth mainly due to the increasing number of network devices, such as smartphones, computers, and the Internet of Things (IoT) (Dias *et al.*, 2017). As a result, the high occurrence of network intrusions is inevitable (Kumar *et al.*, 2013). Besides, the difficulties of intrusion detection have increased because of the proliferation of heterogeneous computer networks. According to Malaysia Computer Emergency Response Team (MyCERT , 2020), there were 5,508,357 botnet and malware cases in just a year of 2020. Thus, intrusion detection systems (IDS) are now necessities for today's networks.

Network intrusions will cause data to become inaccessible and insecure. A secured computer network must have three components: data confidentiality, data integrity, and data availability (Kumar *et al.*, 2013; Saba *et al.*, 2022). Therefore, an intrusion detection system (IDS) is necessary to detect and prevent intrusions in the computer network (Tharewal *et al.*, 2022; Wang *et al.*, 2022). The IDS should classify Internet activity as either a normal or intrusive.

1.2 Problem Statement

There are a few challenges of IDS in detecting network intrusion activities. Firstly, the unbalanced class distribution of network intrusion data sets is a challenge for IDS. A data set is unbalanced in class distribution if it contains one or more classes with sizes much bigger than the others. The former is called the majority class, and the latter is called the minority class (Abdelmoumin *et al.*, 2022). The CICIDS2017 data set used in this study contains the normal network activities that occupy up to 80.3% of the data set. Consequently, minority classes have less effect on the detection than the majority class (G. Weiss, 2004), and it may cause low detection rates for the minority classes (network intrusions). It is because general learning algorithms favour the majority classes so that the accuracy can be maximised.

Another challenge of this study is the overlapping class problem which refers to data of different classes that overlap in the same data space (Zoghi *et al.*, 2022). Once the classes are overlapped, it is hard for the learning algorithms to differentiate them. This problem is normally associated with the unbalanced class distribution and will lead to a low detection rate of minority classes. Both unbalanced class distribution and overlapping class problems can be addressed using multiple-level detection techniques (Das *et al.*, 2022).

1.3 Objectives

This study aims to design a multiple-level detection for effectively detecting network intrusions by:

a. Mitigating the unbalanced class distribution problem
b. Mitigating the overlapping class problem

1.4 Scopes and Limitations

There are two research scopes for this study:

a. The proposed intrusion detection technique is developed based on the CICIDS2017 and UNSW-NB15 data sets.
b. The intrusion detection technique is developed using Python.

1.5 Organisation of the Dissertation

The dissertation is organised as follows. Chapter 2 reviews the data sets used in this study, prior works on intrusion detection techniques and evaluation metrics. Chapter 3 gives the methodology for building multiple-level classification models. Chapter 4 shows the results of single classifiers, TLC type I, and TLC type II, and a discussion regarding the results are made as well. Chapter 5 concludes the study and suggests possible future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Initially, this section reviews the network intrusion detection techniques and the data sets used in this research: UNSW-NB15 and CICIDS2017. Then, the single classifiers commonly used for network intrusion detection are reviewed. It is then followed by reviewing prior works related to network intrusion detection, including single classifiers and multiple classifiers systems (MCS) used in multiple-level detection. Lastly, metrics for evaluating the classification approaches are discussed.

2.2 Intrusion Detection

The Internet and computer systems have become essential tools nowadays. They help people in many ways, for example, business operations using websites and e-mail services. However, network usage is growing exponentially, raising many security issues such as network intrusions. Network intrusions refer to unauthorised attempts to bypass the security system of computers or networks and cause undesirable outcomes (H. Liao *et al.*, 2013; Zhang *et al.*, 2019). Therefore, an intrusion detection system (IDS) is needed to tackle network intrusions. IDS monitors the activities in computer systems or networks automatically. Generally, there are two main types of IDS: signature-based and anomaly-based (Tsai *et al.*, 2009; Jyothsna *et al.* , 2011; Otoum *et al.* , 2022)

2.2.1 IDS Types

Signature-based IDS employs Knowledge-based Detection or Misuse Detection. Signature-based IDS classifies the intrusions by comparing the stored signatures of known intrusions to the activities occurring on networks or computer systems. The IDS will alert if suspicious activity is detected as an intrusion. The main advantage of this IDS is that it is easy to develop and understand, provided the behaviour of network intrusions is known. However, this IDS only detects intrusions with characteristics previously stored in the database. Therefore, a signature must be created for every intrusion, and new intrusions cannot be detected (H. J. Liao *et al.*, 2013; Karoriya *et al.*, 2022). Kumar (2012) proposed developing a signature-based IDS using SNORT, an open-source network intrusion detection system. SNORT can analyse real-time data flow in a network and check the packets against signatures written by users.

Anomaly-based IDS assumes that an intrusion will always differ from normal network activities. Therefore, anomaly-based IDS builds profiles of normal network activities and then compares network activities to the profiles. Thus, anomaly-based IDS needs a training phase to develop the profile of normal activities (Karoriya *et al.*, 2022). When there is a difference between the normal and the observed activity, the system will alarm it as an intrusion. Thus, anomaly-based IDS can detect new or unseen intrusions (Kumar *et al.,* 2013).

However, the false positive rate in anomaly-based IDS is higher than in the signature-based IDS.

There are a few shortcomings of current IDS techniques (Lappas *et al.*, 2007; Samrin, 2017; Karoriya *et al.*, 2022):

a. Current IDS is normally adjusted to detect known service-level intrusions without being updated with novel intrusions.
b. Data overload: current IDS cannot analyse the data efficiently because the amount of data is growing rapidly daily.
c. False positive, also known as false alarm generated by current IDS, is a common problem. A false alarm happens when normal network traffic is wrongly classified as intrusions and treated accordingly.

2.2.2 Data mining and machine learning-based IDS

Data mining and machine learning can help to improve IDS by solving the problems mentioned in section 2.2.1 (Lappas *et al.*, 2007; Samrin *et al.*, 2018; Saranyaa *et al.*, 2020; Gümüşbaş *et al.*, 2021). Data mining processes clean, identify and extract patterns from large network intrusion data. Data mining may improve detection, control false alarms and efficiency (Kumar *et al.*, 2013). Machine learning produces classification models that help classify new instances into one predefined class depending on the attribute values. There are different types of classification approaches used in IDS, such as Artificial Neural Networks, Decision Trees, evolutionary algorithms, Rule Induction, Bayesian methods, K-Nearest Neighbours, etc. (Sivatha *et al.*, 2012; Moustafa *et al.*, 2017; Sharafaldin *et al.*, 2018; Ahmad *et al.*, 2019; Bagui *et al.*, 2019; Halimaa *et al.*, 2019; Krishna *et al.*, 2020; Kurniabudi *et al.*, 2020; Rosay *et al.*, 2020; Panigrahi *et al.*, 2021).

2.2.3 Conclusion

In summary, data mining and machine learning techniques can contribute significantly to IDS. Therefore, we focus on building IDS using data mining techniques. The related work, especially classification techniques that were used in IDS, shall be discussed in the later sections.

2.3 Data Set Overview

2.3.1 CICIDS2017

The CICIDS2017 data set used in this study is provided by the Canadian Institute for Cybersecurity (Sharafaldin *et al.*, 2018). This data set has eight different files, every of one them containing five-day network activities. Sharafaldin *et al.* (2018) analysed and evaluated 11 IDS data sets publicly available since 1998. The study showed that most of the early IDS data sets lack traffic diversity, attack diversity, and volume and do not reflect real-world normal network traffic. Further, they contain some unknown and uncorrelated alert traffic, and the payload, protocol information and destination are anonymised. Therefore, CICIDS2017 was created to address all the issues.

Table 2.1 The class distribution of the CICIDS2017 data set.

No	Normal / Attack Label	Number of instances	% of the total instances
1	BENIGN	2,273,097	80.3004
2	DoS Hulk	231,073	8.1630
3	PortScan	158,930	5.6144
4	DDoS	128,027	4.5227
5	DoS GoldenEye	10,293	0.3636
6	FTP-Patator	7,938	0.2804
7	SSH-Patator	5,897	0.2083
8	DoS slowloris	5,796	0.2048
9	DoS Slowhttptest	5,499	0.1943
10	Bot	1,966	0.0695
11	Web Attack - Brute Force	1,507	0.0532
12	Web Attack - XSS	652	0.0230
13	Infiltration	36	0.0013
14	Web Attack - Sql Injection	21	0.0007
15	Heartbleed	11	0.0004
	Total	2,830,743	100.0000

Table 2.1 shows the CICIDS2017 data set containing 2,830,743 instances and 78 features without duplication. It contains 15 classes, one of which is normal network activity, and the other 14 are network intrusions. The data set is highly unbalanced, and the normal network activities take up 80.3%

of the whole data set.

Feature of the data set

The CICIDS2017 data set has a huge feature space. The data set contains 78 features as follows:

1) Destination Port	27) Bwd IAT Mean	53) Average Packet Size
2) Flow Duration	28) Bwd IAT Std	54) Avg Fwd Segment Size
3) Total Fwd Packets	29) Bwd IAT Max	55) Avg Bwd Segment Size
4) Total Backward Packets	30) Bwd IAT Min	56) Fwd Header Length 1
5) Total Length of Fwd Packets	31) Fwd PSH Flags	57) Fwd Avg Bytes/Bulk
6) Total Length of Bwd Packets	32) Bwd PSH Flags	58) Fwd Avg Packets/Bulk
7) Fwd Packet Length Max	33) Fwd URG Flags	59) Fwd Avg Bulk Rate
8) Fwd Packet Length Min	34) Bwd URG Flags	60) Bwd Avg Bytes/Bulk
9) Fwd Packet Length Mean	35) Fwd Header Length	61) Bwd Avg Packets/Bulk
10) Fwd Packet Length Std	36) Bwd Header Length	62) Bwd Avg Bulk Rate
11) Bwd Packet Length Max	37) Fwd Packets/s	63) Subflow Fwd Packets
12) Bwd Packet Length Min	38) Bwd Packets/s	64) Subflow Fwd Bytes
13) Bwd Packet Length Mean	39) Min Packet Length	65) Subflow Bwd Packets
14) Bwd Packet Length Std	40) Max Packet Length	66) Subflow Bwd Bytes
15) Flow Bytes/s	41) Packet Length Mean	67) Init Win bytes_forwa rd
16) Flow Packets/s	42) Packet Length Std	68) Init_Win_bytes_back ward
17) Flow IAT Mean	43) Packet Length Variance	69) act data pkt fwd
18) Flow IAT Std	44) FIN Flag Count	70) min_seg_size_forwar
19) Flow IAT Max	45) SYN Flag Count	71) Active Mean
20) Flow IAT Min	46) RST Flag Count	72) Active Std
21) Fwd IAT Total	47) PSH Flag Count	73) Active Max
22) Fwd IAT Mean	48) ACK Flag Count	74) Active Min
23) Fwd IAT Std	49) URG Flag Count	75) Idle Mean

24) Fwd IAT Max	50) CWE Flag Count	76) Idle Std
25) Fwd IAT Min	51) ECE Flag Count	77) Idle Max
26) Bwd IAT Total	52) Down/Up Ratio	78) Idle Min

Unbalanced class distribution

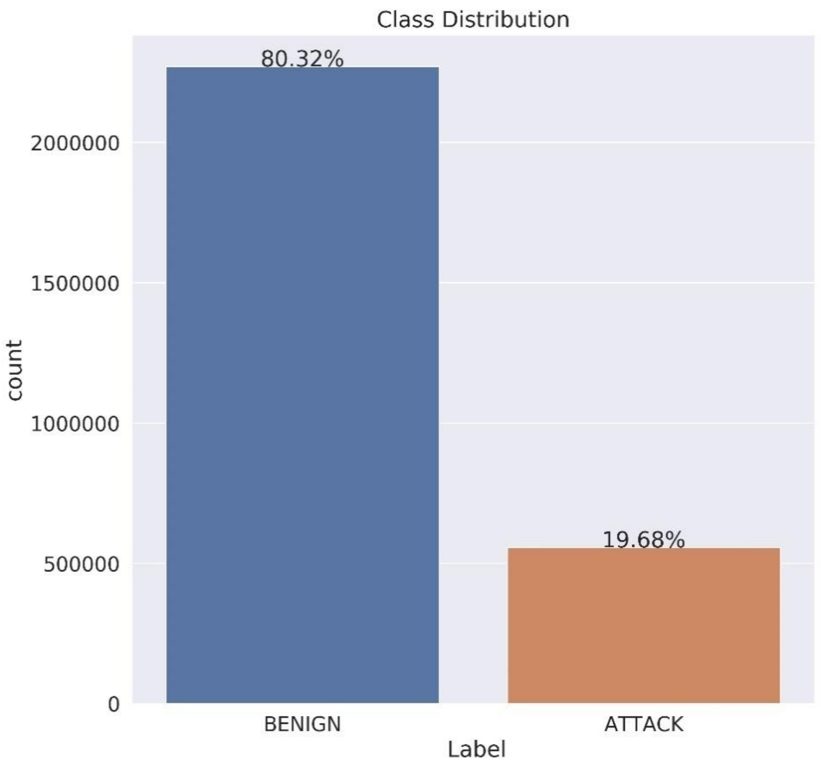

Figure 2.1 The histogram of class distribution after combining the network intrusions into one attack class.

Unbalanced class distribution happens in the CICIDS2017 data set. In this case, the learning algorithm may bias toward the majority class. The normal network activities (BENIGN) are the majority class that occupies 80.32% of the data set, and the 14 network intrusion classes are the minority classes that occupy 19.68% of the data set (Figure 2.1).

Overlapping Class

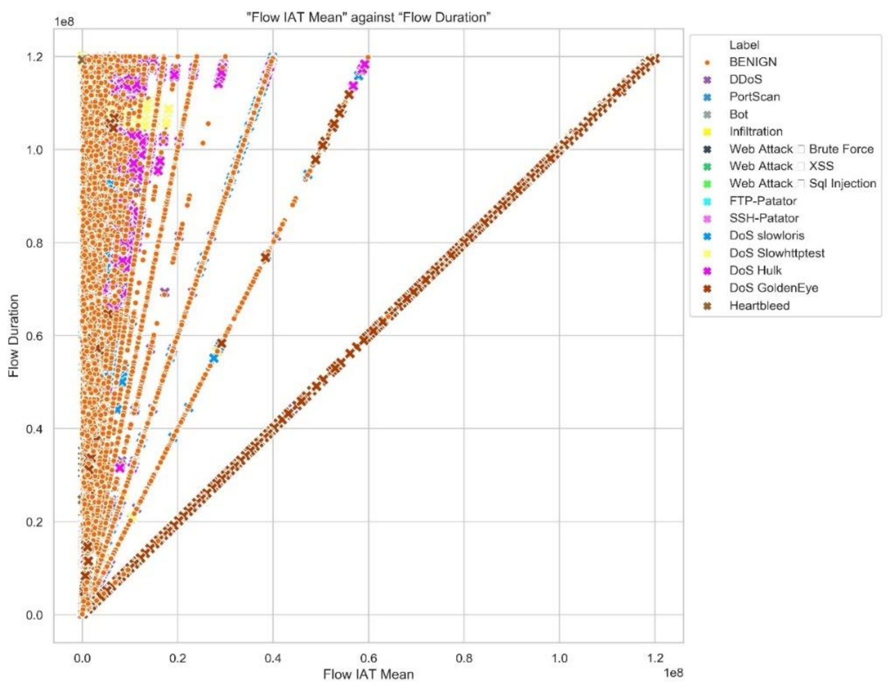

Figure 2.2 The scatter plot of “Flow IAT Mean” against “Flow Duration” from the CICIDS2017 data set

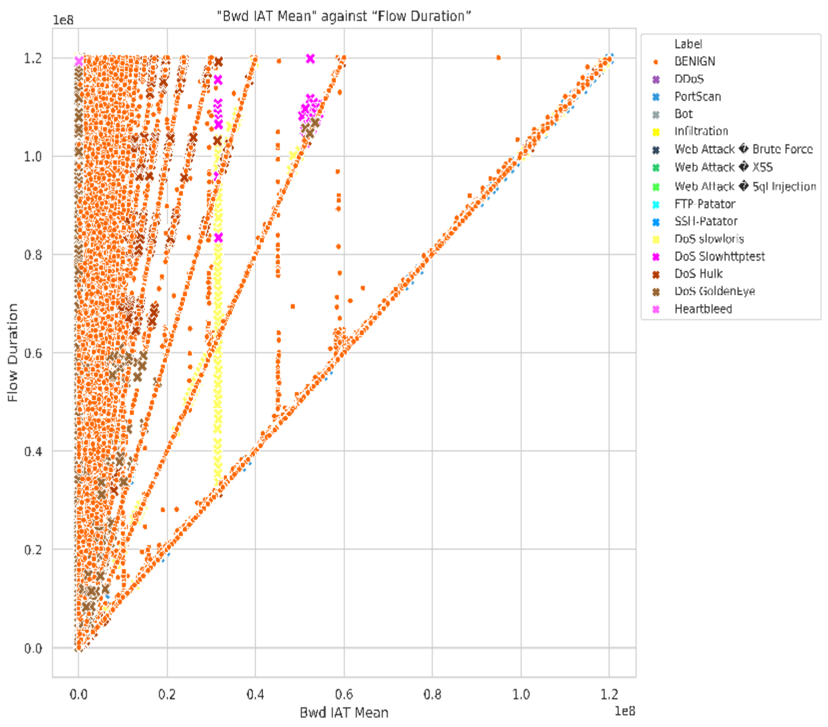

Figure 2.3 The scatter plot of “Bwd IAT Mean” against “Flow Duration” from the CICIDS2017 data set.

Figure 2.2 and Figure 2.3 shows two scatter plots generated from CICIDS2017. The “x” marks represent network intrusions, and the orange dots represent BENIGN. The figures show that the classes in the data set are overlapped with each other.

Missing and Infinity Values

There are a total of 5,734 missing or infinity values. Table 2.2 shows a label that contains missing and infinity values, most of them from BENIGN, DoS Hulk and PostScan. There are several methods to deal with missing and infinity values, such as removing missing values, replacing missing values with a mean value, and replacing infinity values with a maximum value.

Table 2.2 Labels with instances of missing or infinity value

Label	Instances contain missing or infinity value
BENIGN	1,777
DoS Hulk	949
PortScan	126
DDoS	2
DoS GoldenEye	0
FTP-Patator	3
SSH-Patator	0
DoS slowloris	0
DoS Slowhttptest	0
Bot	10
Web Attack - Brute Force	0
Web Attack - XSS	0
Infiltration	0
Web Attack - Sql Injection	0
Heartbleed	0

Types of intrusions

Seven network intrusions (Sharafaldin *et al.*, 2018) included in the CICIDS2017 data set are:

a. Brute force

It is one of the widely used intrusions for password or username cracking. It works by calculating and testing every possible combination of passwords.

b. Heartbleed

It comes from a bug in the OpenSSL cryptography library. An important part of OpenSSL protocols called heartbeat allows two devices to communicate with each other to know they are still connected. Sometimes, one of the devices will send encrypted data called a heartbeat request to the other. Then the second device will reply to prove that the connection is still in place. Heartbleed is executed by making unusual small-sized heartbeat requests with large value in the length field to the server for leaking a victim's information.

c. Botnet

A botnet owner uses numerous Internet-connected devices to perform various tasks: stealing data, sending spams, and allowing intruder access to devices.

d. Denial-of-service (DoS)

The intruder attempts to disable a networked device temporarily. This attack is typically achieved by flooding and paralysing a device with unnecessary connections and stoping legitimate users from making connections.

e. Distributed Denial-of-Service (DDoS)

A DDoS attack requires an intruder to gain control of a network of online machines to carry out an intrusion. This attack happens when multiple systems overload the resources or bandwidth of a targeted device. The goal of DDoS is to cause a service disruption by consuming all the available capacity of devices.

f. Web Attack

An example of a web attack is SQL Injection, which involves an intruder creating a string of SQL commands and then using it to force the database to reply to some information. Another example is Cross-Site Scripting (XSS) which happens when developers do not test their code properly to find the possibility of script injection.

g. Infiltration Attack

The network infiltration from inside is achieved by exploiting the vulnerable software of a victim's computer. After successful exploitation, a backdoor will be formed on the victim's computer. For example, IP sweep, full port scan and service enumerations using Nmap on the victim's network.

2.3.2 UNSW-NB15

This data set was published by the University of New South Wales (UNSW) (Moustafa *et al.*, 2015). This data set was created because old network data sets, such as KDD98, KDDCUP99 and NSLKDD, do not reflect real-world network traffic and have a low footprint attack environment. Therefore, the UNSW-NB15 data set was created to include modern-day normal network activities and synthetic attack vectors. This data set was generated using an IXIA traffic generator with three virtual servers. Servers 1 and 3 are set for normal traffic, while server 2 sets the intrusive activities in the network traffic.

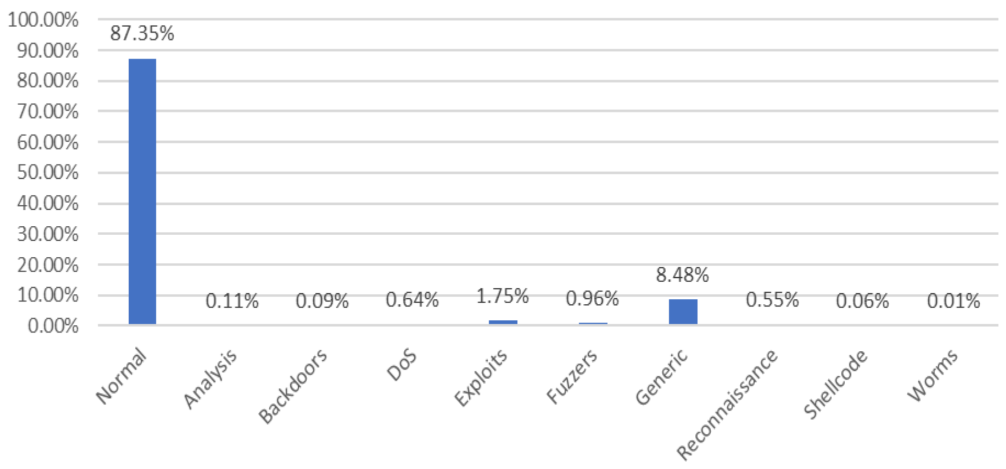

Figure 2.4 The histogram of the class distribution for the UNSW-NB15 data set.

Unbalanced class distribution

Table 2.3 The class distribution of the UNSW-NB15 data set

Class	Sample Size	Percentage
Normal	2,218,764	87.35%
Analysis	2,677	0.11%
Backdoors	2,329	0.09%
DoS	16,353	0.64%
Exploits	44,525	1.75%
Fuzzers	24,246	0.96%
Generic	215,481	8.48%
Reconnaissance	13,987	0.55%
Shellcode	1,511	0.06%
Worms	174	0.01%
Total	2,540,047	100%

From Table 2.3, this data set contains over 2.5 million samples, and normal network traffic takes 87.35% of the data set. This shows that UNSW-NB15 is an unbalanced data set. Most intrusions take less than 1% of the data set, and these classes are the main concern for learning algorithms to learn and detect.

Features of the UNSW-NB15 data set

The 49 features of the UNSW-NB15 data set are:

1) srcip	18) dpkts	35) ackdat
2) sport	19) swin	36) is_sm_ips_ports
3) dstip	20) dwin	37) ct_state_ttl
4) dsport	21) stcpb	38)ct_flw_http_mthd
5) proto	22) dtcpb	39) is_ftp_login
6) state	23) smeansz	40) ct_ftp_cmd
7) dur	24) dmeansz	41) ct srv src
8) sbytes	25) trans_depth	42) ct_srv_dst
9) dbytes	26) res_bdy_len	43) ct_dst_ltm
10) sttl	27) sjit	44) ct_src_ltm
11) dttl	28) djit	45) ct_src_dport_ltm
12) sloss	29) stime	46)ct_dst_sport_ltm
13) dloss	30) ltime	47) ct_dst_src_ltm
14) service	31) sintpkt	48) attack_cat
15) sload	32) dintpkt	49) Label
16) dload	33) tcprtt	
17) spkts	34) synack	

This data set contains six types of features: basic features, content features, time features, general-purpose features, connection features, and labelled features. Table 2.4 shows the description of each feature of the data set.

Table 2.4 Descriptions of the features in the UNSW-NB15 data set.

Features	**Description**
Connection features	The identifier features between hosts, such as client-to-server or server-to-client.
Basic features	The features related to protocol connections.
Content features	The features are related to TCP/IP and HTTP services.
Time features	The features related to time, for example, arrival time between packets, start/end packet time and round-trip time of TCP protocol.
General-purpose features	The features have their purpose of protecting the service of protocols, and they are built from the flow of 100 record connections based on the sequential order of the last time feature.
Labelled features	The features represent the label of each sample.

Overlapped classes of the UNSW-NB15 data set

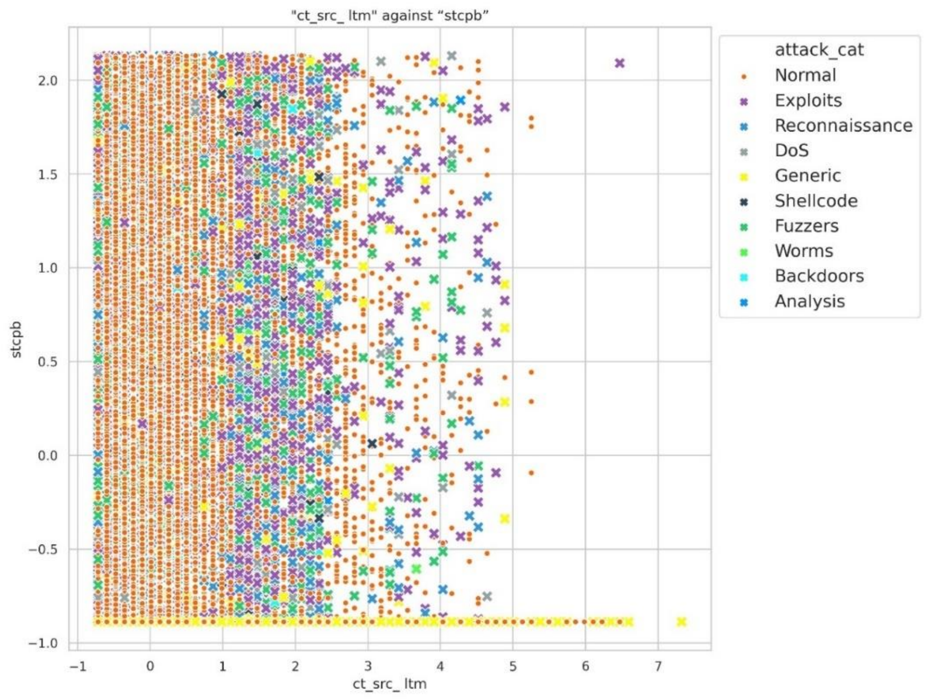

Figure 2.5 The scatter plot of "ct_src_ltm" against "stcpd" from the UNSW-NB15 data set.

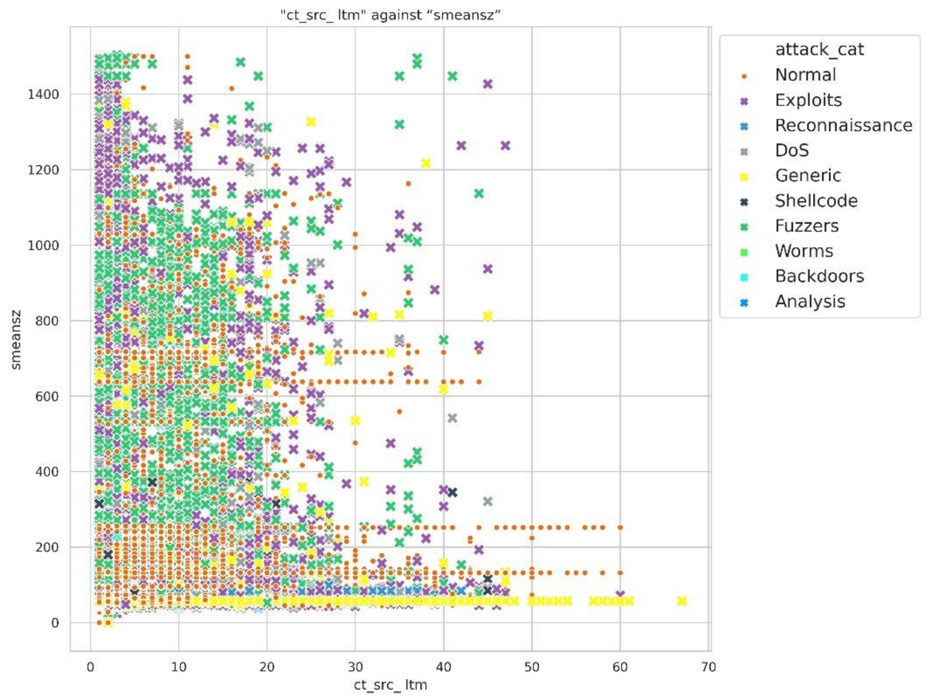

Figure 2.6 The scatter plot of "ct_src_ltm" against "smeansz" from the UNSW-NB15 data set.

Figure 2.5 and Figure 2.6 show the scatter plots generated from the UNSW-NB15 data set. The “x” marks represent network intrusions, and the orange dots represent normal traffic. The plots show that the classes are severely overlapped with each other.

Type of Intrusions in the UNSW-NB15 data set

There are ten classes in this UNSW-NB15 data set: normal and nine intrusions. Table 2.5 shows the type of intrusions and their description.

Table 2. 5 The description of intrusions in the UNSW-NB15 data set

Intrusion	Description
Fuzzers	Attempting to fuzz a program or network by feeding the random data traffic and causing unexpected crashes.
Analysis	It includes different intrusions of the port scan, spam and HTML file penetrations.
Backdoors	It is used to bypass a system's security and authentication to reach a computer or its data.
DoS	A common intrusion causes a server or a network resource to be flooded with traffic, making it inaccessible to the user.
Exploits	It causes unintended or unanticipated behaviour to happen on computer software by taking advantage of a known bug or vulnerability of the systems
Generic	It is a technique against all block-ciphers (with a given block and key size) without considering their structure.
Reconnaissance	It includes packet sniffing, ping sweeping, port scanning and phishing which can simulate intrusions to gather data.
Shellcode	It injects code remotely to exploit a variety of software vulnerabilities.

Worms	It replicates itself to spread to another computer via software vulnerabilities and causes unintended behaviour on the target computer.

2.3.3 Conclusion

Researchers commonly use the CICIDS2017 and UNSW-NB15 data sets to build intrusion detection models. These two data sets contain the latest and diversified intrusions, which reflect real-world network intrusion. Not only that, they also share common problems, which are unbalanced class distribution and overlapping classes. Therefore, these two data sets were selected to evaluate the proposed detection technique in this study.

2.4 Single Classifiers

2.4.1 Naïve Bayes Classifier

Naïve Bayes classifier performs classification based on Bayes' Theorem. This classifier assumes that all features independently and equally contribute to the probability of the class (Berrar, 2018).

Bayes' Theorem calculates the probability of an event happening provided that the probability of another event that has occurred. With regards to a data set, Bayes' Theorem is stated in the following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Where y is the class variable and X is evidence.

- $P(y|X)$ is a posteriori probability, which is the probability of class y, given evidence X is true.
- $P(y)$ is a prior probability, which is the probability of class y before evidence is seen.
- $P(X|y)$ is a likelihood which is the probability of class y after evidence X is seen

Where X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Hence, we reach the result:

$$P(y|x_1,\dots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\dots P(x_n)}$$

When each instance is only matched with one class, calculate the value of the numerator for each class and choose that class in which the value is maximal. This rule is called the maximum posterior rule. The class with maximum posterior is known as the maximum a posteriori (MAP) class and can be calculated $\hat{y}$ for the instance x as follows:

$$\hat{y} = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Pseudocode 1: The Naïve Bayes Classifier (adopted from Firman *et al.*, (2018)**)**

```
Input:  Training data set T,

Features of data set, X= $(x_1, x_2, x_3, \dots, x_n)$

Output: Predicted class of testing data set.

Begin

    1. Read the training data set.
    2. Calculate the probability of X using the Bayes
       theorem in each class.
    3. Calculate the likelihood for each class.
    4. Get the maximum likelihood.

End
```

They are a few examples of Naïve Bayes as follows:

1. **Gaussian Naïve Bayes**: Continuous values associated with each feature are assumed to be distributed according to a normal distribution.
2. **Multinomial Naïve Bayes**: Feature vectors represent the frequencies with which certain occurrences have been created by a **multinomial distribution**. This algorithm is usually used for text classification.
3. **Bernoulli Naïve Bayes**: This algorithm is used for data that are distributed according to multivariate Bernoulli distributions. Each feature is assumed to be a binary-valued variable. For example, this algorithm is used to detect whether a word occurs in a document or not.

There are two advantages of using Naïve Bayes. Firstly, real-time prediction because the Naïve Bayes classifier is a fast learning algorithm. Secondly, multi-class prediction because the Naïve Bayes classifier can predict the probability of multiple variable classes. However, the main disadvantage of Naïve Bayes is that it assumes the mutual independence of all variables involved, which is almost impossible in real life.

2.4.2 Iterative Dichotomiser 3 (ID3)

ID3 is a learning algorithm that generates the decision tree based on the entropy of every feature of the data set (Quinlan, 1986). The entropy measures the amount of uncertainty in the data set and builds a tree by selecting the best feature that yields minimum entropy. However, ID3 can only deal with categorical data. It can build a model in a short time taken, but it cannot deal with missing values.

$$Entropy = \sum_{i=1}^{C} -p_i * log_2(p_i)$$

Where,

p_i = The proportion of the number of elements in class i to the number of elements in the data set.

Pseudocode 2: ID3 (adopted from Quinlan (1986))

```
Input: Data set
Output: ID3 Decision Tree
Begin
    1. Calculate the entropy of every feature of the data set.
    2. Split the data set into subsets using the feature for which the
       resulting entropy after splitting is minimised.
    3. Make a decision tree node containing that feature.
    4. Repeat steps 2 and 3 for the remaining features.
End
```

2.4.3 Classification And Regression Tree (CART)

The classification and regression tree (CART) algorithm is a set of if-then (split) conditions that permit predictions or classification of cases (Breiman *et al.*, 2017). The CART algorithm is scalable to large data set problems and can also handle small data sets. In the CART algorithm, binary trees are built with the Gini index.

The Gini index measures the degree to of a particular variable is misclassified when it is randomly chosen. The Gini index varies from 0 to 1, where 0 means that all elements belong to a certain class or exist in only one class, and 1 means that the elements are randomly distributed across various classes. Therefore, in the CART algorithm, a variable split with a low Gini Index is chosen to split the node.

$$Gini\ (D) = 1 - \sum_{i=1}^{n} (p_i)^2$$

where p_i is the probability of an object being classified to a particular class in the data set D with n classes.

$$Gini_A\ (D) = \frac{|D_1|}{|D|} Gini\ (D_1) - \frac{|D_2|}{|D|} Gini\ (D_2)$$

where a variable A of data set D is split into two subsets D_1 and D_2.

Pseudocode 3: CART algorithm (adopted from Breiman *et al.* (2017))

```
Input:  Data set
Output: CART decision tree
Begin
    1. Generate the Root node of the tree.
    2. Calculate the Gini index for all features.
    3. The feature with minimum Gini index is chosen and use it to split
       the node into two child nodes.
    4. Repeat steps 2-3 until a stopping criterion is reached.
End
```

There are three advantages of the CART algorithm (Singh, 2014). Firstly, CART can deal with both continuous and categorical variables. Next, the CART algorithm identifies the most important features and removes the irrelevant variables. Finally, CART can deal with outliers. However, the disadvantage of the CART algorithm is that it may create an unstable tree that can cause variance when a small change in the data set happens.

2.4.4 K-Nearest Neighbours (KNN)

The KNN algorithm assumes that a class with similarity will exist in close proximity. This assumption provides a nonparametric procedure for predicting an unseen class by finding the distances between the unseen class and all the instances in the data set (Cover *et al.*, 1967; Keller *et al.*, 1985). A K value refers to the number of nearest neighbours to an instance. The K value closest to the unseen class is selected then the assignment of the unseen class is according to the vote of the most frequent class with K.

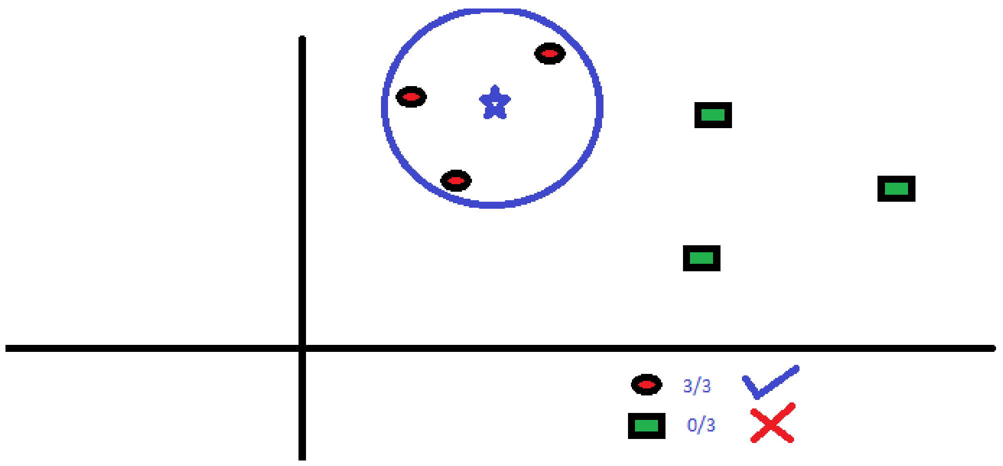

Figure 2.7 The illustration of labelling a new instance using KNN. Adopted from Dixit *et al.* (2019).

Figure 2.7 shows the illustration of labelling a new instance using KNN. In this case, the K value is assigned to 3. The “star” represents a new instance with three nearest neighbours (red oval). Therefore the new instance is labelled as a red oval.

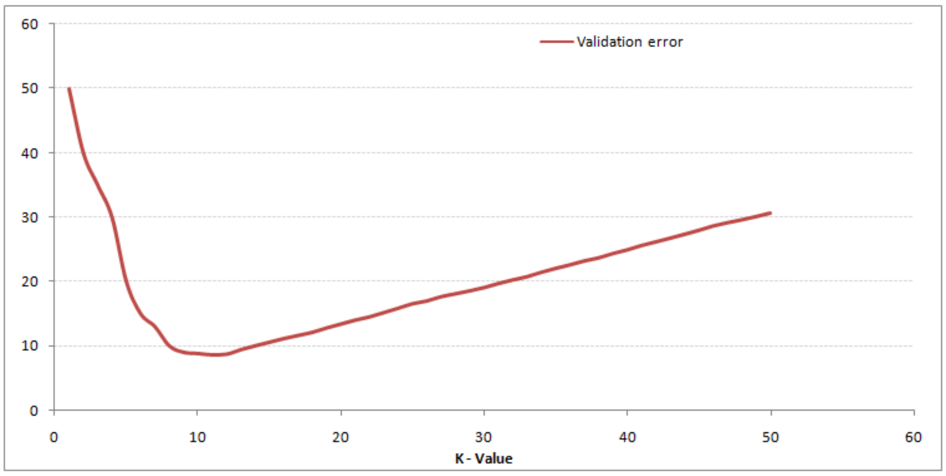

Figure 2.8 The graph of K-value against validation error. Adopted from Dixit *et al.* (2019).

Selecting the best K value will help reduce the classifier's error while maintaining the accuracy of the predictions of the unseen data. For the binary-class problem, the K value is restricted to only odd values to avoid a tie. Figure 2.8 shows that validation error drops initially to reach a minimum value and then increases when K becomes high. Several K values are evaluated to find the best K value with the lowest validation error. This method is called the elbow method.

Pseudocode 4: The KNN algorithm (adopted from Sarkar *et al.* (2000))

```
Input: training data set
Output: label of the test data set
Begin
    1. Load the training data set.
    2. Initialise the value of K.
    3. Calculate the distance between test instances and training
       instances from the data set.
    4. Sorted the distances in ascending order.
    5. Pick the first K entries from the sorted list.
    6. Get the labels of the selected K entries.
    7. Return the majority label.
End
```

There are three advantages of the KNN algorithm. Firstly, the KNN algorithm is simple and easy to implement. Only two parameters are needed to apply KNN: K values and distance metrics, such as Euclidean, Manhattan, Minkowski, etc. Next, KNN does not build any model and needs a training phase. It is because KNN does not learn in the training phase; it simplifies labelling new data based on historical data. Lastly, KNN can be easily implemented for multi-class problems without extra effort.

However, there are also a few disadvantages of the KNN algorithm. Firstly, KNN has limitations in dealing with large and high dimensional data sets. It becomes difficult for the KNN to calculate the distance between instances. Next, KNN is sensitive to noise, missing values and outliers. Lastly, KNN cannot deal with unbalanced data.

2.4.5 Artificial Neural Networks (ANN)

The design of ANN is inspired by the biological nervous system. It imitates the process of the brain processing information (Hornik *et al.*, 1989). An ANN model consists of three layers: input layer, hidden layer and output layer. The model only consists of one input layer and one output layer. The number of hidden layers is based on the objective of the model. Figure 2.9 shows an ANN model with two hidden layers, and the hidden layers consist of interconnected nodes. The ANN is described as fully connected with every node in the next and previous layers (Choraś *et al.*, 2021). The input layer has no computational role as it only passes data to the network. The hidden layer performs computation and transfers data from input nodes to output nodes.

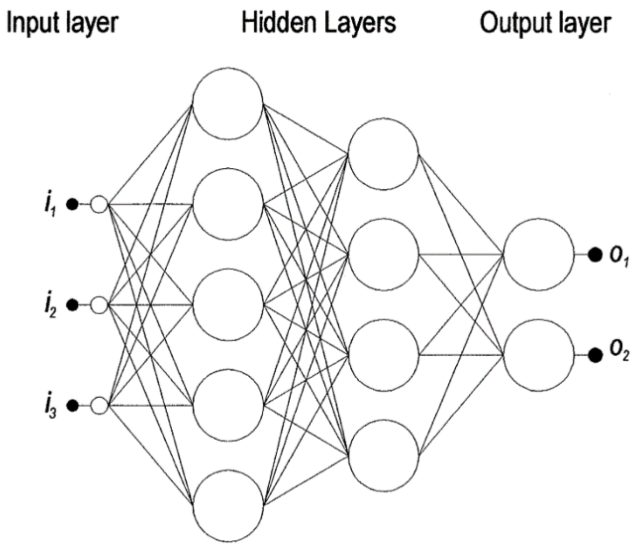

Figure 2.9 An ANN model with hidden layers. Adopted from Dias *et al.* (2017)

Nodes / Neurons

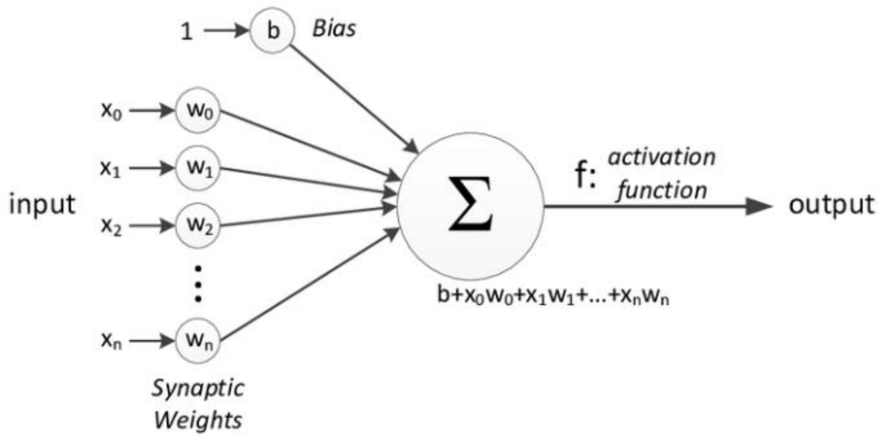

Figure 2.10 A node in the ANN model. **Adopted from Dias *et al.* (2017).**

From Figure 2.10, the equation of output is defined as,

$$output = f(b + \sum X_i W_i)$$

Where,

f is the activation function.

b is the bias.

X is the input data.

W is the weight of inputs.

Figure 2.10 shows a single node, the basic computation unit in an ANN. Each input (X) is associated with a weight (W) which is the importance of the output to other input. A node contains a bias value to shift the activation function to either right or left. The activation function normalises the output of the node.

Types of activation functions (Goodfellow *et al.*, 2016):

1. **Binary Step Function**

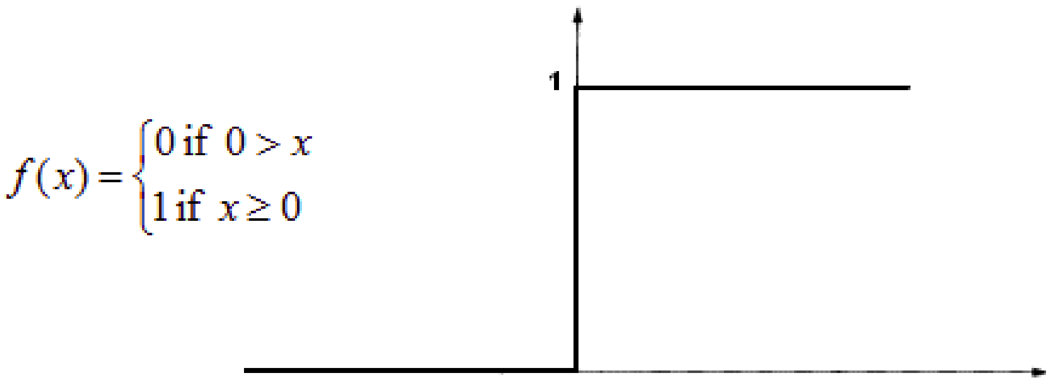

Figure 2.11 Binary step function. Adopted from Sharma *et al.* (2020).

Figure 2.11 shows a binary step function. If the input is above or below a certain value, then the node will be activated and send the same value to the next layer. However, this function cannot be used when dealing with multiple classes problem.

2. **Sigmoid Function**

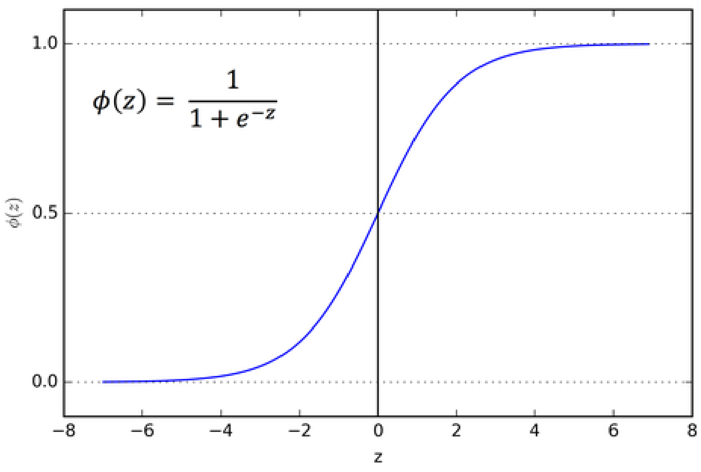

Figure 2.12 Sigmoid function. Adopted from Sharma *et al.* (2020).

The sigmoid function is an S-shaped curve, which maps any real number into a value from 0 to 1 but never reaches those limits. Therefore, it is used for models where the prediction of output probability is needed. The advantages of this function are providing a smooth gradient and normalising the output of each node. The disadvantage of this function is that it will vanish the gradient, which means that when the input value is very high or very low, the output value remains almost unchanged. It will result in the slow updating of the ANN parameters.

3. **Hyperbolic Tangent Function (tanh)**

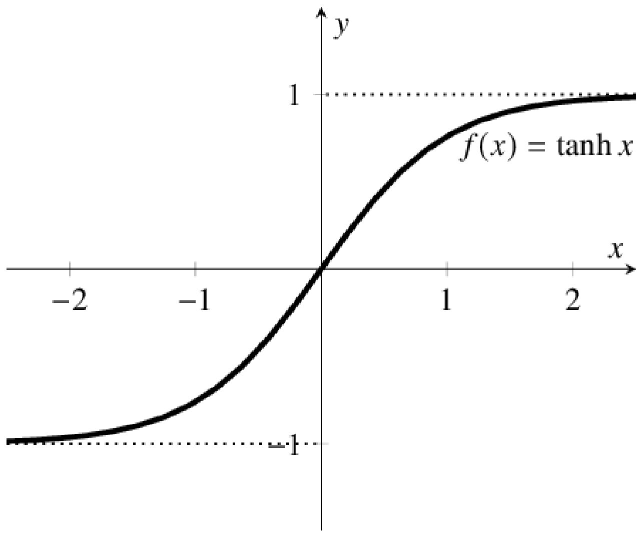

Figure 2.13 Hyperbolic Tangent Function. Adopted from Sharma *et al.* (2020).

The hyperbolic tangent function is similar to the sigmoid function but ranges vary between -1 to 1. The advantage of this function is that a negative input can be mapped to a negative output. So, the neural network is less likely to get stuck during the training phase. The advantages and disadvantages of this function are similar to the sigmoid function. Additionally, this function is zerocentred, which makes the ANN easier to deal with inputs with very high, neutral and very low values.

4. Rectified Linear Unit (ReLu)

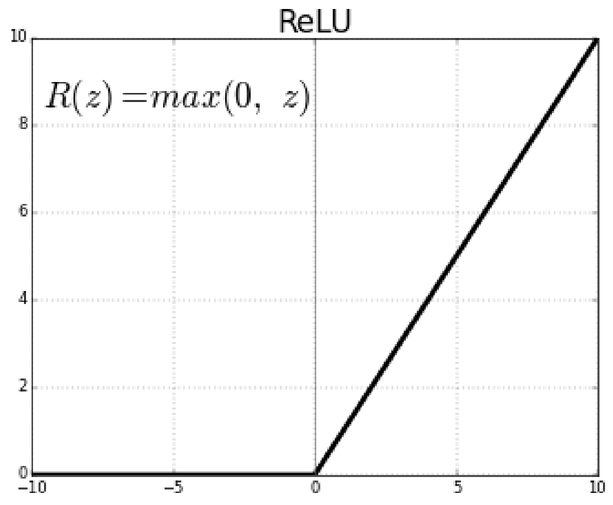

Figure 2.14 Rectified Linear Unit (ReLU). Adopted from Sharma *et al.* (2020).

ReLu function is a simple calculation that outputs the same value when the value is more than 0. Otherwise, the output value will be 0. The advantage of this function is its computational efficiency. The disadvantage of this function is that the ANN cannot perform backpropagation if inputs are negative because the function's gradient becomes 0.

Backpropagation

When training, the ANN performs learning at this layer. An error value is calculated based on the prediction and the actual value (Chiba *et al.*, 2018). The error value calculated is sent back through the ANN model to adjust the weight for the next input. The lower the error value, the closer the difference between the actual and prediction values. As long as there is still a difference, the weight adjustment is needed. This process is known as Back-propagation and is iterated in ANN until the error value is minimised.

Pseudocode 5: ANN (adopted from Saad Assiri (2021))

```
Input: Training data set
Output: Label of the test data set
Begin
    1. Initialise the input layer, output layer and hidden layer of the ANN.
    2. Loads the training data set into the model.
    3. Perform calculations at the nodes.
    4. Generates an output at the output layer.
    5. Calculates the error value between the actual value and the output.
    6. Update the weight of nodes according to the error value.
    7. Repeat steps 2 to 6 until all the training instances had been loaded
       into the model.
    8. Generate the prediction of the test data set.
End
```

There are two advantages of the ANN. Firstly, ANN can learn non-linear and complex problems. This enables ANN to solve real-life problems such as image processing and forecasting. Not only that, ANN can handle missing values. If a data set contains missing values, the computation at nodes can still go on. However, there are some disadvantages of the ANN. Firstly, there is no rule to decide the ANN architecture. The ANN architecture includes the number of nodes in hidden layers and the number of hidden layers. A suitable architecture is mostly achieved based on experience or trial and error. In addition, the training process is slow when the data set dimension is large.

2.5 Prior works

2.5.1 Single Classifiers on Intrusion Detection

Single classifiers refer to the classifier built using one single learning algorithm. The common single classifiers for intrusion detection are Decision Tree, K-Nearest Neighbour, Artificial Neural Network and Naïve Bayes (Sivatha *et al.*, 2012; Moustafa *et al.*, 2017; Sharafaldin *et al.*, 2018; Ahmad *et al.*, 2019; Bagui *et al.*, 2019; Halimaa *et al.* , 2019; Krishna *et al.*, 2020; Kurniabudi *et al.*, 2020; Rosay *et al.*, 2020; Panigrahi *et al.*, 2021).

Table 2.6 Prior works used single classifiers on the UNSW-NB15 data set.

	Khammassi *et al.* (2017)	**Bagui *et al.* (2019)**		**Ahmad *et al.* (2021)**		**Moustafa *et al.* (2016)**
	C4.5	NB	J48	SVM	ANN	DT
Normal	0.90720	-	-	1.0000	0.9860	-
Fuzzer	0.69112	0.9670	0.7518	0.9520	0.5800	-
Analysis	0.09929	0.9099	0.0664	0.8720	0.7280	-
Backdoors	0.06925	0.9296	0.0222	0.9790	0.8100	-
DoS	0.04113	0.8417	0.0599	0.7620	0.6670	-
Exploits	0.92317	0.9067	0.6427	0.9050	0.8390	-
Generics	0.97937	0.9661	0.9780	0.9790	0.9580	-
Reconnaisance	0.76150	0.9759	0.7950	0.9830	0.9080	-
Shellcode	0.47468	0.9815	0.4312	0.6850	0.5420	-
Worms	0.38462	0.9318	0.4090	-	-	-
Accuracy	**0.8142**	**-**	**-**	**0.9567**	**0.9167**	**0.8556**

Table 2.6 compares prior works that used single classifiers on the UNSW-NB15 data set. Overall, the single classifiers perform well in accuracy but still suffer low detection rates (or true positive rate, TPR) for most intrusion classes. Some works did not show the TPR of each class, i.e., Kasongo *et al.* (2020) and Moustafa *et al.*(2016). Thus, the classifiers' performance on the minority intrusion classes cannot be determined.

Table 2.7 Prior works that used single classifiers on the CICIDS2017 data set.

	Alrowaily *et al.* (2019)		**Ferrag *et al.* (2020)**		**Krishna *et al.* (2020)**	**Kurniabudi *et al.* (2020)**	
	KNN	MLP	RepTree	J48	KNN	J48	RT
BENIGN	-	-	0.9517	0.9496	-	0.9990	0.9980
Bot	-	-	0.4776	0.4776	-	0.6980	0.7320
DDoS	-	-	0.9979	0.9979	-		
DoS GoldenEye	-	-	0.6643	0.6729	-		
DoS Hulk	-	-	0.9222	0.9360	-		
DoS Slowhttptest	-	-	0.7536	0.8033	-	0.9990	0.9970
DoS slowloris	-	-	0.9273	0.9388	-		
Heartbleed	-	-	1.0000	1.0000	-		
Infiltration	-	-	0.8333	0.6667	-	0.0000	0.0000
PortScan	-	-	0.9988	0.9857	-	0.9990	0.9930
FTP-Patator	-	-	0.9918	0.9955	-	0.9930	0.9920
SSH-Patator	-	-	1.0000	1.0000	-		
WA - Brute Force	-	-	0.7082	0.6041	-		
WA - Sql Injection	-	-	0.5000	0.5000	-	0.9490	0.9250
WA - XSS	-	-	0.3250	0.4125	-		
Accuracy	0.9946	0.9626	0.9340	0.9348	0.9984	0.9987	0.9976

Table 2.7 compares the prior works that used single classifiers on the unbalanced and overlapped CICIDS2017 data set. Overall, single classifiers gave high accuracies. However, this does not mean that the detection rate for each intrusion class is good. Most authors do not show the detection rate for each class, and the performance against the intrusions cannot be determined. However, the work conducted by Kurniabudi *et al.*(2020) showed a true positive rate for each class. Their study shows that the minority classes, Infiltration and Bot, suffer from low and average detection rates.

2.5.2 Multiple Classifiers Systems (MCS)

A Multiple Classifiers System (MCS) refers to multiple classifications that aim to produce a better prediction than single classifier systems. MCS minimises overlapping class and unbalanced class distribution problems (Podolak, 2008; Woźniak *et al.*, 2014; Song *et al.*, 2021). There are three types of MCS: (i) Serial, (ii) Parallel and (iii) Hybrid (Figure 2).

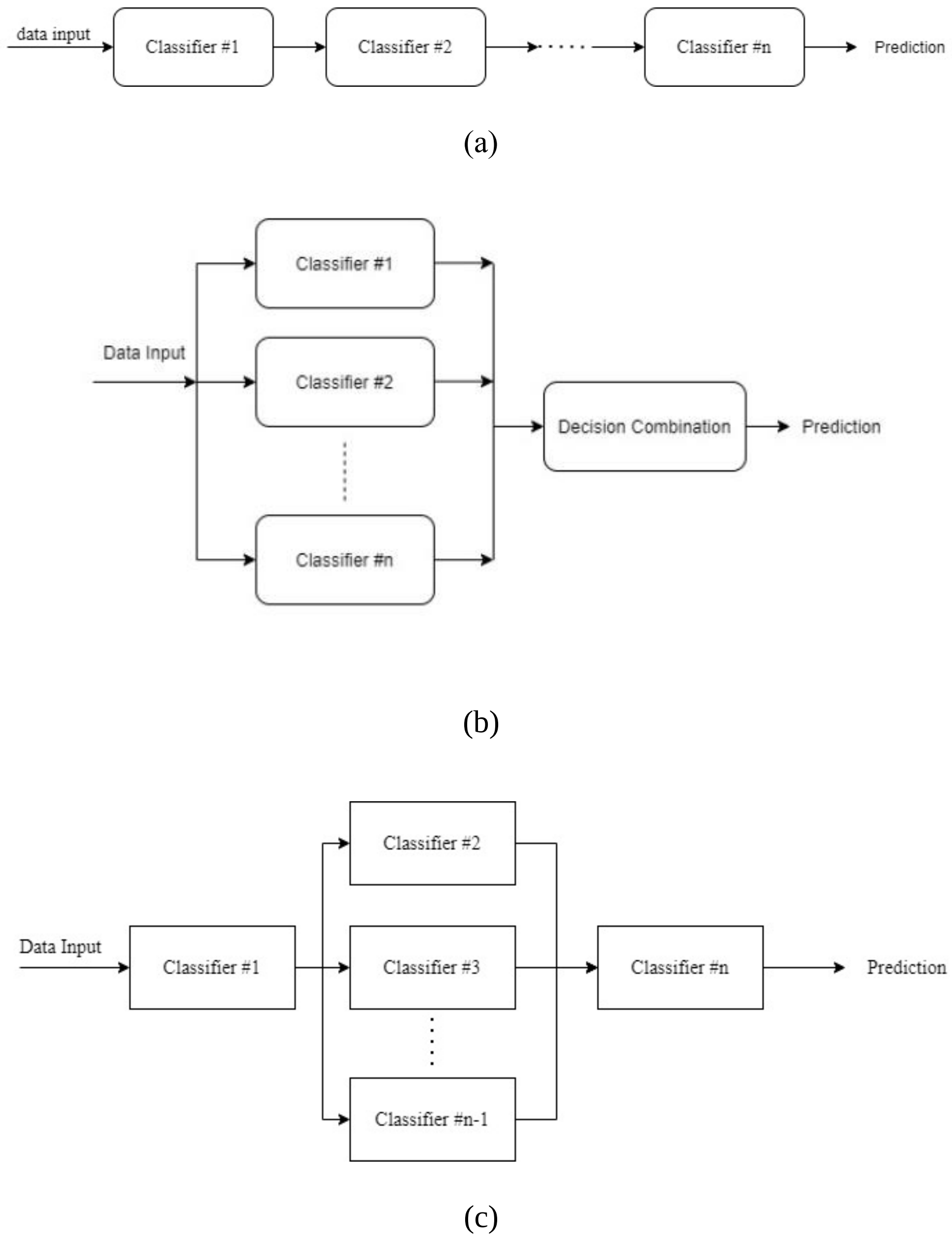

Figure 2.15 Various MCS combinations: (a) serial, (b) parallel, and (c) hybrid. Adopted from Rahman *et al.* (2003); Mohandes *et al.* (2018)

As shown in Figure 2.15(a), the serial combination of MCS interconnected the classifiers sequentially, which means the result from the first classifier shall load to the next classifier for further processing (Kim *et al.*, 2002, Mohandes *et al.*, 2018). The classifier arrangement in a serial combination of MCS is important for performance. For example, the first classifier detects the general intrusion classes and the subsequent classifiers detect detail intrusion types.

Xiang *et al.* (2008) proposed a serial classifier that used decision trees and Bayesian clustering. The network data set used for this study was the KDDCUP99. For this study, a model with a four-level classification was proposed. At the first level, the decision tree was used to categorise three classes: DoS, Probe and Others (U2R, R2L and Normal traffic). U2R, R2L and Normal traffic were grouped because they had similar criteria and were difficult to differentiate. At the second level, Bayesian clustering was used to further classify U2R, R2L and Normal traffic by labelling U2R and R2L as “Attack”; therefore, they have only two classes (Attack and Normal) at this level. At the third level, decision trees were used to classify U2R and R2L. The final level further classified the attack type into a more detailed attack class by using decision trees. This proposed approach achieved a low false alarm of 3.23%. Yulianto *et al.* (2019) proposed a classification model using Adaptive Boosting (AdaBoost) on the CICIDS2017 date set. With AdaBoost, several base learners are serially combined, and the weight of the next base learner shall be updated based on the performance of the previous base learner (Freund *et al.,* 1999). This proposed model achieved an overall accuracy of 81.83%.

The parallel combination of MCS usually loads a data set to several classifiers simultaneously (Kim *et al.*, 2002). Then, the final result is based on a decision combination algorithm, as shown in Figure 2.15(b). Majority voting is the common decision combination method used in parallel MCS. This method decides the final result depending on an agreement for more than half of the classifiers with the same classification decision.

Swami *et al.* (2020) proposed a voting-based IDS approach for intrusion detection. The data set used for this approach is UNSW-NB15, CICIDS2017 and NSL-KDD data set. The data set is loaded into three different base classifiers, and the final decision is based on the output of the base classifiers. This proposed model achieved 88.05% accuracy for the UNSW-NB15, 97.77% for the CICIDS2017 data set and 99.68% for the NSL-KDD data set.

The hybrid MCS combines a serial MCS and a parallel MCS, as shown in Figure 2.15(c). The input data is loaded to the first classifier, and the output from the first classifier shall be the input to a parallel combination MCS. Then, a single classifier shall merge the output of the previous parallel combination MCS. The hybrid combination of MCS inherits advantages from serial and parallel MCS, which detect certain classes serially and use majority voting parallelly to decide the output.

Azzaoui *et al.* (2020) proposed a two-stage hybrid approach for intrusion detection. The data sets used for this study were CICIDS2017 and NSL-KDD.

The proposed approach had classifiers arranged serially, but a parallel classifier was applied at the first stage. Data were re-labelled into two classes at the first stage: Normal and Anomaly, classified using a random forest classifier that was arranged parallelly in the proposed approach. The instances classified as intrusions were then sent to the second stage. KNN was used at the second stage, and instances were further classified into intrusion classes. This proposed model achieved 99.21% accuracy for the CICIDS2017 data set and 99.56% for the NSL-KDD data set.

The results of prior works that used MCS on UNSW-NB15 and CICIDS2017 are summarised in Table 2.8 and Table 2.9.

Table 2.8 Prior works that used MCS on the UNSW-NB15 data set.

	Papamartzivanos *et al.* (2018)	**Ren *et al.*, (2019)**	**Swami *et al.* (2020)**	**Tama *et al.* (2020)**
	Serial MCS (GA + DT)	Parallel MCS (RF)	Parallel MCS (Voting)	Hybrid MCS (RF + Bagging)
Normal	0.9739	0.9670	-	-
Fuzzers	0.6442	0.3810	-	-
Analysis	0.2045	0.0610	-	-
Backdoors	0.6732	0.4030	-	-
DoS	0.1429	0.4610	-	-
Exploits	0.7622	0.6630	-	-
Generics	0.8137	0.9690	-	-
Reconnaisance	0.4607	0.8200	-	-
Shellcode	0.3639	0.7800	-	-
Worms	0.1818	0.7950	-	-
Accuracy	0.8433	0.9280	0.8929	0.9245

Table 2.9 Prior works used MCS on the CICIDS2017 data set.

	Yulianto *et al.* (2019)	**Swami *et al.* (2020)**	**Ferrag et al. (2020)**		**Azzaoui *et al.* (2020)**
	Serial MCS (AdaBoost + EFS + SMOTE)	Parallel MCS (Voting -RF, k-NN, and MLP)	Parallel MCS (RF)	Hybrid MCS (REP Tree, Jrip, Random Forest)	Hybrid MCS (Random Forest + k-Nearest Neighbour)
BENIGN / Normal	-	-	0.9812	0.9886	-
Bot	-	-	0.4968	0.4647	-
DDoS	-	-	0.9982	0.9988	0.9996
DoS GoldenEye	-	-	0.6757	0.6757	0.9990
DoS Hulk	-	-	0.9516	0.9678	
DoS Slowhttptest	-	-	0.8135	0.9384	
DoS slowloris	-	-	0.9376	0.9776	
FTP-Patator	-	-	0.9973	0.9963	-
Heartbleed	-	-	1.0000	1.0000	-
Infiltration	-	-	0.8333	1.0000	0.7916
PortScan	-	-	0.9988	0.9988	0.9995
SSH-Patator	-	-	0.9982	0.9991	-
WA - Brute Force	-	-	0.7041	0.7327	0.9990
WA - Sql Injection	-	-	1.0000	0.5000	0.4625
WA - XSS	-	-	0.3750	0.3063	0.9622
Accuracy	0.8183	0.9777	0.9559	0.9667	0.9921

2.5.3 Conclusion

The intrusions, which are the minorities, are the main concern for the research of intrusion detection. However, the prior works show that the detection rates for the minority intrusion classes are unsatisfactory. Single classifiers do not truly mitigate the unbalanced and overlapped problems with a low detection rate on minorities. On the other hand, MCS can improve the detection rate of minorities, but there is still room for improvement, especially for some intrusion classes. Overall, single classifiers produced lower overall accuracies than the MCS.

Serial MCS have better properties to mitigate the unbalanced and overlapped problems. It is because of the structure of the serial MCS. Serial MCS interconnected the classifiers sequentially, and the result from the first classifier shall load to the next classifier for further processing will help mitigate the unbalanced problem, for instance, by detecting the majority class at the first level of the classifier and then loading the data set to the second classifier without the interference of the majority class. Serial MCS will also help mitigate the problems of the overlapping classes since the majority class is usually the main cause of the overlapping because the majority class will usually occupy a large area in the feature space and overlap with the minority class.

For parallel MCS, the data set is loaded into several classifiers simultaneously, and then all the classifiers may still suffer from the problem of the unbalanced and overlapped data set. Same for hybrid MCS because it contains the properties of the parallel MCS.

For these reasons, it motivated us to use serial MCS for the design of our work to mitigate the unbalanced class distribution and overlapping classes problems better.

2.6 Evaluation Metrics

Evaluation metrics are an essential part of evaluating the performance of a classifier. The metrics measure and summarise a classifier’s quality when testing it with new unseen data. The classification problem can be categorised into binary, multi-class and multi-labelled classification. This study only focused on evaluation metrics for binary and multi-class classification. The evaluation metrics were used to select the best learning algorithm for the classification problem.

In this study, positive represents normal traffics, and negative represents intrusions. Several evaluation metrics derived from the confusion matrix are shown in Table 2.10.

Table 2.10 The confusion matrix for a two-class classification problem.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- True positive (TP) represents the normal traffic correctly classified.
- False positive (FP) represents the normal traffic misclassified as intrusion traffic.
- True negative (TN) represents the intrusion traffic correctly classified.
- False negative (FN) represents the intrusion traffic misclassified as normal traffic.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy measures the ratio of the total number of correctly classified instances and the total number of instances (Salih *et al.*, 2021), as in the equation shown above. Many of the reviewed prior works used this metric because it is the simplest way to review the overall classifier's performance. However, accuracy is not a suitable evaluation metric for unbalanced data sets. The overall accuracy could be high because majority classes are well classified, even though the detection rate for minority classes is low (Thabtah *et al.*, 2020).

$$TPR = \frac{TN}{TN+FP}$$

True Positive Rate (TPR), also known as Recall, refers to the percentage of intrusions correctly detected over the total number of actual intrusions (equation shown above) (Salih *et al.*, 2021). TPR can separately review the detection rate for both majority and minority classes. For intrusion detection studies, minority intrusions are the main concern. Therefore, several prior works (Xiang *et al.*, 2008; Dias *et al.*, 2017; Wang *et al.*, 2020; Song *et al.*, 2021) used TPR as the evaluation metric.

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

False Positive Rate (FPR), also known as False Alarm, refers to the percentage of normal traffic classified as intrusions (Salih *et al.*, 2021), as in the equation shown above. In intrusion detection, a false positive is an intrusion alert caused by normal traffic that is classified wrongly as intrusions. A few prior works used FPR as the evaluation metric (Vijayanand *et al.*, 2018; Wang *et al.*, 2020).

2.6.1 Conclusion

We used True Positive Rate (TPR) and accuracy as the evaluation metrics in this study. Both data sets used in this study, especially the CICIDS2017 data set, are unbalanced. If the majority class is correctly classified, then the accuracy shall be high even though the minority classes are wrongly classified. Therefore, accuracy is a less suitable metric for evaluating minority classes. Complementing accuracy with True Positive Rate (TPR) to examine learning algorithms' performance is a better option. It is because TPR can examine the detection performance for every single data set class.

CHAPTER 3

METHODOLOGY

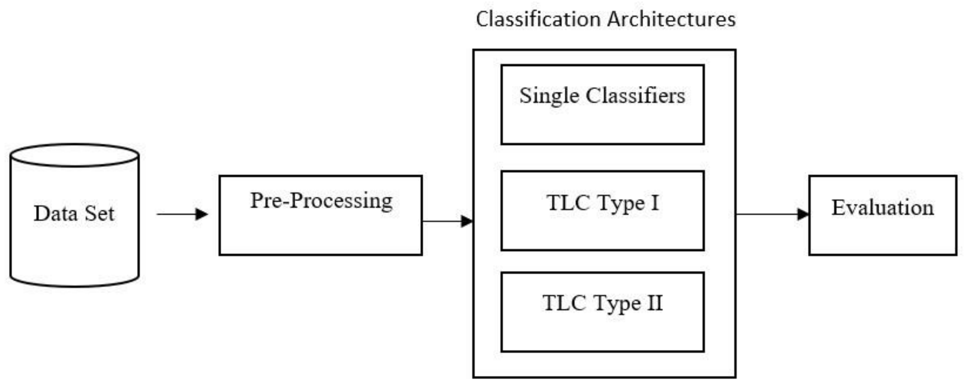

Figure 3.1 Overview of the methodology of this study

This study aims to increase the detection rate for network intrusions, especially intrusions that are small in size. Two data sets were involved: CICIDS2017 and UNSW-NB15. The methodology is shown in Figure 3. At the pre-processing stage, data cleaning was done by replacing the missing and infinity value with numerical values. Not only that, Z-score normalisation is done to standardise the data set. We evaluated three classification architectures: single classifiers, TLC type I and TLC type II. We used the true positive rate complement with accuracy to evaluate the performance of the classification architectures.

3.1 Combining Classes

Table 3.1 The class distribution of CICIDS 2017 before and after combining classes.

Original Class	Class distribution	Class after combined	Class distribution after combined
BENIGN	80.3000%	BENIGN	80.300%
Bot	0.0690%	Bot	0.069%
DDoS	4.5227%	DoS	13.449%
DoS GoldenEye	0.3636%		
DoS Hulk	8.1630%		
DoS Slowhttptest	0.1943%		
DoS slowloris	0.2048%		
Heartbleed	0.0004%		
Infiltration	0.0013%	Infiltration	0.001%
PortScan	5.6144%	PortScan	5.614%
FTP-Patator	0.2804%	Brute-Force	0.489%
SSH-Patator	0.2083%		
WA- Brute Force	0.0532%	Web Attack	0.077%
WA- Sql Injection	0.0007%		
WA- XSS	0.0230%		

Besides using the original CICIDS 2017 data set, we also created another one by combining attack classes to reduce the unbalanced class distribution problem. The CICIDS2017 data set is unbalanced, where BENIGN occupied 80.3% of the data set. Most intrusion classes occupied less than 1% of the data set. Therefore, classes with similar characteristics were combined, as shown in Table 3.1. Such combining can also be found in Panigrahi *et al.* (2018) study. By doing this, the small intrusion classes can be grouped into larger classes. For instance, by combining intrusions similar to DoS, a larger class that occupies 13.449% of the data set can be formed. Then, it is easier to detect this large class by the learning algorithm and thus, reduce the effect of the unbalanced class distribution problem. Classes in UNSW-NB15 were not combined. It is because the data set has ten distinct classes that are not possible for the combining purpose.

3.2 Pre-processing

Data pre-processing is a technique that transforms the raw data into an understandable data format by the machine. The issues for real-world data sets are usually incomplete and inconsistent data. Therefore, data pre-processing is often used to solve the issues.

The data set utilised in this study contains missing and infinity values. In this study, the missing values were replaced with the mean value of the data set features. Infinity values were replaced with ten times the maximum values of the feature to represent larger values for the learning algorithm to recognise.

$Z = \frac{x-\mu}{\sigma}$, where

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

Z-score normalisation is needed for data preprocessing because real-world data sets are usually inconsistent due to different units, ranges, or scales (Sahu *et al.*, 2020). When the differences in the scales across features of the data set occur, it may increase the learning algorithm's difficulty in forming the model. Therefore, we applied Z-score normalisation to standardise the data into a specific range [-1, 1]. After applying Z-score normalisation, the features were rescaled in the normal distribution form with a mean of zero and a standard deviation of one. The data set becomes consistent and more effective for the learning algorithm to learn the data after applying Z-score normalisation.

3.3 Classification Architectures

3.2.1 Single Classifiers

Five single classifiers, namely, Decision Tree – Information Gain (ID3), Decision Tree – Gini Index (CART), K-Nearest Neighbour(KNN), Neural Network – Multi-Layer Perceptron(MLP) and Gaussian Naïve Bayes (GNB), were evaluated. It is then proceeded with evaluating the proposed TLCs.

3.2.2 Two-level classification (TLC) type I

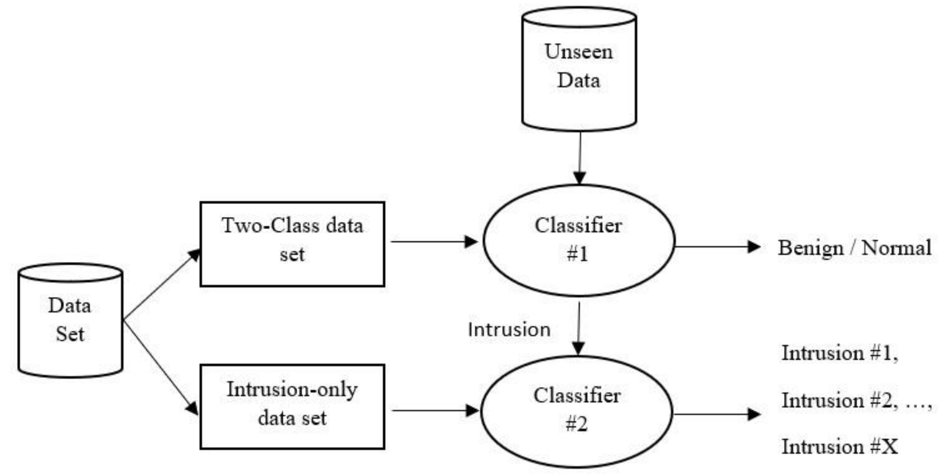

Figure 3.2 The architecture of the Two-level Classification Type I

Figure 3.2 shows the architecture of TLC type I. Firstly, all the intrusions in the data set were combined into a group; this will form a two-class data set containing only a normal/benign class and a general intrusion class. A new data set was also created by removing the normal class and leaving it with only intrusion classes. The two-class data set was loaded into the first-level classifier for training, and the intrusions-only data set was loaded into the second-level classifier for training. The classification starts from an unseen data set that was loaded into the first-level classifier, and then the instances classified as intrusions will pass to the second-level classifier. The second-level classifier classified the detected intrusions into specific intrusion classes.

Such a design mitigates the unbalanced class distribution and overlapping class problems. The first level requires two-class data set where all intrusions are combined into one general intrusion class. The combination will reduce the overwhelming effect from the normal/benign class (majority class) on the intrusions classes. The combination will also reduce the overlapping classes problem because the characteristics of certain intrusions classes are about the same.

3.2.3 Two-level classification (TLC) type II

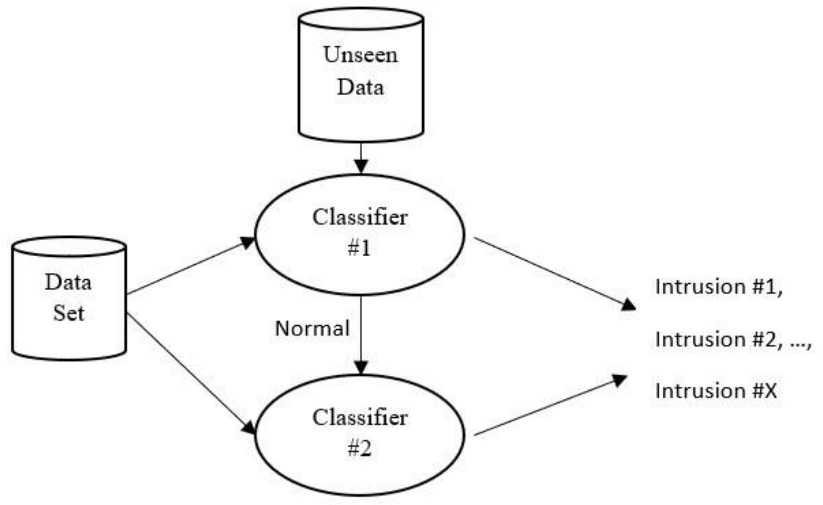

Figure 3.3 The architecture of the Two-level Classification Type II

Figure 3.3 shows the architecture of TLC type II. The original data set was loaded into the first and second-level classifiers for training. The classification starts with an unseen data set being loaded into the first-level classifier that is good at detecting normal/benign traffic. Then, the instances classified as normal/benign shall be passed to the second level. A different classifier is used at the second level, which helps detect intrusions (minority classes) that are misclassified as normal/benign at the first level. Therefore, the classifier at the second level should be good at detecting the minority intrusion classes.

The TLC type II tackles the majority and minority classes separately by having the strong classifier detecting normal/benign traffic at the first level and then placing the classifier that is good in detecting minority intrusion classes at the second level. Such arrangement will help mitigate the unbalanced class distribution and overlapping class problems. The type II architecture also mitigates the problem that intrusions (minority classes) are misclassified as normal/benign at the first level.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

The experiments were conducted using a workstation running Ubuntu 16.04 system with an Intel Core i9-7920X 2.90Hz processor and 64GB RAM. The data sets involved were UNSW-NB15 and CICIDS2017. There were two types of the CICIDS2017 data set: the original data set with 15 classes and the other data set rearranged into seven classes. Three architectures were tested: single classifiers, TLC type I and TLC type II. 70% of the respective data set was used as the training set and 30% as the testing set. It is because the sample size of some minorities was small. Therefore, the ratio of 70:30 gives the classifier enough sample, particularly minority intrusions, to train and test. 10-fold cross-validation was applied for the training set to optimise the parameters of the classifiers.

4.2 10-Fold Cross-Validation for Training Set

K-fold cross-validation is the technique to divide the data set into K partitions, and it is often used to assess a classification model’s accuracy and robustness against unseen data. 10-fold cross-validation was applied to the training set (70% of the data set) to optimise the parameters of the classifiers. The mean accuracy and the standard deviation of the cross-validation for the CICIDS2017 and UNSW-NB15 training sets are shown in Tables 4.1 - 4.3.

Table 4.1 Result of 10-fold cross-validation for the training set of the UNSW-NB15 data set.

Classifiers	Means accuracy	Standard deviation
CART	0.9815	0.00026
ID3	0.9809	0.00032
KNN	0.9758	0.00025
MLP	0.9822	0.00048
GNB	0.8202	0.00432

Table 4.2 Result of 10-fold cross-validation for the training set of the CICIDS2017 data set (15 classes).

Classifiers	Means accuracy	Standard deviation
CART	0.9985	0.00011
ID3	0.9986	0.00011
KNN	0.9978	0.00010
MLP	0.9867	0.00077
GNB	0.2415	0.03028

Table 4.3 Result of 10-fold cross-validation for the training set of the CICIDS2017 data set (7classes).

Classifiers	Means accuracy	Standard deviation
CART	0.9989	0.00008
ID3	0.9989	0.00008
KNN	0.9982	0.00011
MLP	0.9864	0.00091
GNB	0.2756	0.00391

Tables 4.1 - 4.3 shows that most single classifiers have good accuracy with low standard deviation except GNB for the CICIDS2017 data sets.

4.3 Single Classifiers

Experiments were conducted using single classifiers, i.e., Decision Tree-information gain (ID3), Decision Tree-Gini Index (CART), Neural Network-Multi Layer Perceptron (MLP), Gaussian Naïve Bayes (GNB) and K-Nearest Neighbours (KNN). The TPR for each class and the overall accuracy of the data sets shall be tabulated and discussed.

4.3.1 Single Classifiers for UNSW-NB15

Table 4.4 Result of single classifiers for UNSW-NB15. The numbers in bold show the low TPR for certain classes.

	CART	**ID3**	**KNN**	**MLP**	**GNB**
Normal	0.9978	0.9979	0.9968	0.9947	0.8390
Analysis	**0.0809**	**0.1096**	**0.0100**	**0.0137**	**0.1930**
Backdoors	**0.0744**	**0.0730**	**0.0072**	**0.0029**	**0.7525**
DoS	**0.3426**	**0.3691**	**0.2491**	**0.0361**	**0.0086**
Exploits	**0.7934**	**0.7665**	**0.7456**	0.9020	**0.1898**
Fuzzers	**0.5800**	**0.5124**	**0.4271**	**0.5157**	**0.2260**
Generic	0.9863	0.9865	0.9787	0.9833	0.9340
Reconnaissance	**0.7541**	**0.7619**	**0.6177**	**0.7645**	**0.1213**
Shellcode	**0.5673**	**0.5673**	**0.2163**	**0.3091**	0.9912
Worms	**0.4808**	**0.4615**	**0.0000**	**0.1154**	**0.3846**
Accuracy	0.9816	0.9808	0.9760	0.9777	0.8198

From Table 4.4, CART gave the best performance with an overall accuracy of 0.9816. Analysis, Backdoors, DoS, Exploits, Fuzzers, Reconnaissance, Shellcode and Worms generally showed below-average or low detection rates by most single classifiers. Such detection rates were due to the unbalanced class distribution of the data set as the learning algorithms tend to favour the majority class (Normal), which occupies 87.4% of the data set. Not only that, the overlapping class problem, as shown in Figures 2.5 and 2.6, contributed to the problem as all the classes were severely overlapped.

4.3.2 Single Classifiers for CICIDS2017

CICIDS2017 (15 classes)

Table 4.5 Result of single classifiers for the CICIDS2017 data set (15 classes). The numbers in bold show the low TPR for certain classes.

	CART	ID3	KNN	MLP	GNB
BENIGN	0.9991	0.9992	0.9986	0.9916	**0.0836**
Bot	0.8220	0.8034	**0.7153**	**0.3814**	**0.6356**
DDoS	0.9996	0.9997	0.9990	0.9990	0.9614
DoS GoldenEye	0.9948	0.9958	0.9929	0.9673	0.8935
DoS Hulk	0.9989	0.9989	0.9974	0.9421	**0.6664**
DoS Slowhttptest	0.9618	0.9630	0.9879	0.9879	**0.6297**
DoS Slowloris	0.9914	0.9937	0.9925	0.9885	**0.5566**
FTP-Patator	0.9966	0.9979	0.9966	0.9849	0.9954
Heartbleed	1.0000	1.0000	1.0000	1.0000	1.0000
Infiltration	0.8182	0.8182	**0.1818**	**0.5455**	0.8182
PortScan	0.9996	0.9997	0.9997	0.9984	0.9888
SSH-Patator	0.9949	0.9972	0.9796	0.4941	0.9893
WA - Brute Force	**0.7345**	**0.7677**	**0.7788**	**0.1018**	**0.1018**
WA - Sql Injection	**0.3333**	**0.3333**	**0.0000**	**0.0000**	1.0000
WA - XSS	**0.3929**	**0.4082**	**0.3265**	**0.0255**	0.9031
Accuracy	0.9986	0.9987	0.9980	0.9859	**0.2317**

From Table 4.5, ID3 gave the best performance with an accuracy of 0.9987. Bot, Infiltration, WA – brute force, WA-Sql injection and WA-XSS showed below-average or low detection rates by most single classifiers. The low detection rates were due to the unbalanced class distribution of the data set, as the learning algorithms tend to favour the majority class (BENIGN). Not only that, the overlapping class problem, as shown in Figures 2.2 and 2.3, contributed to the problem as all the classes were severely overlapped.

CICIDS2017 (7 classes)

Table 4.6 Result of single classifiers for the CICIDS2017 data set (7 classes). The numbers in bold show the low TPR for certain classes.

	CART	**ID3**	**KNN**	**MLP**	**GNB**
BENIGN	0.9991	0.9992	0.9986	0.9903	**0.1087**
Bot	0.8288	**0.7949**	**0.7153**	**0.3695**	**0.6373**
Brute-Force	0.9973	0.9981	0.9899	**0.7735**	**0.6145**
DoS	0.9985	0.9987	0.9978	0.9754	**0.6914**
Infiltration	0.9091	**0.7273**	**0.3636**	**0.1818**	1.0000
PortScan	0.9997	0.9997	0.9997	0.9953	0.9889
Web Attack	0.9679	0.9740	0.9281	**0.0734**	0.9098
Accuracy	0.9989	0.9990	0.9983	0.9864	**0.2399**

As shown in Table 4.6, ID3 gave the best accuracy of 0.9990. The overall accuracy slightly improved compared to the 15-class data set. Good TPRs were also obtained after rearranging the classes into bigger groups, except for two intrusion classes, Bot and Infiltration. Such improvement was due to the grouping of the minority classes into bigger groups, thus mitigating the unbalanced class and overlapping problems.

4.3.3 Selecting Single Classifiers for the Proposed TLCs

Experiments were conducted using five single classifiers, i.e., ID3, CART, MLP, GNB and KNN, on the data sets. The single classifiers were then selected based on their performance to form the proposed TLCs.

For both proposed TLCs, Classifier #1 should be good at detecting the majority class (Normal or BENIGN). On the other hand, Classifier #2 should be good in detecting the minority classes (Intrusions). Such an arrangement mitigates the effect of the unbalanced class distribution from the majority class.

Three top-performing single classifiers were selected for each data set to form the proposed TLCs. Table 4.4 shows that ID3, CART, and MLP gave the highest TPRs in detecting the majority class (Normal) for the UNSW-NB15 data set. On the other hand, Tables 4.5 and 4.6 show that ID3, CART, and KNN gave the highest TPRs in detecting the majority class (Normal) for the CICIDS2017 data sets (15 classes and seven classes). Therefore they were qualified as Classifier #1. The selected six classifiers do not have much difference in TPR for many of the minority intrusion classes. Therefore, they can be used as Classifier #2 as well.

4.4 Two-Level Classification (TLC) Type I

The experiments were then conducted using the proposed two-level classification Type I. All intrusions were grouped to form a general intrusion class at the first level, forming a binary classifier that detected only “Normal” and “Intrusions”. The binary classifier then passed the samples classified as intrusions to the second-level classifier for further classification. The second level classifier classified the detected intrusions into specific intrusion classes. Such a design was attempted to mitigate the unbalanced class distribution and overlapping class problems.

4.4.1 TLC Type I for UNSW-NB15

Table 4.7 Result of TLC type I for the UNSW-NB15 data set. The numbers in bold show the low TPR for certain classes.

	CART + ID3	CART + MLP	ID3 + CART	ID3 + MLP	MLP + CART	MLP + ID3
Normal	0.9973	0.9971	0.9970	0.9974	0.9910	0.9910
Analysis	**0.1096**	**0.1108**	**0.1096**	**0.0772**	**0.1196**	**0.1370**
Backdoors	**0.0730**	**0.0043**	**0.0801**	**0.0043**	**0.0801**	**0.0773**
DoS	**0.3746**	**0.0216**	**0.3282**	**0.0210**	**0.3276**	**0.3759**
Exploits	**0.7584**	0.8929	**0.7635**	0.8941	**0.7713**	**0.7666**
Fuzzers	**0.5935**	**0.6079**	**0.5554**	**0.5859**	**0.6191**	**0.6195**
Generic	0.9870	0.9815	0.9861	0.9816	0.9863	0.9871
Reconnaissance	**0.7610**	**0.7798**	**0.7560**	**0.7822**	**0.7464**	**0.7471**
Shellcode	**0.5453**	**0.4349**	**0.5806**	**0.4305**	**0.5872**	**0.5607**
Worms	**0.4808**	**0.0962**	**0.4231**	**0.0962**	**0.4231**	**0.4808**
Accuracy	0.9810	0.9805	0.9801	0.9806	0.9756	0.9759

The results of TLC Type I were obtained after applying it to UNSW-NB15. As shown in Table 4.7, the best performer was CART + ID3, which had an accuracy of 0.9810. There were eight classes with average and low detection rates (bold numbers) by all the combinations of TLC Type I: Analysis, Backdoors, DoS, Exploits, Fuzzers, Reconnaissance, Shellcode and Worms. In general, there is no improvement in using TLC type I when compared to the single classifiers (Refer to Table 4.4).

4.4.2 TLC Type I for CICIDS2017

Table 4.8 Result of TLC type I for the CICIDS2017 data set (15 classes). The numbers in bold show the low TPR for certain classes.

	CART + ID3	**CART + KNN**	**ID3 + CART**	**ID3 + KNN**	**KNN + CART**	**KNN + ID3**
BENIGN / Normal	0.9991	0.9991	0.9991	0.9991	0.9986	0.9986
Bot	0.8136	0.8136	**0.7780**	**0.7780**	**0.7169**	**0.7169**
DDoS	0.9996	0.9992	0.9997	0.9993	0.9994	0.9994
DoS GoldenEye	0.9958	0.9961	0.9958	0.9968	0.9919	0.9926
DoS Hulk	0.9987	0.9988	0.9988	0.9989	0.9974	0.9974
DoS Slowhttptest	0.9485	0.9497	0.9612	0.9642	0.9855	0.9873
DoS slowloris	0.9902	0.9908	0.9937	0.9942	0.9919	0.9914
FTP-Patator	0.9966	0.9958	0.9975	0.9958	0.9971	0.9962
Heartbleed	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Infiltration	0.8182	**0.5455**	**0.5455**	**0.5455**	**0.1818**	**0.1818**
PortScan	0.9996	0.9997	0.9996	0.9997	0.9997	0.9997
SSH-Patator	0.9949	0.9932	0.9960	0.9943	0.9796	0.9796
WA - Brute Force	**0.7699**	**0.7920**	**0.7699**	**0.7920**	**0.7212**	**0.7212**
WA - Sql Injection	**0.5000**	**0.1667**	**0.3333**	**0.1667**	**0.0000**	**0.1667**
WA - XSS	**0.3776**	**0.3622**	**0.3265**	**0.3316**	**0.3214**	**0.3418**
Accuracy	0.9986	0.9986	0.9986	0.9986	0.9980	0.9980

The results of TLC Type I were collected after applying it to the CICIDS 2017 data set (15 classes). As shown in Table 4.8, the best performer was CART

+ ID3, which had an accuracy of 0.9986 and only three minority classes with TPR below average. There were three classes with average and low detection rates (bold numbers) by all classifiers: WA-Brute Force and WA-Sql Injection. There is no improvement in TLC type I compared to the single classifiers (Refer to Table 4.5).

Table 4.9 Result of TLC type I for the CICIDS2017 data set (7 classes). The numbers in bold show the low TPR for certain classes.

	CART + ID3	**CART + KNN**	**ID3 + CART**	**ID3 + KNN**	**KNN + CART**	**KNN + ID3**
BENIGN /Normal	0.9991	0.9991	0.9991	0.9991	0.9986	0.9986
Bot	0.8136	0.8119	0.8186	0.8169	**0.7153**	**0.7153**
Brute-Force	0.9973	0.9957	0.9976	0.9959	0.9901	0.9901
DoS	0.9985	0.9985	0.9987	0.9988	0.9977	0.9978
Infiltration	0.8182	**0.6364**	0.9091	**0.7273**	**0.3636**	**0.3636**
PortScan	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
Web Attack	0.9618	0.9388	0.9771	0.9434	0.9327	0.9235
Accuracy	0.9989	0.9988	0.9989	0.9989	0.9983	0.9983

Results of TLC Type I applied to the CICIDS 2017 (7 classes) were also collected, and they are shown in Table 4.9. The best performers were ID3 + CART, with an accuracy of 0.9989. There is also no improvement using the CICIDS 2017 data set (7 classes) when comparing the two-level classification type I to single classifiers overall and each intrusion class (Table 4.6).

4.5 Two-Level Classification (TLC) Type II

The experiments were continued with TLC Type II. The first level classifier shall detect the majority class – Benign/Normal very well, and the samples classified as normal/benign are passed to the second level classifier for further classification. Referring to Table 4.4, CART, ID3 and MLP performed well on the majority and some minority classes for UNSW-NB15. Tables 4.5 and 4.6, on the other hand, showed that CART, ID3 and KNN performed well on the majority and some minority classes for CICIDS2017. Combining these single classifiers was thus attempted for TLC Type II.

4.5.1 TLC Type II for UNSW-NB15

Table 4.10 Result of TLC type II for UNSW-NB15. The numbers in bold show the low TPR for certain classes.

	CART + ID3	**CART + MLP**	**ID3 + CART**	**ID3 + MLP**	**MLP + CART**	**MLP + ID3**
Normal	0.9966	0.9941	0.9966	0.9941	0.9941	0.9940
Analysis	**0.0934**	**0.0809**	**0.0959**	**0.1133**	**0.0710**	**0.0822**
Backdoors	**0.0887**	**0.0744**	**0.0987**	**0.0730**	**0.0114**	**0.0114**
DoS	**0.3245**	**0.3430**	**0.3806**	**0.3693**	**0.0389**	**0.0397**
Exploits	0.8184	0.8008	**0.7691**	**0.7717**	0.9118	0.9122
Fuzzers	**0.6584**	**0.6773**	**0.6593**	**0.6439**	**0.6432**	**0.6166**
Generic	0.9850	0.9864	0.9859	0.9866	0.9835	0.9836
Reconnaissance	**0.7829**	**0.7574**	**0.7698**	**0.7636**	**0.7712**	**0.7714**
Shellcode	**0.6799**	**0.5850**	**0.5894**	**0.5717**	**0.3201**	**0.3311**
Worms	**0.5192**	**0.4808**	**0.4808**	**0.4615**	**0.1346**	**0.1154**
Accuracy	0.9817	0.9795	0.9812	0.9788	0.9787	0.9784

Table 4.10 shows that when applying to the UNSW-NB15 data set, TLC type II showed a slight overall improvement in accuracy compared to the single classifiers (refer to Table 4.4) and better performance than TCL Type I (refer to Table 4.7). The best performer was CART + ID3, with an accuracy of 0.9817. There is improvement in detecting intrusions. However, certain intrusions classes still suffered from low TRP.

4.5.2 TLC Type II for CICIDS2017

Table 4.11 Result of TLC type II for the CICIDS2017 data set (15 classes). The numbers in bold show the low TPR for certain classes.

	CART + ID3	**CART + KNN**	**ID3 + CART**	**ID3 + KNN**	**KNN + CART**	**KNN + ID3**
BENIGN / Normal	0.9990	0.9984	0.9990	0.9985	0.9984	0.9985
Bot	0.8661	0.8593	0.8661	0.8475	0.8593	0.8475
DDoS	0.9997	0.9997	0.9997	0.9997	0.9993	0.9994
DoS GoldenEye	0.9964	0.9961	0.9968	0.9971	0.9958	0.9968
DoS Hulk	0.9990	0.9993	0.9990	0.9993	0.9993	0.9993
DoS Slowhttptest	0.9739	0.9903	0.9739	0.9927	0.9909	0.9927
DoS slowloris	0.9931	0.9931	0.9937	0.9948	0.9925	0.9937
FTP-Patator	0.9983	0.9979	0.9983	0.9992	0.9979	0.9992
Heartbleed	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Infiltration	0.8182	0.8182	0.8182	0.8182	0.8182	0.8182
PortScan	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997
SSH-Patator	0.9972	0.9966	0.9972	0.9972	0.9966	0.9972
WA - Brute Force	**0.7345**	**0.7345**	**0.7721**	**0.7677**	**0.8296**	**0.8274**
WA - Sql Injection	**0.3333**	**0.3333**	**0.3333**	**0.3333**	**0.3333**	**0.3333**
WA - XSS	**0.3980**	**0.3980**	**0.4184**	**0.4082**	**0.3673**	**0.3571**
Accuracy	0.9986	0.9982	0.9986	0.9983	0.9982	0.9983

Table 4.11 shows that applying TLC Type II to CICIDS 2017 (15 classes) showed no improvement to single classifiers and TLC Type I (Refer to Table 4.5 and Table 4.8). The detection rate TPR for minority classes (WA – brute force, WA - Sql Injection and WA-XSS) remained average or low.

Table 4.12 Result of TLC type II for the CICIDS2017 data set (7 classes). The numbers in bold show the low TPR for certain classes.

	CART + ID3	**CART + KNN**	**ID3 + CART**	**ID3 + KNN**	**KNN + CART**	**KNN + ID3**
BENIGN / Normal	0.9990	0.9984	0.9990	0.9985	0.9984	0.9985
Bot	0.8627	0.8712	0.8627	0.8458	0.8712	0.8458
Brute-Force	0.9986	0.9981	0.9986	0.9986	0.9978	0.9983
DoS	0.9990	0.9993	0.9990	0.9993	0.9993	0.9993
Infiltration	0.9091	0.9091	0.9091	**0.7273**	0.9091	**0.7273**
PortScan	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
Web Attack	0.9755	0.9771	0.9771	0.9771	0.9740	0.9725
Accuracy	0.9989	0.9985	0.9990	0.9985	0.9985	0.9985

As shown in Table 4.12, improvement was observed applying TLC Type II to the CICIDS 2017 data set (7 classes) compared to single classifiers and TLC Type I (refer to Table 4.6 and Table 4.9). It shows that all classes obtained detection rates of TPR higher than 0.8. The best performer was CART + ID3, with an accuracy of 0.9990. The TPR of all minority classes had been improved to 0.8 and above.

4.6 Comparing with the Prior Works

4.6.1 UNSW-NB15 Data Set

Table 4.13. Comparing the proposed approaches and the prior works using the UNSW-NB15 data set. The bold numbers show the best detection rate for the classes in the data set.

	Our study (TLC Type I-CART + ID3)	**Our study (TLC Type II -CART +ID3)**	**Papamartzivanos *et al.* (2018) Serial MCS (GA + DT)**	**Ren *et al.* (2019) (Parallel MCS-Random forest)**
Normal	0.9973	0.9966	0.9739	0.967
Analysis	**0.1096**	0.0934	0.6442	0.381
Backdoors	0.0730	0.0887	**0.2045**	0.061
DoS	0.3746	0.3426	**0.6732**	0.403
Exploits	0.7584	**0.8184**	0.1429	0.461
Fuzzers	0.5935	0.6584	**0.7622**	0.663
Generic	**0.9870**	0.9863	0.8137	0.969
Reconnaissance	0.7610	0.7829	0.4607	**0.820**
Shellcode	0.5453	0.6799	0.3639	**0.780**
Worms	0.4808	0.5192	0.1818	**0.795**
Accuracy	0.9810	**0.9817**	0.8433	0.928

The results of the proposed approaches are compared with the best prior work utilising MCS with serial and parallel architectures from Table 2.8. There is no comparison with the prior work in Table 2.8 that used the hybrid architecture as the work does not show the detection rate for single classes. As shown in Table 4.13, TLC Type II (CART + ID3) outperforms the other researchers' work by obtaining an accuracy of 0.9817. By comparing the detection rate of the minority classes between our work and the prior works, our work improved the detection rate for the minority classes. Nevertheless, there is room for improvement to improve the detection rates for the minority classes.

4.5.2 CICIDS2017 Data Set

Table 4.14. Comparing our study and the prior work on the CICIDS2017 data set (15 classes). The bold numbers show the best detection rate for the classes in the data set.

	Our Study (TLC Type I -CART + ID3)	**Our Study (TLC Type II - ID3 + CART)**	**Ferrag et al. (2020) Parallel MCS (RF)**	**Ferrag et al. (2020) Hybrid MCS (REP Tree, Jrip, Random Forest)**	**Azzaoui et al. (2020) Hybrid MCS (RF + KNN)**
BENIGN	0.9991	0.9990	0.9812	0.9886	-
Bot	0.8136	**0.8661**	0.4968	0.4647	-
DDoS	0.9996	**0.9997**	0.9982	0.9988	0.9996
DoS GoldenEye	0.9958	0.9968	0.6757	0.6757	
DoS Hulk	0.9987	**0.9990**	0.9516	0.9678	**0.9990**
DoS Slowhttptest	0.9485	0.9739	0.8135	0.9384	
DoS Slowloris	0.9902	0.9937	0.9376	0.9776	
FTP-Patator	0.9966	**0.9983**	0.9973	0.9963	-
Heartbleed	**1.0000**	**1.0000**	**1.0000**	**1.0000**	-
Infiltration	0.8182	0.8182	0.8333	**1.0000**	0.7916
PortScan	0.9996	**0.9997**	0.9988	0.9988	0.9995
SSH-Patator	0.9949	0.9972	0.9982	**0.9991**	-
WA - Brute Force	0.7699	0.7721	0.7041	0.7327	**0.9990**
WA - Sql Injection	0.5000	0.3333	**1.0000**	0.5000	0.4625
WA - XSS	0.3776	0.4184	0.3750	0.3063	**0.9622**

Accuracy	**0.9986**	**0.9986**	0.9559	0.9667	0.9921

Table 4.14 compares the proposed approaches and the prior works on the CICIDS2017 data set (15 classes). The proposed approaches are compared with the best prior work utilising MCS with parallel and hybrid architectures from Table 2.9. There is no comparison with the prior work in Table 2.9 that used the serial architecture as the work does not show the detection rate for single classes. The TLC Type II (ID3 + CART) outperformed with an overall accuracy of 0.9986. Compared to the prior works, TLC Type II (ID3 + CART) improved the detection rate (TPR) for many minority classes.

Table 4.15 Comparing our study and the prior work on the CICIDS2017 data set (7 classes). The bold numbers show the best detection rate for the classes in the data set.

	Our Study (TLC I - ID3 +CART)	**Our Study (TLC II - ID3 + CART)**	**Gupta et al. (2022)**		
			XGBoost	**RF**	**CSE-IDS**
BENIGN / Normal	0.9991	0.9990	0.9800	**1.0000**	0.8600
Bot	0.8186	**0.8627**	0.1000	0.0100	0.8300
Brute-Force	0.9976	**0.9986**	-	-	-
DoS	0.9987	**0.9990**	0.3500	0.0500	0.9800
Infiltration	0.9091	0.9091	**0.9400**	0.8100	0.5000
PortScan	**0.9997**	**0.9997**	0.9900	0.0000	0.9900
Web Attack	**0.9771**	**0.9771**	0.0000	0.0000	0.8100
Accuracy	0.9989	**0.9990**	0.7600	0.5700	0.9200

A good performance was also observed using TLC Type II on the CICIDS2017 data set (7 classes). Very few researchers used this variant of the CICIDS data set. The result obtained in this study is compared with Gupta et al. (2022) (Table 4.15). The TCL Type II (ID3 + CART) outperformed the researchers' MCS combinations, including XGBoost, Random Forest and CSE-IDS - a layered approach built on cost-sensitive deep learning and ensemble algorithms (Gupta et al., 2022), in overall accuracy and TPR for five minority intrusion classes.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

The problems of network intrusion data were unbalanced class distribution and overlapping classes. These problems have caused classifiers to detect the minority intrusion classes ineffectively as classifiers normally favour majority classes (usually the normal network traffic).

We proposed two different arrangements for the multiple classifiers systems to mitigate the two problems: TLC Type I and Type II. Both TLCs were evaluated using CICIDS2017 and UNSW-NB15 data sets. Besides using the original CICIDS2017 data set, we combined the data set classes to mitigate the two problems further.

The proposed TLC Type II has successfully improved the overall detection rate and detection for the minority intrusion classes. The best overall accuracy obtained using the UNSW-NB15, and CICIDS2017 data sets was 98.17% and 99.90% using the TLC combination of CART + ID3 and ID3 + CART, respectively.

There is still room for improvement in detecting the minority intrusion classes, particularly the UNSW-NB15 data set. Other architectures of MCS are considered in future work to improve the detection rates for the minority intrusion classes.

REFERENCES

Abdelmoumin, G. *et al.* (2022) 'A Survey on Data-Driven Learning for Intelligent Network Intrusion Detection Systems', *Electronics (Switzerland)*, 11(2), pp. 1–22. doi: 10.3390/electronics11020213.

Ahmad, M. *et al.* (2021) 'Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set', *Eurasip Journal on Wireless Communications and Networking*. Springer International Publishing, 2021(1). doi: 10.1186/s13638-021-01893-8.

Ahmad, T. and Aziz, M. N. (2019) 'Data preprocessing and feature selection for machine learning intrusion detection systems', *ICIC Express Letter*, 13(2), pp. 93–101.

Alrowaily, M., Alenezi, F. and Lu, Z. (2019) *Effectiveness of Machine Learning Based Intrusion Detection Systems*, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer International Publishing. doi: 10.1007/978-3-030-24907-6_21.

Azzaoui, H. and Boukhamla, A. (2020) 'Two-Stages Intrusion Detection System Based On Hybrid Methods', in *Proceedings of the 10th International Conference on Information Systems and Technologies*, pp. 1–7.

Bagui, Sikha *et al.* (2019) 'Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset', *Security and Privacy*, 2(6), pp. 1–13. doi: 10.1002/spy2.91.

Berrar, D. (2018) 'Bayes' theorem and naive bayes classifier', *Encyclopedia of*

Bioinformatics and Computational Biology: ABC of Bioinformatics, 1–3, pp. 403–412. doi: 10.1016/B978-0-12-809633-8.20473-1.

Breiman, L. *et al.* (2017) *Classification and regression trees*, *Classification and Regression Trees*. doi: 10.1201/9781315139470.

Chiba, Z. *et al.* (2018) 'A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection', *Computers and Security*. Elsevier Ltd, 75, pp. 36–58. doi: 10.1016/j.cose.2018.01.023.

Choraś, M. and Pawlicki, M. (2021) 'Intrusion detection approach based on optimised artificial neural network', *Neurocomputing*, 452, pp. 705–715. doi: 10.1016/j.neucom.2020.07.138.

Cover, T. M. and Hart, P. E. (1967) 'Nearest Neighbor Pattern Classification', *IEEE Transactions on Information Theory*. doi: 10.1109/TIT.1967.1053964.

Das, A., Pramod and Sunitha, B. S. (2022) 'Anomaly-based Network Intrusion Detection using Ensemble Machine Learning Approach', *International Journal of Advanced Computer Science and Applications*, 13(2), pp. 635–645. doi: 10.14569/IJACSA.2022.0130275.

Dias, L. P. *et al.* (2017) 'Using artificial neural network in intrusion detection systems to computer networks', *2017 9th Computer Science and Electronic Engineering Conference, CEEC 2017 - Proceedings*, pp. 145–150. doi: 10.1109/CEEC.2017.8101615.

Dixit, M. *et al.* (2019) 'Internet traffic detection using naïve bayes and K-Nearest neighbors (KNN) algorithm', *2019 International Conference on*

Intelligent Computing and Control Systems, ICCS 2019. IEEE, (Iciccs), pp. 1153–1157. doi: 10.1109/ICCS45141.2019.9065655.

Ferrag, M. A. *et al.* (2020) ‘RDTIDS: Rules and decision tree-based intrusion detection system for internet-of-things networks’, *Future Internet*, 12(3), pp. 1–14. doi: 10.3390/fi12030044.

Firman, M. *et al.* (2018) ‘Illiteracy Classification Using K Means-Naïve Bayes Algorithm’, 2, pp. 153–158.

Freund, Y., Schapire, R. and Abe, N. (1999) ‘A short introduction to boosting’, *Journal-Japanese Society For Artificial Intelligence*. JAPANESE SOC ARTIFICIAL INTELL, 14(771–780), p. 1612.

G. Weiss (2004) ‘Mining with rarity: A unifying framework.’, *SIGKDD Explorations*, 6(1), pp. 7–19. doi: 10.1145/1007730.1007734.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Adaptive Computation and Machine learning*, *Adaptive Computation and Machine Learning Series*. doi: 10.1016/B978-0-12-391420-0.09987-X.

Gümüşbaş, D. *et al.* (2021) ‘A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems’, *IEEE Systems Journal*, 15(2), pp. 1717–1731. doi: 10.1109/JSYST.2020.2992966.

Gupta, N., Jindal, V. and Bedi, P. (2022) ‘CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems’, *Computers and Security*. Elsevier Ltd, 112, p. 102499. doi: 10.1016/j.cose.2021.102499.

Halimaa, A. and Sundarakantham, K. (2019) ‘Machine learning based intrusion detection system', in *2019 3rd International conference on trends in electronics and informatics (ICOEI)*, pp. 916–920.

Hornik, K., Stinchcombe, M. and White, H. (1989) 'Multilayer feedforward networks are universal approximators', *Neural Networks*. doi: 10.1016/0893-6080(89)90020-8.

Jyothsna, V. and Prasad, V. V. R. (2011) 'A Review of Anomaly based IntrusionDetection Systems', 28(7), pp. 26–35.

Karoriya, M. T. S. B. and Gogate, G. (2022) 'A Review of Intrusion Detection Systems'.

Keller, J. M. and Gray, M. R. (1985) 'A Fuzzy K-Nearest Neighbor Algorithm', *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(4), pp. 580–585. doi: 10.1109/TSMC.1985.6313426.

Khammassi, C. and Krichen, S. (2017) 'A GA-LR wrapper approach for feature selection in network intrusion detection', *Computers and Security*. Elsevier Ltd, 70, pp. 255–277. doi: 10.1016/j.cose.2017.06.005.

Kim, E., Kim, W. and Lee, Y. (2002) 'Combination of multiple classifiers for the customer ' s purchase behavior prediction', 34, pp. 167–175.

Krishna, K. V., Swathi, K. and Rao, B. B. (2020) 'A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset', (January). doi: 10.35940/ijrte.E6580.018520.

Kumar, V. (2012) 'Signature Based Intrusion Detection System Using SNORT', *International Journal of Computer Applications & Information Technology - IJCAIT*.

Kumar, Vinay and Kumar, Vinod (2013) ‘Intrusion Detection using Data Mining Techniques : A Study through Different Approach’, *International Journal of Applied Science & Technology Research Excellence*, 3(4).

Kurniabudi *et al.* (2020) ‘CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection’, *IEEE Access*, 8, pp. 132911–132921. doi: 10.1109/access.2020.3009843.

Lappas, T. and Pelechrinis, K. (2007) ‘Data Mining Techniques for (Network) Intrusion Detection Systems’, *Department of Computer Science and Engineering UC Riverside, Riverside CA*, 92521.

Liao, H. *et al.* (2013) ‘Intrusion detection system : A comprehensive review’, *Journal of Network and Computer Applications*. Elsevier, 36(1), pp. 16–24. doi: 10.1016/j.jnca.2012.09.004.

Liao, H. J. *et al.* (2013) ‘Intrusion detection system: A comprehensive review’, *Journal of Network and Computer Applications*. doi: 10.1016/j.jnca.2012.09.004.

Mohandes, M., Deriche, M. and Aliyu, S. (2018) ‘Classifiers Combination Techniques : A Comprehensive Review’, 3536(c), pp. 1–14. doi: 10.1109/ACCESS.2018.2813079.

Moustafa, N. and Slay, J. (2015) ‘UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)’, *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, (November). doi: 10.1109/MilCIS.2015.7348942.

Moustafa, N. and Slay, J. (2016) ‘The evaluation of Network Anomaly

Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set', *Information Security Journal*, 25(1–3), pp. 18–31. doi: 10.1080/19393555.2015.1125974.

Moustafa, N. and Slay, J. (2017) 'The significant features of the UNSW-NB15 and the KDD99 data sets for Network Intrusion Detection Systems', *Proceedings - 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2015*. IEEE, pp. 25–31. doi: 10.1109/BADGERS.2015.14.

MyCERT : Incident Statistics - Malaysia Botnet Drones and Malware Infection 2020 (no date) *Mycert.org.my*.

Otoum, Y., Liu, D. and Nayak, A. (2022) 'DL-IDS: a deep learning–based intrusion detection framework for securing IoT', *Transactions on Emerging Telecommunications Technologies*, 33(3). doi: 10.1002/ett.3803.

Panigrahi, R. *et al.* (2021) 'A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets', *Mathematics*, 9(7). doi: 10.3390/math9070751.

Panigrahi, R. and Borah, S. (2018) 'A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems', *International Journal of Engineering and Technology(UAE)*, 7(3.24 Special Issue 24), pp. 479–482.

Papamartzivanos, D., Gómez Mármol, F. and Kambourakis, G. (2018) 'Dendron: Genetic trees driven rule induction for network intrusion detection systems', *Future Generation Computer Systems*. Elsevier B.V., 79, pp. 558–574. doi: 10.1016/j.future.2017.09.056.

Podolak, I. T. (2008) ‘Hierarchical classifier with overlapping class groups’, *Expert Systems with Applications*, 34(1), pp. 673–682. doi: 10.1016/j.eswa.2006.10.007.

Quinlan, J. R. (1986) ‘Induction of decision trees’, *Machine Learning*, 1(1), pp. 81–106. doi: 10.1007/bf00116251.

Rahman, A. F. R. and Fairhurst, M. C. (2003) ‘Multiple classifier decision combination strategies for character recognition: A review’, *International Journal on Document Analysis and Recognition*, 5(4), pp. 166–194. doi: 10.1007/s10032-002-0090-8.

Ren, J. *et al.* (2019) ‘Building an Effective Intrusion Detection System by Using Hybrid Data Optimization Based on Machine Learning Algorithms’, *Security and Communication Networks*, 2019. doi: 10.1155/2019/7130868.

Rosay, A., Carlier, F. and Leroux, P. (2020) ‘MLP4NIDS: An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset’, 1, pp. 240–254. doi: 10.1007/978-3-030-45778-5_16.

Saad Assiri, A. (2021) ‘Efficient Training of Multi-Layer Neural Networks to Achieve Faster Validation’, *Computer Systems Science and Engineering*, 36(3), pp. 435–450. doi: 10.32604/csse.2021.014894.

Saba, T. *et al.* (2022) ‘Anomaly-based intrusion detection system for IoT networks through deep learning model’, *Computers and Electrical Engineering*. Elsevier Ltd, 99(February), p. 107810. doi: 10.1016/j.compeleceng.2022.107810.

Sahu, A. *et al.* (2020) *Data Processing and Model Selection for Machine*

Learning-based Network Intrusion Detection; Data Processing and Model Selection for Machine Learning-based Network Intrusion Detection.

Salih, A. A. and Abdulazeez, A. M. (2021) 'Evaluation of classification algorithms for intrusion detection system: A review', *Journal of Soft Computing and Data Mining*, 2(1), pp. 31–40.

Samrin, R. (2017) 'Review on Anomaly based Network Intrusion Detection System', *Department of Computer Science and Engineering UC Riverside, Riverside CA*, 92521.

Samrin, R. and Vasumathi, D. (2018) 'Review on anomaly based network intrusion detection system', *International Conference on Electrical, Electronics, Communication Computer Technologies and Optimization Techniques, ICEECCOT 2017*, 2018-Janua, pp. 141–147. doi: 10.1109/ICEECCOT.2017.8284655.

Saranyaa, T. *et al.* (2020) 'Performance Analysis of Machine Learning Algorithms in Intrusion Detection System : A Review', *Procedia Computer Science*, 171(2020), pp. 1251–1260. doi: 10.1016/j.procs.2020.04.133.

Sarkar, M. and Leong, T. Y. (2000) 'Application of K-nearest neighbors algorithm on breast cancer diagnosis problem.', *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*, pp. 759–763.

Sharafaldin, I., Lashkari, A. H. and Ghorbani, A. A. (2018) 'Toward generating a new intrusion detection dataset and intrusion traffic characterization', *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018-Janua(Cic), pp. 108–116. doi: 10.5220/0006639801080116.

Sharma, Siddharth, Sharma, Simone and Anidhya, A. (2020) ‘Understanding Activation Functions in Neural Networks’, *International Journal of Engineering Applied Sciences and Technology*, 4(12), pp. 310–316.

Singh, S. (2014) ‘COMPARATIVE STUDY ID3 , CART AND C4 . 5 DECISION TREE ALGORITHM : A SURVEY’, 27(27), pp. 97–103.

Sivatha, S. S., Geetha, S. and Kannan, A. (2012) ‘Expert Systems with Applications Decision tree based light weight intrusion detection using a wrapper approach’, *Expert Systems With Applications*. Elsevier Ltd, 39(1), pp. 129–141. doi: 10.1016/j.eswa.2011.06.013.

Song, C. *et al.* (2021) ‘Intrusion detection based on hybrid classifiers for smart grid ☆’, *Computers and Electrical Engineering*. Elsevier Ltd, 93(May), p. 107212. doi: 10.1016/j.compeleceng.2021.107212.

Swami, R., Dave, M. and Ranga, V. (2020) ‘Voting-based intrusion detection framework for securing software-defined networks’, *Concurrency Computation* , 32(24), pp. 1–16. doi: 10.1002/cpe.5927.

Tama, B. A. *et al.* (2020) ‘An enhanced anomaly detection in web traffic using a stack of classifier ensemble’, *IEEE Access*, 8, pp. 24120–24134. doi: 10.1109/ACCESS.2020.2969428.

Thabtah, F. *et al.* (2020) ‘Data imbalance in classification: Experimental evaluation’, *Information Sciences*. Elsevier Inc., 513, pp. 429–441. doi: 10.1016/j.ins.2019.11.004.

Tharewal, S. *et al.* (2022) ‘Intrusion Detection System for Industrial Internet of Things Based on Deep Reinforcement Learning’, *Wireless Communications and*

Mobile Computing, 2022. doi: 10.1155/2022/9023719.

Tsai, C. F. *et al.* (2009) ‘Intrusion detection by machine learning: A review’, *Expert Systems with Applications*. Elsevier Ltd, 36(10), pp. 11994–12000. doi: 10.1016/j.eswa.2009.05.029.

Vijayanand, R., Devaraj, D. and Kannapiran, B. (2018) ‘Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection’, *Computers and Security*. Elsevier Ltd, 77, pp. 304–314. doi: 10.1016/j.cose.2018.04.010.

Wang, B. *et al.* (2020) ‘A deep hierarchical network for packet-level malicious traffic detection’, *IEEE Access*, 8, pp. 201728–201740. doi: 10.1109/ACCESS.2020.3035967.

Wang, W. *et al.* (2022) ‘Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions’, *Computers and Security*. Elsevier Ltd, 112, p. 102537. doi: 10.1016/j.cose.2021.102537.

Woźniak, M., Graña, M. and Corchado, E. (2014) ‘A survey of multiple classifier systems as hybrid systems’, *Information Fusion*, 16(1), pp. 3–17. doi: 10.1016/j.inffus.2013.04.006.

Xiang, C., Yong, P. C. and Meng, L. S. (2008) ‘Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees’, *Pattern Recognition Letters*, 29(7), pp. 918–924. doi: 10.1016/j.patrec.2008.01.008.

Yulianto, A., Sukarno, P. and Suwastika, N. A. (2019) ‘Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset', *Journal of Physics: Conference Series*, 1192(1). doi: 10.1088/1742-6596/1192/1/012018.

Zhang, Y. *et al.* (2019) 'Network Intrusion Detection : Based on Deep Hierarchical Network and Original Flow Data', *IEEE Access*. IEEE, 7, pp. 37004–37016. doi: 10.1109/ACCESS.2019.2905041.

Zoghi, Z. and Serpen, G. (2022) 'Ensemble Classifier Design Tuned to Dataset Characteristics for Network Intrusion Detection'.

List of Publication

(a) Ho, Y.B., Yap, W.S. and Khor, K.C. (2022). ‘Mitigating Unbalanced and Overlapping Class Problems of Large Network Intrusion Data Using a Two-Level Classifier Technique’. *Turkish Journal of Electrical Engineering & Computer Sciences.* (Submitted for journal consideration)

(b) Ho, Y. B., Yap, W. S. and Khor, K. C. (2021) ‘The Effect of Sampling Methods on the CICIDS2017 Network Intrusion Data Set’, *IT Convergence and Security*. Singapore: Springer Singapore, pp. 33–41. doi: 10.1007/978-981-16-4118-3_4.