

# A Convolutional Neural Network based Model with Improved Activation Function and Optimizer for Effective Intrusion Detection and Classification

by

Solaiman Kabir  
16301042

Sadman Sakib  
16101124

Md. Akib Hossain  
16301028

Safi Islam  
16341006

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering  
BRAC University  
April 2020

© 2020. BRAC University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



---

Solaiman Kabir  
16301042



---

Sadman Sakib  
16101124



---

Md. Akib Hossain  
16301028



---

Safi Islam  
16341006

# Approval

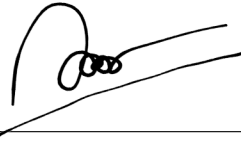
The thesis/project titled “A Convolutional Neural Network based Model with Improved Activation Function and Optimizer for Effective Intrusion Detection and Classification” submitted by

1. Solaiman Kabir (16301042)
2. Sadman Sakib (16101124)
3. Md. Akib Hossain (16301028)
4. Safi Islam (16341006)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 07, 2020.

## Examining Committee:

Supervisor:  
(Member)



---

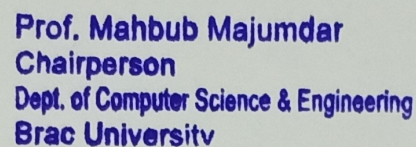
Dr. Muhammad Iqbal Hossain  
Assistant Professor  
Department of Computer Science and Engineering  
BRAC University

Program Coordinator:  
(Member)

---

Dr. Mahbub Alam Majumdar  
Professor and Chairperson  
Department of Computer Science and Engineering  
BRAC University

Head of Department:  
(Chair)



**Prof. Mahbub Majumdar**  
**Chairperson**  
**Dept. of Computer Science & Engineering**  
**Brac University**

---

Dr. Mahbub Alam Majumdar  
Professor and Chairperson  
Department of Computer Science and Engineering  
BRAC University

# Abstract

In today's world, technological advancements have entangled our financial, social and many more other aspects of lives to the internet or some network. Moreover, with the development of IoT technologies, it has spread over to our transportation, home-appliances and more devices. It is also a security risk because all of our sensitive and private knowledge on the Internet is exposed to a growing amount of cyber-attacks. An Intrusion Detection System can identify a cyber-attack while it is ongoing or prior to it. We are conscious of the evolving Machine Learning and Deep Learning developments, the most sophisticated multi-functional methods created by humans that can be utilized to overcome this issue. Alongside identification, precise classification of intrusion is of considerable significance for the administrator to take decisive actions. In this study, we have used the dataset CIC-IDS-2018 that is the biggest and most recent labeled dataset of intrusions. This dataset comprises of six varieties of attacks. Our thesis proposes a CNN Model with mish activation function and Ranger optimizer. The model reaches an accuracy of 0.989 that is the highest in multiclass classification with this dataset.

**Keywords:** Intrusion Detection System (IDS), Multiclass Classification, CNN, DNN, Machine Learning.

## **Acknowledgement**

We are very much thankful to Dr. Muhammad Iqbal Hossain for his precious ideas and continuous supervision which made us to complete our project. Our thanks and appreciations also go to people who have willingly helped us out with their abilities in this thesis

# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Acknowledgment</b>	<b>iv</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>Nomenclature</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Problem Statement . . . . .	3
1.3 Objectives and Contribution . . . . .	4
1.4 Thesis Structure . . . . .	4
<b>2 Related Work</b>	<b>5</b>
2.1 Background . . . . .	5
2.1.1 Intrusion Detection System . . . . .	5
2.1.2 Machine Learning and Deep Learning Algorithms . . . . .	7
2.1.2.1 Convolutional Neural Network . . . . .	7
2.1.2.2 Deep Neural Network . . . . .	8
2.1.2.3 Random Forest Classifier . . . . .	8
2.1.2.4 Decision Tree Classifier . . . . .	9
2.1.2.5 Extra Tree Classifier . . . . .	9
2.1.2.6 Bagging . . . . .	10
2.2 Literature Review . . . . .	11
<b>3 Dataset</b>	<b>13</b>
3.1 Dataset Description . . . . .	13
3.2 Type of Attacks . . . . .	15
3.3 Class Imbalance in Dataset . . . . .	16
3.4 Features in Dataset . . . . .	17

<b>4</b>	<b>Methodology</b>	<b>21</b>
4.1	Work Plan . . . . .	21
4.1.1	Dataset Preprocessing . . . . .	22
4.1.2	Feature Selection . . . . .	22
4.1.3	Split Dataset into Training and Test Set . . . . .	23
4.1.4	Synthetic Minority Oversampling Technique (SMOTE) . . . . .	23
4.1.5	Training the Classifier . . . . .	23
4.1.6	Evaluating the Performance . . . . .	23
<b>5</b>	<b>Implemented Algorithms</b>	<b>24</b>
5.1	Proposed CNN Model Structure . . . . .	24
5.1.1	Proposed CNN Model Parameters . . . . .	24
5.1.2	Activation function . . . . .	26
5.1.3	Optimizer . . . . .	27
5.2	DNN Architecture . . . . .	27
5.3	Random Forest Classifier Parameters . . . . .	27
5.4	Decision Tree Classifier Parameters . . . . .	27
5.5	Extra Tree Classifier Parameters . . . . .	28
5.6	Bagging Classifier Parameters . . . . .	28
<b>6</b>	<b>Evaluation</b>	<b>29</b>
6.1	Experimental Setup . . . . .	29
6.2	Results . . . . .	29
6.2.1	Result of CNN model . . . . .	30
6.2.2	Result of DNN model . . . . .	30
6.2.3	Result of Random Forest Classifier . . . . .	31
6.2.4	Result of Decision Tree Classifier . . . . .	31
6.2.5	Result of Extra Tree Classifier . . . . .	32
6.2.6	Result of Bagging Classifier . . . . .	32
6.3	Analysis . . . . .	33
<b>7</b>	<b>Conclusion</b>	<b>36</b>
	<b>References</b>	<b>39</b>

# List of Tables

3.1	Type of attacks in CSE-CIC-IDS2018 dataset. . . . .	15
3.2	Number of occurrences of each attack type in CSE-CIC-IDS2018 . . . .	16
6.1	Performance summary of CNN model . . . . .	30
6.2	Performance summary of DNN model . . . . .	30
6.3	Performance summary of Random Forest Classifier . . . . .	31
6.4	Performance summary of Decision Tree Classifier . . . . .	31
6.5	Performance summary of Extra Tree Classifier . . . . .	32
6.6	Performance summary of Bagging Classifier . . . . .	32
6.7	Summarized result of all trained model in the thesis . . . . .	33



# List of Figures

2.1	Basic design of IDS system . . . . .	6
2.2	Convolutional Neural Network Structures [13] . . . . .	7
2.3	Multilayer perceptron with two hidden layers [14] . . . . .	8
3.1	Heatmap of the dataset . . . . .	14
3.2	Percentage instance of classes in CSE-CIC-IDS2018 . . . . .	16
3.3	Features in CIC-IDS2018 dataset . . . . .	20
4.1	Work Plan for proposed model . . . . .	21
4.2	Zero variance features in CSE-CIC-IDS2018 dataset . . . . .	22
5.1	Structure of Convolution Layers in proposed CNN model . . . . .	25
5.2	Structure of Fully Connected Layers in proposed CNN model . . . . .	25
5.3	Mish Activation Function [30] . . . . .	26
5.4	Hyperparameters for the optimizer . . . . .	27
5.5	Parameters for Random Forest Classifier . . . . .	27
5.6	Parameters for Decision Tree Classifier . . . . .	27
5.7	Parameters for Extra Tree Classifier . . . . .	28
5.8	Parameters for Bagging Classifier . . . . .	28
6.1	Accuracy comparison of CNN and other models. . . . .	33
6.2	Precision comparison of CNN and other models. . . . .	34
6.3	Recall comparison of CNN and other models. . . . .	34
6.4	F1-score comparison of CNN and other models. . . . .	34

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*ANN* Artificial Neural Network

*CNN* Convolutional Neural Network

*DDoS* Distributed Denial of Service

*DNN* Deep Neural Network

*DoS* Denial of Service

*FN* False Negative

*FP* False Positive

$H(S)$  The Entropy of set S

*HIDS* Host Intrusion Detection System

*IDS* Intrusion Detection System

*IoT* Internet of Things

*LSTM* Long Short Term Memory

*LVQ* Learning vector quantization

*MLP* Multilayer Perceptron

*NIDS* Network Intrusion Detection System

*SMOTE* Synthetic Minority Oversampling Technique

*SVM* Support Vector Machine

$T$  the number of subset created from splitting set S by a given attribute A such that,

$$S = \cup_{t \in T} T$$

$TN$  True Negative

$TP$  True Positive

$H(t)$  Entropy of subset t

$P(t)$  The proportion of the number of elements in t to the number of elements in set S

# Chapter 1

## Introduction

Intrusion or cyber invasions are the technological counterpart to a robbery in real life that can be potentially more deleterious as the victim is unaware of the act. We are conscious of how valuable can information be in this age. An intruder is a person tries to gain unauthorized access to a network or personal computer to extract this invaluable information or incapacitate the system to interrupt regular activities. The target of the attacker may be an important person, organization or a regular person. Basically, it can be anyone who is available on the internet.

Exponential rise of Internet of Things (IoT) have allowed users to connect every aspect of their life to the internet and to be recorded digitally [1]. It integrated into our appliances (e.g. refrigerator, lights, security system) and lifestyle (e.g. smart watches, smartphones). As this information are deposited in a server, they are susceptible to any data theft or server breach. In March of 2017, One of the North America's biggest consumer credit reporting agency 'Equifax' had a data breach where hundreds of millions of people's private information were stolen through a breach in their server network [2]. The personal data is offered to prospective buyers for a significant amount. It is frightening to think about the minimal privacy we have with our data. Moreover, several cases of cyber-attacks have been carried out to take down a competing, steal sensitive information from country officials or hack into the military database. Infiltrating inside any of these networks would cause a huge crisis. This is why many developing countries are spending their considerable resources in their cybersecurity department. In March 2020, U.S. Health and Human Services suffered a DDoS attempt from cyber attackers to halt their routine network operation during COVID-19 pandemic [3].

An intrusion detection system has one primary objective that is to monitor and identify malicious activities from third party attacker. It is understood as a secondary stage of protection. The attackers can weaken the first stage such as firewalls using various utilities and mechanisms. Therefore, it is very crucial to have an IDS on the network for protect against unexpected threats. Intrusion detection system can be classified to two fundamental classes - signature-based and anomaly-based. Signature based IDS works by matching new threats with recorded threats for identification. On the other hand, Anomaly based IDS has been modeled after learning through patterns of intrusion or anomaly to identify unknown attacks. We can categorize intrusion detection can be based on network or host [4], [5].

## 1.1 Motivation

Tremendous amount of research has been conducted to develop and implement an effective Intrusion Detection mechanism. This concept of attack detection first appeared in 1980 intending to improve computer security and prevent unauthorized access to computer information [6]. Researchers are trying to provide new methods to combat with dangerous online invasions. This field is emerging due to the changing nature of Cyber-attacks and more consciousness among the users to protect themselves virtually. Online security seems to be fatal, to some, a priority over offline security. However, for independent researchers, it has been cumbersome due to lack of valid datasets since Intrusion Detection System is highly data dependent. Study of new types and signature of attacks are compulsory. We have found that CSE-CIC-IDS2018 [7] is the most recent dataset available in this field which is based upon network flow data. This dataset contains six types of intrusion labeled inside. Humanly it is not possible to study high dimensional statistical data and find patterns of an attack. Machine Learning and Deep Learning has come forward in last decade to ease the problem. These techniques allow researchers to extract, analyze and find pattern in data which would take years for an analyst to do manually. Moreover, We and our supervisor have our utmost faith that we could come forward with an efficient Deep Learning Model that would detect an attack and accurately classify.

## 1.2 Problem Statement

Cyber-attacks are the virtual equivalence of real-life crimes. It is threatening considering that we frequently are not conscious of an attack until it is too late. An intrusion can be seen as a burglary or invasion to our houses. There have been an increased number of network breaches and data theft in recent years. On February 4, 2018 Bangladesh Bank faced the one of the biggest security breaches popularly known as Bangladesh Bank cyber heist. The attackers made a breach to the Bangladesh Bank System altering the software that connects to SWIFT network and made 35 transaction requests from Bangladesh Bank SWIFT account at Federal Reserve Bank of New York to send one billion dollars to different bank account around the world [8]. However, they were only successful in transferring \$81 million dollars and rest of the transactions were denied. Still \$81 million for any country is a fatal damage and this robbery did not require a team of robbers or a heist team, the entire heist was carried out remotely. In This unfortunate event exhibits the severity of intrusions and cyber-attacks. Data theft is another major concern that have been emerging recently in large. In February 2019, 617 Million online identities were stolen through few different network attacks and they were put up for purchasing on Dark web which can be seen as the black market of internet. The database was enlisted on Dream Market, a dark web market place that provides drugs, weapons and other illegal commodities. These accounts range from fitness, photography to heritage services. This growth in personal data have seem to scare the customers and lose their trust over the services causing many of the businesses to lose their users. Therefore, it is indispensable for any network to identify an attack and notify the network administrator to take commendable actions to obstruct treacherous online invasions [9]. However, it has been noticed that most IDS models are only able to detect and not identify the type of an attack as a result of binary classifiers. There are not a lot of proposed models based on latest intrusion data that provide very high accuracy in terms of multiclass attack classification. It is crucial for system administrator to understand what type of attack the system is undergoing to act on it since cyber-attacks are tricky and time is influential on the damage. The higher time it takes to act on it, the bigger the damage usually is. Therefore, we cannot argue with the fact that classification of attacks is vital for Intrusion Detection System. This paper proposes a Deep Learning Model using that is efficiently, effectively able to achieve a very high accuracy in detecting and classifying the major attack types.

## 1.3 Objectives and Contribution

Our thesis focuses on applying Convolutional Neural Network (CNN) architecture in intrusion detection and classification. Some modifications and tweaking were made such as using mish activation function, Ranger optimizer to yield the finest performance. We have used an up to date dataset available from CIC (Canadian Institute of Cybersecurity) named CIC-IDS-2018 that is a successor to CIC-IDS-2017 dataset and resolved the flaws with the previous one. In our research, we have trained other previously used machine learning algorithms that performed well with this dataset such as Random Forest Classifier, Extra Tree Classifier, Decision Tree Classifier, Bagging Classifier to compare with our model. We also present a Deep Neural Network (DNN) architecture that performs better than the classical machine learning algorithms but cannot outperform the CNN model. We believe this model can be implemented in any networks with the help of correct framework to accurately identify any attacks in real-time.

## 1.4 Thesis Structure

Chapter 1: Introduction where Motivation, Problem Statement, Objectives and Contributions are discussed

Chapter 2: Related Work where background and literature review is discussed. The background has two more parts, these parts are divided into Intrusion Detection System where history of IDS has been discussed and Machine Learning and Deep Learning Algorithms where we provide detailed description about the implemented algorithms.

Chapter 3: Dataset where Dataset Description, Type of Attacks, Class Imbalance in Dataset and Features in Dataset are discussed

Chapter 4: Methodology where work plans are discussed. In the work plan section, we have splitted the plan into Dataset Pre-processing, Feature Selection, Split Dataset into Training and Test Set, SMOTE oversampling, Training the Classifier, Evaluating the Performance.

Chapter 5: Our Implemented algorithms where we have talked about DNN architecture, Random Forest Classifier Parameters, Extra Tree Classifier Parameters, Decision Tree Classifier Parameters, Bagging Classifier Parameters and our proposed CNN structure which would have also three parts and they are Proposed CNN Model Parameters, Activation Function and Optimizer.

Chapter 6: Evaluation includes Experimental Setup, Results where we show result of CNN Model along with the other models and after that we have analyzed the result.

Chapter 7: Conclusion, we have concluded the research till now and talked about our future plans.

# Chapter 2

## Related Work

This chapter has been divided into background and literature review.

### 2.1 Background

#### 2.1.1 Intrusion Detection System

Network base services have become a common thing in our lives because of huge level of using computers in all over the world. With increasing of the usage, different types of problems also started growing and for preventing these problems people started using Anti-Virus, Firewall, Intrusion Detection System (IDS) and many more tools [10]. Basically, system administrator would detect malicious activities by carefully monitoring user's actions. The following advancement in detecting intrusion included audit logs which were checked by system administrators for signs of suspicious movement. In late of 1970 and earlier in 1980, administrators usually printed audit reports on paper which frequently assembled in a shape of man's height after the end of the week. This manual procedure was not effective at all to stop attacks. The time when storage became easily available, audit logs migrated online. In spite of developing programs by researchers, analysis was slow. In early of 1990, real time intrusion detection began. This has allowed the identification of attacks and failed attacks as they occurred and that was needed to be solved by real time response. More recent efforts in intrusion prevention activities have focused on designing technologies that consumers of large networks can deploy effectively. It is not a simple task to consider the growing security issues, endless new types of attack and constant changes in the computer environment surrounding it [11].

Intrusion detection is divided into two forms based on their placement: 1) Network Intrusion detection 2) Host Intrusion Detection. Network intrusion Detection systems (NIDS) are ideally positioned to make a judgment about whether there has been an unauthorized or unusual interference in the network. This contributes to the system reminding the administrator of the attack. HIDS works inside a networked system by monitoring and analyzing their host's application and system. A HIDS just tracks the data that comes in and out of the system and notifies the administrator or client about any bizarre behavior. Intrusion can usually be identified by 1) Signature-based IDS and 2) Anomaly-based IDS, A signature-based IDS tracks all network packets and identifies threats by looking for unique patterns or identified malicious instructions. An anomaly-based IDS tracks network traffic and

correlates it to an defined standard to decide what the network considers natural in terms of latency, protocols, interfaces and other resources. Hybrid monitoring blends exploitation with the identification of abnormalities. It helps to raise the identification rate of documented intrusions and to lower the false positive rate of attacks which are not known. Based on reaction we can categorize IDS in active IDS and Passive IDS. Active IDS will take prompt action and notify the administrator of every threat. At the other side passive IDS stores intrusion log information and only notifies the user after that. The IDS can be broken down into Online IDS and offline IDS depending on frequency use. Offline IDS is used to evaluate pre-logged data to detect attacks where Online IDS is Offline IDS analyzes pre-logged data to detect intrusion while Online IDS utilizes fresh data to detect an assault [12].

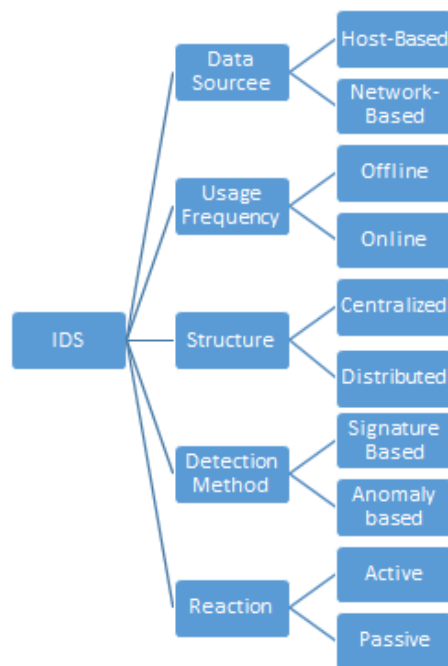


Figure 2.1: Basic design of IDS system



## 2.1.2 Machine Learning and Deep Learning Algorithms

### 2.1.2.1 Convolutional Neural Network

Convolutional Neural Network is a deep learning algorithm which is very effective in computer vision sector such as recognizing persons, digits and various types of classification. Convolutional Neural Network derived its name from Convolution operation that is used to extract information from predefined kernel or window. Small Matrix Numbers are taken in this process or also known as kernel and Move it over the image and convert it according to kernel values. In this process subsequent feature maps are calculated in following way:

$$G[m, n] = (f * h)[m, n] = f(x) = \sum_j \sum_k h[j, k] f[m - j, n - k] \quad (1)$$

In this case, the input image is  $f$  and the kernel is  $h$ . The matrix rows and column indexes are respectively  $m$  and  $n$ .

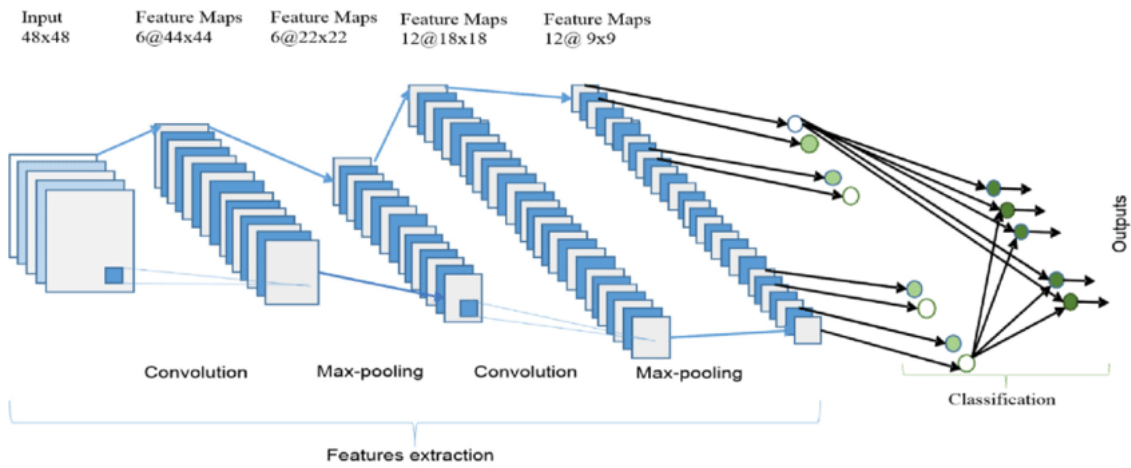


Figure 2.2: Convolutional Neural Network Structures [13]

### 2.1.2.2 Deep Neural Network

Neural networks are very much similar to neurons in human brain. The architecture of artificial neural network is that it consists of one input layer, one output layer, and there are one or more hidden layers in between of those. Neural network was first introduced in 1943 and it has been improving ever since. Figure 2.3 shows the configuration of a multilayer perceptron (MLP) with two hidden layers which is the feed forward neural networks architecture. The signals are propagated via an activation function from the input layer to the output layer. Outputs of a neuron are calculated as (2) where  $z$  is the output of a neuron determined by input  $x$  and weight  $w$ . So, it is the sum of multiplication of input and weight. This output then passes through an activation function. The input for next layer is the output from activation function (3). These weights are updated by the back propagation (BP) algorithm. A Deep Neural Network (DNN) is also a BP multilayer perceptron but with more than two hidden layers.

$$z = w^T x + b \quad (2)$$

$$y = \sigma(z) \quad (3)$$

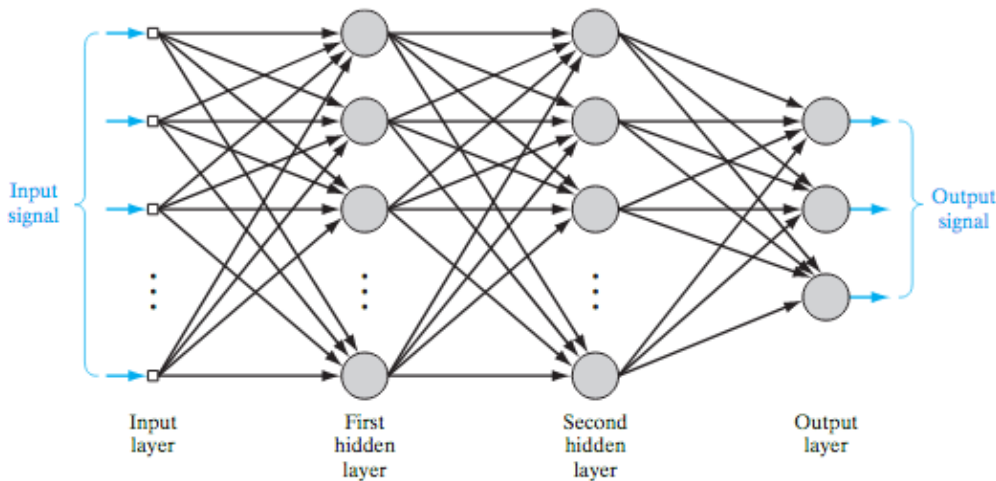


Figure 2.3: Multilayer perceptron with two hidden layers [14]

### 2.1.2.3 Random Forest Classifier

One commonly utilized algorithms for grappling with both categorical targets or grouping problems as well as continuously defined (numerical) targets such as regression problems is Random Forest. The algorithm uses multiple decision trees, each with the same nodes, but with different data leading to specific leaves. It merges decision-making from different decision trees to find a solution that reflects the sum of all those decision trees. Actually, this algorithm uses randomly splits the entire data. After that, it uses a variety of decision trees for each data sample and grabs predictions from each tree. Finally, there is a voting system; prediction with most votes is selected as the final prediction. So, it increases the predictive power

of algorithm along with preventing overfitting. This is an assembly of randomized decision trees that are in turn used for classification and regression.

#### 2.1.2.4 Decision Tree Classifier

Decision tree can be defined as a supervised learning algorithm that can measure as well as evaluate problems of regression and classification. This tree is best equipped to be able to predict the response for a given data. The decision tree structure begins from the root node that defines the tree's principal junction. It represents all the given samples and will be further divided into sub nodes. Dividing a node to sub-node is known as splitting. The resulting sub-nodes divide into more sub-nodes, these are called decision nodes. Leaf nodes are those nodes which are unable to divide anymore. Pruning is the process of removing a node from its tree whereas splitting is the addition of a node.

Each node in the tree describes the test case specific attributes and the outgoing/descending edge represents the answers to the question. This action is repeated for every node in the tree until each test case is classified. Decision tree uses multiple algorithm to divide a specific node in order to create multiple sub-nodes. The homogeneity of the resultant sub-node increases with the increase in creation of sub-nodes.

In decision tree selecting the attribute for the root node has to be identified. The best methods to identify these attributes are:

1) Gini Index: Gini Index measures the probability of an randomly chosen variable could be wrong misclassified.

$$Gini = 1 - \sum_{i=1}^C P_i^2 \quad (4)$$

2) Information gain: When the root node splits into sub nodes, the Entropy changes. The Entropy is the randomness of an attribute of set S.

$$IG(A, S) = H(S) - \sum_{t \in T} P(t).H(t) \quad (5)$$

$$H(S) = \sum_{i=1}^c p_i \cdot \log_2(p_i) \quad (6)$$

#### 2.1.2.5 Extra Tree Classifier

Another name for Extra Tree classifier is Extremely Randomized Trees classifier, which is an adaptation of a random forest. The whole sample is used for each stage and decision boundaries are selected at arbitrarily instead of the best like Random Forest. Considering the use case scenarios, output might be like a regular random forest, often just a little bit better.

### 2.1.2.6 Bagging

”Bagging” or bootstrap aggregation is a kind of Machine Learning method used by ensemble learning which build platform for machine learning. This is very simple but a powerful ensemble method. This idea comes on a hope of getting better fitting for models. Main thinking of bagging is to consider each bagging part as an individual brain. Machine Learning might have a single smart brain without bagging to work but with bagging the process is done by many weak brains or less strong brains together. As different brains would have different domain of thinking and sometimes those thinking may be overlapped, we would get a better result that we would get from a single brain.

Overfitting the training data with individual trees while bagging with decision trees is not that distressing. To ensure effectiveness, every decision trees tend to be grown to the depth and trees are not removed as well. Those trees should have strong variance as well as low bias. The bagging parameter is the amount of samples and therefore the amount of trees to be used. These numbers are determined by growing the sum of trees that continues running until precision starts to demonstrate improvement.

## 2.2 Literature Review

Ambwani et al. [15] implemented SVM for multiclass classification using KDD'99 dataset. This multi class classification model was used to correctly classify five classes in the given dataset.

Gao et al. [16] used the KDD'99 dataset by DARPA to implement an intrusion detection system. In the study, Deep Belief Networks (DBN) was used to perform malicious software classification. Comparing the results of DBN to ANN and SVM, DBN has an accuracy of 93.49% while the other two yields accuracy of 82.30% and 86.82% respectively.

Zhang et al. [17] proposed a LeNet-5 and LSTM neural network-based model for intrusion detection. It is found that CNN+LSTM2 model has the best performance in binary classification with an accuracy score of 0.999.

Babenko et al. [18] worked with only DDoS (Distributed Deny of Service) Attack with LVQ Model (Learning vector quantization) and claimed that the identification system failure does not exceed 10% in the sample series. The average number of identification errors is 84% . The usage of LVQ neural network based DDOS recognition models to tackle the problem of event classification is a viewpoint for use of intrusion detection systems, enterprise information protection management systems, and decision support systems.

Zhou et al. [19] in their literature have compared several machine learning algorithms to test their performance on classification of the attacks in CSE-CIC-IDS2018 in a one-vs-one procedure. This paper compares the performance between six machine learning algorithms to recognize unknown attacks where they have used the dataset CSE-CIC-IDS2018 for training and created a dataset with new attacks to test the performance of the models.

Kanimozhi et al. [20] describes an ANN model using the CIC-IDS 2018 dataset for binary classification of intrusions. The study shows classification of the packet data into two categories, malicious and normal. The model provides 99.97% accuracy score and very low FPR. The observed area under ROC curve for this model is found to be 0.999.

Ullah et al. [21] presents a two-stage model for anomaly-based intrusion detection. CICIDS2017 and UNSW-15 datasets were used for evaluation of the model. This study shows the implementation of SMOTE oversampling and various SMOTE technique. However, borderline1 algorithm provided the best outcome. In first stage, network anomalies or new attacks can be recognized implementing a decision tree classifier while the second level is held for classifying into trained classes implementing a random forest classifier. The authors used Precision, recall, and F score as metrics of evolution. After completing the training process, the CICIDS2017 dataset scores 100% and the UNSW-15 dataset scored 97% at testing phase.

Lin et al. [22] used LSTM for constructing an attack detection and attack type classification model. The model provides an accuracy of 96.2% on the CICIDS-2018 dataset. SMOTE oversampling technique was used to handle to class imbalance by generating synthetic samples for the minority classes. An Attention Mechanism (AM) was used to enhance the model performance.

Kim et al. [23] studied a CNN model for the purpose of utilizing for intrusion detection. The model in the literature is trained and evaluated on CICIDS-2018 dataset. The studied model shows two convolution layers, two pooling after each

of them and lastly a fully connected layer for classification. The data was reshaped into an image of size 13x6 and fed into the model. Instead of training one model on the entire dataset, the subsets of the dataset were trained. The literature provides performance measure of their proposed model on ten separate subsets instead of the entire dataset. So overall accuracy of the model is not measured.

Azwar et al. [24] differentiated multiple machine learning approaches and found their limitations and drawbacks while running the algorithm on the CIC-IDS 2017 dataset. This literature underlines the utilization of Random Forest, Decision Tree, Xgboost on the dataset. The final accuracy accomplished from the mentioned strategies is 92%.

Panigrahi et al. [25] used Deep Enforcement Learning Algorithm. Basically, they used Deep Q Network Algorithm, which is mainly value-based Re-enforcement Learning Algorithm Technique. Their Main goal was detect attack without depending on past experience. They worked on 85 different attributes which considered as an effective way of detecting different types of attacks.

# Chapter 3

## Dataset

### 3.1 Dataset Description

Our focus in this study is the working with the dataset CSE-CIC-IDS2018 [7] that was created by the Canadian Institute of Cybersecurity (CIC) in 2018. It is hosted on amazon webserver and has to be downloaded using the AWS CLI as a package that includes data of network (PCAP), comma-separated values file (CSV) and log files. We used the csv files for this study. Total size of the data is 6.89 gigabytes which is distributed across ten csv files. Other existing intrusion datasets are DARPA98, KDD99, ISC2012, ADFA13 and UNSW-15. But they are dated and the attacks described are not applicable for current time. CSE-CIC-IDS2018 is a comparatively newer dataset that contains attack data properly labeled with 80 features and a total of approximately 16.23 million rows. The eighty features in this dataset are based on the statistics of network flow. The author and creator of the dataset, Sharafaldin et al. illustrates the importance of network traffic flow in intrusion detection in their study [26]. Many other researchers also believe that flow data can give us a better insight into the network traffic [27]. The features are extracted using CICFlowmeter from network packet data (pcap) files included in the package. This dataset contains 14 attack labels and 6 sort of attacks that have been described in Figure 3.1.

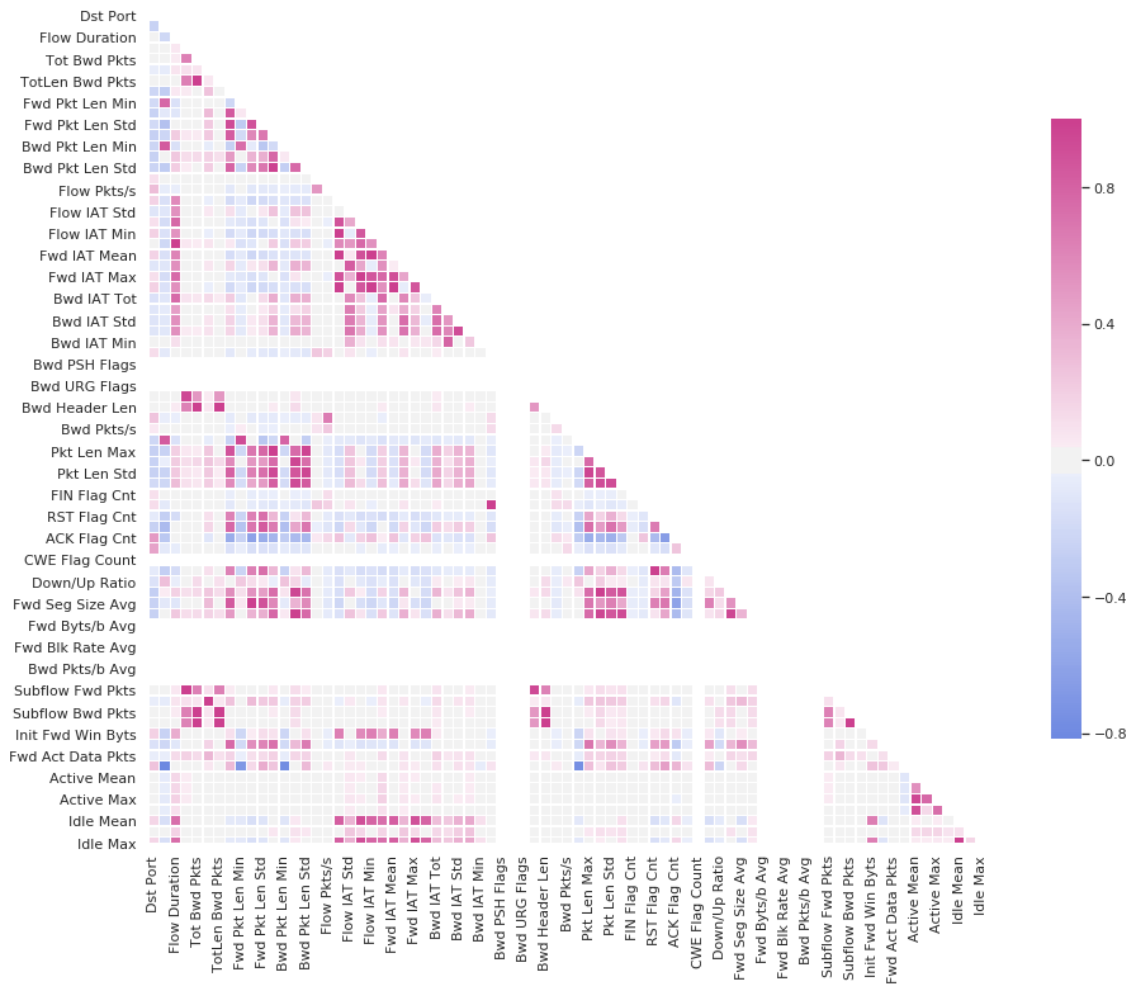


Figure 3.1: Heatmap of the dataset



## 3.2 Type of Attacks

The dataset contains six types of well-known and up-to-date attacks. It is very important to work with recent attacks that previous datasets lack due to the changing nature of attacks with time. There are fourteen labels mentioned in the dataset based on the tools used. The available attack types are Denial of Service (DoS), Distributed Denial of Service (DDoS), Bruteforce, Botnet, Infiltration and SQL Injection.

Attack Type	Labels	Description
DoS (Denial of Service)	DoS attacks-Hulk	DoS is a type of attack where the attacker tries to cease the resources making it unavailable for the user to use. The attacker usually sends an unusual number of requests making it crash.
	DoS attacks-SlowHTTPTest	
	DoS attacks-Slowloris	
	DoS attacks-GoldenEye	
DDoS (Distributed Denial of Service)	DDoS attack-LOIC-UDP	A DDoS attack resembles denial of service attack but here attackers use multiple systems to flood the bandwidth of target network to make it unavailable.
	DDoS attack-LOIC-HTTP	
	DDoS attack-HOIC	
Bruteforce attack	SSH-Bruteforce	Bruteforce attacks are used to gain unauthorized access into a network by decoding a weak password. Bruteforce attack uses a dictionary to match with the victim's password and tries different combinations.
	FTP-Bruteforce	
	Bruteforce -XSS	
	Bruteforce-Web	
Botnet	Botnet	A botnet deploys a number of bots to a system to run malicious activities such as keystroke logging, screen capturing, spying using the webcam, starting multiple malicious activities on victim's computer.
Infiltration	Infiltration	Infiltration from inside attack is demonstrated as an attacker sends a malicious file to the target via email which upon opening deploys a backdoor on target system for the intruder to access. After that the attacker has full access to victim's computer and access the data.
Web attacks	SQL Injection	Web attacks described in the dataset use vulnerabilities of MySQL web application to access the system and run malicious activities

Table 3.1: Type of attacks in CSE-CIC-IDS2018 dataset.

### 3.3 Class Imbalance in Dataset

Class imbalance in a dataset means that number of instances of classes not having a balanced proportion. Imbalance of classes is visible in a lot of classification problems with real life data.

CSE-CIC-IDS2018 dataset has class imbalance where 'Benign' class has the most amounts of data in the entire dataset. This imbalance makes it difficult for accurate classification. Imbalanced datasets often seem to make the classification biased towards class with majority instances causing misclassification of the class with minority instances [28]. We have used SMOTE oversampling technique to resolve the issue of class imbalance. SMOTE is seen to have improved the performance of the classifier in many literatures [21], [22], [29]. Figure 3.2 displays a bar chart of the percentage per class instance to give a better understanding of the imbalance.

Attack Type	Count
Benign	13484708
DDoS	1263933
DoS	654300
Bruteforce	381790
Bot	286191
Infiltration	161934
SQL Injection	87

Table 3.2: Number of occurrences of each attack type in CSE-CIC-IDS2018

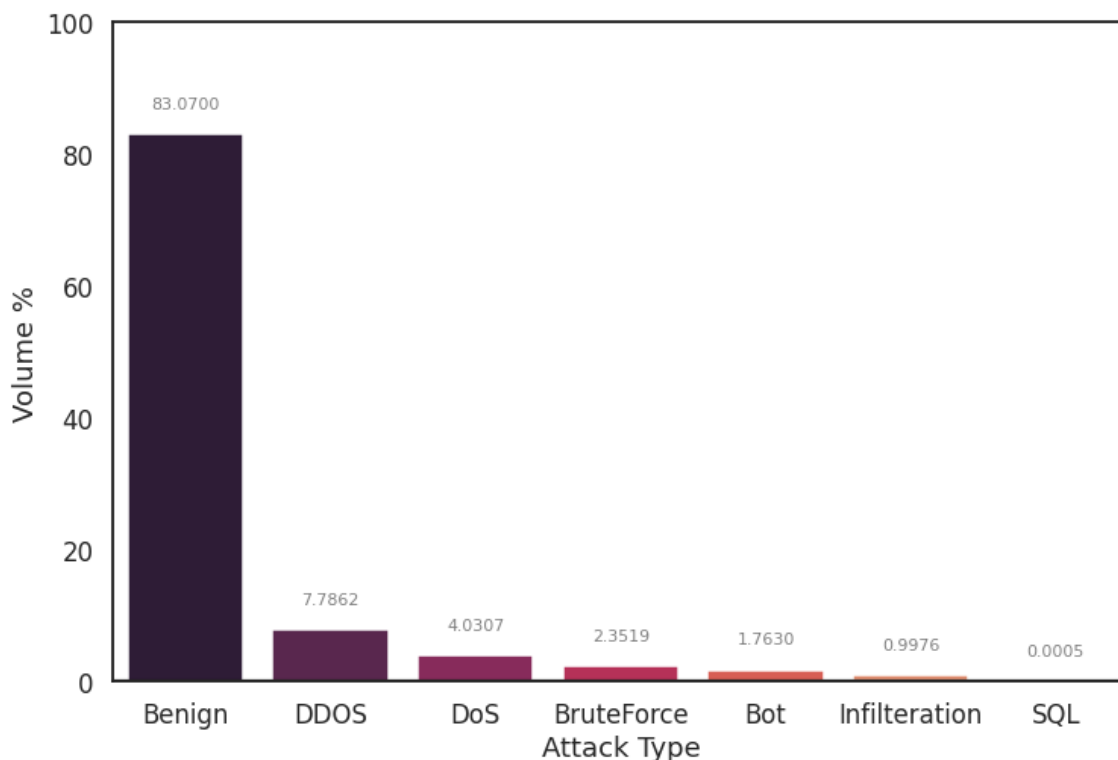


Figure 3.2: Percentage instance of classes in CSE-CIC-IDS2018

### 3.4 Features in Dataset

CSE-CIC-IDS-2018 contains 79 features and one target label. Features were extracted utilizing CICFlowmeter that creates Bidirectional Flows, where the first packet moves in both the forward (origin to goal) and backward (goal to origin) direction. Subsequently, the measurable features such as Duration, No. of bundles, No. of bytes, Length of bundles, and so forth are likewise determined independently in the forward and backwards [10]. Features in this dataset are mentioned in Table 3.1.

Feature Name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction
Bw_pkt_l_std	Standard deviation size of packet in backward direction
fl_byt_s	flow byte rate that is number of packets transferred per second
fl_pkt_s	flow packets rate that is number of packets transferred per second
fl_iat_avg	Average time between two flows
fl_iat_std	Standard deviation time two flows

fl_iat_max	Maximum time between two flows
fl_iat_min	Minimum time between two flows
fw_iat_tot	Total time between two packets sent in the forward direction
fw_iat_avg	Mean time between two packets sent in the forward direction
fw_iat_std	Standard deviation time between two packets sent in the forward direction
fw_iat_max	Maximum time between two packets sent in the forward direction
fw_iat_min	Minimum time between two packets sent in the forward direction
bw_iat_tot	Total time between two packets sent in the backward direction
bw_iat_avg	Mean time between two packets sent in the backward direction
bw_iat_std	Standard deviation time between two packets sent in the backward
bw_iat_max	Maximum time between two packets sent in the backward direction
bw_iat_min	Minimum time between two packets sent in the backward direction
fw_psh_flag	Number of times the PSH flag was set in packets travelling in the forward
bw_psh_flag	Number of times the PSH flag was set in packets travelling in the
fw_urg_flag	Number of times the URG flag was set in packets travelling in the
bw_urg_flag	Number of times the URG flag was set in packets travelling in the
fw_hdr_len	Total bytes used for headers in the forward direction
bw_hdr_len	Total bytes used for headers in the forward direction
fw_pkt_s	Number of forward packets per second
bw_pkt_s	Number of backward packets per second
pkt_len_min	Minimum length of a flow
pkt_len_max	Maximum length of a flow
pkt_len_avg	Mean length of a flow
pkt_len_std	Standard deviation length of a flow

pkt_len_va	Minimum inter-arrival time of packet
fin_cnt	Number of packets with FIN
syn_cnt	Number of packets with SYN
rst_cnt	Number of packets with RST
pst_cnt	Number of packets with PUSH
ack_cnt	Number of packets with ACK
urg_cnt	Number of packets with URG
cwe_cnt	Number of packets with CWE
ece_cnt	Number of packets with ECE
down_up_ratio	Download and upload ratio
pkt_size_avg	Average size of packet
fw_seg_avg	Average size observed in the forward direction
bw_seg_avg	Average size observed in the backward direction
fw_byt_blk_avg	Average number of bytes bulk rate in the forward direction
fw_pkt_blk_avg	Average number of packets bulk rate in the forward direction
fw_blk_rate_avg	Average number of bulk rate in the forward direction
bw_byt_blk_avg	Average number of bytes bulk rate in the backward direction
bw_pkt_blk_avg	Average number of packets bulk rate in the backward direction
bw_blk_rate_avg	Average number of bulk rate in the backward direction
subfl_fw_pk	The average number of packets in a sub flow in the forward direction
subfl_fw_byt	The average number of bytes in a sub flow in the forward direction
subfl_bw_pkt	The average number of packets in a sub flow in the backward direction
subfl_bw_byt	The average number of bytes in a sub flow in the backward direction
fw_win_byt	Number of bytes sent in initial window in the forward direction

bw_win_byt	# of bytes sent in initial window in the backward direction
Fw_act_pkt	# of packets with at least 1 byte of TCP data payload in the forward
fw_seg_min	Minimum segment size observed in the forward direction
atv_avg	Mean time a flow was active before becoming idle
atv_std	Standard deviation time a flow was active before becoming idle
atv_max	Maximum time a flow was active before becoming idle
atv_min	Minimum time a flow was active before becoming idle
idl_avg	Mean time a flow was idle before becoming active
idl_std	Standard deviation time a flow was idle before becoming active
idl_max	Maximum time a flow was idle before becoming active
idl_min	Minimum time a flow was idle before becoming active

Figure 3.3: Features in CIC-IDS2018 dataset

# Chapter 4

## Methodology

### 4.1 Work Plan

Our workplan for this paper can be visualized in Figure 4.1.

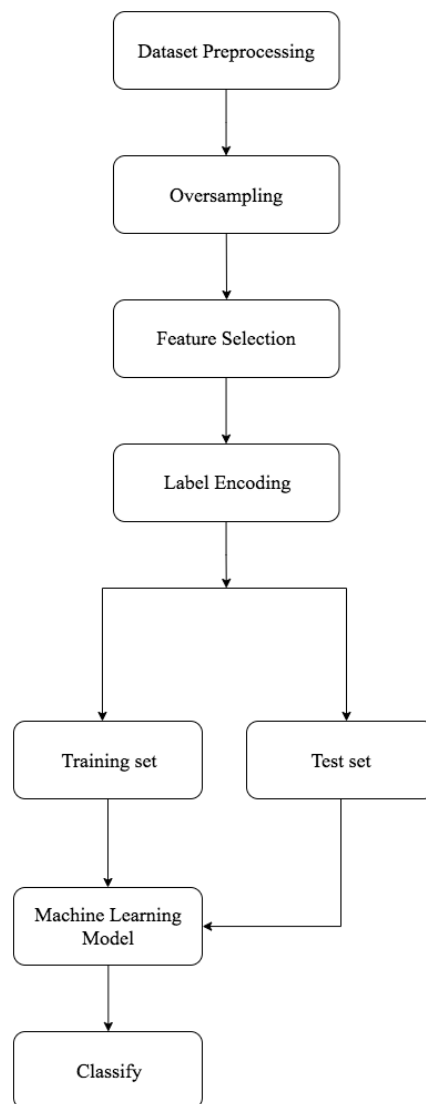


Figure 4.1: Work Plan for proposed model

### 4.1.1 Dataset Preprocessing

Like any other real-life datasets CSE-CIC-IDS-2018 contains noisy data and errors. Preprocessing of the dataset is required to make it usable for classification. The following steps were carried out in dataset preprocessing part:

1. Convert to pickle: The dataset is initially available in csv format but was converted to pickle format to reduce memory usage and faster processing. Python pickle package is used to serialize and de-serialize an object.
2. Replace NaN values: NaN, stands for Not a Number. It is a data type where the value is undefined and cannot be used as a floating-point number. NaN values in a dataset often gives NaN value error in many classifiers or negatively affect the result since numeric values are required. All the NaN values were replaced with zero.
3. Replace Infinity values: Some points in the dataset were Infinity. They were replaced by maximum value of the column.
4. Standardization: Standardizing the features of a dataset is required for many classifiers. Features are standardized by eliminating the mean and scaling to variance of 1. For a sample  $x$ , it is defined as:

$$z = \frac{(x - \mu)}{\sigma} \quad (7)$$

Here,  $\mu$  denotes the sample mean and  $\sigma$  denotes the standard deviation. Sci-kit learn StandardScaler function was used for standardization in this paper.

5. Drop duplicate rows: We have used `drop_duplicates` function in pandas library to remove all the reoccurring rows in the dataset.

### 4.1.2 Feature Selection

Careful selection of features is really important as it affects the performance, training time of a classifier. There are some redundant features which degrade the training performance and use more resources. Feature selection should be logical and done carefully. The dataset has 79 features as mentioned before.

Timestamp was removed as time of attack should not matter for classification. After that we looked for features with zero variance. Zero variance features have no change in data points throughout the dataset so keeping them would be irrelevant. We have found such eight features which are given in Table 4.1. After removing these features, we are left with 68 features that we have used in our experiment.

Features with zero variance
<code>bwd_blk_rate_avg, bwd_byts_b_avg, bwd_pkts_b_avg, bwd_psh_flags, bwd_urg_flags,</code> <code>fwd_blk_rate_avg, fwd_byts_b_avg, fwd_pkts_b_avg</code>

Figure 4.2: Zero variance features in CSE-CIC-IDS2018 dataset



### 4.1.3 Split Dataset into Training and Test Set

We need a portion of data for our classifier to train with and the rest for testing the performance of the classifier. The general convention is to train with largest portion and test with a smaller portion of data. We have use Train\_test\_split function from Sklearn Library in python with train-test ratio of 70:30 to split our dataset. Stratify parameter was turned True to make sure all classes maintain this ratio of train and test data.

### 4.1.4 Synthetic Minority Oversampling Technique (SMOTE)

We have previously mentioned that the dataset has class imbalance issue. To overcome that we have implemented the Synthetic Minority Oversampling Technique (SMOTE) [30] to oversample the minority classes to bring balance throughout the dataset. This technique generates new instances of the class with low number of data points. The next data points are synthesize by recognizing the K-Nearest Neighbors in a class and measuring the distance between neighbor and the sample. The distance is multiplied by 0 or 1 to synthesize the next data point. SMOTE generates data points that are similar to its neighbors. One good side of this technique is that it does not generate the same instance multiple times unlike it's other alternative Random Over Sampler that randomly picks value and duplicates it.

In our thesis, we have used the imbalanced-learn library in python to oversample the minority classes. It is to be noted that, oversampling was done after the train-test split of the dataset and only applied to the minority classes of the training set and not test set.

### 4.1.5 Training the Classifier

In our thesis, we propose an architecture of Convolutional Neural Network which we have trained with the CICIDS-2018 dataset. We have trained a few other deep learning and machine learning models that have been used in previous literatures to assess the performance of the model. The selected classifiers are Deep Neural Network, Random Forest Classifier, Extra Tree Classifier, Decision Tree Classifier, and Bagging.

### 4.1.6 Evaluating the Performance

For multiclass classification only accuracy does not give us the correct evaluation. Precision , Recall , F1-score are some appropriate metrics for evaluation machine learning model [31].

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (8)$$

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (11)$$

# Chapter 5

## Implemented Algorithms

### 5.1 Proposed CNN Model Structure

In our proposed CNN architecture for intrusion detection, we have to reshape our feature vector into a 2-dimensional image. Since we have a 68 feature vector, it was reshaped into image of size 17 x 4 as input for our model. This input is fed into our convolutional neural network. It can be split into two sections – firstly, the Convolution layers and secondly, fully connected back propagation layers. Two 2d convolution layers were used with filter 32 and 64 respectively with padding and kernel size of 3x3. A Batch Normalization and a max pooling layer was used after both of the convolution layers. The output was then reshaped into a vector using a Flatten layer. Flatten layer output was then fed into the fully connected layers for classification. The fully connected layers have 4 hidden layers with (8,64,128,256) neurons. Usually ReLU (Rectified Linear Unit) is used as activation function and Adam as optimizer in CNN as we have seen in previous literatures. In our model, we have made changes to that and used mish [32] as activation function for the hidden layers and Ranger [33] as optimizer for the model to enhance the performance.

#### 5.1.1 Proposed CNN Model Parameters

Hyperparameters regulate the performance of a model. This requires a lot of tweaking and experiment to find the right parameters to achieve the best results. We have found the parameters that provides the most optimal result through a lot of trials and errors. The hyperparameters as well as the arrangement of the layers are mentioned in Table 5.1 and 5.2.

<b>Convolution Layers</b>	<b>Hyperparameters</b>
Convolution 2d	filter size = 32, kernel size = (3,3), padding='same', activation=mish , input_shape=(17, 4, 1), kernel_regularizer= l2
BatchNormalization	
MaxPooling 2d	pool_size=(2, 2)
Convolution 2d	filter size = 64, kernel size = (3,3), padding='same', activation=mish , kernel_regularizer= l2
BatchNormalization	
MaxPooling 2d	pool_size=(2, 2)
Flatten	

Figure 5.1: Structure of Convolution Layers in proposed CNN model

<b>Fully Connected Layers</b>	<b>Hyperparameters</b>
Dense	Units = 8, activation=mish
Dropout	Rate = 0.2
Dense	Units = 64, activation=mish
Dropout	Rate = 0.2
Dense	Units = 128, activation=mish
Dropout	Rate = 0.2
Dense	Units = 256, activation=mish
Dense (Output layer)	Units = 7, activation=softmax

Figure 5.2: Structure of Fully Connected Layers in proposed CNN model

### 5.1.2 Activation function

We have used Mish as our activation function in place of mostly used ReLU activation function. It is defined as (8). Figure 6 shows the graph of Mish activation function.

$$f(x) = x \cdot \tanh(\ln(1 + e^x)) \quad (12)$$

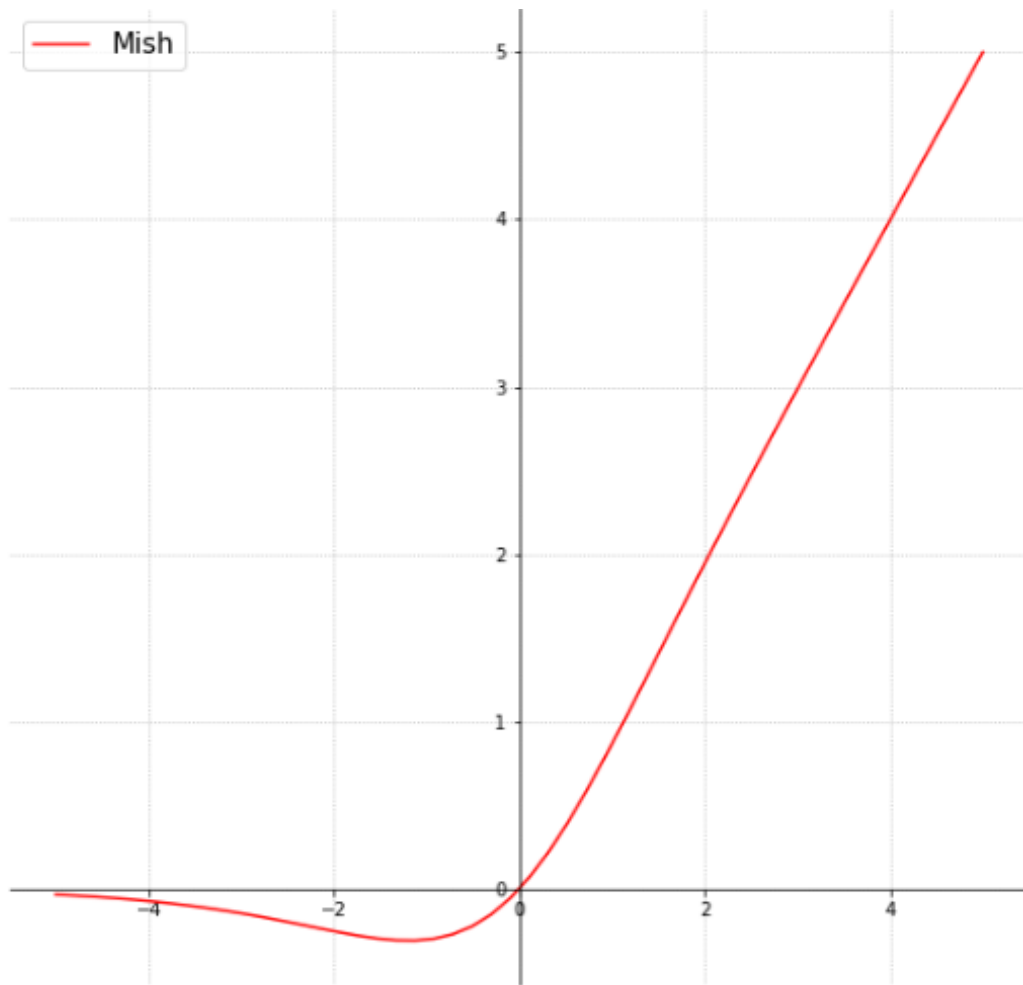


Figure 5.3: Mish Activation Function [30]

### 5.1.3 Optimizer

In our model we have used Ranger optimizer that is formed by merging Rectified Adam (RAdam) into recently introduced LookAhead optimizer. This seem to give a better result than Adam optimizer. One of the reasons being Adam can fall into local minima without proper warmup where Rectified Adam outperforms implementing a Rectifier that dynamically controls the adaptive momentum throughout the training [33]. Look Ahead optimizer which was recently introduced, updates two sets of weights and looks ahead at the sequence of fast weights provided by a secondary optimizer choosing the search direction. LookAhead provides faster training and better convergence than other optimizers [34]. When these two optimizer, RAdam and LookAhead are combined together, they tend to co-operate and work well together minimizing variance and lowering the convergence time. The hyperparameters used for the optimizer are given in Table 7.

RAdam	total_steps = 10000, warmup_proportion = 0.1, min_lr = 1e-5
LookAhead	optimizer = RAdam, sync_period = 6, slow_step_size = 0.5

Figure 5.4: Hyperparameters for the optimizer

## 5.2 DNN Architecture

DNN model used in this literature is a Back Propagation Neural Network with four hidden layers of size 8, 64, 128 and 256. We have used Mish as our activation function of the hidden layers and Ranger as the choice of optimizer for the model. The structure is the same as the fully connected layers of the CNN model presented in this paper displayed in figure 5.2.

## 5.3 Random Forest Classifier Parameters

Random Forest classifier parameters used in this paper are displayed in Figure 5.4.

n_estimators = 100, criterion = 'gini', min_samples_split = 2
---

Figure 5.5: Parameters for Random Forest Classifier

## 5.4 Decision Tree Classifier Parameters

Decision classifier parameters used in this paper are displayed in Figure 5.5.

splitter='best'
-----------------

Figure 5.6: Parameters for Decision Tree Classifier

## 5.5 Extra Tree Classifier Parameters

Random Forest classifier parameters used in this paper are displayed in Figure 5.6

n estimators = 100, criterion = 'gini', min samples split = 2, min samples leaf = 1
---

Figure 5.7: Parameters for Extra Tree Classifier

## 5.6 Bagging Classifier Parameters

Random Forest classifier parameters used in this paper are displayed in Figure 5.7.

n estimators = 10, bootstrap = True
-------------------------------------

Figure 5.8: Parameters for Bagging Classifier

# Chapter 6

## Evaluation

### 6.1 Experimental Setup

The experiments were carried out on a local machine with 16 gigabytes of RAM, AMD Ryzen 5 2600 CPU and Nvidia GTX1050Ti GPU on python version 3.7. Python is the most used language in machine learning and data science because of its vast available library. Python libraries used to build our classification model are Pandas, Numpy, Tensorflow, Keras, Scikit-Learn and Imbalanced-Learn. Keras is a high-level neural network API for tensorflow. Tensorflow is a computational library in python created by Google that can be utilized for implementation of deep learning and machine learning models.

### 6.2 Results

For comparing the performance of the algorithm, we have used Accuracy, Precision, Recall, F1-score, macro average and weighted average for empirical evaluation of the models. We have used `classification_report` function from Sci-kit Learn library in python to calculate the performance metrics.

### 6.2.1 Result of CNN model

Performance summary of our proposed CNN model is displayed in Table 6.1. The model has been able to yield an accuracy of 0.99 on test set.

Label	Precision	Recall	F1-score
Benign	0.99	1	0.99
DDoS	0.99	0.99	0.99
DoS	0.99	0.98	0.98
Bruteforce	1	0.99	1
Bot	1	0.99	0.99
Infiltration	0.96	0.93	0.95
SQL Injection	0.87	0.81	0.85
Accuracy	0.99		
Macro avg	0.97	0.96	0.96
Weighted avg	0.98	0.99	0.98

Table 6.1: Performance summary of CNN model

### 6.2.2 Result of DNN model

Performance summary of our DNN model is displayed in Table 6.2. The model has been able to yield an accuracy of 0.98 on test set.

Label	Precision	Recall	F1-score
Benign	0.98	0.99	0.99
DDoS	0.96	0.98	0.97
DoS	0.96	0.96	0.96
Bruteforce	0.98	0.99	0.99
Bot	0.93	0.98	0.96
Infiltration	0.86	0.78	0.83
SQL Injection	0.82	0.72	0.76
Accuracy	0.98		
Macro avg	0.92	0.91	0.92
Weighted avg	0.97	0.98	0.98

Table 6.2: Performance summary of DNN model



### 6.2.3 Result of Random Forest Classifier

Performance summary of Random Forest Classifier is displayed in Table 6.3. The model has been able to yield an accuracy of 0.95 on test set.

Label	Precision	Recall	F1-score
Benign	0.93	0.97	0.95
DDoS	1	1	1
DoS	0.96	0.94	0.93
Bruteforce	0.84	0.94	0.89
Bot	0.99	1	1
Infiltration	0.67	0.58	0.53
SQL Injection	0.92	0.42	0.58
Accuracy	0.95		
Macro avg	0.9	0.84	0.84
Weighted avg	0.95	0.95	0.95

Table 6.3: Performance summary of Random Forest Classifier

### 6.2.4 Result of Decision Tree Classifier

Performance summary of Decision Tree Classifier is displayed in Table 6.4. The model has been able to yield an accuracy of 0.93 on test set.

Label	Precision	Recall	F1-score
Benign	0.92	0.94	0.93
DDoS	1	1	1
DoS	0.97	0.9	0.93
Bruteforce	0.84	0.94	0.89
Bot	1	1	1
Infiltration	0.5	0.43	0.46
SQL Injection	0.79	0.73	0.76
Accuracy	0.93		
Macro avg	0.86	0.85	0.85
Weighted avg	0.92	0.93	0.92

Table 6.4: Performance summary of Decision Tree Classifier

### 6.2.5 Result of Extra Tree Classifier

Performance summary of Extra Tree Classifier is displayed in Table 6.5. The model has been able to yield an accuracy of 0.92 on test set.

Label	Precision	Recall	F1-score
Benign	0.92	0.94	0.93
DDoS	1	1	1
DoS	0.96	0.9	0.93
Bruteforce	0.84	0.92	0.89
Bot	0.99	0.98	0.99
Infiltration	0.51	0.38	0.43
SQL Injection	0.75	0.58	0.65
Accuracy	0.92		
Macro avg	0.85	0.81	0.83
Weighted avg	0.91	0.92	0.91

Table 6.5: Performance summary of Extra Tree Classifier

### 6.2.6 Result of Bagging Classifier

Performance summary of Bagging Classifier is displayed in Table 6.6. The model has been able to yield an accuracy of 0.93 on test set.

Label	Precision	Recall	F1-score
Benign	0.92	0.96	0.94
DDoS	1	1	1
DoS	0.96	0.9	0.93
Bruteforce	0.84	0.94	0.89
Bot	1	1	1
Infiltration	0.54	0.36	0.43
SQL Injection	0.79	0.58	0.67
Accuracy	0.93		
Macro avg	0.86	0.82	0.84
Weighted avg	0.92	0.93	0.92

Table 6.6: Performance summary of Bagging Classifier

### 6.3 Analysis

Metrics of each of the classifiers have been summarized in Table 18. From this table we can analyze and come to a verdict of which model performed the best for our IDS.

Model	Precision		Recall		F1-Score		Accuracy
	WeightedAverage	MacroAverage	WeightedAverage	MacroAverage	WeightedAverage	MacroAverage	
CNN	0.98	0.97	0.99	0.96	0.98	0.96	0.99
DNN	0.97	0.92	0.98	0.91	0.98	0.92	0.98
RandomForest	0.95	0.9	0.95	0.84	0.95	0.84	0.95
Decision Tree	0.92	0.86	0.93	0.85	0.92	0.85	0.93
Extra Tree	0.91	0.85	0.92	0.81	0.91	0.83	0.92
Bagging	0.92	0.86	0.93	0.82	0.92	0.84	0.93

Table 6.7: Summarized result of all trained model in the thesis

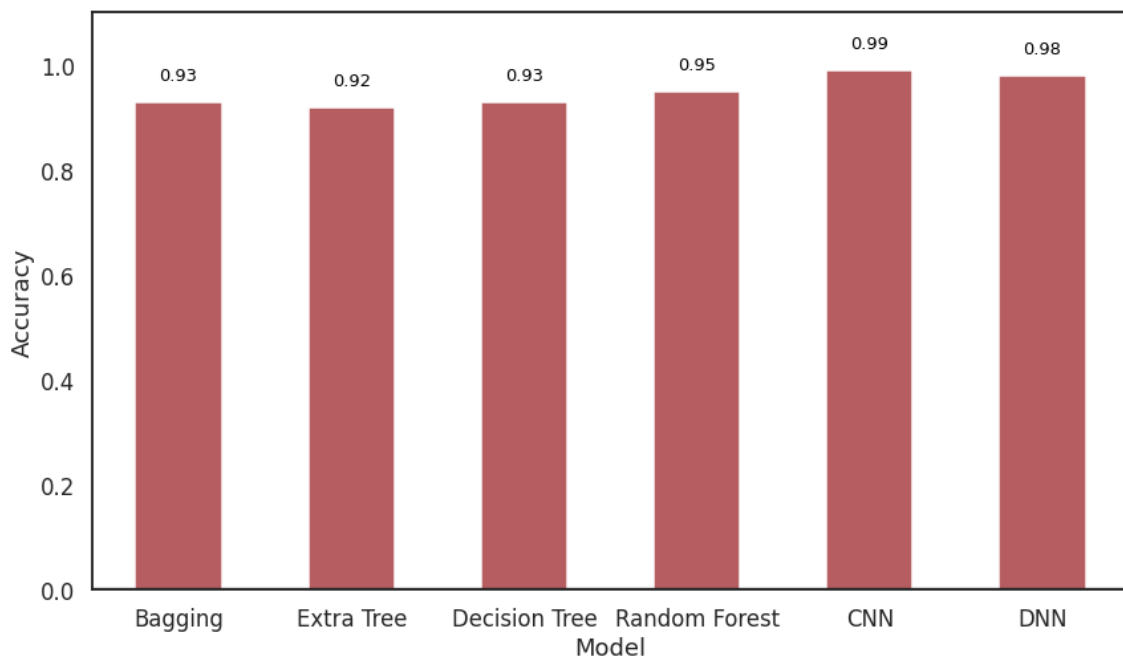


Figure 6.1: Accuracy comparison of CNN and other models.

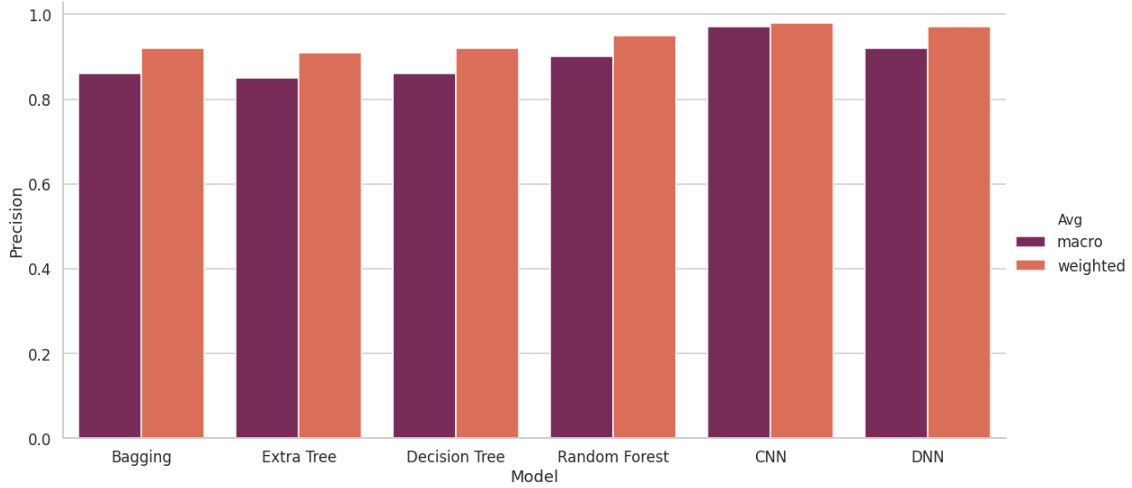


Figure 6.2: Precision comparison of CNN and other models.

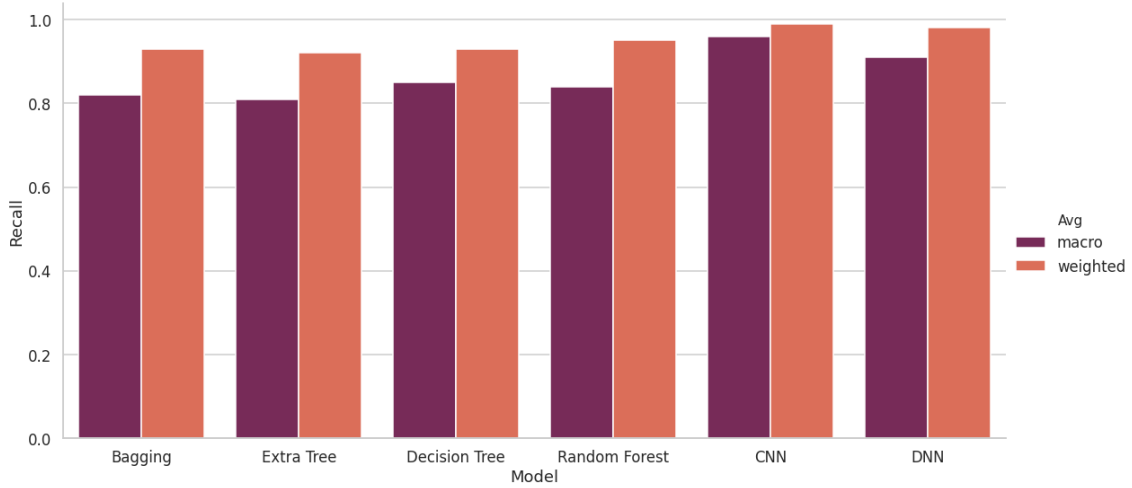


Figure 6.3: Recall comparison of CNN and other models.

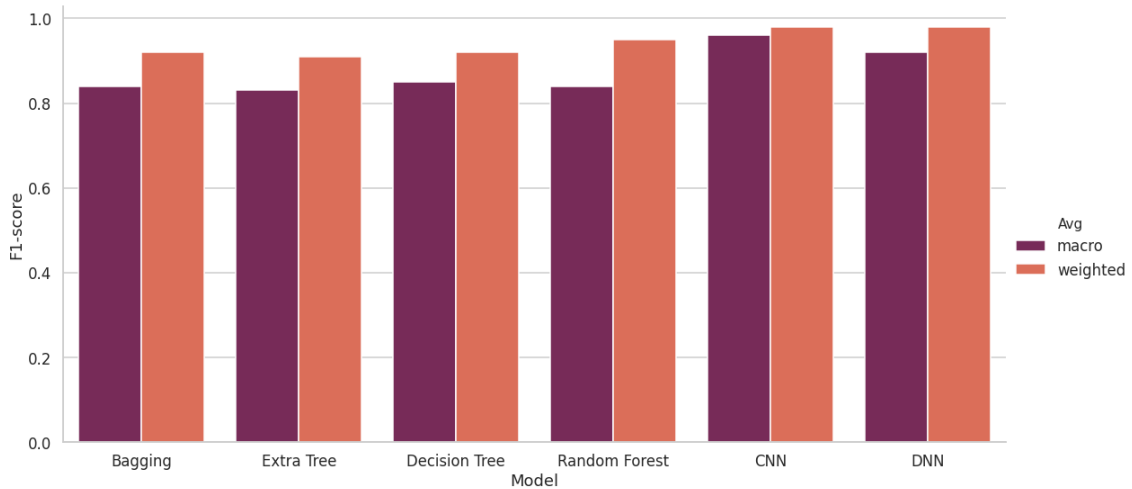


Figure 6.4: F1-score comparison of CNN and other models.

From the Table 6.7, Figure 6.1 and Figure 6.3, it is visible that CNN has the highest accuracy (8) as well as weighted and macro average recall. If we look at formula of Recall (10), it is the true-positive-rate or sensitivity of a model. Also, this can be called detection rate in case of intrusion detection system. It is necessary for any model to have high detection rate if we want to deploy in a real network system. Therefore, we can say that our proposed CNN model works better than other classification model providing the highest accuracy and detection rate. So it is implementable in real-life scenarios to detect intrusions and properly classify them.

# Chapter 7

## Conclusion

We believe that the Convolutional Neural Network model is better than other Intrusion detection models with the advantage of a successful classification of six different types of intrusion. However, we faced a few challenges while conducting this thesis. Firstly, the codes and implementations were carried out on a local computer. The training process is very time-consuming because of the limited available resources such as memory, processor and graphics card. Also, the parameters for our model was set based on multiple trials and errors and comparison of results. This experiment process could be accelerated on cloud computing systems such as Google Cloud Platform that could enable us to do more experiments based on trials and errors.

We hope that this model will perform well on newer flow-based datasets as well. We want to construct a framework for future implementation to build a fully functional Intrusion Detection System for real-time intrusion detection and classification. Firstly, raw network packet data will be captured and processed through CICFlowMeter for feature extraction. After that, this data will then be preprocessed removing noisy values. Finally, this processed data can be run through our CNN model to detect and classify the intrusion type in real time.

# References

- [1] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, “Iot middleware: A survey on issues and enabling technologies”, *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.
- [2] T. S. Bernard, T. Hsu, N. Perlroth, and R. Lieber, *Equifax says cyberattack may have affected 143 million in the u.s.* The New York Times, The New York Times Company, Sep. 2017. [Online]. Available: [www.nytimes.com/2017/09/07/business/equifax-cyberattack.html](http://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html).
- [3] L. Clarke, *Cyber-attack on us health agency aimed to disrupt covid-19 response*, Mar. 2020. [Online]. Available: <https://tech.newstatesman.com/security/us-health-human-services-department-cyber-attack>.
- [4] J. Cannady, “Artificial neural networks for misuse detection”, Nov. 1998.
- [5] G. Karatas and O. K. Sahingoz, “Neural network based intrusion detection systems with different training functions”, in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018, pp. 1–6.
- [6] J. P. Anderson, “Computer security threat monitoring and surveillance”, James P. Anderson Company, Fort Washington, Tech. Rep., 1980.
- [7] *License*. [Online]. Available: <http://www.unb.ca/cic/datasets/ids-2018.html>.
- [8] K. Zetter, *That insane, \$81m bangladesh bank heist? here’s what we know*, May 2016. [Online]. Available: <https://www.wired.com/2016/05/insane-81m-bangladesh-bank-heist-heres-know/>.
- [9] A. Cuthbertson, *Dark web data dump sees 620 million accounts from hacked websites go on sale*, The Independent. [Online]. Available: <https://www.independent.co.uk/life-style/gadgets-and-tech/news/dark-web-data-hackers-dubsmash-myfitnesspal-myheritage-cyber-security-a8775666.html>.
- [10] A. Özgür and H. Erdem, “Intrusion detection classifiers comparison in different operating environments”, Nov. 2012.
- [11] R. Kemmerer and G. Vigna, “Vigna, g.: Intrusion detection: A brief history and overview. computer 35, 27-30”, *Computer*, vol. 35, pp. 27–30, May 2002. DOI: 10.1109/MC.2002.1012428.
- [12] S. Hossen and A. Janagam, “Analysis of network intrusion detection system with machine learning algorithms (deep reinforcement learning algorithm).”, vol. 4, p. 2020, Apr. 2020. [Online]. Available: <http://www.diva-portal.org/smash/get/diva2:1255686/FULLTEXT02.pdf>.

- [13] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Essen, A. Awwal, and V. Asari, “A state-of-the-art survey on deep learning theory and architectures”, *Electronics*, vol. 8, p. 292, Mar. 2019. DOI: 10.3390/electronics8030292.
- [14] S. S. Haykin, *Neural networks and learning machines*, Third. Upper Saddle River, NJ: Pearson Education, 2009.
- [15] T. Ambwani, “Multi class support vector machine implementation to intrusion detection”, vol. 3, Aug. 2003, 2300–2305 vol.3, ISBN: 0-7803-7898-9. DOI: 10.1109/IJCNN.2003.1223770.
- [16] N. Gao, L. Gao, Q. Gao, and H. Wang, “An intrusion detection model based on deep belief networks”, in *2014 Second International Conference on Advanced Cloud and Big Data*, 2014, pp. 247–252.
- [17] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, “Network intrusion detection: Based on deep hierarchical network and original flow data”, *IEEE Access*, vol. 7, pp. 37 004–37 016, 2019.
- [18] T. Babenko, S. Toliupa, and Y. Kovalova, “Lvq models of ddos attacks identification”, in *2018 14th International Conference on Advanced Trends in Radio-electronics, Telecommunications and Computer Engineering (TCSET)*, 2018, pp. 510–513.
- [19] Q. Zhou and D. Pezaros, *Evaluation of machine learning classifiers for zero-day intrusion detection – an analysis on cic-aws-2018 dataset*, 2019. arXiv: 1905.03685 [cs.CR].
- [20] V. Kanimozhi and T. P. Jacob, “Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing”, in *2019 International Conference on Communication and Signal Processing (ICCSPP)*, 2019, pp. 0033–0036.
- [21] I. Ullah and Q. H. Mahmoud, “A two-level hybrid model for anomalous activity detection in iot networks”, in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–6.
- [22] P. Lin, K. Ye, and C.-Z. Xu, “Dynamic network anomaly detection system by using deep learning techniques”, in Jun. 2019, pp. 161–176, ISBN: 978-3-030-23501-7. DOI: 10.1007/978-3-030-23502-4\_12.
- [23] J. Kim, Y. Shin, and E. Choi, “An intrusion detection model based on a convolutional neural network”, *Journal of Multimedia Information System*, vol. 6, pp. 165–172, Dec. 2019. DOI: 10.33851/JMIS.2019.6.4.165.
- [24] H. Azwar, M. Murtaz, M. Siddique, and S. Rehman, “Intrusion detection in secure network for cybersecurity systems using machine learning and data mining”, in *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 2018, pp. 1–9.
- [25] R. Panigrahi and S. Borah, “A detailed analysis of cicids2017 dataset for designing intrusion detection systems”, vol. 7, pp. 479–482, Jan. 2018.
- [26] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “A detailed analysis of the cicids2017 data set”, in *Information Systems Security and Privacy*, P. Mori, S. Furnell, and O. Camp, Eds., Cham: Springer International Publishing, 2019, pp. 172–188, ISBN: 978-3-030-25109-3.



- [27] G. Munz and G. Carle, “Real-time analysis of flow data for network attack detection”, in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, pp. 100–108.
- [28] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics”, *Information Sciences*, vol. 250, pp. 113–141, Nov. 2013. DOI: 10.1016/j.ins.2013.07.007.
- [29] J.-H. Seo and Y.-H. Kim, “Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection”, *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–11, Nov. 2018. DOI: 10.1155/2018/9704672.
- [30] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, “Smote: Synthetic minority over-sampling technique”, *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, Jan. 2002. DOI: 10.1613/jair.953.
- [31] T. Kautz, B. Eskofier, and C. Pasluosta, “Generic performance measure for multiclass-classifiers”, *Pattern Recognition*, vol. 68, Mar. 2017. DOI: 10.1016/j.patcog.2017.03.008.
- [32] D. Misra, *Mish: A self regularized non-monotonic neural activation function*, 2019. arXiv: 1908.08681 [cs.LG].
- [33] L. Wright, *New deep learning optimizer, ranger: Synergistic combination of radam + lookahead for the best of both*. 2019. [Online]. Available: <https://medium.com/@lessw/new-deep-learning-optimizer-ranger-synergistic-combination-of-radam-lookahead-for-the-best-of-2dc83f79a48d>.
- [34] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, *Lookahead optimizer: K steps forward, 1 step back*, 2019. arXiv: 1907.08610 [cs.LG].