

Denial of Service (DoS) Attack Detection Using Machine Learning



Prepared by: Yousef Ashraf

ID: 171655

A Dissertation submitted to the Faculty of Engineering, The British University in
Egypt, in Partial Fulfillment of the requirements for the bachelor's degree in
electrical engineering

Supervisor: Prof. Gamal Ibrahim

June,2022

Electrical and Communications Engineering Department

Denial of Service (DoS) Attack Detection Using Machine Learning

Dissertation Approved:

Dr.

Committee member

Committee member

External committee member

Affiliation:

Dean Faculty of Engineering

Acknowledgments

First and foremost, praises and thanks to Allah, the Almighty, for his showers of blessings throughout the whole year to complete this research successfully.

I would like to express my deep and sincere gratitude to my research supervisor Prof. Gamal Ibrahim for providing invaluable guidance throughout this research. His vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present it as clearly as possible. It was a great privilege and honor to work under his guidance.

Last but not least, I am extremely grateful to my parents for their prayers, caring and sacrifices for educating and preparing me for my future.

Abstract

In the current era of rapid development of the Internet, methods of information security and safety are increasingly being focused on. As a result, denial of service (DoS) and distributed denial of service (DDoS) attacks are becoming threats to all servers around the world. Since then, methods and services to prevent and minimize damage have also been launched with many different functions. Some outstanding problems when implementing defense against DoS/DDoS attacks is the timing of the attacks detection. The early detection of these attacks needs a highly accurate mechanism to be implemented. In addition, attack methods are increasingly diverse with increasing scale, making prevention methods quickly disabled. After researching a quite enough number of the latest academic studies, I've considered the highly expected effect of applying deep learning and machine learning to the DoS/DDoS attack detection solution. In this research paper, I've used convolutional neural networks (CNN) and deep neural networks (DNN) models in the DDoS attacks detection. As a result, of using the CICDDoS2019 dataset in training my proposed models and comparing their accuracies in detecting and classifying the DDoS attacks. I've proposed three models, Model 1 using CNN in detecting and classifying the incoming DDoS attacks with an accuracy of 99.15%. Model 2 using DNN in detecting and classifying the DDoS attacks with an accuracy of 99.33%. Model 3 using also DNN in only detecting the attacks without any sort of classification with an accuracy of 99.99%. These high accuracy results proved how effective are the Deep learning techniques dealing with the DDoS attacks prevention.

Contents

Acknowledgments.....	3
Abstract.....	4
List of Figures	7
Chapter 1: Introduction	9
1.1. Background.....	9
1.2. Problem Statement	11
1.3. Motivation.....	11
1.4. Research Objective	12
Chapter 2: DDoS Attacks.....	12
2.1. Attack Procedure.....	12
2.2. Types of Attacks	13
2.2.1. SYN Flooding (Half Open Connection)	13
2.2.2. User datagram protocol (UDP) flood attack	14
2.2.3. Low rate Denial of Service (LDoS)	16
2.2.4. Internet Control Message Protocol (ICMP)	16
2.2.5. Slowloris	17
2.2.6. HTTP-Get Flood	17
2.2.7. Ping of Death	18
2.3. Time Plan	19
2.3.1. Gantt Chart.....	19
2.3.2. Achievements of semester 1	20
Chapter 3: Theoretical Basis	21
3.1. Machine Learning	21
3.1.1. Supervised Learning	21
3.2. Deep Learning.....	22
3.3. Deep learning models	23
3.3.1. Neural Networks	23
3.3.2. Activation functions.....	25
3.3.3. Tensorflow and Keras Libraries.....	26
3.4. Software Defined Networks (SDN)	27
3.5. DDoS Datasets	28
Chapter 4: Literature Review	29

4.1. Classical techniques	29
4.2. Statistical techniques.....	31
4.3. Intelligent techniques	32
4.3.1. Machine learning techniques	32
4.3.2. Artificial Intelligence techniques	34
Chapter 5: Dataset: CIC-DDoS2019.....	38
5.1. Reflection based DDoS attacks.....	39
5.2. Exploitation based DDoS attacks.....	39
Chapter 6: Data Preprocessing.....	40
6.1. Managing Null Entries.....	40
6.2. Handling Categorical Data.....	40
6.3. Standard Deviation.....	41
6.4. Examining Correlation.....	41
6.5. Dataset Standardization	42
6.6. Managing Outliers	42
6.7. Splitting the dataset into Test and Train	43
6.8. Data preprocessing Colab Simulations	43
6.9. Evaluation and comparison of performance	51
Chapter 7: Proposed Models Simulation Results.....	54
7.1. Model 1: Classification using Convolutional Neural Networks (CNN)	56
7.2. Model 2: Classification using Deep Neural Networks (DNN)	60
7.3. Model 3: Identification using Deep Neural Networks (DNN).....	65
Chapter 8: Comparative Analysis	71
Chapter 9: Conclusion and Future works.....	73
9.1. Conclusion	73
9.2. Future works	73
References.....	74
Appendix A.....	82
Abstract.....	82
Problem Statement.....	82
Main and Specific Objective.....	82
Brief Background.....	83
Literature Review.....	83

Research Methodology	84
Simulation work.....	84
Simulation Results	87
Comparative Analysis	91
Conclusion	92
Future works	93
Main References	93

List of Figures

Fig. 1: DDoS attack mechanism [3]	9
Fig. 2: Some large-scale DDoS attacks	10
Fig. 3: TCP Header [10].....	13
Fig. 4: TCP SYN Flood [11]	14
Fig. 5: UDP Flood Mechanism [12].....	15
Fig. 6: UDP flood attack [14]	15
Fig. 7: ICMP attack mechanism [16].....	16
Fig. 8: HTTP-GET flood DDoS attack [20].....	18
Fig. 9: Ping of death attack [22]	19
Fig. 10: Gantt chart	19
Fig. 11: Block diagram of the proposed model	20
Fig. 12: Some machine learning algorithms [23]	21
Fig. 13: Supervised machine learning algorithm [23]	22
Fig. 14: Biological Neural Network.....	23
Fig. 15: Artificial Neural Network.....	23
Fig. 16: Feedforward neural network [29]	24
Fig. 17: Convolutional Neural Networks	25
Fig. 18: Activation functions	25
Fig. 19: SDN architecture	28
Fig. 20: CIC-DDoS2019 dataset attacks[86].....	39
Fig. 21:Correlation Heat Map.....	42
Fig. 22: Reading and viewing the dataset from drive	43
Fig. 23: Dataset dimensions	44
Fig. 24: Dataset features	44
Fig. 25: Statistical analysis.....	45
Fig. 26: Updated dataset dimensions	45
Fig. 27 : Labeled attacks types	46
Fig. 28: No. of samples of each attack	46
Fig. 29: Percentage of each attack.....	47
Fig. 30: Label class distribution	47
Fig. 31: Normalized data	48
Fig. 32: Unique values	48

Fig.33: Numerical Features	49
Fig. 34: Categorical Features	49
Fig. 35: Discrete Numerical Features	49
Fig. 36: Categorical data histogram	50
Fig. 37: Features Heat map	50
Fig. 38: Separating Input and Output attributes	51
Fig. 39: Labeling the types of attacks	54
Fig. 40: Label counts.....	54
Fig. 41: Label Percentage	55
Fig. 42: Dataset Dimensions.....	55
Fig. 43: Label Class Distribution	56
Fig. 44: Train and Test split model 1	57
Fig. 45: Split dimensions model 2	57
Fig. 46: Model 1 Structure	57
Fig. 47: Training and validation for model 1	58
Fig. 48: Plotting Loss v/s No. of epochs model 1	58
Fig. 49: Plotting Accuracy v/s no. of epochs model 1	59
Fig. 51: Model 1 Performance.....	59
Fig. 52: Previously known testing array model 1	59
Fig. 53: Prediction array model 1	60
Fig. 54: Model 1 Report.....	60
Fig. 55: Train and Test Split model 2	61
Fig. 56: Split Dimensions model 2	61
Fig. 57: Model 2 Structure	62
Fig. 58: Training and Validation for model 2.....	62
Fig. 59: Plotting Loss v/s No. of epochs model 2	63
Fig. 60: Plotting Accuracy v/s no. of epochs model 2	63
Fig. 61: Model 2 Performance.....	64
Fig. 62: Prediction array model 2	64
Fig. 63: Testing array model 2.....	64
Fig. 64: Model 2 Report.....	65
Fig. 65: BENIGN and MALIGN.....	66
Fig. 66: Label count	66
Fig. 67: Label Percentage	66
Fig. 68: Splitting Dataset model 3	67
Fig. 69: After Splitting Dataset model 3	67
Fig. 70: DNN Model 3	68
Fig. 71: Training and Validation model 3	68
Fig. 72: Plotting Loss v/s Epochs model 3	69
Fig. 73: Plotting Accuracy v/s Epochs model 3.....	69
Fig. 74: Model 3 performance.....	70
Fig. 75: Testing array model 3.....	70
Fig. 76: Prediction array model 3	70
Fig. 77: Model 3 Confusion Matrix.....	71

Chapter 1: Introduction

1.1. Background

Denial of service (DoS) is well known to be a cyber attack where the legitimate users could not have access on the service anymore. The hacker use some attack methods in order to exhaust the network resources so that he can seize the service. The whole network and system can also be destroyed and the computer resources such as the RAM, Buffer , and the CPU can be totally occupied as a result of these attacks. When any of these resources is consumed by this unexpected traffic , it forms a bottleneck and the system's performance drops or stops, making it impossible for legitimate users to use it.

DoS attacks aren't aimed to steal data or gain access to systems by exploiting security measures. In reality, the public services that DoS attacks are meant to destroy are being used in large part as they were intended. The volume and timing of the attacker's usage pattern, is designed to have a negative influence on other people's ability to access the resource [1].

Denial of service attacks are known as one of the most dangerous and widespread variants of cyber attacks, one of the most damaging and worldwide affecting business. By overflowing a particular machine's bandwidth or resources, the attacker blocks legitimate users from connecting to the network. A DoS is when a single machine is utilized to launch an attack. The distributed DoS is an expanded version of DoS in which the attack is launched from multiple vulnerable devices in order to increase the stress level of the attack on a given machine or system [2].

The distributed denial of service threat method consists of four main parts. as shown in Figure1. Demons or zombies who are agent programs that carry out the attack on the intended victim, A control master program that coordinates the attack and commands the demons or zombies agents under its control, a victim who the attack is held against and an attacker.

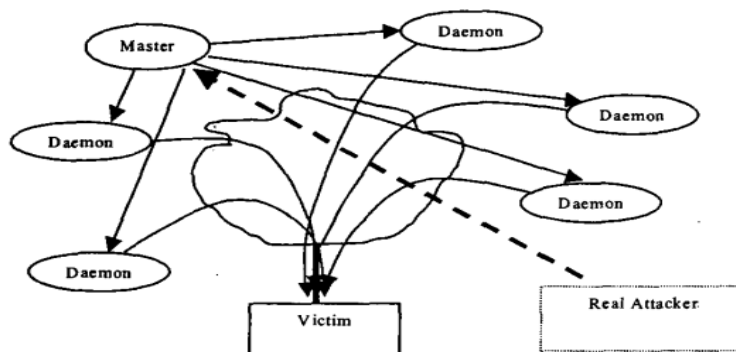


Fig. 1: DDoS attack mechanism [3]

The Distributed Denial of Service (DDoS) multiplies the effect of the normal DoS attack resulting in a stronger effect of the attack carried by the malware author which makes it more harder for the intrusion detection systems to detect or prevent such attacks. In DDoS the attackers typically obtain access to a large number of machines by exploiting their weaknesses in order to establish Botnets or zombies. These Botnets or zombies are used then by the attacker to launch a controlled, massive attack against one or more victims.

The attacks cause a considerable threat in distributed denial of service because they prevent legal traffic exchange between compromised servers and legitimate clients. These attacks can either perform, low volume traffic for long period of time, high volume traffic for short period of time, or even high volume traffic for long period of time. This will automatically leads the targeted network or server to be totally exhausted due to either, bandwidth depletion or resource depletion depending on the type of the DDoS attack held against this server or network.

According to [4], in 2018, Github - a company providing the world's largest source storage service in the United States, has undergone the largest DDOS attack in history, Score up to 1.35Tbps. This is a consequence of weak security of memcached servers, helping attackers easily take control and implementation amplification attacks. This type of attack is recorded similar to the NTP amplification attack but with scale and strength greater many times. Figure 2 below lists some large-scale DDoS attacks recognized so far.

According to [5,] the globe experienced the greatest "DDOS" attack ever directed against Dyn, a US DNS service provider, in 2016. This assault is the result of IoT device security flaws, which allowed an attacker to take control and install a Botnet. The onslaught reached a high of 1.2Tbps.

Attack unit	Description	Time	Highest traffic
BBC	British media	2015	602 Gbps
Spamhaus	A non-profit organization specializes in tracking email spam and spam-related activities.	2013	~300Gbps
Github	Web storage service for source code management.	2018	1.35 Tbps
Cloudflare	A US company specializes in providing network security services.	2014	~400 Gbps
Dyn	DNS service provider.	2016	1.2Tbps

Fig. 2: Some large-scale DDoS attacks

1.2. Problem Statement

Distinguishing between flooding attacks and legitimate user activity is the main challenge in the Denial of Service attacks. The process of detection becomes more harder when an attacker uses a Distributed Denial of Service .

Tracking down the real attacker as unexpected users' PCs are utilized as botnets or zombies to launch the attacks on the victim server is also a tough issue. Furthermore, there are no predefined IP addresses for demons or zombies computers, and even if some of them are discovered and halted, the attacker can always exploit new machines.

Even zombie computers or botnets don't always connect directly with victim servers. They fake the victim server's IP address and send requests to a huge number of reflector machines. The reflectors transmit a huge number of replies to victim servers, because by default they have to reply back to all requests from which they believe they are the source.

The distributed denial of service attacks can not be prevented or detected easily because the malicious traffic from an attacker is the same as the genuine traffic coming from legitimate users. For this reason it is difficult for the targeted servers or intrusion detection systems recognize a difference between both traffics. For this reason there are much of false positives and false negatives.

1.3. Motivation

The huge spread of internet nowadays in everything in our daily life and the evolution in network technology and applications requires a continuous development in network security. DoS attack is considered as one of the most dangerous intrusions. It also became more difficult to identify or prevent DDoS attacks since the innovation in IoT, artificial intelligence technologies, and cloud computing.

The detection of unauthorized DDoS traffic is a challenge in the protection of communication and information resources. The constant increase in the quantity and magnitude of DDoS attacks since their initial appearance in 2000 is direct proof of a growing problem, despite ongoing study into the subject field and the implementation of new prevention and detection solutions.

The wide use of cloud networks nowadays definitely increases the danger of cyber attacks. Cloud security became a very critical issue specially in the coming years as most of the companies , organizations and even individuals are gradually going towards the usage of cloud infrastructure.

As a result of various additional vulnerabilities caused by the cloud's nature such as resource sharing and multi tenancy, serious threats to the availability of cloud services existed due to the denial of service [6].

Researches and security tools have to be up to date as the attackers' tools and techniques are improving continuously. We need new tools and algorithms using more intelligent techniques such as machine learning and deep learning algorithms for the purpose to increase the attack detection efficiency. This intelligent techniques will allow the targeted network or server to recognize the difference between the malicious and legitimate traffic or packets more efficiently.

1.4. Research Objective

In cybersecurity, both Denial of Service (DoS) and Distributed Denial of Service (DDoS) are seen to be of the most damaging cyberattacks. DDoS is known to be a severe attack that interrupts the usual operation of cloud computing services.

The distributed denial of service attacks are classified as serious threats that disrupt network functionality. These attacks have gotten complex and continue to increase rapidly, making it difficult to detect and respond to them. We need to use more intelligent techniques by implementing artificial intelligence algorithms for increasing the attack detection efficiency. This intelligent techniques will allow the targeted network or server to recognize the difference between the malicious and legitimate traffic or packets more efficiently. The main problem that must be fixed in detecting denial of service attacks is to try reducing the false positives and false negatives as much as possible.

Therefore, the objectives of this paper are , mitigating undiscovered harmful traffic that masquerades as legal traffic, investigating an appropriate dataset for the chosen machine learning algorithm and using machine learning in detecting such attacks in order to reduce the false positives and false negatives. Mainly because machine learning strategies can adapt with the changes in attack pattern and tunes themselves accordingly.

Chapter 2: DDoS Attacks

2.1. Attack Procedure

The DoS attack procedure have some steps which are associated. The hacker initially direct a high volume traffic of service requests to the server. As a result, the server responds to this incoming traffic from the attacker and waits for a response back. However, because the transmitted malicious traffic is using spoofed source addresses, the targeted host or server will not receive any reply back and will have to wait for a longer period till connection is terminated.

Server resources held for such requests cannot be released. Thus, if a high volume traffic is sent, server will be fully occupied and the users will not be able to get the service[7].

There are lots of perceptible DoS attack techniques. These techniques are divided according to the types of attacks. Application layer attacks, Protocol based attacks and Volume based attacks are considered to be the main three types of DoS attacks. Application layer attacks as HTTP get flood and Slowloris are measured in requests per seconds, Ping of Death and SYN Flood are examples of protocol based attacks which are measured in packets per seconds, and volume based attacks such as ICMP and UDP floods are measured in bits per seconds.

2.2. Types of Attacks

2.2.1. SYN Flooding (Half Open Connection)

SYN Flooding attack is known as TCP SYN attack as it uses the transmission control protocol (TCP), because as mentioned in [8] TCP is used in more than 90% of DoS attacks. It is also known as half open connections. TCP is considered as the main communication protocol for the internet and it is a protocol based attack.

TCP SYN attacks send a flood of malicious TCP SYN packets to the victim's listening TCP port. TCP SYN attacks can damage any connected system to the Internet that supports TCP network services, such as Mail servers or FTP servers, as well as Web servers [9].

“SYN” is the synchronization flags in the TCP headers which have a size of 1 bit as shown in Figure 3. When a packet is delivered using the transmission control protocol, the SYN flag in the TCP header is set showing the receiving system that it has to store the sequence number of the received packet [10].

		TCP Header																															
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			N S	C W	E R	U R	A C	P C	R S	S S	Y Y	I I	Window Size														
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if data offset > 5. Padded at the end with "0" bytes if necessary.)																															

Fig. 3: TCP Header [10]

Normal TCP connection between two hosts starts with exchanging synchronization and acknowledgment packets between each others to establish the connection, this procedure is known as the three-way handshake.

Half open connections occurred in SYN flooding attacks take use of the three way handshake protocol used in TCP. The server replies by a SYN ACK packet when receiving a SYN request. The connection remains in a half-open status till the client acknowledges the SYN ACK packet, which is generally configured to 75 seconds [9]. Unfortunately, this exploitation done by the attacker is not realized by the victim server.

The attacker uses a spoofed source addresses when transmitting SYN packets to the victim. The victim will reply back to the spoofed source with the SYN/ACK packet and wait till receiving the a last ACK packet for establishing the connection. If the final ACK packet is not received by the victim target, it waits for the connection timeout to terminate the half opened connection.

The attack occur because as shown in Figure 4 below. The number of half open connections that the victim's host TCP port can handle has been exceeded. As a result, the victim host will deny any additional incoming connections until the current sessions timer expire [11].

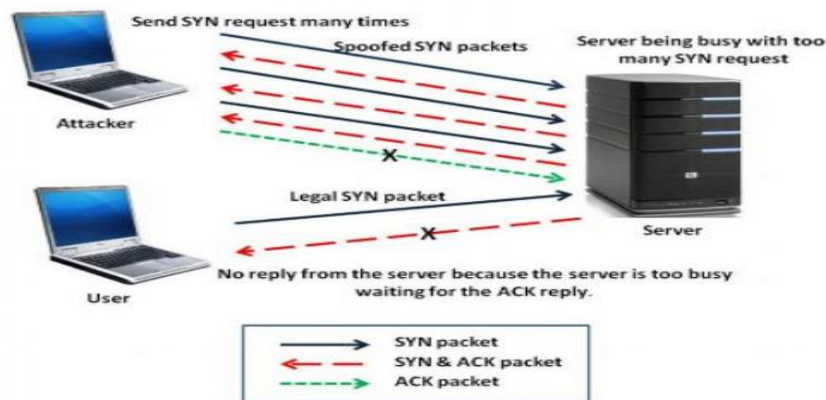


Fig. 4: TCP SYN Flood [11]

2.2.2. User datagram protocol (UDP) flood attack

The User datagram protocol (UDP) is not like TCP, it does not need any previous communication to start the connection. This allows the hacker to transmit a high volume UDP traffic to the victim. The UDP flood is considered as a volume based attack.

When a hacker sends high volume traffic to the victim's computer's random or specified destination ports. If the port is closed, the victim system will reply with appropriate ICMP packets upon receiving the UDP packet requests. A significant quantity of packet replies would cause the system to slow down or collapse [12,13].

DDoS attacks use demons or zombies to flood the victim host with a high volume of UDP traffic which results in network's bandwidth depletion, slowing down the system or crashing the system as shown in Figure 5.

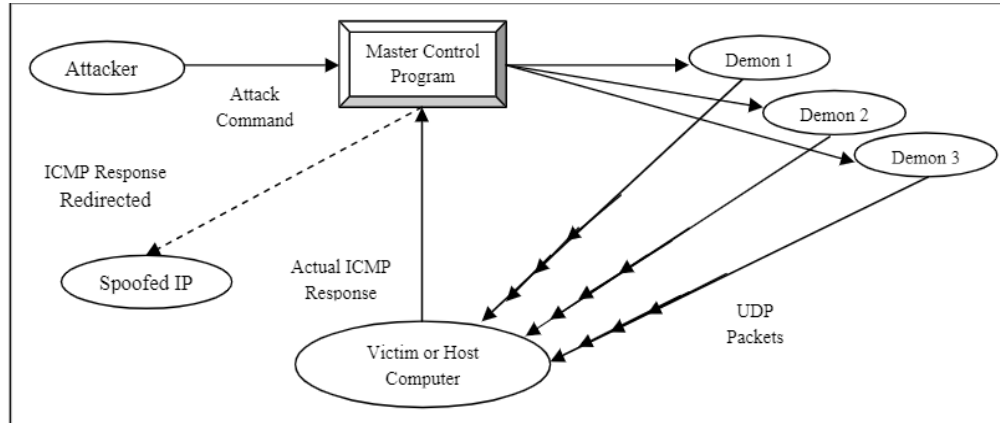


Fig. 5: UDP Flood Mechanism [12]

In IPv6, the UDP flood attack is done by using UDP protocol's CHARGEN and ECHO functions. As shown in Figure 6, the CHARGEN service is supported via UDP's 19th port. This port replies with a random string when trying to attach to it. The ECHO service is supported via UDP's 7th port, which replies with what you have sent to the entering node. The node from which the attack is initiated tries to connect the victim's port 19 to port 7 of another node. The procedure is done by transmitting a fake UDP packet to the victim port 19, with return address towards other's port 7. When this is accomplished we can notice a flood of traffic between the victim and other node [14].

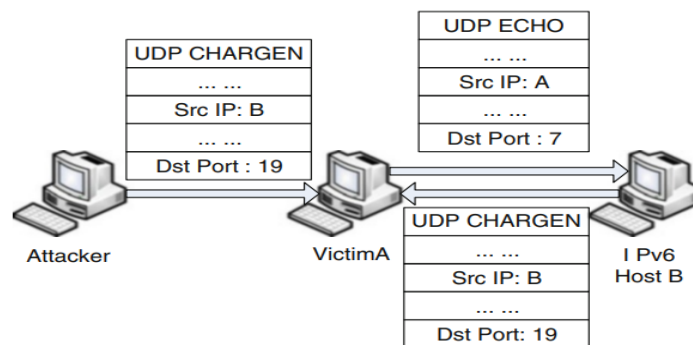


Fig. 6: UDP flood attack [14]

2.2.3. Low rate Denial of Service (LDoS)

Low rate denial of service is one of the most important and effective DoS attacks. Because of the low frequency of these attacks, detection is much more difficult. In LDoS we do not need anymore to make a constant high speed attack traffic to exhaust the network, but instead a high pulsed short term packets are sent in a regular basis causing the network damage.

LDoS is a variant of DoS threats that is more difficult to detect and more dangerous than standard DoS attacks. LDoS degrades the performance of network services by exploiting network congestion control protocols. It transmits short-term pulses on a regular basis to keep the network in a condition of recovery and congestion. The sent attack pulses in LDoS are relatively short in length, resulting in a very low average rate that is unrecognized [15].

2.2.4. Internet Control Message Protocol (ICMP)

ICMP flood is a volume based attack which is known as Ping flood. This type of attack uses the Internet Control Message Protocol which is a helper protocol that does not require a previous setup same as UDP. It is a protocol that checks the condition of the network and generally tells the host of better routes to reach the target. It also detects network faults and reports packet path issues.

ICMP Flood attacks make advantage of the Internet Control Message Protocol (ICMP), that is involved in sending an ICMP ECHO REQUEST packet to a remote host to see whether it is alive or not. Figure (7) depicts a DDoS ICMP flood attack in which the attacker sends a large number of faked ICMP echo request packets to the victim server. As a result of these packets requesting a response from the victim, the targeted network's bandwidth has become saturated rejecting any connection requests from legitimate users [16].



Fig. 7: ICMP attack mechanism [16]

2.2.5. Slowloris

Slowloris is a type of Slow DoS attacks (SDoS) and it is an application layer attack. Slowloris is difficult to be detected because it resembles actual user communication over a sluggish internet connection. The web server input queue is overloaded from the Distributed Denial of Service (DDoS) through the establishment of a high number of transmission control protocol connections [17].

It sends fractional HTTP requests indefinitely via these TCP established connections. As a result, while waiting for those fractional attack requests to be finished, the servers under attack leave the connections open [18].

In this way, for a DDoS attack the attacker can establish a huge number of connections with target host or server with the help of zombies. This will result in no space for new connections to be established. When a genuine user requests a resource from the server, the server will reject these requests.

2.2.6. HTTP-Get Flood

HTTP-Get Flood attack is an application layer DDoS attack. In the application layer it is the most used attack. In this type of DDoS attack, a high volume traffic of hypertext transfer protocol requests are sent to the victim web server automatically. An intrusion detection system (IDS) cannot identify if these HTTP-GET requests are malicious or legitimate since they have acceptable forms and are sent over normal TCP connections. As a result, servers must treat all requests as normal requests, eventually exhausting their resources [19].

A bot acts like any other authorized user, first establishes a genuine TCP connection to communicate with the target host. It then sends the server a significant number of HTTP GET queries and waits for responses the same way that an authorized user would. As shown in Figure 8, HTTP GET requests from bots are quickly gathered in the request queue. This will subsequently cause any incoming requests from legitimate users to be dropped [20].

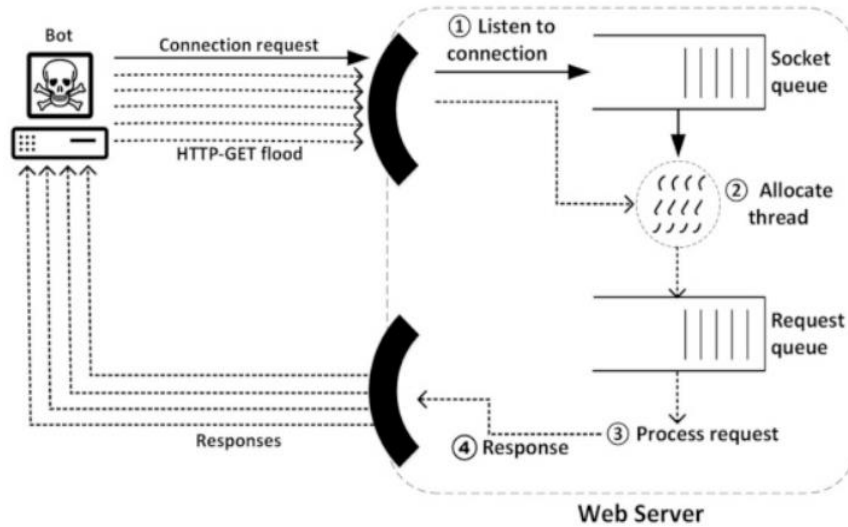


Fig. 8: HTTP-GET flood DDoS attack [20]

2.2.7. Ping of Death

Ping of death is a protocol based DoS attack . Oversized ping packets that are sent from the attacker can not be handled by computers resulting in a variety of system issues, including crashes.

Ping of Death occurs when an attacker sends a high volume of ping requests which are ICMP Echo Requests to the target server with packets of large size trying to knock it offline or exhaust its resources as it will be too busy replying with ICMP echo responses [21].

The maximum size of an IP packet, including headers, is 65535 bytes. Therefore, ping packets bigger than the maximum packet size might violate the Internet Protocol (IP). Computer systems were never designed to handle such packets. Malformed packets are usually sent in fragments by the attackers. As shown in Figure 9 the victim server system will reassemble the fragment, but the packet is excessive, resulting in memory overflows [22].

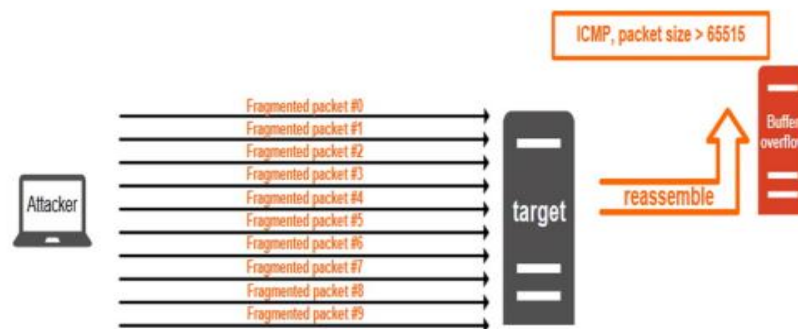
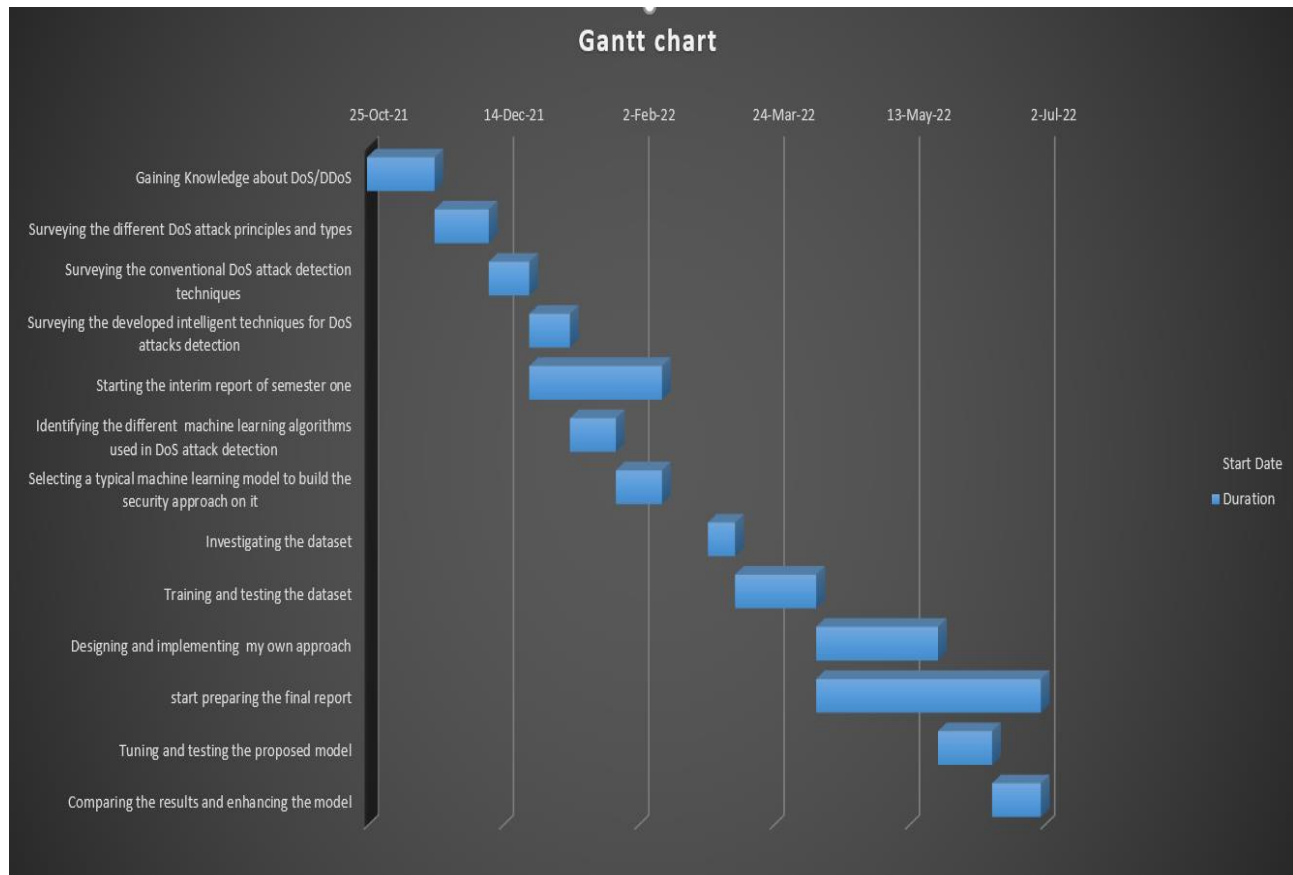


Fig. 9: Ping of death attack [22]

2.3. Time Plan

2.3.1. Gantt Chart



	Comparing the results and enhancing the model	Tuning and testing the proposed model	start preparing the final report	Designing and implementing my own approach	Training and testing the dataset	Investigating the dataset	Selecting a typical machine learning model to build the security approach on it	Identifying the different machine learning algorithms used in DoS attack detection	Starting the interim report of semester one	Surveying the developed intelligent techniques for DoS attacks detection	Surveying the conventional DoS attack detection techniques	Surveying the different DoS attack principles and types	Gaining Knowledge about DoS/DDoS
Start Date	13-Jun-22	24-May-22	9-Apr-22	9-Apr-22	10-Mar-22	28-Feb-22	25-Jan-22	8-Jan-22	24-Dec-21	24-Dec-21	9-Dec-21	19-Nov-21	25-Oct-21
Duration	18	20	83	45	30	10	17	17	49	15	15	20	25

Fig. 10: Gantt chart

2.3.2. Achievements of semester 1

Here is a brief of the achievements and tasks done in semester 1.

- Starting by Gaining as much as possible knowledge about what means Denial of Service (DoS) , its variants and classifications.
- Surveying the attack principles and how such attacks are threatening cybersecurity and how DoS is affecting the availability of information security.
- Surveying different DoS attack types , how they differ from each others and the procedure of each attack.
- Surveying the different conventional or classical DoS attack detection techniques.
- Surveying the developed intelligent techniques for DoS attacks detection and prevention.
- Start preparing the interim report of semester one in parallel with other tasks.
- Submitting the interim report.

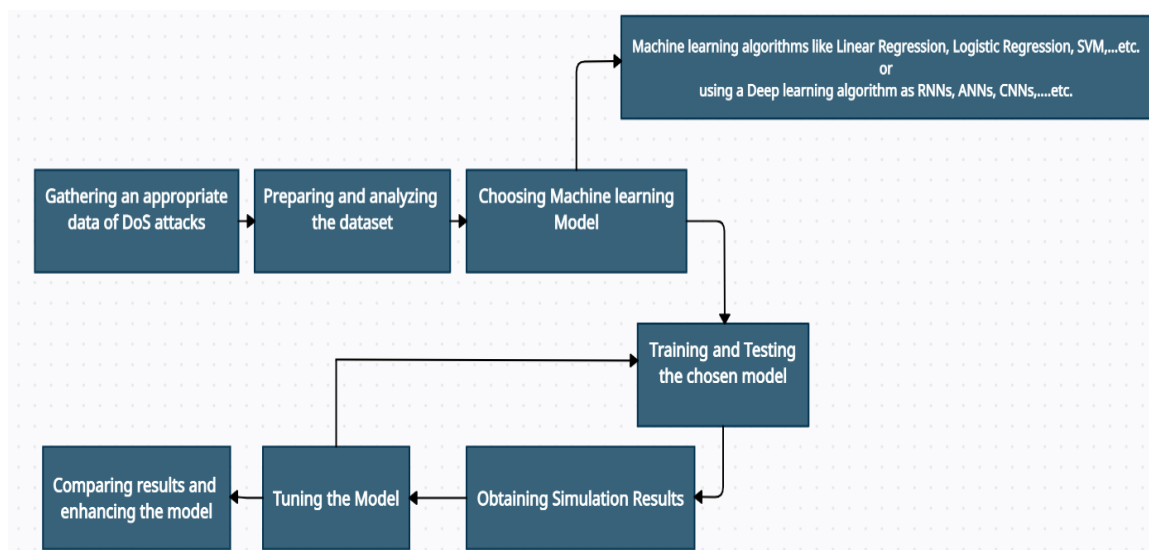


Fig. 11: Block diagram of the proposed model

Chapter 3: Theoretical Basis

3.1. Machine Learning

What is Machine learning?

As cited by A.Dey [23], machine learning is used in order to teach machines how to process data more efficiently. Sometimes, after looking at the data, we cannot interpret the patterns or extract information from them. In this case, we can apply machine learning. With the countless datasets available, the need for machine learning is growing everyday.

Many different industries nowadays like pharmaceuticals and military are using machine learning to extract relevant information from data. The main job of machine learning is to learn from data. There are many studies that have found ways for machines to learn from themselves [24], [25]. There are many mathematicians and programmers who applied multiple approaches and algorithms for getting solution to this problem and let machines learn from themselves. Some of them are shown in Figure [12] below.

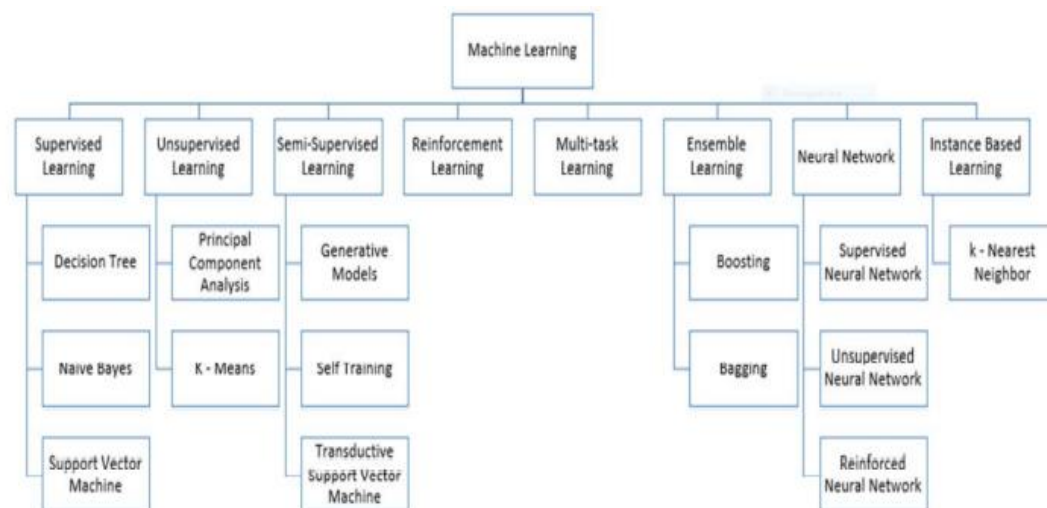


Fig. 12: Some machine learning algorithms [23]

3.1.1. Supervised Learning

Supervised learning algorithms are those that need internal support. The data input must be separated into two parts including one for testing and the other for training processes. The

training set has an output value that needs to be predicted or classified. All algorithms learn models from the training set and then apply them to the testing set for evaluation [26]. The supervised learning algorithm is shown in Figure [13] below.

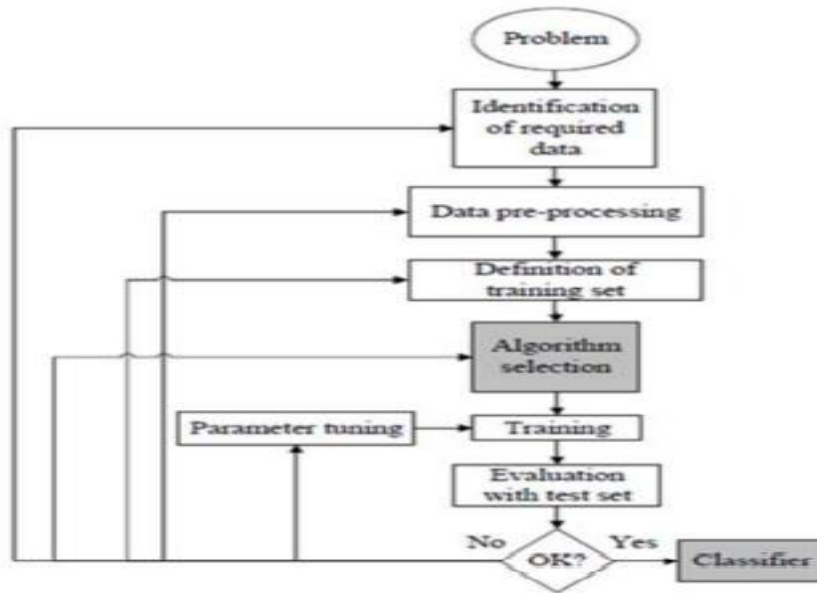


Fig. 13: Supervised machine learning algorithm [23]

3.2. Deep Learning

What is deep learning?

According to X. Du et al. [27], deep learning in the past was developed from artificial neural networks, and now it has become a popular field of machine learning. Research on artificial neural networks began in the 1940s. W. S. McCulloch and W. Pitts [28], have proposed the McCulloch-Pitts model for analyzing and synthesizing features of neural networks.

The concept of deep learning was first introduced in 2006. After that, deep learning continued to thrive.

Some big companies, such as Google and Facebook, have had a lot of successful research and applications in many different fields using deep learning. For example, when Google's AlphaGo defeated Lee Sedol (a Korean professional Go player), showing the very strong learning ability of deep learning. Furthermore, Google's Deep Dream an amazing software that not only classifies images but also creates strange artificial drawings based on its own knowledge. In addition, Facebook with Deep Text (a text comprehension technique) based on deep learning, can classify a huge amount of data, providing feedback service after identifying user conversations and delete spam messages.

3.3. Deep learning models

3.3.1. Neural Networks

Neural networks or Artificial Neural Networks (ANNs) are inherited from the concept of human biological neurons. By studying the neuron's structure in the human brain. To understand neural networks, one needs to understand how neurons work. A neuron consists of four main parts: dendrites, nucleus, soma and axon as shown in Figure [14] below.

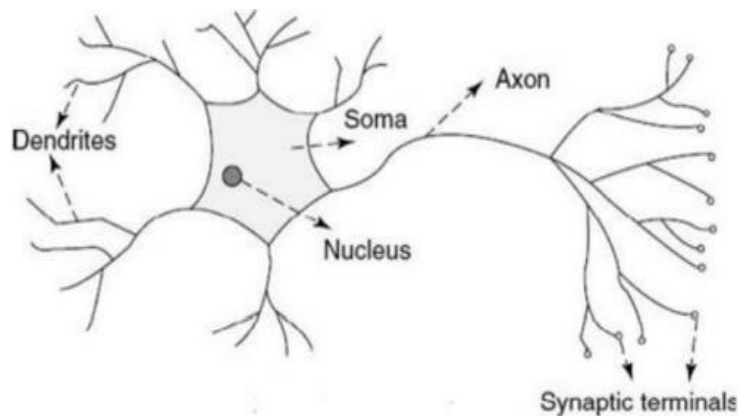


Fig. 14: Biological Neural Network

A neural network's dendrites receive electrical impulses. These electrical impulses are processed by Soma. The process's output is conveyed by the axon to the dendritic gate, where it is transmitted to the next neuron. The nucleus is the neuron's beating heart. Neural networks are the connections within neurons that allow electrical impulses to move throughout the brain.

In machine learning, an artificial neural network functions in a similar way. The input layer takes input data (similar to dendrite), hidden layers process data (similar to soma and axon), and finally, the output layer communicates the computed findings (similar to a dendrite terminal). Figure [15] illustrates an example artificial neural network topology.

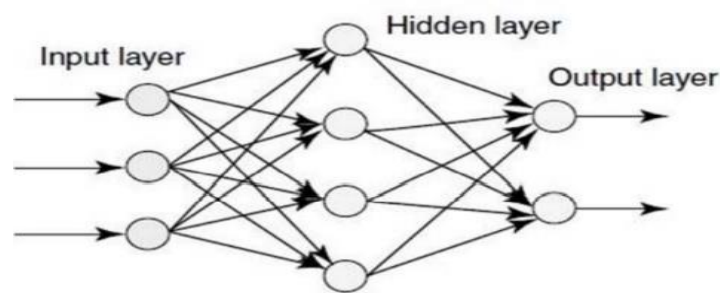


Fig. 15: Artificial Neural Network

According to A. Shrestha and A. Mahmood [29], neural networks can be divided into the following categories:

- Feedforward Neural Network
- Recurrent Neural Network (RNN)
- Radial Basis Function Neural Network)
- Self-organizing feature-mapping neural network
- Modular Neural Network
- Convolutional Neural Network

In this paper I will be using both convolutional neural network (CNN) and forward deep neural network (FDNN). Information flows in just one way out from the input layer heading through the output layer passing by any hidden nodes if exist in the forward deep neural networks. They do not form any loops or inferences. Figure [16] shows a specific implementation of a multilayer differential neural network with forward computed values and functions. Z is the sum of the weights of the inputs, and y represents a nonlinear function (F) of Z in each class. We represent the weights between 2 units in adjacent classes by the subscript while (b) represents the difference value of the unit.

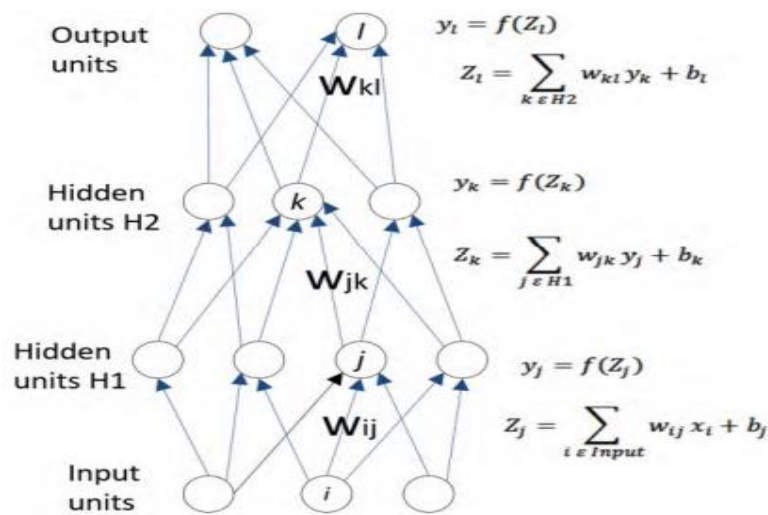


Fig. 16: Feedforward neural network [29]

The convolutional neural network (CNN) is a variation on the regular neural network. Instead of employing completely linked hidden layers as detailed in the prior section, CNN employs a unique network topology that alternates between pooling layers and convolution as shown in the Figure[17] below.

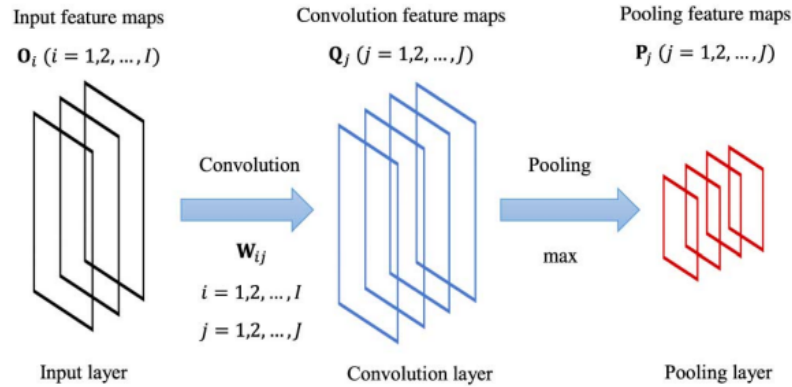


Fig. 17: Convolutional Neural Networks

3.3.2. Activation functions

E. Chigozie et al. [30] define activation functions as functions used in neural networks to calculate the total of input weights and biases, which determine whether or not the neuron is activated. It manipulates the data using a gradient procedure, often lowering the gradient, and then delivers a result for a neural network including data parameters. The activation function, which is utilized to drive the neural network's output, can be linear or non-linear depending on the function it represents. Figure [18] depicts some activation functions, which are expressed in the formulas below.

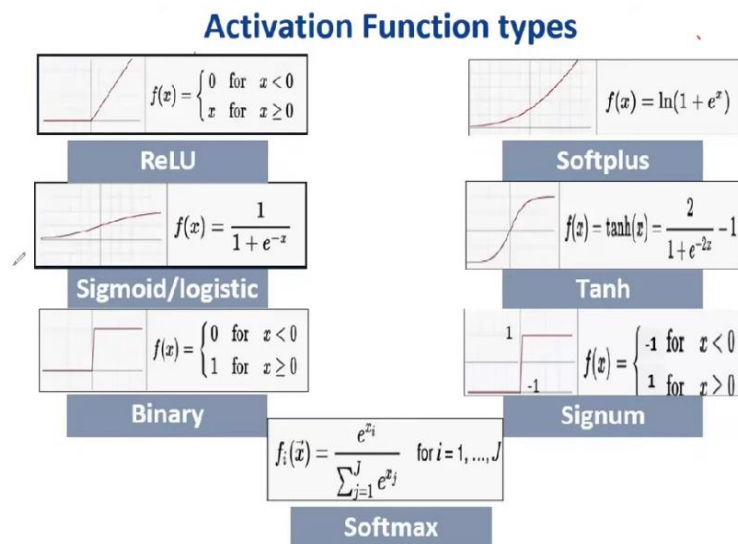


Fig. 18: Activation functions

1. Softmax function:

$$f(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

2. Relu function:

$$f(x) = \max(0, x)$$

- Loss function

The loss function is simply a function used to evaluate how good the algorithm model is. If the prediction is good, the function will give a low value, otherwise the function will give a high value. The formula of the loss function is shown as follows (2.3).

$$Loss = - \sum_{i=1}^{outputsize} y_i \cdot \log \hat{y}_i$$

3.3.3. Tensorflow and Keras Libraries

Tensorflow is currently the most famous deep learning library developed by Google. It was first announced at the end of 2015, while the first stable version came out in 2017.

Tensorflow is an open source library under the Apache license Opensource. People can use, tweak, and redistribute the refined version without paying Google anything. Keras is a source code library under the MIT license. Keras provides a high-level API, on top of deep learning libraries such as Tensorflow, Theano, CNTK with the goal of simplifying access to the deep learning platform. Keras was built with the intention of providing a friendly, modular, and

extensible interface. Google's Tensorflow team chose to support Keras in 2017, which resides directly in the kernel of the tensorflow library.

3.4. Software Defined Networks (SDN)

What is Software Defined Network?

The concept of Software Defined Network (SDN) is primarily set to present the idea and work with OpenFlow at Stanford University (Stanford, CA, USA). According to D.kreutz and colleagues, as the original definition, SDN refers to a network architecture, where the forwarding state is managed by the Data Plane, which is a remote control Plane separately remotely controlled. SDN architecture is defined to have 4 elements.

1. Control Plane and Data Plane are separated. Control features are moved out of network devices, these network devices become simple packet forwarding.
2. The transition choice is made based on the flow rather than the destination. A stream is described as a set of field values that serve as both a filter and a collection of actions. A thread in the context of SDN/Openflow is a sequence of packets between a source and a destination. At the transition device, all packets in a stream are subject to the same service regulations. The flow abstraction provides for the unification of many types of networks of network devices, such as routers, switches, firewalls, and middleboxes. Flow programming provides tremendous freedom, but is restricted to the deployment of flow-tables.
3. The logic control is removed from internal entities known as SDN Controllers or Network Operating Systems (NOS). NOS is a server-based software platform that provides abstract basic resources to aid in the development of transition devices based on an abstract and logical network mode. Its function is comparable to that of a regular operating system.
4. The network may be designed using software programs on the first level of NOS that communicate with Data Plane devices. Cover is the fundamental feature of SDN and is regarded as its primary value.

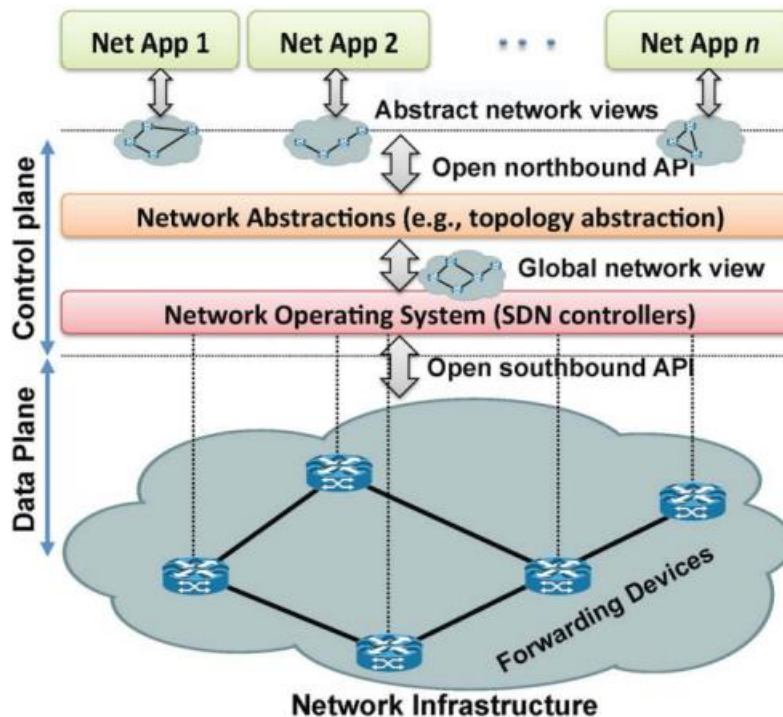


Fig. 19: SDN architecture

3.5. DDoS Datasets

Currently, there are many datasets in the field of information security, most of them deal with the problem of detecting and defending against DoS/DDoS attacks.

M. Ring et al. [31], had a detailed survey of datasets related to information security, in this part I only extract datasets related to DoS/DDoS attacks.

Booters is a denial of service attack service used by hackers against educational institutions. J. J. Santanna et al. [32], published a dataset using 9 types of Booters attacks against computers in their network. The results were logged as packets with up to two-hundred fifty Giga Bytes of network traffic. Single transmitted packets were not identified, however different sorts of assaults were separated into different folders. The dataset is open to the public, but the identities of the assault types have been withheld owing to information privacy.

ISCX 2012 [33], It was created in 2012 by the Canadian Institute for Cyber security (CIC) by capturing packets in a simulated network environment for 1 week. A. Shiravi et al. [33], used dynamic methods to generate datasets with normal traffic as well as malicious traffic. They divided the dataset into two versions. The alpha version which defines attack scenarios and the beta version which defines common user scenarios including emailing, web surfing, etc. Based on this method, the author has created a fairly complete dataset, which contains common types of attacks including DoS and DDoS attacks.

“CIC-IDS-2017” [34] and “CSE-CIC-IDS-2018” [35] were also made by the Canadian Institution for Cybersecurity (CIC). In which, CIC-IDS-2017 was published in 2017. I. Sharafaldin et al. [34], used the same techniques as when building the ISCX 2012 dataset. However, with this new version, the author has added more attack types, more details on how to label attacks as well as introduced the CICFlowMeter network traffic analysis software [36]. With the results of this software, researchers can easily apply both deep learning and machine learning methods without having to preprocess hundreds of GBs of packets information. Moreover, in 2018, the author released the dataset CSE- CIC-IDS-2018 (referred to as CICIDS2018), combined with Amazon Web Services (AWS) cloud service, to be able to create a complex simulation network close to the network’s realistic grid. Since then, the author applies many attack scenarios to make this dataset the most complete and best dataset available today.

DARPA [37]. DARPA is a widely used dataset for network intrusion detection, created by MIT Lincoln Lab network simulation. Many sorts of attacks are included in the dataset, including DoS, buffer overflow, port scan, and rootkits.

KDD CUP 99 [38]. KDD CUP has been developed based on the "DARPA-98" dataset, which has become a popular dataset then. This dataset contains no packet or flow information, but simply contains packets’ indices. It offers TCP connection characteristics, high-level properties such as the series of unsuccessful login attempts, and Twenty additional attributes.

NSL-KDD [39]. NSL-KDD is an upgraded version of KDD CUP 99 with the removal of most of the redundant information. This dataset contains 150,000 data points that have been separated into training and test sets.

Chapter 4: Literature Review

4.1. Classical techniques

Several classical techniques were developed for SYN Flooding attacks detection. Y. Ohsita et al. [40] have created a technique for more effectively recognizing malicious SYN traffic by taking into consideration incoming traffic time fluctuation. Using TCP flags, packets were classified into five groups based on their flow, those that perform the three way handshake, those that are ended by a Reset (RST) packet, and so on. Regardless of traffic time variations, the approach can detect attacks quickly and correctly. Attacks at a lower rate, on the other hand, cannot be identified since traffic with lower rate attacks still follows the normal distribution.

Also Haining Wang et al. [9] had developed another technique by identifying the SYN flooding malicious traffic at routers connected to endpoint devices, rather than observing continuous traffic at the front end or on the target servers. The techniques of detection simplicity comes from its statelessness and minimal computing overhead, which make it impervious to flooding attacks. The protocol behavior of TCP SYN-FIN (RST) pairings is the basis for the

detection technique. When detecting a SYN attack, this approach not only issues warnings, but also records information regarding the location of the source flooding if the attack is detected at the first router.

In LDoS attacks some detection methods have been proposed such as a feature-based defense strategy, where Z. Wu et al. [42] had developed a method for detecting LDoS threats based on sequence matching. As a result, LDoS attack traffic has a high degree of periodicity, a low transmission rate, and a wide spread. They observed that bioinformatics solution in arranging the sequences may be used to detect LDoS malicious traffic by comparing the sequence similarity of scattered LDoS pulses at the targeted server or hot terminal. They presented a method for identifying LDoS threats that make use of the local sequence arrangement method of Smith Waterman, which computes the matching score between two patterns. Experimental outcomes reveals that the proposed detection algorithm performs well in terms of detection performance.

Some methods were proposed in order to try detecting and preventing ICMP flooding DDoS attacks. Some methods also were proposed to traceback the ICMP attack so that we can know the attack source. A. Izaddoost et al. [43] proposed a method in tracing back ICMP flood attacks called (Intention-driven iTrace) where a model was suggested which takes into account incoming traffic directed to the target server, and by adjusting the iTrace model which is intention derived, tracking of ICMP incoming traffic will be more efficient. This method helps in maximizing the number of ICMP traceback messages. The victim's route to the source of the attack may be correctly reconstructed when essential routers detect more effective traceback messages coming from the ICMP flood. We can see that by this method we can more precisely identify the source of the launched attack and the location of the attacker.

O. E. Elejla et al. [44], developed a study that examines and categorizes the detection algorithms of current intrusion detection systems that have been suggested or upgraded to address ICMPv6-based DDoS assaults. O. E. Elejla et al. have clarified that “DoS and DDoS attacks using Internet Control Message Protocol version six (ICMPv6) messages seem to be the most effective techniques against Internet Protocol version six (IPv6). Although Internet Protocol version four (IPv4) intrusion detection systems (IDSs) could work in IPv6, they were unable to manage security concerns such as ICMPv6-based DDoS attacks due to new IPv6 capabilities.” Therefore, they conclude that Intelligent approaches, particularly machine learning, are thought to be in the frontline in this area since they can learn and perhaps develop a successful model based on the attack dataset supplied.

Another method was developed for detecting the ICMP echo requests that causes flooding attacks. H. Harshita [45] had proposed that the bandwidth of the ICMP packet must be limited to a threshold value of 1000 bits/sec, if any ICMP packet exceeds this amount, the router will reject it. Because, as the attacker's bandwidth is larger than that of the target, no attack occurs. This is an efficient technique in order to block the ICMP flooding attack.

Hussain et al. [46] examines the impact of UDP malicious traffic on the performance of several queuing algorithms such as Deficit Round Robin (DRR), Random Early Discard (RED),

Stochastic Fair Queue (SFQ), Fair Queue (FQ) and Droptail (DT). According to the findings of the study, It was concluded that SFQ is more efficient when dealing with UDP malicious traffic than the rest of queuing techniques.

As a result of the examination of the r-continuous bits matching rule, Rui et al. [47] developed Eigenvalue matching. The negative selection process was enhanced in both detector construction and "black hole" detection using the new matching rule. To detect all non self modes, eigenvalue filter windows and detectors are employed. The resulted simulations show that the cybersecurity detection system can identify spoofed IP addresses found in UDP floods and ensure that legitimate users can access the server.

Thapngam et al. [48] presented a behavior-based approach capable of distinguishing DDoS attack traffic and traffic coming from legitimate users. Comparable detection algorithms can extract repeated properties of packet arrivals by employing Pearson's correlation coefficient. The detection accuracy was validated using extensive simulations. By running trials on many datasets results show that the suggested technique can distinguish attack traffic from legal traffic with a speedy response.

Yatagai et al. [49] also offered HTTP-GET flood detection approaches based on page access behaviour analysis. They have presented two detection techniques, one relying on page surfing order and the other on a link between browsing duration and page information size. By examining the detection rates false negative and false positive, the results suggest that the used strategies are effective at detecting the HTTP-GET flood attack.

4.2. Statistical techniques

R. Doriguzzi-Corin et al. [50] stated that measuring the statistical properties of network traffic properties is a common method for DDoS attack detection. Involving continuous observations of variability changes the entropy of the header field in the packets. Entropy, according to this definition, is a measure of the variation or unpredictability in a data collection. An entropy-based DDoS attack detection method was introduced in academic studies in the 2000s, based on the assumption, during a bandwidth DDoS attack, the randomness of the traffic characteristics varies or change suddenly. The reason is typically from a DDoS attacks, usually a large number of compromised devices are used in delivering frequent requests to the target. Thus, these attacks cause either a decrease in the delivery of some traffic attributes, such as the destination IP address, or an increase in the delivery of others, as the source IP address for example. The way a DDoS attack is determined often depends on the mean of the thresholds of these distribution metrics.

L. Feinstein et al. [51] presented a DDoS detection technique based on computing the entropy of the source IP and the Chi-squared distribution. The author observed the change in source IP entropy and the Chi-squared statistic through which the variation of valid traffic is small, compared with the deviation caused by the attack traffic. Similarly, P. Bojovic et al. [52] have combined entropy with bandwidth traffic characteristics to detect bandwidth DDoS attack.

A common limitation of entropy-based techniques is the compulsion to choose an exact detection threshold. Different network systems will have band offsets different throughputs, leading to the challenge of applying an exact threshold that minimizes false positives in different attack scenarios. One solution proposed by P.Kumar et al. [53], is to dynamically assign threshold values that can automatically adapt to normally fluctuating network traffic.

4.3. Intelligent techniques

4.3.1. Machine learning techniques

In [54] M.Wang et al. had proposed their model where the chosen features used to identify the attack were normally chosen by hand based on some personal expertise, and the detection model is anticipated to perform well in detection. Optimal selection of the characteristics that perform the best was the vital difficulty for building an effective detector. Therefore, to demonstrate and solve the proposed challenge, they integrated sequential feature selection with MLP to select the best features during the training stage, and they built a feedback system to dynamically reconstruct the detector when significant detection errors were detected. On the NSL-KDD dataset with MLP, they provide accuracy, detection rate, and FAR of 97.66, 94.88 percent, and 0.62, respectively.

S. Wankhede and D. Kshirsagar [55] proposed a method that is primarily targeted at application layer DoS attacks detection like the HTTP-Get Flood attacks. They classified the CIC IDS 2017 dataset using machine learning methods like multi layer perceptron (MLP) and random forest (RF). The outcomes of RF and MLP were reviewed, and it was found that RF performance exceeds that of MLP.

D. J. Prathyusha and G. Kannayaram [56] developed a model using artificial immune systems to mitigate DDoS attacks in cloud computing by recognising the most likely attack aspects. They used the KDD cup 99 dataset. This technique is able to recognize dangers and responding in accordance with the behavior of humans' biological resistance mechanisms. Based on extensive theoretical and performance study, the suggested system can detect anomalous entries with high detection precision and a low rate false negatives. The model computes the extracted features' probability and entropy. It also computes affection of extracted features using the entropy notion. The system identifies DDoS assaults if the calculated affinity value exceeds the specified alignment value. With an immunological algorithm that employs six possible characteristics, it achieves accuracy, precision, specificity, and sensitivity of 96.56, 95.6, 91.9, and 96.4 percent, respectively.

In 2016, M. Alkasassbeh et al. [57], proposed some machine learning techniques for detecting such DDoS attacks. As there were no shared data sets representing current DDoS attacks at multiple network levels, a new dataset was built to be used. The dataset contained twenty-seven features and five groups. The machine learning techniques used included Naive

Bayes, Multilayer Perceptron and Random Forest and were used to categorize the DDoS assaults. MLP classifier was found to have the greatest accuracy rate of 98.63 percent, followed by RF at 98.02 percent and NB at 96.91 percent.

J. L. Berral et al. [58] extend the framework proposed by Zhang and Parashar (2006) by using machine learning for detecting and blocking DDoS attacks. All nodes in the architecture are capable of understanding on their own and respond to a variety of conditions. To detect high traffic volumes, the widely known cumulated sum algorithm is employed. Classifiers and detectors, such as Naive Bayes, are used to identify patterns of normal traffic. To classify messages, each node has an algorithm which examines the cumulative means sum for each time unit to a specified level. The approach is capable of detecting and preventing distributed denial of service flood threats at an early stage.

Many researchs have been conducted to identify DoS/DDoS assaults using machine learning. Decision Tree, K-means, Linear Regression, Random Forest, Naïve Bayes, SVM (linear, RBF, Polynomial kernel), and Gaussian EM were among the machine learning techniques utilized by Z. He et al. [59]. The Linear SVM method produced the best results, giving a 99.7% accuracy with a very small false negatives rate of 0.07%.

Similarly, R. Doshi et al. [60] have applied some machine learning methods to detect denial of service attack at the source in the network of IoT devices. He has mainly detected IoT devices that are used to become DDoS attack tools. The results showed that the machine learning methods gave a very high accuracy. The Linear SVM model gave a very well accuracy up to 99.1%.

In 2018, A. A. Abdulrahman and M. K. Ibrahim [61], examined the CICIDS 2017 dataset, which comprises benign and DDoS assault network traffic and fulfils certifiable standards and is publicly available. They used four algorithms in order to categorize the attacks which are, C5.0, Random Forest, SVM and Naive-Bayes. They obtain that the classifier with the highest accuracy was Random forest with 86.8 percent. While both the C5.0 and the Random forest had the same precision = 99 percent.

G. Ramadhan et al. [62] develop a TCP malicious DDoS identification system which is a technique of the Artificial Immune System (AIS) depending on the dendritic cell algorithm. One of the unique approaches for detecting DDoS. The system is made up of two major parts, gathering data and data analysis same as human immune systems. DCA is divided into four phases which are, context evaluation, classification, detection as well as preprocessing and initialization. The technology warns of a DDoS attack by sending out warning signals. Warning signals include danger, PAMA, inflammation, and safe. PAMA is considered as a reliable indicator of a malfunction, and varied indications suggest different types of infections.

J. D. Ndibwile et al. [63] presents a basic network design that distinguishes malicious traffic of DDoS from genuine traffic by using a fraudulent web server, a real web server, and a bait server. At the network gateway, the architecture makes use of a personalized Intrusion Prevention System that employs rules developed by deep classification using a tree machine

learning algorithm. A random tree machine learning strategy with defined datasets was employed in order to avoid false positive traffic. The suggested method efficiently classifies traffic as harmful or authentic.

M. Aamir and S. M. A. Zaidi [64] suggested a methodology based on four major phases which are dataset acquisition, feature engineering, ML model evaluation, and findings. The dataset was generated by methodically searching public and confirmed databases for evidence of DDoS attacks. The study represents a good framework that combines feature engineering and machine learning with a specified flow of experimentation. It offers complete solutions that can be relied on to minimize data overfitting and collinearity issues while identifying DDoS assaults. The author emphasized the significance of the procedures prior to the deployment of the ML models and described the precision, false positives, and recall of the various techniques to the defined work framework. The K-nearest neighbor technique resulted the best performance overall compared to the other techniques. The Random Forest model resulted the best performance dealing with datasets with distinct feature types in low dimensions.

C. Wang et al. [65], firstly, a genetic algorithm was employed to analyze the “NSL-KDD” dataset, with the list of features reduced from 41 to 17, as well as the amount of data reduced to about 41 percent of the original dataset. The true positive, false positive, and detection rate of the Random Tree approach were 99.8109 percent, 0.998 percent, and 0.002 percent, respectively.

A. R. Yusof et al. [66], suggested an attack detection model that is trained and evaluated on the NSL-KDD 2009 dataset, and its performance is compared to some standard features selection methods. In order to identify the most important attributes, the NSLKDD dataset was utilized as the attack data and certain attribute selection strategies, such as reliability subset analysis and DDoS distinctive features, were used. The model achieved a 91.7 percent accuracy utilizing 17 reduced features.

IP spoofing is one of the most destructive techniques that attackers use in their attempt of knocking down a target network as they masquerade as genuine addresses. B. S. Kiruthika Devi et al. [67] developed a model which is made of a rate limitation mechanism based on interfaces, online monitoring system and a malicious traffic detection system. Algorithms and automated tools were used by the monitoring system to analyze the effect of the attack. The malicious traffic detection system validates incoming traffic employing a hop count verification algorithm. Combining the hop count verification algorithm with a trained support vector machine model leads to reach to a high accuracy in false positive reduction when comparing to other techniques as decision tree and random forest.

4.3.2. Artificial Intelligence techniques

Due to the relatively small number of samples in the datasets, traditional or classical machine learning models could not analyze the data efficiently. In this regard, Md. Z. Hasan et al. [68] planned to apply a Deep Convolution Neural Network model to successfully locate edge nodes at an early stage. Because shallow machine learning approaches were unable to perform

traffic analysis as expected in the smaller dataset sample, DCNN was investigated. According to the experiment results, DCNN has a higher accuracy of 99 percent than Nave Bayes, support vector machine and K-nearest neighbor, which are considered of the fundamental machine learning techniques with accuracy scores of 79 percent, 88 percent, and 93 percent, respectively.

T. A. Tuan et al. [69] suggested many machine learning algorithms in 2019. They tested machine learning techniques for identifying Botnet DDoS assaults in an experiment. The KDD99 and UNBS-NB 15 exposure datasets are used in the assessment for Botnet DDoS attack detection. When compared to the other approaches, unsupervised learning yields the greatest outcomes. When applied to the UNBS-NB 15 dataset, the obtained accuracy was 94.78 percent, the highest accuracy, and when applied to the KDD99 dataset, the obtained accuracy was 98.08 percent, the best accuracy.

In parallel with the researches applying machine learning, researches applying deep learning is appearing recently, In which a number of works are comparing the results of these two methods to have a better overview.

A. Koay et al. [70] have proposed a method that combines three models such as RNN, MLP and ADT to create an E3ML model. The author trains and tests the models on two datasets, ISCX2012 and DARPA. When comparing the obtained results, even when combining the models, the results of E3ML are still equivalent to those of the RNN deep learning model. This shows the outstanding learning ability of this deep learning model.

M. Zhu et al. [71], described a novel anomaly detection approach based on deep learning models, especially the feedforward neural network and convolutional neural network models. Several experiments using a popular NSL-KDD dataset were used to evaluate the models' performance. In defining anomaly categories and detecting network anomalies, FNN and CNN models were shown to be more accurate than J48, Random Tree, Naïve Bayes, Random Forest, and SVM, which are shallow machine learning techniques.

L. Fernandez Maimo et al. [72], proposed a unique 5G cyber protection architecture for efficiently and swiftly identifying cybercrime in 5G wireless services. Deep learning algorithms were used in their design to evaluate network traffic by extracting characteristics from routing information. The used model was LSTM Recurrent Network that has been trained to distinguish temporal variations of intrusions. According to the testing results, each system or machine in the 5G network will be capable to use one architecture or another depending on parameters such as physical resource availability, CPU and GPU.

R. Karimazad and A. Faraahi [73] have presented a DDoS attack detection approach depending on different properties and features of attack packets. This was done through studying and analyzing the incoming network traffic by using Radial Basis Function (RBF) which is a non linear Artificial Neural Network (ANN) that consists of a two layer hidden neurons. On a simulated network and Dataset the proposed algorithm was constructed and tested. The dataset and received packets from the simulated network were then employed in training the model. The detection accuracy was found to approximately 96 percent for the simulated network

while approximately 98 percent for the dataset. The proposed technique can detect the DDoS attacks with a very high detection rates as demonstrated.

In DDoS the used zombies are programmed to send various types of packets to the victim target or network. These victim networks are wirelessly controlled, either by self implanted trojans or by the attacker. For this purpose, A. Saied et al. [74] developed a technique using artificial neural networks algorithm for identifying and neutralizing known and new distributed denial of service attacks in real time scenarios. The threats were detected by identifying the following necessary information as the sequential number of packets, IP address of destination , IP address of source, length of the packet, packets sequential number, destination port and others in order to identify samples of DDoS and normal genuine traffic from users. They used these parameters to train their proposed ANN algorithm. The model was designed to detect ICMP, TCP and UDP DDoS attacks. The proposed model when evaluated, it shows an accuracy up to 98 percent which is an acceptable accuracy when compared to other models. The only disadvantage was that the model was not able to identify the particular type of the DDoS attack.

D. Perakovi et al. [75] developed an artificial neural network (ANN)-based detection and classification model system. The constructed model using artificial neural networks was used to identify the attacks by classifying the incoming traffic into four categories, UDP malicious traffic, normal traffic flow , DNS DDoS malicious traffic, and chargen DDoS malicious traffic. DDoS detection criteria includes packet length, source IP address, destination Address, and protocol. Because of the similarity in characteristics between both the UDP attacks and genuine traffic, the efficiency of identifying and classifying UDP malicious traffic is somewhat lower. Machine learning-based classifiers like artificial neural networks are professionals at detecting patterns in datasets by utilizing features used to characterize the data. The generated model's simulation results demonstrated an accuracy of 95.6 percent in the classification of pre-defined traffic classes.

X. Yuan et al. [76], combined CNN and RNN and proposed what they called the Deep Defense method. They used a sliding-window technique to process raw data from the ISCX2012 dataset to convert from packet-based to window-based. The author divided the dataset into a large part and a small part. The large part contained most of the packets and they were recorded on day 15, while the small part were packets recorded on day 14. As reported, the 3LSTM model achieved the highest accuracy with the large dataset with 98.41%, while that GRU achieved the highest accuracy with 98.417% with the remaining dataset.

S. S. Roy et al. [77], investigated the potential of Deep Neural Network as a classifier for various forms of intrusion assaults. A comparison study with Support Vector Machine was also conducted . The results of the experiments reveal that the accuracy of intrusion detection using Deep Neural Network is excellent. The training frame has been assigned to 75% of the data set, while the validation frame has been assigned to 25%. The deep neural network had an accuracy of 99.9944 percent, whereas the Support vector machine had an accuracy of 84.635 percent.

Z. Jadidi et al. [78], admitted that the possibility of new threats in the future adds more difficulty to the intrusion detection. They have also reached that establishing an approach for identifying the anomaly based cyberattacks is an essential thing. They presented an attack detection method using the artificial neural networks . A Perceptron neural network with many layers and one dense layer was used. They looked how using the gravitational search algorithm may help optimizing the connectivity weights of the designed multi layer perceptron network. The proposed model was found to have 99.43 percent accuracy in distinguishing between legitimate and malignant traffic. The gravitational search algorithm performance was compared to that of classic gradient descent training techniques and a particle swarm optimization approach. The results suggested that the gravitational anomaly based detection proposed system is successful at detecting flow based anomalies.

In 2020 S. Sumathi and N. Karthikeyan [79], examined the network's performance for a publicly available dataset, they have employed a deep neural network (DNN) design with a cost reduction approach. The suggested method makes use of the following datasets, DARPA 1999, CINFICKER, KDD CUP, and DARPA 2000. The simulation results show that the deep neural network cost minimization method outperforms the existing approach in terms of 99 percent detection accuracy with high throughput, very little false reduction, reduced overhead, less packet drop, high data rate, less false reduction and high average latency.

M. S. Elsayed et al. [80] offered DDoSNet which is an intrusion detection solution for DDoS attacks in SDNs. The approach is based on the Deep Learning. They have designed a model which integrates the recurrent neural network (RNN) model with an autoencoder. They had tested their model on the newly published dataset CICDDoS2019. The data preprocessing phase was essential where some network properties were not evaluated for model training. To generate a binary classification model, the training step was then carried out by partitioning the dataset according to the attack type and training the model in this manner. The model consists of an input layer that takes the features as an input, a hidden layer of the RNN and an autoencoder, and an output layer where the binary classification is done. When compared the model to other existing approaches, they found a considerable improvement in the attack detection.

C. Yin et al. [81], compared the results of the training done on the NSL-KDD dataset on the recurrent neural networks (RNN) deep learning model with the machine learning models J48, ANN, support vector machine and random forest. The analysis revealed that RNN has higher accuracy than the classical machine learning methods.

E. Min et al. [82], combined random forest (RF) and convolutional neural networks (CNN) algorithms to be used in training their model which is called TR-IDS on the ISCX2012 dataset and they compare the results with other individual machine learning methods such as SVM, NN, CNN, RF , the results show that their combined model gives better results with high accuracy up to 99.13%.

K. Wu et al. [83], introduced an intrusion detection system using CNNs for multiclass classification. They train their model on the NSL-KDD dataset, where the dataset is encoded as an 11x11 array. The final results was compared with the RNN model, which has higher accuracy

when classifying DoS attacks, but the exported model is 20 times more complex than the RNN model.

R. Doriguzzi-Corin et al. [50], used a CNN model with a focus on Conv1D and named it LUCID. The authors aim to build a compact model that can be deployed on hardware-constrained devices. They trained the model on a series of datasets such as ISCX2012, CICIDS2017 and CICIDS2018. Here, the authors have collected only 11 features on each packet for 100 packets in 100 seconds. By exploiting the capabilities of Conv1D, Relu, and Max pooling, according to the report, the author has achieved very good results when compared with other studies on the respective datasets. There, with the combination of training on all three datasets mentioned above, they achieved an accuracy of up to 99.50%, with a level of 40 times more compact than the DeepDefense model of [76].

R. Basnet et al. [84], have developed a technique using deep neural networks to the dataset CSE-CIC-IDS2018 to classify many types of attacks. The authors employed two dense hidden layers, the first is having a number of units or neurons equal to that of dataset characteristics while the second with 128 neurons. They employed the n-fold cross validation approach instead of testing by dividing the dataset into a training and testing portions. According to the report, with this model, when classifying DoS/DDoS attacks, the results are always greater than 99%.

Chapter 5: Dataset: CIC-DDoS2019

I. Sharafaldin et al. [85], created a new dataset, called CICDDoS2019. This dataset addresses all present inadequacies. They have also offered a family classification procedure based on a collection of characteristics of the network flow, which they have realized while utilizing the produced dataset. Finally, they presented the most significant feature sets for detecting various forms of DDoS assaults together with their weights.

As mentioned by The Canadian Institute for Cybersecurity [86] “The dataset is completely labelled, with 80 network traffic characteristics gathered and calculated for all benign and denial of service attack flows. This was accomplished with the use of the CICFlowMeter software”.

They have investigated unique attacks that may be conducted out at the application layer using TCP and UDP protocols and proposed a new classification. As illustrated in Figure [20], they classified the assaults into two types: reflection based and exploitation based attacks.

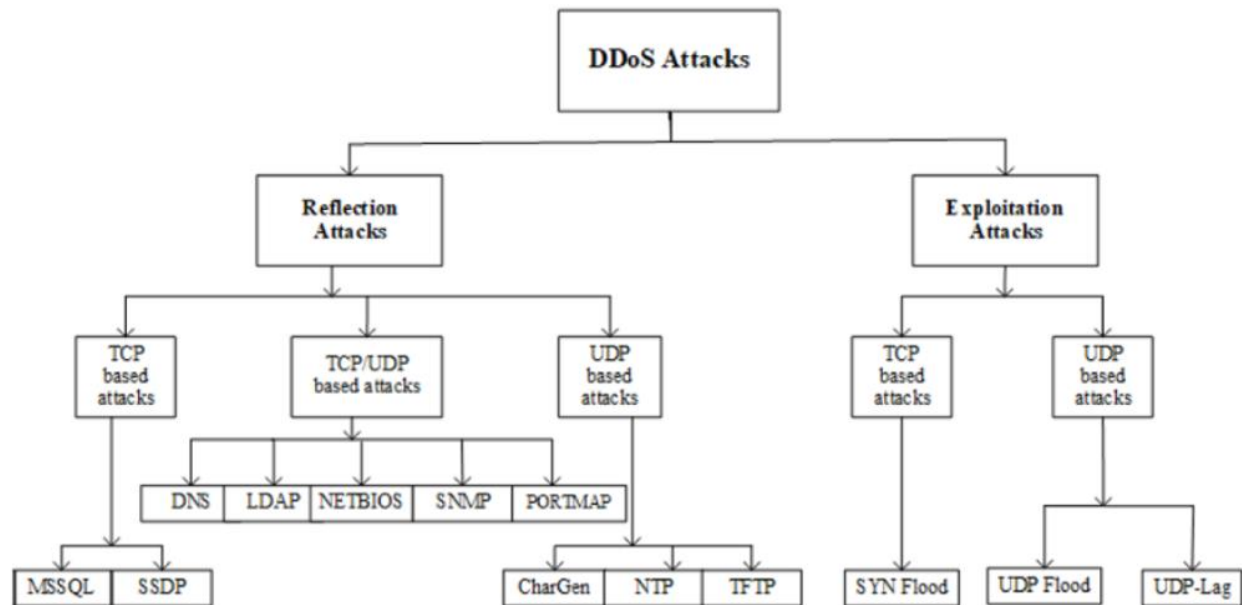


Fig. 20: CIC-DDoS2019 dataset attacks[86]

5.1. Reflection based DDoS attacks

The reflection based DDoS threats as stated by I. Sharafaldin et al. [85] “are considered as a mask of the attacker's identity through the use of a legitimate third party. Attackers transmit data to reflector nodes with the source Ip configured to the target victim's IP domain in order to overwhelm the target with response packets.”

As mentioned by the CIC “These attacks can be carried out by combining application layer protocols with transport layer protocols such as the Transmission control protocol (TCP) or the User datagram protocol (UDP)” , or by combining both.

5.2. Exploitation based DDoS attacks

I. Sharafaldin et al. [85] have stated that “Exploitation based DDoS attacks are also attacks in which the attacker's identity is concealed through the use of a genuine third party. Attackers also send packets to reflector servers with the source IP address configured to the target victim's IP address same as done in the Reflection based DDoS attacks, in order to flood the target with response packets. These attacks can also be conducted using the application layer protocols in conjunction with transport layer protocols.”

As previously stated by the CIC “The UDP-Lag attack is one that breaks the connection between the client and the server. This assault is commonly utilized in online gaming when players try to slow down or disrupt the movement of other players in order to overpower them.” There are two methods for carrying out this attack either, by using a software program that runs on the network and swallows the bandwidth of many other users or by using a physical switch termed as a lag switch.

In this paper, I will only be using NetBIOS, LDAP, PORTMAP and MSSQL from the reflection based attacks, and all the “exploitation based” attacks which are UDP, SYN and UDP-Lag.

Chapter 6: Data Preprocessing

Preprocessing is a process where raw data is converted into an adequate format of data by refilling incomplete data and deleting irregular, noisy, or redundant data. Several procedures are done on the data to minimize dimensionality and preserve consistency for simplicity of processing. The following stages were engaged in the data pre-processing stage.

First of all, I have downloaded the CIC-DDoS2019 dataset from the Canadian Institute for Cybersecurity (CIC) website. After that I uploaded the dataset on my Google Drive so that I can easily use it on Colab.

6.1. Managing Null Entries

It has been observed that data redundancy can have a major influence on study outcomes. Although there are numerous strategies for filling an incomplete data frame, it is better to drop that row containing the null data because this could lead to distortion in the algorithms' parameter outcomes. In this research paper, the dataset was inspected for information loss, and the associated rows were eliminated.

6.2. Handling Categorical Data

Categorical variables are variables that reflect data that may be separated into groups. A software cannot examine categorical data unless it is converted to an integer value. This dataset had three categorical characteristics: timestamp, label, and IP address. Each of these columns was examined individually to preserve the data it presented. The machine learning techniques can only deal with float data or integers and that's why, dotted IP addresses could not be used for categorization. It must be converted to an integer representation in either binary or decimal form.

G. Usha et al.[87] have clarified that the timestamp “provides the date and time of the attack, it was an important component to investigate. New columns were developed to record each portion of the timestamp in order to transform it to a numerical representation. The output label included eight separate properties, seven of which signified different forms of assaults and one which represented a regular connection. The one-hot-encoding approach was used to transform these numbers to multi column binary representation.” However, after the time stamp function has been encoded, it will be deleted due to its lack of diversity.

6.3. Standard Deviation

It is a data diffusion parameter, which is essential to verify expansion for each column as variance with value near to zero indicates that the amount is virtually unchanged all over the dataset and it increases the complexity and thus, computation cost. As a result, it is necessary for any feature column with deviation close to zero to be removed.

Forward & backward average bytes/bulk, flag counts, URG flags ,PSH flags, Time stamp, and URG flags after encoding were all columns with extremely near to zero standard deviation in the dataset utilized in this paper.

6.4. Examining Correlation

The level to which two or more independent variables are linearly connected is referred to as correlation. This is an effective method for decreasing data dimensionality by deleting characteristics that have little relevance with the output label. The seaborn library in Python is used to generate a correlation heat map, which displays the correlation coefficient as well as a color coding that shows the strength of the relationship between all elements of the dataset as shown in Figure [21]below. Features with a nonzero positive integer correlation with the resulted label are thus desirable for identification.

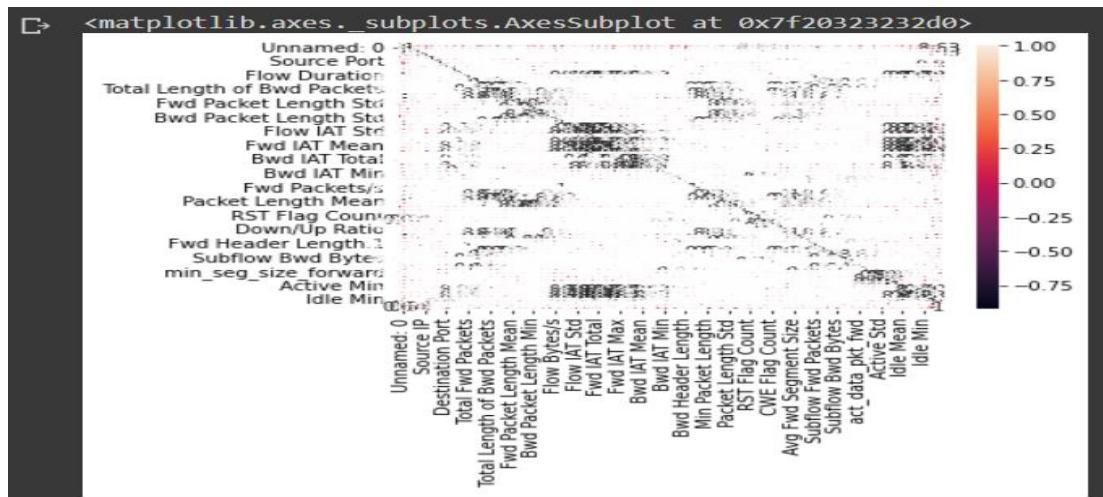


Fig. 21:Correlation Heat Map

6.5. Dataset Standardization

As mentioned by G. Usha et al. [87] Standardization is “The process of changing data so that the mean becomes 0 and the variance becomes 1 is known as standardization. It is designed to bring the magnitude of all selected characteristics within the same range, reducing computation time and preventing the system from being misled into believing a feature is more relevant just because it has higher scale values”.

6.6. Managing Outliers

The noisy data or outliers are considered as data points that behave differently to the dataset's structure or distribution rules. Handling these noisy data points necessitates proper recognition of such samples of data by analyzing data dissemination and detecting locations that deviate from the dataset's tendency. Outliers is an obstacle that faces the learning of any algorithm used by machine learning and therefore, make it harder for the system to generalize, resulting in inaccurate outputs. As a result, reducing the outliers is a critical stage in data cleansing.

There are several ways for detecting outliers in a dataset, including visualization methods such as box plot and scatter plot analysis, as well as statistical tools such as the interquartile range approach.

Visualization techniques are particularly simple since data points that do not follow the main pattern of the data may be identified, nevertheless, these approaches cannot be employed when the dataset is excessively vast. As a result of the big dataset, the IQR approach was utilized to identify outliers. The interquartile range for the dataset is computed using this approach by deducting the 75th and 25th percentiles.

- Interquartile Range (iqr) = $q_7 - q_5$

This range is being used to compute both lower and upper boundaries for data. An outlier is any sample of data that goes far from the specified range. The upper and lower limit ranges are statistically defined as values in the Gaussian distribution that are three standard deviations apart from the mean. Data falls outside of the calculated range is termed to be an outlier and is viewed to be an exception to the distribution.

- Upperbound = $q_5 - 1.5 * iqr$

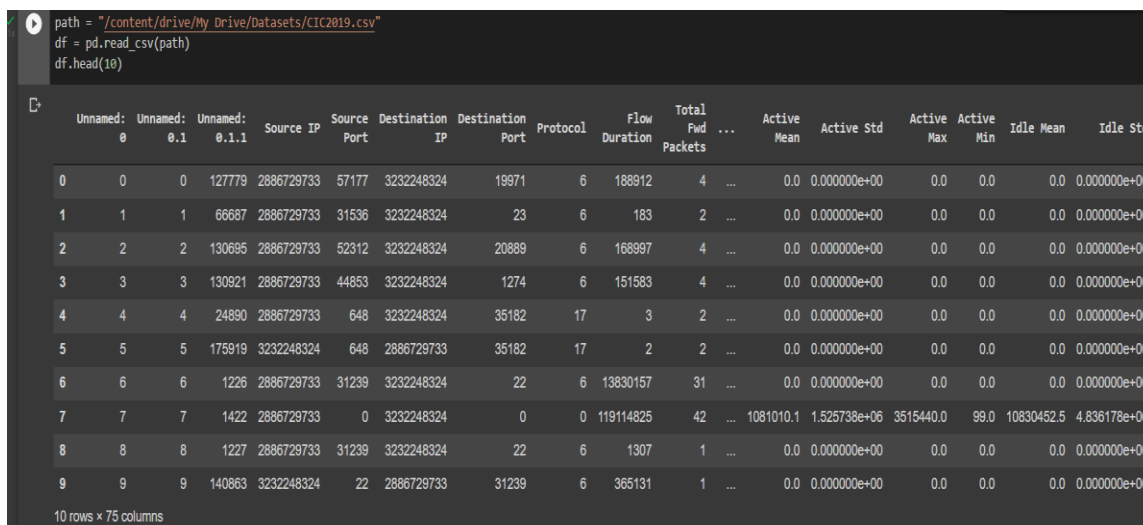
- Lowerbound = $q_7 - 1.5 * iqr$

6.7. Splitting the dataset into Test and Train

In this paper, the I've divided the dataset into two portions in a 75:25 ratio. The larger partition is employed in training the algorithms, while the smaller one is considered to be the validation data which is utilized to test the previously trained algorithm's performance on unknown data using multiple performance metrics.

6.8. Data preprocessing Colab Simulations

I started by mounting the google drive to the Colab in order to read the dataset and then viewing it.



```
path = "/content/drive/My Drive/Datasets/CIC2019.csv"
df = pd.read_csv(path)
df.head(10)
```

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Source IP	Source Port	Destination IP	Destination Port	Protocol	Flow Duration	Total Fwd Packets	...	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std
0	0	0	127779	2886729733	57177	3232248324	19971	6	188912	4	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
1	1	1	66687	2886729733	31536	3232248324	23	6	183	2	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
2	2	2	130695	2886729733	52312	3232248324	20889	6	168997	4	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
3	3	3	130921	2886729733	44853	3232248324	1274	6	151583	4	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
4	4	4	24890	2886729733	648	3232248324	35182	17	3	2	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
5	5	5	175919	3232248324	648	2886729733	35182	17	2	2	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
6	6	6	1226	2886729733	31239	3232248324	22	6	13830157	31	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
7	7	7	1422	2886729733	0	3232248324	0	0	119114825	42	...	1081010.1	1.525738e+06	3515440.0	99.0	10830452.5	4.836178e+06
8	8	8	1227	2886729733	31239	3232248324	22	6	1307	1	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00
9	9	9	140863	3232248324	22	2886729733	31239	6	365131	1	...	0.0	0.000000e+00	0.0	0.0	0.0	0.000000e+00

10 rows x 17 columns

Fig. 22: Reading and viewing the dataset from drive

When reviewing the Dataset dimensions, I found that the dataset has 1,245,798 rows and 75 columns, which means the dataset has 1,245,798 samples and 75 features.

```

▶ print("This Dataset has {} rows and {} columns".format(df.shape[0], df.shape[1])) #Starting data preprocessing
  #Data domensions
└─ This Dataset has 1245798 rows and 75 columns

```

Fig. 23: Dataset dimensions

- The dataset columns features are shown below.

```

▶ df.columns
└─ Index(['Unnamed: 0', 'Unnamed: 0.1', 'Unnamed: 0.1.1', 'Source IP',
        'Source Port', 'Destination IP', 'Destination Port', 'Protocol',
        'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets',
        'Total Length of Fwd Packets', 'Total Length of Bwd Packets',
        'Fwd Packet Length Max', 'Fwd Packet Length Min',
        'Fwd Packet Length Mean', 'Fwd Packet Length Std',
        'Bwd Packet Length Max', 'Bwd Packet Length Min',
        'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
        'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
        'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
        'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean',
        'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags',
        'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s',
        'Bwd Packets/s', 'Min Packet Length', 'Max Packet Length',
        'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance',
        'SYN Flag Count', 'RST Flag Count', 'ACK Flag Count',
        'URG Flag Count', 'CWE Flag Count', 'Down/Up Ratio',
        'Average Packet Size', 'Avg Fwd Segment Size',
        'Avg Bwd Segment Size', 'Fwd Header Length.1', 'Subflow Fwd Packets',
        'Subflow Fwd Bytes', 'Subflow Bwd Packets', 'Subflow Bwd Bytes',
        'Init_win_bytes_forward', 'Init_win_bytes_backward',
        'act_data_pkt_fwd', 'min_seg_size_forward', 'Active Mean',
        'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std',
        'Idle Max', 'Idle Min', 'Inbound', 'Label'],
        dtype='object')

```

Fig. 24: Dataset features

- Some descriptive statistical analysis for each feature

Denial of Service (DoS) Attack Detection Using Machine Learning

```
[31] df.describe() #Descriptive statistics of dataset
```

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Source IP	Source Port	Destination IP	Destination Port	Protocol	Flow Duration	Total Fwd Packets	...	Active Mean	Active Std	Active I
count	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	1.245798e+06	...	1.245798e+06	1.245798e+06	1.245798e+06
mean	6.228985e+05	6.228985e+05	1.074318e+05	2.891303e+09	2.302203e+04	3.166084e+09	3.186662e+04	1.486217e+01	2.202223e+06	3.375339e+00	...	1.085408e+04	9.885855e+03	2.130414e+04
std	3.596310e+05	3.596310e+05	1.089688e+05	1.506559e+08	2.447340e+04	4.198623e+08	1.958453e+04	4.363049e+00	1.051741e+07	1.920314e+02	...	1.717704e+05	1.473230e+05	2.918235e+05
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.724045e+07	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	...	0.000000e+00	0.000000e+00	0.000000e+00
25%	3.114492e+05	3.114492e+05	2.503600e+04	2.886730e+09	7.690000e+02	3.232248e+09	1.479500e+04	1.700000e+01	1.000000e+00	2.000000e+00	...	0.000000e+00	0.000000e+00	0.000000e+00
50%	6.228985e+05	6.228985e+05	7.408800e+04	2.886730e+09	9.447000e+03	3.232248e+09	3.190300e+04	1.700000e+01	1.000000e+00	2.000000e+00	...	0.000000e+00	0.000000e+00	0.000000e+00
75%	9.343478e+05	9.343478e+05	1.462510e+05	2.886730e+09	4.755075e+04	3.232248e+09	4.894600e+04	1.700000e+01	4.800000e+01	2.000000e+00	...	0.000000e+00	0.000000e+00	0.000000e+00
max	1.245797e+06	1.245797e+06	6.652080e+05	3.752141e+09	6.553400e+04	4.294967e+09	6.553500e+04	1.700000e+01	1.199975e+08	8.686600e+04	...	4.050800e+07	2.135244e+07	4.553668e+07

8 rows x 75 columns

Fig. 25: Statistical analysis

After removing the (Unnamed) feature columns and the 0.0 correlation value column features, when reviewing the new dataset dimensions, it became 1,245,798 rows and 37 columns.

```
df.columns
```

```
Index(['Source Port', 'Destination IP', 'Destination Port', 'Protocol',
      'Total Length of Fwd Packets', 'Fwd Packet Length Max',
      'Fwd Packet Length Min', 'Fwd Packet Length Mean',
      'Bwd Packet Length Max', 'Bwd Packet Length Min',
      'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
      'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Fwd IAT Mean',
      'Fwd IAT Std', 'Fwd PSH Flags', 'Fwd Packets/s', 'Min Packet Length',
      'Max Packet Length', 'Packet Length Mean', 'Packet Length Variance',
      'RST Flag Count', 'ACK Flag Count', 'URG Flag Count',
      'CWE Flag Count', 'Average Packet Size', 'Avg Fwd Segment Size',
      'Avg Bwd Segment Size', 'Subflow Fwd Bytes', 'Init_win_bytes_forward',
      'Init_win_bytes_backward', 'Idle Std', 'Inbound', 'Label'],
      dtype='object')
```

```
print("This Dataset has {} rows and {} columns".format(df.shape[0], df.shape[1])) #Starting data preprocessing
#Data domensions
```

This Dataset has 1245798 rows and 37 columns

Fig. 26: Updated dataset dimensions

Renaming the Label feature column with the real attacks names instead of their indices, by replacing '0' with 'BENIGN', '1' with 'NETBIOS', '2' with 'LDAP', '3' with 'MSSQL', '4' with 'Portmap', '5' with 'Syn', '6' with 'UDP' and '7' with 'UDPLag'.

```
#df.columns
df.Label.unique()
df['Label'] = df['Label'].replace('0', 'BENIGN')
df['Label'] = df['Label'].replace('1', 'NetBIOS')
df['Label'] = df['Label'].replace('2', 'LDAP')
df['Label'] = df['Label'].replace('3', 'MSSQL')
df['Label'] = df['Label'].replace('4', 'Portmap')
df['Label'] = df['Label'].replace('5', 'Syn')
df['Label'] = df['Label'].replace('6', 'UDP')
df['Label'] = df['Label'].replace('7', 'UDPLag')
df['Label'] = df['Label'].astype('str')
df.Label.unique()

array(['LDAP', 'NetBIOS', 'MSSQL', 'UDP', 'BENIGN', 'Portmap', 'UDPLag',
       'Syn'], dtype=object)
```

Fig. 27 : Labeled attacks types

- Calculating the number of samples and the percentage of each type of attacks

```
df.Label.value_counts()

LDAP      200000
NetBIOS    200000
MSSQL      200000
UDP        200000
Syn        200000
Portmap    186960
BENIGN     56965
UDPLag     1873
Name: Label, dtype: int64
```

Fig. 28: No. of samples of each attack

```

BENIGN = df[df['Label'] == 'BENIGN']
NetBIOS = df[df['Label'] == 'NetBIOS']
MSSQL=df[df['Label'] == 'MSSQL']
UDP=df[df['Label'] == 'UDP']
LDAP=df[df['Label'] == 'LDAP']
Syn=df[df['Label'] == 'Syn']
Portmap=df[df['Label'] == 'Portmap']
UDPLag=df[df['Label'] == 'UDPLag']
print('Number of BENIGN:',round((len(BENIGN)/df.shape[0])*100,2),'%')
print('Number of NetBIOS:',round((len(NetBIOS)/df.shape[0])*100,2),'%')
print('Number of UDP:',round((len(UDP)/df.shape[0])*100,2),'%')
print('Number of UDPLag:',round((len(UDPLag)/df.shape[0])*100,2),'%')
print('Number of Portmap:',round((len(Portmap)/df.shape[0])*100,2),'%')
print('Number of Syn:',round((len(Syn)/df.shape[0])*100,2),'%')
print('Number of MSSQL:',round((len(MSSQL)/df.shape[0])*100,2),'%')
print('Number of LDAP:',round((len(LDAP)/df.shape[0])*100,2),'%')

```

```

➞ Number of BENIGN: 4.57 %
Number of NetBIOS: 16.05 %
Number of UDP: 16.05 %
Number of UDPLag: 0.15 %
Number of Portmap: 15.01 %
Number of Syn: 16.05 %
Number of MSSQL: 16.05 %
Number of LDAP: 16.05 %

```

Fig. 29: Percentage of each attack

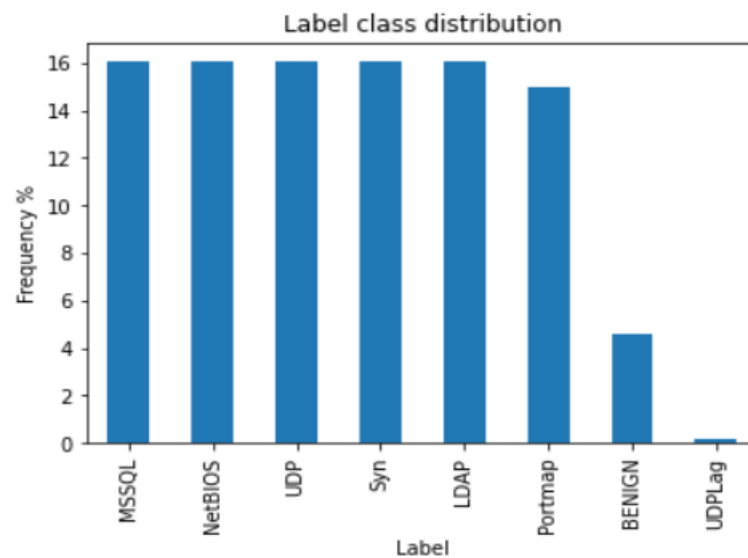


Fig. 30: Label class distribution

Denial of Service (DoS) Attack Detection Using Machine Learning

- Normalizing the data, but before normalizing the data I replaced the types of attacks with their indices in the Label column feature, as we can not normalize string data types.

```
[28] #Normalizing the data
normalized_df=(df-df.mean())/df.std()
normalized_df=normalized_df.drop('Flow Packets/s',axis=1)
normalized_df=normalized_df.drop('Flow Bytes/s',axis=1)

normalized_df.head()
```

	Source Port	Destination IP	Destination Port	Protocol	Total Length of Fwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Bwd Packet Length Max	Bwd Packet Length Min	CWE Flag Count	Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size	Subflow Fwd Bytes	Init_win_bytes_forwa	
0	1.395596	0.157586	-0.607399	-2.031188	-0.725421	-0.954992	-0.935156	-0.945029	-0.050579	0.342849	...	-0.101473	-0.933777	-0.945029	0.015085	-0.725421	-0.0545
1	0.347887	0.157586	-1.625958	-2.031188	-0.733744	-0.954992	-0.935156	-0.945029	-0.050579	0.342849	...	-0.101473	-0.933085	-0.945029	0.015085	-0.733744	7.0276
2	1.196809	0.157586	-0.560525	-2.031188	-0.725421	-0.954992	-0.935156	-0.945029	-0.050579	0.342849	...	-0.101473	-0.933777	-0.945029	0.015085	-0.725421	-0.0545
3	0.892029	0.157586	-1.562081	-2.031188	-0.725421	-0.954992	-0.935156	-0.945029	-0.050579	0.342849	...	-0.101473	-0.933777	-0.945029	0.015085	-0.725421	-0.0545
4	-0.914218	0.157586	0.169285	0.489985	-0.424394	-0.494002	-0.472024	-0.480570	-0.085259	-0.138378	...	-0.101473	-0.467788	-0.480570	-0.095478	-0.424394	-0.3622

5 rows x 35 columns

Fig. 31: Normalized data

- Fitting unique values from each column of the normalized features

```
#Fitting unique values from each column of features
print(normalized_df.apply(lambda col: col.unique()))
```

Source Port	[1.3955958101410246, 0.34788678383352983, 1.19...
Destination IP	[0.15758556942075008, -0.6653475895920182, 0.1...
Destination Port	[-0.6073991496353963, -1.625958413952753, -0.5...
Protocol	[-2.0311882463058, 0.4899845510661981, -3.4063...
Total Length of Fwd Packets	[-0.7254207907132191, -0.7337441094492392, -0....
Fwd Packet Length Max	[-0.9549922605226313, -0.49400173616032994, 0....
Fwd Packet Length Min	[-0.9351562527047765, -0.47202435946987037, -0...
Fwd Packet Length Mean	[-0.9450285741767762, -0.4805695650824725, -0....
Bwd Packet Length Max	[-0.0505794780308098, -0.08525948336493036, 5....
Bwd Packet Length Min	[0.3428492031899198, -0.13837830265633044, 10....
Bwd Packet Length Mean	[0.015085322357032947, -0.09547845799245676, 2...
Bwd Packet Length Std	[-0.07687141363393221, 5.686272903449474, 4.89...
Flow IAT Mean	[-0.17876127436483283, -0.21615052464352935, -...
Flow IAT Std	[-0.17544482177276566, -0.22287632512464317, -...
Fwd IAT Mean	[-0.1783296566860204, -0.2270058781120439, -0....

Fig. 32: Unique values

We have now 36 numerical feature columns and only one categorical feature column.

```
#The dataset Numerical Features
numerical_features = [feature for feature in df.columns if df[feature].dtypes != 'O']
print("The number of numerical features is",len(numerical_features),"and they are : \n",numerical_features)

The number of numerical features is 36 and they are :
[' Source Port', ' Destination IP', ' Destination Port', ' Protocol', 'Total Length of Fwd Packets', ' Fwd Packet Length Max', ' Fwd
```

Fig.33: Numerical Features

```
#Categorical Features
categorical_features = [feature for feature in df.columns if df[feature].dtypes == 'O']
print("The number of categorical features is",len(categorical_features),"and they are : \n",categorical_features)

The number of categorical features is 1 and they are :
['Label']
```

Fig. 34: Categorical Features

We have found 7 discrete numerical features, which are ‘Protocol’, ‘Fwd PSH Flags’, ‘RST Flag Count’, ‘ACK Flag Count’, ‘URG Flag Count’, ‘CWE Flag Count’ and ‘Inbound’.

```
df[discrete_feature].head(10)
```

	Protocol	Fwd PSH Flags	RST Flag Count	ACK Flag Count	URG Flag Count	CWE Flag Count	Inbound
0	6	0	0	1	0	0	1
1	6	0	0	1	0	0	1
2	6	0	0	1	0	0	1
3	6	0	0	1	0	0	1
4	17	0	0	0	0	0	1
5	17	0	0	0	0	0	0
6	6	0	0	1	0	0	1
7	0	0	0	0	0	0	1
8	6	1	1	0	1	1	1
9	6	0	0	0	1	1	0

Fig. 35: Discrete Numerical Features

Denial of Service (DoS) Attack Detection Using Machine Learning

- Analyzing the categorical data by creating a histogram

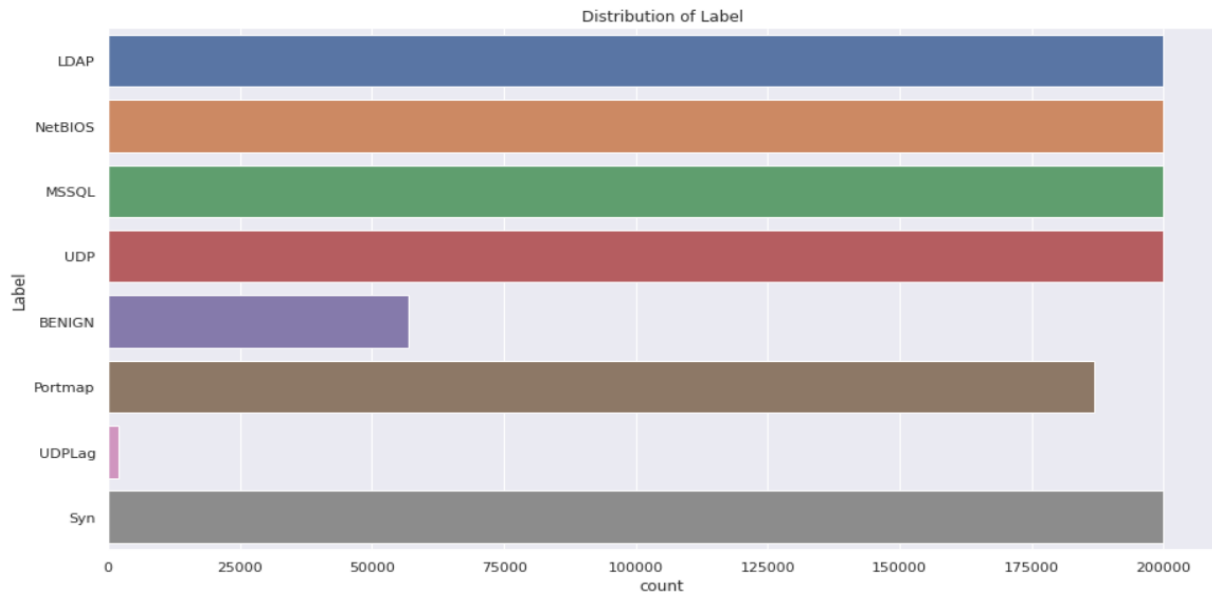


Fig. 36: Categorical data histogram

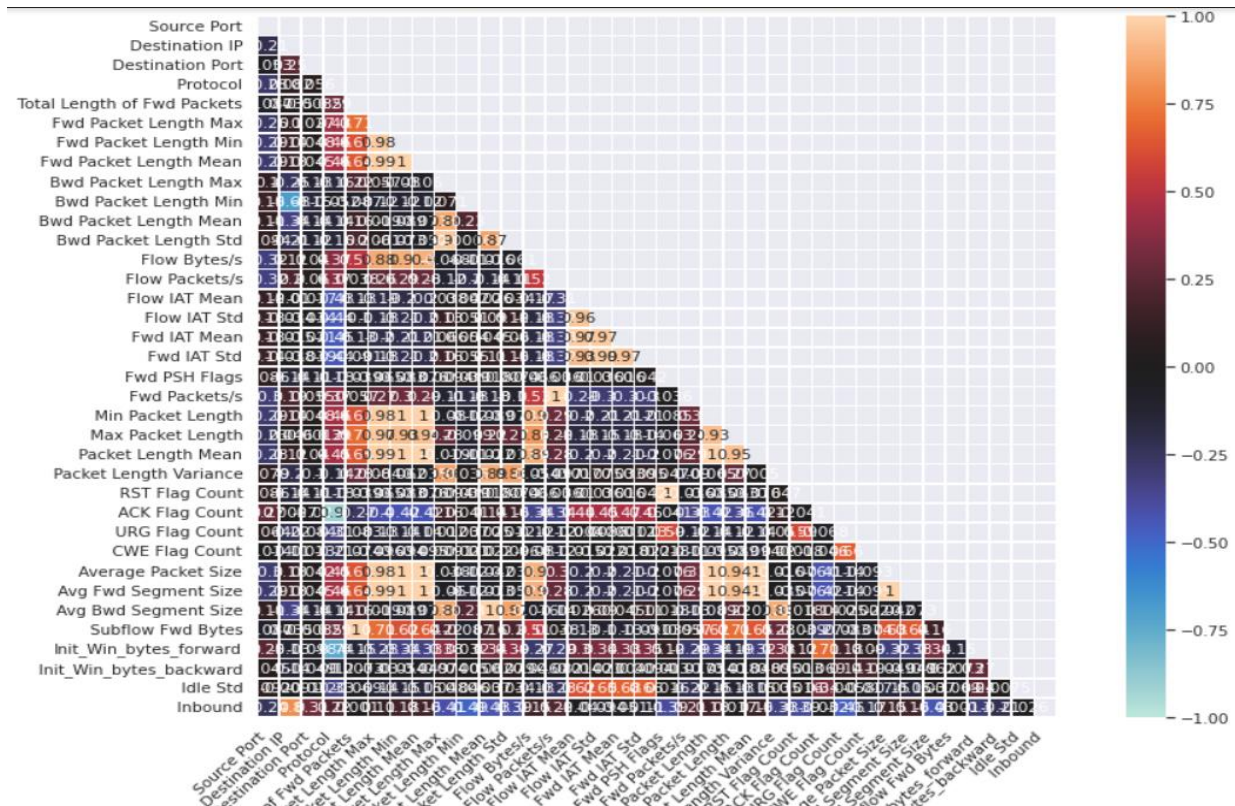


Fig. 37: Features Heat map

We have then to separate the dataset into input features and outputs as shown in Figure [38] below, where x is a variable containing the input features columns, while y is the output column 'Label' containing 8 outputs which are 'BENIGN', 'NETBIOS', 'LDAP', 'MSSQL', 'Portmap', 'SYN', 'UDP', 'UDPLag'.

```
[54] #Separating input and output attributes
      x = normalized_df.drop(['Label'], axis=1)
      y = normalized_df['Label']
```

Fig. 38: Separating Input and Output attributes

6.9. Evaluation and comparison of performance

The performance evaluation stage is critical for understanding just how much the model would perform on unknown data. All of the models presented in this study were evaluated employing eight distinct measurement criteria. Those eight parameters allow us to assess multiple models and measure the utility of findings for each classification label. The performance was assessed using the matrices indicated below.

- **Confusion Matrix:**

It's a table that displays all of the model's results. This matrix's rows are the predictions created for each classification, while the columns are the actual category designations. With N class labels, the confusion matrix organizes the anticipated outcomes as rows and the accurate IDs as columns. The leading diagonal reflects the model's accurate predictions since, columns and rows indicate classifications from first to last Nth label. The following words are used to offer a more thorough analysis:

- False Positive:** The false positive parameter denotes to the amount of data whereby the algorithm erroneously estimates the given class.
- True Positive:** The true positive parameter represents the number of data inputs that were successfully categorized.

- iii. **True Negative:** The true negative is number of samples or data points accurately categorized as not associated with a particular class. That result may be produced by adding all of the values in the database with the exception of the row and column corresponding to that specific class.
- iv. **False Negative:** The false negative parameter corresponds to the quantity of data from one class that are wrongly forecasted as belonging to another class by the model.
- **F1 Score:** F1 score parameter is a harmonic ratio that assists us in balancing precision and memory. It allows us to combine the aforementioned two measures into a single score, allowing us to determine the model's performance by examining both accuracy and recall at the same moment.

$$\text{F1 Score} = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \right)$$

- **Support:** It displays the total probability of data in a certain category. That shows us the amount of specimens were used in the research and provides us to assess the dataset's dissemination using classes.
- **Macro-average:** Because each class's prior measurements were derived individually, we need a way to generalize our findings across the whole model. All of these indicators may be summarized using weighted and macro average. Total of the obtained values of the abovementioned metrics on a macro-scale average (f1 score, recall, Precision, etc.) over all classes is computed and divided by (N) which is the total number of classes.

$$\text{Macro Average} = \frac{\text{Score of class1} + \dots + \text{Score of class } N}{N}$$

- **Recall:** This statistic is a proportion of the sample size controls and indicators by the model to the entire sample within that category or class. Sometimes it is referred to as susceptibility since, it reflects how vulnerable the model is to the existence of a given class.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- **Precision:** Precision is a statistic that may be used to assess the quality or accuracy of a model's estimation for a certain class. It is the number of times the model accurately guessed the outcome to be a specific label among all the times the algorithm indicated an outcome would be a given class. Therefore, calculating precision is given by the following formula.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- **Weighted average:** Weighted average is an average approach that considers the dataset's unequal class distribution of data. In this case, the metric value for every class is determined by multiplying the quantity of entering data within this class or category and then divide it by the total amount of data in the dataset.

$$\text{Weighted Average} = \frac{\sum(\text{Score of } i\text{th class} * \text{Support of } i\text{th class})}{\text{Total number of samples in the dataset}}$$

- **Accuracy:** Accuracy of the model is calculated as shown below by dividing the number of correctly classified predictions by the entire trials of predictions. The shortcoming of this statistic that it won't tell you which classes the model successfully detected and which it wrongly categorized.

$$\text{Accuracy} = \frac{\text{Correctly classified observations}}{\text{Total number of observations classified}}$$

Chapter 7: Proposed Models Simulation Results

In model 1 and 2, we will be Identifying whether the incoming packet is an attack (MALIGN) or non-attack (BENIGN) and if it is an attack the model will be classifying specifically the type of the attack. I replaced the types of attacks in the 'Label' feature column from '0', '1', '2', '3', '4', '5', '6', '7' with their corresponding attacks names which are 'BENIGN', 'NetBIOS', 'LDAP', 'MSSQL', 'Portmap', 'Syn', 'UDP', 'UDPLag' as shown in Figure [39] below.

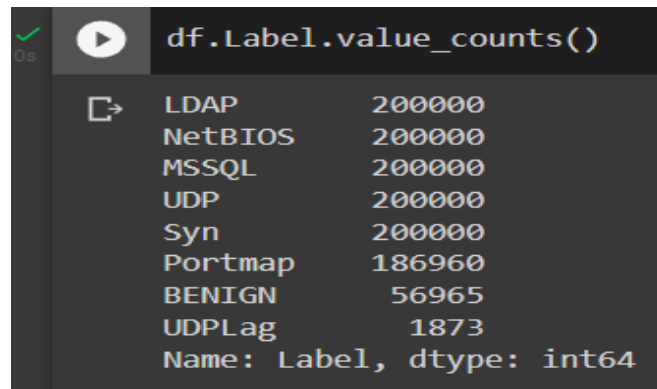


```
#df.columns
df.Label.unique()
df['Label'] = df['Label'].replace('0', 'BENIGN')
df['Label'] = df['Label'].replace('1', 'NetBIOS')
df['Label'] = df['Label'].replace('2', 'LDAP')
df['Label'] = df['Label'].replace('3', 'MSSQL')
df['Label'] = df['Label'].replace('4', 'Portmap')
df['Label'] = df['Label'].replace('5', 'Syn')
df['Label'] = df['Label'].replace('6', 'UDP')
df['Label'] = df['Label'].replace('7', 'UDPLag')
df['Label'] = df['Label'].astype('str')
df.Label.unique()

array(['LDAP', 'NetBIOS', 'MSSQL', 'UDP', 'BENIGN', 'Portmap', 'UDPLag',
       'Syn'], dtype=object)
```

Fig. 39: Labeling the types of attacks

Now we can see that we have equal numbers of LDAP, NetBIOS, MSSQL, UDP and Syn attacks which is 200000, while 186960 Portmap attacks, 1873 UDPLag attacks, 56965 BENIGN (non-attacks).

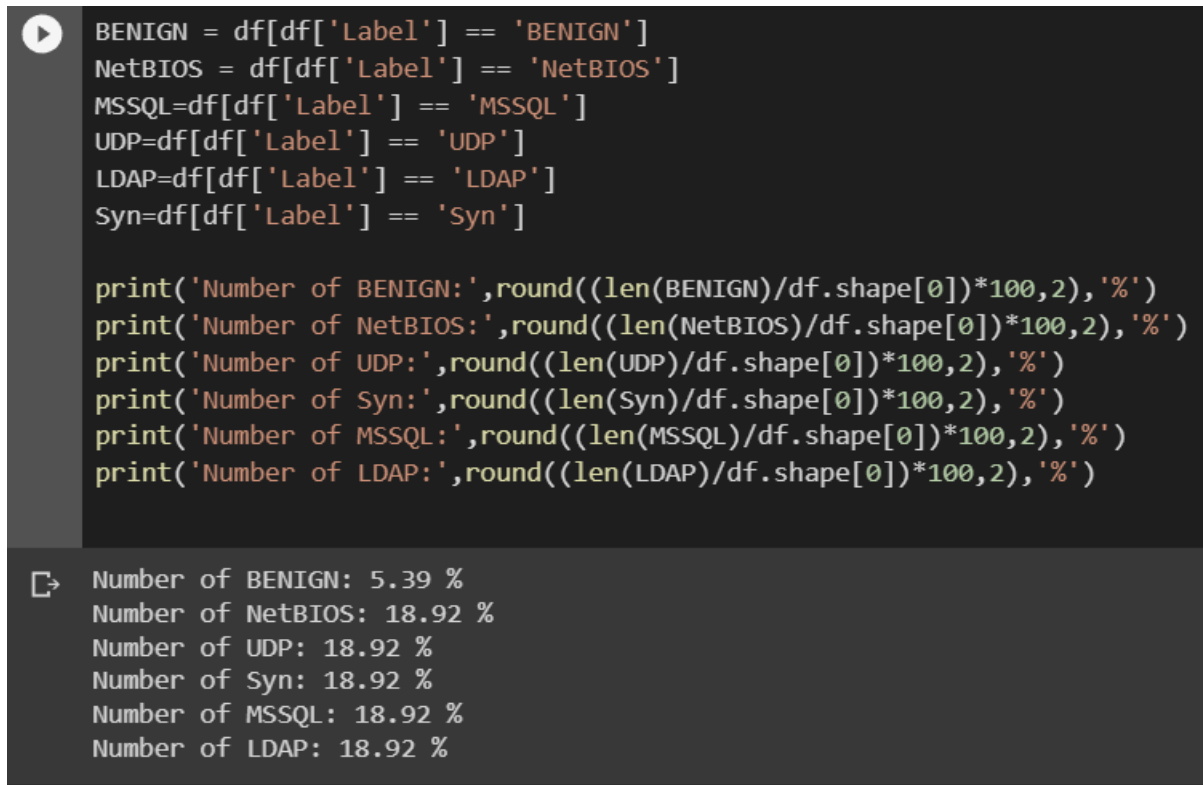


```
df.Label.value_counts()

LDAP      200000
NetBIOS   200000
MSSQL     200000
UDP       200000
Syn       200000
Portmap   186960
BENIGN    56965
UDPLag    1873
Name: Label, dtype: int64
```

Fig. 40: Label counts

We then removed the Portmap and UDPLag attacks from the dataset and after removing them we got the following Label percentages as shown in Figure [41] below.



```

BENIGN = df[df['Label'] == 'BENIGN']
NetBIOS = df[df['Label'] == 'NetBIOS']
MSSQL=df[df['Label'] == 'MSSQL']
UDP=df[df['Label'] == 'UDP']
LDAP=df[df['Label'] == 'LDAP']
Syn=df[df['Label'] == 'Syn']

print('Number of BENIGN:',round((len(BENIGN)/df.shape[0])*100,2),'%')
print('Number of NetBIOS:',round((len(NetBIOS)/df.shape[0])*100,2),'%')
print('Number of UDP:',round((len(UDP)/df.shape[0])*100,2),'%')
print('Number of Syn:',round((len(Syn)/df.shape[0])*100,2),'%')
print('Number of MSSQL:',round((len(MSSQL)/df.shape[0])*100,2),'%')
print('Number of LDAP:',round((len(LDAP)/df.shape[0])*100,2),'%')

```

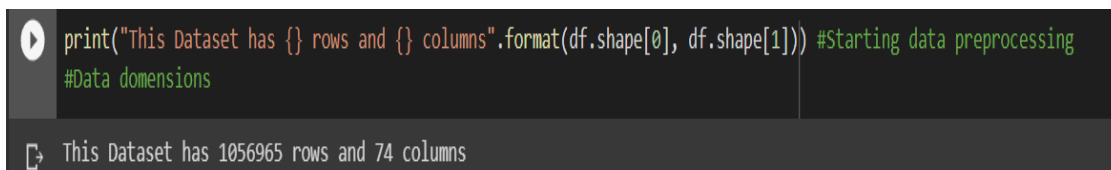
```

➞ Number of BENIGN: 5.39 %
Number of NetBIOS: 18.92 %
Number of UDP: 18.92 %
Number of Syn: 18.92 %
Number of MSSQL: 18.92 %
Number of LDAP: 18.92 %

```

Fig. 41: Label Percentage

After removing the Portmap and UDPLag attacks from the dataset we got the following updated dataset dimensions.



```

print("This Dataset has {} rows and {} columns".format(df.shape[0], df.shape[1])) #Starting data preprocessing
#Data domensions

```

```

➞ This Dataset has 1056965 rows and 74 columns

```

Fig. 42: Dataset Dimensions

Also the Label class distribution introduced before in the data preprocessing section have been changed as follows.

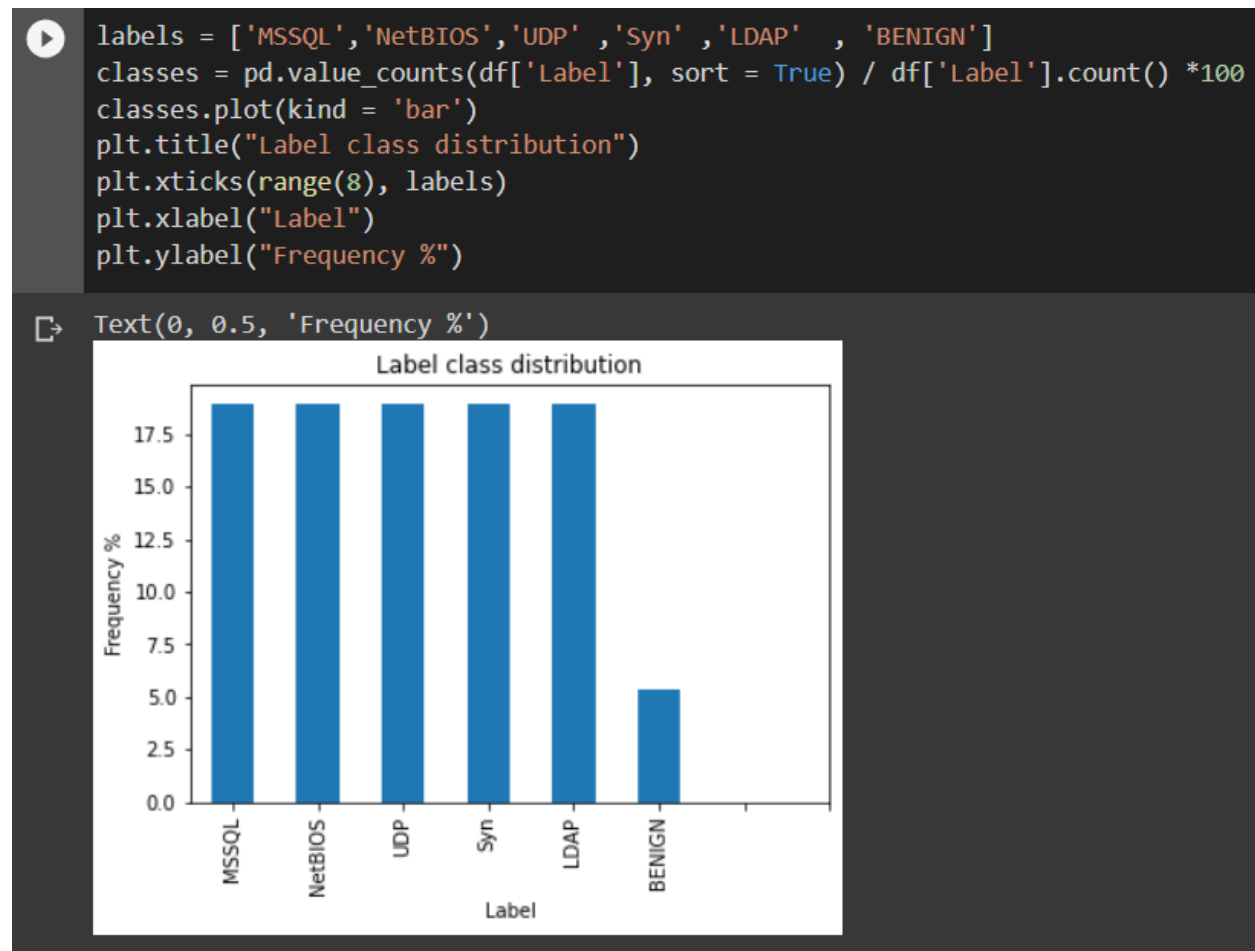


Fig. 43: Label Class Distribution

7.1. Model 1: Classification using Convolutional Neural Networks (CNN)

In model 1, We will be using Convolutional Neural Networks for detecting and classifying the DDoS attacks.

Starting by splitting the updated and normalized dataset into input and output training and testing separated datasets with a ratio 75 : 25, getting the following dimensions which are (792723,34) for input training while (264242,34) for input testing, (792723,6) for output training and (264242,6) for output testing as shown below.


```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
print(X_train.shape, X_test.shape)

```

(792723, 34) (264242, 34)

Fig. 44: Train and Test split model 1

```

print('xtrain={}, ytrain={}, xtest={}, ytest={}'.format(X_train.shape,y_train.shape,X_test.shape,y_test.shape))

```

xtrain=(792723, 34, 1), ytrain=(792723, 6), xtest=(264242, 34, 1), ytest=(264242, 6)

Fig. 45: Split dimensions model 2

Convolutional Neural Network (CNN) Structure:

For feature extraction from data, two 1D convolutional neural network layers are combined. The first layer is made up of 64 filters of size 3, while the second layer is made up of 32 filters of the same size. The collected features are subsequently processed via a layer of max pooling, which decreases complexity by just choosing most common features. These max pooling layer attributes are flattened to a 1D vector and input into a deep forward neural network with two hidden layers. The first layer is made up of 15 neurons, whereas the second is made up of 10 neurons. The result is then sent to a final classification layer of 6 neuron units.

```

Model: "sequential_4"

```

Layer (type)	Output Shape	Param #
conv1d_8 (Conv1D)	(None, 32, 64)	256
conv1d_9 (Conv1D)	(None, 30, 32)	6176
max_pooling1d_4 (MaxPooling 1D)	(None, 15, 32)	0
flatten_4 (Flatten)	(None, 480)	0
Hidden_layer_1 (Dense)	(None, 15)	7215
Hidden_layer_2 (Dense)	(None, 10)	160
Output_layer (Dense)	(None, 6)	66

```

Total params: 13,873
Trainable params: 13,873
Non-trainable params: 0

```

Fig. 46: Model 1 Structure

I then started the training and validation processes. The training pipeline has 20 epochs and a batch size with 2000 data points.

```
Epoch 1/20
397/397 [=====] - 60s 148ms/step - loss: 0.2484 - accuracy: 0.9159 - val_loss: 0.0636 - val_accuracy: 0.9770
Epoch 2/20
397/397 [=====] - 57s 144ms/step - loss: 0.0525 - accuracy: 0.9822 - val_loss: 0.0506 - val_accuracy: 0.9820
Epoch 3/20
397/397 [=====] - 57s 143ms/step - loss: 0.0431 - accuracy: 0.9859 - val_loss: 0.0406 - val_accuracy: 0.9883
Epoch 4/20
397/397 [=====] - 57s 143ms/step - loss: 0.0383 - accuracy: 0.9876 - val_loss: 0.0380 - val_accuracy: 0.9888
Epoch 5/20
397/397 [=====] - 57s 143ms/step - loss: 0.0361 - accuracy: 0.9883 - val_loss: 0.0424 - val_accuracy: 0.9853
Epoch 6/20
397/397 [=====] - 57s 143ms/step - loss: 0.0346 - accuracy: 0.9890 - val_loss: 0.0390 - val_accuracy: 0.9872
Epoch 7/20
397/397 [=====] - 56s 142ms/step - loss: 0.0336 - accuracy: 0.9895 - val_loss: 0.0329 - val_accuracy: 0.9904
Epoch 8/20
397/397 [=====] - 58s 146ms/step - loss: 0.0324 - accuracy: 0.9899 - val_loss: 0.0351 - val_accuracy: 0.9900
Epoch 9/20
397/397 [=====] - 56s 142ms/step - loss: 0.0316 - accuracy: 0.9903 - val_loss: 0.0315 - val_accuracy: 0.9905
Epoch 10/20
397/397 [=====] - 57s 143ms/step - loss: 0.0311 - accuracy: 0.9904 - val_loss: 0.0344 - val_accuracy: 0.9902
Epoch 11/20
397/397 [=====] - 56s 142ms/step - loss: 0.0300 - accuracy: 0.9908 - val_loss: 0.0320 - val_accuracy: 0.9902
Epoch 12/20
397/397 [=====] - 56s 142ms/step - loss: 0.0298 - accuracy: 0.9909 - val_loss: 0.0304 - val_accuracy: 0.9909
Epoch 13/20
```

Fig. 47: Training and validation for model 1

- Plotting the epoch loss against the no. of epochs

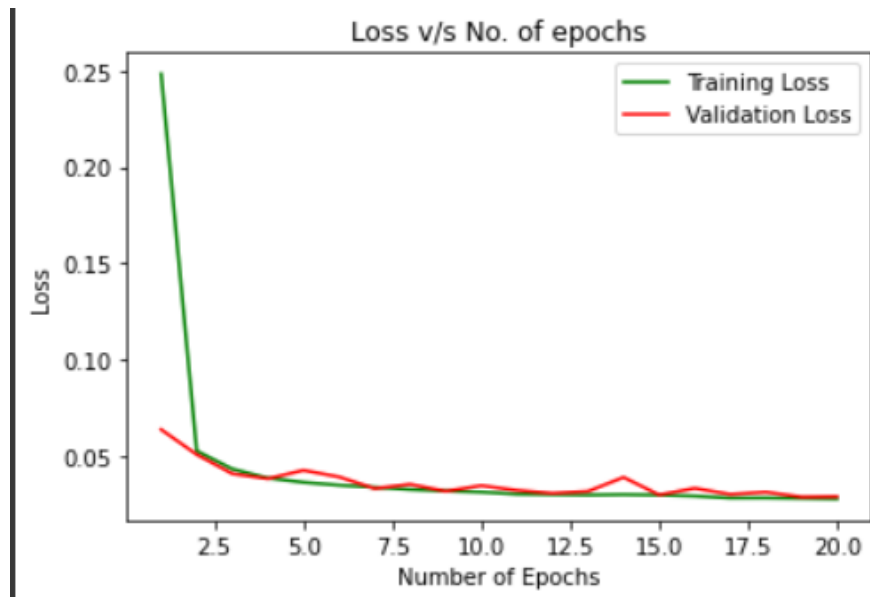


Fig. 48: Plotting Loss v/s No. of epochs model 1

- Plotting the accuracy against the number of epochs

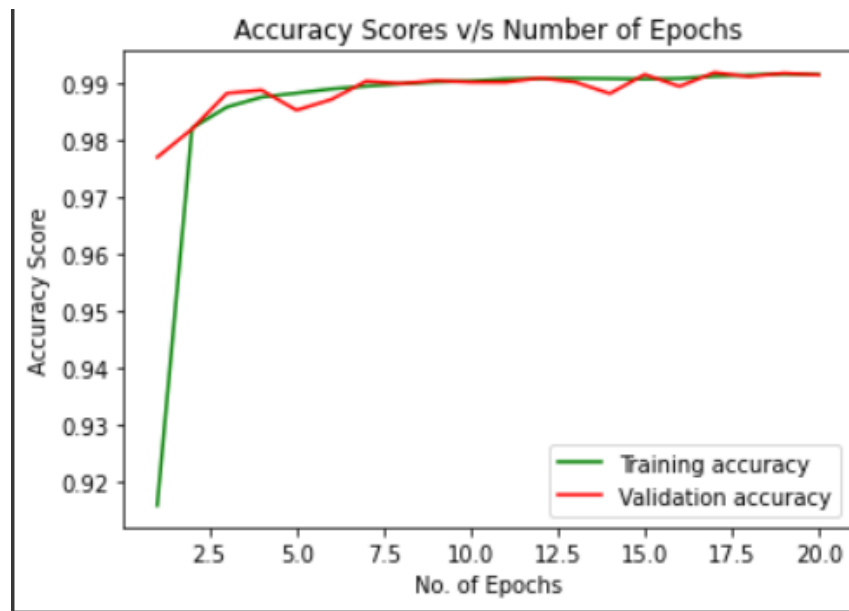


Fig. 49: Plotting Accuracy v/s no. of epochs model 1

Calculating the overall Convolutional Neural Network model 3 accuracy and loss, I got the following results as shown in Figure [50] below. A loss of approximately 0.0287 with an accuracy of approximately 99.15% .

```
[23] Classifier_accuracy=[]
      loss, accuracy = m2.evaluate(X_test, y_test)
      print('Accuracy of Deep neural Network : %.2f'% (accuracy*100))
      Classifier_accuracy.append(accuracy*100)

8258/8258 [=====] - 23s 3ms/step - loss: 0.0287 - accuracy: 0.9915
Accuracy of Deep neural Network : 99.15
```

Fig. 51: Model 1 Performance

For making sure that the CNN model is working perfectly, I applied some predictions using the input testing features and see the output results to compare with. I used an array of 110 samples as shown in the figure below, and I got an approximately identical prediction outputs with only difference in one or two samples which is normal as we don't have a 100% accuracy.

```
#Previously known testing array
b[:110]

array([2, 5, 2, 3, 0, 3, 1, 1, 4, 4, 2, 3, 1, 2, 1, 1, 2, 1, 1, 5, 1,
       1, 2, 4, 5, 2, 4, 2, 4, 5, 5, 1, 2, 2, 1, 2, 2, 4, 5, 0, 3, 1, 1,
       1, 5, 1, 2, 1, 2, 1, 3, 5, 3, 2, 1, 0, 1, 5, 4, 4, 5, 2, 5, 4, 3,
       0, 5, 2, 2, 4, 4, 1, 2, 2, 1, 2, 1, 1, 3, 5, 5, 2, 4, 2, 2, 0, 3,
       1, 4, 2, 2, 0, 4, 2, 5, 5, 4, 1, 5, 2, 3, 2, 4, 2, 3, 5, 3, 5, 1])
```

Fig. 52: Previously known testing array model 1

```

▶ #Prediction array
a[:110]

array([2, 5, 2, 3, 0, 3, 1, 1, 4, 4, 2, 3, 1, 2, 1, 1, 2, 1, 1, 1, 5, 1,
       1, 2, 4, 5, 2, 4, 2, 4, 5, 5, 1, 2, 2, 1, 2, 2, 4, 5, 0, 3, 1, 1,
       1, 5, 1, 2, 1, 2, 1, 3, 5, 3, 2, 1, 0, 1, 5, 4, 4, 5, 2, 5, 4, 3,
       0, 5, 2, 2, 4, 4, 1, 2, 2, 1, 2, 1, 1, 3, 5, 5, 2, 4, 2, 2, 0, 3,
       1, 4, 2, 2, 0, 4, 2, 5, 5, 4, 1, 3, 2, 3, 2, 4, 2, 3, 5, 3, 5, 1])

```

Fig. 53: Prediction array model 1

At last we got the confusion matrix and the classification report as shown in Figure 76 below, in order to evaluate the model's accuracy.

```

Convolution Neural Network
Accuracy: 99.147751
Confusion Matrix =
[[14430      1      0      0      3      0]
 [   7 49730      5      8      1      4]
 [   3      8 49834     48      6      0]
 [   4      0     46 48897      1 1220]
 [   3      0      0      6 49881      1]
 [   1      3      1     871      1 49218]]
Recall = 0.9914775092528818
Classification Report =

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14434
1	1.00	1.00	1.00	49755
2	1.00	1.00	1.00	49899
3	0.98	0.97	0.98	50168
4	1.00	1.00	1.00	49891
5	0.98	0.98	0.98	50095
accuracy			0.99	264242
macro avg	0.99	0.99	0.99	264242
weighted avg	0.99	0.99	0.99	264242

```

F1 Score = 0.9924209089793093

```

Fig. 54: Model 1 Report

7.2. Model 2: Classification using Deep Neural Networks (DNN)

In model 2, we will be using Deep Neural Networks in order to detect and classify the DDoS attacks. Firstly, we will start by splitting the updated and normalized dataset into input

and output training and testing separated datasets with a ratio 75 : 25, getting the following dimensions which are (792723,34) for input training while (264242,34) for input testing, (792723,6) for output training and (264242,6) for output testing as shown below.

```
[ ] #Splitting Data into training and testing
    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
    print(x_train.shape, x_test.shape)

(792723, 34) (264242, 34)
```

Fig. 55: Train and Test Split model 2

```
[ ] print('xtrain={}, ytrain={}, xtest={}, ytest={}'.format(x_train.shape,y_train.shape,x_test.shape,y_test.shape))

xtrain=(792723, 34), ytrain=(792723, 6), xtest=(264242, 34), ytest=(264242, 6)
```

Fig. 56: Split Dimensions model 2

Deep Neural Network (DNN) model Structure:

Firstly, I have developed more than one model to find the more efficient one. I have constructed models with 2 to 6 concealed layers and 32 neuron units each layer. I then started to lower and variate hidden layers from six to three after achieving the highest degree of accuracy. Finally, as shown in Figure [57] below, I achieved the simplest model with the maximum accuracy.

After trying more than one design, I got the best results with this one. My Deep Neural Network model for attack identification is composed of an input layer of 34 neurons as the number of features in the dataset is 34. The model consists of 3 hidden layers, the three layers consisted of 17 neurons each, and finally the output layer only consists of 2 neurons as it just identify weather the output is an attack or non-attack. I have used relu activation function for the three hidden layers while sigmoid for the output layer.

Model: "sequential_6"

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 17)	595
Hidden_Layer_2 (Dense)	(None, 17)	306
Hidden_Layer_3 (Dense)	(None, 17)	306
Output_Layer (Dense)	(None, 6)	108

=====
 Total params: 1,315
 Trainable params: 1,315
 Non-trainable params: 0
 =====

Fig. 57: Model 2 Structure

I used the Keras package with the Tensorflow backend to train the deep neural network model and left all dense layer parameters at their default levels. The training pipeline has 100 epochs and a batch size with 2000 data points

Epoch 1/100
397/397 - 3s - loss: 0.1455 - accuracy: 0.9519 - val_loss: 0.0572 - val_accuracy: 0.9810 - 3s/epoch - 7ms/step

Epoch 2/100
397/397 - 2s - loss: 0.0533 - accuracy: 0.9820 - val_loss: 0.0493 - val_accuracy: 0.9853 - 2s/epoch - 5ms/step

Epoch 3/100
397/397 - 2s - loss: 0.0493 - accuracy: 0.9831 - val_loss: 0.0724 - val_accuracy: 0.9746 - 2s/epoch - 5ms/step

Epoch 4/100
397/397 - 2s - loss: 0.0475 - accuracy: 0.9837 - val_loss: 0.0433 - val_accuracy: 0.9863 - 2s/epoch - 5ms/step

Epoch 5/100
397/397 - 2s - loss: 0.0455 - accuracy: 0.9846 - val_loss: 0.0413 - val_accuracy: 0.9866 - 2s/epoch - 5ms/step

Epoch 6/100
397/397 - 2s - loss: 0.0444 - accuracy: 0.9844 - val_loss: 0.0515 - val_accuracy: 0.9808 - 2s/epoch - 5ms/step

Epoch 7/100
397/397 - 2s - loss: 0.0493 - accuracy: 0.9833 - val_loss: 0.0451 - val_accuracy: 0.9858 - 2s/epoch - 6ms/step

Epoch 8/100
397/397 - 3s - loss: 0.0434 - accuracy: 0.9851 - val_loss: 0.0453 - val_accuracy: 0.9820 - 3s/epoch - 8ms/step

Epoch 9/100
397/397 - 2s - loss: 0.0429 - accuracy: 0.9849 - val_loss: 0.0423 - val_accuracy: 0.9861 - 2s/epoch - 5ms/step

Epoch 10/100
397/397 - 2s - loss: 0.0422 - accuracy: 0.9853 - val_loss: 0.0488 - val_accuracy: 0.9825 - 2s/epoch - 5ms/step

Epoch 11/100
397/397 - 2s - loss: 0.0399 - accuracy: 0.9862 - val_loss: 0.0418 - val_accuracy: 0.9848 - 2s/epoch - 5ms/step

Epoch 12/100

Fig. 58: Training and Validation for model 2

- Plotting the epoch loss against the no. of epochs

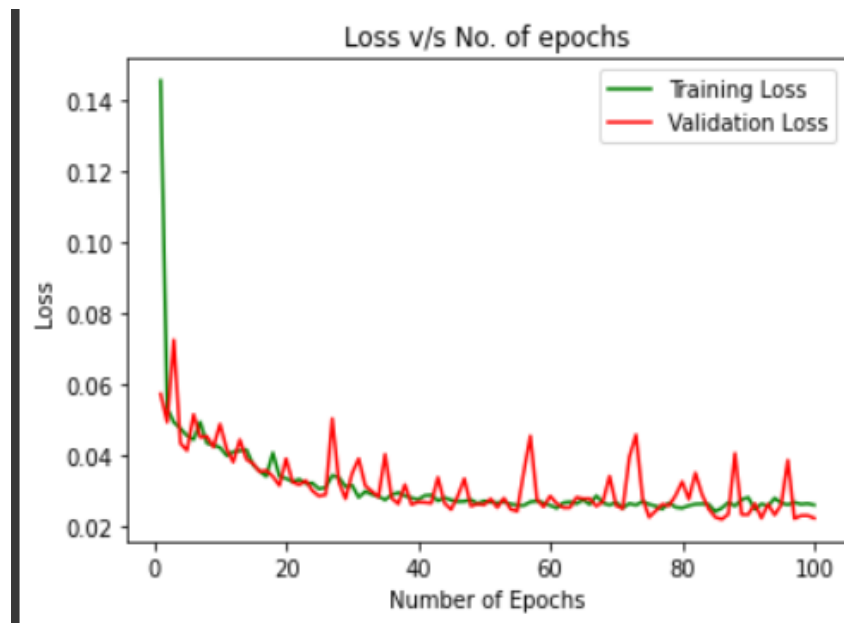


Fig. 59: Plotting Loss v/s No. of epochs model 2

- Plotting the accuracy against the number of epochs

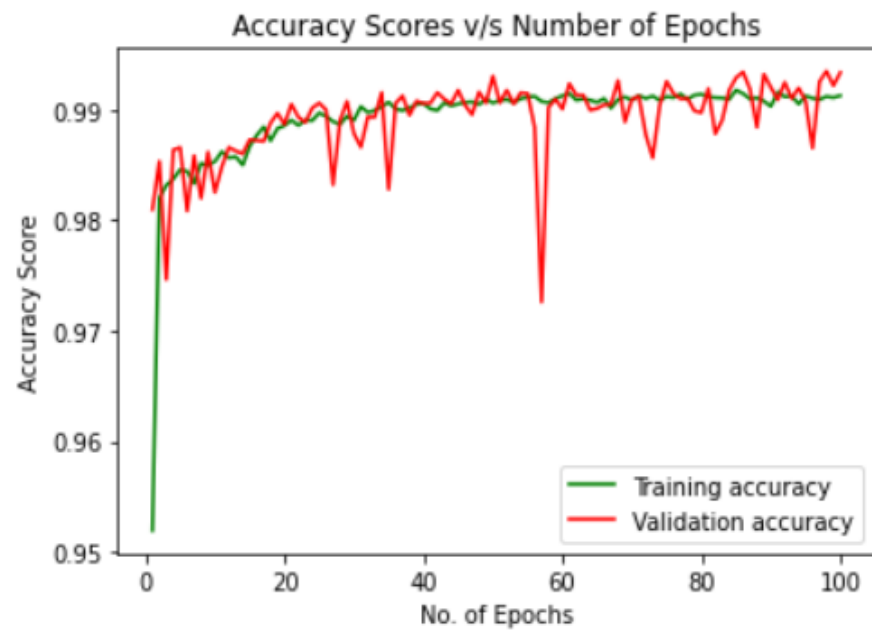


Fig. 60: Plotting Accuracy v/s no. of epochs model 2

When finally calculating the overall Deep Neural Network model 2 accuracy and loss, I got the following results as shown in Figure [61] below. A loss of approximately 0.0223 which is a small value for loss and therefore, I got an accuracy of approximately 99.33% .

```

Classifier_accuracy=[]
loss, accuracy = model.evaluate(X_test, y_test)
print('Accuracy of Deep neural Network : %.2f'% (accuracy*100))
Classifier_accuracy.append(accuracy*100)

8258/8258 [=====] - 16s 2ms/step - loss: 0.0223 - accuracy: 0.9933
Accuracy of Deep neural Network : 99.33

```

Fig. 61: Model 2 Performance

Also we have to make sure that the proposed DNN model is working perfectly, I apply some predictions using the input testing features and see the output results to compare with. I used a random number of samples of 110 and I got an identical prediction outputs as shown in the figure below.

```

[96] #Prediction array
a[:110]

array([2, 5, 2, 3, 0, 3, 1, 1, 4, 4, 2, 3, 1, 2, 1, 1, 2, 1, 1, 1, 5, 1,
       1, 2, 4, 5, 2, 4, 2, 4, 5, 3, 1, 2, 2, 1, 2, 2, 4, 5, 0, 3, 1, 1,
       1, 5, 1, 2, 1, 2, 1, 3, 5, 3, 2, 1, 0, 1, 5, 4, 4, 5, 2, 5, 4, 3,
       0, 5, 2, 2, 4, 4, 1, 2, 2, 1, 2, 1, 1, 3, 5, 5, 2, 4, 2, 2, 0, 3,
       1, 4, 2, 2, 0, 4, 2, 5, 5, 4, 1, 3, 2, 3, 2, 4, 2, 3, 5, 3, 5, 1])

```

Fig. 62: Prediction array model 2

```

#Previously known testing array
b[:110]

array([2, 5, 2, 3, 0, 3, 1, 1, 4, 4, 2, 3, 1, 2, 1, 1, 2, 1, 1, 1, 5, 1,
       1, 2, 4, 5, 2, 4, 2, 4, 5, 5, 1, 2, 2, 1, 2, 2, 4, 5, 0, 3, 1, 1,
       1, 5, 1, 2, 1, 2, 1, 3, 5, 3, 2, 1, 0, 1, 5, 4, 4, 5, 2, 5, 4, 3,
       0, 5, 2, 2, 4, 4, 1, 2, 2, 1, 2, 1, 1, 3, 5, 5, 2, 4, 2, 2, 0, 3,
       1, 4, 2, 2, 0, 4, 2, 5, 5, 4, 1, 5, 2, 3, 2, 4, 2, 3, 5, 3, 5, 1])

```

Fig. 63: Testing array model 2

At last we got the confusion matrix and the classification report as shown in Figure [64] below, in order to evaluate the model's accuracy.

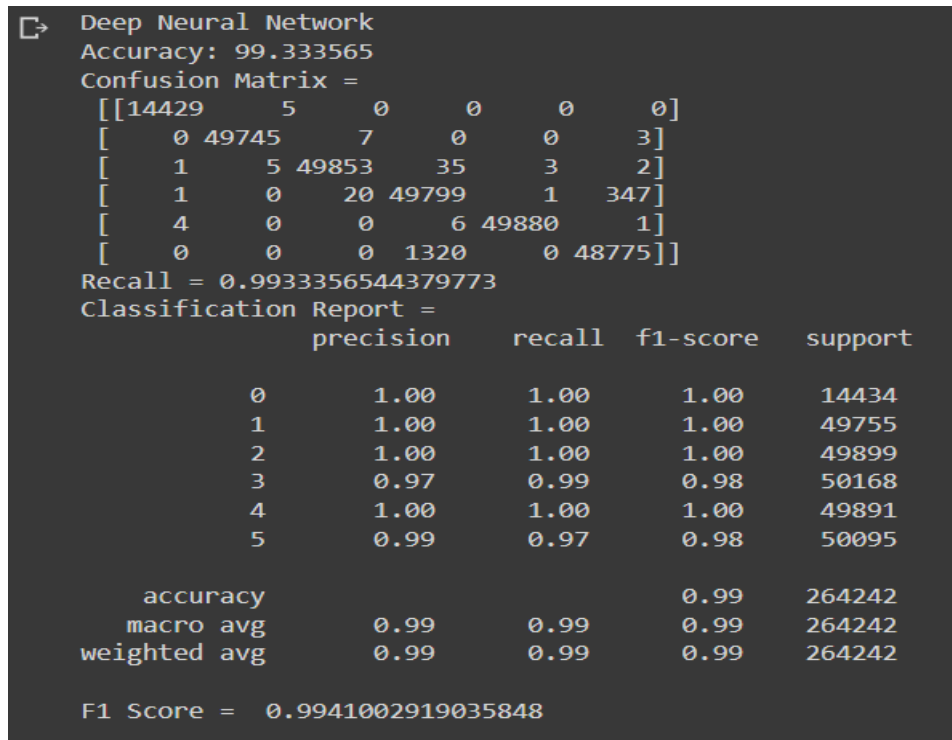


Fig. 64: Model 2 Report

7.3. Model 3: Identification using Deep Neural Networks (DNN).

In Model 3 we are just Identifying whether the incoming packet is an attack (MALIGN) or non-attack (BENIGN). We are not doing any sort of classification in this model, so I replace the types of attacks in the 'Label' feature column into '0' for the non-attack (BENIGN) and '1' for the rest of attacks as shown in Figure [65] below.

```

#df.columns
df.Label.unique()
df['Label'] = df['Label'].replace('0', 'BENIGN')
df['Label'] = df['Label'].replace('1', 'MALIGN')
df['Label'] = df['Label'].replace('2', 'MALIGN')
df['Label'] = df['Label'].replace('3', 'MALIGN')
df['Label'] = df['Label'].replace('4', 'MALIGN')
df['Label'] = df['Label'].replace('5', 'MALIGN')
df['Label'] = df['Label'].replace('6', 'MALIGN')
df['Label'] = df['Label'].replace('7', 'MALIGN')
df['Label'] = df['Label'].astype('str')
df.Label.unique()
array(['MALIGN', 'BENIGN'], dtype=object)

```

Fig. 65: BENIGN and MALIGN

Now after converting the labels into attack and non-attack, we got 1,188,833 attacks (MALIGN) and 56,965 non attacks (BENIGN) as shown in Figure 40 below. We observed that we have 4.57 % Non-Attacks and 95.43 % Attacks.

```

df.Label.value_counts()
MALIGN    1188833
BENIGN     56965
Name: Label, dtype: int64

```

Fig. 66: Label count

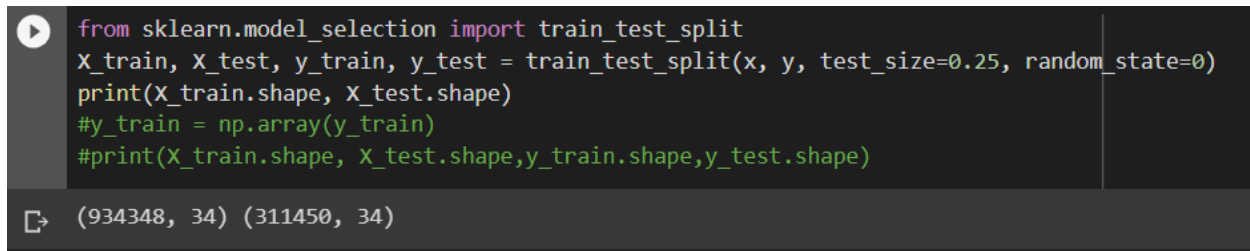
```

BENIGN = df[df['Label'] == 'BENIGN']
MALIGN = df[df['Label'] == 'MALIGN']
print('Number of Non-Attacks(BENIGN):', round((len(BENIGN)/df.shape[0])*100,2), '%')
print('Number of Attacks(MALIGN):', round((len(MALIGN)/df.shape[0])*100,2), '%')
Number of Non-Attacks(BENIGN): 4.57 %
Number of Attacks(MALIGN): 95.43 %

```

Fig. 67: Label Percentage

Now I will separate the dataset into training set and testing set with a ratio of 75% for training and 25% for testing as shown in Figure [68] below.

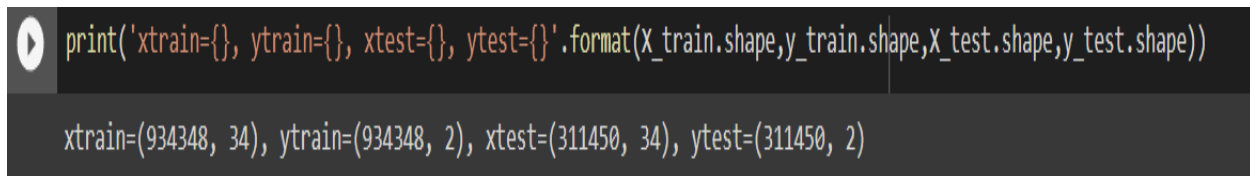


```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
print(X_train.shape, X_test.shape)
#y_train = np.array(y_train)
#print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

↳ (934348, 34) (311450, 34)

Fig. 68: Splitting Dataset model 3

The figure below shows the shape and dimensions of each element after splitting, as we can see the input features for (x) are 34 while the output (y) has only two outputs, either BENIGN [0] or MALIGN [1].



```
print('xtrain={}, ytrain={}, xtest={}, ytest={}'.format(X_train.shape, y_train.shape, X_test.shape, y_test.shape))
```

xtrain=(934348, 34), ytrain=(934348, 2), xtest=(311450, 34), ytest=(311450, 2)

Fig. 69: After Splitting Dataset model 3

Deep Neural Network (DNN) model Structure:

Same as the previous DNN model, I have developed more than one model to find the more efficient one. I have constructed models with 2 to 6 concealed layers and 32 neuron units each layer. I then started to lower and variate hidden layers from six to three after achieving the highest degree of accuracy. Finally, as shown in Figure [70] below, I achieved the simplest model with the maximum accuracy.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 17)	595
Hidden_Layer_2 (Dense)	(None, 17)	306
Hidden_Layer_3 (Dense)	(None, 17)	306
Output_Layer (Dense)	(None, 2)	36

=====
 Total params: 1,243
 Trainable params: 1,243
 Non-trainable params: 0

Fig. 70: DNN Model 3

I used the Keras package with the Tensorflow backend to train the deep neural network model and left all dense layer parameters at their default levels. The training pipeline has 100 epochs and a batch size with 2000 data points.

```

Epoch 1/100
468/468 - 3s - loss: 0.0105 - accuracy: 0.9975 - val_loss: 0.0026 - val_accuracy: 0.9992 - 3s/epoch - 5ms/step
Epoch 2/100
468/468 - 2s - loss: 0.0015 - accuracy: 0.9996 - val_loss: 0.0010 - val_accuracy: 0.9997 - 2s/epoch - 4ms/step
Epoch 3/100
468/468 - 2s - loss: 7.5943e-04 - accuracy: 0.9998 - val_loss: 9.2976e-04 - val_accuracy: 0.9998 - 2s/epoch - 4ms/step
Epoch 4/100
468/468 - 2s - loss: 6.9359e-04 - accuracy: 0.9998 - val_loss: 6.9748e-04 - val_accuracy: 0.9998 - 2s/epoch - 4ms/step
Epoch 5/100
468/468 - 2s - loss: 7.1202e-04 - accuracy: 0.9998 - val_loss: 5.8680e-04 - val_accuracy: 0.9998 - 2s/epoch - 4ms/step
Epoch 6/100
468/468 - 2s - loss: 5.6080e-04 - accuracy: 0.9999 - val_loss: 7.6746e-04 - val_accuracy: 0.9998 - 2s/epoch - 4ms/step
Epoch 7/100
468/468 - 2s - loss: 5.1898e-04 - accuracy: 0.9999 - val_loss: 4.0718e-04 - val_accuracy: 0.9999 - 2s/epoch - 4ms/step
Epoch 8/100
468/468 - 2s - loss: 5.0211e-04 - accuracy: 0.9999 - val_loss: 5.9723e-04 - val_accuracy: 0.9998 - 2s/epoch - 4ms/step
Epoch 9/100
468/468 - 2s - loss: 3.8548e-04 - accuracy: 0.9999 - val_loss: 5.6867e-04 - val_accuracy: 0.9999 - 2s/epoch - 4ms/step
Epoch 10/100
468/468 - 2s - loss: 3.7040e-04 - accuracy: 0.9999 - val_loss: 3.0782e-04 - val_accuracy: 0.9999 - 2s/epoch - 4ms/step

```

Fig. 71: Training and Validation model 3

- Plotting the epoch loss against the no. of epochs

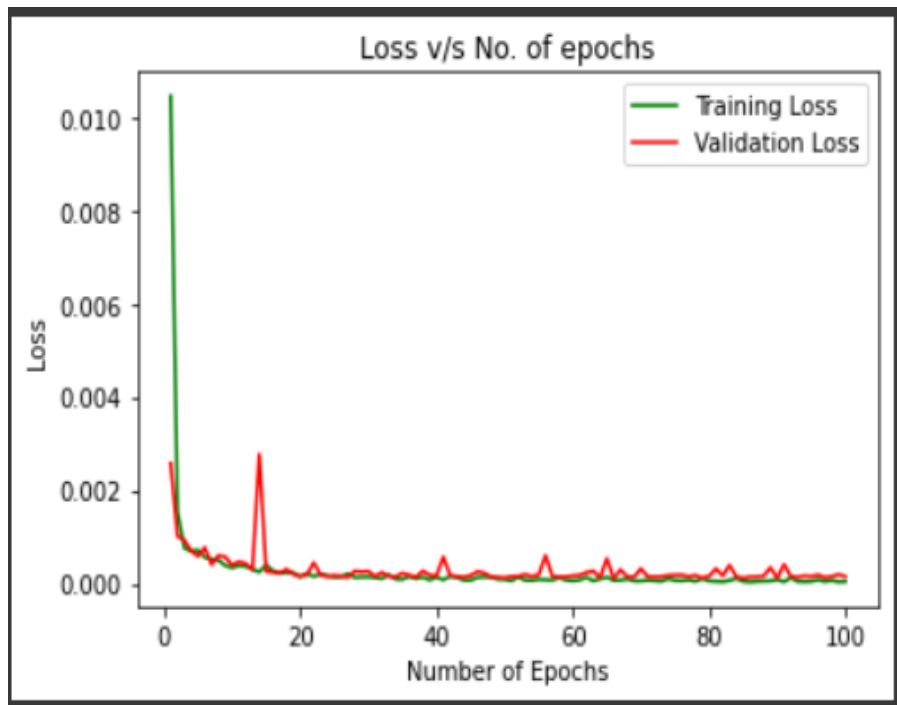


Fig. 72: Plotting Loss v/s Epochs model 3

- Plotting the accuracy score against the no. of epochs

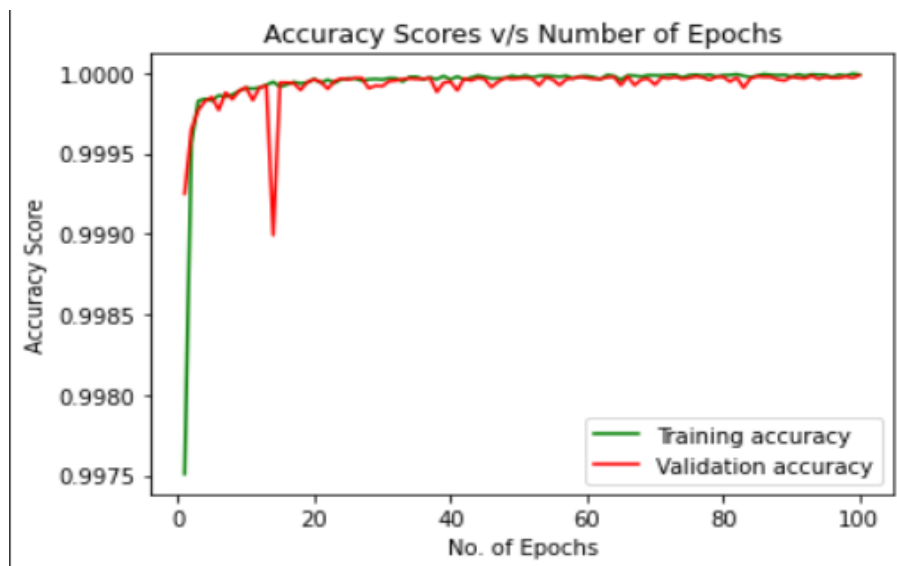


Fig. 73: Plotting Accuracy v/s Epochs model 3

When finally calculating the overall Deep Neural Network model accuracy and loss, I got the following results as shown in Figure [74] below. A loss of approximately 1.4784×10^{-4} (0.00014784) which is a very small value for loss and therefore, I got an accuracy of approximately 100%.

```

▶ Classifier_accuracy=[]
  loss, accuracy = model.evaluate(X_test, y_test)
  print('Accuracy of Deep neural Network : %.2f'% (accuracy*100))
  Classifier_accuracy.append(accuracy*100)

9733/9733 [=====] - 14s 1ms/step - loss: 1.4784e-04 - accuracy: 1.0000
Accuracy of Deep neural Network : 100.00

```

Fig. 74: Model 3 performance

In order to make sure that the proposed DNN model is working perfectly, I apply some predictions using the input testing features and see the output results to compare with. I used a random number of samples of 110 and I got an identical prediction outputs as shown in the figure below.

```

▶ #Previously known testing array
  b[:110]

↳ array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

```

Fig. 75: Testing array model 3

```

▶ #Prediction array
  a[:110]

↳ array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

```

Fig. 76: Prediction array model 3

We got the confusion matrix and the classification report as shown in Figure [77] below, in order to evaluate the model's accuracy.

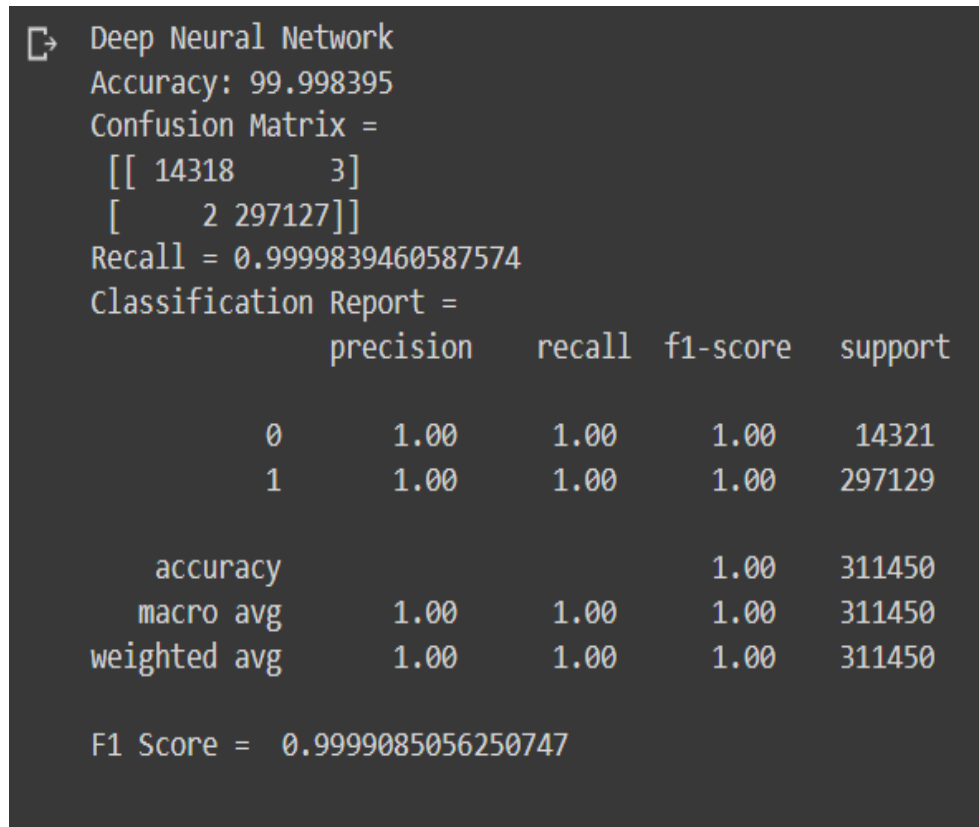


Fig. 77: Model 3 Confusion Matrix

Chapter 8: Comparative Analysis

A system was suggested by G. Usha et al.[87] with the goal of providing an efficient method for detecting DDoS assaults in a network. They have also stated that, in comparison to other methodologies and intrusion detection systems, Artificial Intelligence has been widely employed. To identify and categorize the DDoS attacks, they had employed a variety of machine learning methods such as Stochastic Gradient Descent (SGD), Extreme Gradient Boosting, Naive Bayes, K-nearest neighbor, and a deep learning architecture using the Convolutional Neural Networks (CNN).

They had used the CICDDoS2019 dataset in training their proposed models and compare their accuracies in classifying the DDoS attacks.

G. Usha et al.[87], applied their different models' algorithms on the CICDDoS2019 dataset which was reduced to 34 different characteristics of incoming traffic after pre-processing. They then trained the algorithms on the training dataset, and compared their algorithms after that based on their behavior over the test dataset, taking into account recall, f1 scores, and accuracy.

They obtained from the Convolutional Neural Network (CNN) classifier an overall accuracy of 83.89 percent, as it performed extremely badly in the categorization of UDPLag attacks, obtaining an F1 score of 0.12. It also performed badly with Portmap attacks with the same F1 score of 0.12. It had an acceptable performance at classifying NetBIOS, earning an F1 score of 0.68. It performed excellent at identifying all of the other classes, earning a macro-average F1 score of 0.723.

In this paper, I used the same dataset CICDDoS2019 from the Canadian Institute of Cybersecurity (CIC). I performed the data pre-processing in the same manner as done in [87], in order to get the same reduced dataset with the 34 features. I then compared results obtained from my different proposed models to that obtained by G. Usha et al. in [87]. In this paper, I am only interested in the proposed deep learning classifier using the Convolutional Neural Networks (CNN), and I will compare my results only with this model.

I have proposed 3 different models. In model 1, the model is capable of detecting whether the incoming packet is an attack or not and if so it classify which type of attack. I removed the Portmap and UDPLag attacks from the dataset as I have discovered that they were the reason that lowered the accuracy in the model proposed by G. Usha et al.[87]. I used the same convolutional neural network structure used in [87] but I have made a small modification by adding a Forward Deep Neural Network for the data to enter after coming out from the pooling layer.

This model achieves better accuracy and performs perfectly than that proposed by G. Usha et al.[87] in attacks classifications. I got an overall accuracy of 99.15%, while achieving F1 score of 1 for classifying BENIGN, NETBIOS , LDAP , Portmap. It also achieved F1 score of 0.98 in classifying MSSQL and Syn attacks.

In model 2, the model is capable of Identifying whether the incoming packet is an attack or not and classifying the type of the attack. I have also used in this model the dataset with the Portmap and UDPLag attacks removed. I used in this model a Feed forward Deep neural Network of 3 hidden layers.

This model achieves also a better accuracy and performs perfectly than the proposed model in [87]. I got an overall accuracy of 99.33%, while achieving F1 score of 1 for classifying BENIGN, NETBIOS , LDAP , Portmap. It also achieved F1 score of 0.98 in classifying MSSQL and Syn attacks, which is approximately the same F1 scores achieved by my modified CNN model.

In model 3, the model proposed is capable of only identifying while the incoming packet is an attack or not. I have used here the dataset without removing the Portmap and UDPLag

attacks, but I have modified the label feature to only contain a non-attack with label '0' or an attack with label '1'.

This model achieves also a perfect accuracy of approximately 100%. The model achieved F1 score of 1 for classifying both BENIGN (non-attack) and MALIGN(attack).

Chapter 9: Conclusion and Future works

9.1. Conclusion

Deep learning techniques can automatically extract high level features from low level ones and are more powerful than conventional machine learning techniques. We can also conclude that artificial neural networks are more perfect dealing with sequential data than convolutional neural networks (CNN) does, and this appeared significantly when the (CNN) model performs better after adding the Feed Forward Deep Neural Network before the final sequential classification layer and after the max pooling layer. In this research paper, I've proposed three models. Model 1 performed DDoS attack detection and classification of the attack's type using convolutional neural networks (CNN) for its training and testing. Model 2 also performed DDoS attack detection and classification using feed forward deep neural networks (DNN) for training and testing. Model 3 performed only DDoS attack detection as it only identifies whether the incoming packet is an attack or not, using also feed forward deep neural network (DNN). The developed three models were applied to CIC-DDoS2019 dataset after preprocessing. The performance of these three models was compared with similar testing conditions proposed in [87] and it was found that the three models achieved an overall accuracy of, Model 1: 99.15% , Model 2: 99.33%, and Model 3: 99.99%.

9.2. Future works

The use of neural networks is restricted to the academic and scientific worlds. As a result, by combining this technique in the form of a hybrid framework or architecture, with other neural networks, an useful attack detection system with extremely high learning rate, speedy attack detection and low error rate may be constructed. Also for the CIC-DDoS2019 dataset we can check for the Portmap attack features problem and increase the number of UDPLag attacks in the dataset as they are lowering the model performance when included.

References

- [1] R. Paffenroth and C. Zhou, “Modern Machine Learning for Cyber-defense and Distributed Denial of Service Attacks,” *IEEE Engineering Management Review*, pp. 1–1, 2019, doi: 10.1109/emr.2019.2950183.
- [2] F. Yihunie, E. Abdelfattah, and A. Odeh, “Analysis of ping of death DoS and DDoS attacks,” *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, May 2018, doi: 10.1109/lisat.2018.8378010.
- [3] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, “Distributed denial of service attacks,” *SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. “Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions” (Cat. No.00CH37166)*, doi: 10.1109/icsmc.2000.886455.
- [4] W. Staff, “A 1.3-Tbs DDoS Hit GitHub, the Largest Yet Recorded,” *WIRED*, Mar. 2018. <https://www.wired.com/story/github-ddos-memcached/>.
- [5] L. H. Newman, “What We Know About Friday’s Massive East Coast Internet Outage,” *Wired*, Oct. 21, 2016. <https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/>.
- [6] A. Bonguet and M. Bellaiche, “A Survey of Denial-of-Service and Distributed Denial of Service Attacks and Defenses in Cloud Computing,” *Future Internet*, vol. 9, no. 3, p. 43, Aug. 2017, doi: 10.3390/fi9030043.
- [7] W. Liu, “Research on DoS Attack and Detection Programming,” *IEEE Xplore*, Nov. 01, 2009. <https://ieeexplore.ieee.org/abstract/document/5368799> (accessed Mar. 25, 2021).
- [8] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, “Inferring Internet denial-of-service activity,” *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115–139, May 2006, doi: 10.1145/1132026.1132027.
- [9] Haining Wang, Danlu Zhang, and Kang G. Shin, “Detecting SYN flooding attacks,” *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, doi: 10.1109/incom.2002.1019404.
- [10] M. Bogdanoski, T. Shuminoski, and A. Risteski, “International Journal of Computer Network and Information Security(IJCNIS),” *International Journal of Computer Network and Information Security(IJCNIS)*, vol. 5, no. 8, p. 1, Accessed: Feb. 10, 2022. [Online]. Available: <https://www.mecs-press.org/ijcnis/ijcnis-v5-n8/v5n8-1.html>.
- [11] S. H. C. Haris, R. B. Ahmad, and M. A. H. A. Ghani, “Detecting TCP SYN Flood Attack Based on Anomaly Detection,” *IEEE Xplore*, Sep. 01, 2010. <https://ieeexplore.ieee.org/abstract/document/5635797>.

- [12] Aarti Singh, AARTI, and D. Juneja, “Agent Based Preventive Measure for UDP Flood Attack in DDoS Attacks,” *ResearchGate*, Aug. 2010.
https://www.researchgate.net/publication/50315626_Agent_Based_Preventive_Measure_for_UDP_Flood_Attack_in_DDoS_Attacks (accessed Apr. 30, 2021).
- [13] S. S. Kolahi, K. Treseangrat, and B. Sarrafpour, “Analysis of UDP DDoS flood cyber attack and defense mechanisms on Web Server with Linux Ubuntu 13,” *2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA '15)*, Feb. 2015, doi: 10.1109/iccspa.2015.7081286.
- [14] J. Gao and Y. Chen, “Detecting DOS/DDOS Attacks Under Ipv6,” *Proceedings of the 2012 International Conference on Cybernetics and Informatics*, pp. 847–855, Apr. 2013, doi: 10.1007/978-1-4614-3872-4_110.
- [15] D. Tang, L. Tang, W. Shi, S. Zhan, and Q. Yang, “MF-CNN: a New Approach for LDoS Attack Detection Based on Multi-feature Fusion and CNN,” *Mobile Networks and Applications*, Jan. 2020, doi: 10.1007/s11036-019-01506-1.
- [16] H. Harshita, “Detection and Prevention of ICMP Flood DDOS Attack,” *International Journal of New Technology and Research*, vol. 3, no. 3, p. 263333, Mar. 2017, Accessed: Feb. 11, 2022. [Online]. Available: <https://www.neliti.com/publications/263333/detection-and-prevention-of-icmp-flood-ddos-attack>.
- [17] M. Sikora, T. Gerlich, and L. Malina, “On Detection and Mitigation of Slow Rate Denial of Service Attacks,” *IEEE Xplore*, Oct. 01, 2019. <https://ieeexplore.ieee.org/document/8970844> (accessed Feb. 12, 2022).
- [18] T. Shorey, D. Subbaiah, A. Goyal, A. Sakxena, and A. K. Mishra, “Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools,” *IEEE Xplore*, Sep. 01, 2018. <https://ieeexplore.ieee.org/abstract/document/8554590> (accessed Nov. 26, 2021).
- [19] T. Yatagai, T. Isohara, and I. Sasase, “Detection of HTTP-GET flood Attack Based on Analysis of Page Access Behavior,” *IEEE Xplore*, Aug. 01, 2007. <https://ieeexplore.ieee.org/abstract/document/4313218>.
- [20] K. Singh, P. Singh, and K. Kumar, “Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges,” *Computers & Security*, vol. 65, pp. 344–372, Mar. 2017, doi: 10.1016/j.cose.2016.10.005.
- [21] K. Elleithy, D. Blagovic, W. Cheng, and P. Sideleau, “Denial of Service Attack Techniques: Analysis, Implementation and Comparison,” *School of Computer Science & Engineering Faculty Publications*, Jan. 2005, [Online]. Available: https://digitalcommons.sacredheart.edu/computersci_fac/52/.

- [22] A. Abdollahi and M. Fathi, “An Intrusion Detection System on Ping of Death Attacks in IoT Networks,” *Wireless Personal Communications*, Jan. 2020, doi: 10.1007/s11277-020-07139-y.
- [23] A. Dey, “Machine learning algorithms : A review,” in *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 7, pp. 1174–1179, 2016.
- [24] M. Welling and D. Bren in *A First Encounter with Machine Learning*, University of California Irvine, 2010.
- [25] M. T. Bowles, *Machine Learning in Python: Essential Techniques for Predictive Analysis*. ”John Wiley & Sons Inc”, 2015.
- [26] S. B. Kotsiantis, “Supervised machine learning: A review of classification techniques,” in *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies, (NLD)*, p. 3–24, IOS Press, 2007.
- [27] X. Du, Y. Cai, S. Wang, and L. Zhang, “Overview of deep learning,” *IEEE Xplore*, Nov. 01, 2016. <https://ieeexplore.ieee.org/document/7804882>
- [28] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/bf02478259.
- [29] A. Shrestha and A. Mahmood, “Review of Deep Learning Algorithms and Architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/access.2019.2912200.
- [30] E. Chigozie, W. Nwankpa, A. Ijomah, S. Gachagan, and Marshall, “Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,” 2018. [Online]. Available: <https://arxiv.org/pdf/1811.03378.pdf>.
- [31] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A Survey of Network-based Intrusion Detection Data Sets,” *arXiv e-prints*, p. arXiv:1903.02460, Mar. 2019.
- [32] J. J. Santanna *et al.*, “Booters — An analysis of DDoS-as-a-service attacks,” *IEEE Xplore*, May 01, 2015. <https://ieeexplore.ieee.org/abstract/document/7140298> (accessed Oct. 26, 2020).
- [33] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Computers & Security*, vol. 31, no. 3, pp. 357–374, May 2012, doi: 10.1016/j.cose.2011.12.012.
- [34] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, doi: 10.5220/0006639801080116.

- [35] “A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) - Registry of Open Data on AWS,” *registry.opendata.aws*. <https://registry.opendata.aws/cse-cic-ids2018/>
- [36] “Applications | Research | Canadian Institute for Cybersecurity | UNB,” *www.unb.ca*. <https://www.unb.ca/cic/research/applications.html>
- [37] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 DARPA off-line intrusion detection evaluation,” *Computer Networks*, vol. 34, no. 4, pp. 579–595, Oct. 2000, doi: 10.1016/s1389-1286(00)00139-0.
- [38] “KDD Cup 1999 Data,” *kdd.ics.uci.edu*. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [39] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, doi: 10.1109/cisda.2009.5356528.
- [40] Y. Ohsita, S. Ata, and M. Murata, “Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically,” *IEEE Xplore*, Nov. 01, 2004. <https://ieeexplore.ieee.org/document/1378371?arnumber=1378371> (accessed Dec. 17, 2021).
- [41] Haining Wang, Danlu Zhang, and Kang G. Shin, “Detecting SYN flooding attacks,” *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, doi: 10.1109/incom.2002.1019404.
- [42] Z. Wu, Q. Pan, M. Yue, and L. Liu, “Sequence alignment detection of TCP-targeted synchronous low-rate DoS attacks,” *Computer Networks*, vol. 152, pp. 64–77, Apr. 2019, doi: 10.1016/j.comnet.2019.01.031.
- [43] A. Izaddoost, M. Othman, and M. F. A. Rasid, “Accurate ICMP TraceBack Model under DoS/DDoS Attack,” *IEEE Xplore*, Dec. 01, 2007. <https://ieeexplore.ieee.org/document/4426009>.
- [44] O. E. Elejla, B. Belaton, M. Anbar, and A. Alnajjar, “Intrusion Detection Systems of ICMPv6-based DDoS attacks,” *Neural Computing and Applications*, vol. 30, no. 1, pp. 45–56, Dec. 2016, doi: 10.1007/s00521-016-2812-8.
- [45] H. Harshita, “Detection and Prevention of ICMP Flood DDOS Attack,” *International Journal of New Technology and Research*, vol. 3, no. 3, p. 263333, Mar. 2017, Accessed: Feb. 11, 2022. [Online]. Available: <https://www.neliti.com/publications/263333/detection-and-prevention-of-icmp-flood-ddos-attack>.
- [46] S. M. Hussain and G. R. Beigh, “Impact of DDoS attack (UDP Flooding) on queuing models,” *IEEE Xplore*, Sep. 01, 2013. <https://ieeexplore.ieee.org/document/6749629> (accessed Feb. 08, 2022).

- [47] R. Xu, W. Ma, and W. Zheng, "Defending against UDP Flooding by Negative Selection Algorithm Based on Eigenvalue Sets," *IEEE Xplore*, Aug. 01, 2009. <https://ieeexplore.ieee.org/document/5283529?arnumber=5283529> (accessed Feb. 08, 2022).
- [48] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS attack traffic from flash crowd through packet arrival patterns," *IEEE Xplore*, Apr. 01, 2011. <https://ieeexplore.ieee.org/abstract/document/5928950> (accessed Nov. 25, 2020).
- [49] T. Yatagai, T. Isohara, and I. Sasase, "Detection of HTTP-GET flood Attack Based on Analysis of Page Access Behavior," *IEEE Xplore*, Aug. 01, 2007. <https://ieeexplore.ieee.org/abstract/document/4313218>.
- [50] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, Jun. 2020, doi: 10.1109/tnsm.2020.2971776.
- [51] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," *IEEE Xplore*, Apr. 01, 2003. <https://ieeexplore.ieee.org/abstract/document/1194894> (accessed Mar. 19, 2021).
- [52] P. D. Bojović, I. Bašičević, S. Ocovaj, and M. Popović, "A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method," *Computers & Electrical Engineering*, vol. 73, pp. 84–96, Jan. 2019, doi: 10.1016/j.compeleceng.2018.11.004.
- [53] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, Dec. 2018, doi: 10.1109/TNSM.2018.2861741.
- [54] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Computers & Security*, vol. 88, p. 101645, Jan. 2020, doi: 10.1016/j.cose.2019.101645.
- [55] S. Wankhede and D. Kshirsagar, "DoS Attack Detection Using Machine Learning and Neural Network," *IEEE Xplore*, Aug. 01, 2018. <https://ieeexplore.ieee.org/document/8697702> (accessed Feb. 08, 2022).
- [56] D. J. Prathyusha and G. Kannayaram, "A cognitive mechanism for mitigating DDoS attacks using the artificial immune system in a cloud environment," *Evolutionary Intelligence*, Jan. 2020, doi: 10.1007/s12065-019-00340-4.
- [57] M. Alkasassbeh, G. Al-Naymat, A. B.A, and M. Almseidin, "Detecting Distributed Denial of Service Attacks Using Data Mining Techniques," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 1, 2016, doi: 10.14569/ijacsa.2016.070159.

- [58] J. L. Berral, N. Poggi, J. Alonso, R. Gavaldà, J. Torres, and M. Parashar, “Adaptive distributed mechanism against flooding network attacks based on machine learning,” *Proceedings of the 1st ACM workshop on Workshop on AISec - AISec '08*, 2008, doi: 10.1145/1456377.1456389.
- [59] Z. He, T. Zhang, and R. B. Lee, “Machine Learning Based DDoS Attack Detection from Source Side in Cloud,” *IEEE Xplore*, 2017. <https://ieeexplore.ieee.org/document/7987186>.
- [60] R. Doshi, N. Aphorpe, and N. Feamster, “Machine Learning DDoS Detection for Consumer Internet of Things Devices,” *2018 IEEE Security and Privacy Workshops (SPW)*, May 2018, doi: 10.1109/spw.2018.00013.
- [61] A. A. Abdulrahman and M. K. Ibrahim, “Evaluation of DDoS attacks Detection in a New Intrusion Dataset Based on Classification Algorithms,” *Iraqi Journal of Information & Communications Technology*, vol. 1, no. 3, pp. 49–55, Feb. 2019, doi: 10.31987/ijict.1.3.40.
- [62] G. Ramadhan, Y. Kurniawan, and C.-S. Kim, “Design of TCP SYN Flood DDoS attack detection using artificial immune systems,” *IEEE Xplore*, Oct. 01, 2016. <https://ieeexplore.ieee.org/document/7849626> (accessed Feb. 09, 2022).
- [63] J. D. Ndibwile, A. Govardhan, K. Okada, and Y. Kadobayashi, “Web Server Protection against Application Layer DDoS Attacks Using Machine Learning and Traffic Authentication,” *IEEE Xplore*, Jul. 01, 2015. <https://ieeexplore.ieee.org/document/7273365> (accessed Feb. 09, 2022).
- [64] M. Aamir and S. M. A. Zaidi, “DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation,” *International Journal of Information Security*, vol. 18, no. 6, pp. 761–785, Apr. 2019, doi: 10.1007/s10207-019-00434-1.
- [65] C. Wang, H. Yao, and Z. Liu, “An efficient DDoS detection based on SU-Genetic feature selection,” *Cluster Computing*, vol. 22, no. S1, pp. 2505–2515, Mar. 2018, doi: 10.1007/s10586-018-2275-z.
- [66] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, “Adaptive feature selection for denial of services (DoS) attack,” *IEEE Xplore*, Nov. 01, 2017. <https://ieeexplore.ieee.org/document/8270429> (accessed Apr. 12, 2022).
- [67] B. S. Kiruthika Devi, G. Preetha, G. Selvam, and S. Mercy Shalinie, “An impact analysis: Real time DDoS attack detection and mitigation using machine learning,” *IEEE Xplore*, Apr. 01, 2014. <https://ieeexplore.ieee.org/document/6996133> (accessed Feb. 18, 2022).
- [68] Md. Z. Hasan, K. M. Z. Hasan, and A. Sattar, “Burst Header Packet Flood Detection in Optical Burst Switching Network Using Deep Learning Model,” *Procedia Computer Science*, vol. 143, pp. 970–977, 2018, doi: 10.1016/j.procs.2018.10.337.
- [69] T. A. Tuan, H. V. Long, L. H. Son, R. Kumar, I. Priyadarshini, and N. T. K. Son, “Performance evaluation of Botnet DDoS attack detection using machine learning,” *Evolutionary Intelligence*, Nov. 2019, doi: 10.1007/s12065-019-00310-w.

- [70] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A new multi classifier system using entropy-based features in DDoS attack detection," *IEEE Xplore*, Jan. 01, 2018. <https://ieeexplore.ieee.org/document/8343104> (accessed Apr. 16, 2022).
- [71] M. Zhu, K. Ye, and C.-Z. Xu, "Network Anomaly Detection and Identification Based on Deep Learning Methods," *Lecture Notes in Computer Science*, pp. 219–234, 2018, doi: 10.1007/978-3-319-94295-7_15.
- [72] L. Fernandez Maimo, A. L. Perales Gomez, F. J. Garcia Clemente, M. Gil Perez, and G. Martinez Perez, "A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018, doi: 10.1109/access.2018.2803446.
- [73] R. Karimazad and A. Faraahi, "An Anomaly-Based Method for DDoS Attacks Detection using RBF Neural Networks." Accessed: Feb. 13, 2022. [Online]. Available: <http://ipcscit.com/vol11/9-ICNEE2011-N019.pdf>.
- [74] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," *Neurocomputing*, vol. 172, pp. 385–393, Jan. 2016, doi: 10.1016/j.neucom.2015.04.101.
- [75] D. Peraković, M. Periša, I. Cvitić, and S. Husnjak, "Artificial neuron network implementation in detection and classification of DDoS traffic," *IEEE Xplore*, Nov. 01, 2016. <https://ieeexplore.ieee.org/document/7818791> (accessed Feb. 09, 2022).
- [76] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," *IEEE Xplore*, May 01, 2017. <https://ieeexplore.ieee.org/abstract/document/7946998> (accessed Jan. 24, 2021).
- [77] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. V. Krishna, "A Deep Learning Based Artificial Neural Network Approach for Intrusion Detection," *Communications in Computer and Information Science*, pp. 44–53, 2017, doi: 10.1007/978-981-10-4642-1_5.
- [78] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and M. Sheikhan, "Flow-Based Anomaly Detection Using Neural Network Optimized with GSA Algorithm," *IEEE Xplore*, Jul. 01, 2013. <https://ieeexplore.ieee.org/abstract/document/6679866> (accessed Apr. 13, 2022).
- [79] S. Sumathi and N. Karthikeyan, "Detection of distributed denial of service using deep learning neural network," *Journal of Ambient Intelligence and Humanized Computing*, May 2020, doi: 10.1007/s12652-020-02144-2.
- [80] M. S. Elsayed, N.-A. . Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," *IEEE Xplore*, Aug. 01, 2020. <https://ieeexplore.ieee.org/abstract/document/9217754/>
- [81] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/access.2017.2762418.

- [82] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, “TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest,” *Security and Communication Networks*, vol. 2018, pp. 1–9, Jul. 2018, doi: 10.1155/2018/4943509.
- [83] K. Wu, Z. Chen, and W. Li, “A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks,” *IEEE Access*, vol. 6, pp. 50850–50859, 2018, doi: 10.1109/access.2018.2868993.
- [84] R. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, “Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks.” Accessed: Apr. 16, 2022. [Online]. Available: <http://isyou.info/jisis/vol9/no4/jisis-2019-vol9-no4-01.pdf>
- [85] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,” *2019 International Carnahan Conference on Security Technology (ICCST)*, Oct. 2019, doi: 10.1109/ccst.2019.8888419.
- [86] “DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” www.unb.ca. <https://www.unb.ca/cic/datasets/ddos-2019.html>
- [87] G. Usha, M. Narang, and A. Kumar, “Detection and Classification of Distributed DoS Attacks Using Machine Learning,” *Computer Networks and Inventive Communication Technologies*, pp. 985–1000, 2021, doi: 10.1007/978-981-15-9647-6_78.

Appendix A

Executive Summary: Denial of Service (DoS) Attack Detection Using Machine Learning

Abstract

In the current era of rapid development of the Internet, methods of information security and safety are increasingly being focused on. As a result, denial of service (DoS) and distributed denial of service (DDoS) attacks are becoming threats to all servers around the world. Since then, methods and services to prevent and minimize damage have also been launched with many different functions. In this research paper, I've used convolutional neural networks (CNN) and deep neural networks (DNN) models in the DDoS attacks detection. As a result, of using the CIC-DDoS2019 dataset in training my proposed models and comparing their accuracies in detecting and classifying the DDoS attacks. I've proposed three models, Model 1 using CNN in detecting and classifying the incoming DDoS attacks with an accuracy of 99.15%. Model 2 using DNN in detecting and classifying the DDoS attacks with an accuracy of 99.33%. Model 3 using also DNN in only detecting the attacks without any sort of classification with an accuracy of 99.99%. These high accuracy results proved how effective are the Deep learning techniques dealing with the DDoS attacks prevention.

Problem Statement

Distinguishing between flooding attacks and legitimate user activity is our main challenge. The process of detection becomes more harder when an attacker uses a Distributed Denial of Service. The DDoS attacks can't be prevented or detected easily because the malicious traffic from an attacker is seen to be the same as the genuine traffic coming from legitimate users. Tracking down the real attacker is also considered a tough issue as unexpected users' PCs are exploited and used by the attacker as botnets or zombies to launch the attacks on the victim server or device.

Main and Specific Objective

We need to use more intelligent techniques with the help of machine learning and deep learning algorithms for increasing the attack detection efficiency. This intelligent techniques will allow the targeted network or server to recognize the difference between the malicious and

legitimate traffic or packets more efficiently. The main problem that must be fixed in detecting denial of service attacks is to try reducing the false positives and false negatives as much as possible.

Therefore, the objectives of this research paper are , mitigating undiscovered harmful traffic that masquerades legal traffic, investigating an appropriate dataset for the chosen machine learning algorithm and using machine learning in detecting such attacks in order to reduce the false positives and false negatives. Mainly because machine learning strategies can adapt with the changes in attack pattern and tunes themselves accordingly.

Brief Background

The Denial of service (DoS) is a known cyber attack where the legitimate users could not have access on the service anymore. The hacker use some attacking methods in order to exhaust the network resources so that he can seize the service. The whole network and system can also be destroyed and the computer resources such as the RAM, Buffer , and the CPU can be totally occupied as a result of these attacks. When any of these resources is consumed by this unexpected traffic , it forms a bottleneck and the system's performance drops or stops, making it impossible for legitimate users to use it. DoS attacks aren't aimed to steal data or gain access to systems by exploiting security measures.

Literature Review

Several techniques and approaches were developed for Denial of Service (DoS) and Distributed Denial of Service (DDoS) detection and prevention. There are many studies providing various classical and intelligent techniques for these attacks detection. From the developed classical techniques, Y. Ohsita et al. [1] have created a technique for more effectively recognizing malicious SYN traffic by taking into consideration incoming traffic time fluctuation. Regardless of traffic time variations, the approach can detect attacks quickly and correctly. However, attacks at a lower rate, cannot be identified as incoming attacks with lower rates still follows the normal distribution traffic.

Another classical method was developed for detecting the ICMP echo requests that causes flooding attacks. H. Harshita [2] had proposed that the bandwidth of the ICMP packet must be limited to a threshold value of 1000 bits/sec , if any ICMP packet exceeds this amount, the router will reject it.

Traditional machine learning algorithms, such as K-Nearest neighbor, Support Vector Machine, and Naïve Bayes, cannot analyze the data efficiently due to a relatively limited number of samples in the datasets. Regarding this Md. Z. Hasan et al. [3] aimed to use a Deep Convolution Neural Network model to find edge nodes effectively at an early stage. According to the findings of the experiment, DCNN resulted in an accuracy of 99 percent which is better

than Naïve Bayes, K-nearest neighbor and Support Vector Machine, which are basic machine learning algorithms with accuracy scores of 79 percent, 93percent, and 88 percent, respectively.

X.Yuan et al. [4] developed a DDoS attack detection method based on deep learning called Deep Defense. They designed a recurrent deep neural network to learn patterns from sequences of network traffic. They reduced the error rate from 7.517% to 2.103% when compared to the conventional machine learning methods.

R. Basnet et al. [5], applied a deep neural network model to the dataset CSE-CIC-IDS2018 to classify many types of attacks. The authors used 2 dense hidden layers, with the first layer having the same number of units as the number of features recorded in the dataset and the second layer having 128 units. Instead of using the method of testing by separating the dataset into a training set and a test set, they used the n-fold cross-validation method. According to the report, with this model, when classifying DoS/DDoS attacks, the results are always greater than 99%.

Research Methodology

Deep learning techniques are shown to be more accurate and efficient in DDoS attacks detection as, the conventional machine learning techniques are limited by the shallow representation models. Deep learning approaches can automatically extract high-level features from low-level ones and gain powerful representation and inference. Therefore, in this paper Deep learning techniques will be implemented for the purpose of Distributed Denial of Service (DDoS) attacks detection and prevention.

Simulation work

In this paper, I used the dataset CIC-DDoS2019 from the Canadian Institute of Cybersecurity (CIC) [6].

I. Sharafaldin et al. [7], created a new dataset, called CICDDoS2019. This dataset addresses all present inadequacies. They have also offered a new detection and family classification strategy based on a collection of the network flow characteristics, which they have realized while utilizing the produced dataset. Finally, they presented the most significant feature sets for detecting various forms of DDoS assaults together with their weights.

The dataset has been fully labelled with 80 network traffic features collected and computed for all benign flows and denial of service attacks flows.

In this paper, I will only be using NetBIOS, LDAP, PORTMAP and MSSQL from the reflection based attacks, and all the “exploitation based” attacks which are UDP, SYN and UDP-Lag.

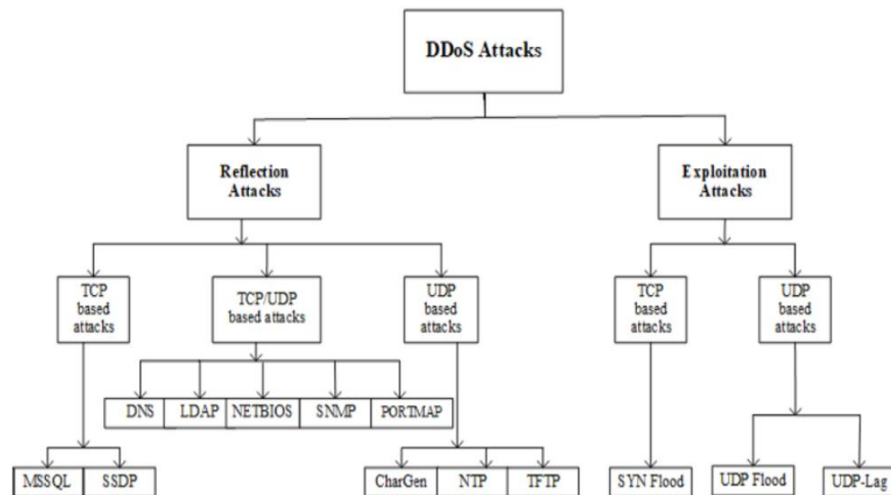


Fig.20 : CIC-DDoS2019 dataset attacks

Data Preprocessing

Data pre processing is the process of converting raw data into a proper format of data by refilling incomplete data and deleting inconsistent, redundant, or noisy data. Several procedures are done on the data to minimize dimensionality and preserve consistency for simplicity of processing. The mentioned stages below were engaged in the data pre-processing stage.

- **Managing Null Values**
- **Handling Categorical Data**
- **Calculating the Standard Deviation**
- **Examining Correlation**
- **Dataset Standardization**
- **Dealing with outliers**

My Proposed approaches

I have proposed 3 different models. In model 1, the model is capable of detecting whether the incoming packet is an attack or not and if so it classifies the type of attack. In this model I've used Convolutional Neural Networks (CNN) for the detection and classification of the DDoS attacks.

For feature extraction from data, two 1D convolutional neural network layers are combined. The first layer is made up of 64 filters of size 3, while the second layer is made up of 32 filters of the same size. The collected features are subsequently processed via a layer of max pooling, which decreases complexity by just choosing most common features. These max

pooling layer attributes are flattened to a 1D vector and input into a deep forward neural network with two hidden layers. The first layer is made up of 15 neurons, whereas the second is made up of 10 neurons. The result is then sent to a final classification layer consists of 6 neuron units.

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv1d_8 (Conv1D)	(None, 32, 64)	256
conv1d_9 (Conv1D)	(None, 30, 32)	6176
max_pooling1d_4 (MaxPooling 1D)	(None, 15, 32)	0
flatten_4 (Flatten)	(None, 480)	0
Hidden_layer_1 (Dense)	(None, 15)	7215
Hidden_layer_2 (Dense)	(None, 10)	160
Output_layer (Dense)	(None, 6)	66

=====
 Total params: 13,873
 Trainable params: 13,873
 Non-trainable params: 0

Fig. 69: Model 1 Structure

In Model 2, the model is also capable of identifying whether the incoming packet is an attack or not and classifying the type of the attack. I've used in this model deep neural networks (DNN).

Firstly, I have developed more than one model to find the more efficient one. I have constructed models with 2 to 6 concealed layers and 32 neuron units each layer. I then started to lower and variate hidden layers from six to three after achieving the highest degree of accuracy. Finally, as shown in Figure [59] below, I achieved the simplest model with the maximum accuracy.

Model: "sequential_6"

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 17)	595
Hidden_Layer_2 (Dense)	(None, 17)	306
Hidden_Layer_3 (Dense)	(None, 17)	306
Output_Layer (Dense)	(None, 6)	108

=====
 Total params: 1,315
 Trainable params: 1,315
 Non-trainable params: 0

Fig. 59: Model 2 Structure

In Model 3, the model proposed is capable of only identifying while the incoming packet is an attack or not without applying any sort of classification. I've also used in this model deep neural networks (DNN).

Similar to the previous model, I have developed more than one model to find the more efficient one. My Deep Neural Network model for attack identification is composed of an input layer of 34 neurons as the number of features in the dataset is 34. The model consists of 3 hidden layers, the three layers consisted of 17 neurons each, and finally the output layer only consists of 2 neurons as it just identify weather the output is an attack or non-attack. I have used relu activation function for the three hidden layers while sigmoid for the output layer.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 17)	595
Hidden_Layer_2 (Dense)	(None, 17)	306
Hidden_Layer_3 (Dense)	(None, 17)	306
Output_Layer (Dense)	(None, 2)	36

Total params: 1,243
 Trainable params: 1,243
 Non-trainable params: 0

Fig. 44: DNN Mode

Simulation Results

I've used the Keras package with the Tensorflow backend to train my neural network models. In Model 1, after data preprocessing and splitting the data set into training set and testing set, I then started the training and validation processes. The training pipeline has 20 epochs and a batch size with 2000 data points. After training the data over the proposed model, The epoch loss and the accuracy against the no. of epochs was shown as follows.

Denial of Service (DoS) Attack Detection Using Machine Learning

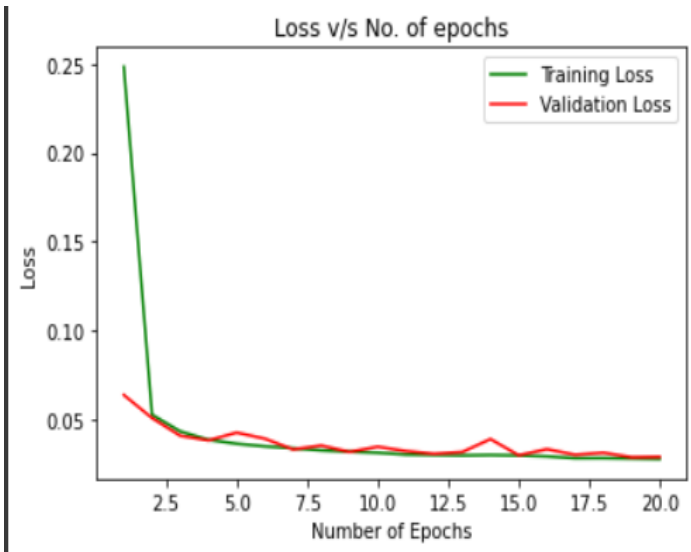


Fig. 71: Plotting Loss v/s No. of epochs

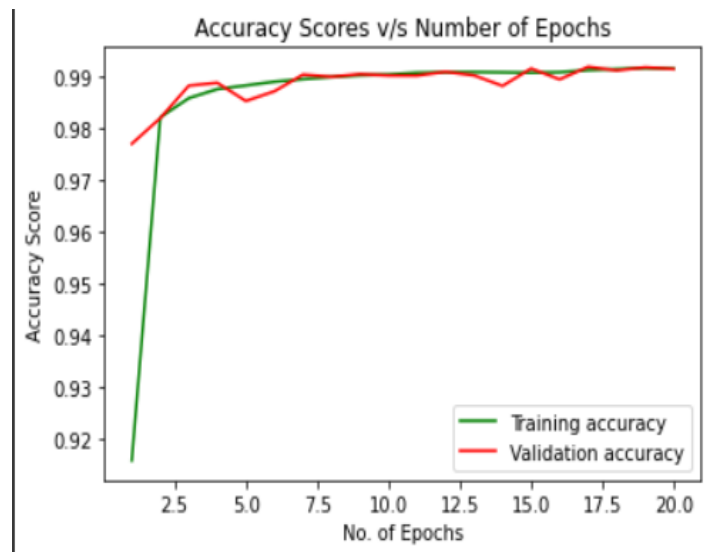


Fig.72: Plotting Accuracy v/s no. of epochs

When Calculating the overall Convolutional Neural Network model 3 accuracy and loss, I got the following results. A loss of approximately 0.0287 with an accuracy of approximately 99.15% .

Finally, we got the confusion matrix and the classification report as shown in Figure [76] below, in order to evaluate the overall model's accuracy.

```

Convolution Neural Network
Accuracy: 99.147751
Confusion Matrix =
[[14430  1  0  0  3  0]
 [ 7 49730  5  8  1  4]
 [ 3  8 49834 48  6  0]
 [ 4  0 46 48897  1 1220]
 [ 3  0  0  6 49881  1]
 [ 1  3  1 871  1 49218]]
Recall = 0.9914775092528818
Classification Report =
              precision    recall  f1-score   support

      0         1.00      1.00      1.00     14434
      1         1.00      1.00      1.00     49755
      2         1.00      1.00      1.00     49899
      3         0.98      0.97      0.98     50168
      4         1.00      1.00      1.00     49891
      5         0.98      0.98      0.98     50095

   accuracy              0.99     264242
  macro avg              0.99      0.99      0.99     264242
 weighted avg              0.99      0.99      0.99     264242

F1 Score = 0.9924209089793093
    
```

Fig. 76: Model 3 Report

In Model 2: The training pipeline has 100 epochs and a batch size with 2000 data points. After training the data over the proposed model, The epoch loss and the accuracy against the no. of epochs was shown as follows.

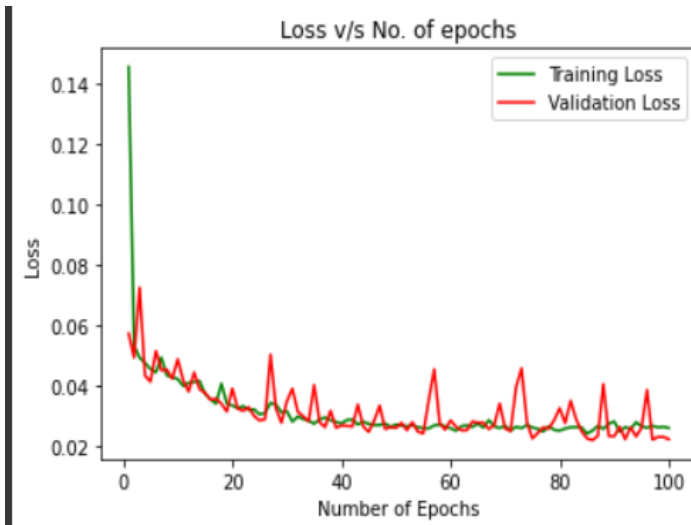


Fig.61: Plotting Loss v/s No. of epochs

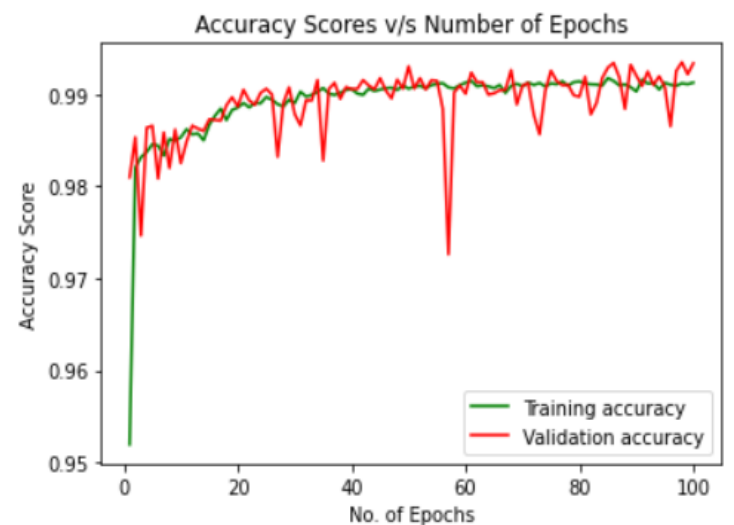


Fig. 62: Plotting Accuracy v/s no. of epoch

When calculating the overall Deep Neural Network model 2 accuracy and loss, I got the following results. A loss of approximately 0.0223 which is a small value for loss and therefore, I got an accuracy of approximately 99.33% . We got the confusion matrix and the classification report as shown in Figure [66] below, in order to evaluate the model's accuracy.

```

Deep Neural Network
Accuracy: 99.333565
Confusion Matrix =
[[14429    5    0    0    0    0]
 [   0 49745    7    0    0    3]
 [    1    5 49853   35    3    2]
 [    1    0   20 49799    1   347]
 [    4    0    0    6 49880    1]
 [    0    0    0 1320    0 48775]]
Recall = 0.9933356544379773
Classification Report =
              precision    recall  f1-score   support

      0              1.00      1.00      1.00      14434
      1              1.00      1.00      1.00      49755
      2              1.00      1.00      1.00      49899
      3              0.97      0.99      0.98      50168
      4              1.00      1.00      1.00      49891
      5              0.99      0.97      0.98      50095

 accuracy              0.99
macro avg              0.99
weighted avg           0.99

F1 Score = 0.9941002919035848

```

Fig. 66: Model Report

In Model 3: The training pipeline has 100 epochs and a batch size with 2000 data points. After training the data over the proposed model, The epoch loss and the accuracy against the no. of epochs was shown as follows.

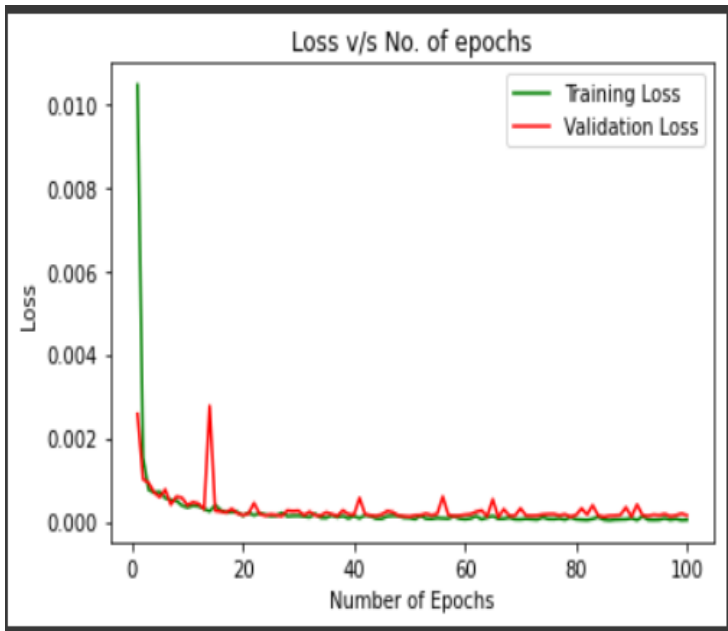


Fig. 47: Plotting Accuracy v/s Epochs

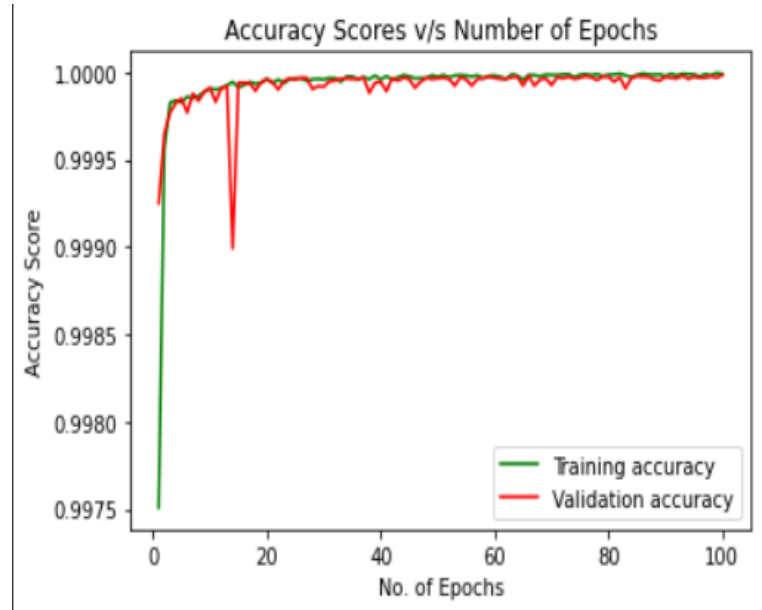


Fig. 46: Plotting Loss v/s Epochs

When finally calculating the overall Deep Neural Network model accuracy and loss, I got the following results. A loss of approximately 1.4784×10^{-4} (0.00014784) which is a very small value for loss and therefore, I got an accuracy of approximately 100%. We got the confusion matrix and the classification report as shown in Figure [51] below, in order to evaluate the model's accuracy.

```

Deep Neural Network
Accuracy: 99.998395
Confusion Matrix =
[[ 14318    3]
 [    2 297127]]
Recall = 0.9999839460587574
Classification Report =
              precision    recall  f1-score   support

      0           1.00        1.00        1.00        14321
      1           1.00        1.00        1.00       297129

   accuracy              1.00              311450
  macro avg           1.00           1.00           311450
 weighted avg           1.00           1.00           311450

F1 Score = 0.9999085056250747

```

Fig. 51: Model 1 Confusion Matrix

Comparative Analysis

A system was suggested by G. Usha et al.[8] with the goal of providing an efficient method for detecting DDoS assaults in a network. They have also stated that, in comparison to other methodologies and intrusion detection systems, Artificial Intelligence has been widely employed. To identify and categorize the DDoS attacks, they had employed a variety of machine learning methods and a deep learning architecture using the Convolutional Neural Networks (CNN).

They had used the CICDDoS2019 dataset in training their proposed models and compare their accuracies in classifying the DDoS attacks.

They have applied their different models' algorithms on the CICDDoS2019 dataset which was reduced to 34 different characteristics of incoming traffic after pre-processing. They then trained the algorithms on the training dataset, and compared their algorithms after that based on their behavior over the test dataset, taking into account recall, f1 scores, and accuracy.

They obtained from the Convolutional Neural Network (CNN) classifier an overall accuracy of 83.89 percent, as it performed extremely badly in the categorization of UDPLag attacks, obtaining an F1 score of 0.12. It also performed badly with Portmap attacks with the same F1 score of 0.12. It had an acceptable performance at classifying NetBIOS, earning an F1 score of 0.68. It performed excellent at identifying all of the other classes, earning a macro-average F1 score of 0.723.

In this paper, I used the same dataset CICDDoS2019 from the Canadian Institute of Cybersecurity (CIC). I performed the data pre-processing in the same manner as done in [8], in order to get the same reduced dataset with the 34 features. I then compared results obtained from my different proposed models to that obtained by G. Usha et al. in [8]. In this paper, I am only interested in the proposed deep learning classifier using the Convolutional Neural Networks (CNN), and I will compare my results only with this model.

I have proposed 3 different models. In model 1, the model is capable of detecting whether the incoming packet is an attack or not and if so it classify which type of attack. I removed the Portmap and UDPLag attacks from the dataset as I have discovered that they where the reason that lowered the accuracy in the model proposed by G. Usha et al.[8]. I used the same convolutional neural network structure used in [8] but I have made a small modification by adding a Forward Deep Neural Network for the data to enter after coming out from the pooling layer.

This model achieves better accuracy and performs perfectly than that proposed by G. Usha et al.[8] in attacks classifications. I got an overall accuracy of 99.15%, while achieving F1 score of 1 for classifying BENIGN, NETBIOS , LDAP , Portmap. It also achieved F1 score of 0.98 in classifying MSSQL and Syn attacks.

In model 2, the model is capable of Identifying whether the incoming packet is an attack or not and classifying the type of the attack. I have also used in this model the dataset with the Portmap and UDPLag attacks removed. I used in this model a Feed forward Deep neural Network of 3 hidden layers.

This model achieves also a better accuracy and performs perfectly than the proposed model in [88]. I got an overall accuracy of 99.33%, while achieving F1 score of 1 for classifying BENIGN, NETBIOS, LDAP, Portmap. It also achieved F1 score of 0.98 in classifying MSSQL and Syn attacks, which is approximately the same F1 scores achieved by my modified CNN model.

In model 3, the model proposed is capable of only identifying whether the incoming packet is an attack or not. I have used here the dataset without removing the Portmap and UDPLag attacks, but I have modified the label feature to only contain a non-attack with label '0' or an attack with label '1'.

This model achieves also a perfect accuracy of approximately 100%. The model achieved F1 score of 1 for classifying both BENIGN (non-attack) and MALIGN(attack).

Conclusion

Deep learning techniques can automatically extract high level features from low level ones and are more powerful than conventional machine learning techniques. We can also conclude that artificial neural networks are more perfect dealing with sequential data than convolutional neural networks (CNN) does, and this appeared significantly when the (CNN) model performs better after adding the Feed Forward Deep Neural Network before the final sequential classification layer and after the max pooling layer. In this research paper, I've proposed three models. Model 1 performed DDoS attack detection and classification of the attack's type using convolutional neural networks (CNN) for its training and testing. Model 2 also performed DDoS attack detection and classification using feed forward deep neural networks (DNN) for training and testing. Model 3 performed only DDoS attack detection as it only identifies whether the incoming packet is an attack or not, using also feed forward deep neural network (DNN). The developed three models were applied to CIC-DDoS2019 dataset after preprocessing. The performance of these three models was compared with similar testing conditions proposed in [8] and it was found that the three models achieved an overall accuracy of, Model 1: 99.15%, Model 2: 99.33%, and Model 3: 99.99%.

Future works

The use of neural networks is restricted to the academic and scientific worlds. As a result, by combining this technique in the form of a hybrid framework or architecture, with other neural networks, an useful attack detection system with extremely high learning rate, speedy attack detection and low error rate may be constructed. Also for the CIC-DDoS2019 dataset we can check for the Portmap attack features problem and increase the number of UDPLag attacks in the dataset as they are lowering the model performance when included.

Main References

- [1] Y. Ohsita, S. Ata, and M. Murata, “Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically,” *IEEE Xplore*, Nov. 01, 2004. <https://ieeexplore.ieee.org/document/1378371?arnumber=1378371> (accessed Dec. 17, 2021).
- [2] H. Harshita, “Detection and Prevention of ICMP Flood DDOS Attack,” *International Journal of New Technology and Research*, vol. 3, no. 3, p. 263333, Mar. 2017, Accessed: Feb. 11, 2022. [Online]. Available: <https://www.neliti.com/publications/263333/detection-and-prevention-of-icmp-flood-ddos-attack>.
- [3] Md. Z. Hasan, K. M. Z. Hasan, and A. Sattar, “Burst Header Packet Flood Detection in Optical Burst Switching Network Using Deep Learning Model,” *Procedia Computer Science*, vol. 143, pp. 970–977, 2018, doi: 10.1016/j.procs.2018.10.337.
- [4] X. Yuan, C. Li, and X. Li, “DeepDefense: Identifying DDoS Attack via Deep Learning,” *IEEE Xplore*, May 01, 2017. <https://ieeexplore.ieee.org/abstract/document/7946998> (accessed Jan. 24, 2021).
- [5] R. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, “Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks.” Accessed: Apr. 16, 2022. [Online]. Available: <http://isyou.info/jisis/vol9/no4/jisis-2019-vol9-no4-01.pdf>
- [6] “DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” *www.unb.ca*. <https://www.unb.ca/cic/datasets/ddos-2019.html>
- [7] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,” *2019 International Carnahan Conference on Security Technology (ICCST)*, Oct. 2019, doi: 10.1109/ccst.2019.8888419.
- [8] G. Usha, M. Narang, and A. Kumar, “Detection and Classification of Distributed DoS Attacks Using Machine Learning,” *Computer Networks and Inventive Communication Technologies*, pp. 985–1000, 2021, doi: 10.1007/978-981-15-9647-6_78.

