الجـمـهورية الجزائرية الديمقراطية الشعبية
**République Algérienne démocratique et populaire**

وزارة التعليم الــعــالي و البحــث العـلمــي
**Ministère de l'enseignement supérieur et de la recherche scientifique**

جــامعة سعد دحلب البليدة
**Université SAAD DAHLAB de BLIDA**

كلية التكنولوجيا
**Faculté de Technologie**

قسم الإلكترونيـك
**Département d'Électronique**

**Master Thesis**

In Telecommunication

Option : Network &Telecommunications

Presented by

Mahi Sid Ali

&

Benslim Borhan Eddine

# Network intrusion detection system using deep learning

Proposed by: Mr. Mehdi Merouane

2022-2023

# Acknowledgement

We would like to take this opportunity to express our heartfelt gratitude to the Almighty God for His infinite blessings, guidance, and unwavering support throughout the entire journey of this thesis. His divine presence has been the source of strength and inspiration, enabling us to overcome challenges and reach this significant milestone in our academic pursuit.

We are deeply indebted to our esteemed promoter, Mr. Mehdi Merouane, whose unwavering dedication, extensive knowledge, and invaluable guidance have played a pivotal role in shaping the direction of this research. His expertise and continuous encouragement have been instrumental in refining our ideas, enhancing the quality of our work, and pushing us to strive for excellence.

We would like to extend our sincere appreciation to the distinguished members of the jury who have graciously agreed to evaluate and assess this thesis. Their valuable time, expertise, and critical feedback will undoubtedly contribute to the enrichment and refinement of this research.

We extend our sincere gratitude to the dedicated faculty members of the Electronic Department, particularly Mr. Hocine Aitsaadi, for their outstanding teaching, mentorship, and support, which have greatly influenced our academic journey and personal development.

Lastly, we would like to express our heartfelt thanks to all those who have contributed to the realization of this thesis. To our families, friends, and loved ones, your unwavering belief in our abilities, encouragement, and endless support have been the driving force behind our success. We are immensely grateful for your presence in our lives.

We express our profound gratitude to all those mentioned above for their invaluable contributions, guidance, and unwavering support throughout our research journey. We are humbled by the blessings and guidance from our divine Creator, whose presence has greatly enriched our academic experience and made this achievement possible.

# Dedication

In heartfelt appreciation and gratitude, we dedicate this thesis to the individuals who have played a significant role in our academic journey. Their unwavering support, love, and encouragement have been invaluable throughout the ups and downs, guiding us towards this achievement.

First and foremost, I offer my deepest gratitude to my God, Allah, for His guidance, blessings, and unwavering presence in my life.

To my beloved parents, [Toufik] and [Benmeradi Leila], your unwavering belief in my abilities, endless sacrifices, and unconditional love have been the driving force behind my success.

To my dear brother Racim and my young sisters, your constant support, encouragement, and shared experiences have made this journey all the more memorable and fulfilling.

To my respected aunt, [Benmeradi Zoubida] and her respectable husband [ taher stanboul], I hope that God will relieve you and grant your success, and to all my aunts, uncles, and grandparents, your love, guidance, and unwavering support have been a source of inspiration and strength.

I also extend my dedication to my dedicated study partner [Benslim Borhan Eddine], and all my dear friends, who have shared this journey with me, offering support, sharing knowledge, and pushing me to strive for excellence.

Sid Ali

## Dedication

To my incredible parents, friends.

I am humbled and honored to dedicate my Master's degree to each and every one of you who have played a significant role in my journey.

To my parents, [Abdelkarim] and [khenfer Louiza] your love, guidance, and sacrifices have been the cornerstone of my success.

To my dear friends, you have been my pillars of strength throughout this educational endeavor. Your support, encouragement and your belief in my potential has been a constant source of motivation, and I cherish the memories we have created together along the way.

To my dedicated study partner [Mahi Sid Ali] thank you for sharing this journey with me. Your intellectual curiosity, collaborative spirit, and relentless pursuit of excellence have inspired me to push my boundaries, not only to enrich my education also forged lifelong connections that I deeply value.

To all of you, your presence in my life has been a gift. Whether it was a comforting word, a lending hand, or a word of encouragement, you have made an indelible impact on my success

With profound appreciation and deep affection.

Borhane Eddine

## ملخص

يلعب اكتشاف اختراقات الشبكة دورا حاسما في ضمان الأمن السيبراني وحماية المؤسسات من الأضرار المحتملة. يركزها البحث على تعزيز أمن شبكات الكمبيوتر من خلال الكشف الفعال عن عمليات الاقتحام والتخفيف من حدتها مع التركيز بشكل خاص على مكافحة هجمات رفض الخدمة الموزعة (DDoS). يتم استكشاف تقنيات التعلم العميق بما في ذلك الشبكات العصبية العميقة (DNN) والشبكات العصبية التلافيفية (CNN) والشبكات العصبية المتكررة (RNN)، لتطوير نظام كشف التسلل (IDS). يتم تقييم أداء هذه النماذج باستخدام مجموعة بياناتCICDDoS2019 كمعيار للكشف عن هجمات DDoS للشبكة. نقوم بتقييم دقيق لنماذج التعلم العميق المقترحة، مما يدل على أدائه المتفوق في الكشف الدقيق عن عمليات اقتحام الشبكة بدقة عالية واستدعاء ودرجة F1. يوضح هذا البحث فعالية شبكات DNN في تحسين أنظمة كشف التسلل بدقة 99,76%، وتمكين المؤسسات من اكتشاف اختراقات الشبكة والاستجابة لها، وتعزيز دفاعات الأمن السيبراني.

**الكلمات المفتاحية:** اقتحام الشبكة، نظام كشف التسلل، التعلم العميق، الشبكات العصبية العميقة، الشبكات العصبية التلافيفية، الشبكات العصبية المتكررة، هجماتDDoS.

## Résumé

La détection des intrusions dans les réseaux joue un rôle crucial dans la garantie de la cybersécurité et la protection des organisations contre les dommages potentiels. Cette recherche se concentre sur l'amélioration de la sécurité des réseaux informatiques en détectant et en atténuant efficacement les intrusions, avec un accent particulier sur la lutte contre les attaques par déni de service distribué (DDoS). Les techniques d'apprentissage profond, notamment les réseaux neuronaux profonds (DNN), les réseaux neuronaux convolutifs (CNN) et les réseaux neuronaux récurrents (RNN), sont explorées pour développer un système de détection d'intrusion (IDS). La performance de ces modèles est évaluée en utilisant l'ensemble de données CICDDoS2019 comme référence pour la détection des attaques DDoS en réseau. Nous évaluons rigoureusement les modèles d'apprentissage profond proposés, démontrant leur performance supérieure dans la détection précise des intrusions de réseau avec une précision, un rappel et un score F1 élevés. Cette recherche démontre l'efficacité des DNN avec 99.76% de précision dans l'amélioration des systèmes de détection d'intrusion, permettant aux organisations de détecter et de répondre aux intrusions de réseau, renforçant ainsi les défenses de cybersécurité.

**Mots clés :** Intrusions dans les réseaux, système de détection des intrusions, apprentissage profond, réseaux neuronaux profonds, réseaux neuronaux convolutifs, réseaux neuronaux récurrents, attaques DDoS.

## Abstract

The detection of network intrusions plays a crucial role in ensuring cyber security and protecting companies from potential damages. This research focuses on enhancing the security of computer networks by effectively detecting and mitigating intrusions, with a specific emphasis on combating distributed denial of service (DDoS) attacks. Deep learning techniques, including deep neural networks (DNN), convolutional neural networks (CNN), and recurrent neural networks (RNN), are explored for developing an intrusion detection system (IDS). The performance of these models is evaluated using the CICDDoS2019 dataset as a benchmark for network DDoS attack detection. We rigorously evaluate the proposed deep learning models, demonstrating their superior performance in accurately detecting network intrusions with high precision, recall and F1 score. This research demonstrates the effectiveness of DNN in improving intrusion detection systems, with 99.76% accuracy empowering organizations to detect and respond to network intrusions, strengthening cyber security defenses.

**Keywords:** NIDS, IDS, Deep learning, DNN, CNN, RNN, DDoS attacks.

# List of abbreviations

|  |  |
|---|---|
| **AC:** | *Accuracy* |
| **ACK:** | *Acknowledge* |
| **AGI:** | *Artificial general intelligence* |
| **AIDS:** | Anomaly *Based on Intrusion Detection System* |
| **CNS:** | *Communication and Network Security* |
| **CNN:** | *Convolutional Neural Network* |
| **DDOS:** | *Distributed Denial of Service* |
| **DOS:** | *Denial of Service* |
| **DNN:** | *Deep Neural Network* |
| **DNS:** | *Domain Name System* |
| **FPR:** | *False Positive Rate* |
| **GRU:** | *Gated recurrent unit* |
| **HIDS:** | *Host Based on Intrusion Detection System* |
| **IDS:** | *Intrusion Detection system* |
| **IEEE:** | *Institute of Electrical and Electronics Engineers* |
| **IOT:** | *Internet of Things* |
| **LDAP:** | *Lightweight Directory Access Protocol* |
| **LSTM:** | *Long short term memory* |
| **Malware:** | *Malicious Software* |
| **MSSQL:** | *Microsoft Structured Query Language* |
| **NetBios:** | *Network Basic Input Output System* |
| **NIDS:** | *Network Based on Intrusion Detection System* |

**NTP:** *Network Time Protocol*

**POD:** *Ping of Death*

**PR:** *Precision*

**RC:** *Recall*

**ReLU:** *Rectified Linear Unit*

**RNN:** *Recurrent Neural Network*

**ROC:** *Receiver Operating Characteristic*

**RPC:** *Remote Procedure Call*

**SIDS:** Signature *Based on Intrusion Detection System*

**SNMP:** *Simple Network Management Protocol*

**SSDP:** *Simple Service Discovery protocol*

**SYN:** *Synchronization*

**SQL:** *Structured Query Language*

**TFTP:** *Trivial File Transfer Protocol*

**TPR:** *True Positive Rate*

**TCP:** *Transmission Control Protocol*

**UDP:** *User data gram protocol*

# Table of contents

# List of figures

# List of tables

# General introduction

A computer system is a system that connects two or more computing devices for transmitting and sharing information. With the increased reliance on technology, it is becoming increasingly important to safeguard all aspects of internet information and data. Data integrity has become one of the most critical issues for organizations to address as the internet and computer networks increase in size [1].

With the world as connected as it is, anyone who does anything online should be aware of the possibility of a cyber- attack. This is especially critical for businesses because there is a lot of risk, including the protection of their customers. Learning more about cyber- attacks, can significantly improve your online safety [2].

The more we rely on technology, the more critical cyber-security becomes. Cyber security is a collection of technological approaches and processes to protect networks and digital information against malicious activities. It is critical for protecting organizations and individuals from spammers and cyber criminals, as breaches can result in financial losses and trust erosion [3]. In order to achieve this, security solutions such as firewalls, anti-virus software, and intrusion detection systems (IDS) are frequently implemented in systems and devices. IDS is an important tool of cyber security that monitors network traffic to identifies malicious and harmful actions that couldn't be identified by traditional firewall [4]. However IDS that rely on traditional techniques are unable to detect new or advanced attacks and suffering from high false positive and negative rate.

To address these limitations, more advanced techniques such as machine learning and deep learning-based intrusion detection have been developed to provide good performance with high accuracy by reducing false positive and negative rates. These techniques make IDS adaptable to new and complex attacks and make the process automatic and fast without slowing down the network. Deep learning and machine learning algorithms open new doors for intrusion detection systems to develop new technologies and approaches in real time to evaluate the performance and protect computer networks and systems from new cyber-attacks.

This study focuses on using the CIC-DDOS-2019 dataset, which contains both normal and malicious traffic. It aims to collect, balance, process the dataset, codify the output based on whether the traffic is malicious or benign, normalize, build models, train and test them, and evaluate trained models. The main contribution is to work with the most recent dataset with a specific type of security attack.

This thesis is organized into three chapters, each addressing specific aspects of the research topic. Chapter 1 serves as the literature review, presenting a comprehensive foundation for understanding the significance of intrusion detection and the potential of deep learning to enhance intrusion detection systems.

Chapter 2 focuses on the methodology employed in this study, offering a detailed explanation of the various steps involved in developing an Intrusion Detection System (IDS) using deep learning techniques. It encompasses aspects such as dataset selection and preprocessing, as well as the implementation of Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) within the IDS framework.

The final chapter, Chapter 3, delves into the technical environment and libraries utilized throughout the research. Additionally, this chapter presents the results of our proposed solution, providing an analysis and evaluation of the effectiveness of the developed IDS based on deep learning techniques.

# chapter 1

# Literature review

## 1.1 Introduction

The importance of technology in our everyday routine cannot be denied in today's digital world. We rely on it for communication, work, shopping, or accessing information. While making things easier for us, technology has simultaneously opened up new ways for cyber threats and attacks.

Since the number and severity of cyber-attacks have increased in recent years, cyber-security has become a critical problem. Cyber threats are getting increasingly sophisticated, and the consequences can be serious, including data breaches, financial losses, and reputational damage.

This chapter aims to provide a comprehensive overview of cyber-attacks and intrusion detection techniques, as well as the advantages and limitations of using deep learning for intrusion detection. It also highlights the importance of effective intrusion detection to maintain network security in the face of evolving cyber threats.

## 1.2 Overview of cyber-attacks

Any effort to obtain unauthorized access to a computer, computing system, or computer network with the goal of doing damage is considered a cyber-attack. A cyber-attack attempt by any individual or group of individuals regarded as cybercriminals using various attack strategies from anywhere to disable, disrupt, destroy, or control computer systems, as well as to change, block, delete, modify, or steal data stored on these systems [5].

Figure **1.1** shows some sources of cyber threats.

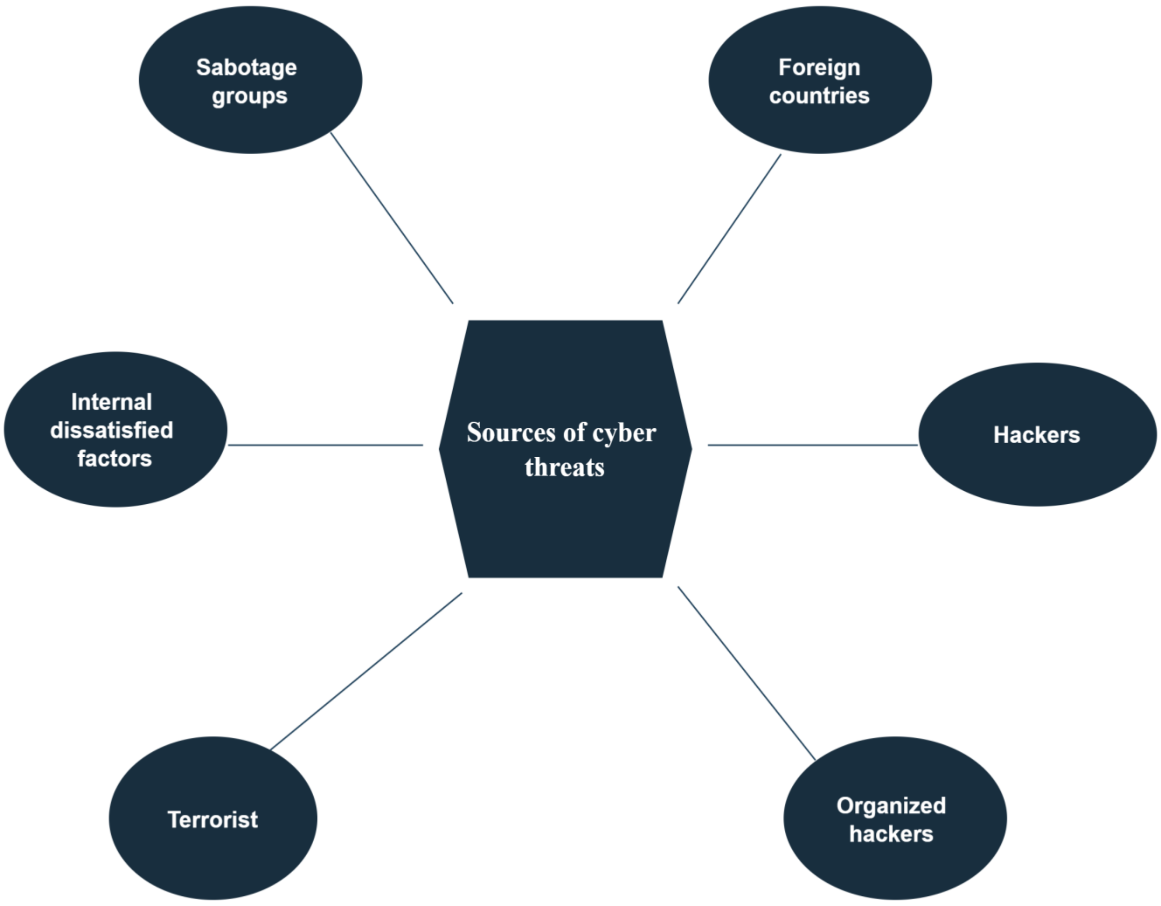

**Figure 1.1:** The source of cyber threats **[**6]

Cyber-attacks are typically either criminal or political in nature. Adversaries might be private individuals, state actors, or criminal organizations. But the major reason for these attacks is to look at the goals behind each of them. Because criminals do not always look for the same thing. Some cybercriminals want money or information, while others may simply wish to cause problems. Then there are individuals who attack systems with the intention of destroying them for personal gain, such as angry ex-employees [7].

The cyber-attacks had a popular objective in damaging network infrastructure availability by understanding the most common cyber-attacks.

Individuals and organizations can take proactive action to prevent the impact of cyber-attacks, figure **1.2** illustrates the important cyber-attacks types. The most common types of cyber-attacks are:

- **Phishing**

Phishing is a type of cyber-attack that uses spoofed email, SMS, phone, social media, and social engineering techniques to trick a victim into sharing important information, such as passwords or account numbers, or into downloading a malicious file that would install viruses on their computer or phone [7].

- **Malware**

Malware, or malicious software, is any program or code designed to do harm to a computer, network, or server. This includes ransomware, Trojans, spyware, viruses, worms, key-loggers, bots and any other type of malware assault that exploits software in a malicious way. It can appear when the victim clicks on an unknown risky link via an untrusted email attachment, which can install harmful and malicious software [4].

- **Spoofing**

Spoofing is a technique in which a cybercriminal disguises as a known or trusted source. By doing so, the adversary gains access to the target's systems or devices, with the ultimate objective of stealing information, extorting money, or putting malware or other malicious software on the device [8].

- **Denial of service (DOS) and distributed denial of service (DDOS)**

A denial-of-service can flood the networks systems or servers with false requests to drain out the bandwidth and affect the resources.

A DDoS attack is also an attack on system’s resources, but it is launched from a large number of other host machines that are infected by malicious software controlled by the attacker through this the system will be completely offline and will take a long time to respond [9].

There are different types of DDOS attack such:

- **TCP syn flood attack**

This attack target TCP/IP protocol, which is responsible for establishing connection between devices on the internet. During SYN flood attack, the attacker floods the target with a massive amount of TCP SYN packet without completing the handshake process by sending the final ACK packet. This flood the target server with half open connections, leading to denial of service making the victim inaccessible to legitimate users.

- **Ping of death**

Ping of death (POD) is type of DDOS attack in which the attacker attempt to crash or freeze the target system by sending abnormally large “ping” packet to the victim. This packet exceeds the maximum permitted size for IP packets (65535 bytes)

Ping of death (POD) is type of DDOS attack in which the attacker send abnormally large ping packet to the system with the maximum allowable size for IP packet 65535 bytes, causing it to crash or freeze the target system [10].

- **Botnets**

Botnets are networks of infected computers or devices controlled by cyber attacker in order to lunch DDOS attack. The compromise devices are referring to bots or zombies used to overload the bandwidth and processing capacity, spamming and stealing information. Botnets pose a huge danger to online security and have the potential to cause significant harm to individuals, businesses, and even entire countries [11].

- **SQL injection**

Structured query language, when you use SQL injection as a cyber-attack, you are trying to take control of, and maybe steal from a database. Cybercriminals exploit vulnerabilities in data-driven applications by introducing malicious code into databases, gaining access to sensitive information [7].

- **DNS tunneling**

DNS tunneling is a method of encoding and transmitting data packets through the Domain Name System (DNS) protocol. The idea is to hide these packets within DNS queries and responses. This technique can be used to communicate with remote servers without being detected. However, it’s often used by attackers to extract sensitive information from compromised system [12].

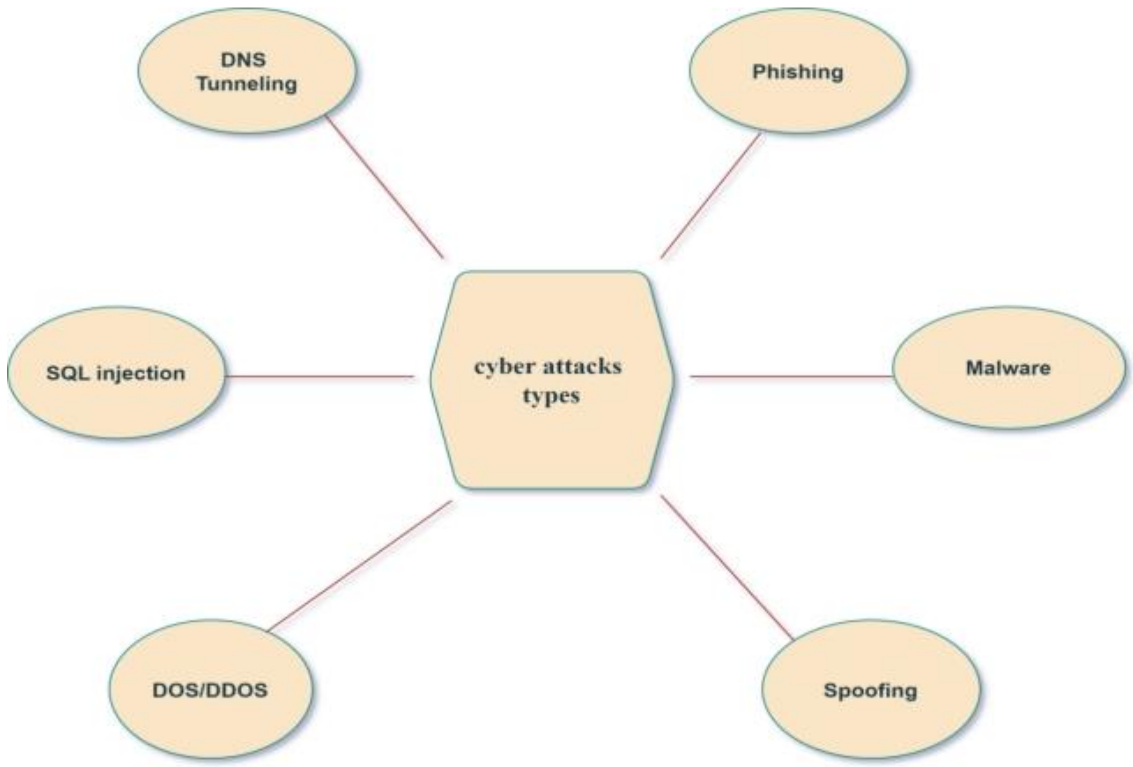

**Figure 1.2:** Cyber-attacks types [6]

## 1.3 Intrusion detection system

Cyber security is vital to protecting our sensitive information, digital assets, and privacy from harmful cyber activity. One essential component of an effective cyber-security strategy is an intrusion detection system. An IDS is a software or hardware-based tool designed to find unexpected events that may indicate an attack will happen, is happening, or has happened by monitoring system and network activity.

The main objective of an intrusion detection system is to detect and respond quickly to unauthorized actions, reducing the damage caused by cyber-attacks by providing real time alerts and notifications. In fact, the classification of IDS is dependent on various steps, including deployment objective, detection methods, and reactions, in order to achieve the basic security level.

## 1.4 Deployment

When it comes to intrusion detection systems two main approaches are commonly employed: network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS). Figure **1.3** shows the deployment of IDS.

- **Network based IDS (NIDS)**

The NIDS focuses on monitoring and analyzing network traffic for potential security threats. This type is effective at detecting attacks that are aimed at the network layer, such as denial-of-service (DoS) attacks. It is considered as passive devices that can be retrofitted to existing networks with minimal effort.

- **Host based IDS (HIDS)**

HIDS monitors the activities on individual hosts or endpoints, such as servers or mobile devices. HIDS analyzes system events like file changes, logins and network connections. Moreover, it has the ability to detect Trojans, malwares and viruses [13].

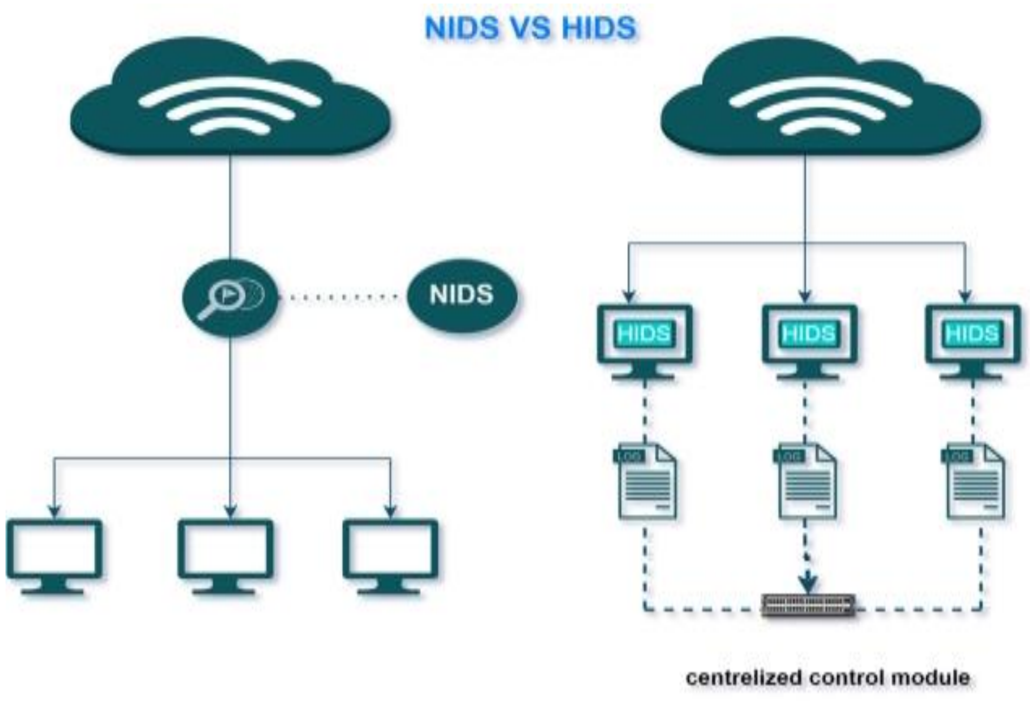

**Figure 1.3:** Network and host-based IDS topology [14]

## 1.5 Detection methods and responses

Detecting intrusions is based on two main categories:

- **Signature based IDS (SIDS)**

Signature based or Misuse IDS uses a database of known attack signatures to identify malicious traffic or system activity by monitoring packets. They are useful for Identifying attacks that have been previously identified. However, there is a lag time between a new threat being discovered and its signature being applied to IDS, making it difficult to detect it. This kind of detection is effective tool for detecting known attacks with low false positive rate and easy to implement and maintain [15].

- **Anomaly based IDS (AIDS)**

Use a machine learning algorithm to identify abnormal behavior. In this method the IDS will compare incoming traffic with the normal patterns of traffic to detect any unusual behavior. When a pattern deviates from the regular baseline, it triggers an alarm [16].

In addition, IDS have two detection responses: alerting response (passive) and defensive response (active). The passive type can find out the attack, log then generate an alert, on the other hand the active IDS can detect, log, notify and block threats in real time [4]. Figure **1.4** is a diagram representing the taxonomy of intrusion detection system.

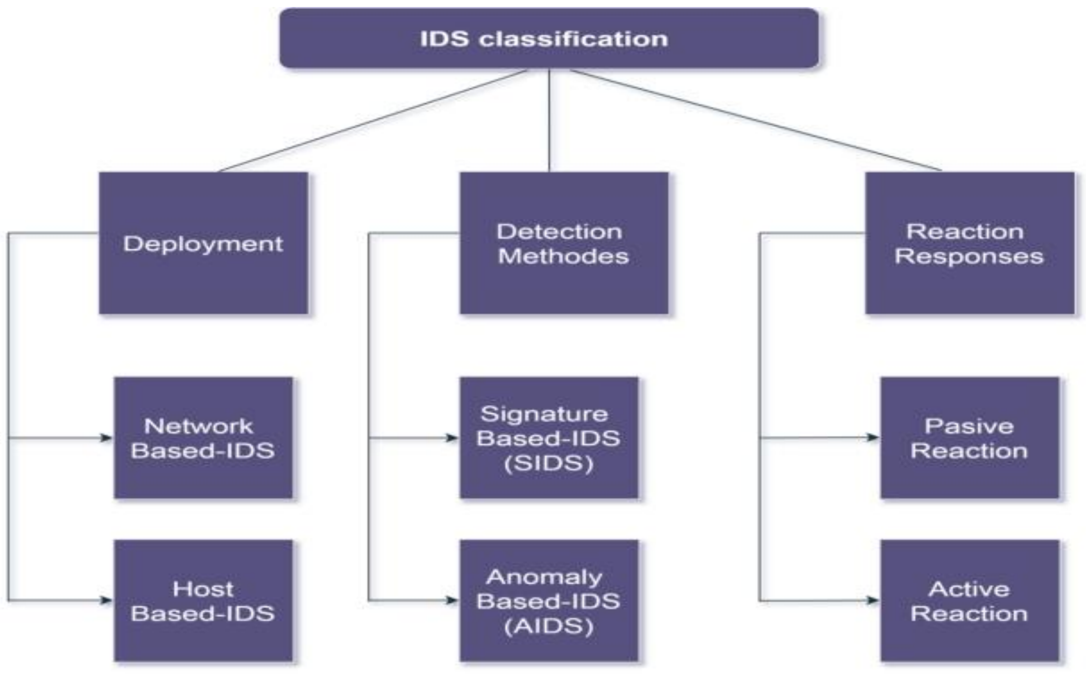

**Figure 1.4:** Taxonomy of intrusion detection system

While intrusion detection system provides significant benefits in terms of network security, they do have certain limitations. Signature based IDS are unable to detect unknown and complex attacks. To maintain accuracy, signature-based IDS necessitate regular updates to their signature database. Moreover, anomaly-based IDS generate high number of false positives making it difficult to distinguish between genuine threats and false alarms. Additionally, they require substantial resources to operate effectively [16].

## 1.6 Intrusion detection using deep learning

Deep learning is a subset of machine learning that uses artificial neural networks to simulate the learning process of the human brain, it is based on the idea that algorithms can learn to recognize patterns and make predictions by analyzing large amounts of data.

Deep learning algorithms are capable of processing large and complex datasets with many layers of abstraction, unlike typical machine learning algorithms, which are made to handle relatively basic datasets with a small number of features.

Deep learning is being used in a wide range of applications, including speech and picture recognition, natural language processing, self-driving automobiles, and healthcare. It has also shown promise in the field of cyber security, where it is being used to develop more advanced intrusion detection systems and identify new and emerging cyber threats.

However, there are also challenges in applying deep learning, such as the need for large amounts of high-quality data to learn accurate representations of the underlying patterns and relationships in the data [17].

Deep learning has shown promising results in the field of intrusion detection systems IDS which are used to detect unauthorized access and malicious activities in computer networks. By using deep learning algorithms in IDS, it can learn to extract relevant features from raw data automatically and make accurate predictions in real-time, which reduces the need for manual feature engineering, and it also can adapt to the changes in network traffic, allowing IDS systems to continue accurately detecting security threats even as network traffic patterns change [18].

In the last few years, there were many projects in this domain with different architectures, datasets and across various domains, including general networks and IOT networks.

2015 saw the first major case of deep learning being used for intrusion detection. A presentation titled "Deep Learning for Anomaly Detection in Network Traffic" at the IEEE

Conference on Communications and Network Security (CNS), however, deep learning techniques have been widely explored for intrusion detection since then leading to advancements in architectures, algorithms, and evaluation methodologies such as:

- Project: "Deep IDS"

Deep Learning Intrusion Detection System" (2017), using NSL-KDD dataset and a combination of convolution neural networks (CNNs) and long short-term memory (LSTM) networks.

- Project: "Deep Packet"

A Novel Approach for Intrusion Detection in Network Traffic Using Deep Learning" (2018) they evaluated their model on the CICIDS2017 where raw packet payloads were processed by a deep neural network.

- Project: "DeepAD"

Intrusion Detection in IOT Networks Using Deep Auto encoders" (2019) this project was to develop an intrusion detection system for Internet of Things (IOT) networks using deep auto encoders, The researchers collected IOT network traffic data and used it to train their deep learning model.

- Project: "Deep Traffic

Deep Learning-Based Intrusion Detection System for Autonomous Vehicles" (2021), the goal of this research was to create a deep learning-based intrusion detection system for autonomous cars, The study's deep learning model was trained using real-world traffic data gathered from autonomous cars [19].

## 1.7 Different deep learning architectures for intrusion detection

Deep learning is a subfield of machine learning that uses artificial neural networks that are inspired by the structure and the function of the human brain. A neural network is composed of layers of interconnected processing units, or neurons, that receive input signals (the input layer), process them (hidden layers), and generate output signals (the output layer). The term "deep" refers to the use of multiple layers in the neural network Architecture, and the connection between layers is the key aspect of its architecture as each layer processes the output of the previous processes [17]. Figure **1.5** shows the artificial neural network architecture.

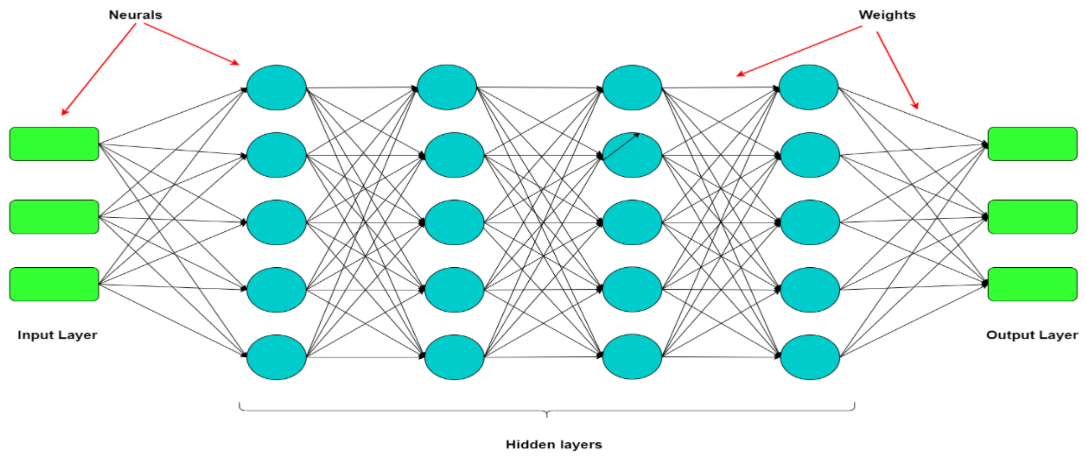

**Figure 1.5:** Artificial neural network [20]

There are several deep learning architectures that have been used in intrusion detection, and they are all different in their design principles, memory mechanisms, and the capacities for recognizing various kinds of patterns and interdependence in data. The choice of the architecture will depend on the details of the intrusion detection task and the type of network traffic data being analyzed. The most commonly used architectures are:

- The Convolutional neural network (CNN) is a multilayer neural network that was inspired by the animal visual cortex. The architecture is particularly used in image processing and image recognition, but in the early days it was concentrated on reading handwritten characters (neural language processing), including postal codes. Early layers in a deep network identify characteristics (like edges), while later layers reassemble this information into higher-level input qualities. The architecture is multilayer, and the principle is to extract features from an input rather than classify it. For an image example as shown below in figure **1.6**. A convolutional layer receives the picture's receptive fields as input and uses them to extract information from the input image. The next stage is pooling, which down samples the extracted features to make them less dimensional while still preserving the most crucial data (usually by max pooling).

A fully linked multilayer perception is then fed by a further convolution and pooling stage. A group of nodes (in this example, one node per detected number) that identify elements of the picture make up the network's final output layer. The network is trained via back propagation; the CNN has been successfully applied to video recognition, natural language processing and various tasks of classification. In intrusion detection, CNNs can be used to extract spatial patterns from network traffic data, such as packet headers or payload information [20].

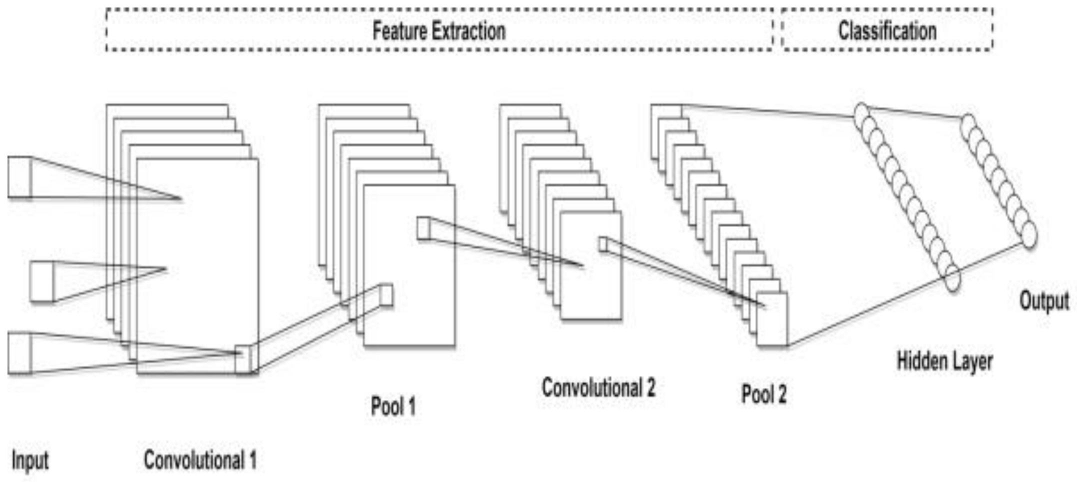

**Figure 1.6:** CNN architecture principal [20]

Recurrent neural networks (RNNs) are one of the fundamental network designs that other deep learning architectures are constructed from. The purpose of an RNN is to manage sequential data and record temporal dependencies. RNNs, as opposed to feed forward networks, feature recurrent connections that let data persist and move over time, this allows RNNs to maintain memory of past inputs and model problems over time. Because of this, they are appropriate for modeling network traffic data with a sequential character, like packet arrival times or session logs. RNN can have many architectures, the main differentiator in a network is feedback, which may come from a hidden layer, the output layer, or a mix of both. It is generally used for speech recognition, handwriting recognition and other classification tasks [20]. Figure **1.7** shows the recurrent neural network architecture.

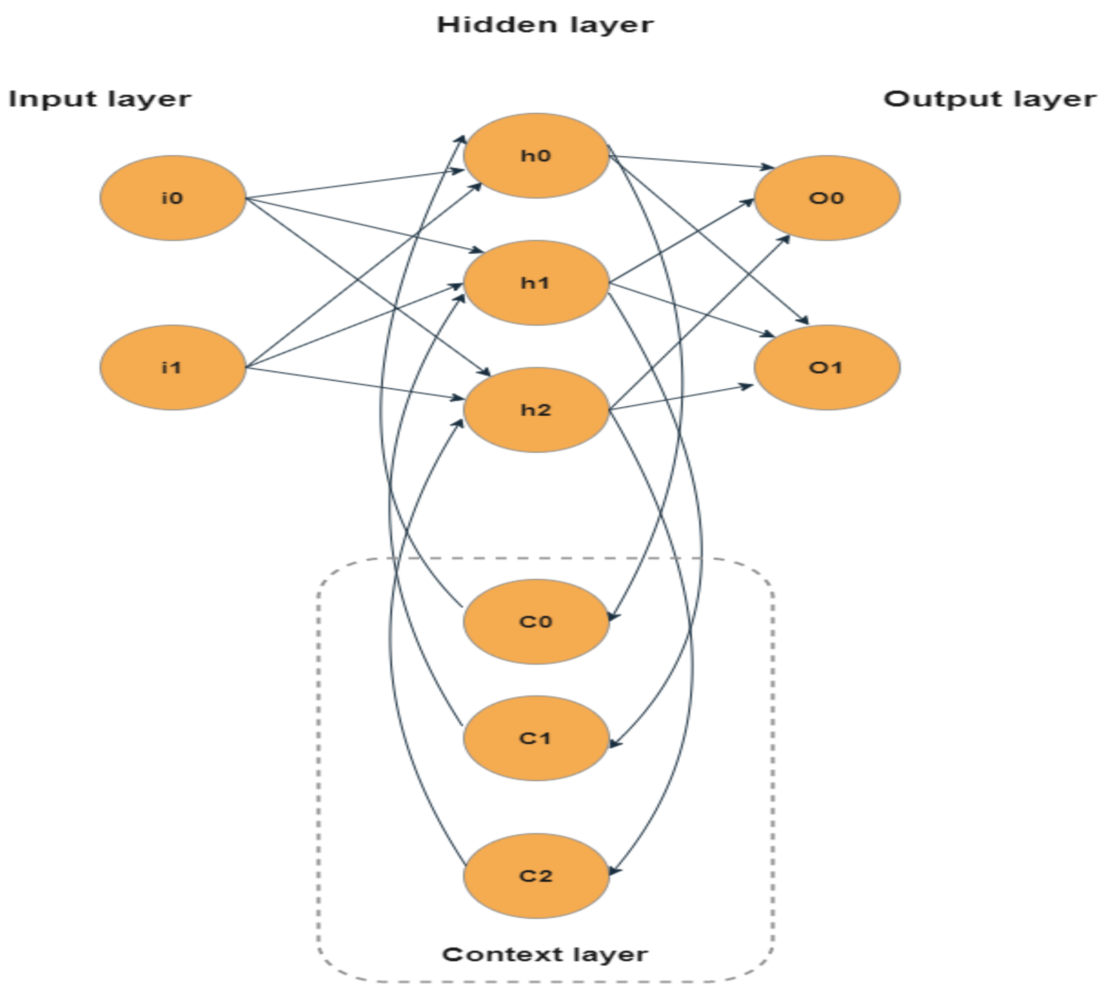

**Figure 1.7:** RNN Architecture [20]

Long Short-Term Memory (LSTM): Hochreiter and Schimdhuber invented the LSTM in 1997, but it has gained prominence as an RNN architecture for a variety of applications recently. Products that you use every day, like cell phones, conversational speech recognition…ext. The vanishing gradient problem is addressed by the LSTM, a particular kind of RNN that makes it possible to learn long-term dependencies, it has the concept of memory cells. The memory cell's ability to keep its value for a limited or unlimited period of time depending on its inputs enables it to recall essential information rather than simply it’s most recent calculated value. Three gates in the LSTM memory cell regulate the flow of data into and out of the cell. The input gate regulates when fresh data may enter the memory. The cell can recall new data by controlling when an old piece of information is lost. The output gate, in addition, regulates when the data stored in the cell is used in the output from the cell. Weights are also present in the cell and they regulate each gate. These weights are optimized using the training method, popularly known as BPTT (back propagation in time), depending on the resultant network output error. LSTMs can be effective in capturing complex temporal patterns in network traffic [20]. Figure **1.8** shows the LSTM architecture.

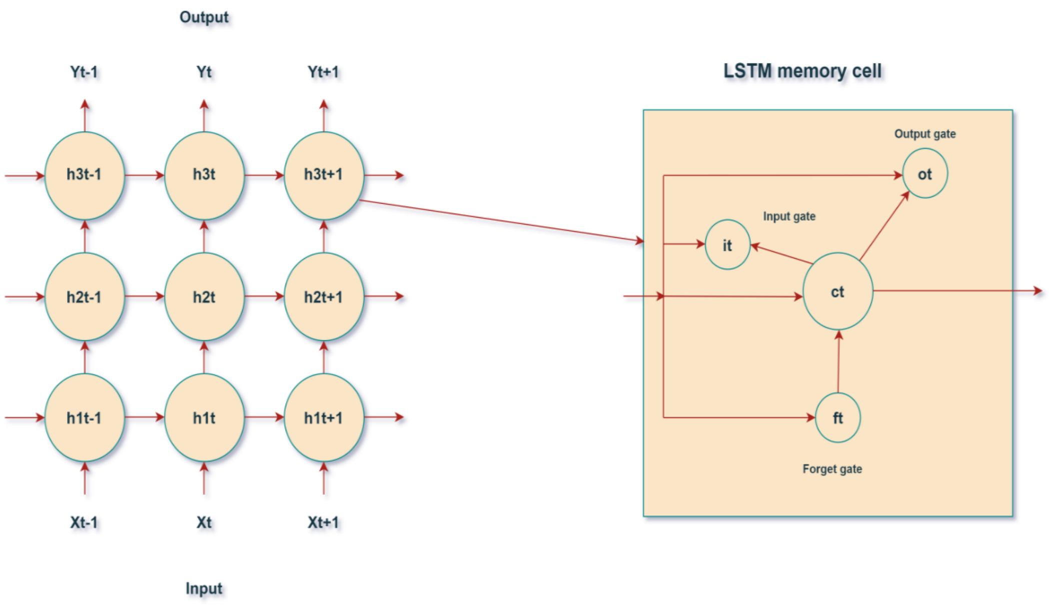

**Figure 1.8**: LSTM architecture [20]

Gated recurrent unit (GRU): The gated recurrent unit is a simplified version of the LSTM, was introduced in 2014. The output gate from the LSTM model is replaced with two gates in this model. A reset gate and an update gate are these. How much of the preceding cell's contents should be preserved is indicated by the update gate. The reset gate specifies how to combine incoming input with the contents of the preceding cell. Simply by changing the reset gate to 1 and the update gate to 0, a GRU may represent a typical RNN. The GRU is easier to understand, can be learned more rapidly, and has the potential to be more effective when used. However, the LSTM can be more expressive and produce better outcomes with more data. and it is commonly used in picture captioning, handwriting recognition, speech recognition, gesture recognition, natural language text compression …ext [20]. Figure **1.9** shows the GRU cell architecture.

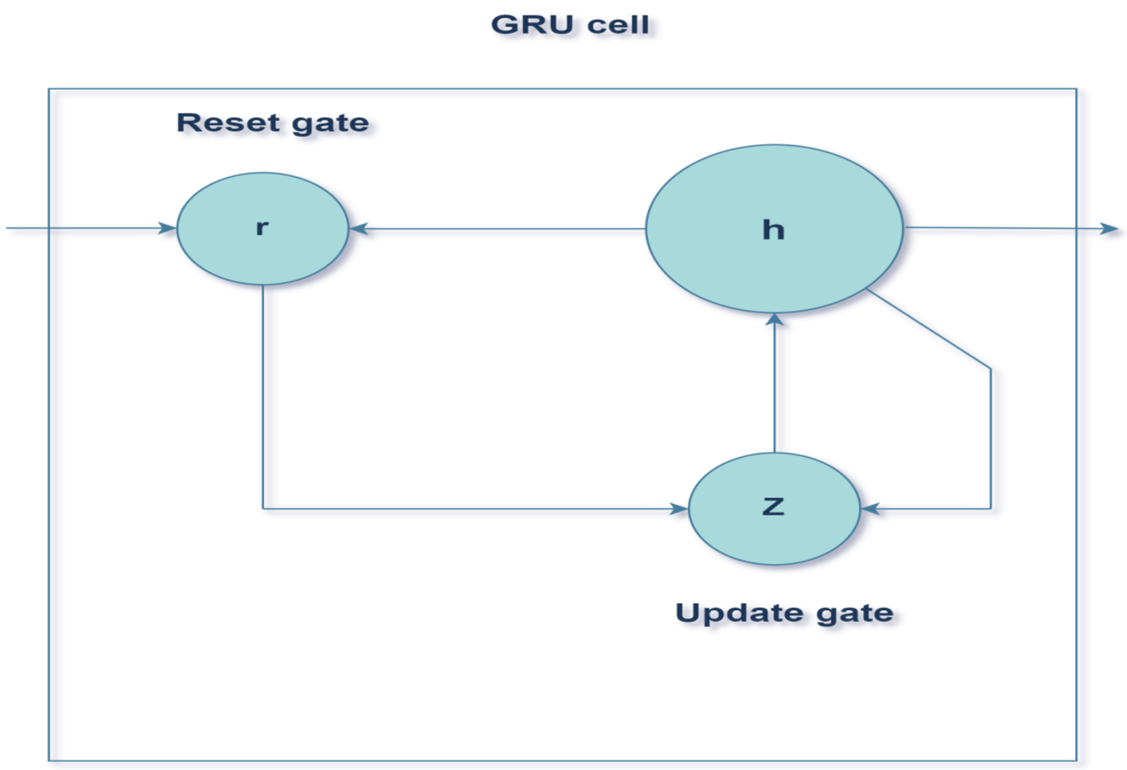

**Figure 1.9:** GRU cell [20]

Autoencoders: LeCun discovered the first known application of autoencoders in 1987. Three layers make up this particular ANN variant: input, hidden, and output layers. It is an unsupervised learning model called an autoencoder that tries to recreate the input data. They are made up of a decoder, which reconstructs the input from the representation, and an encoder, which translates the input to a lower-dimensional representation. Using the proper encoding function, the input layer is encoded into the hidden layer. The hidden layer has a significantly smaller number of nodes than the input layer does. The compressed version of the original input is present in this hidden layer. The output layer uses a decoder function to attempt to recreate the input layer. Autoencoders learn continuously using backward propagation. For this reason, autoencoders are classified as self-supervised algorithms. It can be used for anomaly detection by learning the normal patterns in network traffic and identifying deviations from the learned representation [20]. Figure **1.10** shows the Autoencoders architecture.

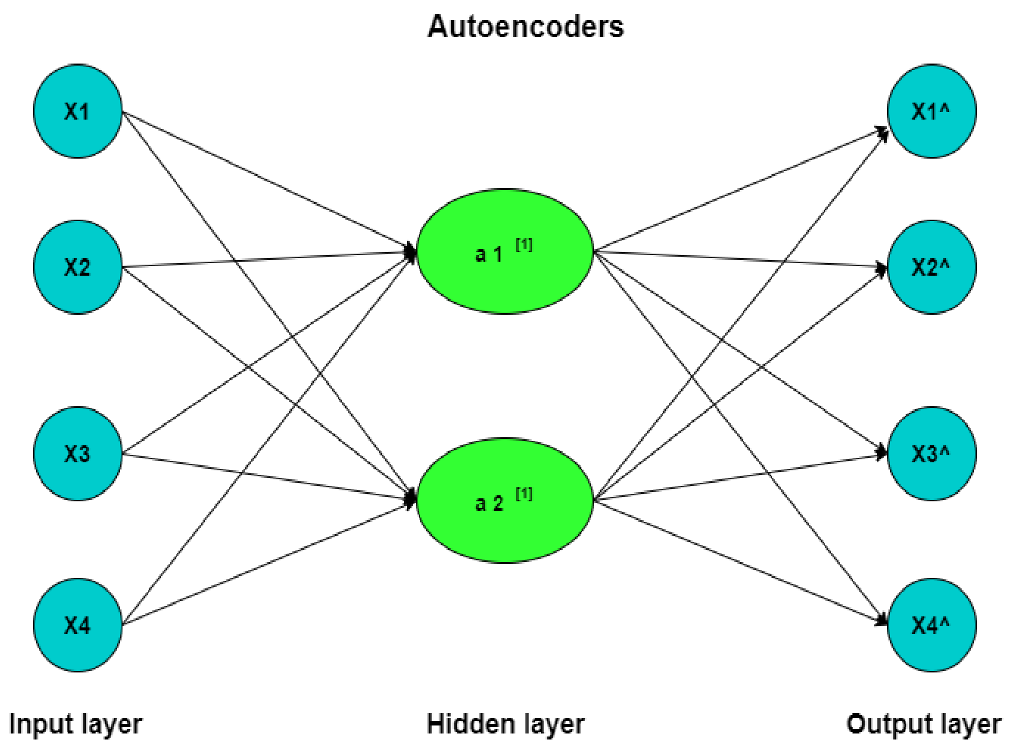

**Figure 1.10:** Autoencoders architecture [20]

Practically the architecture chosen relies on several elements, including the data's type, the intrusion detection system's specific objectives, and the resources at hand. In order to enhance the performance of intrusion detection, research in this area is ongoing, and new designs or modifications of current ones are always being studied such as hybrid architectures that leverage the strengths of many architectures and models to capture spatial and temporal features simultaneously, leading to improved intrusion detection performance.

## 1.8 Transfer learning

- **What is transfer learning**

Transfer learning means reusing a previously trained model on a different issue. Due to its ability to train deep neural networks with fairly little data, it is now highly popular in deep learning. Since the majority of real-world situations do not have millions of labeled data points to train such complicated models, this is now considered as helpful solutions in the data science sector. The fundamental idea behind transfer learning is to use what a model learned from one assignment to help you with another, for example, in training a classifier model to predict whether an image contains food or not, we could use the knowledge it gained during the training to recognize if the picture contains drinks. So, we can say transfer learning involves using a learning model that has already been trained to solve a new but related issue [21].

- **Principal of transfer learning works**

The main concept of transferred learning is to apply what a model has learnt from a task with a lot of labeled training data to a new task with little to no training data. We begin the learning process using patterns discovered while completing a comparable activity, as opposed to starting from scratch.

In neural network, edges are often detected by earlier layers, followed by certain task-specific properties in the intermediate layers. The early and intermediate layers are utilized in transfer learning, whereas the last layers are just retrained. It helps in utilizing the labeled data from the first job it was trained on. Figure **1.11** shows the transfer learning concept.

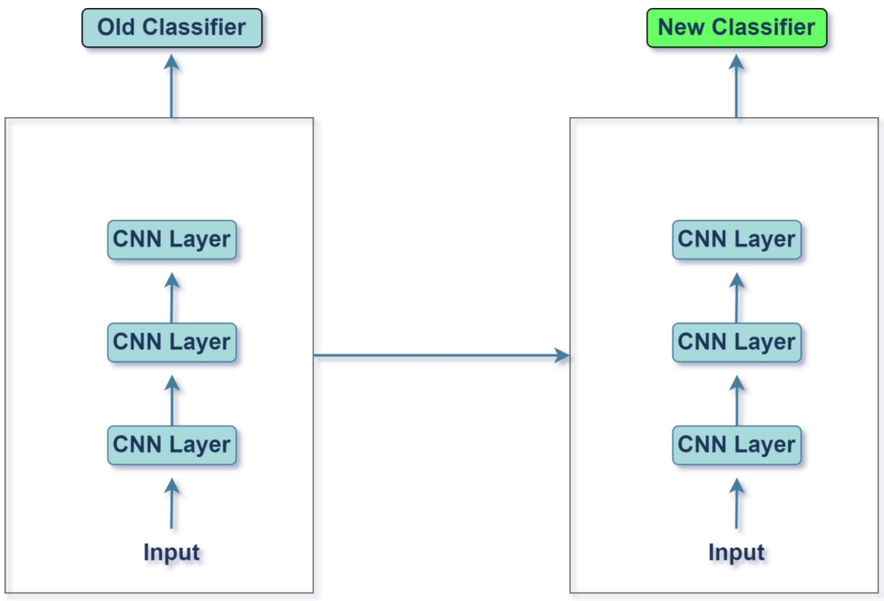

**Figure 1.11:** Transfer learning concept [22]

We can say it is the process of transferring as much information as feasible from the task the model was trained on to the current task. Depending on the issue and the information, this knowledge might take many different forms. For instance, the construction of models could facilitate the identification of new objects [23].

- **Why we use transfer learning**

For a neural network to be trained from the start, a lot of data is typically required. And sometimes that data is not always accessible. Transfer learning is useful in this situation. Because the model has previously been trained, transfer learning allows for the construction of reliable machine learning models with relatively minimal training data. This is particularly useful in natural language processing since producing sizable labeled data sets typically necessitates expert expertise, and it can occasionally take days or even weeks to train a deep neural network from scratch on a challenging job, training time is also shortened. Transfer learning is one of the most promising methods that might eventually lead to artificial general intelligence (AGI).

We can say we generally use Transferred learning for three raisons:

- There isn't enough labeled training data to start from scratch and train your network.
- There is already a network that has been pre-trained to do a comparable task and this network is often trained on enormous quantities of data.
- When the input for tasks 1 and 2 is the same.

There are a lot of libraries that provides a variety of pre-trained models for feature extraction, transfer learning, prediction, and fine-tuning such as Keras and Tensorflow [22].

## 1.9 Challenges and limitation of deep learning in intrusion detection

The capacity of deep learning-based intrusion detection systems (IDS) to automatically learn and detect complex patterns in network traffic has increased their appeal. However, they have a number of difficulties and restrictions, such as:

- The amounts of labeled training data: deep learning models need a lot of labeled training data. It is difficult to get a broad and representative dataset for intrusion detection since actual attacks are uncommon and frequently proprietary or sensitive, the need of labeled data in the training phase for a model (which may be expensive and time-consuming to gather) would have trouble adjusting to new or zero-day threats without a lot of human labeling and retraining work [24].

- Generalization problem: Deep learning models often have difficulty generalizing well to new or unanticipated attacks that differ greatly from the training set. They might miss new attacks or raise false alarms when innocuous network traffic exhibits traits that are similar to known attacks.

- Resource-intensive training and maintenance: Deep learning models frequently require a significant amount of computing time and resources for training, hyperparameters tuning, and model update. Maintaining up-to-date IDS is difficult since models must be constantly retrained or updated to react to changing attack strategies [25].

- Adversarial attacks: Deep learning models are susceptible to adversarial assaults, in which an attacker plays with the input data to deceive the model's judgment. The effectiveness of the IDS might be jeopardized by adversarial attacks that manage to avoid detection or result in false positives or negatives [26].

- Interpretability and explainability: Deep learning models are sometimes referred to as "black boxes," which makes it challenging to comprehend the logic behind their predictions. Since security analysts must know the decision-making process and recognize the elements assisting in the detection, interpretability and explainability are essential for intrusion detection systems [25].

## 1.10 Conclusion

In this chapter, we have presented a comprehensive overview of the cyber security field and emphasized the increasing significance of combating network intrusions, specifically distributed denial of service (DDoS) attacks. We have also highlighted the potential of deep learning algorithms, such as deep neural networks (DNNs), convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in enhancing intrusion detection capabilities.

The literature review chapter establishes the research context, identifies knowledge gaps, and guides subsequent research activities.

# chapter 2
# Methodology

## 2.1 Introduction

Our research problem statement focuses on the development of an intrusion detection system (IDS) for networks using deep learning techniques. Cyberattacks, especially Distributed Denial of Service (DDoS) attacks, pose significant threats to network security in the current digital environment. The goal of our research is to use the power of deep learning algorithms to successfully find and stop these attacks. To address this issue, comprehensive datasets are essential for training and evaluating intrusion detection systems. This chapter introduces our suggested methodology by describing the dataset and preparing it for deep learning model training.

## 2.2 Proposed methodology

The proposed methodology for constructing an effective intrusion detection system using deep learning techniques consists of four major steps: data comprehension, preprocessing, training, and testing, as shown in figure **2.1**. Each stage is essential for constructing a robust and accurate intrusion detection model.

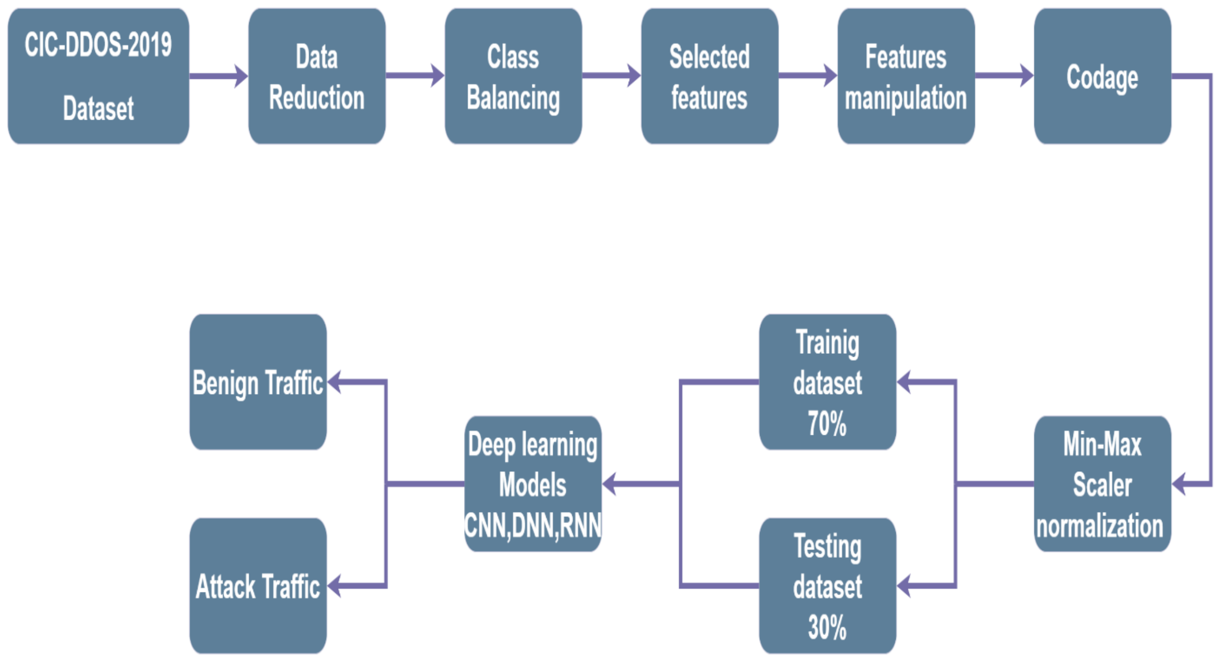

**Figure 2.1:** General architecture

## 2.3 Dataset

For the purpose of this study, we decided to use the latest dataset known as CIC-DDoS-2019 that was made available by the Canadian Cyber security Institute (CIC) for the detection of distributed denial of service attacks. CICDDoS2019 is the latest dataset, and compared to the other network traffic datasets, it has a large number of samples. It also includes both incoming and outgoing traffic from the most recent DoS and DDoS attacks, whereas other datasets only include a small number of DoS attack situations. Also, the CICDDoS2019 dataset has more than 80 features linked to network flow and eleven types of traffic from the latest DoS and DDoS attacks that were collected over a real-time.

The full data set contains 50,063,112 instances; of which 50,006,249 are DDoS attacks and that 56,863 are instances of benign network traffic (legitimate /normal). Table **2.1** and table **2.2** provide a breakdown of the instances for each DDoS attack type. Notably, the dataset includes both reflection- and exploitation-based DDoS attacks. Figure **2.2** shows the representation of the data set.

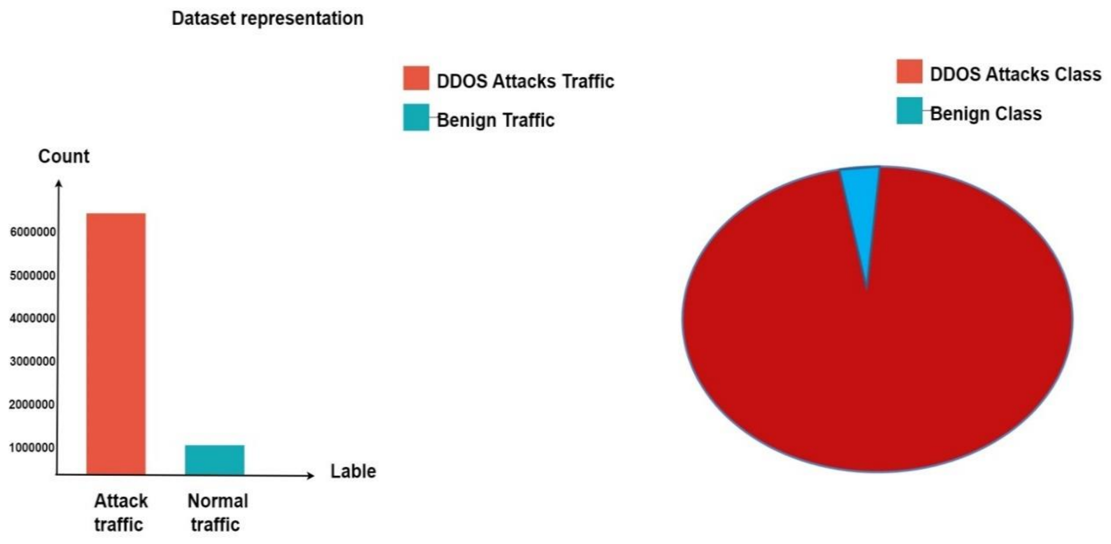

**Figure 2.2:** CIC-DDOS-2019 representation

### 2.3.1 DDOS attack based on reflection

A reflection attack is a DDoS attack where an attacker spoofs the target's IP address and sends requests to servers that operate UDP or TCP-based services. After receiving the request, the server will react to it by delivering a response to the IP address of the target.

This "reflection" of traffic amplifies the attack and overwhelms the target. Reflection attacks can be based on either TCP such as MMSQL and SSDP, or UDP like TFTP and NTP or both protocols TCP/UDP such as DNS, LDAP, NetBIOS, SNMP, and PortMap [27].

### 2.3.2 DDOS attack based on exploitation

On the other hand, exploitation-based DDoS attacks target weaknesses inside the protocols or within the infrastructure of the target. The objective of these attacks is to exhaust the target's resources and disrupt its services by exploiting flaws in how TCP or UDP connections are established and maintained. Attacks that are based on exploitation use TCP and UDP as their foundation. The TCP attacks consist of SYN flood attacks, which are effective because they tap into the handshake mechanism of a TCP connection. On the other hand, a UDP attack, often known as a UDP flood and UDP-Lag are attacks based on UDP protocol [28]. Figure **2.3** classify DDOS attacks on reflection and exploitation.

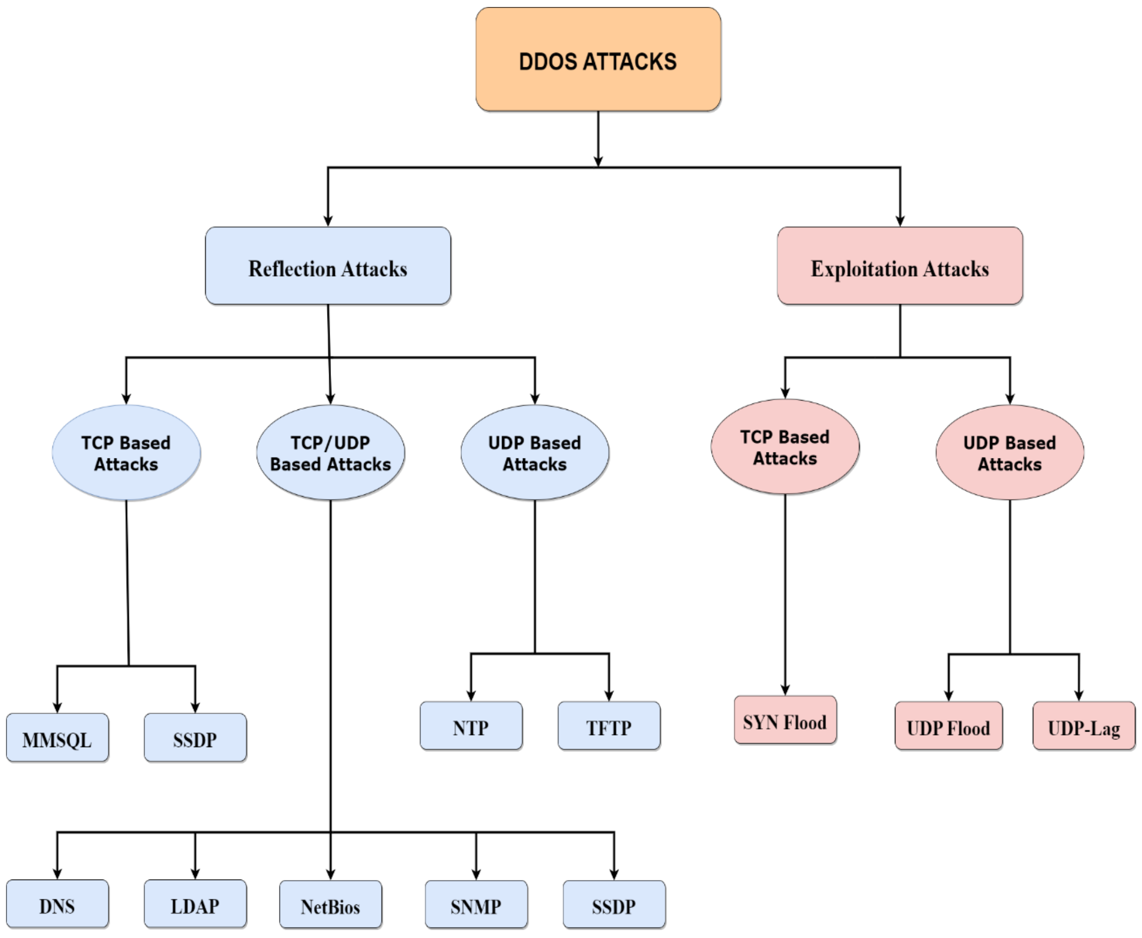

**Figure 2.3:** DDOS attacks based on reflection and exploitation [29]

### 2.3.3 CIC DDOS 2019 attacks

- SYN FLOOD: This attack uses the normal TCP three-way handshake (sending SYN, SYN-ACK, and ACK) to use resources on the targeted network [30]. Figure **2.4** represent SYN flood attack.

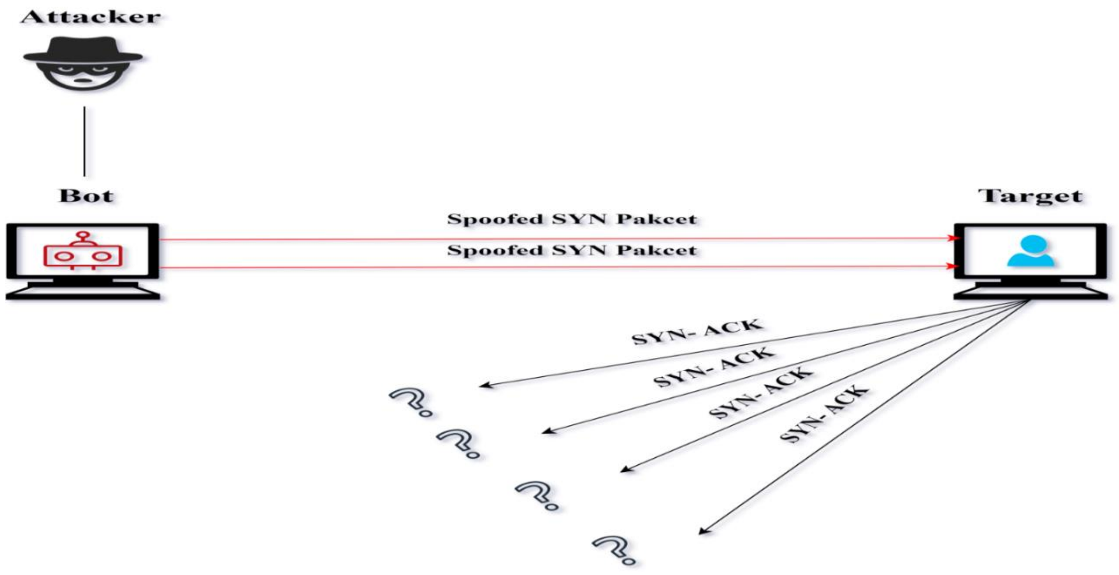

**Figure 2.4:** SYN flood attack **[**31**]**

- UDP (UDP Flood): The attacker sends a large number of UDP packets to the target server, which can overwhelm the server and cause it to crash [32]. Figure **2.5** represent UDP flood attack.

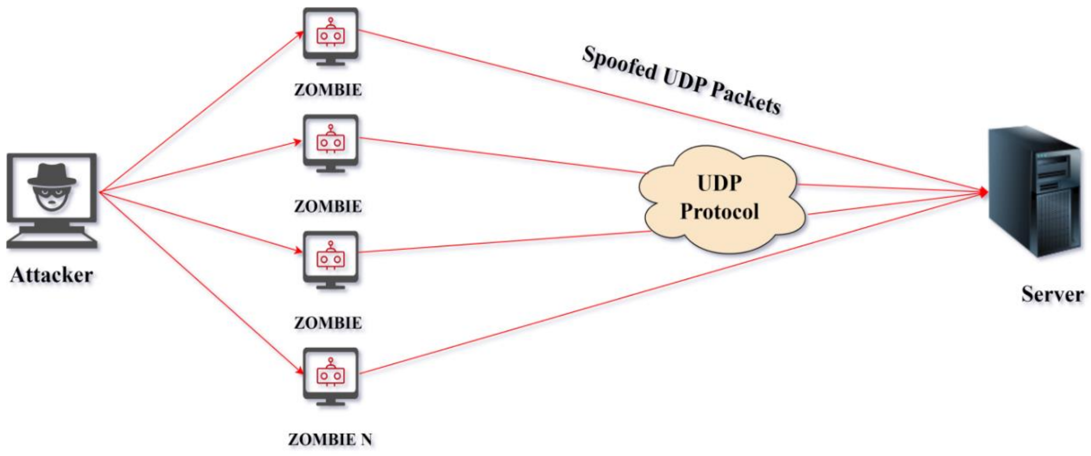

**Figure 2.5:** UDP Flood attack [33]

- UDP_Lag: This attack breaks client-server communication. Utilized for real-time applications that needs quick data delivery, such as online gaming and video conferencing. Sending a large number of UDP packets can cause real-time applications to lag or delay [32].
- MMSQL: Microsoft Structured Query Language is a form of attack that allows attackers to execute malicious SQL commands.
- SSDP: Simple service discovery protocol, The SSDP protocol is used by devices such as routers, and other internet-connected devices to discover and communicate with other devices on a network. this attack focus on sending massive traffic to a targeted victim, overloading the targeted network, and causes the web resource to become unavailable [28]. Figure **2.6** represent SSDP attack.

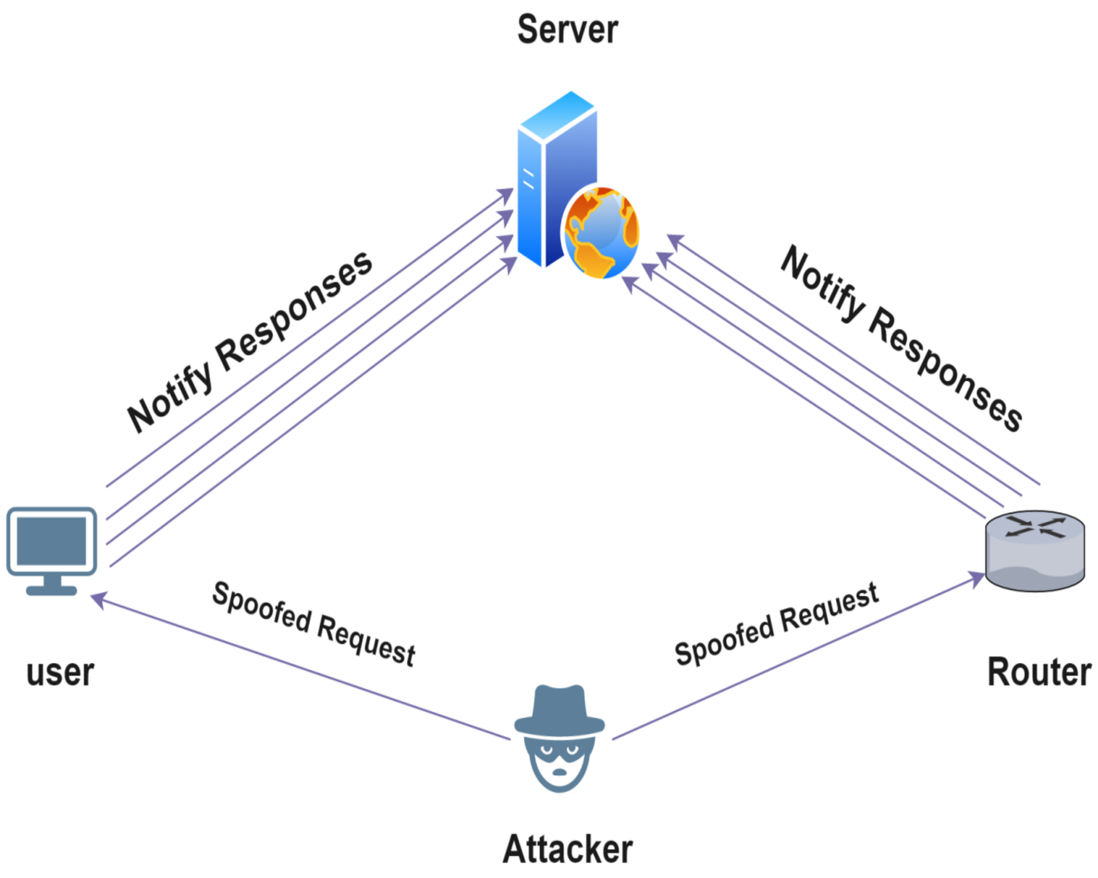

**Figure 2.6:** Simple service discovery protocol attack [28]

-

- NTP: Network Time Protocol (NTP) is a networking protocol used to synchronize computers' timers across a network. NTP is used to make sure that all of the devices on a network share the same time with a very accurate source of time. An attacker uses the (NTP) server to send a lot of UDP data traffic to a particular client-server network or to other networks. This attack might disable the destination and its network infrastructure [30]. Figure **2.7** represent NTP attack.

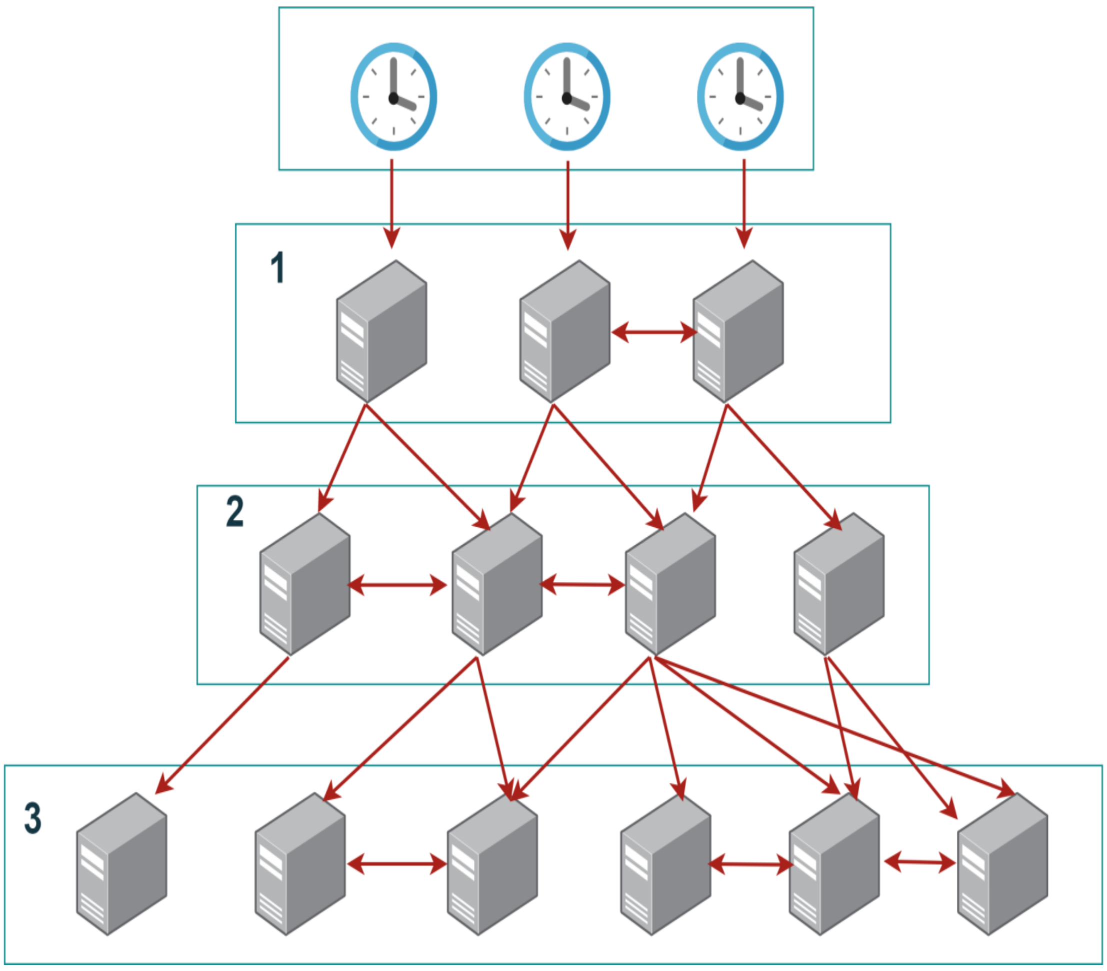

**Figure 2.7:** Network time protocol attack [34]

- TFTP: Trivial file transfer protocol, is simplify version of FTP. In particular, an attacker makes a request for a file by default, and the victim's TFTP server sends the data back to the host that made the request [30].

- DNS: The attacker sends a lot of DNS queries to open resolvers. This makes the resolvers send back big responses to the target victim, which overloads its network. Figure **2.8** represent DNS attack.

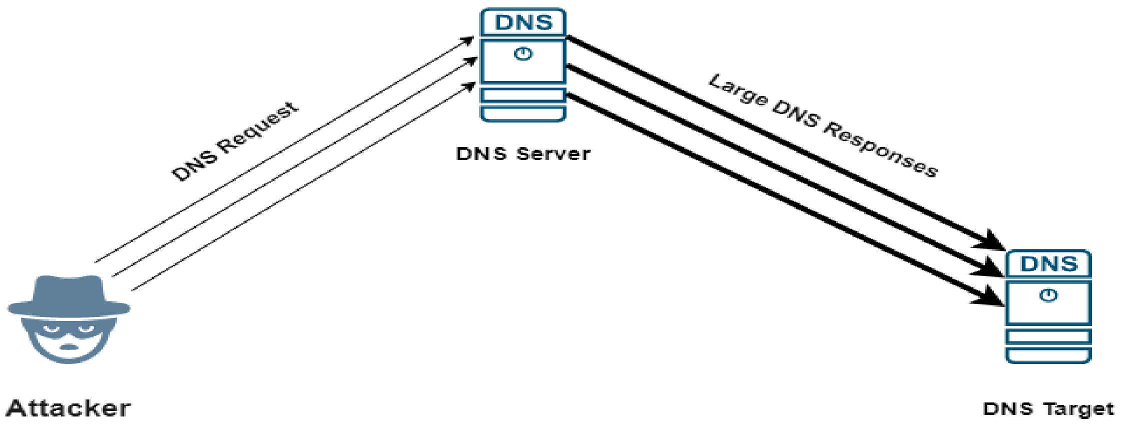

**Figure 2.8:** Domain name system attack [35]

- LDAP: Lightweight Directory Access Protocol attacks target organizations that utilize LDAP for user authentication and directory services. LDAP injections generate malformed queries to obtain access and potentially modify directory data [36].

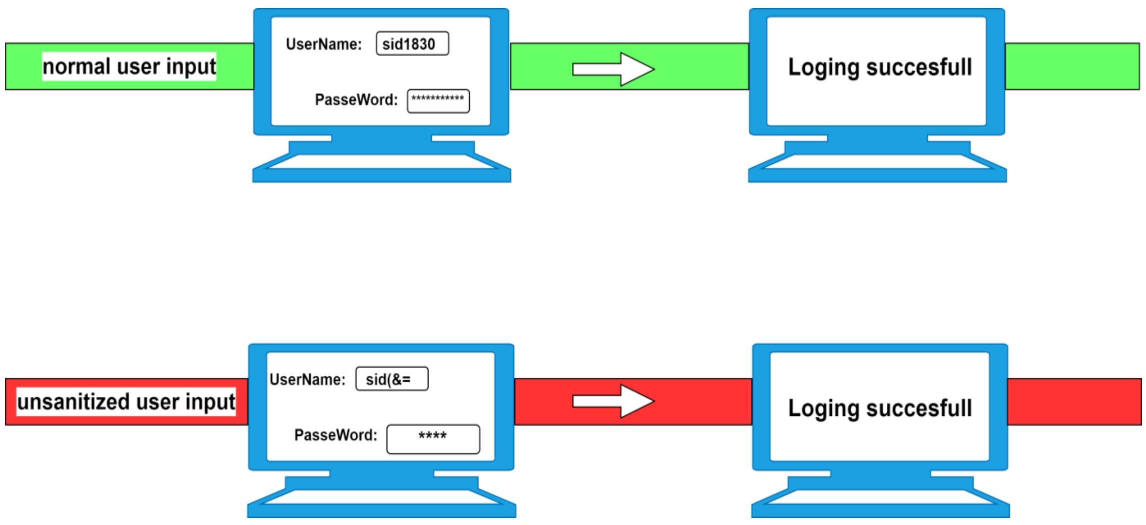

**Figure 2.9:** LDAP DDOS attack [36]

- NetBIOS: Network basic input output system, is an API that allows applications to communicate within a LAN, hackers exploit NetBIOS to infect and take control of large number and then use them of part of DDOS attack.

- SNMP: Simple network management protocol, is a protocol used for network management and monitoring, the attacker use it to sends a lot of SNMP queries with a fake IP address to a lot of connected devices, which then respond to the fake IP address [37]. Figure **2.10** represent SNMP DDOS attack.

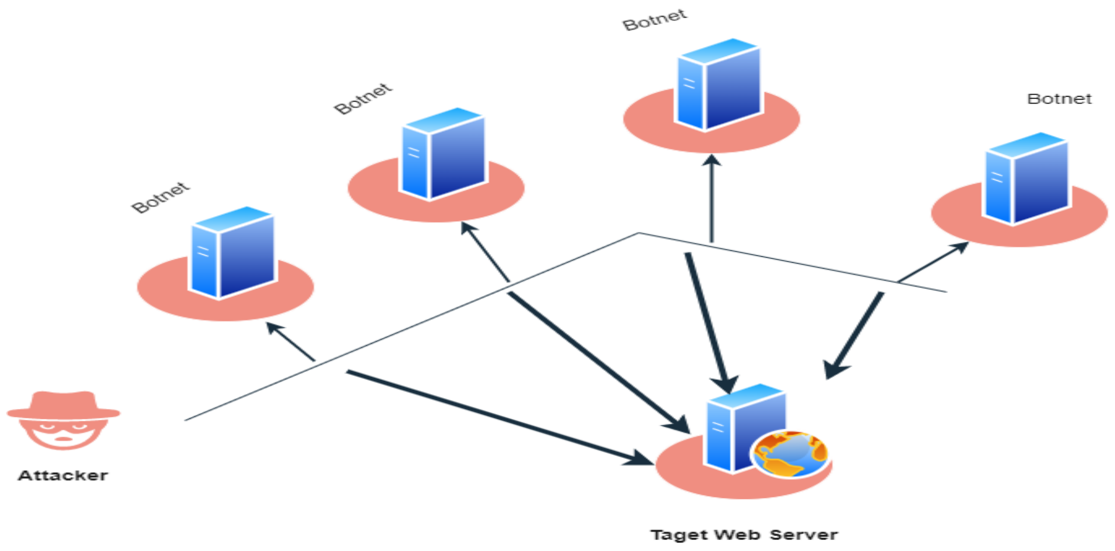

**Figure 2.10:** SNMP DDOS attack [37]

- WebDDOS: is a type of cyber-attack that floods a website or web service with an excessive quantity of traffic, slowing it down or making it impossible for users to access.
- PORTMAP: A Portmap DDoS attack exploits a vulnerability targets the Portmapper service of a system. Portmapper is a service that enables clients of remote procedure call (RPC) to find the port number associated with a specific RPC service [38]. Figure **2.10** represent Portmap DDOS attack.

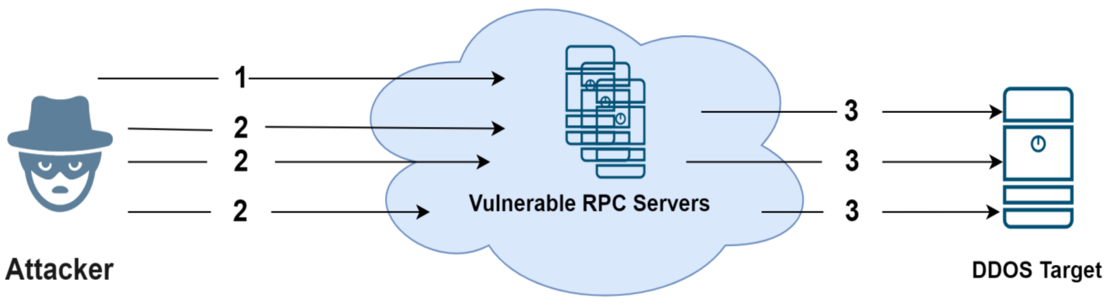

**Figure 2.11:** Portmap DDOS Attack [38]

It is important to note that the CIC-DDoS-2019 dataset has been further segmented into two separate subsets, one for training purposes and the other for testing purposes. The training collection is about 22 gigabytes (GB) and has instances from 12 different types of attacks. This dataset was used to build and improve the models for detecting intrusions. The efficiency of the trained models and their capacity for generalization is evaluated with the use of a test dataset that is approximately 8 gigabytes in size and contains instances from seven different types of attacks.

| Classe Label | Number of instances |
|---|---|
| BENIGN | 56,863 |
| DDoS_DNS | 5,071,011 |
| DDoS_LDAP | 2,179,930 |
| DDOS_MMSQL | 4,522,492 |
| DDOS_NetBios | 4,093,279 |
| DDOS_NTP | 1,202,642 |
| DDOS_SNMP | 5,159,870 |
| DDOS_SSDP | 2,610,611 |
| DDOS_SYN | 1,582,289 |
| DDOS_TFTP | 20,082,580 |
| DDOS_UDP | 3,134,645 |
| DDOS_UDP-Lag | 366,461 |
| DDOS_WebDDOS | 439 |

**Tableau 2-1:** Number of records for each category of DDOS attacks in the train set

| Classe Label | Number of instances |
|---|---|
| DDOS_Portmap | 191694 |
| DDOS_NetBios | 3455899 |
| DDOS_MMSQL | 5775786 |
| DDOS_LDAP | 2113234 |
| DDOS_UDP | 3782206 |
| DDOS_UDP-Lag | 725165 |
| DDOS_SYN | 4320541 |

**Tableau 2-2:** Number of records for each category of DDOS attacks in the test set

## 2.4 Preprocessing

The CIC-DDoS-2019 dataset needs to be preprocessed before it can be used to train deep learning models. Several methods were used to make sure the data was correct and to improve the intrusion detection system's quality and performance. Three key steps were performed: data reduction, data cleaning, and train-test split.

### 2.4.1 Data reduction

Data reduction was necessary due to the dataset's large volume. To manage this, both the training and testing datasets were reduced using the same procedure. Regardless of having a computer with 12 GB of RAM, it was difficult to manage such an enormous quantity of information we analyzed and processed each CSV file individually. Random selection of a predefined number of instances was performed for the majority class, resulting in a smaller and well-balanced dataset. All instances of the minority class were retained. The reduced and balanced datasets were concatenated into "Train00.csv" and "Test00.csv" files.

### 2.4.2 Data cleaning

In the data cleaning step, various actions were taken to ensure dataset quality. To prevent having to store the same data twice, we deleted duplicate columns by keeping the first original feature and removing the duplicate. Instances with missing or invalid values (e.g., +inf, -inf, NaN) were eliminated. The **flow packet** column was converted from string to a numerical data type. Empty cells were filled with zeros to maintain data integrity. The **label** column was encoded for binary classification, with normal traffic assigned a value of zero and all attack traffic assigned a value of one. In order to improve the performance of the intrusion detection system, several irrelevant columns were eliminated from the dataset. These columns include **source IP, destination IP, flow ID, similar HTTP, unamed0,** and **timestamp**, as their impact on intrusion detection was determined to be minimal. After the data cleansing, we left with 83 features which were unique and important.

### 2.4.3 Train and test split

Finally, our dataset has been normalized to facilitate data processing for our model, and we have used the train test split function to divide the training data and test the model using a sample of the training data.

## 2.5 IDS implementation

Deep learning typically employs multiple information-processing layers within a hierarchical structure. Deep learning refers to the investigation and publication of multiple deep neural network (DNN) architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). This section focuses on the implementation of convolutional neural networks (CNNs), and recurrent neural networks (RNNs) for intrusion detection.

### 2.5.1 DNN based IDS

A Deep Neural Network (DNN) is an Artificial Neural Network (ANN) with more layers in between the input and output levels. Figure **2.12** illustrate our DNN model architecture. The DNN is made up of five parts: neurons, weights, connections, biases, and functions. The inputs (X= $x_1$, $x_2, \ldots . x_N$)are linked with the weights (W= $w_1$, $w_2, \ldots . w_N$). The full-fledged deep learning system is based on multilayer perception and uses different activation functions that give real values instead of Boolean values like in classic perception. To assist in changing input weights and minimizing "loss", mathematically, this can be written as follows [30]:

$$\mathbf{Y} = \delta\left(\sum_{n=1}^{n} \boldsymbol{Wixi} + \boldsymbol{b}\right) = = \delta(\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{X} + \boldsymbol{b}) \qquad (2\text{-}1)$$

**W:** weights

**X:** inputs

**b:** biases

**δ:** Activation functions

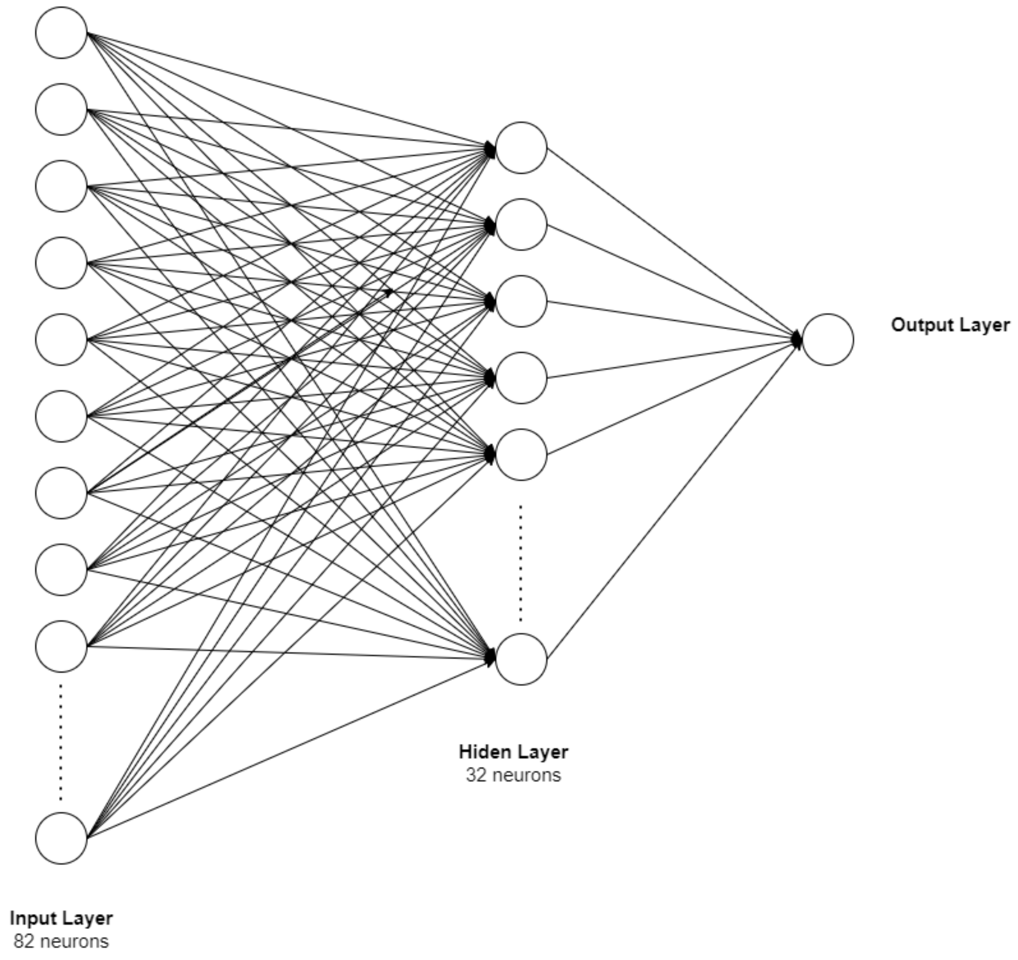

**Figure 2.12:** Deep neural network model

### 2.5.2 CNN based IDS

A Convolutional neural network (CNN) is a neural network with one or more convolutional layers that are primarily used for image recognition, classification, and processing tasks, also have been utilized for understanding in Natural Language Processing (NLP) and speech recognition [39]. In a typical CNN-based intrusion detection system, network traffic data is preprocessed and transformed into a suitable format before being fed into the CNN model. A CNN typically consists of three layers: convolutional, pooling, and fully connected.

In this work the input layer receives network traffic data in the form of training data shape (x_train.shape [1], 1). The convolutional layers are accountable for applying a collection of learnable filters to the input data, thereby capturing local patterns and dependencies.

The first convolutional layer has 82 filters and a kernel size of 3, while the second has 32 filters and the same kernel size. Both convolutional layers are activated using the ReLU function.

The pooling layers, which are implemented with MaxPooling1D, down sample the output of the convolutional layers, reducing the dimensionality while preserving the most pertinent information, after each pooling layer, dropout layers are added to avoid over fitting and make the model more flexible. Following the convolutional and pooling layers, the output is transformed into a one-dimensional vector before being transmitted to the fully connected layers. The dense layers, or as they are known, the fully connected layers, are responsible for classification based on the learned features. Following two dense layers and the ReLU activation function is a dropout layer. The final dense layer is a single unit with a Softmax activation function that generates a probability score between 0 and 1 for binary classification. The model is trained with labeled data and optimized with Adam. It employs ReLU and Softmax activation functions to introduce non-linearity and generate binary classification predictions. Figure **2.13** illustrate our CNN model architecture.

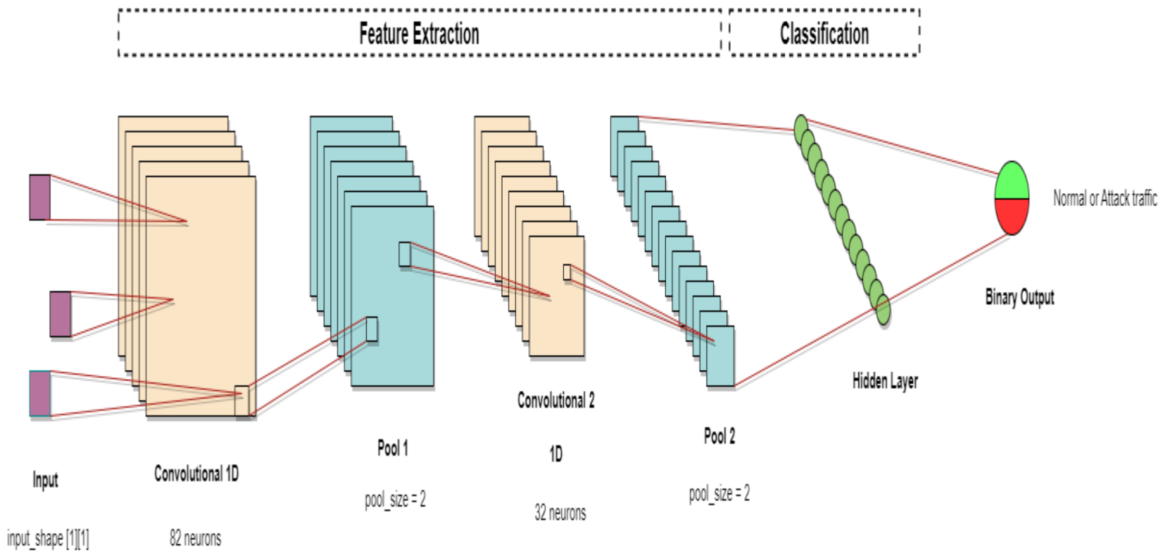

**Figure 2.13:** Convolutional neural network model

### 2.5.3 RNN based IDS

An RNN model (Recurrent Neural Network) is a form of neural network that is utilized in machine learning and natural language processing activities. RNN’s have architectural that works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer. This makes them particularly well-suited for tasks that involve sequences of data [40].

The RNN architecture is chosen for its ability to capture temporal dependencies in sequential data, making it suitable for analyzing network traffic patterns. In our work, the model utilizes the LSTM (Long Short-Term Memory) layer, known for its effectiveness in handling long-term dependencies. Our model consists of multiple layers, including the LSTM layer, dropout layers, and dense layers. The input shape of the LSTM layer is determined by the shape of the training data, which is indicating the number of time steps and features. We employ the ReLU activation function to introduce non-linearity, enhancing the model's capacity to capture complex patterns. Figure **2.14** illustrate our RNN model architecture.

To prevent over fitting, dropout layers are added after the LSTM layer and the dense layer. These layers randomly drop out a fraction of the units during training, reducing the model's reliance on specific patterns and improving generalization. By doing so, the model becomes more robust and performs better on unseen network traffic samples. The dense layers, or fully connected layers, are responsible for classification based on the learned features. The final dense layer consists of a single unit with a sigmoid activation function, producing a probability score between 0 and 1 for binary classification. The model is trained with labeled data and optimized with Adam optimizer. It employs ReLU and Sigmoid activation functions to introduce non-linearity and produce predictions for binary classification.

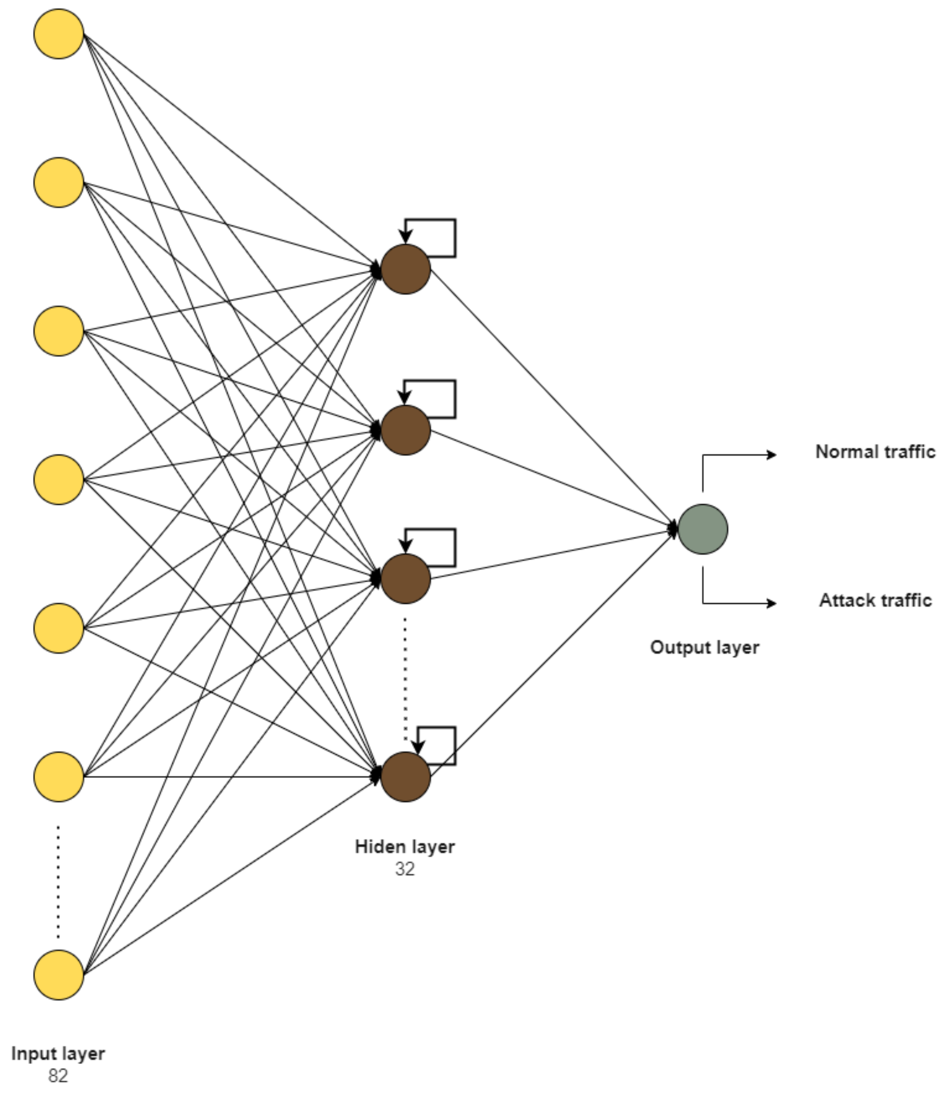

**Figure 2.14:** Recurrent neural network model

## 2.6 Activation functions in intrusion detection models

In neural network models, such as DNN, CNN, and RNN-based intrusion detection systems (IDS), activation functions play a crucial role. In our models we used Rectified Linear Unit (ReLU) and Softmax as an activation function.

### 2.6.1 ReLU activation function

The ReLU activation function is a widely used non-linear function that introduces non-linearity into the neural network model. It is represented by

$$\mathbf{f(x) = max\ (0, x)} \qquad \textbf{(2-2)}$$

where x is the input to the activation function. The ReLU activation function is a threshold-based function that returns the input value if it is positive and zero otherwise. By setting negative values to zero, ReLU effectively introduce non-linearity, allowing neural networks to model complex relationships between features. ReLU has gained popularity due to its simplicity, computational efficiency, and its ability to address the vanishing gradient problem [41].

### 2.6.2 Softmax activation function

In the output layer of classification models, including intrusion detection systems, the Softmax activation function is frequently applied. It is used to generate probabilities for various classifications, such as regular traffic and attack categories. It creates a probability distribution from a vector of real numbers, allowing the model to assign class probabilities. By dividing the total value of all the exponentials, the Softmax function determines the exponential of each input element and normalizes it. This normalization makes sure that the output numbers, which indicate probabilities, range from 0 to 1 and add up to 1[42].

Mathematical equation:

$$p_i = e^{x_i} / (\Sigma x \, e^{x_j}) \qquad \textbf{(2-3)}$$

**Pi:** Represent the probability

**e:** Is the base of the natural logarithm (approximately 2.71828)

**xi:** Inputs

## 2.7 Fine-tuning pre-trained models for intrusion detection using transfer learning

### 2.7.1 Transfer learning-based IDS

Transfer learning is a powerful technique in intrusion detection that improves the performance and efficiency of intrusion detection systems. It involves using pre-trained models trained on large-scale datasets for general tasks and adapting them for specific intrusion detection tasks. This approach leverages existing knowledge and representations to detect network intrusions more effectively. Pre-trained models capture relevant features for intrusion detection, such as spatial and temporal patterns. By fine-tuning these models, we enhance their ability to identify specific types of network intrusions without starting from scratch [43].

### 2.7.2 Fine-tuning based IDS

Fine-tuning utilizes transfer learning by adjusting trained models for similar tasks. During fine-tuning, pre-trained models are customized for improved intrusion detection. Relevant layers are modified to capture features and patterns in network traffic data. For example, in CNN models, filters adjust for spatial patterns, while in RNN models, LSTM units or layers tuned to capture temporal dependencies. To retain learned representations, early layers for general features are frozen. Later layers, including classification, are updated with new weights for intrusion detection adaptation. This tailoring to network traffic characteristics enhances effectiveness in identifying intrusions [44].

In our case, we performed random tuning of the neuron or filter values in our models, except for the first layers, which were determined based on the input shape of our dataset. For the hidden layers, we initially started with a small number of neurons to capture the basic patterns. This random tuning process is a common practice when exploring different architectures. Additionally, hyperparameters such as learning rate and batch size were adjusted to optimize the fine-tuning process. Once we achieved satisfactory accuracy, we stopped modifying these values, indicating that we found a configuration that produced good performance in intrusion detection. Figure **2.15** is the Conceptual diagram of our proposed implementation for deep learning methods.

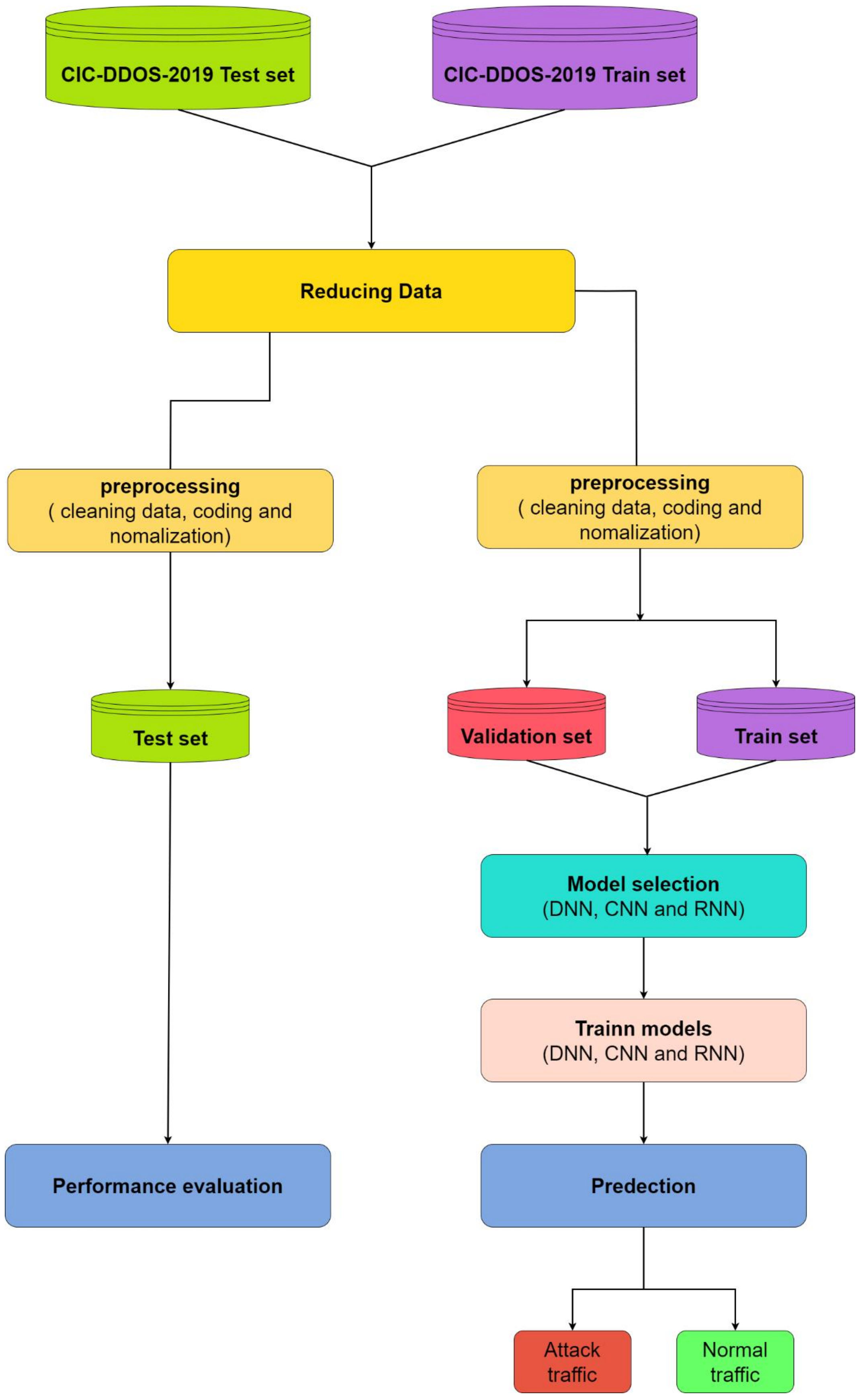
CIC-DDOS-2019 Test set
CIC-DDOS-2019 Train set
Reducing Data
preprocessing ( cleaning data, coding and nomalization)
preprocessing ( cleaning data, coding and nomalization)
Test set
Validation set
Train set
Model selection (DNN, CNN and RNN)
Trainn models (DNN, CNN and RNN)
Performance evaluation
Predection
Attack traffic
Normal traffic

**Figure 2.15 :** Conceptual diagram of our proposed implementation for deep learning methods

## 2.8 Evaluation metrics:

Evaluation metrics play a crucial role in assessing the performance and effectiveness of intrusion detection systems. These metrics provide quantitative measures to evaluate the system's ability to accurately classify network traffic and detect potential intrusions. In this section, we will explore several key evaluation metrics commonly used in intrusion detection and provide a brief explanation of their mathematical functions.

- True positive (TP)

True positives are instances correctly classified as positive (intrusions) by the intrusion detection system. These are actual positive instances predicted as positive.

- True negative (TN)

True negatives are instances correctly classified as negative (non-intrusions) by the intrusion detection system. These are actual negative instances predicted as negative.

- False positive (FP)

False positives are instances incorrectly classified as positive (intrusions) by the intrusion detection system. These are actual negative instances predicted as positive.

- False negative (FN)

False negatives (FN) are instances incorrectly classified as negative (non-intrusions) by the intrusion detection system. These are actual positive instances predicted as negative.

Accuracy measures the overall correctness of the intrusion detection system's predictions. It is calculated by dividing the number of correctly classified instances (true positives and true negatives) by the total number of instances. The mathematical formula for accuracy is:

$$\text{AC} = \frac{(\mathbf{TP} + \mathbf{TN})}{(\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN})} \qquad \textbf{(2-4)}$$

Precision measures the proportion of correctly identified positive instances (true positives) out of all instances predicted as positive. It focuses on minimizing false positive predictions. The mathematical formula for precision is:

$$\text{PR} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FP})} \quad \textbf{(2-5)}$$

Recall, also known as sensitivity or true positive rate (TPR), measures the proportion of actual positive instances that are correctly identified by the intrusion detection system. It focuses on minimizing false negative predictions. The mathematical formula for recall is:

$$\text{RC} = \frac{\mathbf{TP}}{(\mathbf{TP} + \mathbf{FN})} \quad \textbf{(2-6)}$$

The F1 score is a combined metric that balances precision and recall, providing a single value that represents the system's performance. It is the harmonic mean of precision and recall, with equal importance given to both metrics. The mathematical formula for the F1 score is:

$$\text{F1} = \frac{\mathbf{2}\times(\mathbf{PR}\times\mathbf{RC})}{(\mathbf{PR} + \mathbf{RC})} \quad \textbf{(2-7)}$$

The false positive rate measures the proportion of instances predicted as positive (intrusions) incorrectly, out of all instances that are actually negative (non-intrusions). It is calculated as:

$$\text{FPR} = \frac{\mathbf{FP}}{(\mathbf{FP} + \mathbf{TN})} \quad \textbf{(2-8)}$$

The confusion matrix is a tabular representation of the performance of an intrusion detection system. It presents the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It provides a comprehensive view of the system's classification results and is often used to derive other evaluation metrics.

## 2.9 Conclusion

Throughout this enlightening chapter, we have unveiled our groundbreaking methodology, which focuses on harnessing the immense potential of deep learning algorithms to create a robust intrusion detection system. Specifically designed to counteract and thwart distributed denial of service (DDOS) attacks, our approach capitalizes on the intrinsic capabilities of these algorithms, enabling swift and accurate identification and mitigation of such malicious activities, ultimately fortifying network security.

# chapter 3
# Results and analyze

## 3.1 Introduction

In this chapter we are going to discuss the results and analyze our research on developing an intrusion detection system using deep learning techniques, we will go over the experimental design, the environment of the work, the effect and the impact of hyperparameters, evaluation of transfer learning, and analyze the vulnerability of deep learning-based intrusion detection models to adversarial attacks.

## 3.2 Experimental setup

As we all know deep learning is a computationally expensive field that necessitates gear capable of effectively handling complicated calculations deep learning projects require the minimum fundamental software and hardware needs to make an experiment. We used a sturdy hardware arrangement that could handle the computing needs of our studies to meet these resource availability criteria, we have used a computer that contain an Intel(R) HD Graphics 4600 as a GPU, an Intel(R) Core (TM) i7-4790CPU @3.60GHz and a 16.0 GO for the volatile memory, as hardware and for the software we have started with setting up a local Python development environment using the Anaconda platform

## 3.3 Anaconda

Anaconda is an open-source distribution of the Python programming language, it was built especially for activities involving data science machine learning and deep learning projects, It offers a thorough selection of pre-installed libraries, tools, and frameworks, which makes it simpler to manage and set up the required software environment for data analysis and machine learning applications. Popular libraries like NumPy, pandas, and Scikit-learn are all included in Anaconda, as well as Jupyter Notebook, which enables interactive and exploratory data analysis [45].

## 3.4 Jupyter notebook

Jupyter notebook is an open-source web tool enables users to create and share documents with live code, graphics, and narrative prose. Users may write and execute code, see data, and record their research in a single interface thanks to the interactive environment it offers. Jupyter Notebook is a flexible tool for data exploration, experimentation, and collaboration since it supports a number of computer languages, including Python, R, and Julia [46].

## 3.5 Libraries and Frameworks

### 3.5.1 Pandas

It is a powerful Python data analysis and manipulation module. It offers data structures and operations for successfully handling and working with structured data, such as data in a table or a time series. In data science projects, Pandas is frequently used for data cleaning, processing, and exploration operations [47].

### 3.5.2 NumPy

It is an essential Python package for scientific computing. It offers effective multi-dimensional array objects as well as a huge selection of mathematical operations that may be applied to these arrays. In the Python data science environment, NumPy serves as a core library that many other libraries and frameworks rely on [48].

### 3.5.3 Keras

It is a Python-based open-source deep learning library. Users may design complicated models with less code since it offers a high-level interface for deep neural network construction and training. Keras is renowned for its flexible architecture, user-friendly features, and interoperability with other deep learning frameworks, including TensorFlow [49].

### 3.5.4 TensorFlow

It is a well-known deep learning framework that is open-source and was created by Google. It provides a wide range of materials, frameworks, and tools for creating and implementing machine learning models. TensorFlow is a flexible computational graph abstraction that makes it possible to execute sophisticated neural networks effectively on a variety of hardware platforms [50].

### 3.5.5 Scikit-learn

It is a flexible Python machine learning library. It offers a broad variety of tools and methods for jobs including dimensionality reduction, clustering, regression, and classification. In machine learning projects, Scikit-learn is frequently used for data preparation, model assessment, and model selection [51].

### 3.5.6 Matplotlib

It is a Python plotting library. It makes it possible to create visualizations that are static, animated, and interactive. It can produce a variety of graphs, charts, and plots to efficiently examine and present data since it has a large range of plotting tools and customization possibilities [52].

## 3.6 Dataset

In this work, we utilized three deep learning models for training and testing on the CIC DDos 2019 dataset. This dataset is widely regarded as one of the most up-to-date resources for studying DDoS attacks. To address the challenge of dealing with a large volume of data in each file of the CIC DDos 2019 dataset, we focused on creating multiple samples and testing them based on our research objectives. Additionally, we recognized the necessity of using a substantial amount of data to effectively apply deep learning models.

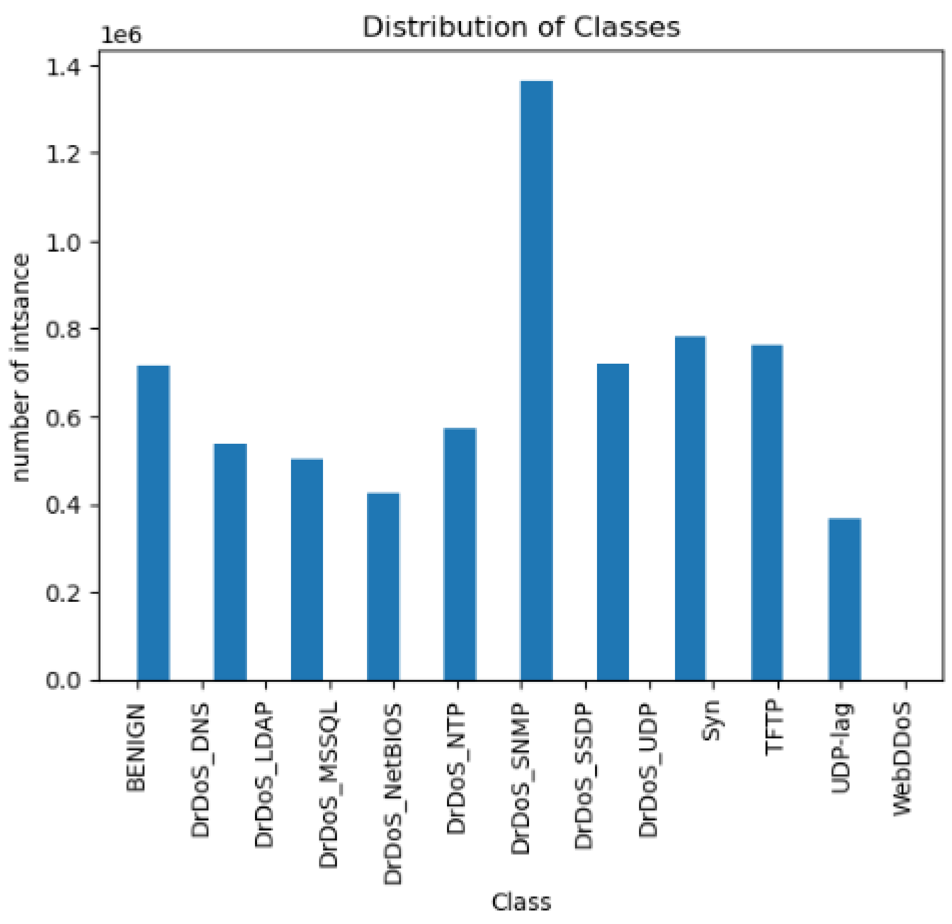

**Figure 3.1:** Class distribution using the under sampling

After applying the under-sampling method that gave us a random size from the majority class from a data file and keeping the minority class as it is, we have concatenated all the samples of each file into one file. We coded the classes of the output class with the name “label” to turn the output classes into two classes either, benign traffic or abnormal traffic. Figure **3.1** shows the class distribution using the under sampling method.

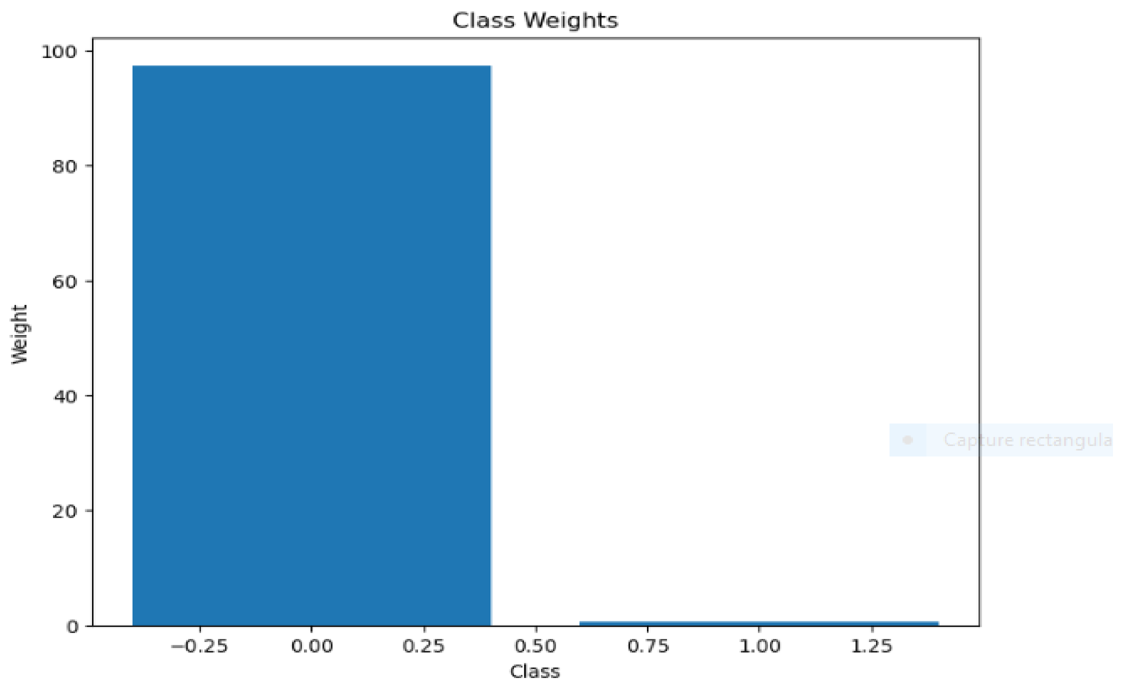

**Figure 3.2:** Classes weight after coding into binary

Finally, to solve the problem of unbalanced dataset that can lead to problems in the training phase of our models, we have used the smote technique to keep the train dataset balanced. We ended up working with this dataset sample, the tables below **3-1** and **3-2** show the size of the dataset using the under-sampling technique and after coding the label class and using the smote technique on it:

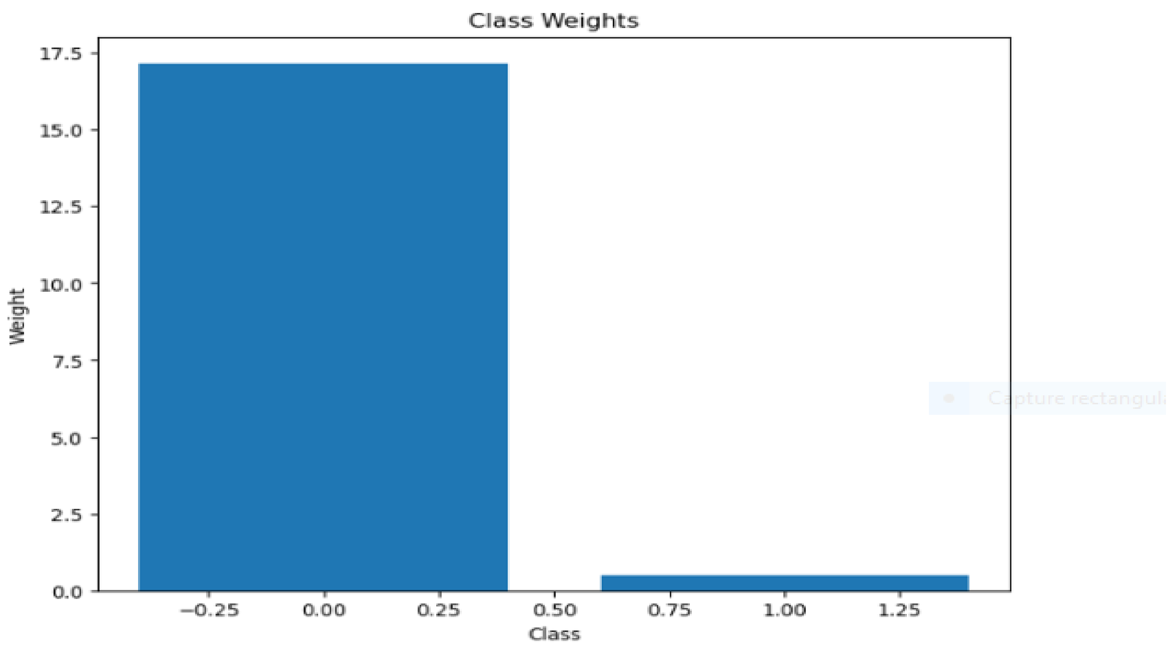

**Figure 3.3:** Classes weight after using smote

| Name | Number of instances |
|---|---|
| Syn | 784000 |
| DrDoS_SSDP | 763000 |
| TFTP | 761000 |
| DrDoS_UDP | 719000 |
| DrDoS_DNS | 680400 |
| DrDoS_SNMP | 602800 |
| DrDoS_NTP | 574600 |
| DrDoS_LDAP | 537333 |
| DrDoS_MSSQL | 501500 |
| DrDoS_NetBIOS | 426750 |
| UDP-lag | 366461 |
| WebDDoS | 439 |
| BENIGN | 34661 |

**Tableau 3-1:** Number of instances from each attack in train dataset

| Name | Number of instances |
|---|---|
| 1 | 6473100 |
| 0 | 194193 |

**Tableau 3-2:** Number of instances after concatenating coding and over sampling

## 3.7 Features selections

The CIC DDoS 2019 dataset contain 89 features "columns" we have selected 83 of them the most important ones and we have deleted 6 features that have no real impact on the intrusion detection precession. Table **3-3** illustrate the selected features.

| Features | Type |
|---|---|
| Unnamed | int64 |
| Source port | int64 |
| Destination port | int64 |
| Prorotocl | int64 |
| Flow Duration | int64 |
| Total Fwd Packets | int64 |
| Total Backward Packets | int64 |
| Total Length of Fwd Packets | float64 |
| Total Length of Bwd Packets | float64 |
| Fwd Packet Length Max | float64 |

| Features | Type |
|---|---|
| Fwd Header Length | int64 |
| Bwd Header Length | int64 |
| Fwd Packets/s | float64 |
| Bwd Packets/s | float64 |
| Min Packet Length | float64 |
| Max Packet Length | float64 |
| Packet Length Mean | float64 |
| Packet Length Std | float64 |
| Packet Length Variance | float64 |
| FIN Flag Count | int64 |
| SYN Flag Count | int64 |
| RST Flag Count | int64 |
| PSH Flag Count | int64 |

| | |
|---|---|
| Fwd Packet Length Min | float64 |
| Fwd Packet Length Mean | float64 |
| Fwd Packet Length Std | float64 |
| Bwd Packet Length Max | float64 |
| Bwd Packet Length Min | float64 |
| Bwd Packet Length Mean | float64 |
| Bwd Packet Length Std | float64 |
| Flow Bytes/s | float64 |
| Flow Packets/s | float64 |
| Flow IAT Mean | float64 |
| Flow IAT Std | float64 |
| Flow IAT Max | float64 |
| Flow IAT Min | float64 |
| Fwd IAT Total | float64 |
| Fwd IAT Mean | float64 |
| Fwd IAT Std | float64 |
| Fwd IAT Max | float64 |
| Fwd IAT Min | float64 |
| Bwd IAT Total | float64 |
| Bwd IAT Mean | float64 |
| Bwd IAT Std | float64 |
| Bwd IAT Max | float64 |
| Bwd IAT Min | float64 |
| Fwd PSH Flags | int64 |
| Bwd PSH Flags | int64 |
| Fwd URG Flags | int64 |
| Bwd URG Flags | int64 |

| | |
|---|---|
| ACK Flag Count | int64 |
| URG Flag Count | int64 |
| CWE Flag Count | int64 |
| ECE Flag Count | int64 |
| Down/Up Ratio | float64 |
| Average Packet Size | float64 |
| Avg Fwd Segment Size | float64 |
| Avg Bwd Segment Size | float64 |
| Fwd Header Length | int64 |
| Fwd Avg Bytes/Bulk | int64 |
| Fwd Avg Packets/Bulk | int64 |
| Fwd Avg Bulk Rate | int64 |
| Bwd Avg Bytes/Bulk | int64 |
| Bwd Avg Packets/Bulk | int64 |
| Bwd Avg Bulk Rate | int64 |
| Subflow Fwd Packets | int64 |
| Subflow Fwd Bytes | int64 |
| Subflow Bwd Packets | int64 |
| Subflow Bwd Bytes | int64 |
| Init_Win_bytes_forward | int64 |
| Init_Win_bytes_backward | int64 |
| act_data_pkt_fwd | int64 |
| min_seg_size_forward | int64 |
| Active Mean | float64 |
| Active Std | float64 |
| Active Max | float64 |
| Active Min | float64 |
| Idle Mean | float64 |
| Idle Std | float64 |
| Idle Max | float64 |
| Idle Min | float64 |
| Inbound | int64 |
| Label | int64 |

**Tableau 3-3:** The set of characteristics used for the detection.

For the test dataset we had worked with the same techniques used in the train dataset the table **3-4** shows the attacks name and the number of instances in the test set:

| Name | Number of instances |
|---|---|
| MSSQL | 111760 |
| NetBIOS | 105680 |
| UDP | 104466 |
| LDAP | 102480 |
| Syn | 102257 |
| BENIGN | 48163 |

**Tableau 3-4:** Number of instances from each attack in test dataset

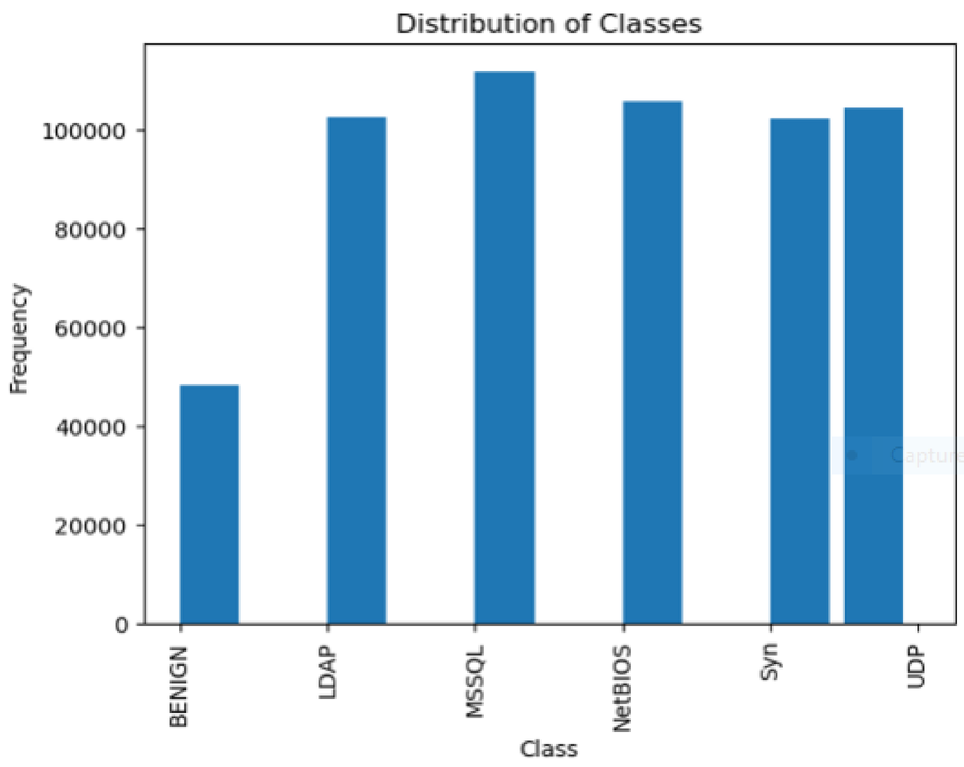

**Figure 3.4:** Classes distribution for test set

The number of selected features in the dataset 83 from 89 each has its weight and it is important in the model learning process for intrusion detection:

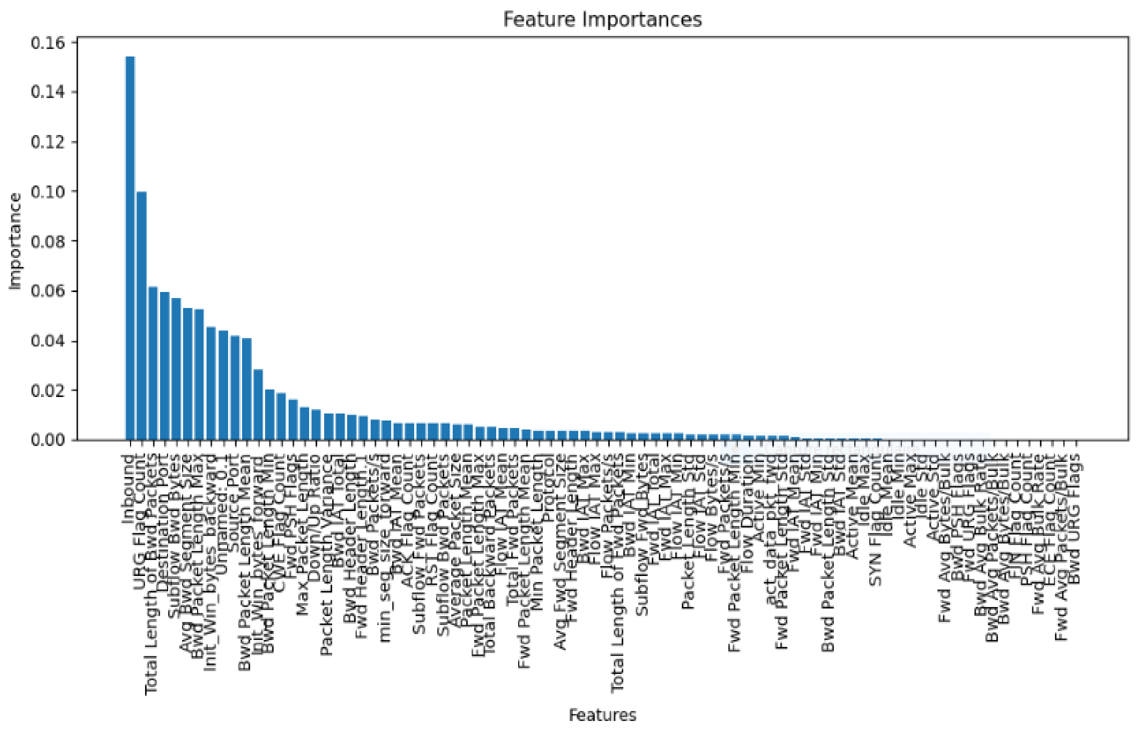

**Figure 3.5:** Features importance in CIC DDOS 2019 dataset

## 3.8 Models classification

In this work we have tried three models Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) models were used in identifying Distributed Denial of Service (DDoS) threats. These models were chosen owing to their shown ability to handle complicated data patterns and sequential information. The purpose was to investigate their performance in properly recognizing DDoS attacks and separating them from regular network traffic.

### 3.8.1 DNN model

We have train the DNN model using 128 in the batches size running 10 epochs with 70% for training and 30% for validation, the figure 6 and 7 shows the training and the validation accuracy over epochs, and the training and the validation loss over epochs for DNN model in CIC DDos 2019 dataset

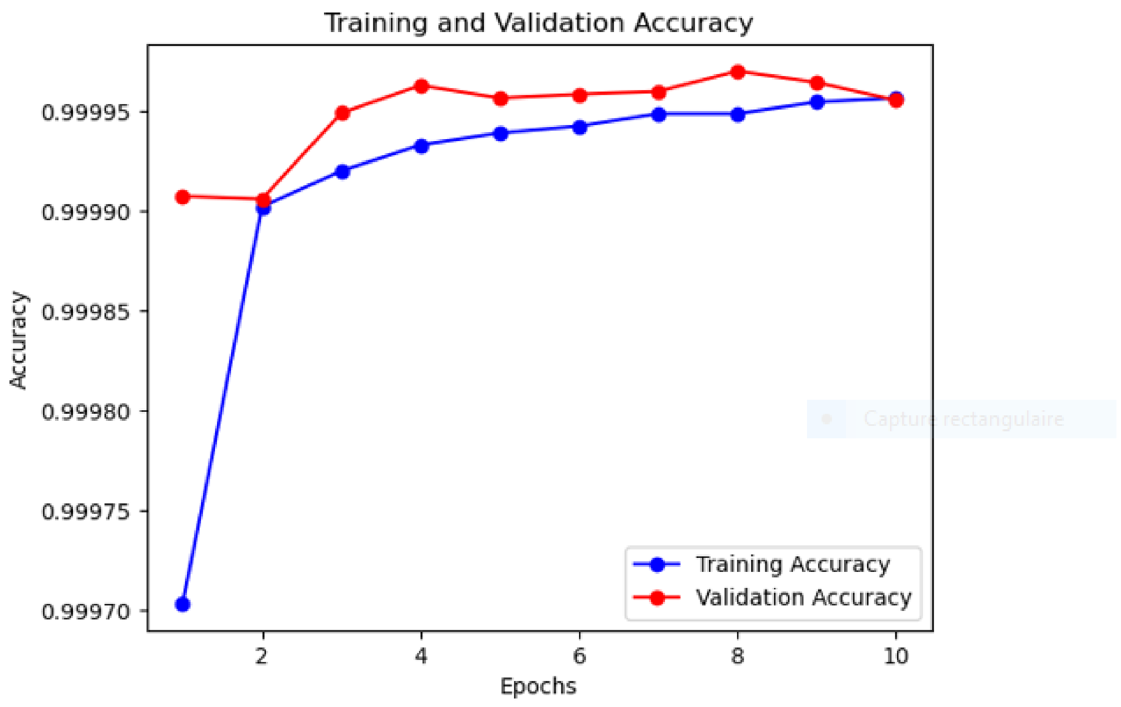

**Figure 3.6:** Graphical representation for DNN model training and validation accuracy

The figure **3.6** shows the highest training accuracy in epoch 10 by 99.995% and the lowest accuracy was in epoch 1 by 99.970%, for the validation accuracy the highest was in epoch 8 by 100% and the lowest was in epoch 1 by 99.990%.

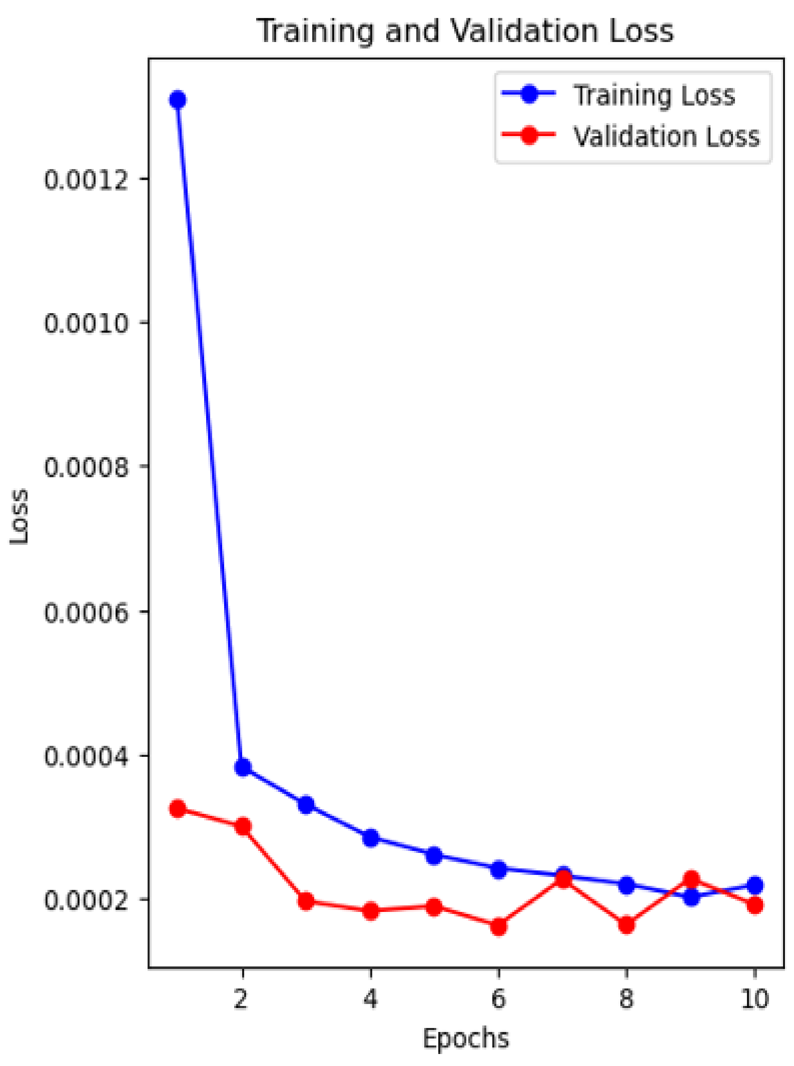

**Figure 3.7:** Graphical representation for DNN model training and validation loss

The figure **3.7** shows the training loss and the validation loss running DNN model over 10 epochs, the highest loss in the training in epoch 1 by 0.0013 and the lowest loss in the 9th epoch by 0.00023, the validation highest loss is 0.00031 and the lowest loss is 6th epoch by 0.0002

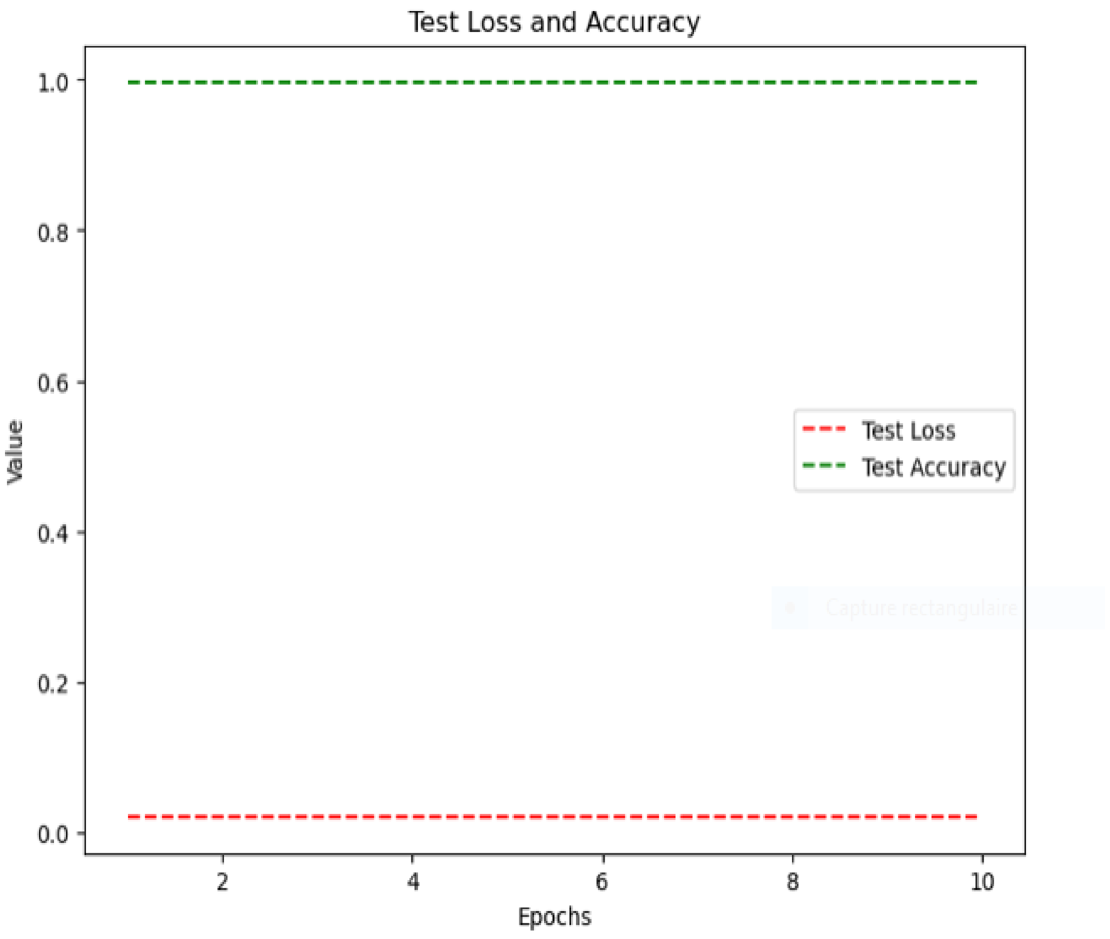

**Figure 3.8:** Graphical representation of DNN test loss and accuracy

The figure **3.8** represents a loss and accuracy of the testing data set on the DNN trained model which show an accuracy of 99.76% and a 0.013% loss.

| | Precision | Recall | F1-score | support |
|---|---|---|---|---|
| 0 | 97% | 96% | 96.5% | 47593 |
| 1 | 100% | 100% | 100% | 507425 |
| Accuracy | | | 99% | 555018 |
| Macro avg | 99% | 98% | 98% | 555018 |
| 1weighted avg | 99.7% | 99.6% | 99.6% | 555018 |

**Tableau 3-5 :** Classification report for DNN model

The evaluation of the model is depending on those metrics showed in table **3-5**, the table shows the precision metric as it is the accuracy of correctly predicted DDoS attacks out of all predicted attacks, instance reached a precision of 100% for class 1 and 97% for class 0, all detected attacks were in fact DDoS attacks.

The recall metric measures how well the model can distinguish DDoS attacks from all other types of attacks. The model successfully identified the majority of DDoS assaults in both classes, with recalls of 96% for class 0 and 100%, respectively.

The F1-score is a metric that combines recall and accuracy into one number and offers a fair evaluation of the model's performance. The DNN model has good accuracy in recognizing DDoS attacks, with an F1-score of 96.5% for class 0 and 100% for class 1.

The support column shows the number of instances of each class in the dataset, which in this case are 47,593 for class 0 and 507,425 for class 1.

Overall, the DNN model performs exceptionally well in identifying DDoS attacks, with an accuracy of 100%. The model's ability to generalize effectively across classes is demonstrated by the macro average F1-score of 99%, while the weighted average F1-score of 99.7% reflects the model's skill in reliably recognizing DDoS attacks throughout the whole dataset.

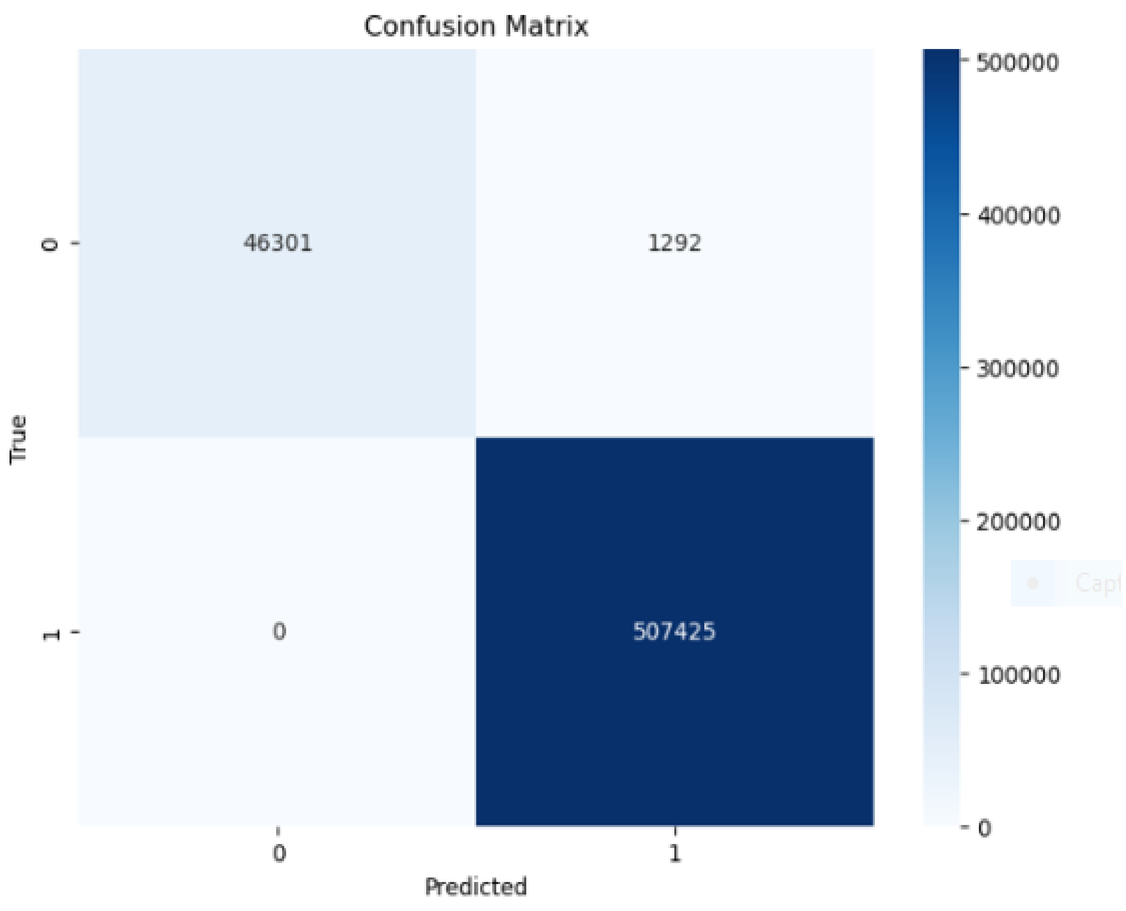

**Figure 3.9:** Illustration of DNN confusion matrix

The confusion matrix provides a detailed breakdown of the model's predictions compared to the actual ground truth labels. The confusion matrix in this scenario shows two classes: class 0, which represents non-DDoS traffic, and class 1, which represents DDoS attacks. The matrix is made up of four parts:

**True Positives (TP):** The number of accurately anticipated DDoS attacks is represented in the bottom-right element. In this scenario, the model correctly recognized all DDoS attempts, yielding a total of 507,425.

**False Positives (FP):** The number of non-DDoS incidents mistakenly identified as DDoS attacks is shown in the top-right element. In this case, the model incorrectly identified 1,292 non-DDoS incidents as DDoS assaults.

**False Negatives (FN):** The number of genuine DDoS attacks missed by the model is indicated in the bottom-left element. Surprisingly, the confusion matrix displays no false negatives, indicating that the model correctly identified all cases of DDoS.

**True Negatives (TN):** The number of accurately anticipated non-DDoS incidents is represented in the bottom-left element. The model correctly recognized 46,301 non-DDoS incidents in this scenario.

The ROC curve is a graphical depiction of a binary classification model's performance. At various categorization thresholds, it depicts the trade-off between the true positive rate (TPR) and the false positive rate (FPR).

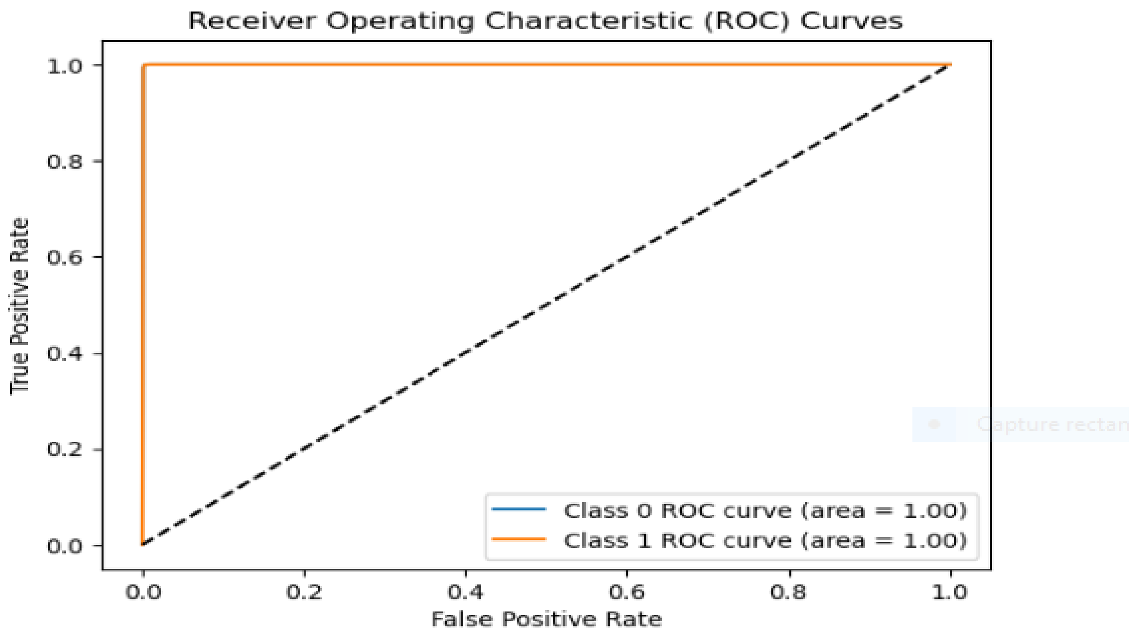

**Figure 3.10:** The Roc curve of DNN model

In Figure **3.10**, the curve forms a 90-degree angle above the center line. This shape indicates that the classification model has a high true positive rate (TPR) but also a high false positive rate (FPR). While the high TPR implies a good detection of positive cases, the high FPR suggests a significant number of false positive predictions.

### 3.8.2 CNN model

We trained the CNN model on CIC DDos 2019 dataset using 500 batches and 10 epochs, with 70% for training and 30% for validation. Figures and show the training and validation accuracy over epochs, as well as the training and validation loss over epochs.

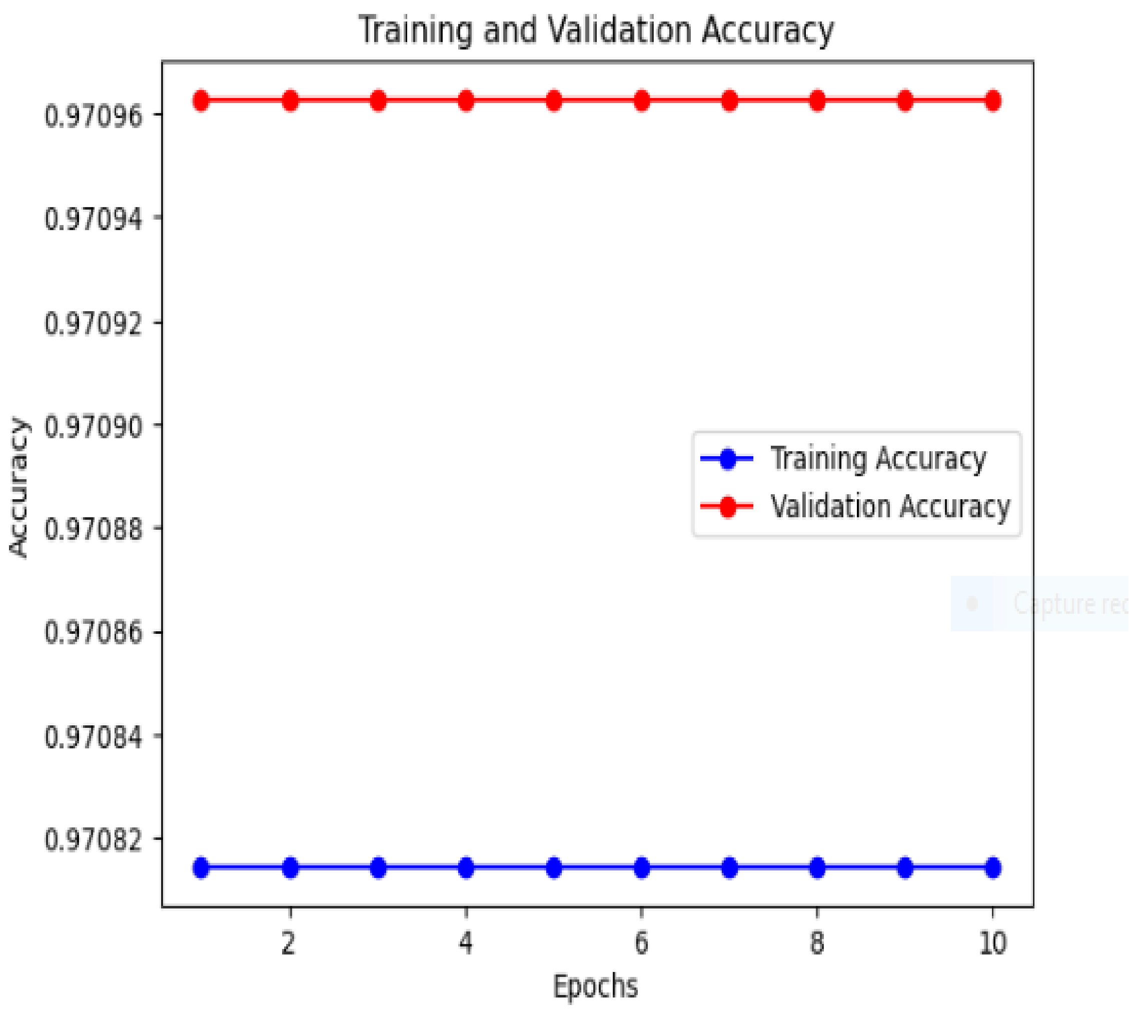

**Figure 3.11:** Graphical representation of CNN model training and validation accuracy

Figure **3.11** indicates that the maximum training accuracy was 97.08% in all epoch 10. The best validation accuracy was 97.10% in all 10 epochs.

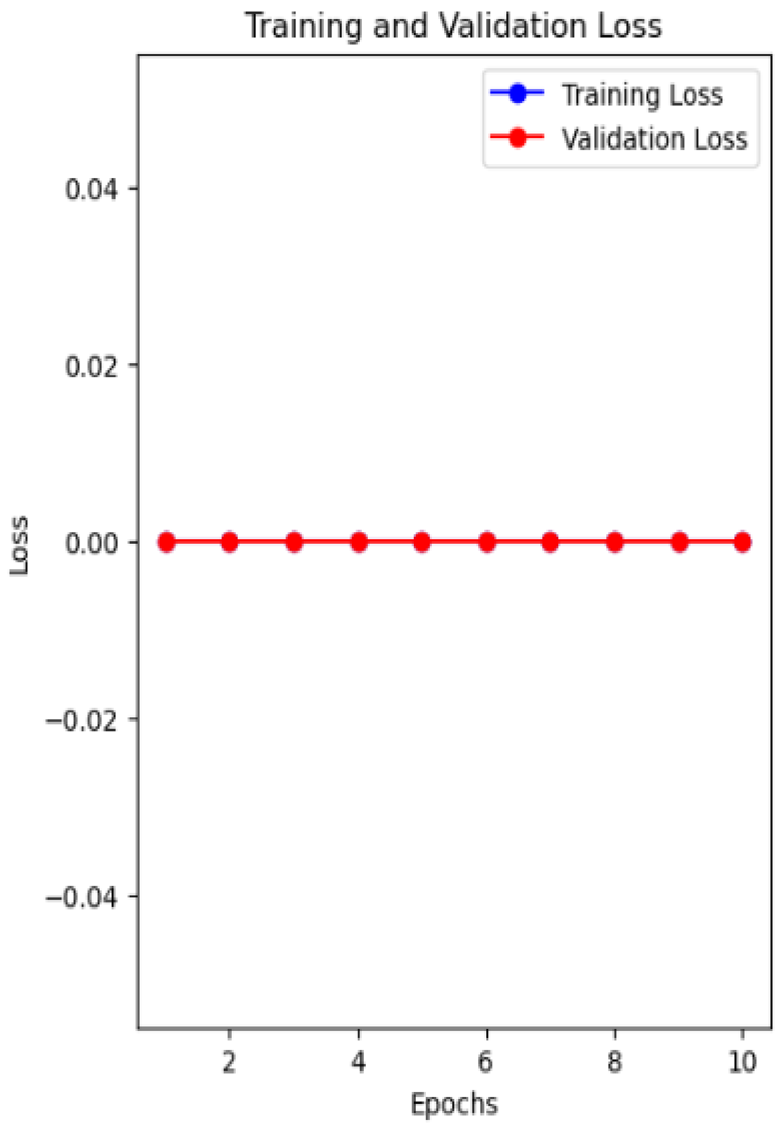

**Figure 3.12:** Graphical representation of CNN model training and validation accuracy

The figure **3.12** below illustrates the training and validation losses for a CNN model trained over ten epochs. In each epoch, the training loss remained constant at 00%. Similarly, the validation loss had a maximum value of 00% throughout all epochs.

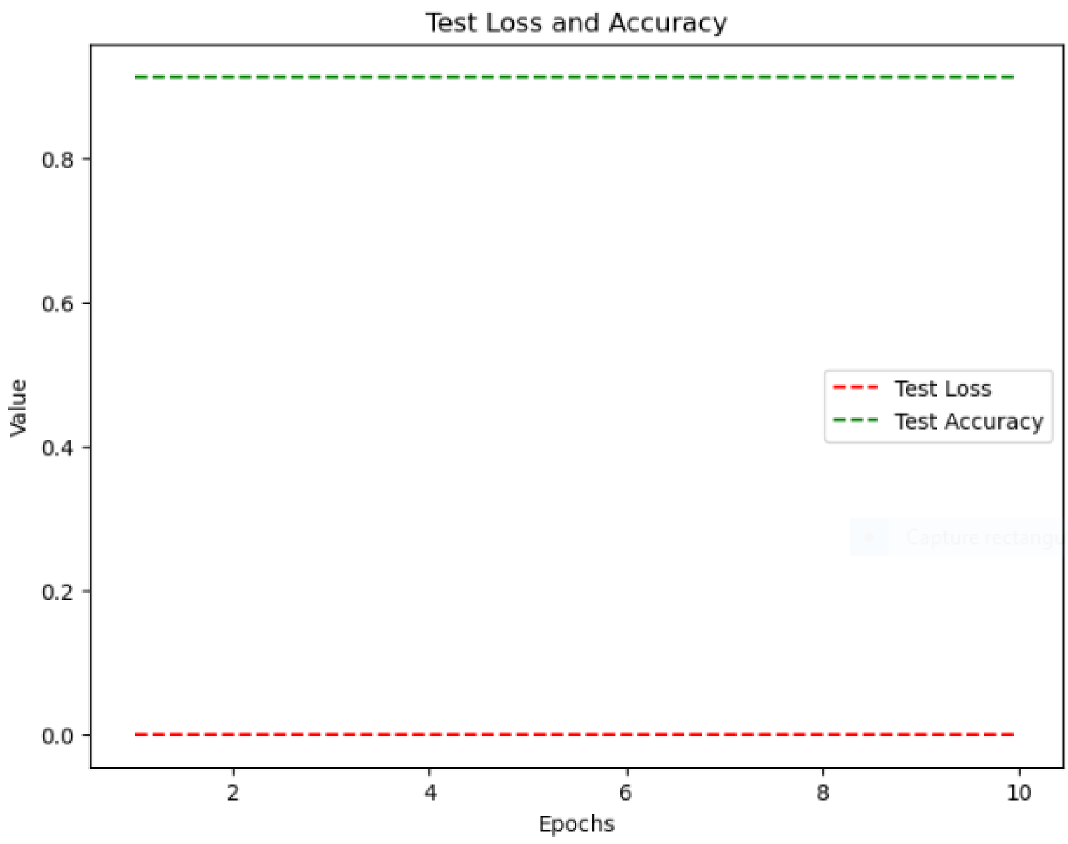

**Figure 3.13:** Graphical representation of test loss and accuracy for CNN model

Figure **3.13** shows the loss and accuracy of the testing data set on the CNN trained model, which shows an accuracy of 91.42% and a loss of 00%.

| | Precision | Recall | F1-score | support |
|---|---|---|---|---|
| 0 | 00% | 00% | 00% | 47593 |
| 1 | 91% | 100% | 95% | 507425 |
| Accuracy | | | 91% | 555018 |
| Macro avg | 46% | 50% | 48% | 555018 |
| 1weighted avg | 84% | 91% | 87% | 555018 |

**Tableau 3-6:** Classification report for CNN model

In Table **3-6**, we observe the precision values, which measure the accuracy of the model in making positive predictions. The precision for class 0 is recorded as 00%, indicating that there were no true positive predictions for class 0. On the other hand, the precision for class 1 is recorded as 91%, suggesting that 91% of the positive predictions for class 1 were accurate.

The recall for class 0 is given as 00%, suggesting that the model did not identify any true positive cases for this class. The recall for class 1 is 100%, suggesting that the model correctly detected all positive cases in class 1.

The F1-score for class 0 is 00%, suggesting poor performance owing to inadequate accuracy and recall. The F1-score for class 1 is 95%, suggesting high performance with a good mix of precision and recall.

Support**:** The number of instances in each class is indicated in the support column. There are 47,593 instances of class 0 and 507,425 examples of class 1 in this scenario

Accuracy**:** The total correctness of the model's predictions is represented by accuracy. The stated accuracy is 91%, suggesting that the model correctly predicted 91% of the time.

Macro avg**:** Macro avg computes the average of the metrics (precision, recall, and F1-score) for all classes, giving each class equal weight. Precision, recall, and F1-score are given as 46%, 50%, and 48%, respectively.

Weighted avg**:** A weighted average produces metrics by taking into account the support of each class, resulting in a weighted assessment of performance. Precision, recall, and F1-score are given as 84%, 91%, and 87%, respectively, based on weighted averages.

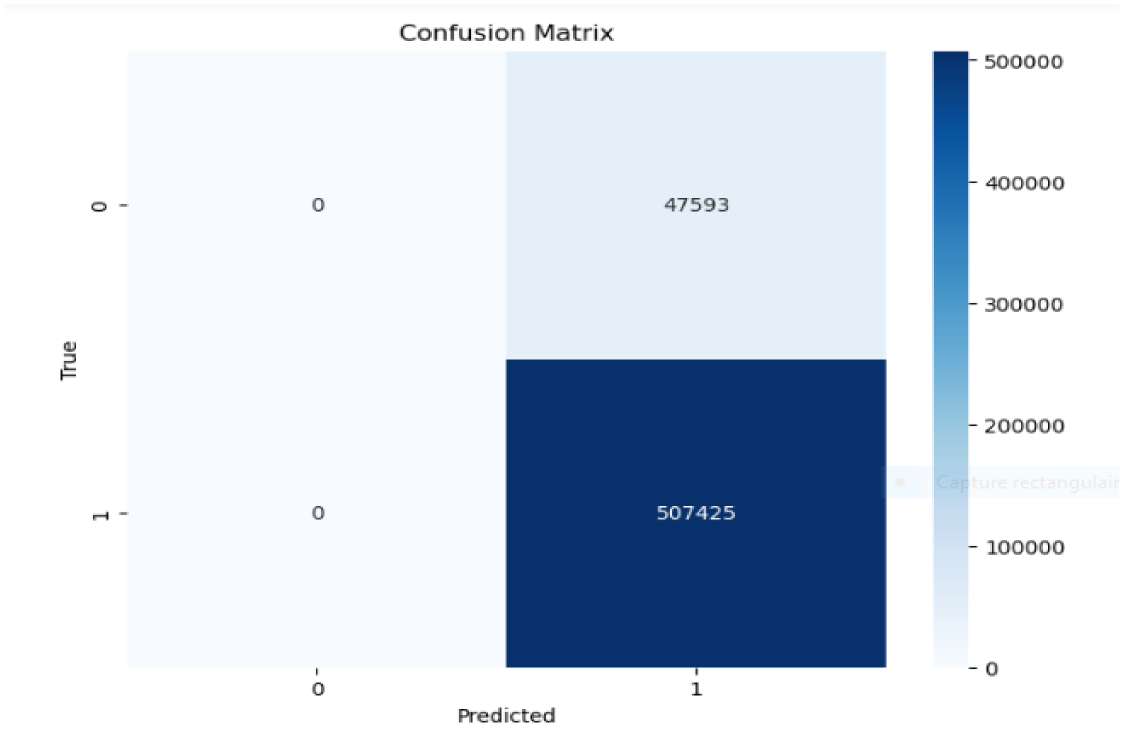

**Figure 3.14:** Illustration of CNN confusion matrix

The figure **3.14** presented confusion matrix relates to a Convolutional Neural Network (CNN) model's performance in identifying Distributed Denial of Service (DDoS) assaults:

**True Positives (TP**): The number of accurately anticipated DDoS attacks is indicated in the matrix's bottom-right entry. In this scenario, the number is 507,425, indicating that the CNN model correctly recognized all DDoS attacks.

**False Positives (FP):** The number of non-DDoS incidents mistakenly identified as DDoS assaults is represented in the top-right element. The value 47593 in this confusion matrix indicates that the CNN model incorrectly categorizes non-DDoS incidents as DDoS assaults.

**False Negatives (FN):** The number of genuine DDoS attacks missed by the CNN model is indicated in the bottom-left element. The result is 0 in this scenario, suggesting that the model correctly recognized all instances of DDoS attacks.

**True Negatives (TN):** The number of accurately anticipated non-DDoS incidents is represented in the top-left element. This position's value is 0, indicating that the CNN model didn’t categorize any of the non-DDoS incidents.

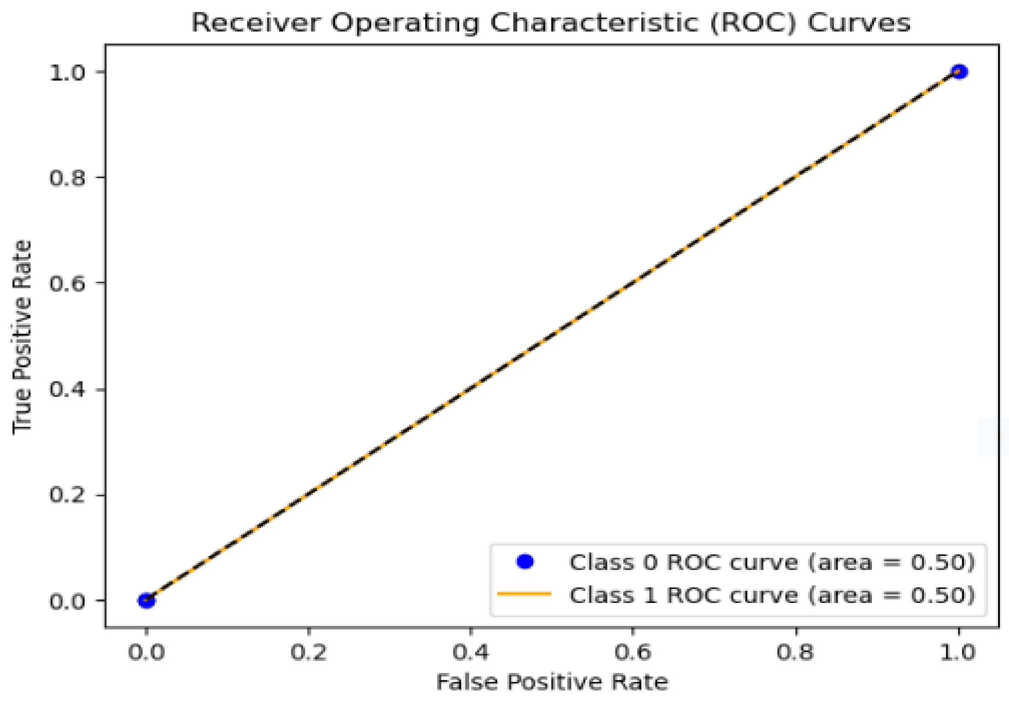

**Figure 3.15:** The Roc curve of CNN model

The ROC curve for the CNN model is a straight line near the midpoint it implies that the model's predictions lack discriminating ability in distinguishing between positive and negative events. This means that the model is failing to differentiate the classes properly and is producing random or arbitrary predictions.

The TPR (true positive rate) and FPR (false positive rate) are almost similar across multiple categorization levels in this case, resulting in a diagonal line. This implies that the model performs no better than random chance in properly identifying positive examples and avoiding false positives.

### 3.8.3 RNN model

On the third sample we trained the RNN model On the CIC DDos 2019 dataset, with 800 batch sizes and 10 epochs, with 70% for training and 30% for validation. Figures 6 and 7 demonstrate the training and validation accuracy as well as the training and validation loss as a function of epoch.

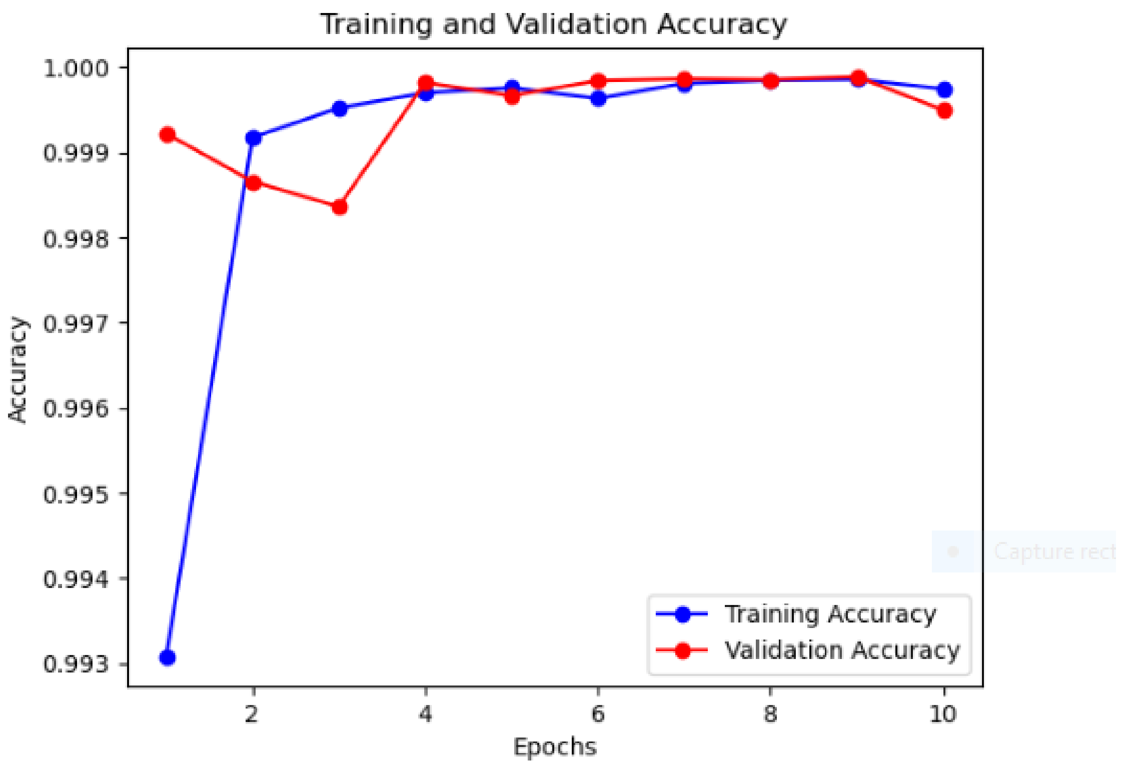

**Figure 3.16:** Graphical representation for RNN model training and validation accuracy

The figure **3.16** shows the highest training accuracy in epoch 9 by 99.99% and the lowest accuracy was in epoch 1 by 99.31%, for the validation accuracy the highest was in epoch 4 by 99.99% and the lowest was in epoch 3 by 99.84%.

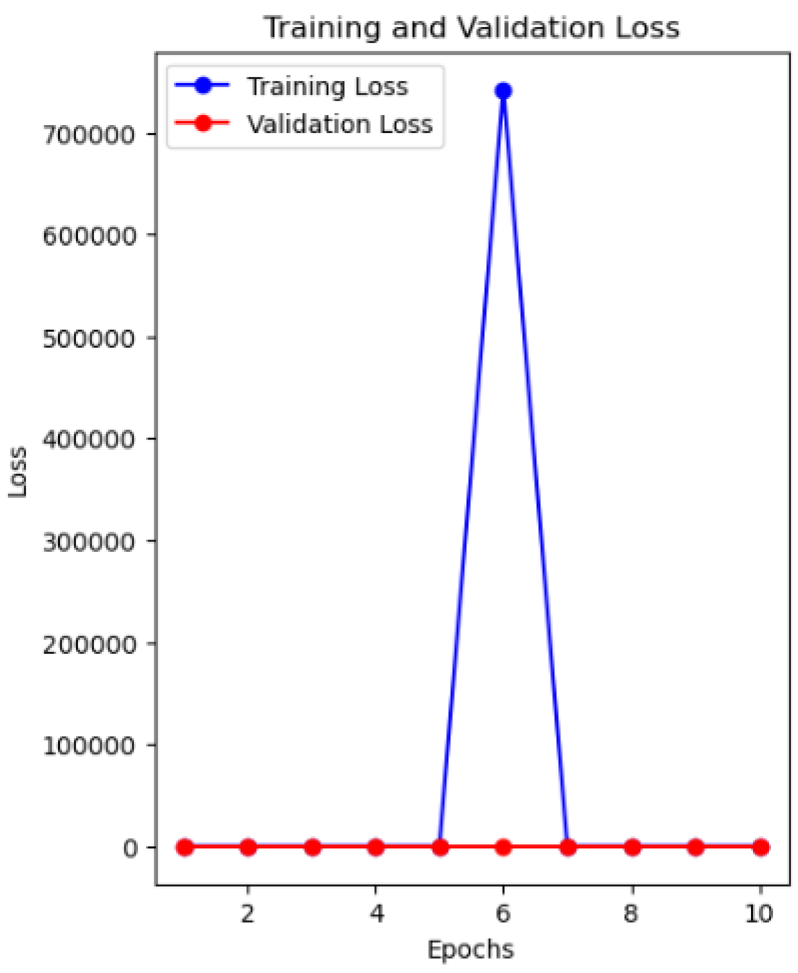

**Figure 3.17:** Graphical representation for RNN training and validation loss

Figure **3.17** displays the training loss and validation loss of the DNN model over 10 epochs. The highest training loss was observed in epoch 6, reaching 742019. The lowest training loss occurred in epoch 5, with a value of 0.0011. As for the validation loss, it was highest in epoch 5, recorded at 7.9652, while the lowest validation loss was observed in the10th epoch, with a value of 0.0012.

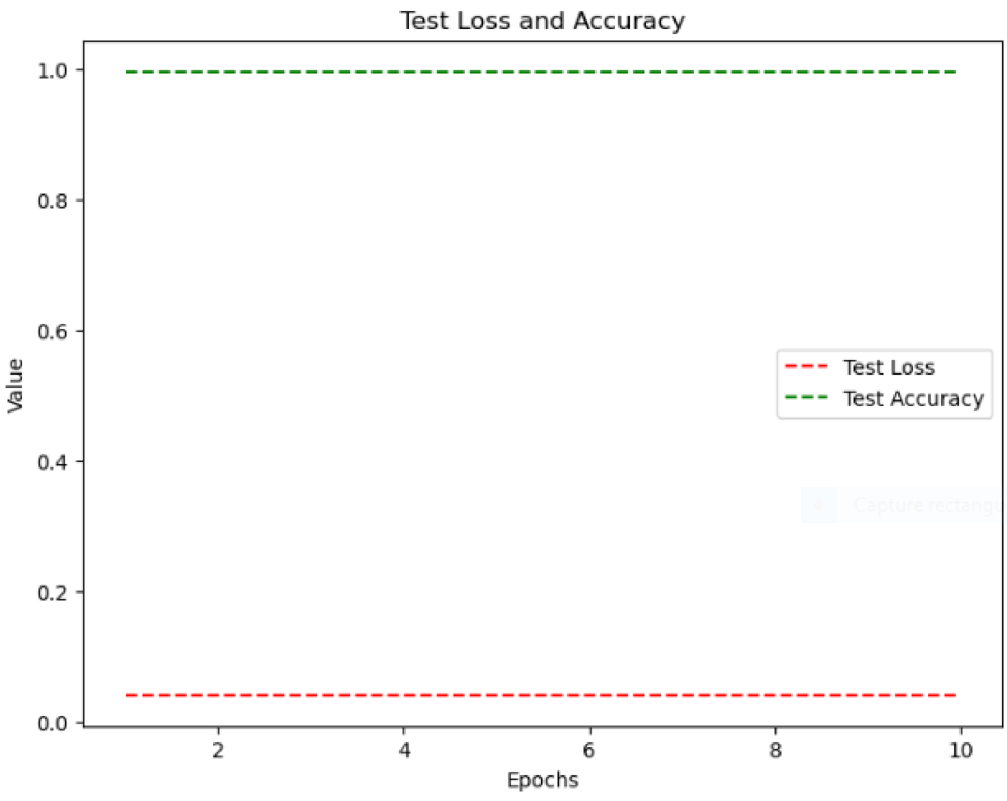

**Figure 3.18:** Graphical representation of RNN model test loss and accuracy

The figure **3.18** represents a loss and accuracy of the testing data set on the DNN trained model which show an accuracy of 99.63% and a 0.041% loss.

| | Precision | Recall | F1-score | support |
|---|---|---|---|---|
| 0 | 99% | 96% | 97% | 47593 |
| 1 | 99% | 99% | 99% | 507425 |
| Accuracy | | | 99% | 555018 |
| Macro avg | 99% | 97.5% | 98% | 555018 |
| 1weighted avg | 99% | 98.7% | 98.8% | 555018 |

**Tableau 3-7:** classificationn report for RNN model

Table **3.7** describes the classification report for the RNN model, particularly the precision values for class 0 and class 1. A precision of 99% for class 0 indicates that 99% of the positive predictions for class 0 were accurate. Similarly, a precision of 99% for class 1 suggests that all positive predictions for class 1 were accurate.

Recall detects positive cases. The recall for class 0 is 96%, suggesting that the model successfully detected 96% of the positive cases in class 0. The recall for class 1 is 99%, suggesting that the model correctly detected all positive cases in class 1.

The F1-score the harmonic mean of accuracy and recall, the F1-score for class 0 is 97%, indicating a good combination of precision and recall. The F1-score for class 1 is 99%, suggesting a perfect combination of precision and recall.

Support is the number of instances in each class is indicated there are 47,593 instances of class 0 and 507,425 examples of class 1 in this scenario.

Accuracy is the total correctness of the model's predictions is represented by accuracy. The stated accuracy is 99%, suggesting that the model correctly predicted 99% of the time.

Macro avg computes the average of the metrics (precision, recall, and F1-score) the macro-averaged precision, recall, and F1-score are given as 99%, 97.5%, and 98%.

Weighted average produces metrics by taking into account the support of each class, resulting in a weighted assessment of performance. Precision, recall, and F1-score are all given as 99%, 98.7%, and 98.8%, respectively.

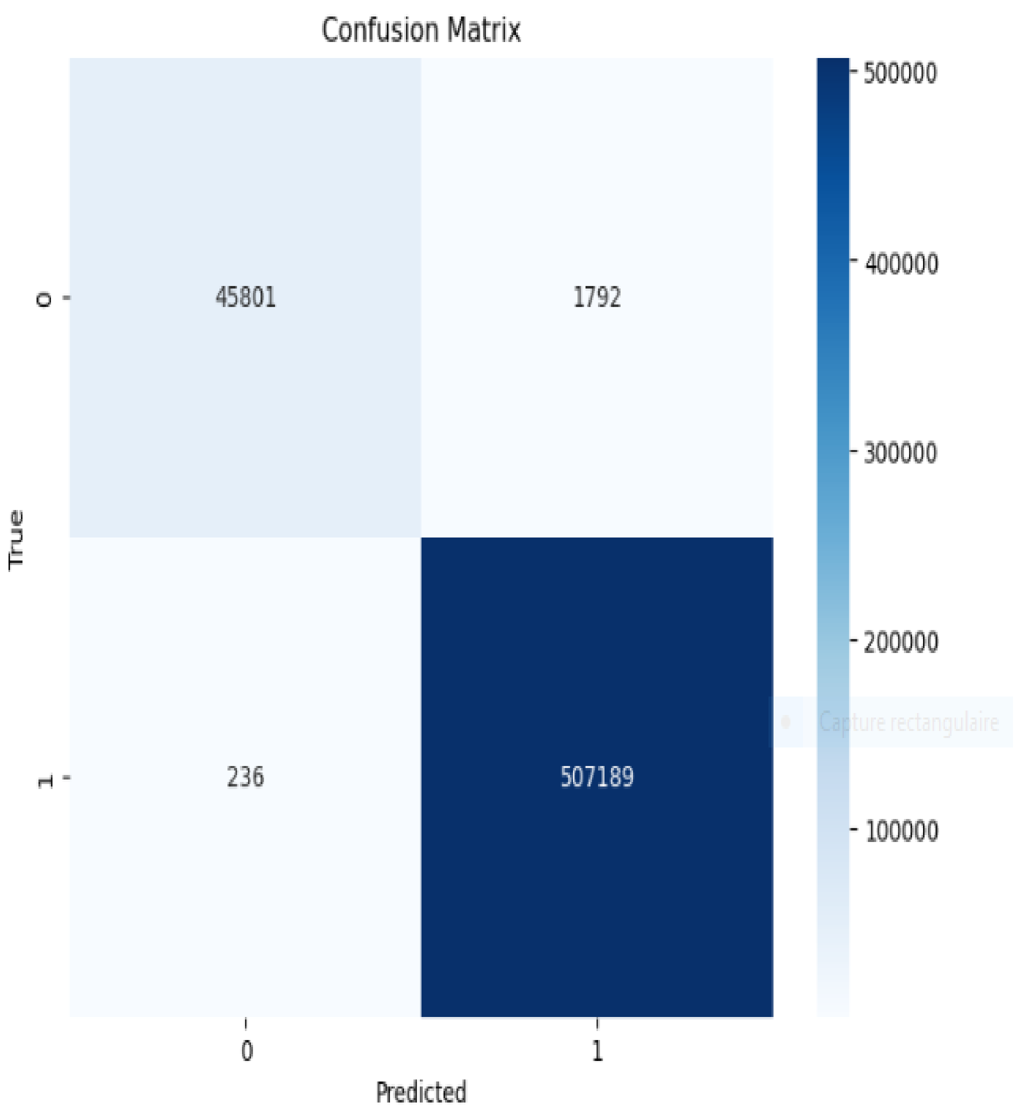

**Figure 3.19:** Illustration of RNN confusion matrix

**True Positives (TP):** The number of accurately anticipated cases of class 1 is represented in the bottom-right element. This position's value is 507,189, suggesting that the model correctly categorized 507,189 DDoS attacks.

**False Positives (FP):** The number of occurrences of class 1 that were wrongly categorized as class 0 is indicated in the bottom-left element. The value of 236 in this confusion matrix indicates that the model failed to detect 236 instances of DDoS assaults.

**False Negatives (FN):** The number of incidents that were wrongly categorized as class 1 (DDoS assaults) but are in fact class 0. The number in this case is 1,792, indicating that the model incorrectly identified 1,792 instances of non-DDoS traffic as DDoS assaults.

**True Negatives (TN):** The number of accurately anticipated instances of class 0 (non-DDoS traffic) the number is 45,801 in this case, meaning that the model properly recognized 45,801 instances of non-DDoS traffic.

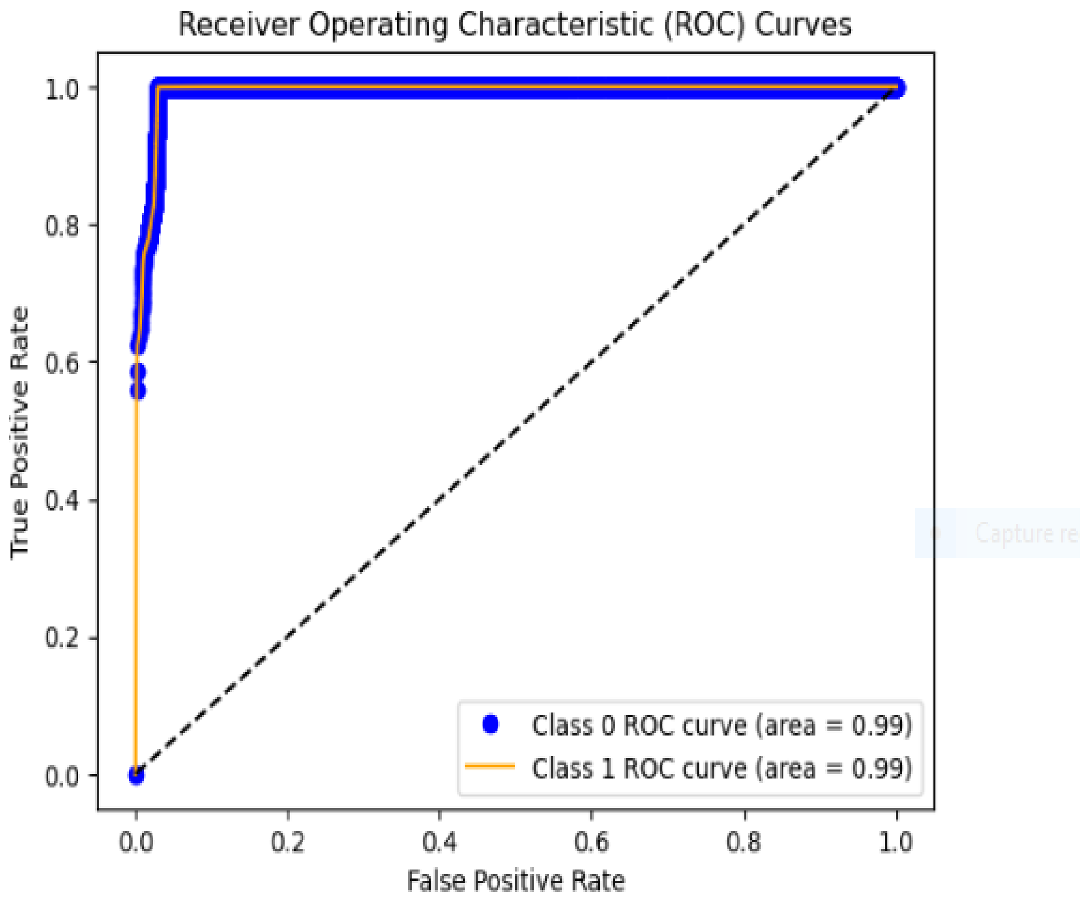

**Figure 3.20 :** The Roc curve of the RNN model

Figure **3.20** shows a curve that makes an almost 90-degree angle above the center line, indicating that the categorization model performed admirably. This shape denotes a model with a high true positive rate (TPR) and a low false positive rate (FPR).

## 3.9 Conclusion

We have used the CICDDoS2019 Data-set to train three discriminating deep learning models for identifying DDoS attacks

| | Macro Avg (all classes) | | | TPR | FPR | TNR | FNR |
|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1score | | | | |
| DNN | 99% | 98% | 98% | 507,425. | 0 | 46,301 | 1,292 |
| CNN | 46% | 50% | 48% | 507,425 | 0 | 0 | 47,593 |
| RNN | 99% | 97.5% | 98% | 507,189 | 236 | 45,801 | 1,792 |

**Tableau 3-8:** Results of the 3-model proposed

It is worth noting that this work was fraught with difficulties and complexity. One of the major obstacles was the hardware restrictions (processor, memory, GPU) which affected model training and evaluation. Insufficient computing resources may have hampered the capacity to investigate more complex structures and larger datasets, potentially resulting in even greater performance.

In this work, we have used a subset of data using the Random under sampling for 3 different classifications models. Deferent oversampling strategies with varying percentages were explored to address the data imbalance problem, and many tests were conducted to discover the optimum model with the best results by trying different model hyperparameters for each type of classification. Using the same training and testing datasets. Compared the performance of three models for DDoS intrusion detection: DNN, CNN, and RNN, for the task of DDoS intrusion detection based on CICDDOS2019 dataset. Our research looked at precision, recall, F1-score, true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), and false negative rate (FNR).

The DNN got the highest macro average F1-score of 98% among the models, suggesting exceptional overall performance in identifying DDoS assaults. It had a TPR of 0.98 and an FPR of 0.01, demonstrating its ability to detect positive cases accurately while retaining a low false positive rate. These findings indicate that the DNN model is well-suited for DDoS intrusion detection tasks.

The CNN model, on the other hand, produced a lower macro average F1-score of 48%. While it had a balanced TNR and FNR, suggesting its ability to categorize positive cases only, it had 00% accuracy on detecting negative cases and a 0 on TNR. This implies that the CNN model struggled to detect DDoS attacks properly and had a higher potential for false positives.

The RNN model obtained the same macro average F1-score as the DNN model, suggesting its efficacy in detecting DDoS intrusions. It had a TPR of 0.98, a TNR and FNR that were both balanced. However, it had a significantly larger number of false negatives than the DNN model, resulting in somewhat poorer accuracy.

In conclusion, based on the results obtained in this study, the DNN model demonstrated superior performance in DDoS intrusion detection compared to the CNN and RNN model, however, additional research is needed to explore more advanced designs and overcome hardware and data complexity difficulties in order to improve the accuracy and reliability of DDoS intrusion detection systems.

# General conclusion

This thesis has addressed the critical problem of intrusion detection in network security, specifically focusing on the detection and prevention of distributed denial of service (DDoS) attacks. Thorough research and analysis, we have explored and implemented various deep learning algorithms, including deep neural network (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and fine-tuning of pre-trained models using transfer learning.

This work lays its foundation on a comprehensive understanding of the dataset and the utilization of suitable preprocessing techniques to prepare the data for analysis. By implementing deep learning models, we have effectively demonstrated their capability to capture and analyze complex patterns within network traffic data. The results demonstrate a high level of accuracy in detecting and mitigating network intrusions, highlighting the effectiveness of deep learning techniques in the field of intrusion detection.

However, it is critical to recognize the limitations of this study. One primary challenge was the availability of a large-scale dataset appropriate for training deep learning models. Collecting and classifying a large amount of high-quality data can be time-consuming and resource-intensive. Additionally, the computational requirements for training deep learning models can be demanding, and our hardware limitations-imposed restrictions on the scale and complexity of the models we could utilize.

Future research in intrusion detection with deep learning should address these limitations by improving data preprocessing techniques, exploring interpretability methods, and investigating scalability for real-time deployment. By addressing these challenges and limitations, future research can build upon these findings and advance the field of intrusion detection with deep learning leading to more robust and effective solutions. This research can be expanded in the future by including the model in a robust and scalable IDS framework. The model can be effectively incorporated into real-world network environments by leveraging existing IDS infrastructure, such as traffic capture techniques and data processing pipelines. Future development can involve integrating automated response mechanisms, such as traffic rerouting or rate limiting. However, it may be necessary to establish effective alerting mechanisms and continuously improve the model's accuracy through regular updates and evaluations.

# References

[1] ‘Importance of Network Security: Safety in the Digital World’. https://www.ecpi.edu/blog/importance-of-network-security-safety-in-the-digital-world (accessed May 13, 2023).

[2] Logpoint, ‘What is a Cyber attack? Learn why cyber attacks happen and how to avoid them’, *Logpoint*, Nov. 30, 2020. https://www.logpoint.com/en/blog/cyber-attack/ (accessed May 13, 2023).

[3] ‘What is Cybersecurity & Importance of Cyber Security | Simplilearn’, *Simplilearn.com*. https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-cyber-security (accessed May 08, 2023).

[4] ‘KASSEM.pdf’.

[5] ‘What is a Cyber Attack? Definition, Examples and Prevention TechTarget’, *Security*. https://www.techtarget.com/searchsecurity/definition/cyber-attack (accessed May 12, 2023).

[6] Y. Li and Q. Liu, ‘A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments’, *Energy Reports*, vol. 7, pp. 8176–8186, Nov. 2021, doi: 10.1016/j.egyr.2021.08.126.

[7] Logpoint, ‘What is a Cyber attack? Learn why cyber attacks happen and how to avoid them’, *Logpoint*, Nov. 30, 2020. https://www.logpoint.com/en/blog/cyber-attack/ (accessed May 12, 2023).

[8] ‘What is Spoofing?’, *Forcepoint*, Aug. 10, 2018. https://www.forcepoint.com/cyber-edu/spoofing (accessed May 13, 2023).

[9] ‘Top 10 Most Common Types of Cyber Attacks’, *https://blog.netwrix.com/*. https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/ (accessed May 13, 2023).

[10] ‘What is Ping of Death (PoD) | DDoS Attack Glossary | Imperva’, *Learning Center*. https://www.imperva.com/learn/ddos/ping-of-death/ (accessed May 15, 2023).

[11] ‘What Is a Botnet and Its Functionality? | Radware’. https://www.radware.com/cyberpedia/bot-management/botnet/ (accessed May 15, 2023).

[12] ‘DNS Tunneling - Definition, Examples, & Detection - ExtraHop | ExtraHop’. https://www.extrahop.com/resources/attacks/dns-tunneling/ (accessed May 15, 2023).

[13] ‘(PDF) Intrusion Detection: A Survey’. https://www.researchgate.net/publication/226650646_Intrusion_Detection_A_Survey (accessed May 17, 2023).

[14] S. Cooper, ‘10 top network intrusion detection tools for 2018’, *Comparitech*, Feb. 27, 2019. https://www.comparitech.com/net-admin/network-intrusion-detection-tools/ (accessed Jul. 13, 2023).

[15] S. Rathore, ‘International Journal of Current Trends in Engineering &amp; Technology Enhanced Method for Intrusion Detection over KDD Cup 99 Dataset’, Accessed: May 16, 2023. [Online]. Available: https://www.academia.edu/24357488/International_Journal_of_Current_Trends_in_Engineering_and_Technology_Enhanced_Method_for_Intrusion_Detection_over_KDD_Cup_99_Dataset

[16] A. Lazarevic, V. Kumar, and J. Srivastava, ‘Intrusion Detection: A Survey’, in *Managing Cyber Threats*, 2005, pp. 19–78. doi: 10.1007/0-387-24230-9_2.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.

[18] ‘Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2020). Deep learning for intrusion detection systems: A review. Journal of Information Security and Applications, 50, 102407’.

[19] ‘A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks | IEEE Journals & Magazine | IEEE Xplore’, May 20, 2023. https://ieeexplore.ieee.org/document/8066291 (accessed May 20, 2023).

[20] S. Madhavan and M. T. J. U. 24 janvier 2021 | P. 7 septembre 2017, ‘Deep learning architectures’, *IBM Developer*, May 20, 2023. https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/ (accessed May 20, 2023).

[21] S. J. Pan and Q. Yang, ‘A Survey on Transfer Learning’, *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, Art. no. 10, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[22] ‘What Is Transfer Learning? A Guide for Deep Learning | Built In’, May 20, 2023. https://builtin.com/data-science/transfer-learning (accessed May 20, 2023).

[23] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, ‘A Survey on Deep Transfer Learning’, *arXiv.org*, Aug. 06, 2018. https://arxiv.org/abs/1808.01974v1 (accessed May 20, 2023).

[24] A. L. Buczak and E. Guven, ‘A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection’, *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, Art. no. 2, 2016, doi: 10.1109/COMST.2015.2494502.

[25] R. Sommer and V. Paxson, ‘Outside the Closed World: On Using Machine Learning for Network Intrusion Detection’, in *2010 IEEE Symposium on Security and Privacy*, Oakland, CA, USA: IEEE, 2010, pp. 305–316. doi: 10.1109/SP.2010.25.

[26] Q. Hu, ‘A Survey of Adversarial Example Toolboxes’, in *2021 2nd International Conference on Computing and Data Science (CDS)*, Jan. 2021, pp. 603–608. doi: 10.1109/CDS52072.2021.00109.

[27] ‘What is a Reflection Amplification Attack? | NETSCOUT’. https://www.netscout.com/what-is-ddos/what-is-reflection-amplification-attack (accessed May 23, 2023).

[28] M. Shurman, R. Khrais, and A. Yateem, ‘DoS and DDoS Attack Detection Using Deep Learning and IDS’, *IAJIT*, vol. 17, no. 4A, pp. 655–661, Jul. 2020, doi: 10.34028/iajit/17/4A/10.

[29] ‘DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB’. https://www.unb.ca/cic/datasets/ddos-2019.html (accessed Jul. 14, 2023).

[30] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, ‘Deep Learning-Based Intrusion Detection for Distributed Denial of Service Attack in Agriculture 4.0’, *Electronics*, vol. 10, no. 11, p. 1257, May 2021, doi: 10.3390/electronics10111257.

[31] ‘Attaque DDoS SYN Flood’, *Cloudflare*. https://www.cloudflare.com/fr-fr/learning/ddos/syn-flood-ddos-attack/ (accessed Jul. 14, 2023).

[32] I. Sharafaldin, A. Habibi Lashkari, S. Hakak, and A. Ghorbani, ‘Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy’, Oct. 2019, pp. 1–8. doi: 10.1109/CCST.2019.8888419.

[33] ‘Figure 1. UDP Flooding attack.’, *ResearchGate*. https://www.researchgate.net/figure/UDP-Flooding-attack_fig1_327036867 (accessed Jul. 14, 2023).

[34] ‘Fig. 1. A diagram showing the relationship between the various level of...’, *ResearchGate*. https://www.researchgate.net/figure/A-diagram-showing-the-relationship-between-the-various-level-of-NTP-hierarchy-The-blue_fig1_333838061 (accessed Jul. 14, 2023).

[35] ‘These 6 DNS Attacks Threaten Your Business’, *Defence Intelligence Blog*, May 18, 2017. https://defintel.com/blog/index.php/2017/05/these-6-dns-attacks-threaten-your-business.html (accessed Jul. 14, 2023).

[36] ‘What is an LDAP Injection? Definition and How to Prevent’, *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/LDAP-injection (accessed May 27, 2023).

[37] ‘What is SNMP Reflection and Amplification | DDoS Attack Glossary | Imperva’, *Learning Center*. https://www.imperva.com/learn/ddos/snmp-reflection/ (accessed May 27, 2023).

[38] ‘A New DDoS Reflection Attack: Portmapper; An Early Warning to the Industry’. https://blog.lumen.com/a-new-ddos-reflection-attack-portmapper-an-early-warning-to-the-industry/ (accessed May 27, 2023).

[39] C. T. Bs. H. MIAP, ‘An introduction to Convolutional Neural Networks’, *Medium*, May 27, 2019. https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7 (accessed Jun. 04, 2023).

[40] ‘Recurrent Neural Network (RNN) Tutorial: Types and Examples [Updated] | Simplilearn’, *Simplilearn.com*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn (accessed Jun. 06, 2023).

[41] ‘Activation Function in Neural Network’. https://www.linkedin.com/pulse/activation-function-neural-network-prasad-deshmukh (accessed Jun. 07, 2023).

[42] J. Brownlee, ‘How to Choose an Activation Function for Deep Learning’, *MachineLearningMastery.com*, Jan. 17, 2021. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (accessed Jun. 07, 2023).

[43] E. Rodríguez *et al.*, ‘Transfer-Learning-Based Intrusion Detection Framework in IoT Networks’, *Sensors*, vol. 22, no. 15, p. 5621, Jul. 2022, doi: 10.3390/s22155621.

[44] ‘Fine-tuning a Neural Network explained’. https://deeplizard.com/learn/video/5T-iXNNiwIs (accessed Jun. 07, 2023).

[45] ‘Anaconda | Anaconda Training: A Learning Path for Data Scientists’, *Anaconda*, Dec. 04, 2017. https://www.anaconda.com/blog/anaconda-training-a-learning-path-for-data-scientists (accessed Jun. 03, 2023).

[46] ‘Project Jupyter’. https://jupyter.org (accessed Jun. 03, 2023).

[47] ‘pandas - Python Data Analysis Library’. https://pandas.pydata.org/ (accessed Jun. 03, 2023).

[48] ‘NumPy’. https://numpy.org/ (accessed Jun. 03, 2023).

[49] ‘Keras: Deep Learning for humans’. https://keras.io/ (accessed Jun. 03, 2023).

[50] ‘TensorFlow’. https://www.tensorflow.org/?hl=fr (accessed Jun. 03, 2023).

[51] ‘scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation’. https://scikit-learn.org/stable/ (accessed Jun. 03, 2023).

[52] ‘Matplotlib — Visualization with Python’. https://matplotlib.org/ (accessed Jun. 03, 2023).