الجـــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –

Faculté de TECHNOLOGIE

MASTER'S THESIS

Presented for the obtention of the **MASTER degree**

In: Telecommunications

Speciality: Networks and Telecommunications

By: Krim Sidi Mohammed & Yahlali Amira

Subject

Anomaly- Intrusion Detection Systems based on CSE-CIC-IDS2018 Dataset using Deep Learning Model

Publicly defended, on 05/06/2023, before the jury composed of:

M. BENDIMERAD Yacine	MCA	University of Tlemcen	President
M. MOUSSAOUI Djilali	MCA	University of Tlemcen	Reviewer
M. HADJILA Mourad	MCA	University of Tlemcen	Supervisor
Ms. FERHI Wafaa	PhD student	University of Tlemcen	Co-Supervisor

College year: 2022/2023

Dedication

I dedicate this master's thesis to the people who have been a constant source of inspiration, support, and encouragement throughout my academic journey.

To my parents, who instilled in me a love for learning and taught me the value of hard work. Your unwavering support and belief in my abilities have been the driving force behind my success. I am forever grateful for everything you have done for me and will always strive to make you proud.

To my professors, who have challenged and inspired me to push beyond my limits and pursue excellence. Your guidance, knowledge, and mentorship have been instrumental in shaping my academic and professional growth. I am honored to have learned from you.

To my friends and colleagues, for their constant encouragement, support, and camaraderie. Your presence has made this journey more enjoyable and memorable. I am grateful for the memories we have shared and the connections we have made.

Finally, I dedicate this thesis to myself, for my perseverance, determination, and resilience. This accomplishment is a testament to the hard work, sacrifice, and dedication I have put into my academic and personal growth. I am proud of myself and grateful for the opportunity to pursue my passion.

Sidi Mohammed

Dedication

This master's thesis is dedicated to the individuals who have impacted my academic journey in ways beyond measure.

To my parents, thank you for instilling in me the values of hard work, perseverance, and resilience, and for teaching me to never give up on my dreams.

To My brothers, thank you for being my best friends, and my source of laughter. I would not be where I am today without your love and encouragement. I hope to be a role model and an example to you.

To my colleague and dear friend ***Oussama****, thank you for being a constant source of support and encouragement. Your dedication to our success and your willingness to help in any way possible have been truly invaluable. I am grateful for the memories we shared and the academic challenges we faced together. Thank you for your friendship and support.*

To my dear friends ***Hanane****,* ***Ahlem****, and* ***Ghizlen*** *with whom I have shared the best and most pleasant moments throughout my university journey and all my colleagues.*

And finally, to myself, for the long hours, dedication, and hard work put into completing this journey. This accomplishment is a reminder of my capabilities, and I am proud of what I have achieved. With love and gratitude.

Amira

Acknowledgements

After giving thanks to God the Almighty and the Benevolent, we would like to express our heartfelt gratitude to everyone who participated in the realization of this thesis.

We would also like to thank our supervisor, ***Mr. M. Hadjila****, for his unwavering support and guidance throughout this research. His expertise and insights have been invaluable in shaping the direction of this thesis.*

We are also grateful to ***Miss Ferhi Wafaa****, our co-supervisor, for her contributions to this work. Her input and feedback have been instrumental in helping us to refine and improve our research.*

Finally, we would like to extend our thanks to the members of the jury for their interest in our research.

Abstract

An Intrusion Detection System (IDS) is a rapidly growing field that deals with detecting and responding to malicious network traffic and computer misuse. Artificial Intelligence (AI) plays a significant role in IDSs by providing an effective way to adapt and construct these systems. This thesis proposes an intelligent and efficient network intrusion detection system based on deep learning for attack detection and classification. The model is trained and tested using the realistic cyber defense dataset (CSE-CIC-IDS2018), which required several pre-processing tasks such as eliminating duplicate observations, clearing missing values, converting categorical data to numerical data, and performing feature scaling. Two approaches are proposed: the first maintains all attacks present in the dataset, along with the normal traffic. However, after analyzing the results, it was discovered that certain attacks were susceptible to misdetection. As a result, in the second approach, these misdetection-prone attacks were removed, which led to a significant improvement in accuracy, precision, recall, and F1-score. L2 regularization was implemented to avoid overfitting. The proposed deep learning model achieved impressive results, with an accuracy score of 99.97%, a precision score of 99.66%, a recall of 99.96%, and an F1-score of 99.81%. The findings demonstrate the effectiveness of deep learning in intrusion detection and emphasize the significance of meticulous data analysis and pre-processing.

Keywords: Intrusion Detection System (IDS), Artificial Intelligence (AI), deep learning, attack detection, classification, CSE-CIC-IDS20218 dataset, pre-processing, feature scaling, L2 regularization, evaluation metrics.

Résumé

Un système de détection d'intrusion (IDS) est un domaine en constante évolution qui se concentre sur la détection et la réponse au trafic réseau malveillant et à l'abus d'ordinateurs. L'intelligence artificielle (IA) joue un rôle important dans les IDS en offrant un moyen efficace d'adapter et de construire ces systèmes. Ce mémoire propose un système de détection d'intrusion réseau intelligent et efficace basé sur l'apprentissage en profondeur pour la détection et la classification des attaques. Le modèle est entraîné et testé à l'aide du jeu de données réaliste de défense cybernétique (CSE-CIC-IDS2018), qui a nécessité plusieurs tâches de prétraitement telles que l'élimination des observations en double, la suppression des valeurs manquantes, la conversion des données catégorielles en données numériques et la mise à l'échelle des caractéristiques. Deux approches sont proposées : la première maintient toutes les attaques présentes dans le jeu de données, ainsi que le trafic normal. Cependant, après analyse des résultats, il a été découvert que certaines attaques étaient susceptibles de ne pas être détectées. Par conséquent, dans la deuxième approche, ces attaques sujettes à une mauvaise détection ont été supprimées, ce qui a conduit à une amélioration significative de l'exactitude, de la précision, du rappel et du score F1. Une régularisation L2 a été mise en place pour éviter le surajustement. Le modèle d'apprentissage en profondeur proposé a obtenu des résultats impressionnants, avec un score de précision de 99,97%, un score de précision de 99,66%, un rappel de 99,96% et un score F1 de 99,81%. Les résultats démontrent l'efficacité de l'apprentissage en profondeur dans la détection d'intrusions et soulignent l'importance d'une analyse et d'un prétraitement de données minutieux.

Mots clés: Système de détection d'intrusion (IDS), Intelligence artificielle (IA), apprentissage en profondeur, détection d'attaques, classification,CSE-CIC-IDS20218 dataset, prétraitement, mise à l'échelle des caractéristiques, régularisation L2, mesures d'évaluation.

Table of contents

List of Figures

List of Tables

General introduction

The rapid evolution towards a fully connected lifestyle for all economic and social actors has led to significant advantages in terms of communication, online shopping, and information exchange. However, this interconnectivity has also exposed critical vulnerabilities and challenges for cybersecurity, with cyber threats evolving rapidly and attackers exploiting weaknesses in computer systems.

The scale of the problem is significant, with around 2328 cyber crimes occurring each day and an estimated loss of nearly $6 trillion each year [1]. To address these challenges, cybersecurity experts have developed Intrusion Detection Systems (IDSs) to monitor network traffic for any signs of abnormal behavior or misuse. IDSs have emerged as a valuable tool in the fight against cyber-attacks, providing early warning of potential threats.

IDSs can be broadly classified into two categories: network-based IDS (NIDS) and host-based IDS (HIDS). NIDS analyze network traffic to detect any attempt to subvert the normal behavior of the system, while HIDS detect intrusions by analyzing events on the local system where the IDS is installed.

In recent years, Deep Learning (DL) has emerged as a powerful tool in the field of intrusion detection. DL models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are capable of learning complex patterns from large amounts of data and identifying potential threats automatically, without the need for explicit rules or signatures. DL-based IDSs have shown promising results in detecting network attacks, including denial of service attacks, malware infections, and network intrusions. DL models can detect previously unknown or evolving attacks by learning from the historical network traffic data. This ability to detect unknown threats is particularly valuable in the context of emerging threats, where traditional signature-based methods may fail to detect new attack patterns.

Overall, while the benefits of a fully connected lifestyle are clear, it is important to be aware of the risks posed by cyber threats. By using IDSs and DL-based models, we can better protect ourselves and our networks against these evolving threats.

Our thesis is structured as an investigation of Deep Learning methods and their effectiveness in the field of intrusion detection. Specifically, we will examine approaches that have demonstrated their reliability. The outline of our thesis is as follows:

- Chapter one provides a comprehensive overview of network security and its different forms, including the various types of anomalies that can occur. In addition, we examine a variety of security measures and delve into the subject of intrusion detection systems. This includes an in-depth look at the different types of IDSs, detection methods, achitecture, and deployment scenarios.
- In the second chapter, we provided an introduction to machine learning, deep learning, and artificial intelligence. We then proceeded to discuss the fundamentals of machine learning, followed by a discussion on supervised learning. We compared and contrasted deep learning with machine learning, highlighting the advantages of deep learning. We then delved deeper into deep learning, presenting various neural network architectures and evaluation metrics.
- The third chapter is dedicated to constructing and assessing our model, with a focus on explaining the methodology employed and the various parameters considered.

Finally, we crown this manuscript with a general conclusion and perspectives.

Chapter 1

Network Security & Intrusion Detection System

1.1 Introduction

Network security is the process of developing and implementing a defensive strategy to safeguard the underlying networking infrastructure from unauthorized access, malicious activity, potential threats, or intrusion attempts. It is a critical aspect of any organization's operations, as it ensures the confidentiality, integrity, and availability of data and systems. Intrusion Detection Systems (IDS) are monitoring systems that listen to network traffic in a stealthy manner to detect potential security breaches. They are considered essential components of network security, as they help identify and respond to potential threats in real-time, mitigating the risk of security breaches. IDS can detect both known and unknown threats, making them an indispensable tool for protecting against emerging cyber-attacks.

1.2 Key Pillars of Network Security

The CIA triad, which stands for Confidentiality, Integrity, and Availability, is a fundamental concept in network security [2]. It serves as a set of guiding principles and goals for organizations and individuals to protect information from unauthorized access, modification, or loss. To achieve true network security, all three elements of the CIA triad must be present simultaneously. Therefore, the CIA triad is considered an essential foundation for effective information security practices (see Figure 1.1).

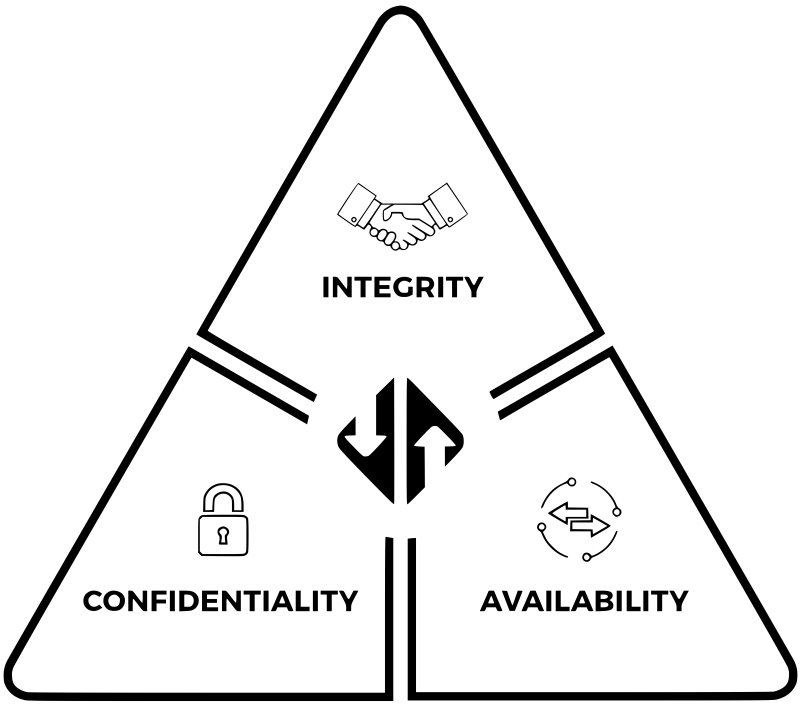


Figure 1.1: CIA triad.

1.2.1 Confidentiality

Confidentiality ensures that access to data is restricted to only authorized individuals or entities. The more sensitive the information, the more stringent the security measures should be. Some measures to ensure confidentiality include encryption, passwords, two-factor authentication, biometric identification, and security tokens. These measures help protect sensitive information from unauthorized access and prevent data breaches. It is important to implement the appropriate confidentiality measures based on the level of sensitivity of the information being protected.

1.2.2 Integrity

Integrity involves maintaining the accuracy and completeness of data. This is done by protecting data from unauthorized modification, accidental changes, or non-human-caused events such as server crashes. To maintain data integrity, various measures can be implemented, such as encryption, hashing, user access controls, checksums, version control, and backups. These measures help prevent unauthorized changes and ensure the accuracy of the data. It is important to implement the appropriate measures based on the sensitivity of the information being protected.

1.2.3 Availability

Availability ensures authorized users have access to data when needed. It involves implementing measures such as redundancy, load balancing, disaster recovery planning, regular maintenance, and monitoring to prevent any disruptions or downtime. By ensuring the availability of network services, organizations and individuals can prevent productivity loss, revenue loss, and other negative consequences that can arise from network downtime.

Apart from the CIA triad, there is another set of measures that should be put in place to ensure the security of information. These measures are known as authentication, authorization, and accounting.

1.3 Threats to Network Security

Networks face various types of security threats that can compromise the security and integrity of our data such as viruses, malware, and hacking. These threats can cause serious damage to the network, resulting in data loss, system downtime, and financial losses. We cite the most common network attacks below.

1.3.1 Denial-of-Service (DoS)

DoS is a type of cyber attack that seeks to disrupt the normal functioning of a computer or other device by overwhelming it with traffic, rendering it unavailable to its intended users [3]. Such attacks can be carried out using a single computer, and typically involve flooding the targeted machine with requests until normal traffic is unable to be processed. The main goal of a DoS attack is to exhaust the resources of the targeted device, leading to denial-of-service to legitimate requests. These attacks can cause significant disruption to businesses and organizations, and various measures can be taken to prevent and mitigate their impact. Figure 1.2 shows the DoS attack.

Figure 1.2: DoS attack.

1.3.1.1 Hulk

Hulk is a DoS attack tool that sends a large number of HTTP requests to a web server in order to overwhelm it and cause it to become unresponsive [4]. This attack is named after the Marvel Comics character, as the tool is designed to smash through web server defenses.

1.3.1.2 GoldenEye

GoldenEye is another DoS attack tool that sends a large number of HTTP requests to a web server, but it also uses encryption and obfuscation techniques to evade detection by security systems [5]. This attack is named after the fictional James Bond villain, as it is designed to be sophisticated and difficult to stop. GoldenEyeSlowloris Slowloris is a DoS attack tool that works by opening multiple connections to a web server and sending partial HTTP requests, but never completing them [6]. This ties up server resources and causes the server to become unresponsive to legitimate requests. This attack is named after the slow loris, a type of primate that moves very slowly, as the attack is designed to slowly drain resources from the targeted server.

1.3.2 Distributed denial-of-service (DDoS)

DDoS attack is a malicious attempt to disrupt the normal traffic of a targeted server, service, or network by overwhelming it with a flood of Internet traffic [7]. DDoS attacks are carried out with networks of infected computers and other devices, known as botnets, which are controlled remotely by the attacker. The attacker sends remote instructions to each bot to direct the attack, causing each bot to send requests to the target's IP address (see Figure 1.3). This can potentially cause the server or network to become overwhelmed, resulting in a denial-of-service to normal traffic. We list two of the most commonly used types of DDoS attacks.

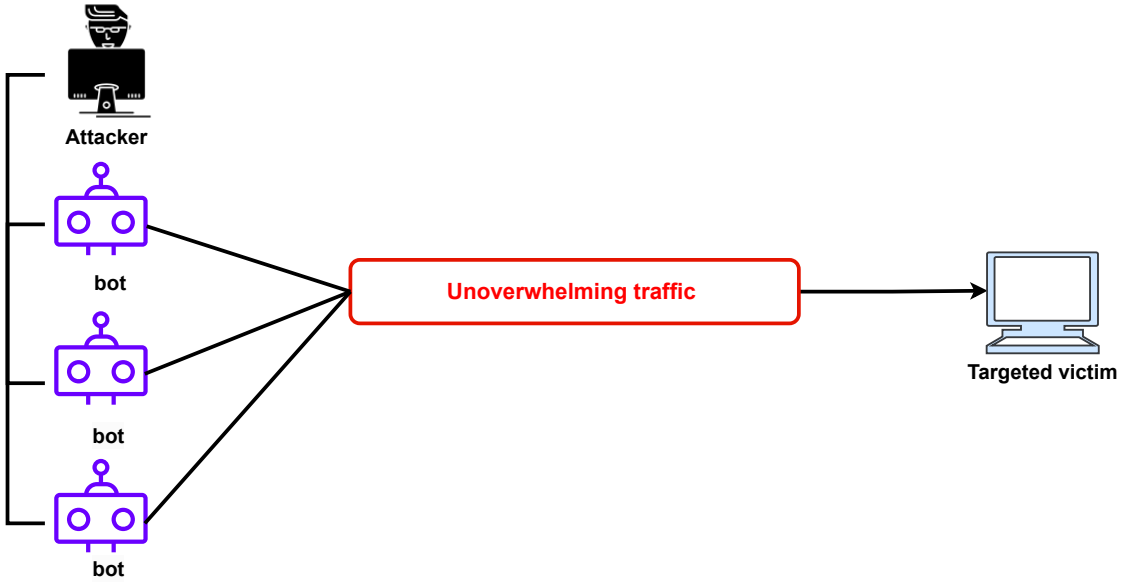


Figure 1.3: DDoS attack.

1.3.2.1 HOIC

HOIC stands for High Orbit Ion Cannon. It is a DDoS attack tool that uses a large number of HTTP GET or POST requests to overwhelm a target server or network [8]. HOIC is known for its ability to conduct highly coordinated and concentrated attacks, which can generate large amounts of traffic and make it difficult to detect the source of the attack.

1.3.2.2 LOIC

LOIC stands for Low Orbit Ion Cannon. It is also a DDoS attack tool that sends a flood of traffic to a target server or network [9]. Unlike HOIC, LOIC can use both HTTP and UDP flood attacks. HTTP flood attacks are similar to those used by HOIC, while UDP flood attacks send a large number of UDP packets to the target,

consuming network resources and causing the target to become unresponsive. The difference between these two attacks, is their level of sophistication. HOIC is known for its ability to conduct more advanced and coordinated attacks, while LOIC is considered to be a more basic and straightforward tool.

1.3.3 Brute force

A brute force attack is a trial-and-error method used to gain unauthorized access to a system or application by repeatedly guessing possible usernames and passwords until the correct combination is found [10]. The attacker may use automated software to generate a large number of consecutive guesses in a short amount of time (see Figure 1.4).

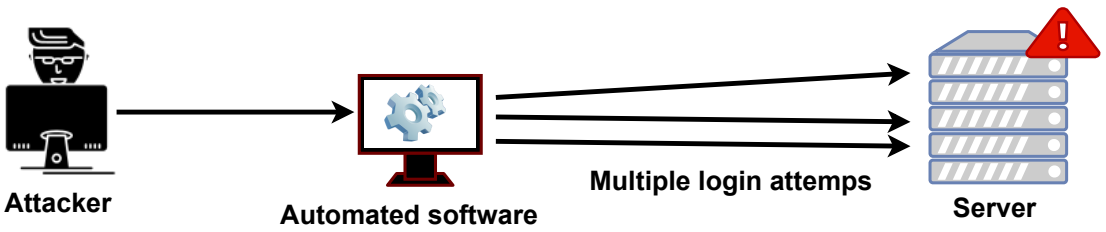


Figure 1.4: Brute force.

Four different types of brute force attacks are listed below:

1.3.3.1 FTP Brute Force

File Transfer Protocol is a network protocol used to transfer files. It uses a client-server model in which users can connect to a server using an FTP client [11]. Authentication takes place with a username and password, typically transmitted in plain text, but can also support anonymous logins if available.

1.3.3.2 SSH Brute Force

This type of attack targets the Secure Shell (SSH) protocol, which is used to remotely access and manage servers [12]. The attacker tries different username and password combinations until they are able to successfully log in to the SSH server. Once they have access, they can perform various malicious activities such as stealing data or installing malware.

1.3.3.3 Web Brute Force

This type of attack targets web-based login systems, such as those used for online banking or email accounts. The attacker uses automated tools to try different

combinations of usernames and passwords until they are able to log in to the system. Once they have access, they can steal sensitive information or perform other malicious activities [13].

1.3.3.4 XSS Brute Force

This type of attack targets web applications that have cross-site scripting (XSS) vulnerabilities [14]. The attacker injects malicious code into the web application, which is then executed when a user visits the infected page. The malicious code can steal the user's session ID, login credentials, or other sensitive information.

1.3.4 SQL injection

Structured Query Language (SQL) is a type of code injection attack that targets SQL databases by inserting malicious SQL statements into entry fields (see Figure 1.5). The attacker can use these statements to retrieve sensitive data from the database, modify existing data, or even destroy the entire database. SQL injection attacks commonly occur over the internet by sending malicious SQL queries to an API endpoint provided by a website or service [15].

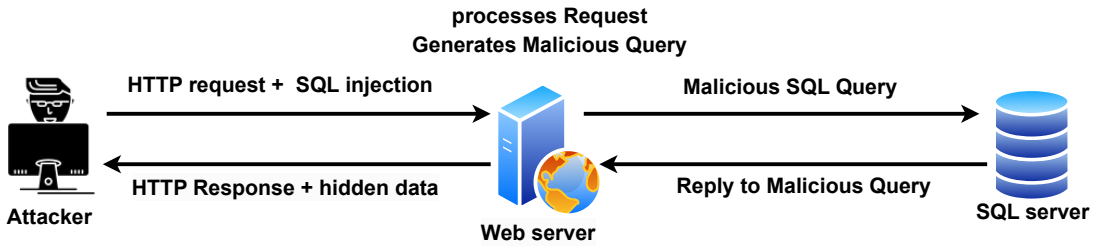


Figure 1.5: SQL Injection.

1.3.5 Infiltration

Infiltration is the act of gaining unauthorized access to a system or network, typically for malicious purposes. Attackers can use a variety of methods to infiltrate a network, including exploiting software or hardware vulnerabilities, using social engineering techniques to trick users into disclosing sensitive information or executing malicious code, or physically breaching security controls.

One form of infiltration involves exploiting vulnerabilities from within the network itself. In this scenario, attackers can use techniques such as sending a malicious file via email to a victim and exploiting an application vulnerability. Once the attack is successful, a backdoor is executed on the victim's computer, allowing the

attacker to scan the internal network for other vulnerable systems and exploit them if possible [16].

1.3.6 Botnet

Botnet is a network of compromised devices, which are controlled by a single entity or attacker, and used to carry out malicious activities. The compromised devices are infected with malware that allows the botmaster to remotely control them and use them for various nefarious purposes, such as DDoS attacks, spamming, phishing, credential stuffing, and more.

One commonly used botnet malware is Zeus, which is a Trojan horse that runs on Microsoft Windows operating systems. It can be used for various criminal activities, including stealing banking information and installing ransomware. Zeus is often spread through drive-by downloads and phishing schemes. Another botnet commonly used is the Ares botnet, which is an open-source botnet that allows the botmaster to remotely execute shell commands, perform persistence, upload/download files, take screenshots, and log keystrokes [17].

In summary, security threats can cause significant harm to a network, resulting in data breaches, downtime, lost productivity, and financial losses. Protecting against these threats requires a comprehensive approach to network security that includes regular security updates to software and hardware, implementing strong passwords, and user education and awareness training to prevent social engineering tactics. Firewalls, antivirus software, intrusion detection systems, and other security measures should also be put in place to prevent attacks and mitigate their impact.

1.4 Network Security Measures

Securing a network can be achieved through several methods, the most common of which are covered below.

1.4.1 Firewalls

Firewalls are an essential tool for securing networks against unauthorized access and malware attacks [18]. They monitor incoming and outgoing traffic and can block traffic from sources that don't meet certain criteria, such as IP address or port number. Although firewalls are easy to deploy and manage, they have limitations in detecting and blocking advanced attacks and internal threats. Figure 1.6 shows firewall.

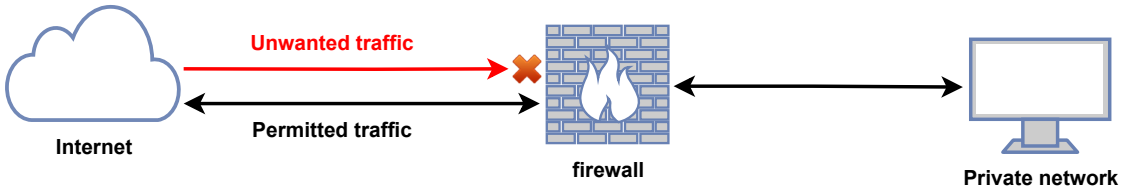


Figure 1.6: Firewall.

There are several types of firewalls [19] commonly used in computer networks, including:

1.4.1.1 Packet filtering firewall

This firewall examines packets at the network layer and filters incoming and outgoing traffic based on pre-defined rules, such as source and destination addresses, port numbers, and protocol type. They are easy to implement and have minimal impact on network performance, making them a popular choice for basic network security. However, they are less secure than other types of firewalls and can be vulnerable to certain types of attacks, such as IP spoofing and Denial-of-Service (DoS) attacks. Packet filtering firewalls provide a simple and cost-effective way to provide basic network security, but they may not provide adequate protection against modern threats and should be used in conjunction with other security measures.

1.4.1.2 Stateful inspection firewall

This firewall keeps track of the state of network connections and uses this information to allow or block traffic based on factors such as state, port, and protocol. The firewall monitors all activity from the opening of a connection until it is closed, and filtering decisions are made based on both administrator-defined rules and context, which refers to using information from previous connections and packets belonging to the same connection. In addition, stateful inspection firewalls can detect and block certain types of attacks, such as SYN floods. By maintaining information about the state of network connections, the firewall can identify and prevent these types of attacks by only allowing packets that belong to a known connection.

1.4.1.3 Next-generation firewall (NGFW)

NGFW is an advanced type of firewall that combines traditional firewall capabilities with additional features such as Intrusion Prevention System (IPS), application visibility and control, SSL inspection, and advanced malware protection. NGFWs use deep packet inspection to identify and block traffic based on the application, user, content, and behavior, providing better security for modern, complex network

environments. They can also detect and block sophisticated attacks, such as zero-day exploits and advanced persistent threats, by analyzing traffic in real-time and using threat intelligence feeds. NGFWs offer a higher level of security and granular control over network traffic compared to traditional firewalls, making them a popular choice for securing network infrastructures against modern threats.

1.4.2 Virtual Private Networks (VPNs)

VPNs provide secure remote access to a network by creating an encrypted tunnel over the internet (see Figure 1.7). They are commonly used by remote workers who need to access company resources from outside the office.

Figure 1.7: VPN.

There are several types of VPNs [20], including:

1.4.2.1 Remote access

A remote access VPN securely connects a device outside the corporate office to a private network over the internet. This type of VPN is commonly used by employees who need to access company resources from outside the office, such as from home or while traveling. The VPN technology has advanced to allow security checks to be conducted on endpoints to ensure they meet certain security requirements before being granted access to the private network. Remote access VPNs provide a secure way for employees to access company resources while also ensuring the confidentiality, integrity, and availability of sensitive data on the private network.

1.4.2.2 Site-to-site

A site-to-site VPN is a type of VPN that connects the corporate office to branch offices over the Internet, creating a secure connection between two or more networks. This type of VPN is commonly used for branch office connectivity, cloud migration, and disaster recovery. Dedicated equipment is used to establish and maintain the connection, making it impractical to have direct network connections between these offices. Site-to-site VPNs are used when distance makes it difficult to have direct network connections between the offices. It is often referred to as network to network access. Site-to-site VPNs provide a secure way for organizations to connect

multiple networks, including two corporate networks or a corporate network and a cloud service provider, while ensuring the confidentiality, integrity, and availability of sensitive data.

1.4.3 Encryption

Encryption is an effective technique to protect the confidentiality and integrity of data by converting plain text into a coded form, making it difficult for unauthorized parties to access the information. This security method can be used in various scenarios such as secure email communication and storage of sensitive data. However, implementing encryption can have an impact on the performance of systems and may require advanced management skills due to the complexity involved. Nonetheless, the benefits of encryption make it a powerful tool in securing sensitive information. There are several types of encryption, including:

1.4.3.1 Symmetric and asymmetric encryption

Symmetric encryption, also known as shared-secret encryption, uses a single key for both encryption and decryption of data. This makes it fast and efficient, making it ideal for large amounts of data. However, the security of encrypted data relies on keeping the key secret. Asymmetric encryption, also known as public-key encryption, uses two different keys: one for encryption and one for decryption. The public key is widely distributed, while the private key is kept secret. While asymmetric encryption is slower than symmetric encryption, it provides better security because the private key is kept secret, making it difficult for attackers to access the encrypted data [21].

1.4.3.2 Hashing

Hashing is a one-way encryption process that takes plain text and produces a fixed-length string of characters, called a hash. The same plain text will always produce the same hash, but it is practically impossible to reverse the process and recover the plain text from the hash, there are some common methods of hashing such as MD5, SHA, bcrypt and scrypt [22].

1.4.4 Access Control

Access control is a network design strategy that allows only compliant, authenticated, and trusted endpoint devices to access network resources and infrastructure, while denying unauthorized access and potential threats. This is achieved through

the deployment of a password, unique user ID, and authentication process to access the network. By restricting access to data and resources to only authorized individuals, access control can prevent unauthorized access and data breaches. It is commonly used to enforce password policies and control user permissions. However, implementing access control can be complex and may lead to a false sense of security if not implemented properly.

1.4.5 Security Information and Event Management (SIEM) Systems

SIEM is a type of software that provides real-time analysis of security alerts generated by network hardware and applications. SIEM systems combine Security Information Management (SIM) and Security Event Management (SEM) technologies to provide a comprehensive view of an organization's security posture [23].

SIEM systems collect security data from various sources, such as firewalls, intrusion detection systems, servers, and other network devices. They then use sophisticated analytics and machine learning algorithms to identify patterns and anomalies that could indicate a security breach or threat. Once a potential security issue is detected, the SIEM system generates an alert that is sent to security analysts or administrators for further investigation. SIEM systems can also automate security response actions, such as blocking an IP address or shutting down a compromised system.

Some of the key features of SIEM systems include:

- **Log collection and management:** Collecting and storing logs from various sources in a central location for analysis and retention purposes.
- **Real-time event correlation:** Correlating security events from multiple sources to provide a comprehensive view of security threats.
- **User activity monitoring:** Monitoring user activity to detect unauthorized access or suspicious behavior.
- **Threat intelligence:** Integrating with external threat intelligence feeds to enhance threat detection and response capabilities.
- **Reporting and compliance:** Generating reports for compliance purposes and to provide visibility into security posture.

Some of the popular SIEM systems in use today include IBM QRadar, Splunk Enterprise Security, LogRhythm, and McAfee Enterprise Security Manager.

1.4.6 Intrusion Detection and Prevention Systems (IDPS)

IDPS are security technologies designed to detect and prevent unauthorized access, misuse, and other security threats to computer systems and networks. IDPS systems operate by monitoring network and system events, analyzing them for signs of malicious activity, and taking action to prevent or mitigate threats.

IDPS systems can be classified into two main categories: network-based IDPS and host-based IDPS. Network-based IDPS systems monitor network traffic for suspicious activity, such as known attack patterns or abnormal behavior, and can take action to block or prevent such activity. Host-based IDPS systems, on the other hand, monitor activity on individual computer systems or hosts, and can detect and prevent malicious activity that may not be visible on the network. Some of the key features of IDPS systems include:

1. Real-time monitoring of network and system events.
2. Analysis of network traffic for signs of malicious activity.
3. Automatic response and prevention mechanisms to stop threats in real-time.
4. Alerting and reporting capabilities to notify security teams of potential threats and breaches.

In conclusion, there are several methods available to secure a network, each with its own strengths and weaknesses. However, the most important method for securing a network is an IDPS, which can detect and respond to threats in real-time. By using a combination of these methods and following best practices such as regularly updating software and using strong passwords, businesses and individuals can significantly reduce their risk of a security breach.

1.5 Intrusion Detection System

An Intrusion Detection System (IDS) is a critical component of any comprehensive network security strategy. Its main purpose is to detect and alert security administrators of potential security breaches or unauthorized access attempts to a network or system. IDS operates by monitoring network traffic, system logs, and other sources for suspicious activities, and it can identify a range of security threats, such as malware infections, attempts to exploit vulnerabilities, and other unauthorized activities [24].

IDS plays a critical role in network security because it allows organizations to detect and respond to security threats before they can cause significant damage.

Without IDS, attackers can potentially gain access to sensitive data, cause system disruptions, and damage an organization's reputation. With IDS in place, security administrators can quickly identify and respond to security incidents, minimize the impact of an attack, and prevent future attacks. Figure 1.8 shows Intrusion Detection System.

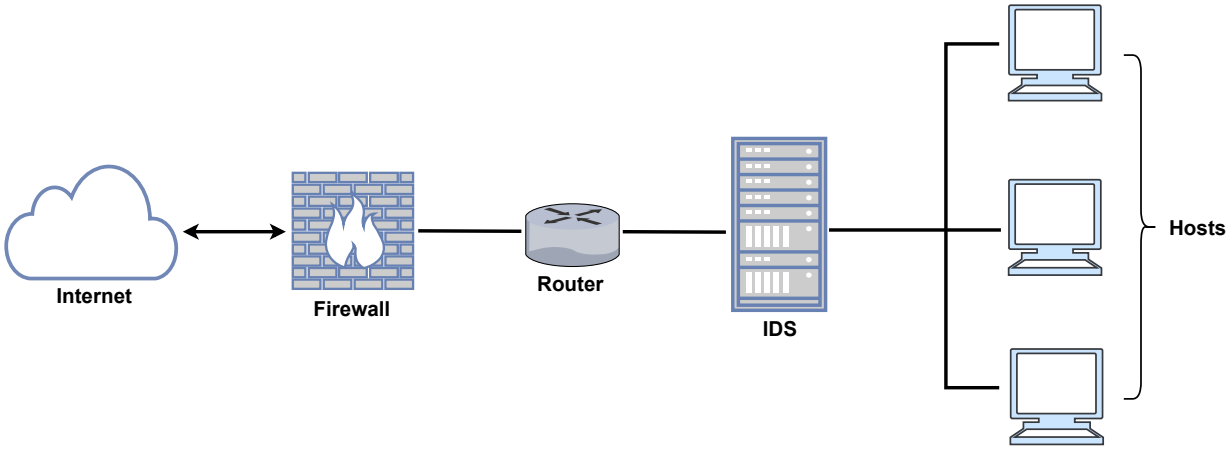


Figure 1.8: Intrusion Detection System.

1.6 Types of Intrusion Detection System

There are different types of IDS systems available, each with its unique strengths and capabilities. The three primary types of IDS are listed below.

1.6.1 Network-based IDS (NIDS)

Network-based Intrusion Detection Systems (NIDS) are a type of IDS that are strategically placed within networks to passively monitor network traffic [25]. They can be either hardware or software-based and can connect to various network mediums, such as Ethernet or FDDI. Typically, NIDS have two network interfaces: one for listening to network traffic in promiscuous mode and another for control and reporting. This allows the NIDS to continuously analyze network traffic and identify potential security threats, such as malware infections or unauthorized access attempts. NIDS can also block attacks and generate reports to help security administrators respond to security incidents. Figure 1.9 shows Network-based IDS.

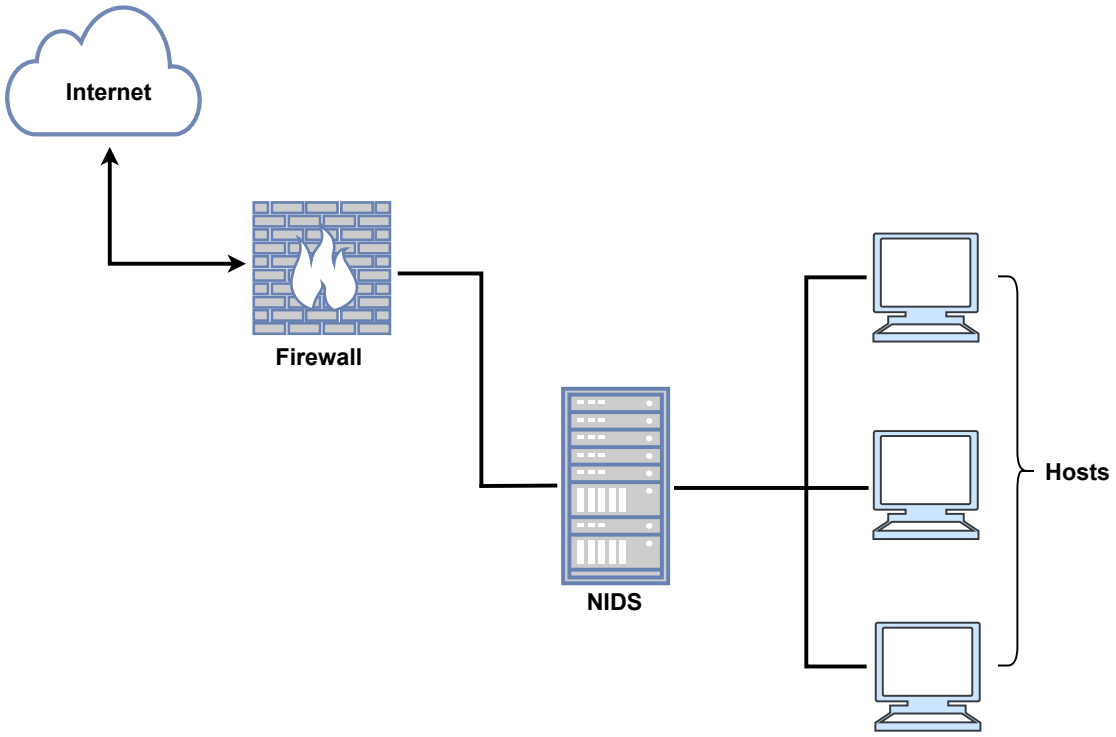


Figure 1.9: Network-based IDS.

1.6.2 Host-based IDS (HIDS)

Host-based Intrusion Detection System (HIDS) is an application that runs on a single machine and monitors a computer or network for suspicious activities, including intrusions created by external actors and misuse of resources or data internally [26]. HIDS software logs suspicious activity and reports it to administrators, allowing them to quickly identify any anomalies and signs of intrusion that may have occurred. HIDS tools monitor log files generated by applications and create a historical record of activities and functions. Most HIDS systems use a combination of signature-based and anomaly-based detection methods, relying on a database of known cyber threats and machine learning techniques to flag malicious behavior. The key function that makes HIDS a must-have is the detection feature, which saves administrators from having to sort through log files for unusual behavior. HIDS software uses rules and policies to search log files and flag those with events or activity that could be indicative of potentially malicious behavior. Figure 1.10 shows Host-based IDS.

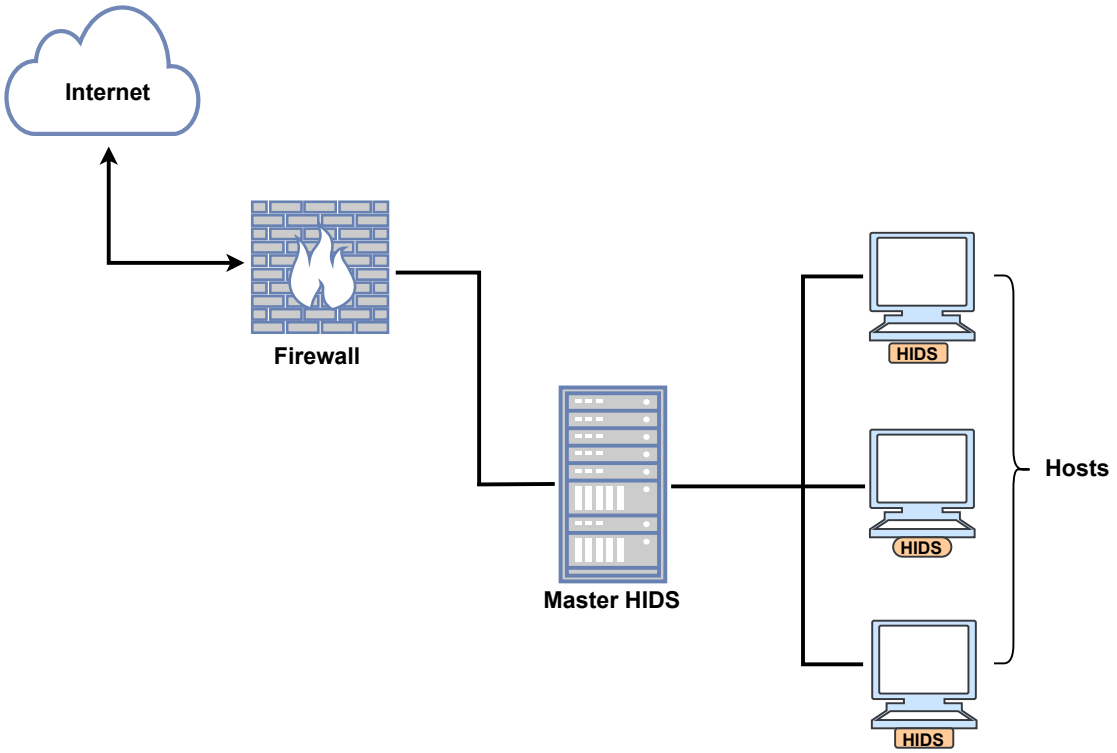


Figure 1.10: Host-based IDS.

HIDS focuses on endpoint behaviors, while NIDS monitors network traffic for abnormalities. NIDS can detect attacks before they happen, and HIDS responds after a breach. Both systems are needed for complete security, and SIEM combines them for real-time analysis. Choosing the best IDS depends on an organization's specific security needs.

1.6.3 Hybrid IDS (HIDS + NIDS)

Hybrid IDS refers to the integration of both Host-based Intrusion Detection System (HIDS) and Network-based Intrusion Detection System (NIDS) to create a more robust and comprehensive security solution [27].

Hybrid IDS can be particularly useful in environments where there are multiple entry points for attackers or where traditional perimeter defenses may not be enough to secure the network. It is also helpful for detecting advanced persistent threats (APTs) that can evade traditional security measures. Some popular Hybrid IDS solutions include Snort, OSSEC, and Suricata. These tools provide a combination of signature-based and anomaly-based detection methods, relying on a database of known cyber threats and machine learning techniques to flag malicious behavior. Figure 1.11 shows Hybrid IDS.

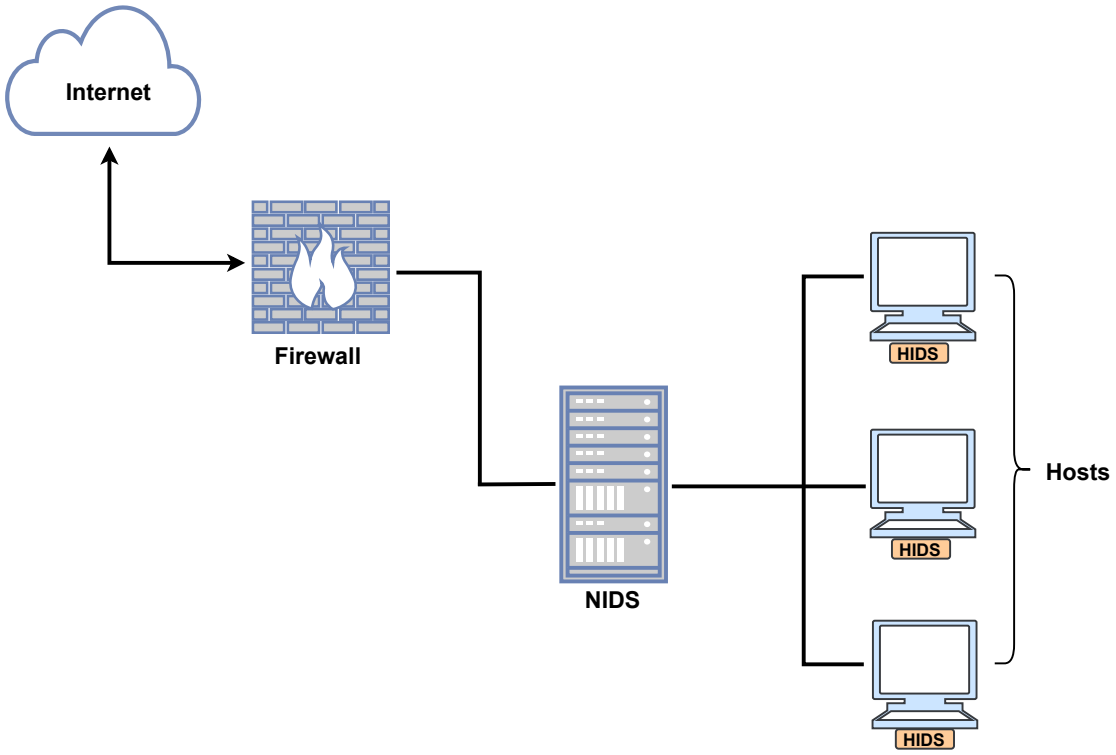


Figure 1.11: Hybrid IDS.

1.7 IDS Detection Methods

Intrusion detection systems primarily employ two techniques for detecting threats and alerting network administrators: signature-based and anomaly-based [28].

1.7.1 Signature-based detection

Signature-based detection is a common and widely used method of intrusion detection. It operates by comparing incoming network traffic with a database of known attack patterns, or signatures, to identify malicious behavior. This method is particularly effective at detecting well-known and frequent attacks, such as malware, phishing, and denial-of-service. It is also relatively easy to deploy and maintain, as signature databases can be regularly updated by vendors or security experts.

However, signature-based detection has some limitations. It cannot detect new or unknown attacks, or variations of known attacks that do not match any signature in the database. False positives can also be a problem, with legitimate traffic sometimes being identified as malicious. Additionally, signature-based detection can be resource-intensive, as it has to scan each data packet against a large number of signatures, which can slow down network performance.

1.7.2 Anomaly-based detection

Anomaly-based detection is a modern and sophisticated approach to IDPS. It involves creating a standard for normal network behavior through statistical analysis, machine learning, or artificial intelligence. This standard is then used to monitor network traffic for any unusual or anomalous activity that could signify a potential attack. If an anomaly is detected, the IDPS alerts or blocks the attack. Anomaly-based detection is particularly effective against new or unknown attacks or variations of existing attacks that have no signature. Additionally, it is adaptable and dynamic, meaning that it can learn from network behavior and modify the standard as necessary.

However, there are some drawbacks to anomaly-based detection. For example, establishing and maintaining a reliable baseline for normal behavior can be challenging, particularly in complex and diverse networks. Additionally, false negatives, or missed attacks, can occur when malicious traffic is disguised as legitimate traffic or mimics normal behavior. Finally, implementing and operating anomaly-based detection can be expensive and complex, requiring advanced technology and expertise.

1.8 IDS Architecture

The architecture of an IDS can vary depending on the specific implementation and requirements of the system [29], but the main three components of an IDS architecture are:

1.8.1 Data collection

These sensors are responsible for gathering data from various sources, such as network traffic, system logs, and other sources, that can be used to identify potential security threats. IDS typically use sensors to collect data from these sources, such as Network Intrusion Detection Systems (NIDS), Host-based Intrusion Detection Systems (HIDS), and Application-based Intrusion Detection Systems (AIDS). Some common sensor technologies used for IDS include Snort, Suricata, Zeek, and OSSEC.

1.8.2 Data pre-processor

data pre-processing involves filtering, normalizing, and extracting features from collected data to create activity records that are relevant for security analysis. The pre-processed data may include information such as source and destination IP addresses, port numbers, protocol types, timestamps, and packet payloads. The goal of data pre-processing is to reduce the volume of data that needs to be analyzed,

improve the quality of the remaining data, and make it more amenable to further analysis.

1.8.3 Intrusion recognition

Intrusion recognition is the process of identifying and classifying security threats detected by an IDS, using various techniques such as signature-based detection, anomaly-based detection, or Machine Learning-based detection to identify and classify security threats.

The goal of IDS architecture is to quickly and accurately detect security threats and minimize their impact on the system or network. The interplay between the data collection, data pre-processing, and intrusion recognition components of IDS architecture is visually represented in Figure 1.12.

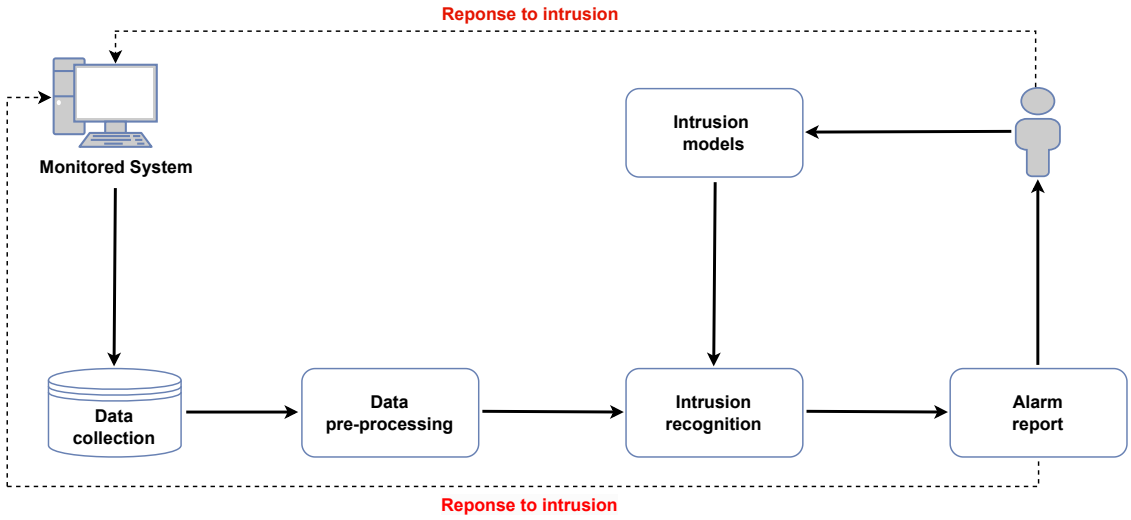


Figure 1.12: Framework of Intrusion Detection System.

1.9 IDS Deployment Scenarios

The deployment scenarios of IDS [30] refer to the various ways in which intrusion detection systems are positioned and configured within a network. These scenarios can be broadly categorized into three types:

1.9.1 Perimeter-based IDS

Perimeter-based IDS is deployed at the edge of the network, typically between the internal network and the internet. Its primary function is to monitor and filter

incoming traffic from the outside and detect external attacks, such as port scanning, vulnerability probing, or malware infections. Perimeter-based IDS is usually implemented as a network-based IDS (NIDS), using sensors or probes that capture and analyze network traffic. It is a common first line of defense against external threats and can complement other security measures, such as firewalls and Intrusion Prevention Systems (IPS).

1.9.2 Internal-based IDS

Internal-based IDS is deployed inside the internal network, typically at critical points or segments where sensitive data or systems are located. Its primary function is to detect and respond to internal threats, such as insider attacks, lateral movement, or unauthorized access. Internal-based IDS can be implemented as a host-based IDS (HIDS), using agents or software that monitor and analyze the behavior of individual hosts or endpoints, or as a NIDS, using sensors or probes that monitor and analyze network traffic. Internal-based IDS can be a valuable complement to access controls, identity management, and Data Loss Prevention (DLP) systems.

1.9.3 Distributed IDS

Distributed IDS is a hybrid deployment scenario that combines both perimeter-based and internal-based IDS. It involves deploying multiple sensors or probes at strategic points throughout the network, both at the edge and inside, to provide comprehensive coverage and visibility. Distributed IDS can be a flexible and scalable solution that adapts to different network architectures, topologies, and security requirements. It can also provide redundancy, load balancing, and failover capabilities, in case of sensor or probe failures or network outages.

Selecting the appropriate IDS deployment scenario is an essential step as each scenario has its own set of advantages and challenges, that depend on the network environment, threat landscape, and security objectives. Therefore, it is important to carefully evaluate and select the appropriate IDS deployment scenario that meets the unique requirements and limitations of the organization.

1.10 Conclusion

In conclusion, network security is a critical aspect of any organization's IT infrastructure, and the threat landscape is constantly evolving. As outlined in this chapter, there are various threats to network security, including DoS, DDoS, brute force attacks, SQL injection, infiltration, and botnets. To protect against these threats, there are several security measures that organizations can implement, such as firewalls, VPNs, encryption, access control, SIEM systems, and IDPS. Intrusion detection systems are an essential component of network security, and there are different types of IDS, including NIDS, HIDS, and hybrid IDS. The choice of IDS deployment scenario, whether perimeter-based, internal-based, or distributed, depends on the specific needs and constraints of the organization. Overall, organizations must stay vigilant and proactive in their approach to network security to mitigate risks and prevent potential cyber attacks. In the following chapter, we will introduce the concept of both Deep Learning and machine learning.

Chapter 2

Deep Learning

2.1 Introduction

The primary objective of artificial intelligence (AI) is to equip computers with the capability to comprehend and interact with the world intelligently. Deep Learning (DL) has emerged as a highly promising method to achieve this goal. Deep Learning utilizes deep hierarchical neural networks to learn high-level concepts from data and represent them in a significant manner. It is a type of Machine Learning, which is why having a fundamental understanding of Machine Learning is essential before diving into Deep Learning. Several concepts used in neural networks have their roots in Machine Learning, and revising them can help in comprehending the techniques of Deep Learning.

Machine Learning and Deep Learning are used in a wide range of industries and domains. Some of the common applications include healthcare, finance, e-commerce, transportation, natural language processing, image and video recognition. In healthcare, Machine Learning and Deep Learning can assist with medical diagnosis and personalized medicine. In finance, they can be used for fraud detection and portfolio optimization. In e-commerce, they can personalize product recommendations and improve customer engagement. In transportation, they can optimize traffic prediction and autonomous driving. In natural language processing, they can develop chatbots and virtual assistants. In image and video recognition, they can detect objects and recognize faces. These are just a few examples of the many applications of Machine Learning and Deep Learning.

Deep Learning is a type of Machine Learning, which in turn is a branch of Artificial Intelligence (as shown in Figure 2.1).

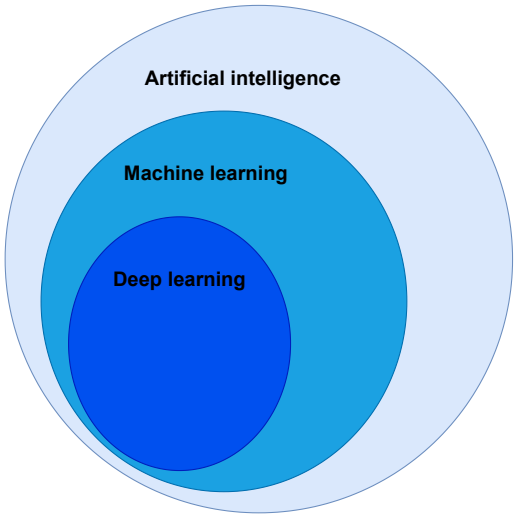


Figure 2.1: Representation of the relationships between AI, ML, and DL.

2.2 Fundamentals of Machine Learning

Machine Learning refers to the ability of computers to automatically learn patterns and relationships in data, without being explicitly programmed to do so. It involves the use of statistical models and techniques that enable the computer to learn patterns from data, and then apply that learning to make predictions or take actions without being explicitly programmed to do so, by using large amounts of data to learn and adapt. Machine Learning is a diverse and rapidly evolving field that employs a variety of different methods to achieve its goals. Among the most common methods are reinforcement learning, supervised learning, and unsupervised learning [31].

- **Reinforcement learning:** The computer learns through trial and error by interacting with an environment and receiving feedback in the form of rewards or penalties. The goal of reinforcement learning is to find an optimal policy, or set of actions, that maximizes the gradual reward over time.
- **Unsupervised learning:** The computer is given an unlabeled dataset, meaning that there are no predefined output or target variables. The goal of unsupervised learning is to discover patterns and relationships in the data, such as clustering similar examples or identifying outliers.
- **Supervised learning :** The computer is trained on a labeled dataset, meaning that each example in the dataset is accompanied by the correct output or target variable. This approach is used when the goal is to learn a mapping from inputs to outputs, so that the computer can make accurate predictions on new, unseen data.

Out of the three methods, supervised learning is often considered the most interesting because it closely resembles the way humans learn.

2.3 Supervised learning

Supervised learning is a powerful tool for solving a wide range of problems, including image recognition, natural language processing, and recommendation systems. One of the advantages of supervised learning is that it allows for the creation of complex models that can capture non-linear relationships between the inputs and outputs (see Figure 2.2).

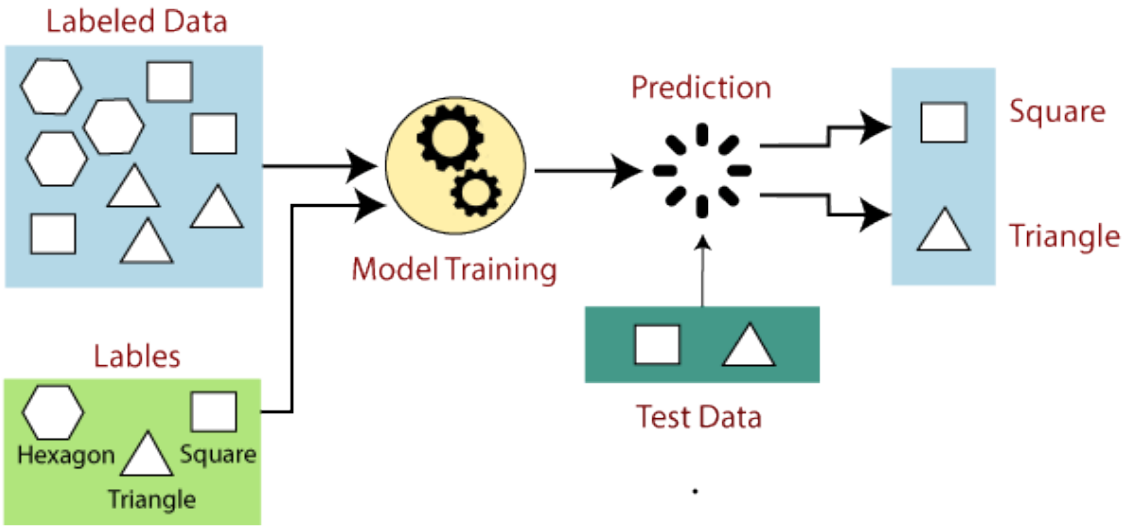


Figure 2.2: Supervised Machine Learning.

To master supervised learning, it is essential to understand and know the following four concepts:

- Dataset.
- Model and its parameters.
- Cost function.
- Learning algorithm.

2.3.1 Dataset

Dataset is a collection of input/output pairs that is used to train and evaluate a supervised learning model. The input data represents the features or attributes of the problem, while the output data represents the target or label that the model is trying to predict.

2.3.2 Model and its parameters

The Model is the mathematical representation of the relationship between the input data and the output data. It is defined by a set of parameters that are adjusted during the training process to minimize the difference between the predicted outputs and the true outputs. The choice of the model and its associated parameters can have a significant impact on the performance of the model.

2.3.3 Cost function

The cost function, also known as the loss function, is a mathematical function that measures the difference between the predicted outputs and the true outputs for a given set of model parameters (see Figure 2.3). The goal of learning algorithm is to find the values of the model parameters that minimize the cost function [32].

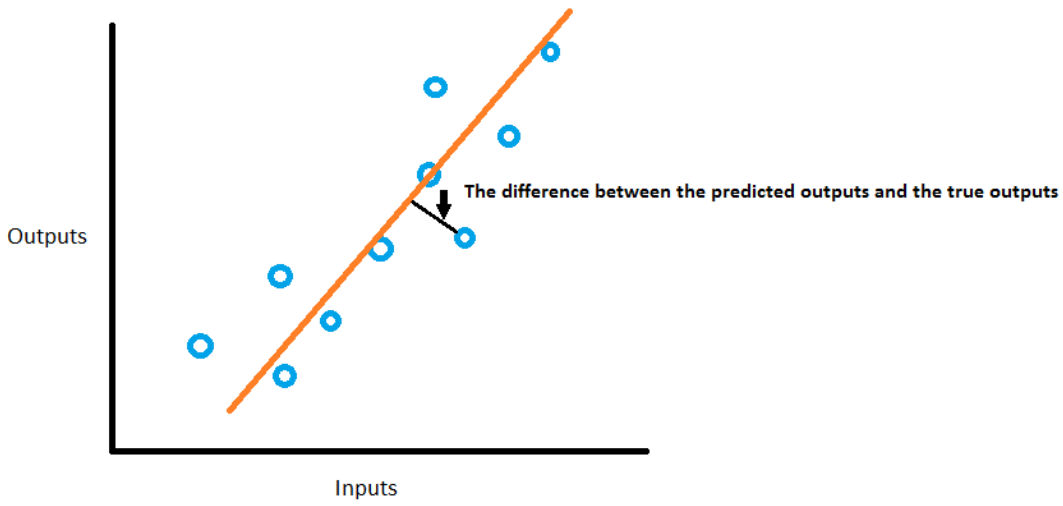


Figure 2.3: Representation of the cost function.

2.3.4 Learning algorithm

A learning algorithm is a set of rules and procedures that are used to update the model parameters based on the difference between the predicted and true outputs. The choice of learning algorithm can have a significant impact on the performance of the model, and there are many different algorithms available for different types of problems.

2.4 Advantages of Deep Learning over Traditional Machine Learning Algorithms

Deep Learning algorithms automatically learn to extract high-level features from raw data by using multiple layers of artificial neural networks. This allows Deep Learning models to learn from vast amounts of data and perform tasks such as image recognition and natural language processing with high accuracy.

On the other hand, traditional Machine Learning algorithms rely on handcrafted features and statistical methods to analyze data and make predictions. They typically require a human expert to engineer the relevant features that are fed into the model. Machine Learning algorithms can perform tasks such as classification and regression, but they may not be as effective as Deep Learning algorithms in handling complex and unstructured data (see Figure 2.4) [33].

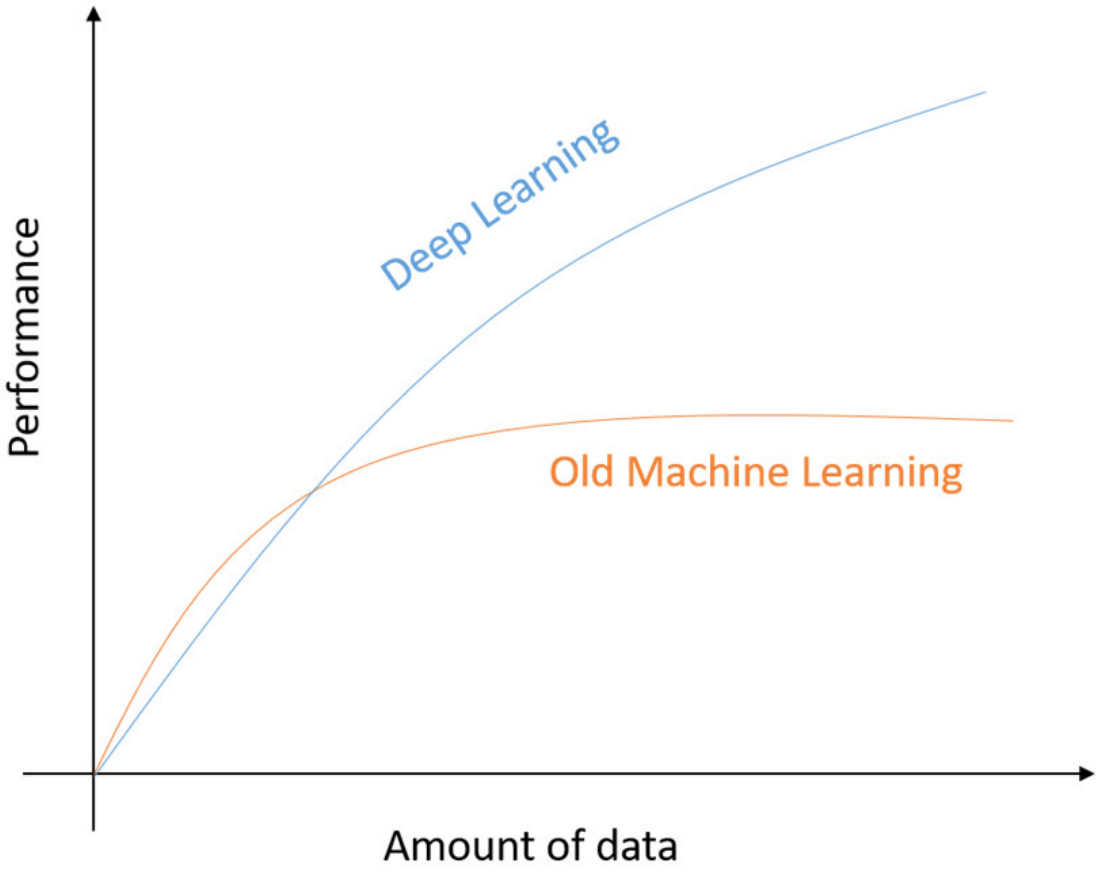


Figure 2.4: The Impact of Data Availability on Algorithm Performance.

2.5 Introduction to Deep Learning

Deep learning is an advanced branch of machine learning that uses neural networks with multiple layers to facilitate the learning process. It can handle large-scale data, and its models are designed to automatically learn hierarchical representations of data from raw input such as images, text, or sound, without the need for explicit feature engineering. Deep learning is based on artificial neural networks, which are modeled after the structure and function of the human brain [34].

2.6 Artificial Neural Networks

Artificial neural networks simulate the mechanism of learning in biological organisms. The human nervous system contains cells, which are referred to as neurons. The neurons are connected to one another with the use of axons and dendrites, and the connecting regions between axons and dendrites are referred to as synapses. These connections are illustrated in Figure 2.5.

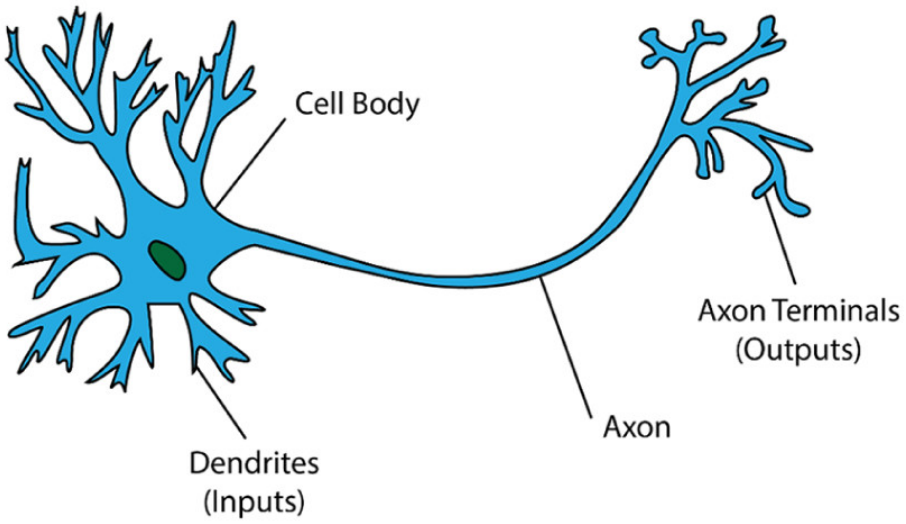


Figure 2.5: Biological neuron structure.

Dendrites receive input signals from other neurons via synapses. These signals are then integrated in the cell body, and if the sum of the signals exceeds a certain threshold, the neuron fires an action potential down its axon. This electrical signal travels down the axon to the endings, where it is transmitted to other neurons in the nervous system.

2.6.1 Components of Artificial Neural Networks

Artificial Neural Networks (ANNs) consist of various components that work together to process information and make decisions. These components include neurons (also known as perceptrons), weights, activation functions, and layers. Neurons are the basic processing units of ANNs, while weights represent the strength of the connections between neurons. Activation functions determine whether a neuron should "fire" based on its inputs, and layers can be used to perform intermediate calculations. By understanding these components and how they interact, we can better understand how ANNs operate and how they can be used to solve complex problems.

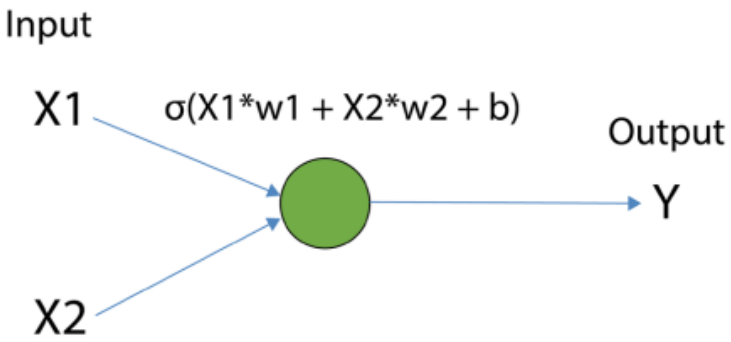


Figure 2.6: A Neural Networks representation with two-input.

In deep learning models, the parameters are referred to as weights (w) and biases (b). These parameters are used to compute the sum of weighted inputs by multiplying the inputs with respective weights and adding a bias term to the result, this can be seen in the calculation above the node in the preceding image. The inputs are X1 and X2, the weights are W1 and W2, and the bias is b. The sum of the weighted inputs and bias is then fed into a nonlinear function, known as the activation function (σ) as shown in Figure 2.6. The output of the neuron is referred to as the activation of that neuron, which is computed using the activation function and bias [33].

2.6.2 Feedforward Neural Networks (FFNNs)

Feedforward neural network (FFNN) is a type of artificial neural network that processes information in a unidirectional manner, from input to output. It consists of an input layer, one or more hidden layers, and an output layer. During training, weights and biases are adjusted to minimize the difference between the predicted output and the actual output.

2.6.2.1 FFNNs architecture

To build a single-layer neural network, neurons are stacked on top of each other in a layer as shown in Figure 2.7. In this architecture, every input value at the input layer is passed to all neurons in the hidden layer. The computation of the sum of weighted inputs and bias, as well as the activation function, is applied independently for each neuron in the hidden layer.

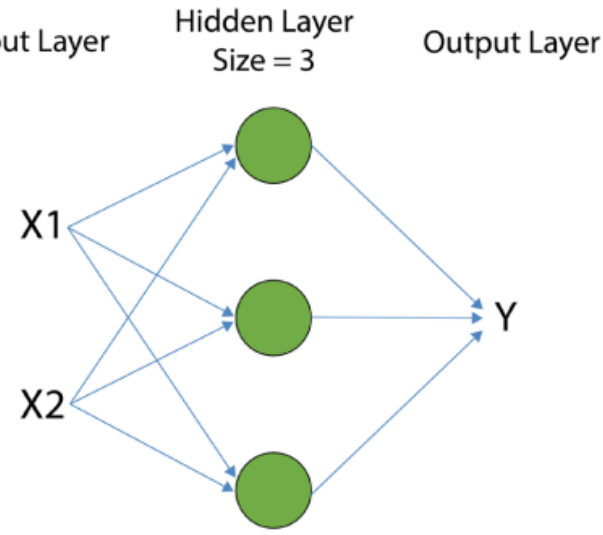


Figure 2.7: Representation of a 2D input neural network with one hidden layer.

Multi-layer neural networks can also be built by stacking multiple layers of processing nodes in a row. Figure 2.8 shows a two-layer neural network with a two-dimensional input.

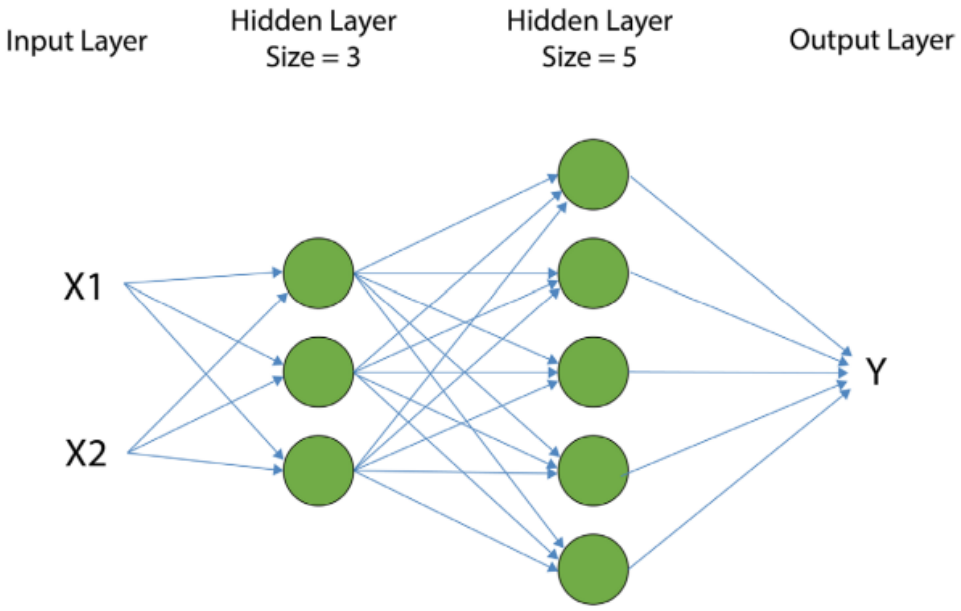


Figure 2.8: Representation of a 2D input with a two hidden layers neural network.

Figures 2.7 and 2.8 show the most common representations of neural networks. Every neural network consists of an input layer, an output layer, and one or more hidden layers. When there is only one hidden layer, the network is called a shallow neural network. On the other hand, a neural network with many hidden layers is called a deep neural network.

The input layers are generally on the left. In the case of Figure 2.8, these are the features X1 and X2, which are input to the first hidden layer with three neurons. Arrows indicate weight values applied to the input. In the second hidden layer, the result of the first hidden layer becomes the input of the second hidden layer. The arrow between the first hidden layer and the second hidden layer represents the weight, which level the output is generally in on the far right, represented by the plane labeled Y in the case of Figure 2.8.

In deep learning the number of neurons in the input layer is equal to the number of features of the input data, and the number of neurons in the output layer is equal to the dimensions of the output data. However, you need to select the number of neurons in the hidden layers or the size of the hidden layers. If you choose a larger size layer, the model becomes more flexible and will be able to model more complex functions. This increase in flexibility comes at the cost of the need for more training data and more computations to train the model on.

The parameters that are required to be selected by the developer are called hyperparameters and include parameters such as the number of layers and the number of neurons in each layer. Common hyperparameters to be chosen include the number of epochs to train for and the loss function to use.

Feedforward neural networks (FFNNs) can encounter difficulties in accurately predicting the output for a given input, especially when dealing with complex problems. In such cases, the weights and biases of the network need to be adjusted to reduce the difference between the predicted output and the actual output. This is where backpropagation comes, as it allows the network to adjust its weights and biases (see Figure 2.9).

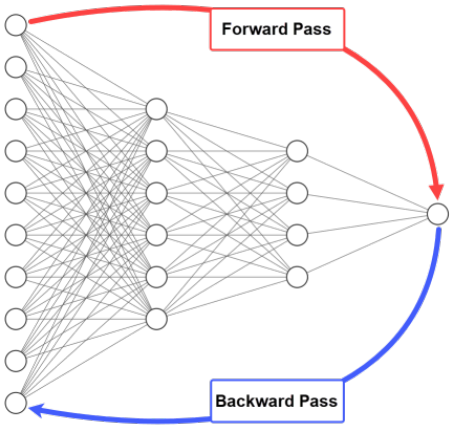


Figure 2.9: Forward and Backward Propagation.

2.6.2.2 Back Propagation

Back propagation is a popular learning algorithm that allows the network to adjust its weights and biases during training, by propagating the error from the output layer back to the input layer (see Figure 2.10). This is done by computing the partial derivatives of the error with respect to the weights and biases of each neuron in the network, using the chain rule of calculus.

Back propagation is used repeatedly during the training phase, where the network is presented with a set of input-output pairs. The weights and biases of the network are adjusted based on the computed partial derivatives, with the goal of minimizing the error between the predicted output and the actual output.

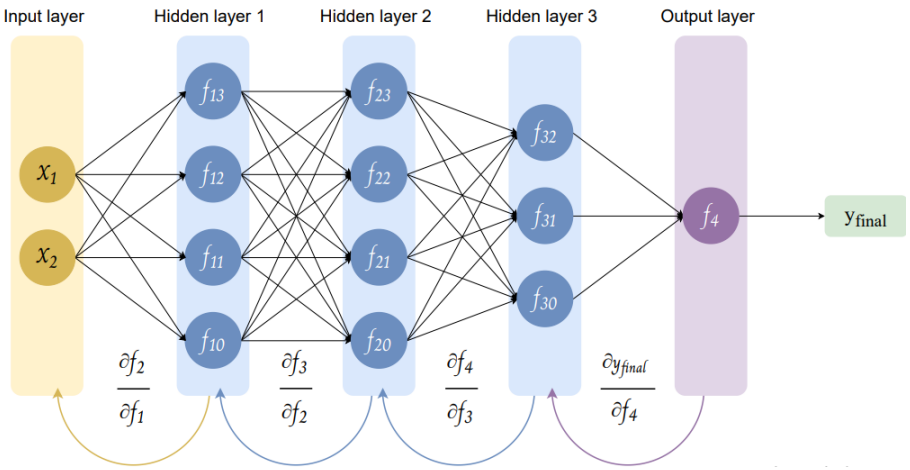


Figure 2.10: Backward Propagation.

In this way, back propagation allows the network to learn and improve its ability to accurately predict the output for a given input.

2.6.2.3 FFNNs applications

Feedforward neural networks (FFNNs) have a wide range of applications in various fields due to their ability to learn complex patterns and make accurate predictions. Here are some common applications of FFNNs:

- **Image and speech recognition**: FFNNs are commonly used in image and speech recognition systems to recognize patterns and make accurate predictions. For example, in image recognition, a FFNN can learn to identify objects in an image by processing the pixel values of the image.
- **Natural language processing**: FFNNs are used in natural language processing to analyze and process human language. For instance, they can be used to classify texts, extract relevant information, and generate responses.

- **Financial forecasting**: FFNNs can be used in financial forecasting to predict market trends and make investment decisions. They can be trained on historical financial data to learn patterns and make predictions about future trends.

- **Medical diagnosis**: FFNNs are used in medical diagnosis to analyze medical images and make accurate diagnoses. For example, a FFNN can be trained on a dataset of medical images to learn to identify signs of disease.

- **Robotics**: FFNNs can be used in robotics to control robotic systems and make accurate predictions about the environment. For example, a FFNN can be used to control a robotic arm to perform a task, such as picking up an object.

Overall, FFNNs are a versatile and powerful tool for solving a wide range of problems in various fields.

2.6.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks or CNNs, are deep neural networks designed for image recognition and computer vision tasks. They extract hierarchical features from input data, similar to the human visual system, and have shown high effectiveness in various image recognition tasks such as object detection, facial recognition, and image classification [35].

2.6.3.1 CNNs architecture

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers (see Figure 2.11).

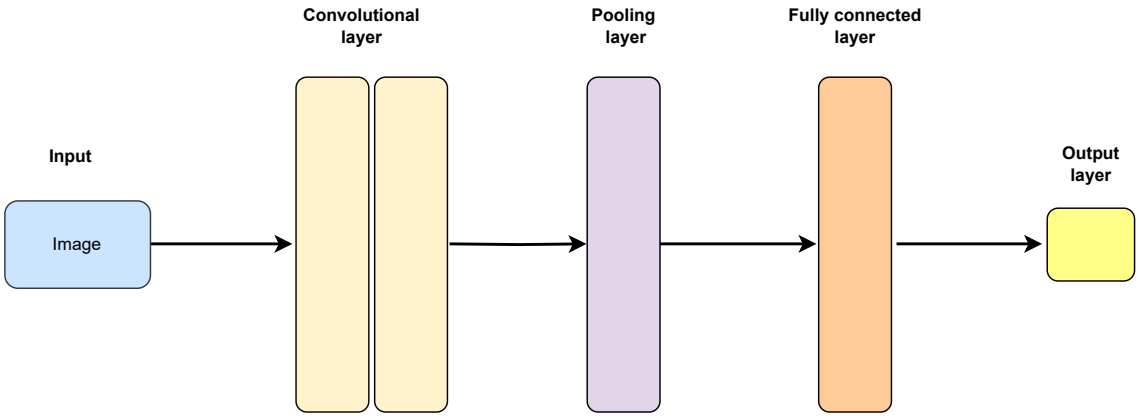


Figure 2.11: CNN Layers.

- **Convolutional Layer:** This layer applies a set of filters to the input image or feature map to extract features that are important for classification. The filters slide over the input data in a process known as convolution, which generates a feature map that highlights the presence of particular features in the input.

- **Pooling Layer:** This layer downsamples the feature maps generated by the convolutional layer, reducing its spatial dimensions while preserving its important features. The most common pooling operation is max pooling, which selects the maximum value within a particular region of the feature map and discards the rest.

- **Fully Connected Layer:** This layer performs the classification task, by taking the high-level features extracted by the convolutional and pooling layers and using them to classify the input image. It is typically implemented as a multi-layer perceptron, or MLP, and is trained using backpropagation and gradient descent.

In addition, CNNs may also include layers such as normalization layers, dropout layers, and activation layers, which can improve their performance and durability.

CNNs are a powerful and flexible tool with a wide range of applications in image recognition and computer vision tasks across various fields. Their ability to learn hierarchical features from raw data makes them ideal for complex tasks. As computing power and data availability increase, CNNs are expected to become even more important in the future.

2.6.4 Reccurent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a type of neural network that can analyze sequential data, such as time series or natural language. Unlike traditional feedforward neural networks, which process inputs in a single direction and have fixed-sized inputs and outputs, RNNs can take variable-length inputs and produce variable-length outputs. This makes them particularly useful for tasks such as speech recognition, machine translation, and sentiment analysis.

2.6.4.1 RNNs architecture

Recurrent Neural Networks (RNNs) are a type of neural network architecture that builds upon the structure of the traditional Feedforward Neural Networks (FFNNs). Like FFNNs, RNNs have an input layer, an output layer, and one or more hidden layers [36].

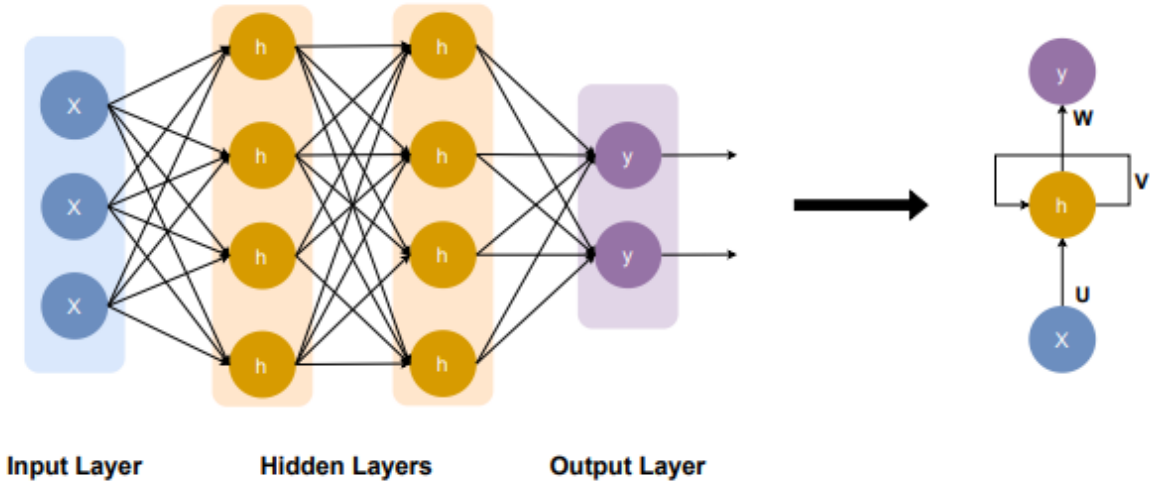


Figure 2.12: Forward and Backward Propagation.

In a Recurrent Neural Network (RNN), the input layer **'x'** processes input data and passes it onto the middle layer **'h'**, which can consist of multiple hidden layers with their own activation functions (see Figure 2.12), weights, and biases. Unlike traditional neural networks that lack memory, RNN uses a recurrent connection between hidden layers to maintain a memory of previous inputs. This enables the RNN to process sequential data such as time-series or natural language. At each time step, the current input is a combination of the input at that time and the previous input. The output at each time step is fed back into the network to improve subsequent outputs. Instead of creating multiple hidden layers, the RNN creates one and loops over it as many times as needed, standardizing the activation functions, weights, and biases across each loop.

Overall, RNNs have proven to be a powerful tool for analyzing sequential data and have shown impressive results in a wide range of applications.

- **Language Modeling and Generating Text:** RNNs are widely used in natural language processing (NLP) tasks, such as language modeling, text generation, and sentiment analysis.
- **Machine Translation:** RNNs are used in machine translation applications, such as translating text from one language to another.
- **Speech Recognition:** RNNs are used in speech recognition (ASR) and voice recognition systems.
- **Image Recognition:** RNNs are used in face detection, object recognition, and optical character recognition (OCR).

2.6.5 Activation Functions

The activation function is used to add non-linearity to each neuron's output. The network would only be a linear function of the input data without an activation function, which would have little ability to simulate complicated interactions between the inputs and outputs. There are several kinds of activation functions that are often applied in neural networks, each having special qualities and advantages. Let's examine some of these activation functions in more detail and how deep learning models employ them.

2.6.5.1 Sigmoid function

The sigmoid function was first introduced in the context of neural networks by McCulloch and Pitts in 1943, and later popularized by Rosenblatt in his perceptron algorithm in 1958. The sigmoid function is a mathematical function commonly used in neural networks and deep learning. It is an activation function that maps any input value to a value between 0 and 1, which can be interpreted as a probability [37].

The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

where x is the input value and exp is the exponential function. The sigmoid function has a characteristic S-shaped curve (see Figure 2.13), and its output values are always between 0 and 1. When the input is large and positive, the sigmoid function outputs a value close to 1, and when the input is large and negative, the sigmoid function outputs a value close to 0. When the input is zero, the sigmoid function outputs 0.5.

Today, the sigmoid function is one of the most commonly used activation functions in neural networks, although its usage has declined in favor of other activation functions such as ReLU due to its tendency to saturate for large inputs .

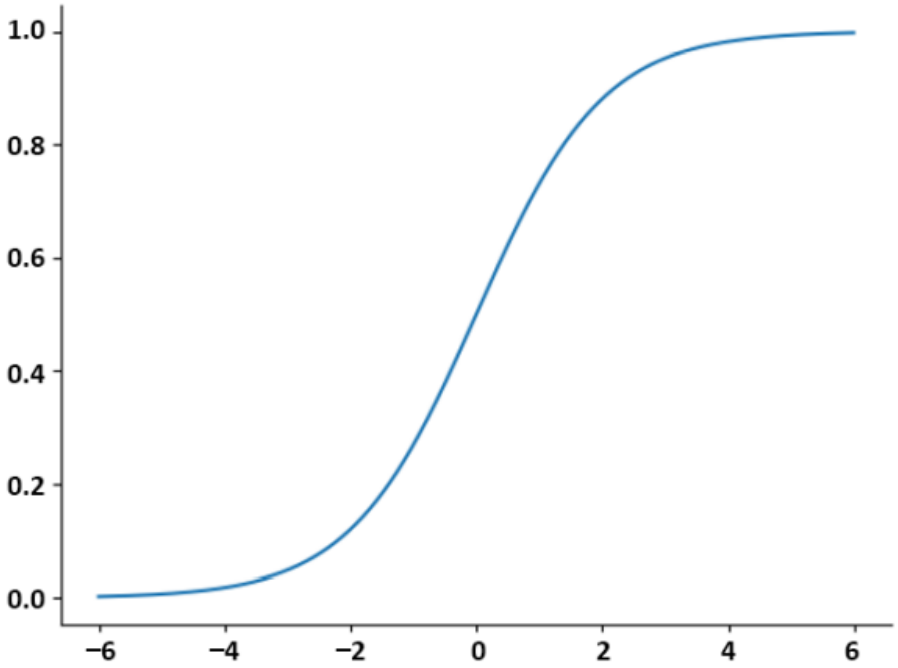


Figure 2.13: Sigmoid function.

2.6.5.2 ReLU function

The ReLU (Rectified Linear Unit) function has become one of the most widely used activation functions in Deep Learning, and has been shown to perform well in a variety of tasks. It was first introduced by Hahnloser et al. in 2000, and later popularized by Krizhevsky et al.

The ReLU function is an activation function used in neural networks. It is a simple yet powerful function that has gained popularity due to its effectiveness and ease of implementation [38].

The ReLU function is defined as:

$$f(x) = \max(0, x) \tag{2.2}$$

where x is the input value.

The ReLU function returns the input value if it is positive, and 0 otherwise. The function has a simple and fast computation, and is non-linear, which allows for the representation of complex non-linear relationships in data (see Figure 2.14).

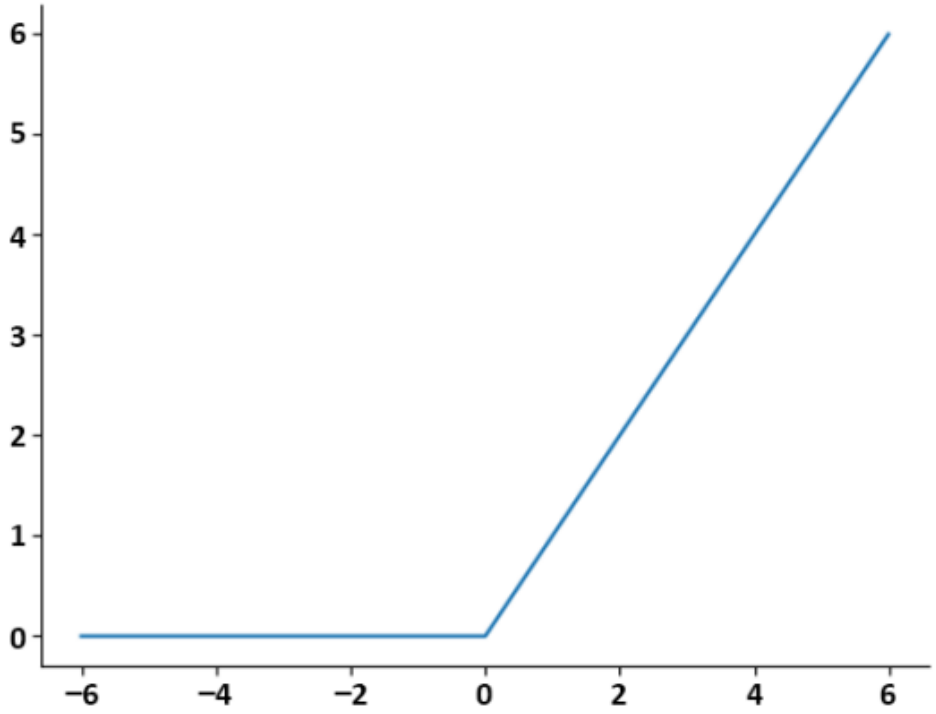


Figure 2.14: ReLU function.

2.6.5.3 Softmax function

The softmax function was introduced in the context of neural networks by Bridle in 1990, and later popularized by the work of Rumelhart, Hinton, and Williams in their seminal book "Parallel Distributed Processing" in 1986. The softmax function is often used as the output activation function in neural networks for multi-class classification problems. It is also commonly used in the calculation of the cross-entropy loss function, which is used as the objective function in many neural network training algorithms [39].

The softmax function is defined as follows:

$$softmax(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.3}$$

where x is a vector of real numbers, K is the number of classes, and j indexes the j-th class.

The softmax function takes as input a vector of K real numbers and outputs a vector of K probabilities, where each element of the output vector corresponds to the probability of the input vector belonging to a particular class, Figure 2.15 shows softmax function.

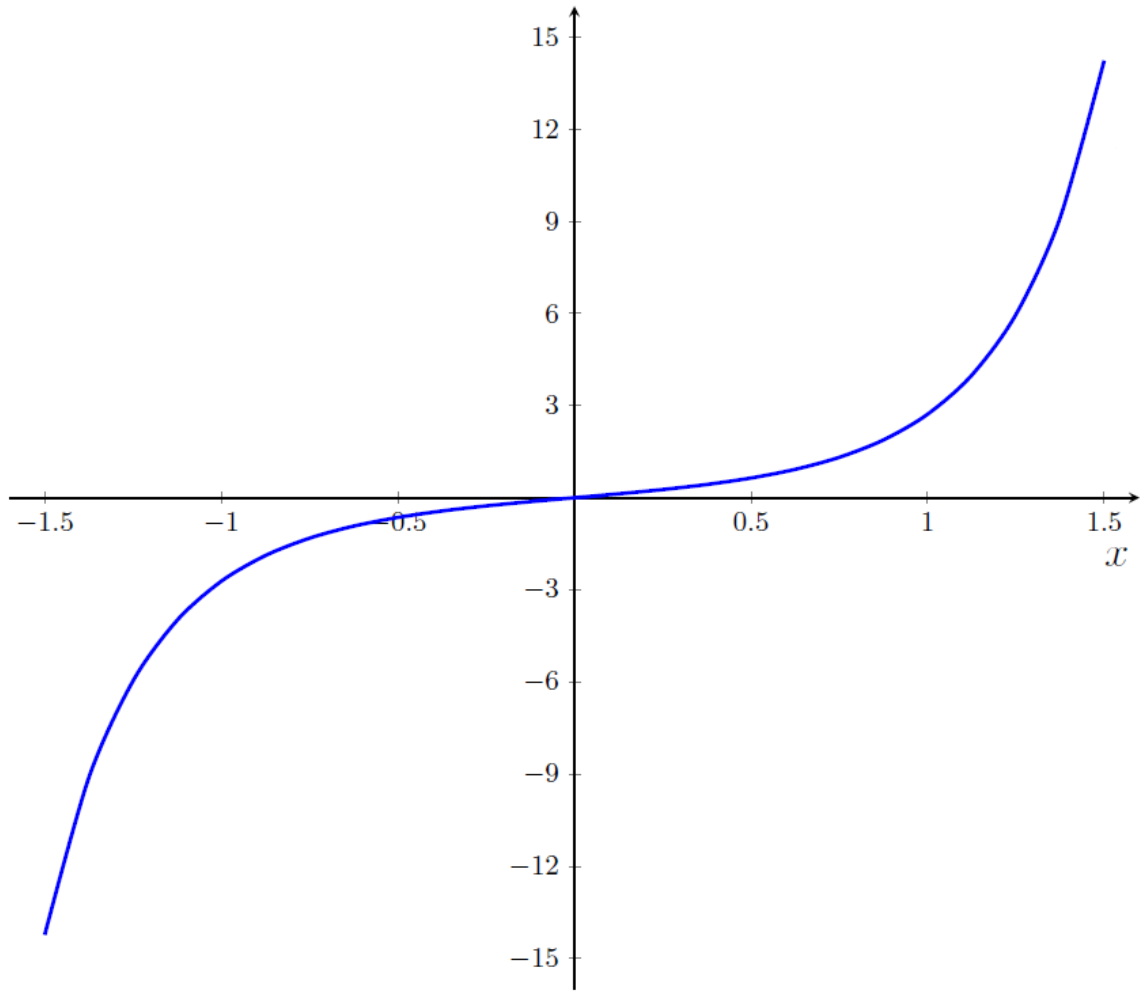


Figure 2.15: Softmax function.

2.6.6 Loss function

The loss function is a mathematical function that is used to evaluate how well a neural network model is performing on a given task. The loss function measures the difference between the predicted output of the model and the actual output, which is known as the ground truth. The objective of training a neural network is to minimize the value of the loss function.

There are various types of loss functions used in deep learning, some of them are:

- Mean Squared Error (MSE) Loss
- Mean Absolute Error (MAE) Loss
- Binary Cross-Entropy Loss
- Categorical Cross-Entropy Loss
- Sparse Categorical Cross-Entropy Loss
- Hinge Loss
- Kullback-Leibler Divergence Loss
- Cosine Similarity Loss
- Triplet Loss

The choice of loss function can have a significant impact on the performance of a neural network model, and selecting an appropriate loss function is an important part of designing a neural network architecture. For example in regression problems, the mean squared error (MSE) loss function is commonly used, while in binary classification problems, the binary cross-entropy loss function is commonly used. In multi-class classification problems, the categorical cross-entropy loss function is often used [40].

2.6.6.1 Binary Classification Loss Functions

Binary Classification Loss Functions are used in deep learning for problems where the output variable is binary, i.e. it can take only two values 0 or 1. Some of the commonly used Binary Classification Loss Functions are binary cross-entropy LOSS and hinge loss.

a. Binary Cross-Entropy Loss/LOG LOSS

This is the most common loss function used in classification problems. The cross-entropy loss decreases as the predicted probability converges to the actual label. It measures the performance of a classification model whose predicted output is a probability value between 0 and 1 as shown in Figure 2.16.

When the number of classes is 2, it's binary classification.

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log\left(1 - \hat{y}_i\right)\right) \tag{2.4}$$

When the number of classes is more than 2, it's multi-class classification.

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \log(\hat{y}_i)) \tag{2.5}$$

where y is the true binary label (0 or 1) and $\hat{y}$ is the predicted probability of the positive class.

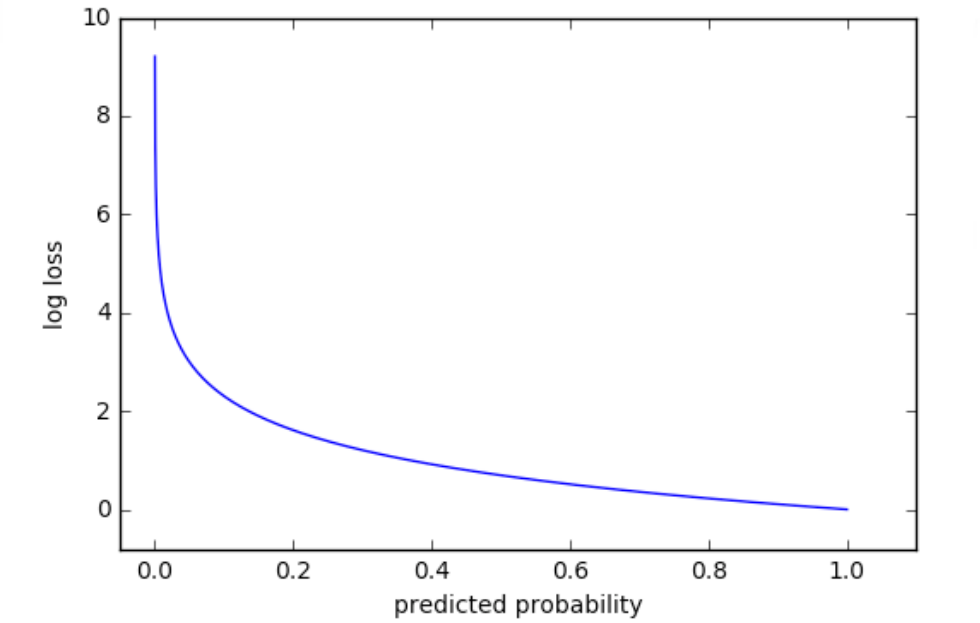


Figure 2.16: Log loss when true label=1.

b. HINGE LOSS

Hinge loss can be used as an alternative to cross-entropy, which was initially developed to use with a support vector machine algorithm. Hinge loss works best with the classification problem because target values are in the set of $\{-1, 1\}$. It allows to assign more error when there is a difference in sign between actual and predicted values. Hence resulting in better performance than cross-entropy.

$$L = \max(0, 1 - yf(x)) \tag{2.6}$$

where y is the true binary label (either -1 or 1) and $f(x)$ is the predicted score of the positive class for input x. The Hinge Loss function only penalizes misclassifications when the predicted score is on the wrong side of the decision boundary (i.e. $yf(x) < 1$), and is zero when the predicted score is correct (i.e. $yf(x) \geq 1$).

The objective of the Hinge Loss is to maximize the margin between the decision boundary and the training examples, while still correctly classifying the training examples.

2.6.6.2 Multi-class Classification Loss Functions

Multi-class classification is the predictive models in which the data points are assigned to multiple possible classes. Here are some commonly used multi-class classification loss functions in deep learning.

a. Categorical Cross-Entropy Loss

Categorical Cross-Entropy Loss is commonly used for multi-class classification problems. It measures the difference between the predicted class probabilities and the actual one-hot encoded class labels. The formula for Categorical Cross-Entropy Loss is:

$$CCE = -\sum_{i=1}^{K} y_i log(\hat{y}_i) \tag{2.7}$$

where y is the one-hot encoded true label vector, $\hat{y}$ is the predicted probability vector, and K is the number of classes.

b. Multi-class Cross-Entropy

One hot encoding process makes multi-class cross-entropy difficult to handle a large number of data. Sparse cross-entropy solves this problem by performing the calculation of error without using one-hot encoding.

$$SCCE = -\sum_{i=1}^{N} log(\hat{y}_{y_i}) \tag{2.8}$$

where y is the vector of true class labels and $\hat{y}$ is the predicted probability matrix.

c. Kullback Leibler Divergence Loss

KL divergence loss calculates the divergence between probability distribution and baseline distribution and finds out how much information is lost in terms of bits. It is used as a loss function to minimize the difference between the predicted class probabilities and the true class probabilities. The formula for KL divergence loss is:

$$KL = -\sum_{i=1}^{K} y_i log(\frac{y_i}{\hat{y}_i}) \tag{2.9}$$

Where K is the number of classes, y_i is the true probability of class i, and $\hat{y}_i$ is the predicted probability of class i. The KL divergence loss measures the difference between the true distribution and the predicted distribution, and is commonly used in probabilistic models.

2.7 Metrics for Evaluating the Performance of Deep Learning Models

2.7.1 Confusion Matrix

The confusion matrix is a tabular representation that assesses the classification model's performance for a given problem. It compares the model's predicted class labels with the true class labels from the test dataset, summarizing the results in a matrix format.

For binary classification problems, the confusion matrix consists of two rows and two columns that represent the predicted and actual classes (see Figure 2.17). The rows indicate the true class labels, while the columns show the predicted class labels. Every cell in the matrix represents the number of instances that belong to a particular combination of predicted and actual class labels.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positive (TP)	False Positive (FP)
Predicted Negative (0)	False Negative (FN)	True Negative (TN)

Figure 2.17: Confusion matrix.

- **True Positive (TP):** The number of instances that are correctly classified as positive. This means that the model predicted the instance to be positive, and the instance is actually positive.

- **True Negative (TN):** The number of instances that are correctly classified as negative. This means that the model predicted the instance to be negative, and the instance is actually negative.

- **False Positive (FP):** The number of instances that are incorrectly classified as positive. This means that the model predicted the instance to be positive, but the instance is actually negative.

- **False Negative (FN):** The number of instances that are incorrectly classified as negative. This means that the model predicted the instance to be negative, but the instance is actually positive.

2.7.1.1 Confusion Matrix for Multi-Class Classification

The confusion matrix for multiclass classification is an extension of the binary confusion matrix to handle problems with more than two classes. It provides a way to evaluate the performance of a multi-class classification model by summarizing the predictions and true class labels in a matrix format.

In multiclass classification, the confusion matrix has n rows and n columns, where n is the number of classes. Each row represents the instances of a true class label, while each column corresponds to the instances of a predicted class label (see Figure 2.18).

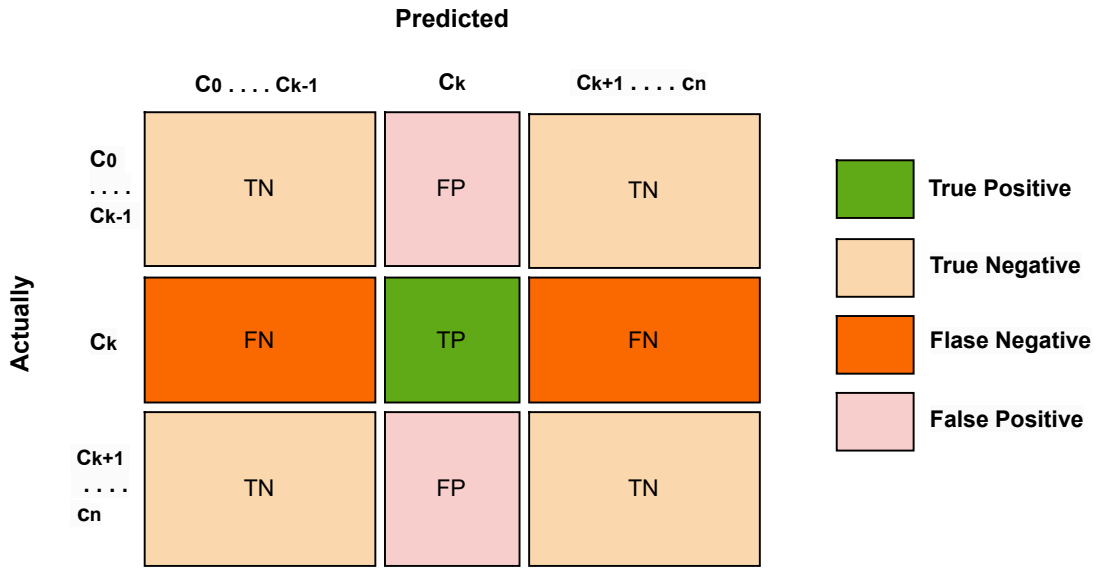


Figure 2.18: Confusion matrix multiclass classification.

2.7.2 Accuracy

Accuracy is one of the most commonly used metrics for classification problems. It measures the percentage of correct predictions made by the model. It is calculated by dividing the sum of true positive and true negative predictions by the total number of predictions made. This widely used metric provides an easy-to-understand measure of a model's overall performance, which can be readily communicated to both technical and non-technical stakeholders. For example, an accuracy of 80% means that a model correctly classified 80 out of 100 instances, providing a simple and clear assessment of its performance.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.10}$$

2.7.3 Precision

Precision measures the percentage of true positives out of all predicted positives. It is calculated as the number of true positives divided by the sum of true positives and false positives. Precision is a good metric when the goal is to reduce false positives, i.e. when we want to be sure that positive predictions are actually correct. Precision is often used in information retrieval where the goal is to retrieve as many relevant documents as possible while minimizing the number of irrelevant documents.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.11}$$

2.7.4 Recall

Recall, also known as sensitivity or true positive rate, is a performance metric used to evaluate a model's ability to correctly identify positive instances. It measures the proportion of actual positive instances that the model successfully classifies as positive, and it is calculated as the number of true positive predictions divided by the sum of true positive and false negative predictions. Recall is especially useful in scenarios where the cost of false negatives is high, such as medical diagnosis. However, recall has limitations, as it does not consider false positives, which may be essential in some applications.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.12}$$

2.7.5 Specificity

Specificity is a metric that measures a model's ability to correctly identify negative instances. It's useful in applications where the cost of false positives is high, such as in fraud detection or spam filtering. High specificity indicates that the model is effective at reducing false positives, but it may come at the cost of lower recall.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.13}$$

2.7.6 F1 score

The F1 score is the harmonic mean of precision and recall, calculated as:

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.14}$$

The F1 score is a suitable metric when both precision and recall are equally important. It balances the two metrics and provides a single measure of performance. The F1 score is commonly used in information retrieval tasks.

2.8 Conclusion

In this second chapter, we explored the fundamentals of machine learning and its subset, deep learning. We saw how deep learning has become a powerful tool in various domains, thanks to the availability of large datasets and advanced computing resources. With this knowledge, we are now well-equipped to apply these techniques in practice. In the next chapter, we will create a model that utilizes Deep Learning algorithms to identify and classify different types of attacks.

Chapter 3

Building and Evaluation a Deep Learning Model

3.1 Introduction

Achieving the best results in Deep Learning requires considering several factors, with choosing the right dataset being the most crucial. While a larger dataset typically leads to better model performance, what matters most is the number of records for each target. To address this, we utilized the **CSE-CIC-IDS2018 dataset**, a collaborative project between the Communication Security Establishment (CSE) and the Canadian Institute for Cybersecurity. This dataset enables the detection and classification of a wide range of attacks, including Bot, DDoS attacks, DoS attacks, and FTP Brute force. Initially, we took a first approach, but after evaluating our model with various methods such as accuracy, precision, recall, and F1 score, we were dissatisfied with the results. Therefore, we decided to improve the model by adopting a better approach.

3.2 Execution environment

Deep Learning relies heavily on hardware to execute tasks effectively. There are a few fundamental minimum hardware requirements that must be fulfilled before Deep Learning can begin [41], the basic minimum hardware requirements to start are:

1. **Graphics processing unit (GPU):** NVIDIA GTX1060, GTX1070 and above
2. **RAM:** 8GB and above.
3. **HDD/SSD:** 1TB HDD and good to have 256GB SSD(faster prepossessing).
4. **CPU:** i5 8th Gen and above.

Even with all the basic hardware requirements being met, dealing with large datasets can be a time-consuming task. However there are numerous cloud computing resources available that offer more RAM, CPUs, and GPUs either for free or for a limited time period. This can make data processing more efficient and significantly reduce the occurrence of errors. This has led us to use Kaggle Kernels as both a hardware and software alternative for our work.

Kaggle [42] is an online community platform that provides a range of tools and resources for data scientists, Machine Learning practitioners, and researchers. It offers a variety of services, including cloud-based Jupyter notebooks, GPU and TPU resources, data visualization tools, and more, all aimed at helping users work with data more efficiently.

In terms of software, Kaggle offers a broad range of tools and packages commonly used in data science and Deep Learning. This includes Jupyter Notebook along with popular programming language **Python** [43], alongside its widely-used libraries such as NumPy, Pandas, Scikit-learn, Tensor-Flow, and Keras.

Jupyter Notebook is a web-based interactive computing environment provided by Kaggle that allows users to create and share codes, visualizations, and texts in a single document. It is a popular tool among data scientists and Deep Learning practitioners, as it supports multiple programming languages such as Python.

Python is a widely used, high-level, interpreted programming language that is open source and offers an excellent approach to object-oriented programming. It is a popular deep learning language used by data scientists for a variety of Deep Learning projects and applications. With its well-defined syntax and dynamic typing, Python is a powerful tool for manipulating complex data structures and algorithms, making it an ideal language for developing Deep Learning models.

Library	**Description**	**Version**
Pandas [44]	Pandas is a Python library that provides data manipulation and analysis tools. It is widely used for data cleaning, data transformation, data exploration and data visualization.	1.3.5
TensorFlow [45]	TensorFlow is an open source library focused on deep neural network training, providing options to deploy deep learning models locally to a database, or to the cloud. It is an indispensable tool for the modern data scientist due to its advanced features and support for building and training deep learning models using the latest technologies.	2.11.0
NumPy [46]	NumPy is a Python library that provides support for large multidimensional arrays and matrices. It is commonly used in scientific computing and numerical analysis.	1.21.6
Scikit-learn [47]	Scikit-learn is a Python library that provides machine learning algorithms for classification, regression, clustering, and dimensionality reduction. It is commonly used in data analysis and predictive modeling.	1.0.2
Matplotlib [48]	Matplotlib is a Python library that provides data visualization tools. It is commonly used to create charts, histograms, scatterplots, and other visualizations to aid in understanding and communicating data.	3.5.3

Table 3.1: List of python libraries.

3.3 Descriptions of CSE-CIC-IDS2018 dataset

The CSE-CIC-IDS2018 dataset on AWS is a network attack detection dataset containing large-scale real network environments with millions of network packets. The dataset contains various types of attacks and normal activities, and provides a wide range of features such as timestamps, packet and flow level details, and network behavior patterns. The CSE-CIC-IDS2018 dataset is used to develop and evaluate new intrusion detection algorithms, test and validate cybersecurity tools, and provide valuable insights into the latest cyberattack trends and patterns. Due to its large size and quality, this collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC) was the perfect choice for our work.

The CSE-CIC-IDS2018 [16] dataset was collected over a period of 10 days, starting Wednesday, February 14th, 2018 at 10:32 am and ending Friday, March 2nd, 2018 at 3:55 pm, the dataset is distributed across ten different files (see Table 3.2) :

Files	**DayActivity**	**Size(MB)**	**Attacks found**
02-14-2018.csv	Wednesday	358.22 MB	Benign 64 % FTP-BruteForce 18 % Other (187589) 18 %
02-15-2018.csv	Tuesday	375.95 MB	Benign 95 % DoS attacks-GoldenEye4 % Other(10990) 1 %
02-16-2018.csv	Friday	333.72 MB	DoS attacks-Hulk 44 % Benign 43 % Other (139891) 13 %
02-20-2018.csv	Tuesday	4.05 GB	Benign 93% DDoS attacks-LOIC-HTTP 7%
02-21-2018.csv	Wednesday	328.89 MB	DDOS attack-HOIC 65% Benign 34% Other (1730) 1%
02-22-2018.csv	Thursday	382.64 MB	Benign 100% Brute Force -Web 0 % Other(113) 0%
02-23-2018.csv	Friday	382.84 MB	Benign 100 % Brute Force -Web0 %0 Other(204) 0%
02-28-2018.csv	Wednesday	209.25 MB	Benign 89% Infilteration 11% Other(33) 0%
03-01-2018.csv	Thursday	107.84 MB	Benign 72% Infilteration 28% Other(25) 0%
03-02-2018.csv	Friday	209.25 MB	Benign 73% Bot27 27%

Table 3.2: Description of files containing CIC-IDS2018 dataset.

The dataset consists of 16,093,053 instances with 84 features, and includes 14 different class labels, 1 normal label and 13 attack labels. The Tables 3.3 and 3.4 provides a comprehensive breakdown of the characteristics of the combined dataset, as well as a detailed account of the occurrence of each class label.

Dataset Name	CIC-IDS2018
Dataset Type	Multi class
Year of release	2018
Total number of instances	16093053
Number of features	84
Number of distinct classes	14

Table 3.3: Overall characteristics of CIC-IDS2018 dataset.

Class Labels	**Number of instances**
BENIGN	13484708
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87

Table 3.4: Class wise instance occurrence of CIC-IDS2018 dataset.

Features	**Dtype**	**Features**	**Dtype**
Flow ID	object	Bwd Pkts/s	float64
Src IP	object	Pkt Len Min	float64
Src Port	int64	Pkt Len Max	float64
Dst IP	object	Pkt Len Mean	float64
Dst Port	int64	Pkt Len Std	float64
Protocol	int64	Pkt Len Var	float64
Timestamp	object	FIN Flag Cnt	int64
Flow Duration	int64	SYN Flag Cnt	int64
Tot Fwd Pkts	int64	RST Flag Cnt	int64
Tot Bwd Pkts	float64	PSH Flag Cnt	int64
TotLen Fwd Pkts	float64	ACK Flag Cnt	int64
TotLen Bwd Pkts	float64	URG Flag Cnt	int64
Fwd Pkt Len Max	float64	CWE Flag Count	int64
Fwd Pkt Len Min	float64	ECE Flag Cnt	int64
Fwd Pkt Len Mean	float64	Down/Up Ratio	float64
Fwd Pkt Len Std	float64	Pkt Size Avg	float64
Bwd Pkt Len Max	float64	Fwd Seg Size Avg	float64
Bwd Pkt Len Min	float64	Bwd Seg Size Avg	float64
Bwd Pkt Len Mean	float64	Fwd Byts/b Avg	int64
Bwd Pkt Len Std	float64	Fwd Pkts/b Avg	int64
Flow Byts/s	float64	Fwd Blk Rate Avg	int64
Flow Pkts/s	float64	Bwd Byts/b Avg	int64
Flow IAT Mean	float64	Bwd Pkts/b Avg	int64
Flow IAT Std	float64	Bwd Blk Rate Avg	int64
Flow IAT Max	float64	Subflow Fwd Pkts	int64
Flow IAT Min	float64	Subflow Fwd Byts	int64
Fwd IAT Tot	float64	Subflow Bwd Pkts	int64
Fwd IAT Mean	float64	Subflow Bwd Byts	int64
Fwd IAT Std	float64	Init Fwd Win Byts	int64
Fwd IAT Max	float64	Init Bwd Win Byts	int64
Fwd IAT Min	float64	Fwd Act Data Pkts	int64
Bwd IAT Tot	float64	Fwd Seg Size Min	int64
Bwd IAT Mean	float64	Active Mean	float64
Bwd IAT Std	float64	Active Std	float64
Bwd IAT Max	float64	Active Max	float64
Bwd IAT Min	float64	Active Min	float64
Fwd PSH Flags	int64	Idle Mean	float64
Bwd PSH Flags	int64	Idle Std	float64
Fwd URG Flags	int64	Idle Max	float64
Bwd URG Flags	int64	Idle Min	float64
Bwd Header Len	int64	Fwd Pkts/s	float64
Fwd Header Len	int64	Label	object

Table 3.5: features present in the CIC-IDS2018 dataset.

3.4 Implementation

The proposed intrusion detection system (IDS) is illustrated in Figure 3.1.

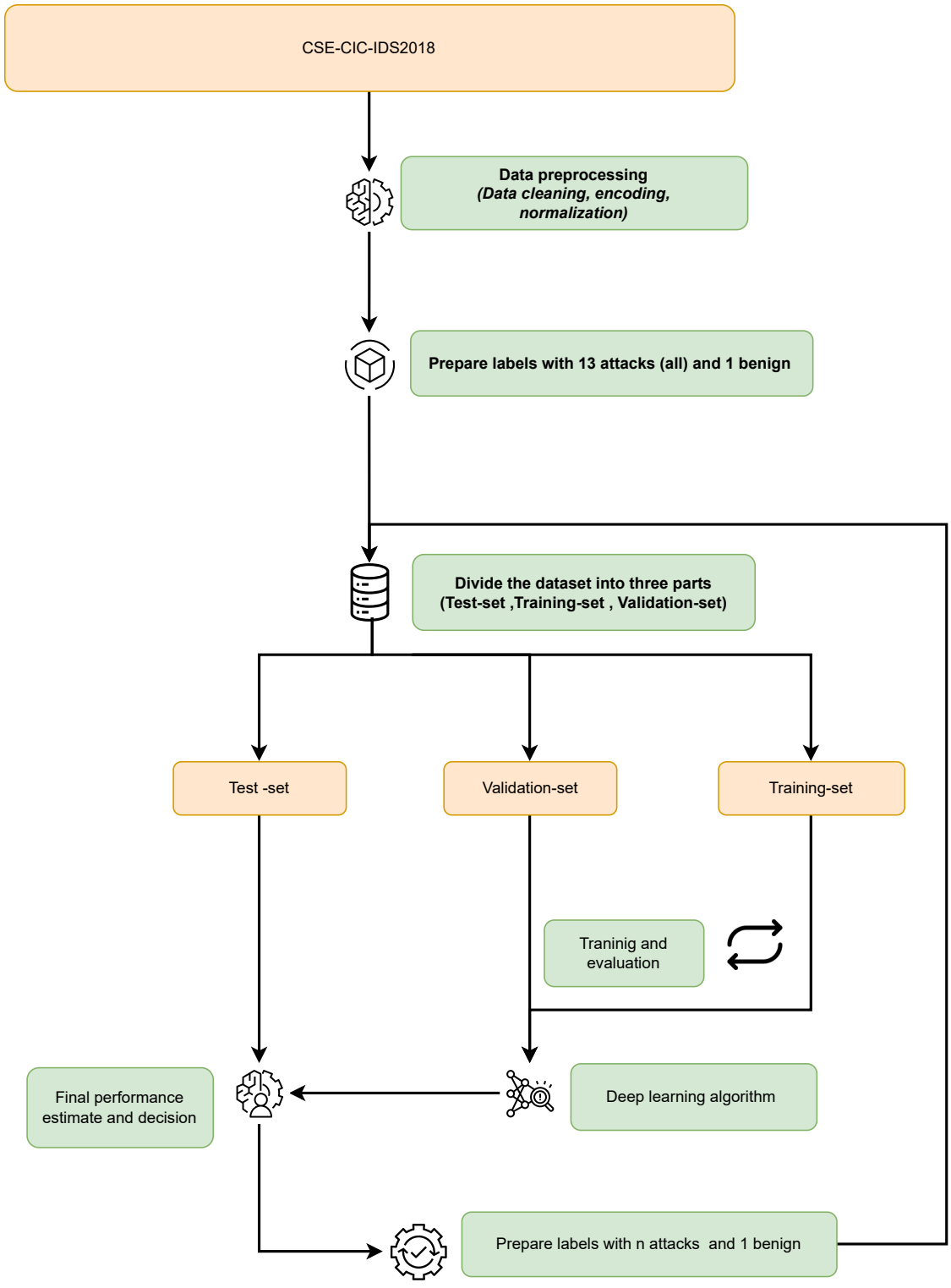


Figure 3.1: Component architecture of the proposed work.

3.5 Data pre-processing

Data preprocessing is the process of cleaning, transforming, and organizing raw data to make it ready for use in Deep Learning algorithms. It involves several techniques, such as data cleaning, handling missing data, and normalization. The aim of data preprocessing is to ensure that the data is consistent, accurate, and free from errors or inconsistencies that may impact the quality of the results. By performing proper data preprocessing, the performance of Deep Learning models and achieve more accurate and reliable predictions.

In Table 3.2, it is shown that the CIC-IDS2018 dataset is split into 10 separate files. Since processing each file individually would be a laborious task, the decision was made to simplify the data processing by merging all 10 files into a single file.

3.5.1 Merging files

Before starting the preprocessing phase, it is necessary for efficiency reasons to combine all files into a single file. However, two problems were encountered during this process. First, there was a type mismatch between the files, where some features were a combination of integers and floats in one file, but in the other all features were objects. To solve this, all features in each file have been converted to the same data type. Secondly, a file with a size of 4.05GB could not be merged due to its large size. As a solution, the benign instances were removed from this file, but the attack instances were kept since they were more relevant for the analysis. Despite the removal of some instances, there were still enough benign instances remaining for analysis.

3.5.2 Data cleaning

After reviewing our dataset, we removed missing values such as NaN (Not a Number) and INF (Infinity) values, as well as duplicates rows that could negatively impact our model's performance. We also identified and removed features that were not relevant to attacks (**Timestamp, Protocol, Flow ID , Src IP, Src Port, Dst IP**), additionally we removed features that contained null values (**Bwd PSH Flags, Bwd URG Flags, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg**), since including irrelevant features can add errors and reduce the accuracy of our model.

3.5.2.1 Cleaned data

We have successfully completed the process of cleaning and processing the raw data, resulting in a refined dataset that is now suitable for analysis. By eliminating missing or inconsistent values, correcting errors, and fixing outliers, we have created a more accurate and reliable dataset. Visualizing the cleaned data provides a clearer understanding of correlations and patterns between variables, which are critical for building models. With this improved dataset, we can confidently continue our analysis and modeling work. A part of the after cleaning process is represented in the Figure 3.2.

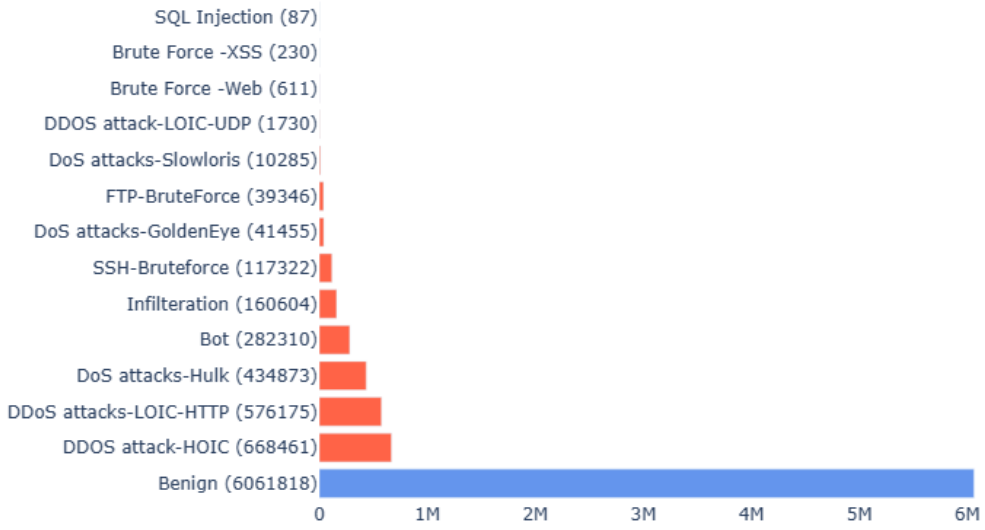


Figure 3.2: Distribution of labels in the Cleaned Dataset.

3.5.3 Label encoding

To fit models to the data, it must be represented in numerical format since the mathematics used in all Machine Learning and Deep Learning algorithms only work on matrices of numbers. Categorical variables such as labels can't be used directly as inputs to the models. Therefore, they need to be converted into numerical form before they can be used.

For example, in our case, we have labels for different types of network traffic, such as "benign", "DDOS-attack-HOIC", and so on. We assign a numerical value

to each label, such as 0 for "benign", 1 for "DDOS-attack-HOIC", and so on. This process is called label encoding. An alternative approach is to use one-hot encoding [49], where each label is represented as a binary vector with a 1 in the position corresponding to the label and 0s elsewhere, for example, "bening" would be represented as [1, 0, 0, ...], "DDOS-attack-HOIC" would be represented as [0, 1, 0, ...], and so on (see Figure 3.3). One-hot encoding ensures that there is no artificial hierarchy among the labels.

Label	Encoded	Convert to one-hot encoded
BENIGN	0	[1,0,0,0,0,0,0,0,0,0,0,0,0,0]
DDOS attack-HOIC	1	[0,1,0,0,0,0,0,0,0,0,0,0,0,0]
DDoS attacks-LOIC-HTTP	2	[0,0,1,0,0,0,0,0,0,0,0,0,0,0]
DoS attacks-Hulk	3	[0,0,0,1,0,0,0,0,0,0,0,0,0,0]
Bot	4	[0,0,0,0,1,0,0,0,0,0,0,0,0,0]
FTP-BruteForce	5	[0,0,0,0,0,1,0,0,0,0,0,0,0,0]
SSH-Bruteforce	6	[0,0,0,0,0,0,1,0,0,0,0,0,0,0]
Infilteration	7	[0,0,0,0,0,0,0,1,0,0,0,0,0,0]
DoS attacks-GoldenEye	8	[0,0,0,0,0,0,0,0,1,0,0,0,0,0]
DoS attacks-Slowloris	9	[0,0,0,0,0,0,0,0,0,1,0,0,0,0]
DDOS attack-LOIC-UDP	10	[0,0,0,0,0,0,0,0,0,0,1,0,0,0]
Brute Force -Web	11	[0,0,0,0,0,0,0,0,0,0,0,1,0,0]
Brute Force -XSS	12	[0,0,0,0,0,0,0,0,0,0,0,0,1,0]
SQL Injection	13	[0,0,0,0,0,0,0,0,0,0,0,0,0,1]

Figure 3.3: Label encoding/One-hot encoding

Once the labels have been encoded, we can isolate the column containing the encoded labels (also known as the target variable) from the original dataset.

3.5.4 Normalization

Normalizing data is an essential step in data preprocessing for statistical analysis and Machine Learning. The purpose of normalization is to transform the data to a standard scale that facilitates comparison and analysis of different variables on an equalized scale. Normalization can help identify patterns, outliers, and relationships within the data that may not be apparent otherwise. Additionally, normalizing the data can improve the performance and interpretability of statistical models by reducing the impact of variables with large scales and preventing bias towards particular features. We relied on min-max normalization as a key step in preprocessing the data.

Min-max normalization is a common and effective method for scaling data to a specific range [50]. This technique transforms the data so that it falls within a specified range (usually between 0 and 1) by subtracting the minimum value and dividing by the range of the data. Min-max normalization is useful for preserving the original distribution of the data while rescaling it to a specific range. Overall, normalization is a fundamental step in our work.

3.5.5 Splitting Data

In deep learning, splitting data into training, validation, and testing sets is a crucial step in model development. The purpose of splitting the data is to assess the model's performance on new, unseen data and to prevent overfitting.

To split the data, the dataset is typically divided into three subsets: training, validation, and testing. In our case, the training dataset comprises 90% of the original data, and the testing dataset comprises 10% of the original data, (see Figure 3.4).
Additionally, 10% of the training dataset is further partitioned into a validation dataset. The purpose of the validation dataset is to assess the model's performance during training and to optimize the model's hyperparameters.
During training, the model is trained on the training dataset and its performance is evaluated on the validation dataset. The testing dataset is held out until the end of the training process, and the model is evaluated on this dataset to assess its performance on unseen data.
By splitting the data into training, validation, and testing sets, it helps us confirm that the model can perform effectively on unfamiliar data and prevent the issue of overfitting, which occurs when the model becomes too closely aligned with the training data.

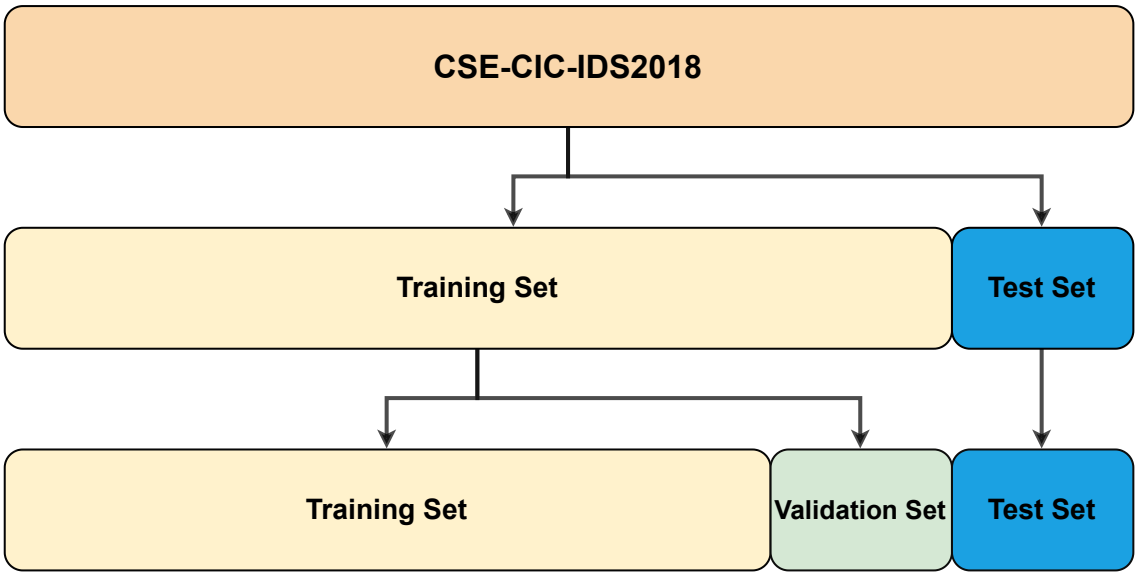


Figure 3.4: Splitting the CIC-IDS2018 dataset into three parts.

3.6 Model creation

Before creating the model, we carried out several essential steps to prepare for Deep Learning. We started by merging multiple files into a single more comprehensive file to create a complete representation of the data. We then performed data cleaning by removing duplicates and irrelevant data points, and addressing outliers or errors to ensure that the model could understand the categorical variables, we then performed one-hot encoding to convert them into numerical values. We also normalized the data to ensure that each feature had a similar scale, which can improve the performance of Deep Learning algorithms. Finally, we split the data into training and testing sets to assess the performance of the model on new unseen data.

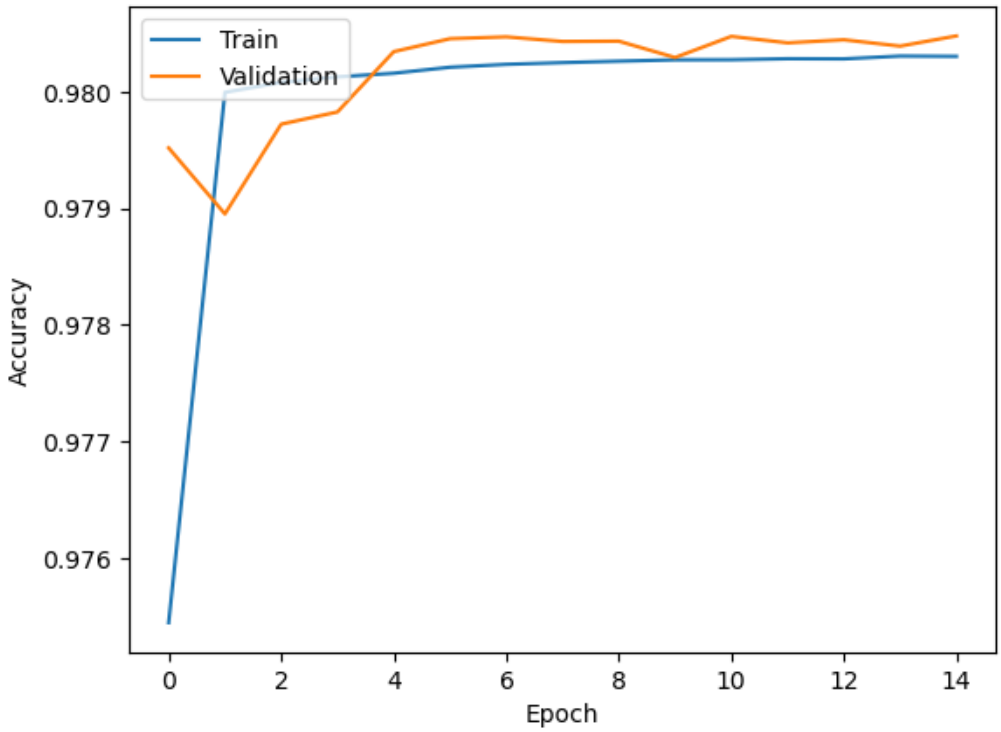
For our Deep Learning model, we chose the Sequential model provided by the Keras library [51]. This framework is widely used for building Deep Learning models due to its flexibility and ease of use. The Sequential model is a linear stack of layers, where each layer is added one after the other. The Sequential model is easy to use and is suitable for building models that involve a single input and output.

Designing and training a neural network involves several key steps. The first step is to choose an appropriate type of neural network architecture that best suits the problem at hand. We relied on the feedforward neural network (FFNN). Once the architecture of a neural network model is selected, the next step is to specify its hyperparameters. We created a model with six layers: an input layer, four hidden layers and one output layer. The input layer has 69 nodes corresponding to the number of features in the input data.

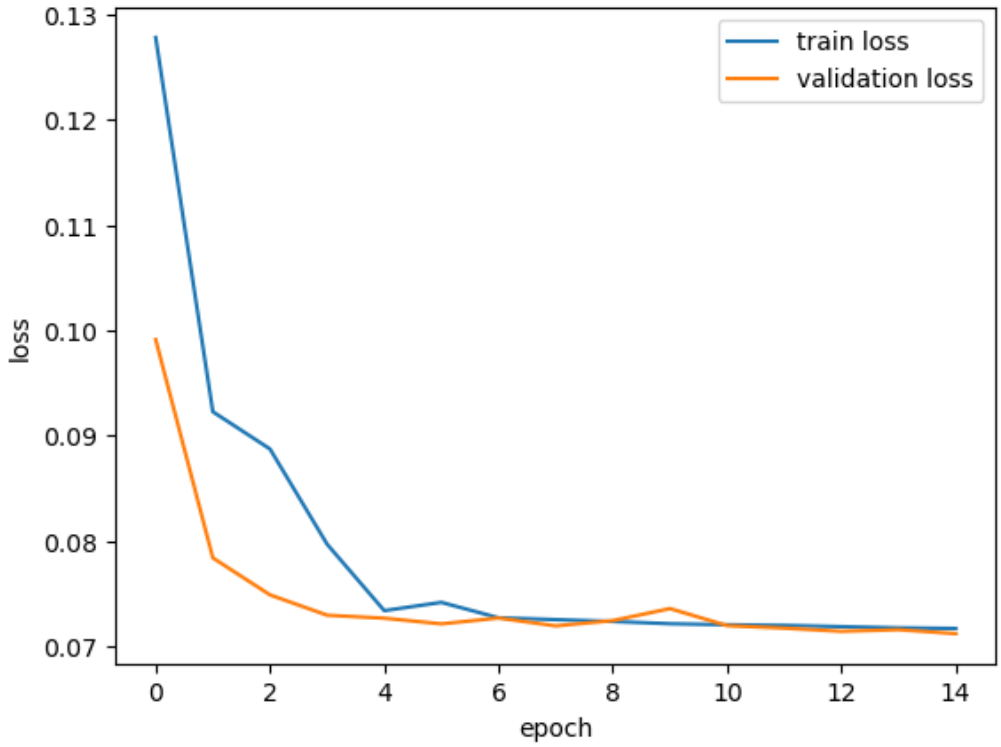
The first hidden layer has 47 neurons, the second has 32, the third has 27, and the fourth has 17. The output layer has 14 neurons, which corresponds to the number of output classes. The Rectified Linear Unit (ReLU) activation function is used for all the hidden layers, which is a commonly used activation function for feedforward neural networks due to its ability to efficiently model nonlinear relationships. The Softmax activation function is used for the output layer, which produces a probability distribution over the different classes and is often used for multiclass classification problems. The optimizer used is Adam, which is a popular optimization algorithm for neural networks due to its ability to efficiently handle large amounts of data and converge to a good solution. The loss function used is Categorical Crossentropy, which is commonly used for multiclass classification problems to measure the difference between the predicted probabilities and the actual class labels.
Finally, the model is trained on the training data, and the weights of the neurons are adjusted to minimize the loss function. This iterative process continues until the model converges to a set of weights that provide the best performance on the training data.

3.7 Results and analysis

After training the neural network model using a 32 batch size for 14 epochs, it is worth noting that during the training process, the dataset was divided into batches, with each batch containing 32 training examples. The batch size determines how many examples are used in each batch and is an important hyperparameter to consider when training neural networks. In this case, we also used a validation split of 0.1 and set the shuffle parameter to True, which means that the dataset was randomly shuffled at the beginning of each epoch to prevent the model from learning the order of the training examples and potentially overfitting to the order of the data. Shuffling the data ensures that the model sees a random sample of the training examples in each batch and allows it to learn more robust representations. The results obtained are shown in the Figure 3.5.



(a) Training and Validation Accuracy.



(b) Training and Validation Loss.

Figure 3.5: Visualizing Model Performance.

Based on the results we achieved, it appears that our model has reached an impressive level of accuracy. During the first epoch of training, the model achieved an accuracy of 98%, indicating its ability to correctly classify data. We were also pleased to see that the model stabilized by the fourth epoch, which suggests that it has likely converged on a reliable set of parameters, which shows no sign of neither overfitting nor underfitting.

Moreover, we observed that our model's loss decreased sharply in the first epoch and then stabilized as the training progressed, implying that it has learned to make more accurate predictions. We believe that this stabilization after a few epochs is a sign that our model has reached an acceptable level of training and is unlikely to improve significantly further. This demonstrates that our model is capable of learning and fitting well to the training data.

Accuracy, Precision, Recall and F1 score were effective metrics for evaluating the performance of our model. They provided us with a more comprehensive evaluation of our model's performance and helped us to make informed decisions about its development. We were able to identify areas where our model could be improved, such as optimizing the decision threshold for classification or balancing the class distribution in the dataset. The results are shown in the Table 3.6 below.

Accuracy	**Precision**	**Recall**	**F1-score**
98.03%	91.52%	80.17%	81.51%

Table 3.6: Evaluation Metrics.

Upon examining the previous table, it's apparent that the recall and F1 score values are notably lower than the accuracy and precision metrics. This suggests that the model may encounter difficulty in identifying true positives and has a higher rate of false negatives. To better understand the drop in percentage, we can utilize the confusion matrix. The confusion matrix provides a detailed breakdown of the model's performance by displaying the number of true positives, true negatives, false positives, and false negatives. By analyzing the confusion matrix (see Figure 3.6), we can pinpoint areas that require improvement and adjust our training approach accordingly to enhance the model's performance.

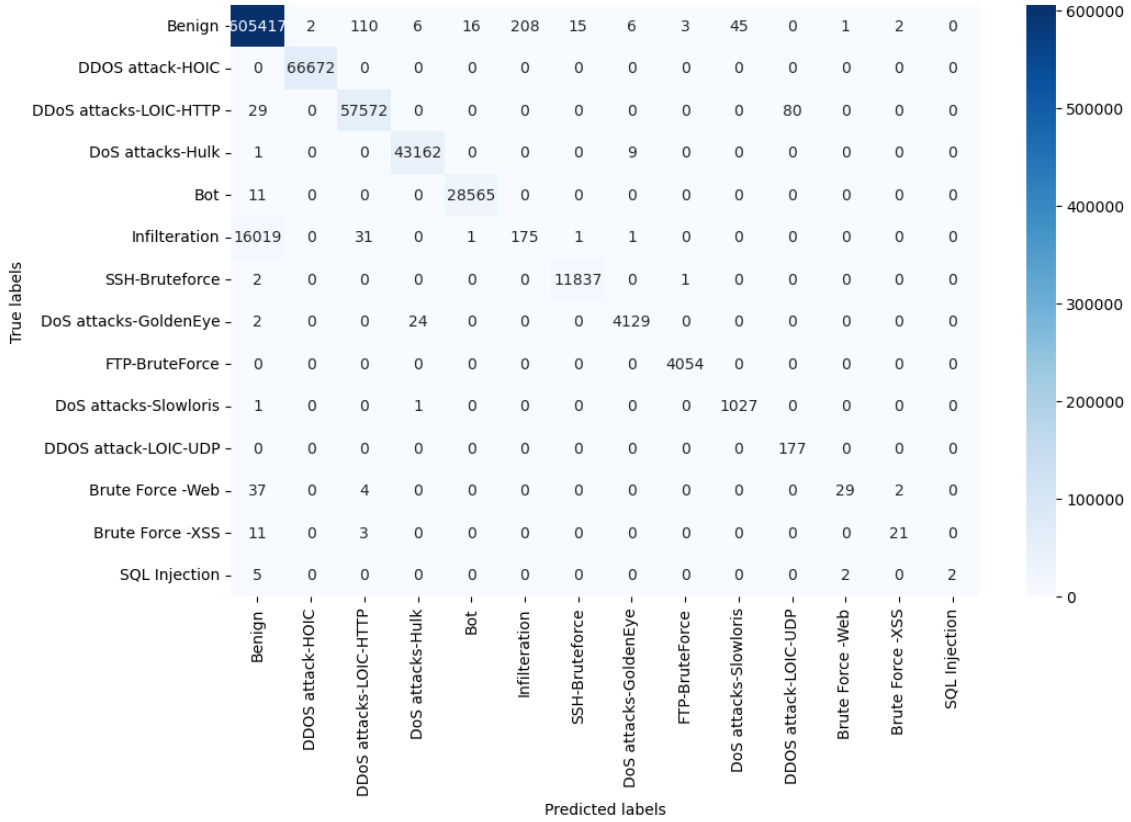


Figure 3.6: Confusion matrix multiclass classification.

After examining the confusion matrix to gain insights into our model's performance, we have identified three attacks that were mostly detected as false negatives: **SQL injection, Brute Force-XSS, and Brute Force-Web**. Additionally one particular attack **"DDOS attack-LOIC-UDP"** that may initially appear to be well detected, but actually has 80 samples that make up 30% of the total samples, which were wrongly classified as false positives, which could be due to insufficient training examples of these attacks (see table 3.4). Insufficient examples can hinder the model's ability to learn the patterns of attacks [52] leading to miss-identification.

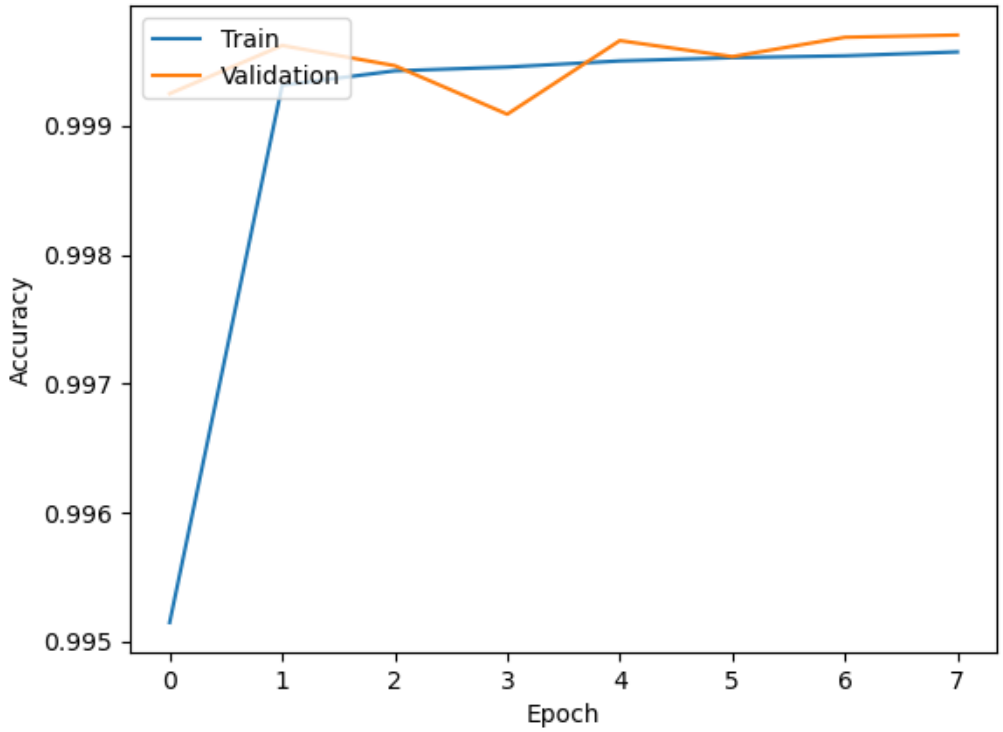
Moreover, the **"Infiltration"** attack was miss-classified as a non-attack, despite having a good amount of training examples. This could be attributed to the nature of the attack [53], which involves mimicking normal human behavior to evade detection and gain access to a system or network.

To address these issues we may need to consider an alternative approach, such as deleting these attacks and considering a model with only 8 attacks.

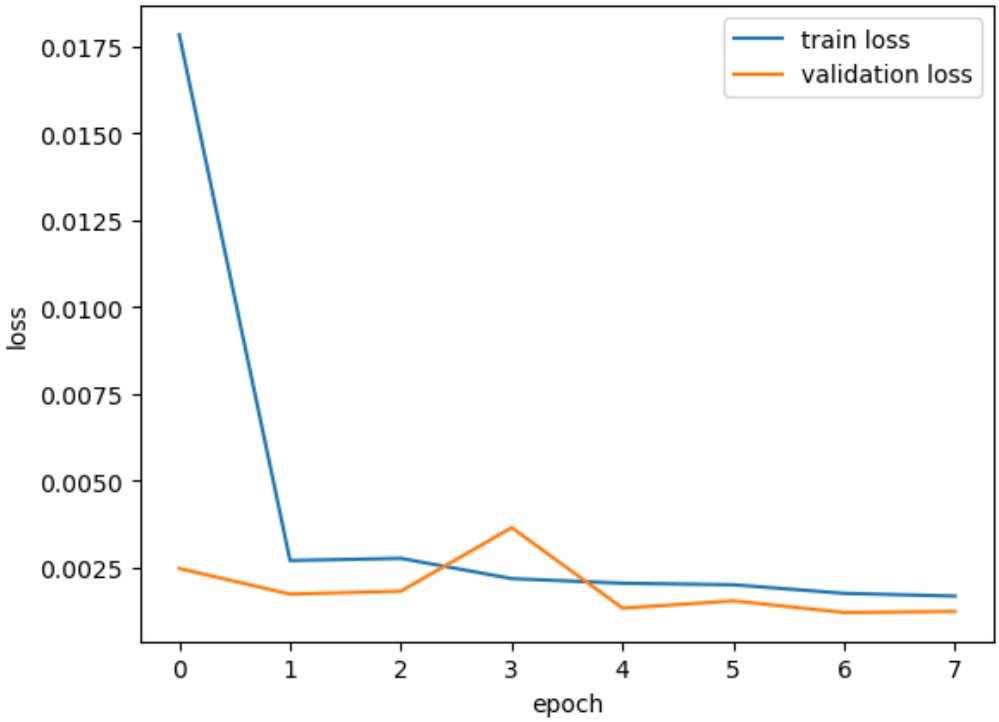
3.8 Improved Approach

In order to enhance our model's performance, we decided to remove the problematic attacks, based on the results of our initial approach. This decision was made particularly because obtaining additional data to improve the model was challenging, and we believe that eliminating these attacks will give our model a better chance to succeed.

We followed the same path as our initial approach, with the exception of removing the five attacks from our target. This change led to a significant improvement in our results, including higher accuracy, precision, recall, and F1 score. Additionally, our model exhibited better detection capabilities. To provide a comprehensive understanding of our findings, we included an in-depth analysis of each attack's confusion matrix, accuracy, F1 score, precision, and recall. The results of our work are presented in the figures below.



(a) Training and Validation Accuracy.



(b) Training and Validation Loss.

Figure 3.7: Visualizing improved Model Performance.

Figure 3.7 visualizes improved model performance in terms of accuracy and loss.

Accuracy	**Precision**	**Recall**	**F1-score**
99.97%	99.66%	99.96%	99.81%

Table 3.7: Evaluation Metrics Performance for the improved model.

Table 3.7 shows evaluation metrics for the improved model while Table 3.8 shows evaluation metrics for each label.

Types attacks	Accuracy	Precision	Recall	F1-score
Benign	99.97%	99.99%	99.97%	99.98%
DDOS attack-HOIC	100%	100%	100%	100%
DDoS attacks-LOIC-HTTP	99.95%	99.80%	99.96%	99.88%
DoS attacks-Hulk	99.99%	99.98%	99.99%	99.99%
Bot	99.98%	99.98%	99.98%	99.98%
FTP-BruteForce	99.97%	99.92%	99.98%	99.95%
SSH-Bruteforce	99.84%	99.72%	99.85%	99.79%
DoS attacks-GoldenEye	99.99%	99.97%	99.99%	99.99%
DoS attacks-Slowloris	99.90%	97.64%	99.91%	98.77%

Table 3.8: Evaluation Metrics for each label.

Figures 3.8 and 3.9 illustrate the model's confusion matrix and confusion matrix for each label respectively.

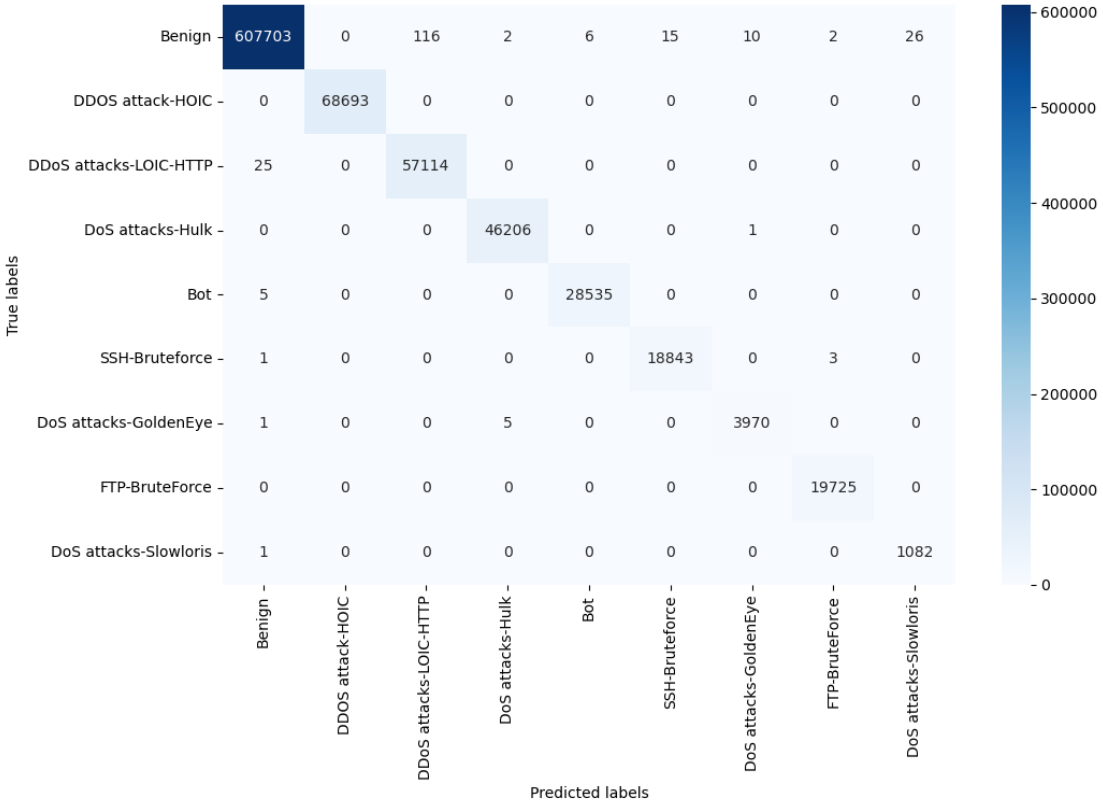


Figure 3.8: Confusion matrix multiclass classification for the improved model.

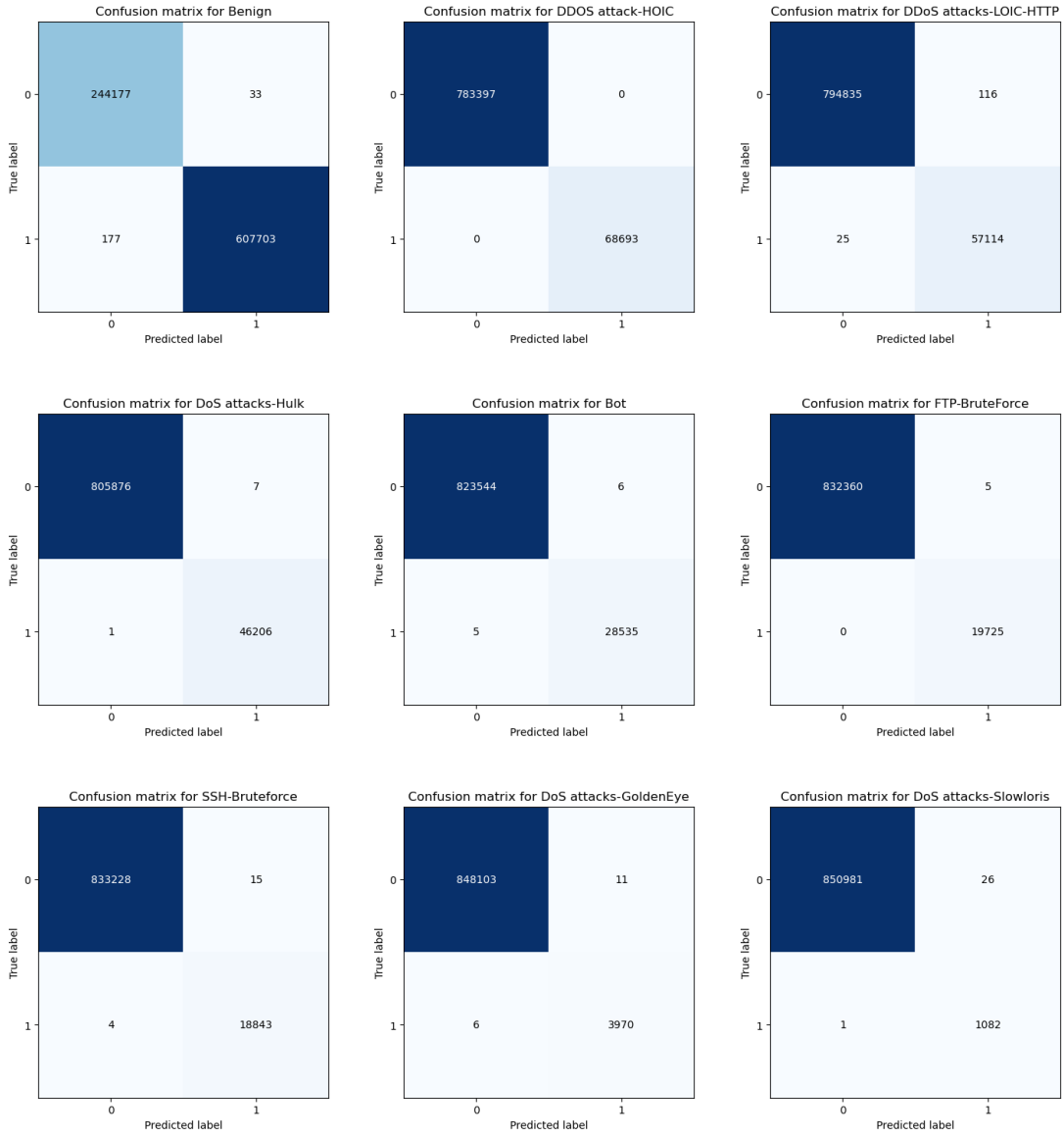


Figure 3.9: Confusion matrix of each label.

3.9 Conclusion

The goal of the final chapter was to present a solution that effectively addresses the various limitations of intrusion detection systems. The proposed solution enhances the overall performance of IDS by increasing the accuracy of detecting and categorizing a wide range of attacks while minimizing false alarms. To ensure that IDS can recognize patterns and identify new and previously unknown attacks, it is important to have a wide range of samples for each type of attack. The experiments conducted validate the effectiveness of the proposed approach, producing highly satisfactory results.

General Conclusion

The field of network security is of utmost importance due to the increasing concern for preventing security breaches that could result in catastrophic consequences such as data theft, network downtime, or financial loss. Network security administrators are always searching for solutions to guarantee a highly secure network environment. One of these solutions is the implementation of deep learning-based IDS.

In this master thesis, we aimed to develop a model for the detection and classification of network anomalies using DL techniques. We selected the CSE-CIC-IDS2018 dataset for training, as the performance of our models depends entirely on the quality of this data. We applied several data pre-processing techniques, including data cleaning, handling missing data, and normalization, to prepare our data for the training phase. After training and testing, we discovered that 5 attacks were misdetected using the approach of taking all attacks and normal traffic. Thus, we adopted a second approach, deleting the misdetected attacks, which yielded promising results. We consider the reliability and efficiency of our approach to be satisfactory.

One limitation we encountered during the experiments was a lack of data, which affected the learning process and test results, highlighting the need for more diverse and representative datasets.

To enhance network security, ongoing training and education on the latest developments and emerging threats in the field of network security are crucial. Future research can explore the use of more sophisticated DL techniques, such as adversarial training or ensemble models, and investigate the integration of IDS with other security mechanisms, such as firewalls, honeypots, and IPS/IDPS. Additionally, future research can focus on developing hybrid models that combine machine learning techniques with traditional rule-based systems and use explainable AI to improve the interpretability of DL-based IDS. By advancing our understanding of deep learning-based IDS, we can better protect against emerging cyber threats and further enhance network security.

Bibliography

[1] "30 important cybersecurity statistics [2023]: Data, trends and more," Zippia. Section: Research. (Feb. 27, 2023), [Online]. Available: https://www.zippia.com/advice/cybersecurity-statistics/ (visited on 04/12/2023).

[2] J. F. Kurose and K. W. Ross, Computer networking: a top-down approach, Seventh edition. Boston: Pearson, 2017, 824 pp., ISBN: 978-0-13-359414-0.

[3] "What is a denial-of-service (DoS) attack?" Cloudflare. (), [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/ (visited on 04/28/2023).

[4] S. Behal and K. Kumar, "Characterization and comparison of ddos attack tools and traffic generators: A review," International Journal of Network Security, vol. 19, no. 3, pp. 383–393, 2017.

[5] "Goldeneye DDos tool in kali linux," GeeksforGeeks. Section: Linux-Unix. (Jun. 21, 2021), [Online]. Available: https://www.geeksforgeeks.org/goldeneye-ddos-tool-in-kali-linux/ (visited on 04/28/2023).

[6] "Slowloris DDOS attack tool in kali linux - GeeksforGeeks." (), [Online]. Available: https://www.geeksforgeeks.org/slowloris-ddos-attack-tool-in-kali-linux/ (visited on 04/28/2023).

[7] S. Chatterjee, HULK, original-date: 2022-05-25T15:52:26Z, Apr. 20, 2023. [Online]. Available: https://github.com/R3DHULK/HULK (visited on 04/28/2023).

[8] "HTTP flood DDoS attack," Cloudflare. (), [Online]. Available: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/ (visited on 04/28/2023).

[9] "UDP flood DDoS attack," Cloudflare. (), [Online]. Available: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/ (visited on 04/28/2023).

[10] "Brute force attack," Techopedia. (Jul. 1, 2020), [Online]. Available: https://www.techopedia.com/definition/18091/brute-force-attack (visited on 04/28/2023).

[11] "How to brute-force FTP credentials & get server access," WonderHowTo. (Mar. 11, 2020), [Online]. Available: https://null-byte.wonderhowto.com/how-to/brute-force-ftp-credentials-get-server-access-0208763/ (visited on 04/28/2023).

[12] R. Ashraf. "How to brute-force SSH in linux," Root Install. (Nov. 9, 2021), [Online]. Available: https://rootinstall.com/tutorial/bruteforce-ssh-in-linux/ (visited on 04/28/2023).

[13] "Brute force attack | OWASP foundation." (), [Online]. Available: https://owasp.org/www-community/attacks/Brute_force_attack (visited on 04/28/2023).

[14] "What is cross-site scripting and how can you fix it?" Acunetix. (), [Online]. Available: https://www.acunetix.com/websitesecurity/cross-site-scripting/ (visited on 04/28/2023).

[15] "What is SQL injection? | cloudflare." (), [Online]. Available: https://www.cloudflare.com/learning/security/threats/sql-injection/ (visited on 04/29/2023).

[16] "IDS 2018 | datasets | research | canadian institute for cybersecurity | UNB." (), [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html (visited on 03/27/2023).

[17] "What is a botnet?" Palo Alto Networks. (), [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-botnet (visited on 04/29/2023).

[18] W. Stallings and L. Brown, Computer security: principles and practice (The William Stallings books on computer and data communications technology), Third edition. Boston: Pearson, 2015, 820 pp., ISBN: 978-0-13-377392-7.

[19] "What is a firewall? - cisco." (), [Online]. Available: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html (visited on 04/29/2023).

[20] "What is a VPN? - virtual private network - cisco." (), [Online]. Available: https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html?dtid=osscdc000283#~types-of-vpns (visited on 04/29/2023).

[21] "What is asymmetric encryption? | asymmetric vs. symmetric encryption," Cloudflare. (), [Online]. Available: https://www.cloudflare.com/learning/ssl/what-is-asymmetric-encryption/ (visited on 04/29/2023).

[22] Q. H. Dang, "Secure hash standard," National Institute of Standards and Technology, NIST FIPS 180-4, Jul. 2015, NIST FIPS 180–4. DOI: 10.6028/NIST.FIPS.180-4. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (visited on 04/29/2023).

[23] "What is security information and event management (SIEM)? | IBM." (), [Online]. Available: https://www.ibm.com/topics/siem (visited on 04/29/2023).

[24] "Intrusion detection system (IDS) - GeeksforGeeks." (), [Online]. Available: https://www.geeksforgeeks.org/intrusion-detection-system-ids/ (visited on 04/29/2023).

[25] "Network based intrusion detection system - an overview | ScienceDirect topics." (), [Online]. Available: https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system (visited on 04/30/2023).

[26] "Host intrusion detection system (HIDS). what is it and how it works." (), [Online]. Available: https://heimdalsecurity.com/blog/host-intrusion-detection-system-hids/ (visited on 04/30/2023).

[27] "A systematic review on hybrid intrusion detection system." (), [Online]. Available: https://www.hindawi.com/journals/scn/2022/9663052/ (visited on 04/30/2023).

[28] "What are the pros and cons of signature-based vs. anomaly-based detection?" (), [Online]. Available: https://www.linkedin.com/advice/0/what-pros-cons-signature-based-vs-anomaly-based (visited on 04/30/2023).

[29] A. A. Ramaki and R. E. Atani, "A survey of it early warning systems: Architectures, challenges, and solutions," Security and Communication Networks, vol. 9, no. 17, pp. 4751–4776, 2016.

[30] K. A. Scarfone and P. M. Mell, "Guide to intrusion detection and prevention systems (IDPS)," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007, Edition: 0, NIST SP 800–94. DOI: 10.6028/NIST.SP.800-94. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf (visited on 05/01/2023).

[31] E. Alpaydin, Introduction to Machine Learning. Cambridge, MA: MIT Press, 2010.

[32] "Loss functions in machine learning | working | different types," EDUCBA. (Sep. 9, 2019), [Online]. Available: https://www.educba.com/loss-functions-in-machine-learning/ (visited on 04/08/2023).

[33] M. Moocarme, M. Abdolahnejad, and R. Bhagwat, The Deep Learning with Keras Workshop: Learn how to Define and Train Neural Network Models with Just a Few Lines of Code. Packt Publishing, 2020, ISBN: 978-1-80056-296-7. [Online]. Available: https://books.google.dz/books?id=jEu6zQEACAAJ.

[34] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," Knowledge-Based Systems, vol. 189, p. 105 124, 2020, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.105124. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705119304897.

[35] "Introduction to convolution neural network," GeeksforGeeks. Section: Misc. (Aug. 21, 2017), [Online]. Available: https://www.geeksforgeeks.org/introduction-convolution-neural-network/ (visited on 04/14/2023).

[36] "Recurrent neural network (RNN) tutorial: Types and examples [updated] | simplilearn," Simplilearn.com. (), [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn (visited on 04/14/2023).

[37] "Sigmoid function? all you need to know in 5 simple points | UNext." (), [Online]. Available: https://u-next.com/blogs/artificial-intelligence/sigmoid-function/ (visited on 04/08/2023).

[38] "ReLU activation function explained | built in." (), [Online]. Available: https://builtin.com/machine-learning/relu-activation-function (visited on 04/08/2023).

[39] "SoftMax activation function: Everything you need to know," InsideAIML. (), [Online]. Available: https://insideaiml.com/blog/SoftMaxActivation-Function-1034 (visited on 04/09/2023).

[40] "Common loss functions in machine learning | built in." (), [Online]. Available: https://builtin.com/machine-learning/common-loss-functions (visited on 04/09/2023).

[41] P. Huilgol. "Getting into deep learning? here are 5 things you should absolutely know," Analytics Vidhya. (Mar. 10, 2020), [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/deep-learning-5-things-to-know/ (visited on 03/27/2023).

[42] "Kaggle: Your home for data science." (), [Online]. Available: https://www.kaggle.com/ (visited on 03/27/2023).

[43] "Python release python 3.7.6," Python.org. (), [Online]. Available: https://www.python.org/downloads/release/python-376/ (visited on 03/27/2023).

[44] "Pandas documentation — pandas 1.5.3 documentation." (), [Online]. Available: https://pandas.pydata.org/docs/ (visited on 03/27/2023).

[45] "API documentation | TensorFlow," TensorFlow. (), [Online]. Available: https://www.tensorflow.org/api_docs (visited on 03/27/2023).

[46] "NumPy 1.21.6 release notes — NumPy v1.24 manual." (), [Online]. Available: https://numpy.org/doc/stable/release/1.21.6-notes.html (visited on 03/27/2023).

[47] "Scikit-learn: Machine learning in python — scikit-learn 1.2.2 documentation." (), [Online]. Available: https://scikit-learn.org/stable/ (visited on 03/27/2023).

[48] "Matplotlib.pyplot — matplotlib 3.5.3 documentation." (), [Online]. Available: https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html (visited on 03/27/2023).

[49] "Sklearn.preprocessing.OneHotEncoder," scikit-learn. (), [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html (visited on 04/02/2023).

[50] "Sklearn.preprocessing.MinMaxScaler," scikit-learn. (), [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (visited on 04/02/2023).

[51] K. Team. "Keras documentation: The sequential model." (), [Online]. Available: https://keras.io/guides/sequential_model/ (visited on 04/03/2023).

[52] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, Apr. 23, 2014. arXiv: 1206.5538[cs]. [Online]. Available: http://arxiv.org/abs/1206.5538 (visited on 04/06/2023).

[53] "How a hacker can infiltrate your network and what can be done about it?" SecureReading. (Oct. 9, 2016), [Online]. Available: https://securereading.com/hacking-hacker-infiltrate-networks/ (visited on 04/07/2023).