الجمهورية الجزائرية الديمقراطية الشعبية

Peoples Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

جامعة غرداية

University of Ghardaia

*Registration N°*
/…../…../…../…../…..

كلية العلوم والتكنولوجيا

Faculty of Science and Technology

قسـم الرياضيات والاعلام الآلي

Department of Mathematics and Computer Science

**End of the study thesis, presented to obtain the diploma**

**Master**

**Domain: Mathematics and Computer Science**
**Sector: Computer Science**
**Speciality: Intelligent Systems for Knowledge Extraction**

**Theme**

# A deep learning approach for network intrusion detection system

**Presented by**

**Mohammed abdelmalek HACINI**

**Defended publicly on 19/06/2022**

**Jury members**

| | | | |
|---|---|---|---|
| **M. Ahmed SAIDI** | **MCB** | **Univ.Ghardaia** | **President** |
| **M. Slimane BELLAOUAR** | **MCA** | **Univ.Ghardaia** | **Examiner** |
| **M. Slimane OULADNAOUI** | **MCB** | **Univ.Ghardaia** | **Supervisor** |

**University Year: 2021/2022**

## Acknowledgement

First of all, I thank my Almighty God, who has guided me in my steps and given me the bravery, the will, and the fortitude to endure the challenges I have met throughout my life.

I want to appreciate and express my great gratitude and sincere thanks to our respected professor M.OULADNAOUI Slimane for support for his encouragement, advice and efforts to complete this work.

We are also glad to thank everyone who advised, guided, or contributed to producing this research.

All words remain feeble to express my heartfelt thanks to all those who have supported me from near or far to accomplish this effort, especially all my friends, for their moral support and presence at my side.

## *Dedicated*

*In the first place, I dedicate this work to the one who has always given herself and sacrificed herself for me, the one who helped me as best she could to succeed, the one who has always been there in my moments of anguish, my darling mother.*

*To my dear father for his presence in my life, his support and all his sacrifices and precious advice.*

*To my dear brothers and sisters for their constant encouragement and moral support.*

*Finally, I dedicate it to all my friends and anybody who supports me and encourages me from near or far throughout my education. I express my deepest feelings and dedicate my humble work to them.*

*Mohammed Abdelmalek HACINI*

# ملخص

يعتبر أمن المعلومات من المجالات الأكثر أهمية والمطلوبة بسبب زيادة عدد الهجمات الإلكترونية على الأفراد والمؤسسات. في الواقع ، تم تطوير العديد من أنظمة الحماية. ومع ذلك ، فقد أثر التطوير المستمر للأساليب المستخدمة في الهجمات الإلكترونية على كفاءة عملهم. ومن هنا تأتي الحاجة إلى بناء أنظمة حماية ذكية لاكتشاف وتحليل التهديدات الإلكترونية بشكل استباقي. توفر هذه الأنظمة رؤى عملية لمحللي أمن الشبكات لاتخاذ قرارات تتميز بالسرعة والدقة المطلوبة. استنادًا على نهج التعلم العميق الذي أثبت مؤخرًا كفاءته في العديد من المجالات ، نقترح في هذا العمل نموذجًا لنظام كشف التسلل الشبكي يعتمد على إطار التعلم الذاتي ($Self - Taught\, Learning$) وبنية التشفير التلقائي التلافيفي. نستخدم مجموعة بيانات NSL-KDD لتدريب نموذجنا وتقييمه. أظهرت النتائج التجريبية أن النموذج المقترح يبلغ دقة جيدة 78.41 % لكل من التصنيف الثنائي والمتعدد.

**كلمات مفتاحية : أمن الشبكات، نظام كشف التسلل الشبكي، التعلم العميق، التعلم الذاتي، بنية التشفير التلقائي التلافيفي، مجموعة البيانات NSL-KDD**

# Abstract

Information security is considered one of the most important and in-demand areas due to the increase in the number of cyberattacks on individuals and organizations. In fact, many protection systems have been developed. However, the continuous development of methods used in cyberattacks has affected their work efficiency. Hence, comes the need for building intelligent protection systems to proactively detect and analyze cyber threats. Those systems provide practical insights for network security analysts to make decisions characterized by required speed and accuracy. Based on a deep learning approach that has recently proven its efficiency in many areas, we propose in this work, a network intrusion detection model based on the Self-Taught Learning (STL) framework and the Convolutional Autoencoder architecture. We use the NSL-KDD dataset to train and evaluate our model. The experimental results show that the proposed model reaches a fairly good accuracy of **78.41**% for both binary and multiclass classification.

**Keywords :** Network security, Network intrusion detection, Deep learning, Self-taught Learning, Convolutional autoencoders, NSL-KDD dataset.

# Résumé

La sécurité de l'information est considérée comme l'un des domaines les plus importants et les plus demandés en raison de l'augmentation du nombre de cyberattaques contre les individus et les organisations. En effet, de nombreux systèmes de protection ont été développés. Cependant, le développement continu des méthodes utilisées dans les cyberattaques a affecté leur efficacité de travail. D'où la nécessité de créer des systèmes de protection intelligents pour détecter et analyser de manière proactive les cybermenaces. Ces systèmes fournissent des informations pratiques aux analystes de la sécurité réseau pour prendre des décisions caractérisées par la rapidité et la précision requises. Basé sur une approche d'apprentissage profond qui a récemment prouvé son efficacité dans de nombreux domaines, nous proposons dans ce travail, un modèle de détection d'intrusion réseau basé sur le framework Self-Taught Learning (STL) et l'architecture dautoencodeurs convolutifs. Nous utilisons le jeu de données NSL-KDD pour entrainer et tester notre modèle. Les résultats expérimentaux montrent que le modèle proposé atteint une assez bonne précision de **78.41**% pour la classification binaire et multiple.

**Mots clés :** Sécurité des réseaux, Détection d'intrusion réseau, Apprentissage en profondeur, Self-taught Learning, Autoencodeurs convolutifs, Ensemble de données NSL-KDD.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | **A**rtificial **I**ntelligence |
| CNN | **C**onvolutional **N**eural **N**etwork |
| DAE | **D**eep **A**uto**E**ncoder |
| DNN | **D**eep **N**eural **N**etwork |
| DoS | **D**enial **o**f **S**ervice |
| GRU | **G**ated **R**ecurrent **U**nits |
| HIDS | **H**ost **I**ntrusion **D**etection **S**ystem |
| IDS | **I**ntrusion **D**etection **S**ystem |
| IP | **I**nternet **P**rotocol |
| LSTM | **L**ong **S**hort **T**erm **M**emory |
| MAC | **M**edia **A**ccess **C**ontrol |
| NIDS | **N**etwork **I**ntrusion **D**etection **S**ystem |
| PCA | **P**rincipal **C**omponent **A**nalysis |
| RNN | **R**ecurrent **N**eural **N**etwork |

# INTRODUCTION

The constant growth of computer systems has led to the increasing dependency of enterprises, organizations, and individuals on computer networks in performing their tasks and providing their services in modern ways. Computer networks have become more vulnerable to attacks, which exposes them to numerous important threats, especially in recent years. Although there are different systems to protect networks from these threats, such as: firewalls, user authentication, and data encryption, these systems have not been able to provide full protection for networks and their systems against threats that are vulnerable to them and that become complex over time. Therefore, there is a need to make extensive use of intrusion detection systems (IDS) to be the second line of defense for computer network systems together with other network security strategies.

Typical intrusion detection systems are based on stored signature patterns for known intrusions to be used to identify malicious or suspicious activities in the information system [28]. This type of approach is called signature-based (also known as misuse or pattern-based). The main disadvantage of this approach is its inability to detect new attacks (signatures not yet known). This led to another approach which is called anomaly detection (also named behavior-based) techniques [28]. The latter can detect both novel and known attacks if they demonstrate large differences from the normal profile. Since anomaly detection techniques signal all anomalies as intrusions, false alarms are expected when anomalies are caused by behavioral irregularity instead of intrusions. Therefore, pattern recognition and anomaly detection techniques are often used in combination to complement each one another. Thus, the building of a highly efficient intrusion detection system to detect these penetrations of different kinds is one of the research areas that are very much important in the field of network security. However, the search for features relevant to attacks and possibly destructive behaviors from data exchanged over the network is not an easy operation.

Intrusion detection is one of the most important topics in the area of protecting the security of networks, organizations, and individuals. Today, there are many approaches used in this area, and there are many intrusion detection systems available, but unfortunately, none of them are without defects yet [52]. Therefore, there was a need for continued research into intrusion detection systems in view to a new propose experiment structures with a good protection rate. This is what motivates us to begin this research, which aims to design and develop a network IDS model based on deep learning approach.

The main objective of this research is to develop and experiment a network IDS based on anomaly detection using the deep learning approach, which has given a promising result in several other fields. The proposed system uses the NSL-KDD dataset [31] (released in 2009), which is a dataset of TCP/IP connections, and which has two types of data. The training data which is used for the development of intrusion detection model, and test data which is used to evaluate the performance of the system developed. Our contribution is to propose and implement a network intrusion detection system on the Self-taught Learning (STL) framework [37] and the Convolutional Auto-encoder architecture. We provide python implementation and experimental results with an extensive evaluation and discussion. The preliminary evaluation shows an encouraging result, with a fairly good precision to classify intrusion from normal traffic.

This document comprises three chapters, organized as follows:

- **Chapter 1 :** The opening chapter is devoted to elementary notions. It is divided into three sections: The first represents the different aspects of computer security and the concepts relating to networks in general. The second describes intrusion detection systems, their different types, working principles, advantages and disadvantages. Finally, we introduce the basic concepts of machine learning, particularly deep learning.
- **Chapter 2 :** A review of some studies related to network intrusion detection systems based on deep learning approaches is given in the second chapter.
- **Chapter 3 :** In the last chapter, we will discuss the process of the development and the implementation of our proposed model and how they might be applied on NSL-KDD datasets. We will also present a study of the result and a brief discussion of those outcomes, with a simple comparison.

# Chapter1

# BACKGROUND

## 1.1 Introduction

In this chapter, we present various basic concepts that are important to facilitate the understanding of the rest of the chapters. It is divided into three sections: the first presents the basic concepts related to information security, especially those related to network security, and the second deals with some kind of detail about intrusion detection systems and their various classifications, and the last section includes a review of deep learning and some model architectures such as convolutional neural networks, recurrent neural networks, and autoencoders.

## 1.2 Security Basics

With the increasing growth of network technology and uses for the exchange of information. We hear daily about attacks on information systems, computers of all kinds, and mobile devices in organizations, banks, universities, schools, and individuals. The most known examples of attacks are denial of service, identity theft, malware, etc.

The need to defend against these attacks has given birth to a new discipline within information technology known as information security, which focuses on the protection of digital information for organisations and individuals.

### 1.2.1 Terminology

This section provides background information about the main attacks that currently exist against information systems, particularly network attacks, denial of service, and data attacks. we begin with the description of some frequently used terminologies in the area of computer security and then describe the different techniques used to carry out the various attacks and we will discuss the basic principles in the subject of information security and networks in general, which help as to understand the various attacks and strategies to avoid, reduce and increase protection.

## Information system

The information system is an integrated set of components, technological, organizational and human means to acquire, process, store and communicate information within an organization or between different organizations [39].

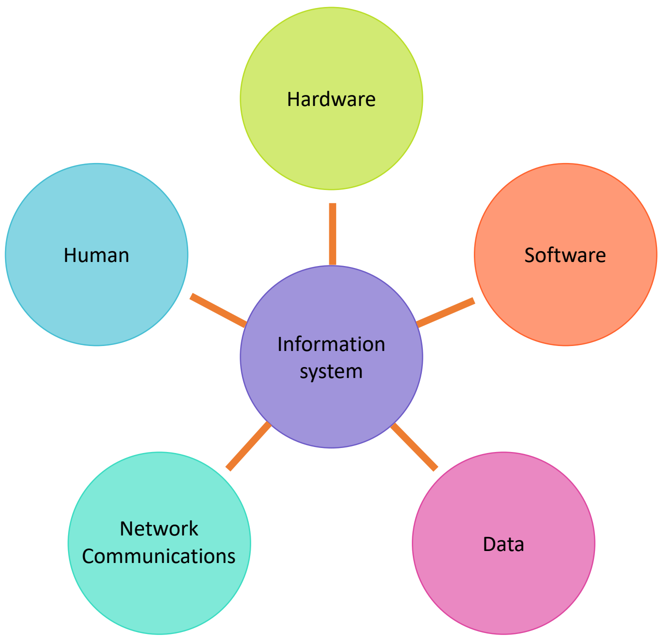


Figure 1.1: The Components of Information Systems

## Information system security

The concept of information system security covers a set of methods, techniques and tools responsible for protecting the resources of an information system in order to ensure the availability of services, the confidentiality and the integrity of information.

Ordinarily, it is described as the protection of information and company against purposeful or accidental activities causing damage to its owners or users. Information security should be focused first of all on eliminating risks, and not on minimizing the adverse impacts of accidents [26].

## Cyber security

The terms cybersecurity and information security are often used interchangeably, but they are two different concepts. Cybersecurity refers specifically to protecting computer systems from unauthorized access. On the other hand, information security is a broader category that encompasses all information assets, whether they are in hard copy or digital form [15]. There are two primary types of cybersecurity threats:

1. External threats: These are threats that originate from outside of an organization and include viruses, malware, and hackers [15].
2. Internal threats: These are threats that originate from within an organization and include disgruntled employees, careless employees or traitor [15].

## Vulnerability

A vulnerability is a weakness or error in a system or a device. This can leave the system open to attack or unauthorized access. Vulnerabilities can be caused by coding errors, outdated software, or simply a lack of security measures. when exploited, can compromise the confidentiality, availability, and integrity of data stored in them through unauthorized access, the elevation of privileges, or denial of service. A code or tool used to take advantage of a vulnerability is called an exploit [15].

## Risk

A risk is a quantifiable likelihood of a threat taking advantage of a vulnerability in a system, or the probability that a threat will exploit a vulnerability.This probability can be difficult to quantify, as it depends on a variety of factors, including the nature of the vulnerability and the sophistication of the threat [15].

### Threat

Something that is a source of danger; capabilities, intentions, and attack methods of adversaries that can exploit or cause harm to a system [15].

### Exploit

An exploit is an attack that takes advantage of vulnerabilities in an operating system, network, or hardware. Frequently, exploits take the form of software or code that aims to gain access to systems or collect network data [15].

## 1.2.2 Security Properties

When it comes to information security, the three main principles are confidentiality, integrity, and availability. These principles are often referred to as CIA. Each one of these principles is important, and must be protected in order to maintain a secure system. Some other properties are sometimes included. Because different applications will have different requirements, a system may be designed to maintain all of these properties or only a chosen subset as needed, as described below [29].

All security controls, mechanisms and safeguards are set up to ensure one or more of these protection concepts. Hackers who organize attacks on systems exploit a combination of vulnerabilities to try to destroy these three principles. Figure 1.2 illustrates the security principles.

### Confidentiality

Confidentiality is one of the most critical security properties of a system. It specifies the only authorized entities that are allowed to access information. This property is necessary to maintain the secrecy of information [29].

There are a variety of mechanisms to ensure confidentiality, the most common are access control and encryption. Access control mechanisms prevent unauthorized entities from reading information until they prove that they are authorized. Encryption does not prevent access to information, but instead makes it challenging to read and understand.

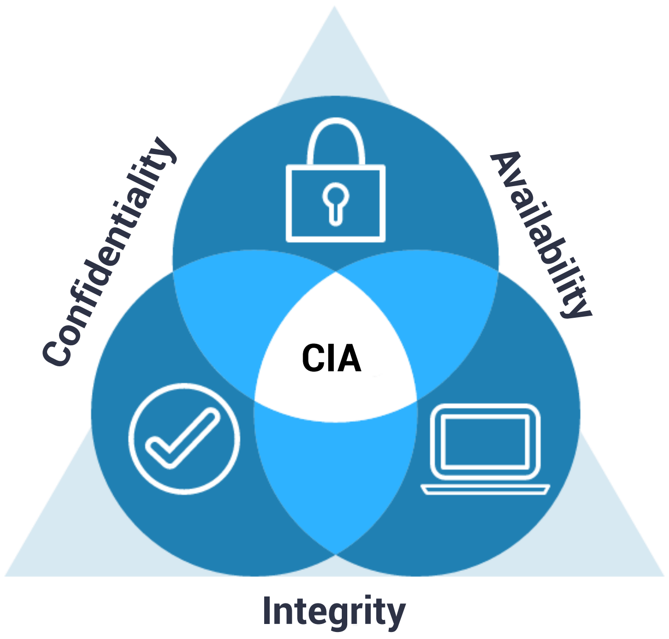


Figure 1.2: The security principles [15]

## Integrity

Information assurance is an essential part of protecting organization's data. One of the essential aspects of information assurance is data integrity - ensuring that only authorized entities can alter information within a system [29]. This property keeps information from being changed when it should not, which is essential for maintaining data reliability. When data integrity is violated, it can have severe consequences for an organization. It can impact data reliability, cause financial losses, and even put people's safety at risk. Therefore, it is essential to ensure data integrity.

Various mechanisms exist to support data integrity and detect when it has been violated. In practice, these mechanisms are similar to the access control mechanisms for confidentiality. Detecting integrity violations may be revealed by:

- Tracking data changes.
- Verifying the source of data.
- Checking the consistency across systems.

#### Availability

In computing, availability is the property that the information on a system is obtainable when needed. Information that is kept secret and unaltered might still be made unavailable by attackers conducting denial of service attacks [29].

#### Non-repudiation

Nonrepudiation provides an assurance that the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity. Further, this concept can apply to any activity, not just the sending and receiving of data. In a more general sense, it is a mechanism to prove that an activity was performed and by whom. Nonrepudiation is typically comprised of authentication, auditing/logging, and cryptography services [53].

#### Authentication

Authentication is the mechanism to verify the identity of the users, process, or device, often as a prerequisite to allowing access to resources in an information system [53].

### 1.2.3 Network Attacks

A computer network is a system in which two or more devices are connected to each other to share resources. Networks can be used for a variety of purposes, including sharing files and folders, printing documents, and accessing the internet. In order to access resources on a network, users must first be authorized to do so. However, networks are also vulnerable to attacks, and attackers can exploit vulnerabilities in order to gain unauthorized access to resources or to cause malicious activity. In this part, we will highlight some of the most common types of network attacks.

#### 1.2.3.1 Denial of Service

A Denial of Service (DoS) attack is one of the main threats to the availability of systems and services. It is designed to prevent a system or service from normally functioning by exploiting known vulnerabilities in the operating system, applications, or even network protocols. Through this type of attack, the attacker prevents authorized users from accessing the service by sending a flood of fake requests to the target system, confusing it and preventing it from responding to legitimate requests [9].

Among the most famous methods used in denial of service attacks is SYN flooding, where the attacker sends many fake connection requests to the target system, which in turn will respond to these requests and then wait for the third stage of the handshake process since the requests are fake (carrying an IP Does not exist at all). The target system keeps waiting for responses that never come, as shown in Figure 1.3.

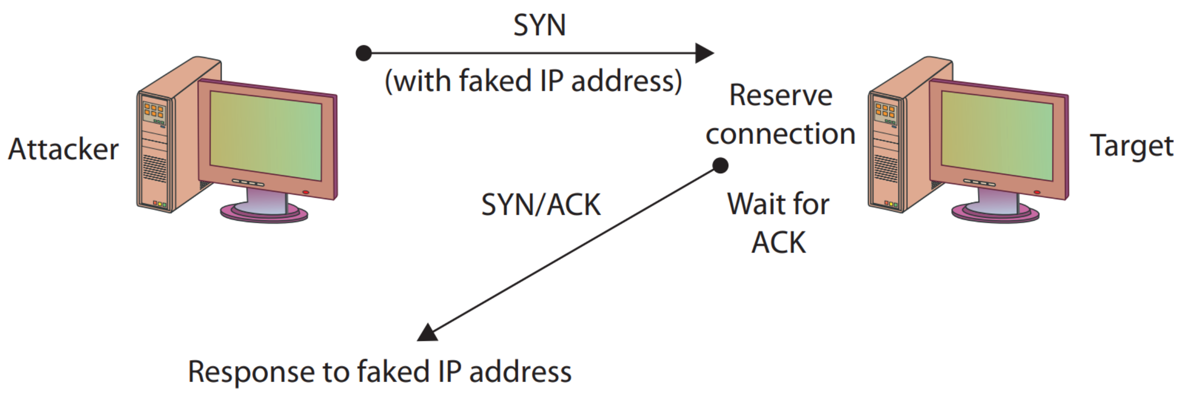


Figure 1.3: A SYN flooding based DoS attack [9]

#### 1.2.3.2 User to Root (U2R)

A user to root attack is among the most dangerous attacks on computer networks because it allows the attacker to illegally gain the privileges of the root user, making him gain complete control over the system. Initially, the attacker connects as a regular user (possibly acquired by password sniffing, dictionary attack, or social engineering) and then obtains root access by exploiting vulnerabilities in software and the system itself [51].

One of the most common ways to get superuser permissions is through a buffer overflow exploit. A buffer overflow occurs when a program copies a lot of data into a fixed buffer without checking the fit of the data. If the data is too large for the buffer, it can flow into adjacent memory locations, potentially allowing an attacker to execute arbitrary code or take control of the system.

#### 1.2.3.3 Remote to Local (R2L)

Remote to Local (R2L) attack is a class of attack whose goal is unauthorized access from a remote machine. An attacker sends packets to a machine over a network and then exploits a vulnerability to gain local access as a user of that machine [51].

There are many possible ways an attacker can gain unauthorized access to a local account on the device. Dictionary, Ftp-Write, Guest, and Xsnoop attacks try to exploit weak or faulty system security policies. The Xlock attack involves social engineering. For the attack to succeed, the attacker must successfully spoof a human factor to enter their password into a screensaver that is a Trojan horse.

#### 1.2.3.4 Probing

Targeted attacks rely on prior knowledge of network systems vulnerabilities. Probing is a class of attacks whose purpose is to gather all information about network systems so that the attacker scans a network to obtain a map of the network and available services. Port scanning is an essential component of network security audits, and attackers can also use it to find and exploit vulnerable systems [51].

In addition, examining the ports can tell us a lot about the network, such as the current hosts, the IP and MAC addresses in use, and the filtering rules in place.

### 1.2.4 Network Security Techniques

Network security is the process of protecting a computer network from unauthorized access, theft, or damage. Security policies define the permissions to use network components and resources. To build an effective network protection strategy, all potential security threats must be identified, and then the most effective set of tools to combat them must be selected. We will discuss some of the common techniques in the following section.

#### 1.2.4.1 Virtual private network (VPN)

A VPN is a technology that lets users and servers securely connect to a private network over the Internet. It allows the creation of a computer network inside the internet or a private network across a public Internet network. This technology is used extensively in corporate environments to allow employees to connect to the corporate network remotely [15]. The encrypted connection helps ensure that sensitive data is safely transmitted, and it prevents unauthorized people from eavesdropping on the traffic. It also enables the user to work securely. Tunneling protocols are used to connect distant computers and servers into one network. These protocols create figurative tunnels to transfer information between a remote server and a terminal through different networks. This concept is illustrated by 1.4.

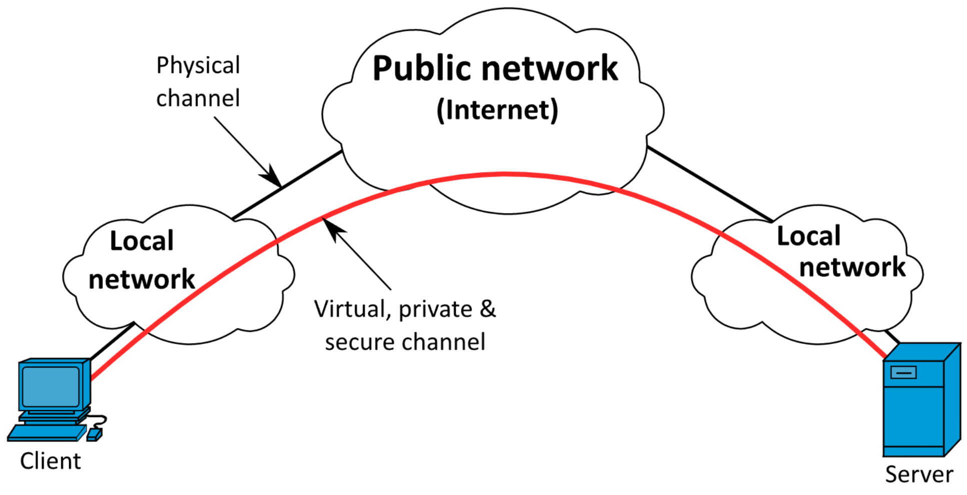


Figure 1.4: VPN Connection [15]

### 1.2.4.2 Firewalls

A firewall is a network security system, it can be either software or hardware based, that is used to enforce a security policy on network connections. By allowing or denying traffic to pass into or out of the network. A firewall can protect a network from unauthorized access and attacks. It resembles a gate guard in a secure facility. The guard examines all the traffic trying to enter the facility, cars with the correct sticker or delivery trucks with the appropriate paperwork are allowed in [9].

Firewalls can be very effective in protecting a network, but they are unable to recognize attacks. They simply block all traffic, except for packets that comply with certain rules. These rules are manually set by network administrators and can vary depending on the security policy of the network. So, the effectiveness of the firewall depends on the skill of the administrator in setting the rules.

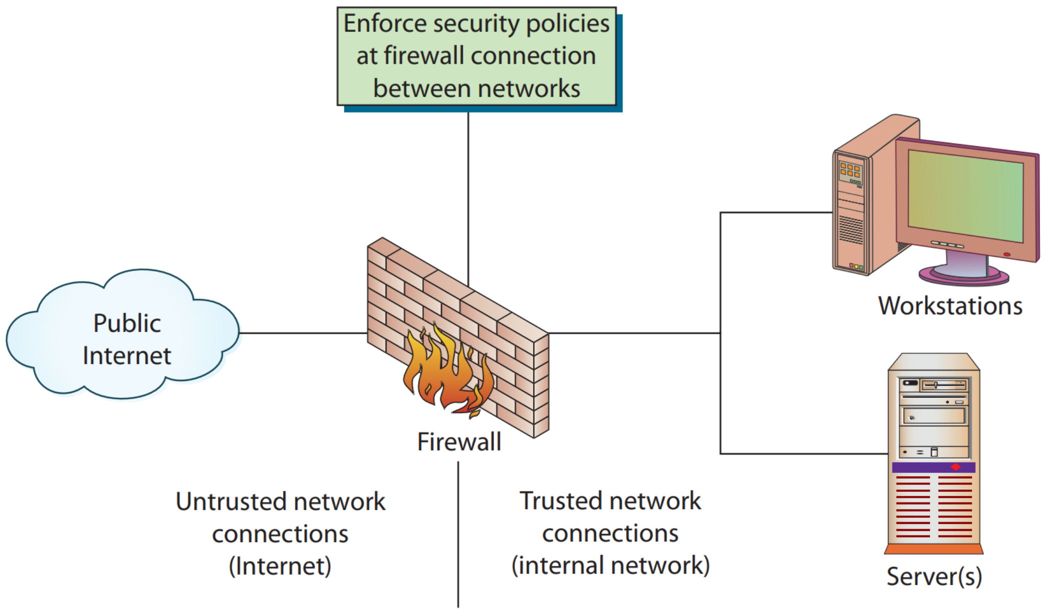

Enforce security policies at firewall connection between networks
Public Internet
Firewall
Workstations
Server(s)
Untrusted network connections (Internet)
Trusted network connections (internal network)

Figure 1.5: How a firewall works [9]

The security policy defines which network devices will operate at specific points within the network. Communications to the Internet pass through the firewall, as shown in the figure 1.5. This firewall will block all traffic except that authorized by the security policy. For example, blocking traffic on a port only tells the firewall that the port is closed. There are many firewalls, such as pfSense [36], OPNSense [33], ClearOS [8] etc.

## 1.3 Intrusion Detection Systems

Firewalls play an important role in network security, but they are not sufficient to protect systems from all possible attacks. One reason is that unauthorized online activities are not only carried out by external attackers, but also by internal sources, such as fraudulent employees or people who abuse their privileges for personal gain or revenge. This type of activity can often go undetected by a firewall. Additionally, firewalls work by only allowing traffic that is predetermined as legitimate, but they do not examine the contents of that traffic. This could allow malicious or unauthorized traffic to pass through the firewall undetected. Another reason that firewalls are not sufficient to protect systems is that they cannot prevent all kinds of attacks. Standard network security solutions with a firewall were not designed to handle network and application layer attacks such as denial of service, worms, viruses, and trojans horses. These attacks can be very damaging and can leave a system vulnerable to further attack. These reasons, along with the increasing prevalence of online threats, have led to the use of intrusion detection system.

### 1.3.1 Definition

An Intrusion Detection System (IDS) is a security system that monitors, using traffic analysis, the set of events occurring in an information system and looks for signs of malicious activity. Intrusions can be caused by attacks from outsiders (such as hackers), or by authorized users trying to gain additional privileges or access to information that they should not have. IDS systems can be based on either signature (matching known patterns of malicious activity) or anomalies (detecting unexpected or unusual behaviour that may be indicative of an attack) [9].

### 1.3.2 Basic architecture of an IDS

Many IDSs have been proposed in both the commercial and research areas since Dorothy Denning produced the first model of intrusion detection at SRI International [41]. Despite the wide range of methodologies used to collect and analyze data by these systems, the majority of them rely on a common architectural framework (Figure 1.6), which includes the following components:

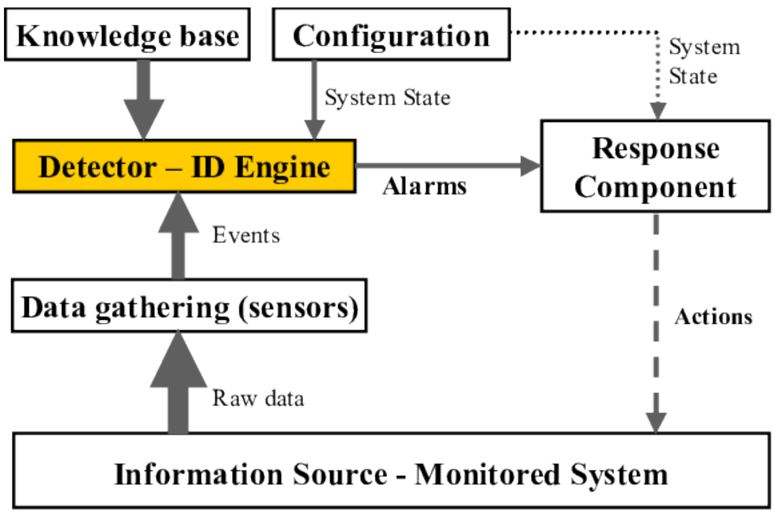


Figure 1.6: Basic architecture of an intrusion detection system (IDS) [41]

- **Data gathering device (sensor):** is responsible for collecting and filtering data from the monitored system and send it to the detection engine.
- **Intrusion Detection Engine:** its role is the matching of collected data from sensors with the corresponding in knowledge base to identify intrusive activities.

- **Knowledge base:** it usually contains a set of rule or signature for different attacks or information about them. The knowledge base is usually provided by security experts or some techniques like statistical measures.
- **Configuration:** this device provides information about the current use state of the IDS in which we can apply security policy.
- **Response component:** passive or active measures (can either be automated or involve human interaction) taken in response to the detection of an attack, to stop it or to correct its effects.

### 1.3.3 Position of the IDS in the network

Intrusion detection systems must be positioned in strategic places to see network traffic to analyze. Most companies and organizations support Network IDS in addition to firewalls. It is important to consider the position of the IDS concerning the firewall [9].

The intrusion detection system can be located between the firewall and the internal network in order to detect the intrusion passed by the firewall as shown in Figure 1.7, or between the servers and the user group for the detection of internal intrusions, or before the firewall as shown in Figure 1.8

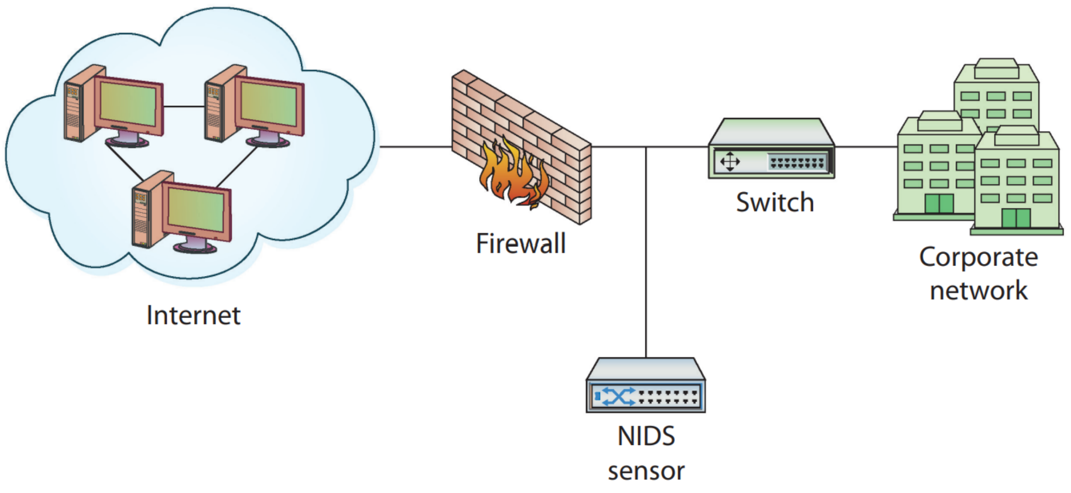


Figure 1.7: NIDS sensor placed behind firewall [9]

### 1.3.4 Classification of IDS

Several ways to classify IDS are described using different analysis and control methods. Each of them has advantages and disadvantages, and they are chosen according to the needs of the system and the purpose it is supposed to achieve [41].

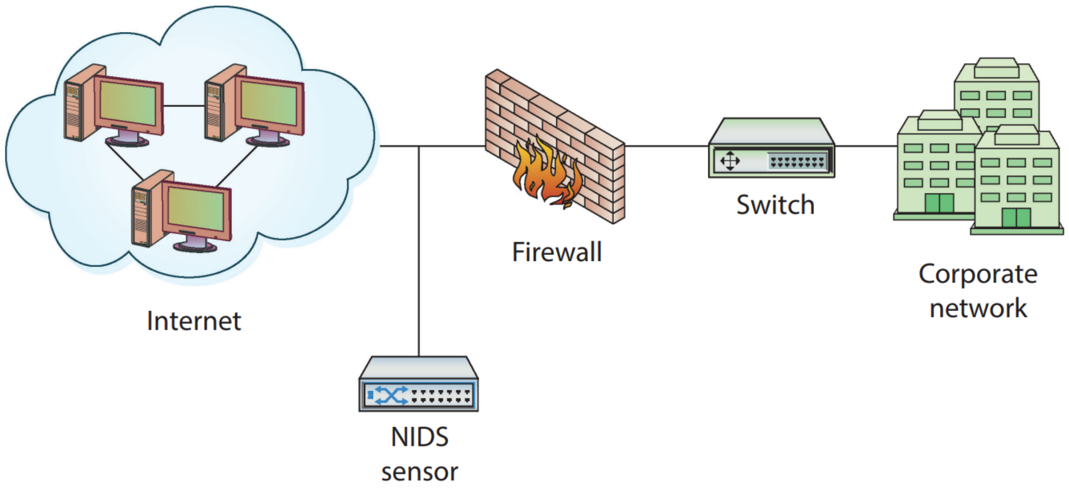


Figure 1.8: NIDS sensor placed in front of firewall [9]

### 1.3.4.1 Classification by source of data

Intrusion detection systems are divided into three groups based on the data source to be analyzed: host-based IDS, network-based IDS and hybrid IDS.

- **Host-based Intrusion Detection System HIDS**

  Host intrusion detection systems (HIDS) are systems that monitor individual hosts or devices in a network for malicious or unauthorized activity. HIDS monitors all inbound and outbound packets of the device and alerts the administrator of any suspicious activities. It can also take snapshots of system files at certain intervals and compare them to see if any critical files were modified or deleted. This allows the administrator to be alerted when required [4]. Figure 1.9 shows the positioning of a HIDS on the network.

  The great benefit of HIDS is that it provides real-time alerts to changes in the system, which can help identify an attack when it is happening. Additionally, HIDS can help identify the source of an attack, which can be very helpful in tracking down and prosecuting the perpetrators. HIDS is not a perfect solution, but it is a valuable tool in the arsenal against system attacks. There are many HIDS, such as SolarWinds Security Event Manager [44], OSSEC [34] etc.

- **Network-based Intrusion Detection System NIDS**

  It is the most common form of an intrusion detection system. These systems detect attacks by listening to network segments or switches by capturing and analyzing network packets. Thus, it can monitor all packets traveling between a group of computers connected to the network by matching one or more packages with the database of signatures for attacks or analyzing the traffic to detect anomalies responsible to incoming

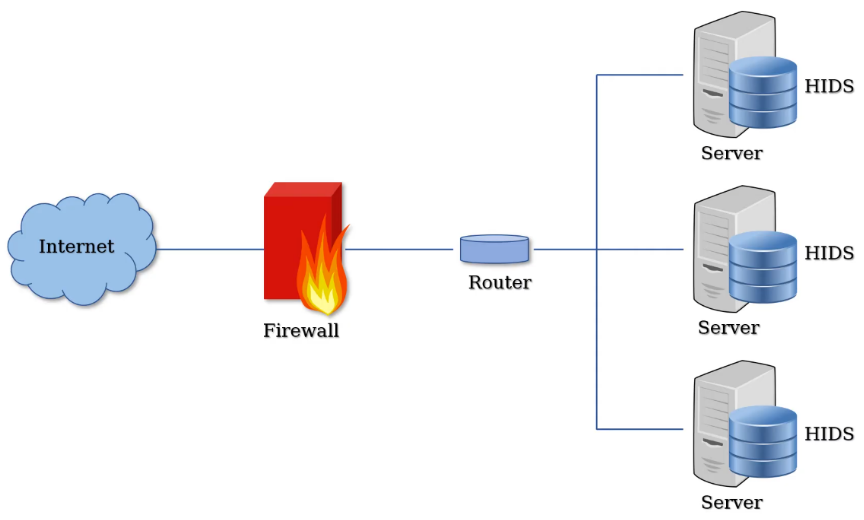


Figure 1.9: HIDS location [4]

packets that may circumvent the firewall [4]. There are many examples of network intrusion detection system such as Snort [47], Suricata [22] etc.

Figure 1.10 shows a network that uses three NIDS units that are placed on strategic network segments so that they can monitor network traffic for all devices on the segment. This configuration represents a standard network topology where subnets comprising public servers are protected by NIDSs. When a public server within a subnet is compromised, the server can become a platform for launching additional vulnerabilities, so it is necessary to ensure careful monitoring to prevent further damage.

- **Hybrid Intrusion Detection**

  Both network-based and host-based intrusion detection systems have their own strengths and benefits. Network-based intrusion detection systems are good at detecting attacks that are targeting the network infrastructure, such as denial-of-service attacks or scanning for vulnerable systems. Host-based intrusion detection systems are good at detecting attacks that are targeting the hosts on the network, such as malware infections or brute-force password attacks.

  Hybrid intrusion detection system is made by combining two or more IDS approaches, with the aim of improving the network's resistance to attacks and misuse, in addition to promoting pain policy and ensuring greater flexibility in application and deployment options. By combining the strengths of each type of intrusion detection system, we can create a more effective and comprehensive intrusion detection system.

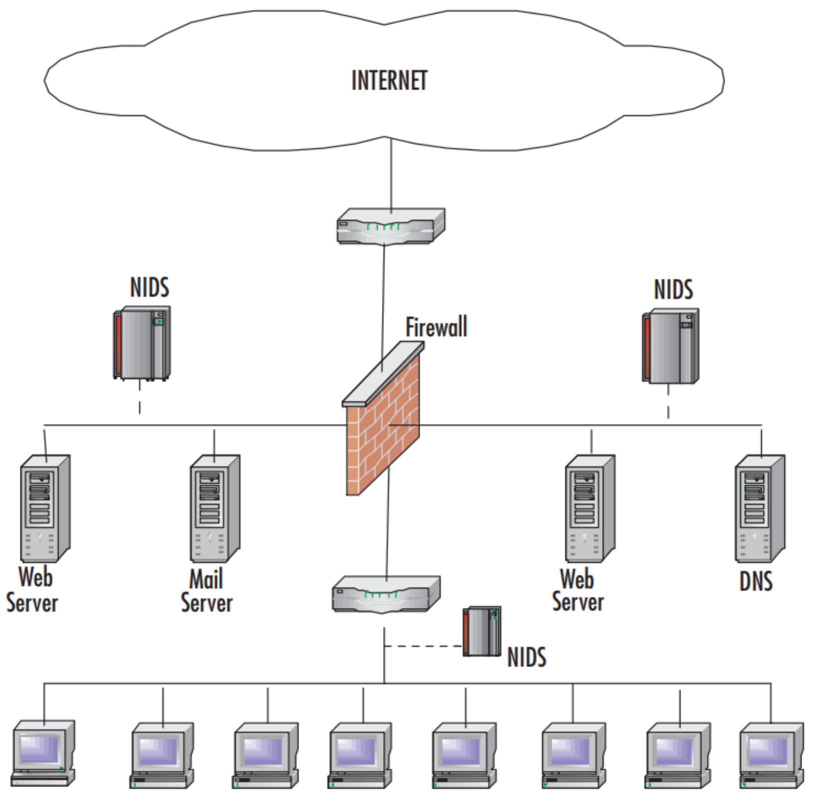


Figure 1.10: NIDS Network [4]

#### 1.3.4.2 Classification by Intrusion detection approaches

Detection methods are the basis of intrusion detection techniques, and are the engine that recognize the malicious activities of a source data. Detection methods analyze the data they monitor and trigger alerts if malicious traffic is perceived. Accordingly, intrusion detection systems can be categorized according to the detection methods used into anomaly based intrusion detection systems, and signature based intrusion detection systems [4].

- **Signature-based Intrusion Detection System**

  Signature-based IDS is a detection technique that uses specific patterns, called signatures, to identify attacks. These signatures can be byte sequences in network traffic, or known malicious instruction sequences used by malware. This terminology originates from antivirus softwares, which refers to these detected patterns as signatures.

  Although signature-based IDS can easily detect known attacks, it is difficult to detect new attacks, for which no pattern is available. As a signature-based IDS monitors the packets traversing the network, it compares these packets to the database of known attack signatures to flag any suspicious behavior.

  **Advantages of Signature-based IDS**

  - detects threats effectively without creating a high number of false alarms.
  - It is difficult to move on.

**Disadvantages of Signature-based IDS**

  - Discover only already know attacks.
  - It needs constant updating of the signature database with new ones.

- **Anomaly-based Intrusion Detection System**

  It depends on defining what is normal or permissible behavior in the system and then reporting any act or event that falls outside the permissible or normal limits.

  An anomaly-based system operates on the assumption that malicious events are different from normal actions, and therefore the differences are sought to detect the attack. These systems constitute profiles of historical data collected during a period of normal operation. It then collects event data to determine when the monitored activity deviates from normal behavior and triggers an alarm accordingly.

  **Advantages of Anomaly-based IDS :**

  - It is good at detecting new and undiscovered attacks.
  - It does not need constant maintenance.

  **Disadvantages of Anomaly-based IDS**

  - It requires a considerable training sample for distinguishing between normal and abnormal traffic.
  - It generates a large number of false alarms.

## 1.4 Deep Learning

Nowadays, deep learning is at the core of intelligence systems due to the sophistication of machines and the abundance of data, and the good results in various fields such as healthcare, visual recognition, text analytics, and cybersecurity. Deep learning is an artificial intelligence technique derived from machine learning. So it is not possible to talk about deep learning without reminding the principles of machine learning and artificial intelligence.

This section begins with a brief overview of the evolution of deep learning, from its first technologies to deep learning. We will also cover the basics of machine learning, including its different methods and branches. We will also look at artificial neural networks (ANN) and their different types. Finally, we introduce the main core deep learning architectures.

### 1.4.1 History

Machine learning is the most advanced branch of artificial intelligence. We will try to cover the essential stages in the development of intelligent systems (artificial intelligence, machine learning, and deep learning) in the following points.

- **1943 :** In [30], Walter Pitts and Warren McCulloch present a mathematical model of biological neurons. This McCulloch Pitts Neuron is extremely limited in its abilities and lacks a learning mechanism. Nonetheless, it will lay the groundwork for deep learning and artificial neural networks.
- **1950 :** In This year the mathematician Alan Turing proposed what he called the Turing test. The test is designed to determine if a computer has a human-like intelligence [54].
- **1952 :** Researcher Arthur Samuel created an early learning machine capable of learning to play checkers. It used annotated guides by human experts and played against itself to learn to distinguish the right moves from bad ones [32].
- **1957 :** Frank Rosenblatt (psychologist) invented the perceptron, the world's first neural network for computers. It successfully stimulated the human brain's thought processes. This is the origin of today's neural networks [32].
- **1960 :** In [25], Henry J. Kelley presents the first-ever continuous back-propagation model. His model is based on Control Theory, but it lays the groundwork for further refinement and will be used in ANN in the future.
- **1962 :** In [12], Stuart Dreyfus presents a backpropagation model that employs a simple derivative chain rule rather than dynamic programming, which was previously used in backpropagation models. This is yet another small step in the direction of deep learning's future.
- **1965-71 :** Alexey Grigoryevich Ivakhnenko and Valentin Grigoryevich Lapa have developed a hierarchical representation of a neural network that employs a polynomial activation function and is trained using the Group Method of Data Handling (GMDH). It is now widely regarded as the first multi-layer perceptron, and Ivakhnenko is the father of deep learning [3].
- **1979-80 :** An ANN learns how to recognize visual patterns. Kunihiko Fukushima developed the "Neocognitron," a pattern-recognition ANN. It is the first convolutional neural network [14].

- **1982 :** This is the year when John Hopfield created Hopfield Network is nothing more than a recurrent neural network. It is a content-addressable memory system that will be useful for future recurrent neural network (RNN) models in the modern deep learning era [3].
- **1985 :** The appearance of Boltzmann Machine, It is a stochastic RNN created by David H. Ackley, Geoffrey Hinton, and Terrence Sejnowski. This neural network has no output layer, only an input layer, and a hidden layer [1].
- **1986 :** Terry Sejnowski (neuroscientist) proposed a program that could learn how to pronounce English words (NETtalk) [45].
- **1989 :** Yann LeCun trains a convolutional neural network to recognize handwritten digits using backpropagation [3].
- **1991 :** Sepp Hochreiter highlights the vanishing gradient problem, which can make deep neural network training more difficult due to the now-famous disappearing or expanding gradient problem [46].
- **1997 :** Schmidhuber and Hochreiter proposed a recurrent neural network framework called Long Short-Term Memory (LSTM) [20].
- **2006 :** Geoffrey Hinton, Ruslan Salakhutdinov, Osindero, and Teh publish the paper "A fast learning algorithm for deep belief nets" [19].
- **2008 :** Andrew NG's group at Stanford has begun campaigning to use GPUs for Deep Neural Network Training [3].
- **2009-2012 :** ImageNet(2009), created by Stanford professor Fei-Fei Li, and AlexNet (2012) developed by Alex Krizhevsky, [3].
- **2014 :** Ian Goodfellow and colleagues disclose the first operational implementation of a generative model based on adversarial networks (GAN) [3].
- **2016 :** AlphaGo, developed by Google's DeepMind, is the first computer program to defeat a professional human player in the Go game [3].
- **2017 :** Vaswani et al. proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models [55].
- **2020 :** The GPT-3 natural language processing system, developed by Open AI, has the extraordinary capacity to generate human-like writing [6].
- **2021 :** The appearance of Vision Transformer (ViT), Which gives excellent results compared to the latest convolutional networks while requiring less computational resources to train [11].

## 1.4.2 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that provides computers with the ability to learn from data. This contrasts with traditional computing algorithms, which are explicitly programmed by humans and rely on pre-defined rules. The purpose of machine learning is to understand data to build data-driven models and programming through the systematic detection of statistically significant patterns in the available data. In the 1930s, Thomas Ross made the first attempt to create a machine that mimic the behaviour of a living creature [40]. Arthur Samuel in 1959, defined machine learning as a "field of study that allows computers to learn without being explicitly programmed" [42].

### 1.4.2.1 Types of Machine Learning

According to what goal to be achieved by using ML, it can be classified into different types as shown in Figure 1.11.

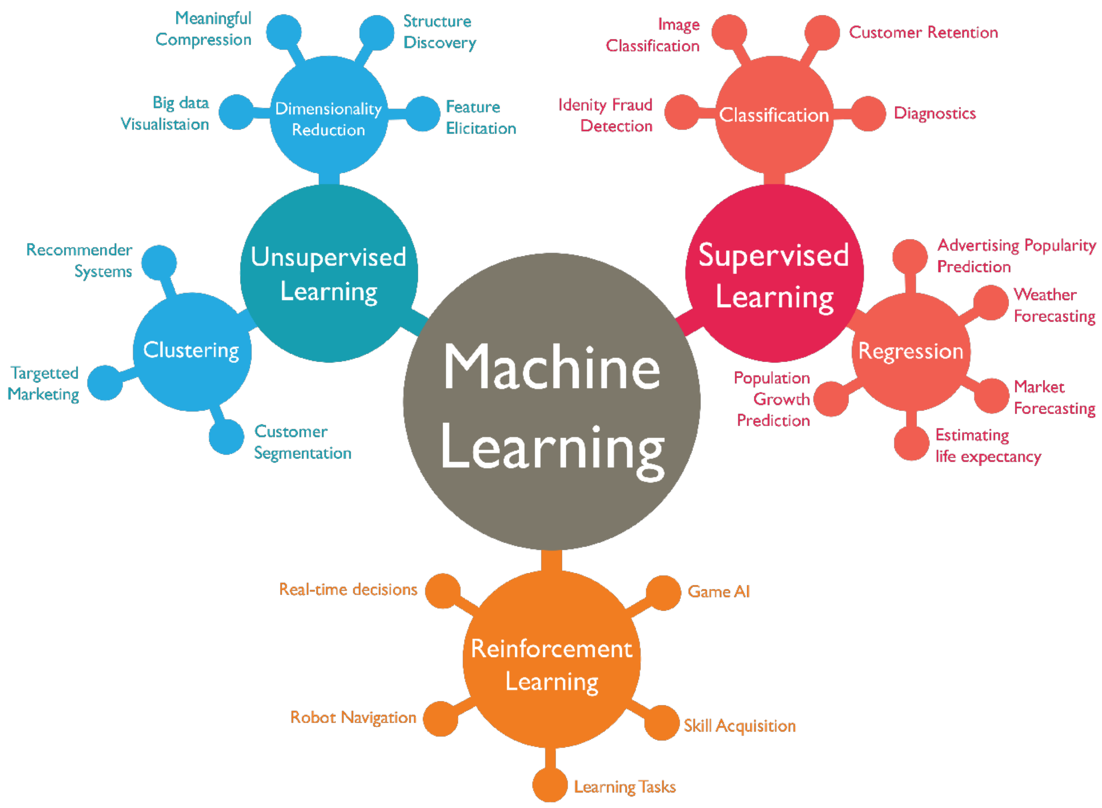


Figure 1.11: Machine learning types and tasks

1. **Supervised Learning :** Supervised learning is a learning technique in which the computer is provided with examples of inputs that have been categorized with desired outputs. The purpose of this method is for the algorithm to be able to learn by comparing its actual output with the studied output to find errors and modify the model accordingly [43]. Supervised learning includes two tasks Regression and Classification.

   (a) **Regression :** We talke about regression when the prediction is continuous values (quantitative), the output takes continuous values, for example predicting house prices. There are many different types of regression, but linear regression is the most common. It is a simple algorithm that finds a straight line that best fits the data. This line can be used to predict future values for the dependent variable.

   (b) **Classification :** Is a data analysis task that involves identifying a model that describes and distinguishes data classes and concepts. The goal is to use this model to predict the class to which a new observation belongs, on the basis of a training set of data. Classification can be used to predict discrete values or classes, in which case the output takes class labels.

2. **Unsupervised Learning :** Unsupervised learning is a technique where the machine uses unlabeled data. In this case, the algorithm learns the internal representation or important features to discover relationships or structures within the input data. The advantage of unsupervised learning is that it can be used to find patterns in data that humans may not easily identify [43]. Many different unsupervised learning algorithms can be used, including clustering and dimensionality reduction.

   (a) **Clustering :** Clustering algorithms can automatically recognize the pattern inside the data by grouping there into clusters so that the data are similar within each cluster and dissimilar from the other. The most famous clustering algorithm is K-means.

   (b) **Dimensionality reduction :** These algorithms reduce the number of dimensions in a dataset while preserving the most important information. This can be useful for reducing the complexity of data and making it easier to analyze. The most well-known algorithm is PCA (Principal Component Analysis).

3. **Reinforcement Learning :** Reinforcement learning is a machine learning algorithm that enables machines to learn how to make decisions. By providing the machine with feedback in the form of rewards or penalties, it can learn how to adapt its approach to achieving the best possible outcome. The most famous reinforcement learning algorithm is Q-Learning, which has been used to train machines in various tasks, including playing video games and controlling robots [43].

#### 1.4.2.2 Machine Learning Applications

Reliance on machine learning systems has increased recently in various fields, and in the following, we mention some applications of machine learning technology:

- Image Recognition (optical character recognition, face detection).
- Sentiment Analysis.
- NLP Natural Language Processing (text generation, translation).
- Anomaly detection (cybersecurity, medicine).
- Speech Recognition.
- Product Recommendations.

### 1.4.3 Artificial Neural Networks

A neural network is a set of an interconnected simple processing units, whose functionality is loosely based on the biological neuron. The behavior of the network is stored in the inter-unit connection weights, obtained by a process of learning from a set of training patterns [2]

An artificial neural network is a layer of nodes (Artificial Neuron) connected between them, consisting of three basic layers:

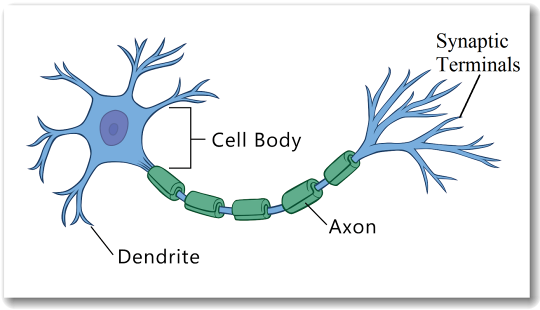
- **Input layer**: it is the first layer responsible for receiving information (data) from the external environment.
- **Hidden (intermediate or invisible) layers**: these layers perform most of the basic work in a network. The layers are made up of neurons responsible for excreting features.
- **Output layer**: after processing with neurons in the previous layers, this layer produces and delivers the final network outputs.

Artificial neurons are a mathematical model inspired by a biological neuron that represents the basic unit of the human brain. This unit is divided into three main parts, as shown in Figure 1.12a.

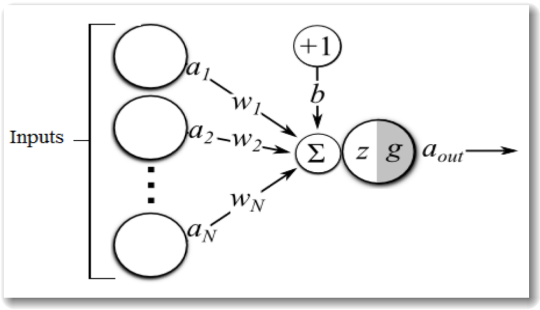
1. **Dendrites**: consist of a group of thin extensions of neurons that typically receive input from many different cell types and which often form synaptic connections.
2. **Cell body** (also known as soma): is the part responsible for producing the activation by processing all the information that comes from the dendrites, and it contains a group of organelles (nucleus, lysosome, centriole,..).
3. **Axon**: its mission is to direct stimulation to other neurons via synaptic terminals.

On the other hand, the artificial neuron shown in figure 1.12b consists of :

1. Inputs: is a vector $(a_1, a_2, \ldots, a_n)$ with weights $(w_1, w_2, \ldots, w_n)$, each input is multiplied by its weight.
2. Linear function z: is a summation function that sums weights after multiplies each of input by their own associated weight, with the addition of the bias b; $z = \sum_1^n a_i w_i + b$
3. Activation function g(z): is mathematical equations that determine the output of an artificial neuron.
4. Output : output the final activation; $a_{out} = g(z)$.



(a) Biological Neuron



(b) Artificial Neuron

Figure 1.12: Biological neuron vs Artificial neuron [2]

### Activation function

Activation functions are used at the end of a hidden unit to introduce non-linear complexities to the model. Figure 1.13 shows some of the most popular activation functions.

A deep neural network is simply an artificial neural network with multiple hidden layers, as shown in figure 1.14.

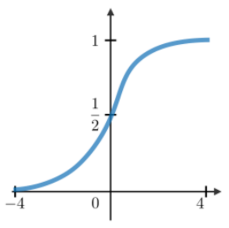
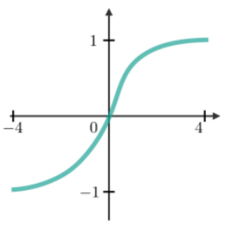
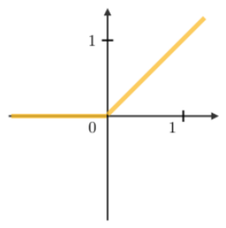
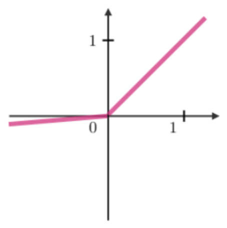
| Sigmoid | Tanh | ReLU | Leaky ReLU |
|---|---|---|---|
| $g(z) = \frac{1}{1+e^{-z}}$ | $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ | $g(z) = \max(0, z)$ | $g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$ |
|  |  |  |  |

Figure 1.13: Examples of some activation functions [2]

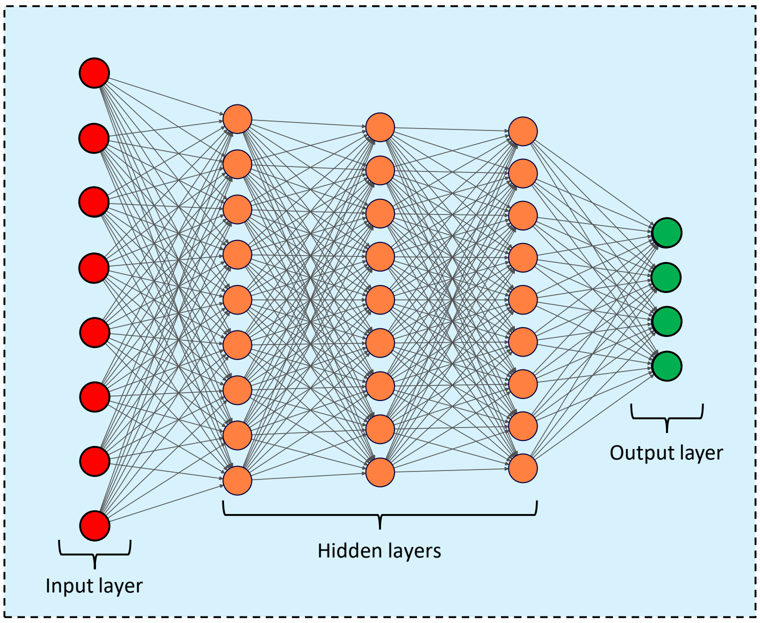


Figure 1.14: Deep Neural Network (DNN) example

## 1.4.4 Convolutional Neural Networks

A convolutional neural network (CNN) is a particular type of feed-forward artificial neural network inspired by the visual cortex and designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become state of the art for many visual applications such as image classification, object detection, and character recognition. And have also found success in natural language processing for text classification. The name "convolution" is derived from a mathematical operation involving the convolution of different functions. The design of a CNN has been divided into layers as shown in figure 1.15.

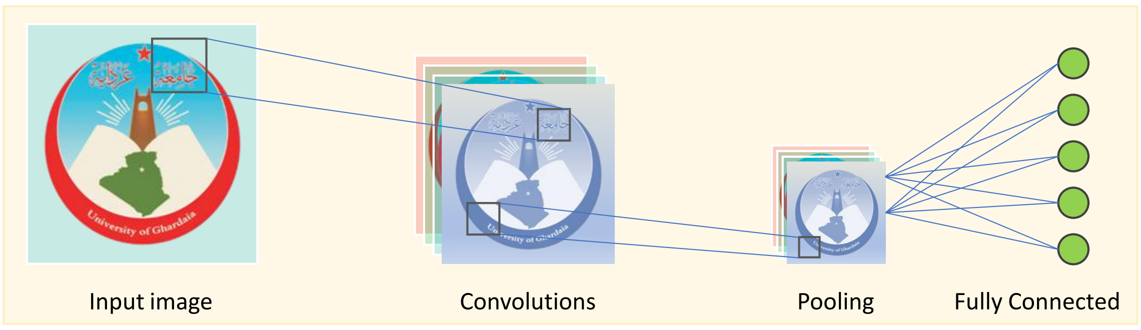


Figure 1.15: CNN Architecture

- **Convolution layer (CONV):** A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image.A convolution operation is applied to produce a features map for each filter on the input data. For convolution, the filter slid across the height and width of the image, and the dot product between every element of the filter and the input is calculated at every spatial position. Figure 1.16 shows an example of the convolution process.

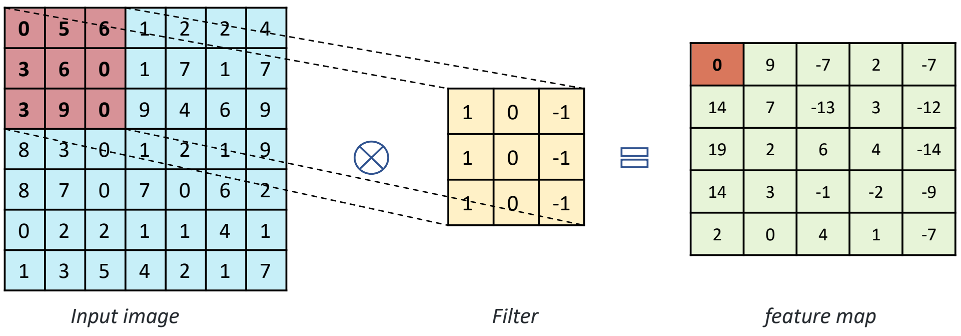


Figure 1.16: Convolution operation

- **Pooling layer (POOL):** The pooling layer reduces the size of the previous layer, typically applied after a convolution layer. There are two common variations of pooling operations: max pooling and average pooling (Figure 1.17).

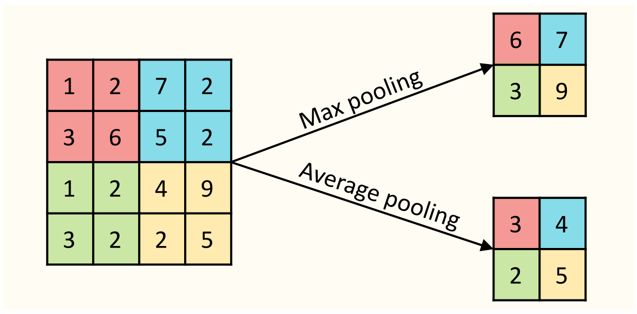


Figure 1.17: Pooling types

- **Flattening Layer :** In this layer, the input size is transformed from shape (width, height, depth) to a one-dimensional array. This is to take advantage of all layer information and be ready to link to the artificial neural network. And it is considered the last step in extracting features.

- **Fully Connected (FC):** It is an artificial neural network that operates on a flattened input where each input is connected to all neurons. FC (Figure 1.18) is the last layer in the CNN architectures and can be used to classify objectives.

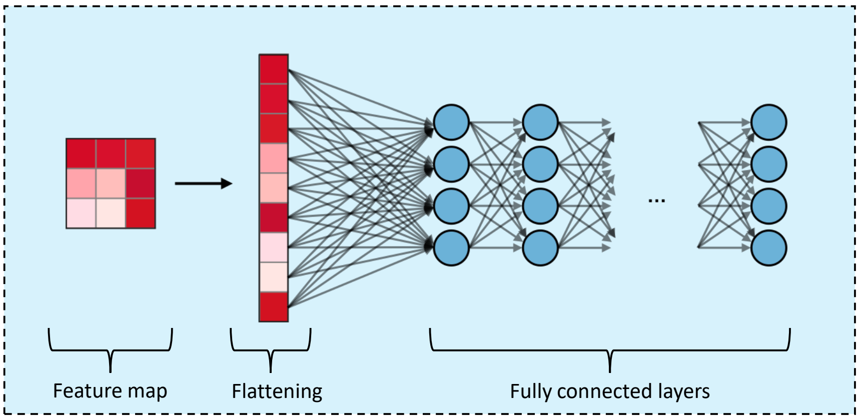
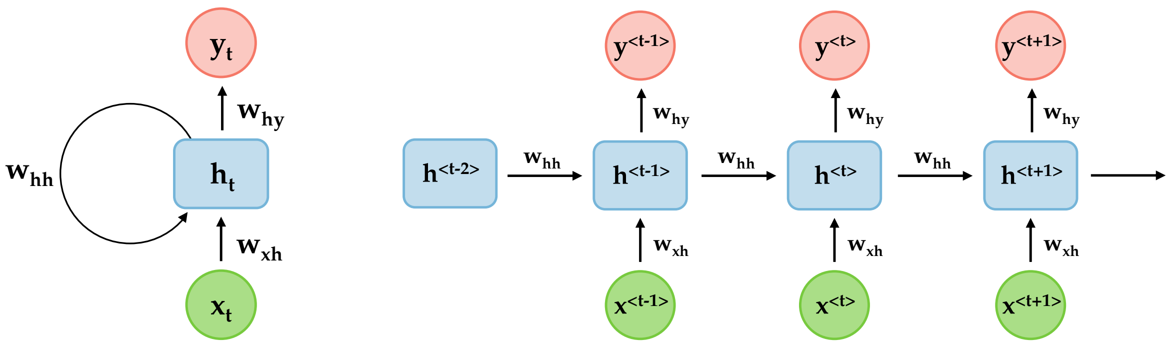


Figure 1.18: Fully connected layers [2]

## 1.4.5 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of artificial neural networks adapted to processing time series and other sequential data like text processing, speech recognition, and DNA sequences. This type of neural network distinguishes the presence of memory that allows previous outputs to be used as inputs while having hidden states. The value of the hidden state at any time moment depends on the value of the hidden state at the previous moment and the inputs at the current moment, as shown in Figure 1.19.



(a) RNN architecture (b) Time-layered representation

Figure 1.19: RNN architecture and it Time-layered representation [2]

- $x^{<t>}$ : designates the input of the network at time step $t$.
- $h^{<t>}$ : represents the hidden state of the network at time step, $t$ and it is computed by equation 1.1.

$$h^{<t>} = g_1 \left( W_{hh} h^{<t-1>} + W_{xh} x^{<t>} + b_h \right) \tag{1.1}$$

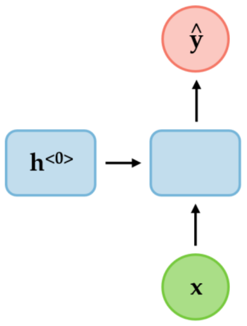
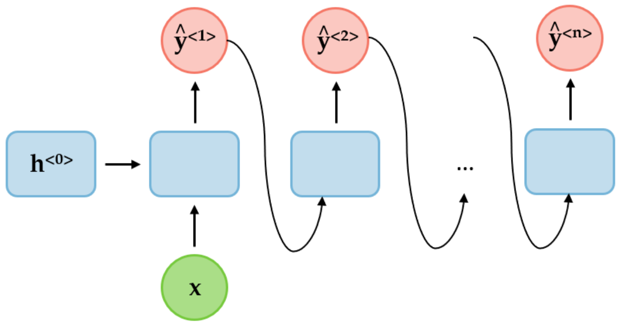
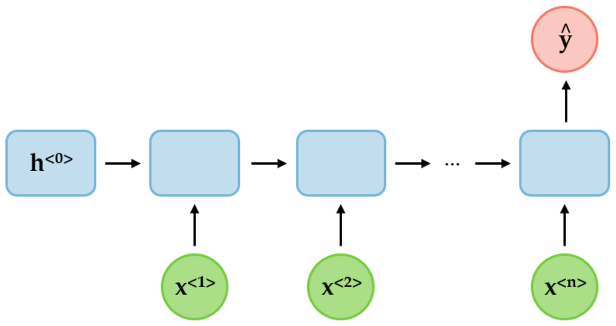
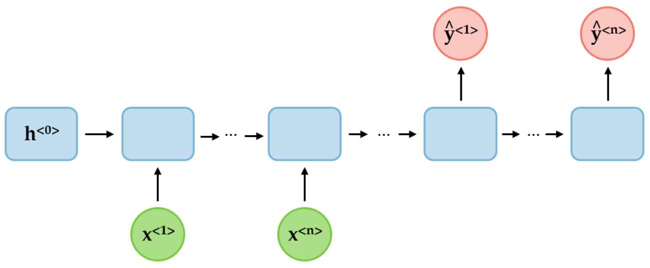
- $y^{<t>}$ : Signified the output of the network at time step $t$ and it is computed by equation 1.2.

$$y^{<t>} = g_2 \left( W_{hy} h^{<t>} + b_y \right) \tag{1.2}$$

Where $w_{xh}$, $w_{hh}$, $w_{hy}$, $b_h$, $b_y$ are coefficients that are shared temporally and $g_1$, $g_2$ activation functions.

The mechanism presented in figure 1.19 allows the network to work in different ways, depending on the size of the sequence we provide as input and the size of the output sequence we expect. Table 1 summarizes some of the ways in which RNN is used and its application.

Table 1.1: Some applications of RNN

| Type of RNN | Illustration | Application |
|---|---|---|
| One-to-One |  | Traditional neural network |
| One-to-many |  | Music generation |
| Many-to-one |  | Sentiment classification |
| Many-to-many |  | Machine translation |

The following table summarizes the advantages and drawbacks of a basic RNN architecture:

Table 1.2: Advantages and drawbacks of basic RNN architecture [2]

| Advantages | Drawbacks |
|---|---|
| • Inputs of any length can be processed<br>• Model size not increasing with size of input<br>• Weights are shared across time | • Due to its recurrent nature, the computation being slow<br>• Difficulty accessing information for a long time<br>• Gradient exploding and vanishing problems |

## Training A Recurrent Neural Network

Recurrent Neural Networks (RNNs) are challenging to train due to their architecture, especially if the input sequence is extensive. Many problems can occur while updating the network weights due to what is known as exploding gradients and vanishing gradients [35].

- *Exploding Gradient* : When those gradients accumulate during an update, the result will be very large.
- *Vanishing Gradients* : When those gradients are small or zero, it will easily vanish.

To address these problems, several effective solutions have been proposed, such as Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs). We will just explain the LSTM architecture.

## Long Short Term Memory (LSTM)

LSTM neural network is an architecture for RNNs introduced by by Hochreiter and Schmidhuber in 1997 [21]. LSTM can be regarded as a modified RNN. The memory of past input is important in the sequence learning problem. LSTM was specially designed and modified to solve the vanishing gradient and explosion gradient problem in long-term training. Due to the memory cell (Figure 1.20), the LSTM can maintain the error values and continue the gradient flow.

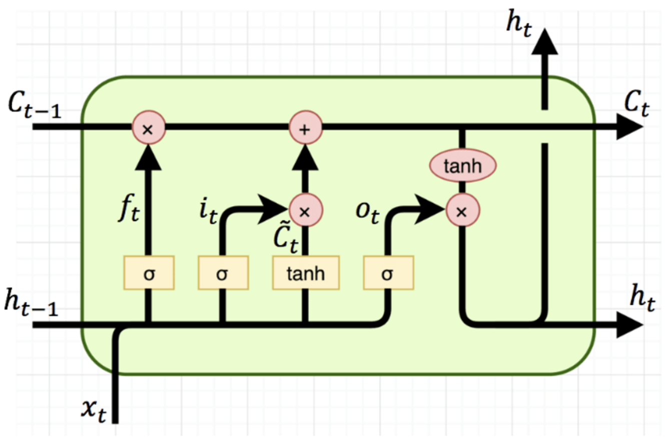


Figure 1.20: LSTM memory cell [10]

- **Cell state** ($c_t$): Vector of fixed shape with random value initialization. It contains the information that was present in the memory after the previous time step.

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{1.3}$$

- **Forget gate** ($f_t$): Changes the cell state, intending to eliminate non-important values from previous time steps. This helps the LSTM network to forget the irrelevant information that does not have any impact on the future price prediction.

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right) \tag{1.4}$$

- **Input gate** ($i_t$): Changes the cell state with the aim of adding new information about the current time step. It adds new information that may affect the stock price movement.

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right) \tag{1.5}$$

- **Output gate** ($o_t$): Decides what the next hidden state should be. The new cell state and the new hidden is then carried over to the next time step. Returns the final relevant information, which will be used for stock price prediction.

$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right) \tag{1.6}$$

- **Hidden state** ($h_t$): It is calculated by multiplying output gate vector by cell state vector.

$$h_t = o_t \odot \tanh(c_t) \tag{1.7}$$

### 1.4.6 Autoencoders

Autoencoder is an artificial neural network designed to learn the feature representation. It consists of two parts: an encoder and a decoder (Figure 1.21). The encoder works on mapping the input data to a low-dimensional representation space to obtain the most appropriate feature, and then decode it back such that the reconstructed input is similar as possible to the original one using the decoder part. The idea was originated in the 1980s and later promoted by the seminal paper by Hinton & Salakhutdinov [18].

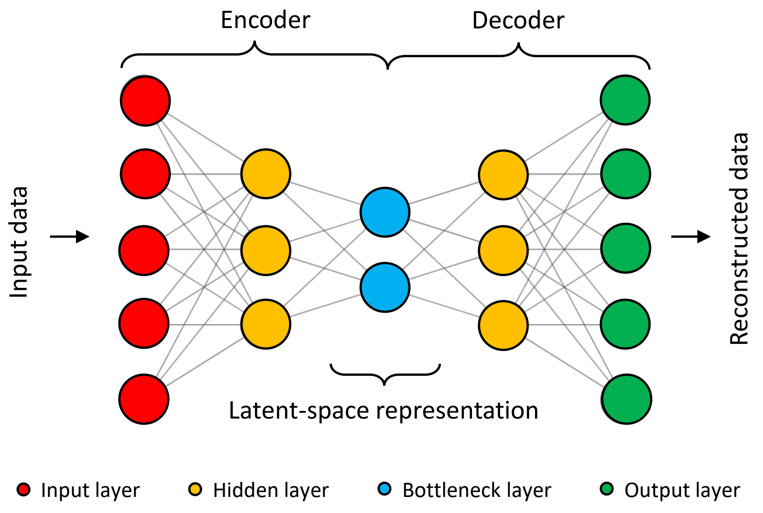


Figure 1.21: Autoencoder architecture

The main purpose of autoencoder training is learning in an unsupervised manner a representation of the data that can be used for various applications such as: dimensionality reduction, clustering, anomaly detection, feature representation and so on. Several kinds of autoencoders are proposed in the literature [5] including sparse autoencoders, denoising autoencoders, contractive autoencoders, variational autoencoders, convolutional autoencoders [16], and so on.

## 1.5 Conclusion

In this chapter, we have touched upon the various basic concepts related to the various aspects of information security and the concepts related to networks in general, as well as intrusion detection systems, their different types, and their working principles, finally, a presentation of the various basic concepts of machine learning, and deep learning. In the next chapter, we survey some typical use of deep learning architectures in the area of network intrusion detection.

# Chapter2

# NETWORK INTRUSION DETECTION USING DEEP LEARNING

## 2.1 Introduction

The increasing reliance on computer networks in business, education, government, and other fields has led to more security incidents. These incidents can have severe consequences for organizations and individuals. Protection systems such as firewalls, encryption, etc., are used to prevent such incidents. A firewall serves as the primary line of defense to protect networks. Therefore, it is essential to use intrusion detection systems for their ability to recognize and reduce malicious attacks. As attackers constantly change attack techniques and find alternative attack methods, intrusion detection systems must also evolve to respond by adopting more sophisticated detection methods.

Many researchers developed network intrusion detection system (NIDS) models. They faced many challenges, including low detection accuracy, emergence of new types of malicious traffic, and error detection rates. They used several methods and techniques to overcome it, one of which is machine learning techniques such as deep learning to develop intrusion detection systems, which will be discussed in this chapter.

Deep learning has proven effective in image recognition, speech recognition, and other areas and has become the preferred solution to many problems. This approach has been gradually applied to intrusion detection in recent years, and it has given impressive detection results compared to the traditional methods. In this part, we will present the most important studies and researches on intrusion detection systems based on deep learning techniques.

## 2.2 Deep neural network based NIDS

### 2.2.1 Shallow neural network based NIDS

A deep neural network was used to detect intrusion in software-defined networking (SDN) by Tang et al [48]. Where they created a deep neural network consisting of an input layer, three hidden layers, and an output layer, as shown in Figure 2.1. In order to train and evaluate the model, the NSL-KDD dataset [31] was used based on the selection of only six basic features from among the forty-one features to detect attacks. This model was implemented to classify network traffic into two categories (normal and abnormal).

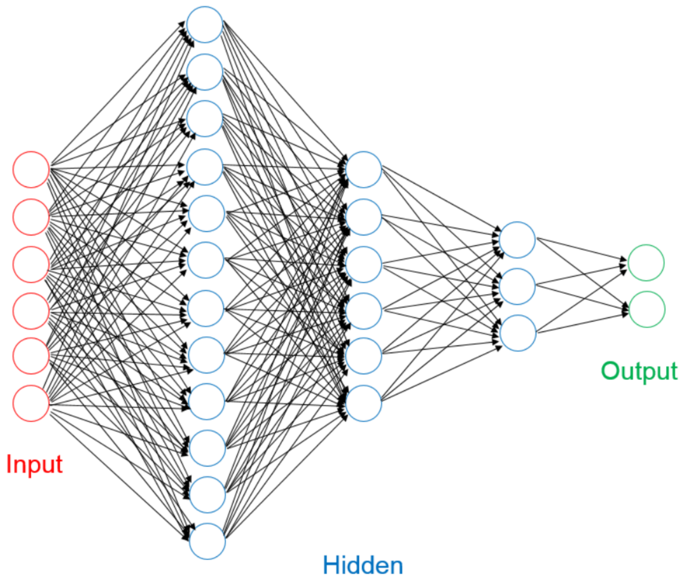


Figure 2.1: Deep Learning Network Model [48]

Experimental results for a model that has been trained using setting parameters are 10 as batch size and 100 for the epoch reported that the learning rate 0.001 was performing more effectively than the others. Where the model achieved an accuracy of 75.75% for the testing dataset and 91.7% for the training dataset.

In another study, Kim et al [24] proposed an IDS framework using a deep neural network consisting of four hidden layers with 100 units and using the ReLU activation function, and the study also used Adam stochastic optimization method. The model was trained on the KDD Cup 99 dataset, which gave an accuracy of 99.01%.

### 2.2.2 Convolutional neural network based NIDS

The model proposed by researchers Sheraz et al [13], uses a deep convolutional neural network (DCNN) for network anomaly detection problems to improve the real-world application in anomaly detection systems. This model consists of an input layer, three pairs of convolutional layers, three fully connected layers and an output layer using one sigmoid unit, as shown in figure 2.2. The NSL-KDD dataset was used for all of the experiments in this research, where the two parts: NSLKDDtrain+ and NSLKDDTrain20p[1], were combined for the training of the model, and the two other parts: NSLKDDTest+ and NSLKDDTest21[2] were combined to evaluate the performance of the trained model.

The dataset has 41 features where three features ('protocol_type', 'service' and 'flag') are symbolic features which needs to be converted to quantitative data before they can be used by DCNN. The authors studied the impact of different category encoding schemes on classification accuracy; they chose Random-Forest algorithm due to its time efficiency, dimensionality and accuracy of classifier. In order to take advantage of the convolutional network, the inputs were converted from a vector of 41 features to a 32x32 2D grayscale image, which helps to discover the localized features and learn high-level relationships between global features.

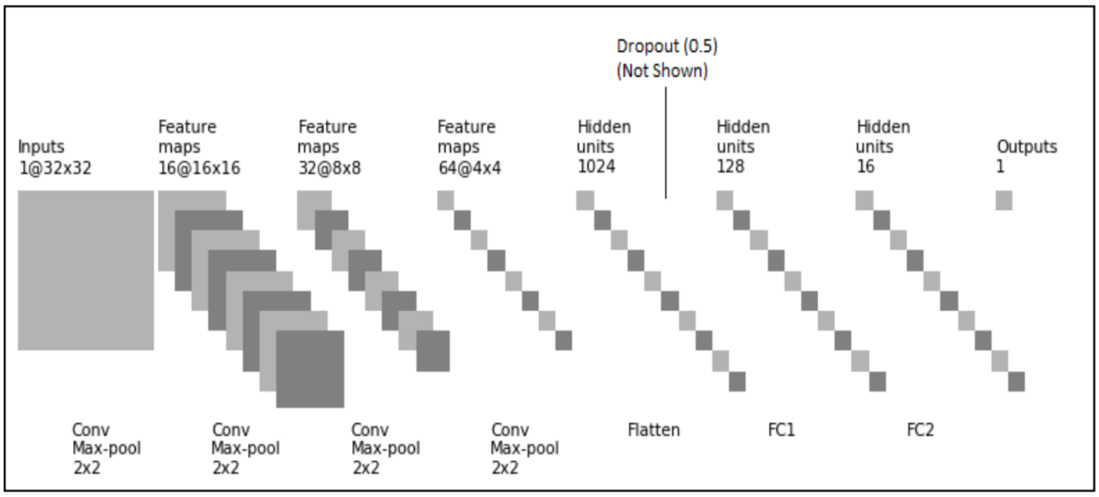


Figure 2.2: Deep Convolutional Neural Network (DCNN) for Intrusion Detection [13]

Experimental results showed that the use of deep convolutional neural networks gave better results than traditional methods such as dicsion tree, naive bayes and MLP, achieving an accuracy of 85.22 % and 69.56% for NSLKDDTest+ and NSLKDDTest21 respectively. The DCNN used in this study is inspired by letNet 5 [27], however this model contains heavy modifications in form of hyper-parameter selection and regularization.

[1] Parts of a training NSL-KDD dataset.
[2] Parts of a testing NSL-KDD dataset.

Teyou and Ziazet [50] proposed a framework for an intrusion detection system based on a Deep Convolutional Neural Network to detect attacks at the industrial control system level. This framework consists of two parts: the first part is devoted to the feature extraction process, using a CNN for this purpose, and the second part is devoted to the classification process, where they use DCNN, as shown in figure 2.3. The study used the benchmark network intrusion NSL-KDD dataset to train the model and evaluate the anomaly detection accuracy.

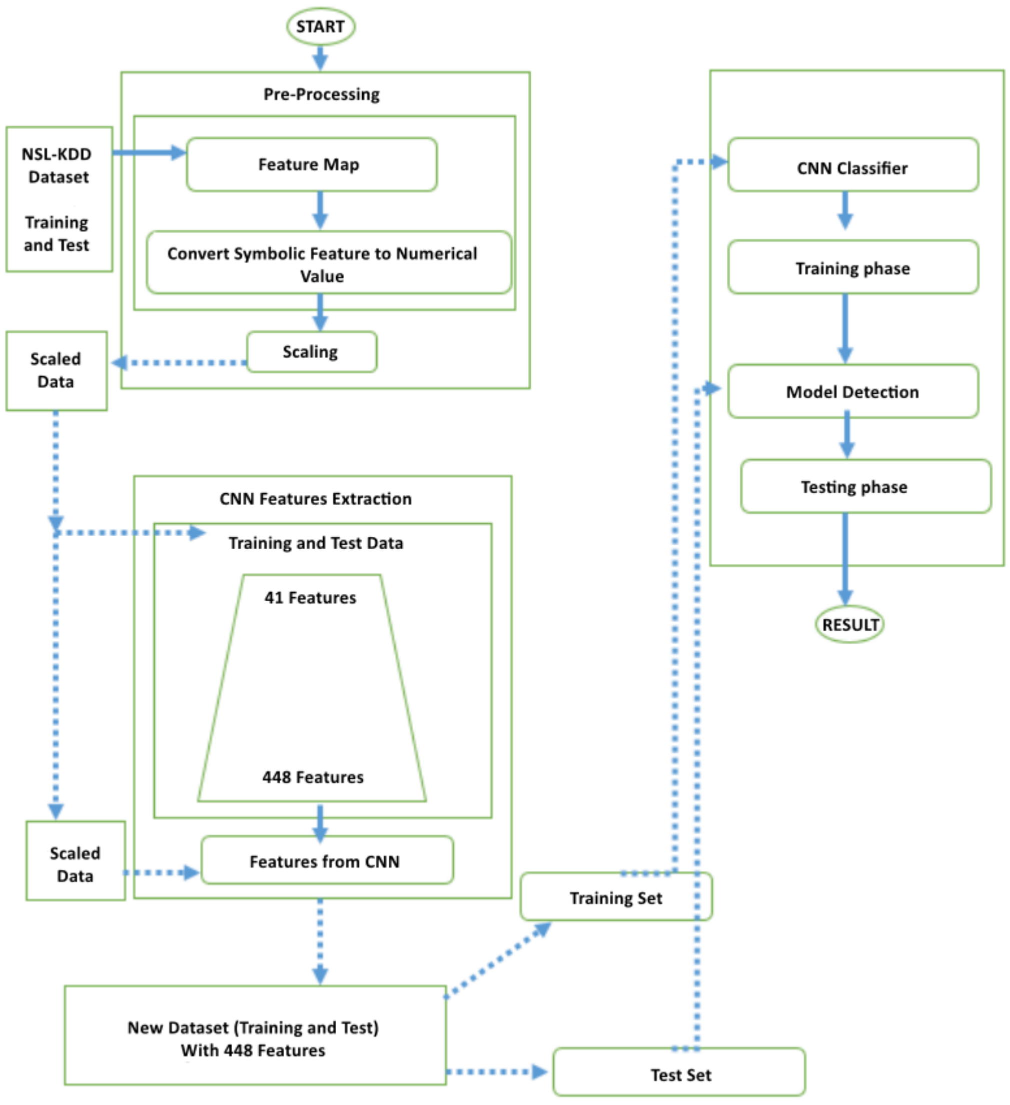


Figure 2.3: General architecture of the proposed system [50]

After conducting several experiments on the models (CNN-SVM, CNN-KNN, CNN-DNN, CNN-CNN), in which CNN was used as a method for extracting features. It was observed that the proposed model achieved better results with an accuracy of 80.07% and 77.15% respectively, on the two classes and five classes classification.

### 2.2.3 Recurrent neural network based NIDS

In the work of Yin et al [56], a Recurrent Neural Network-based intrusion detection system (RNN-IDS) was used. In order to study its effect on the accuracy of intrusion detection in binary and multiclass classification, they conducted experiments using parameters with variant values, such as the number of neurons and the learning rate. This study used the NSL-KDD dataset to train and evaluate the proposed model, as shown in figure 2.4.

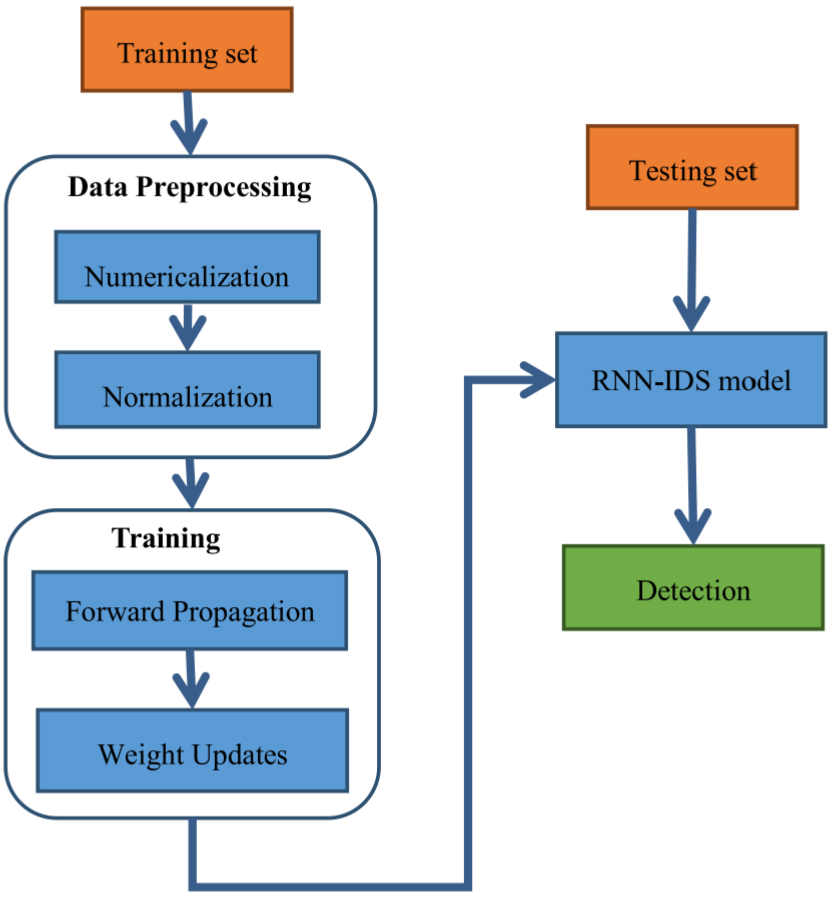


Figure 2.4: Block diagram of proposed RNN-IDS [56]

The experiment results show that the RNN-IDS model gives high accuracy of 83.28% in binary and 81.29% in multiclass classification compared with traditional machine learning models, such as random forest and support vector machine. This accuracy is achieved based on the learning rate of 0.1 and the number of hidden nodes of 80.

In another work by Chaibi et al [7] study the effectiveness of feature selection methods on classification accuracy. They suggested an architecture implemented with two methodologies, as shown in Figure 2.5. In the first one, they use RNN and ANN techniques for the classification process, with a feature selection method of information gain (IG). In the second one, they use an RNN with three methods which are information gain (IG), grain ratio (GR), and correlation attribute (CA), as feature selection.

This study was done on the dataset NSL-KDD which contains 41 features, and after applying the two methodologies, the dataset now comprises 14 features and 29 features, respectively. The model consists of the input layer, two hidden layers with 12 neurons and 26 neurons, respectively, and the output layer for the first methodology. The second methodology is composed of the input layer, two hidden layers with 10 and 23 neurons, respectively. In general, the rectifier function is used as an activation function in the hidden layer and a sigmoid function for the output layer. For the loss function, they used cross-entropy, while for the learning algorithm, they used Adam optimizer.

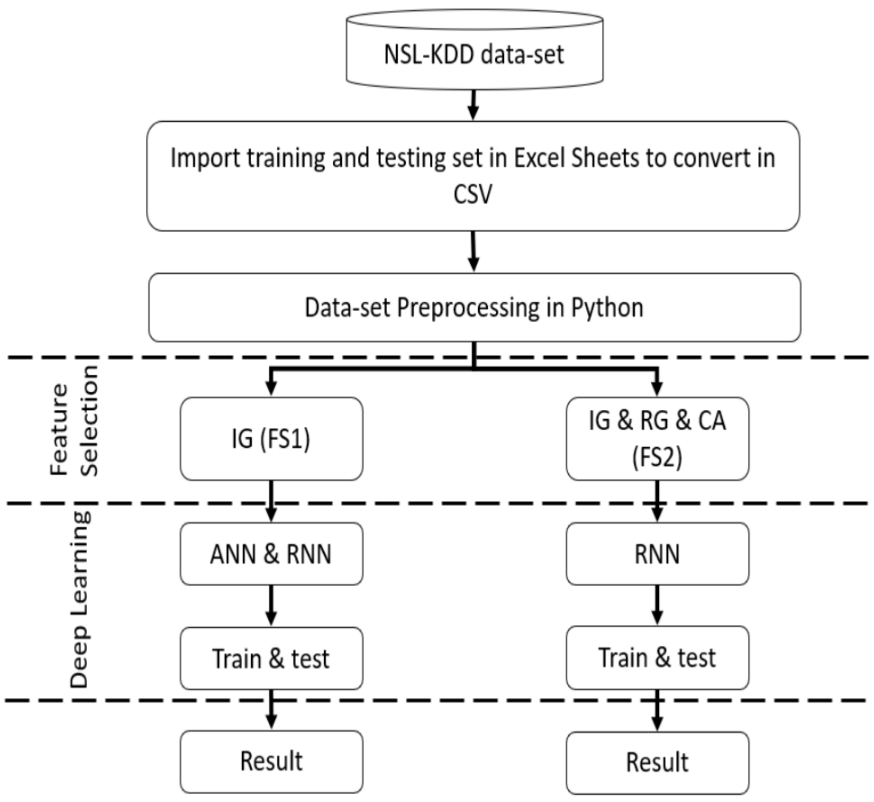


Figure 2.5: Organizational chart for proposed IDS using ANN and RNN [7]

Experimental results showed that the RNN outperforms ANN in the two methodologies. For the first methodology, ANN achieved an accuracy of 98,63% and 98,73% for RNN, and for the second methodology, RNN achieved an accuracy of 99,60%.

### 2.2.4 Autoencoder based NIDS

Javaid et al [23] have developed a network intrusion detection system based on the Self-Taught Learning (STL) approach [37] that consists of two stages for the classification. The first stage is feature representation, in which they used the Sparse Autoencoder. Whereas the second stage is for the classification task, in which they employed the results of the first stage. In this study, they used NSL-KDD for two training stages and also for the evaluation of the model. All the implementation steps are showed in Figure 2.6.

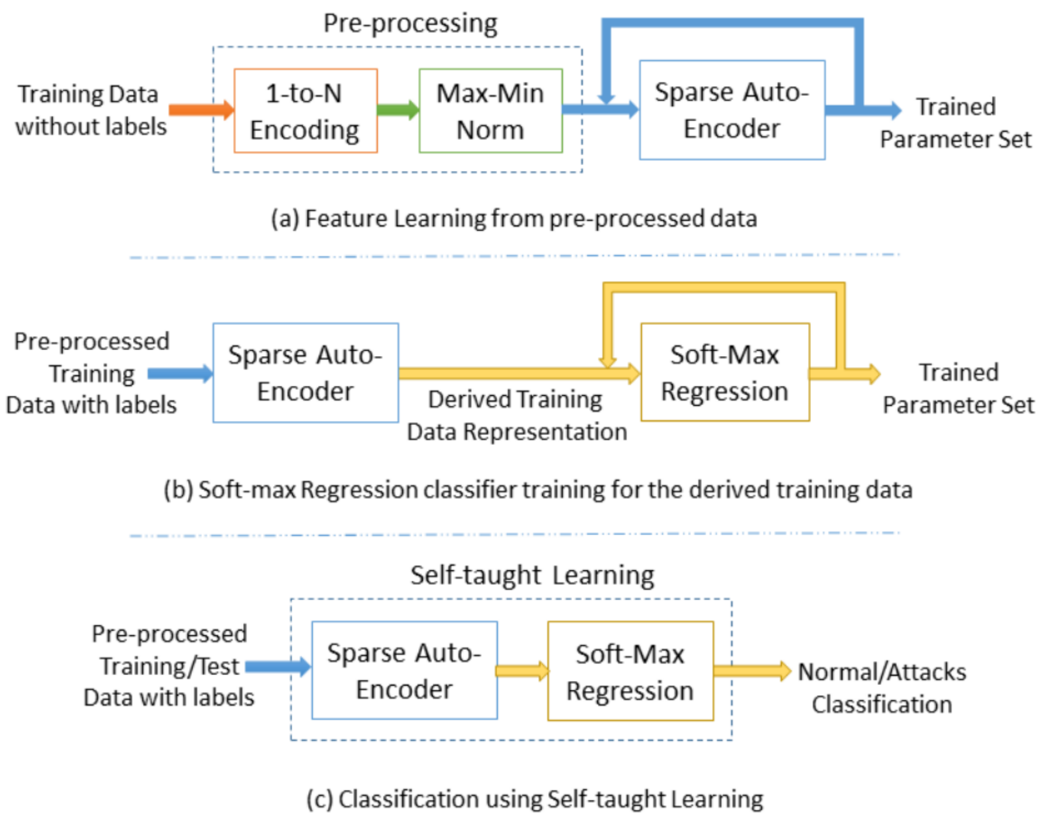

Pre-processing
Training Data without labels
1-to-N Encoding
Max-Min Norm
Sparse Auto-Encoder
Trained Parameter Set
(a) Feature Learning from pre-processed data
Pre-processed Training Data with labels
Sparse Auto-Encoder
Derived Training Data Representation
Soft-Max Regression
Trained Parameter Set
(b) Soft-max Regression classifier training for the derived training data
Self-taught Learning
Pre-processed Training/Test Data with labels
Sparse Auto-Encoder
Soft-Max Regression
Normal/Attacks Classification
(c) Classification using Self-taught Learning

Figure 2.6: Different steps of STL-Based NIDS implementation [23]

In order to evaluate the model, two different types of classification are applied: 2-class (normal, abnormal) classification, 5-class (normal, R2L, U2R, DoS, Porb) classification. To evaluate the classification accuracy, they used the testing dataset. Experimental results show that the proposed model achieved a classification accuracy of 88.39%, 79.10%, for 2-class and 5-class, respectively.

In another study by Rezvy et al [38], a network intrusion detection system model has developed using an Autoencoder and Dense Neural Network. In this model, they employed two main stages. In the first stage, they used an Autoencoder to learn an efficient data representation. In order to achieve this, they used the training with unlabeled data, and they employed a Dense Neural Network to classify attack types in the second stage. Figure 2.7 describes the various stages and the model architecture.

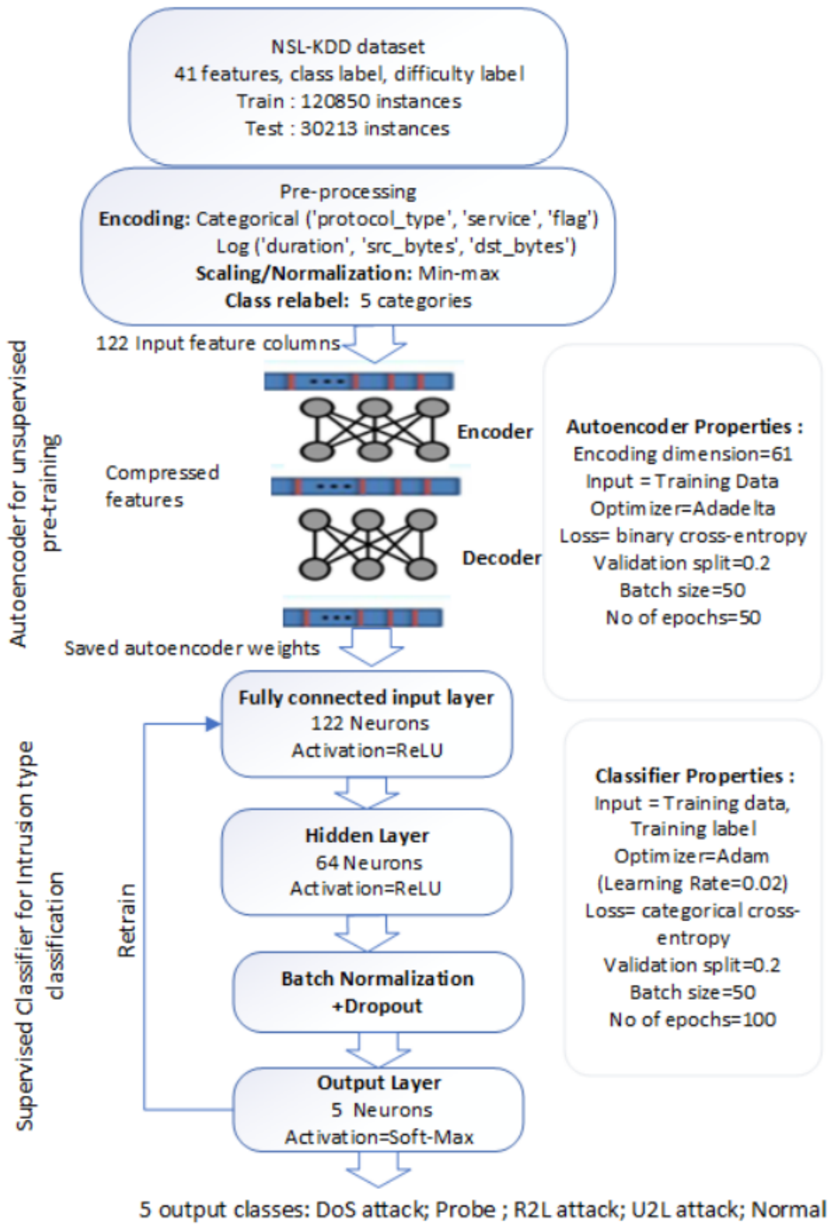

NSL-KDD dataset
41 features, class label, difficulty label
Train : 120850 instances
Test : 30213 instances
Pre-processing
Encoding: Categorical ('protocol_type', 'service', 'flag')
Log ('duration', 'src_bytes', 'dst_bytes')
Scaling/Normalization: Min-max
Class relabel: 5 categories
122 Input feature columns
Autoencoder for unsupervised pre-training
Compressed features
Encoder
Decoder
Autoencoder Properties :
Encoding dimension=61
Input = Training Data
Optimizer=Adadelta
Loss= binary cross-entropy
Validation split=0.2
Batch size=50
No of epochs=50
Saved autoencoder weights
Supervised Classifier for Intrusion type classification
Retrain
Fully connected input layer
122 Neurons
Activation=ReLU
Hidden Layer
64 Neurons
Activation=ReLU
Batch Normalization +Dropout
Output Layer
5 Neurons
Activation=Soft-Max
Classifier Properties :
Input = Training data, Training label
Optimizer=Adam
(Learning Rate=0.02)
Loss= categorical cross-entropy
Validation split=0.2
Batch size=50
No of epochs=100
5 output classes: DoS attack; Probe ; R2L attack; U2L attack; Normal

Figure 2.7: Workflow and architecture of the model proposed in [38]

The model was trained and evaluated using the NSL-KDD dataset for 5-class classification. Experimental results show that the model achieved excellent performance with an accuracy of 99.3% for the five classes of attacks.

## 2.3 Conclusion

In this chapter, we have discussed some deep learning techniques used in developing network intrusion detection systems. We showed their effectiveness in this field compared to traditional machine learning methods. We did not mention those traditional methods in this chapter. We have just concentrated on deep learning methods. In the next chapter, we will propose a model for an intrusion detection system using a deep learning approach.

# Chapter3

# EXPERIMENT

## 3.1 Introduction

In this chapter, we present the proposed model of the network intrusion detection system and the various stages of its training. We also describe the dataset and discuss the experimental results.

## 3.2 Data Understanding

In this research, We relied on the NSL-KDD benchmark dataset to train and evaluate our model because it is the most widely used in the field of building intrusion detection systems and the train and test sets have a reasonable amount of records, making it suitable to conduct the experiment on the complete set without the need to randomly select a small portion. As a result, the evaluation outcomes of various research projects will be consistent and comparable.

The NSL-KDD data set is one of the benchmark datasets for intrusion detection systems proposed by the Canadian Institute of Cybersecurity in 2009 to solve some problems of the KDD'99 dataset discussed by Tavallaee et al [49]. The NSL-KDD dataset contains 125973 and 22544 records for training and testing sets, respectively. Each record consists of 41 features plus a target one. It can be classified into four categories:

- **Basic features:** Is the basic information in the packet header derived from TCP/IP connection without checking the payload, as shown in Table 3.1.
- **Content features:** To detect some types of attacks, such as R2L and U2R, we must examine the content of packets, so we need some features to detect suspicious behavior in the data portion, as shown in Table 3.2.

- **Time-based features:** This type of feature is created based on traffic analysis over a two-second time window, one example being the number of connections made with the same host, as shown in Table 3.3.
- **Host-based features:** These features were created to detect attacks lasting longer than two seconds, so they result from an analysis of a series of connections made to the same host, as shown in Table 3.4.

Those features consists of 38 continuous or discrete numerical features and 3 categorical features (protocol_type, service and flag). Table 3.5 shows the different values of categorical features.

Table 3.1: Basic features of NSL KDD dataset [56]

| N° | Feature name | Description | Type |
|---|---|---|---|
| 1 | duration | length (number of seconds) of the connection | Continuous |
| 2 | protocol_type | type of the protocol, e.g. tcp, udp, etc. | Symbolic |
| 3 | service | network service on the destination, e.g., http, telnet, etc. | Symbolic |
| 4 | flag | normal or error status of the connection | Symbolic |
| 5 | src_bytes | number of data bytes from source to destination | Continuous |
| 6 | dst_bytes | number of data bytes from destination to source | Continuous |
| 7 | land | 1 if connection is from/to the same host/port; 0 otherwise | Symbolic |
| 8 | wrong_fragment | number of ''wrong'' fragments | Continuous |
| 9 | urgent | number of urgent packets | Continuous |

Table 3.2: Content features of NSL KDD dataset [56]

| N° | Feature name | Description | Type |
|---|---|---|---|
| 10 | hot | number of ''hot'' indicators | Continuous |
| 11 | num_failed_logins | number of failed login attempts | Continuous |
| 12 | logged_in | 1 if successfully logged in; 0 otherwise | Symbolic |
| 13 | num_compromised | number of ''compromised'' conditions | Continuous |
| 14 | root_shell | 1 if root shell is obtained; 0 otherwise | Continuous |
| 15 | su_attempted | 1 if ''su root'' command attempted; 0 otherwise | Continuous |
| 16 | num_root | number of ''root'' accesses | Continuous |
| 17 | num_file_creations | number of file creation operations | Continuous |
| 18 | num_shells | number of shell prompts | Continuous |
| 19 | num_access_files | number of operations on access control files | Continuous |
| 20 | num_outbound_cmds | number of outbound commands in an ftp session | Continuous |
| 21 | is_host_login | 1 if the login belongs to the ''hot'' list; 0 otherwise | Symbolic |
| 22 | is_guest_login | 1 if the login is a ''guest''login; 0 otherwise | Symbolic |

Table 3.3: Time-based features of NSL KDD dataset [56]

| N° | Feature name | Description | Type |
|---|---|---|---|
| 23 | count | Number of connections to the same destination host as the current connection in the past two seconds | Continuous |
| 24 | srv_count | Number of connections to the same service (port number) as the current connection in the past two seconds | Continuous |
| 25 | serror_rate | The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23) | Continuous |
| 26 | srv_serror_rate | The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24) | Continuous |
| 27 | rerror_rate | The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23) | Continuous |
| 28 | srv_rerror_rate | The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24) | Continuous |
| 29 | same_srv_rate | The percentage of connections that were to the same service, among the connections aggregated in count (23) | Continuous |
| 30 | diff_srv_rate | The percentage of connections that were to different services, among the connections aggregated in count (23) | Continuous |
| 31 | srv_diff_host_rate | The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24) | Continuous |

Table 3.4: Host-based features of NSL KDD dataset [56]

| N° | Feature name | Description | Type |
|---|---|---|---|
| 32 | dst_host_count | Number of connections having the same destination host IP address | Continuous |
| 33 | dst_host_srv_count | Number of connections having the same port number | Continuous |
| 34 | dst_host_same_srv_rate | The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32) | Continuous |
| 35 | dst_host_diff_srv_rate | The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32) | Continuous |
| 36 | dst_host_same_src_port _rate | The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33) | Continuous |
| 27 | dst_host_srv_diff_host_r ate | The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33) | Continuous |
| 38 | dst_host_serror_rate | 1 if "su root" command attempted; 0 otherwise | Continuous |
| 39 | dst_host_srv_serror_rate | The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32) | Continuous |
| 40 | dst_host_rerror_rate | The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33) | Continuous |
| 41 | dst_host_srv_rerror_rate | The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33) | Continuous |

Table 3.5: Different values for categorical features

| Protocol Type | Service | | | | Flag |
|---|---|---|---|---|---|
| • icmp<br>• tcp<br>• udp | • ftp_data<br>• other<br>• private<br>• http<br>• remote_job<br>• name<br>• netbios_ns<br>• eco_i<br>• mtp<br>• telnet<br>• finger<br>• domain_u<br>• supdup<br>• uucp_path<br>• Z39_50<br>• smtp<br>• csnet_ns<br>• uucp | • ssh<br>• netbios_dgm<br>• urp_i<br>• auth<br>• domain<br>• ftp<br>• bgp<br>• ldap<br>• ecr_i<br>• gopher<br>• vmnet<br>• systat<br>• http_443<br>• efs<br>• whois<br>• Imap4<br>• iso_tsap<br>• tftp_u | • **http_2784**<br>• **harvest**<br>• echo<br>• klogin<br>• link<br>• sunrpc<br>• login<br>• kshell<br>• sql_net<br>• time<br>• hostnames<br>• exec<br>• ntp_u<br>• Discard<br>• nntp<br>• courier<br>• ctf | • daytime<br>• shell<br>• netstat<br>• pop_3<br>• nnsp<br>• IRC<br>• pop_2<br>• printer<br>• tim_i<br>• pm_dump<br>• **red_i**<br>• netbios_ssn<br>• rje<br>• X11<br>• **urh_i**<br>• **http_8001**<br>• **aol** | • S0<br>• S1<br>• S2<br>• S3<br>• SF<br>• SH<br>• RSTR<br>• RSTO<br>• REJ<br>• RSTOS0<br>• OTH |
| **3** | **70** | | | | **11** |

The training set of NSL-KDD contains 22 attack types, while the testing set contains an additional 15 attack types. This attacks is divided into four main categories. Table 3.6 shows the attacks distribution in the NSL-KDD train and test sets.

Table 3.6: The attacks distribution of the NSL-KDD dataset

| **Category** | **Attack type** | **Training set KDDTrain+** | **Testing set KDDTest+** |
|---|---|---|---|
| Normal | Normal | **67343** | **9711** |
| Denial of Service (**DoS**) | Apache2 | - | 737 |
| | Back | 956 | 359 |
| | Land | 18 | 7 |
| | Neptune | 41214 | 4657 |
| | Mailbomb | - | 293 |
| | Processtable | - | 685 |
| | Pod | 201 | 41 |
| | Smurf | 2646 | 665 |
| | Teardrop | 892 | 12 |
| | Udpstorm | - | 2 |
| | Worm | - | 2 |
| | | **45927** | **7460** |
| **Probe** | Ipsweep | 3599 | 141 |
| | Mscan | - | 996 |
| | Nmap | 1493 | 73 |
| | Portsweep | 2931 | 157 |
| | Saint | - | 319 |
| | Satan | 3633 | 735 |
| | | **11656** | **2421** |
| Remote to Local (**R2L**) | Ftp_write | 8 | 3 |
| | Guess_passwd | 53 | 1231 |
| | Httptunnel | - | 133 |
| | Imap | 11 | 1 |
| | Multihop | 7 | 18 |
| | Named | - | 17 |
| | Phf | 4 | 2 |
| | Sendmail | - | 14 |
| | Snmpgetattack | - | 178 |
| | Snmpguess | - | 331 |
| | Spy | 2 | - |
| | Warezclient | 890 | - |
| | Warezmaster | 20 | 944 |
| | Xlock | - | 9 |
| | Xsnoop | - | 4 |
| | | **995** | **2885** |
| User to Root (**U2R**) | Buffer_overflow | 30 | 20 |
| | Loadmodule | 9 | 2 |
| | Perl | 3 | 2 |
| | Ps | - | 15 |
| | Rootkit | 10 | 13 |
| | Sqlattack | - | 2 |
| | Xterm | - | 13 |
| | | **52** | **67** |
| **Total** | | **125973** | **22544** |

## 3.3 Data Preprocessing

Data preprocessing is an essential step in the machine learning process as it takes raw data and turns it into forms that are understandable and usable by machine learning algorithms. Data preprocessing has many important steps, such as data cleaning, data transformation, feature selection, feature encoding, and feature scaling.

### 3.3.1 Features Selection

After analyzing the data set's features, we found a feature called *num_outbound_cmds* that has a zero value for all the samples for both the train and test datasets. Hence, it will be removed, because it does not give any difference for samples. As a result, the feature set becomes 40 features.

### 3.3.2 Features Encoding

Only numerical values can be used in deep learning models. For this reason, the categorical values of the NSL-KDD dataset must be converted to numerical values. For that, we have chosen one-hot encoding as an encoding technique. For example the feature *protocol_type* has three types of values, tcp, udp, and icmp, and encoding result is binary vectors (1,0,0), (0,1,0) and (0,0,1), respectively. *Service* and *flag* features are treated in the same way. As result of this process, 40-dimensional features map into 121-dimensional features.

### 3.3.3 Features Scaling

In order to standardize the data range and reduce the computational power, we have conducted feature scaling using the MinMax method 3.1 on each feature value to map it in the range of 0 to 1.

$$x_i = \frac{x_i - Min}{Max - Min} \tag{3.1}$$

## 3.4 Methodology

In this section, we describe our proposed model for a network intrusion detection system based on the Self-taught Learning (STL) approach [37], which consists of two stages for the classification process. The first stage is feature representation: it transforms the representation of the network traffic feature into another that allows us to better detect data patterns. For this, we used unsupervised learning technique with a Convolutional Autoencoder. The second stage is for the classification task. In this stage, the new feature representation is used as an input. Figure 3.1 shows the two-stage of STL.

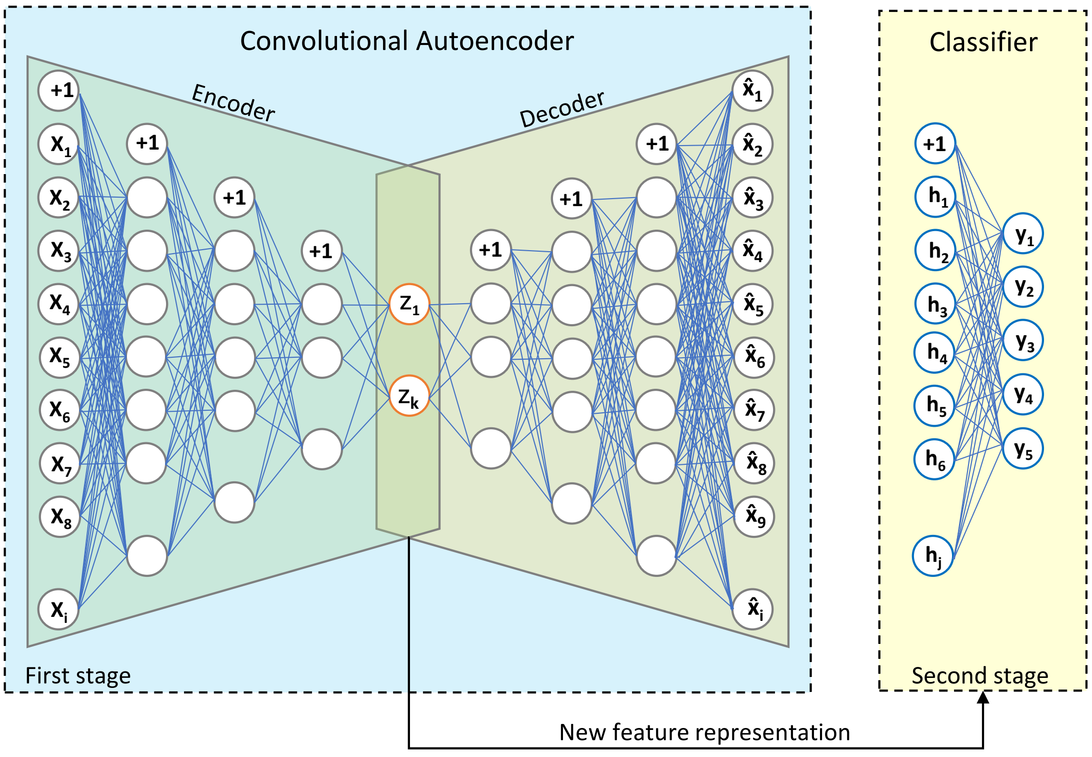

Convolutional Autoencoder
Encoder
Decoder
+1
$x_1$ $x_2$ $x_3$ $x_4$ $x_5$ $x_6$ $x_7$ $x_8$ $x_i$
$z_1$ $z_k$
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$ $\hat{x}_7$ $\hat{x}_8$ $\hat{x}_9$ $\hat{x}_i$
First stage
Classifier
$h_1$ $h_2$ $h_3$ $h_4$ $h_5$ $h_6$ $h_j$
$y_1$ $y_2$ $y_3$ $y_4$ $y_5$
Second stage
New feature representation

Figure 3.1: The two stages of STL

### 3.4.1 Proposed Model

Feature representation is one of the most important stages in the development of machine learning models, as it affects the accuracy and efficiency of the model. As quoted by Jeff Hawkins: *“ The key to artificial intelligence has always been the representation” [17].*

For this reason, we choose unsupervised learning method that we used a convolutional autoencoder, because it combined two properties, features reduction and features extraction, which allows to give better features representation. Based on this, we proposed the model that appears in Figure 3.2. It contains the following steps:

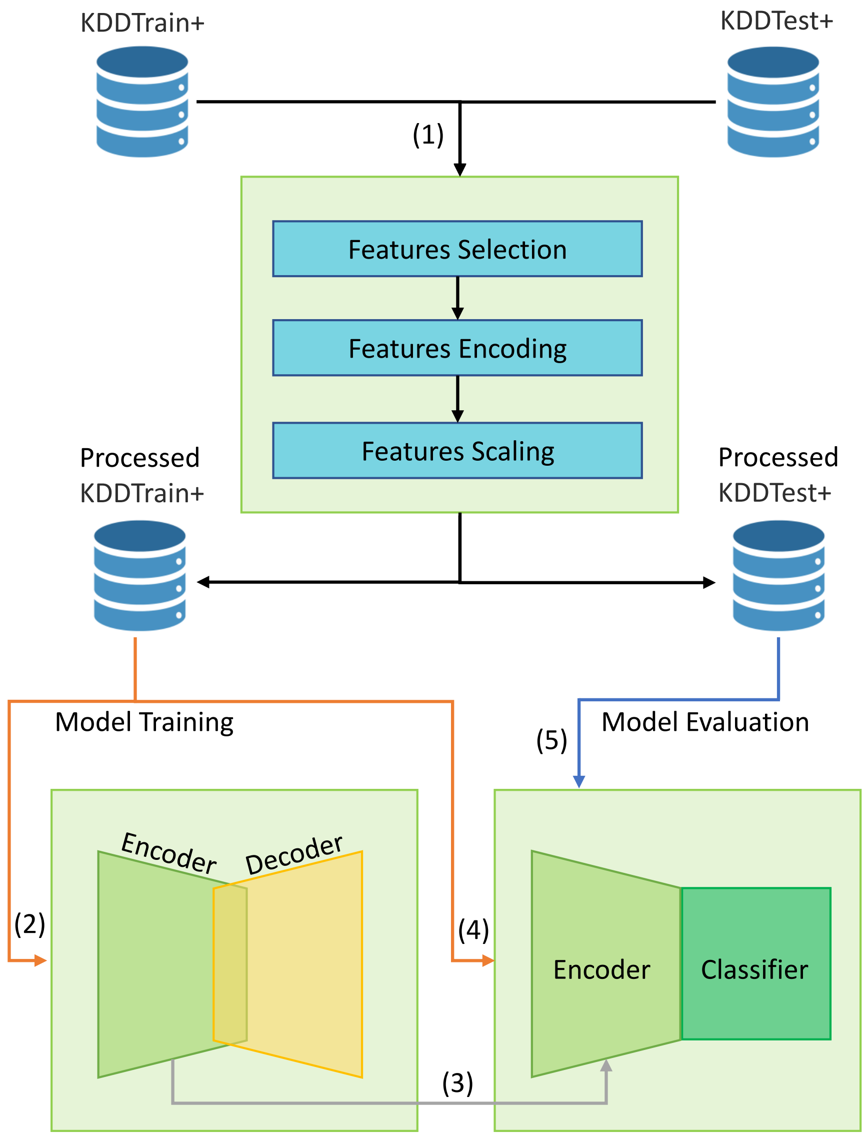


Figure 3.2: Block diagram of the proposed model

- **Step 1:** Dedicated to data preprocessing.
- **Step 2:** Training of convolutional autoencoder using unlabeled data.
- **Step 3:** Save the parameters of encoder and connect it to the classifier.
- **Step 4:** Training of classifier with labeled data.
- **Step 5:** Model evaluation using processed KDDTest+.

### 3.4.2 Model Training

To obtain an effective intrusion detection model, we trained the model using a training set that meets certain conditions in each training method. Table 1 shows a summary of the used training methods.

Table 3.7: Model training methods

| | Training set used in step (2) | Training set used in step (4) |
|---|---|---|
| Method 1 | All the training set samples | All the training set samples |
| Method 2 | Normal samples of the training set | All the training set samples |
| Method 3 | Attacks samples of the training set | All the training set samples |

## 3.5 Model Evaluation

### 3.5.1 Evaluation Metric

The performance of a network intrusion detection system is evaluated by its ability to classify correctly the network traffic as normal or abnormal traffic, For this we have used all performance measures such as Accuracy (AC), Precision (P), Recall (R), and F1-measure (F). They depend on calculating the values of the confusion matrix as shown in the Table 3.8.

Table 3.8: Confusion Matrix

| | | Predicted class ($\hat{y}$) | |
|---|---|---|---|
| | | Positive | Negative |
| Actual class (y) | Positive | TP | FN |
| | Negative | FP | TN |

- **True positive (TP):** Abnormal samples are correctly classified as an abnormal.
- **False positive (FP):** Normal samples are wrongly classified as abnormal.
- **True negative (TN):** Normal samples are correctly classified as normal.
- **False negative (FN):** Abnormal samples are wrongly classified as normal.

Then, we calculate the performance measures from the following formulas :

- **Accuracy (AC):** Defined as the percentage of correctly classified samples over the total number of samples, as shown in 3.2.

$$AC = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{3.2}$$

- **Precision (P):** Defined as the percentage of correct predictions of intrusions over the total predicted intrusions, as shown in 3.3.

$$P = \frac{TP}{TP + FP} \times 100\% \tag{3.3}$$

- **Recall (R):** Defined as the percentage of correct predictions over the total of actual intrusion samples, as shown in 3.4.

$$R = \frac{TP}{TP + FN} \times 100\% \tag{3.4}$$

- **F1-measure (F):** It is considered the most important metric of network intrusion detection, where defined as the harmonic mean of precision and recall and represents a balance between them, as shown in 3.5.

$$F1 = \frac{2 \times P \times R}{P + R} \tag{3.5}$$

### 3.5.2 Experimental results and discussion

In this part, we will present the various experimental results of all the proposed methods with their discussion.

#### 3.5.2.1 Experimental Setup

The experimental environment is more important in preparing models and confirming the results of theoretical studies. Our study used the following settings to implement and test the proposed model.

- **Hardware setup:**
  - CPU : Core i7-6800K 3.40GHz
  - RAM : 16 GO
  - GPU : NIVIDIA GeForce GTX 1660 6G
- **Software setup:**
  - SE : Windows 10 Pro 64 Bits
  - IDE : Microsoft Visual Studio Code with Jupyter Notebook Extension.
  - Anaconda (version 3.0) : It is an integrated platform that contains various packages used in data science and machine learning based on Python and R language.
  - Python (version 3.8.8) : Python is a high-level, public domain interpreted language that is flexible and attempts to express programming concepts with as few codes as possible. Python supports both object and procedural programming, and has a large standard library. Python is an open source language, supported by most operating systems.
  - Tensorflow (version 2.3.0) : is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks, but has a particular focus on training and inference of deep neural networks.
  - Keras (version 2.4.3) : Keras is a deep learning framework that makes it easy to implement and test models.

#### 3.5.2.2 Experimental results of the first method

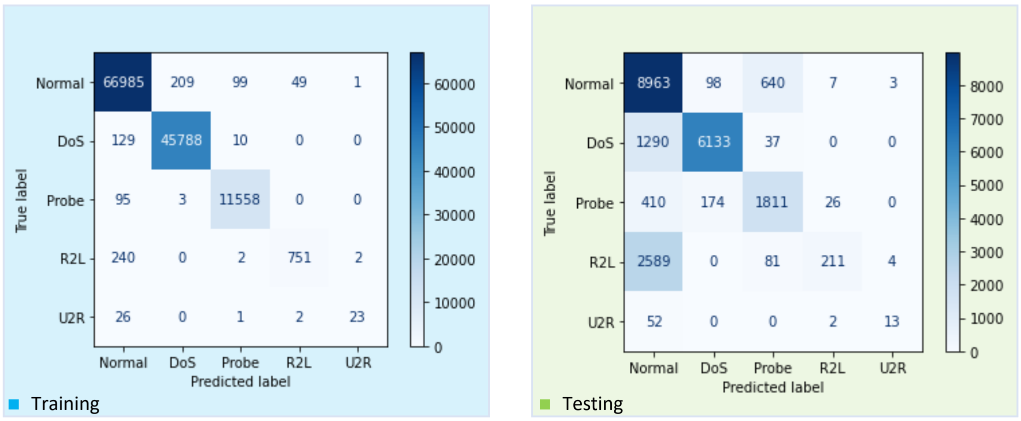


Figure 3.3: Multi-classification confusion matrix of method 1

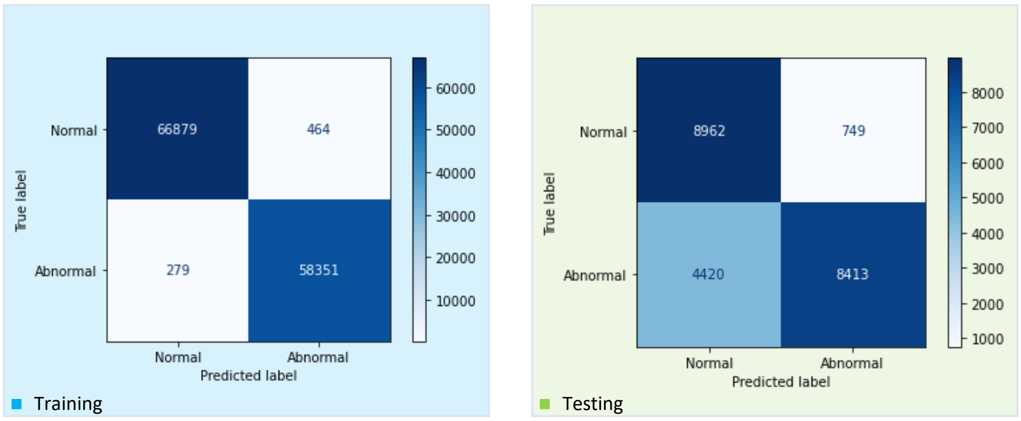


Figure 3.4: Binary-classification confusion matrix of method 1

Table 3.9: The results of the performance measures for method 1

| | Accuracy (%) | | Precision (%) | | Recall (%) | | F-Measure (%) | |
|---|---|---|---|---|---|---|---|---|
| | Training | Testing | Training | Testing | Training | Testing | Training | Testing |
| Multi Classification | 99,31 | 75,98 | 95,98 | 76,87 | 83,60 | 55,20 | 89,36 | 64,26 |
| Binary Classification | 99,41 | 77,07 | 99,39 | 79,39 | 99,41 | 78,91 | 99,40 | 79,15 |

### 3.5.2.3 Experimental results of the second method

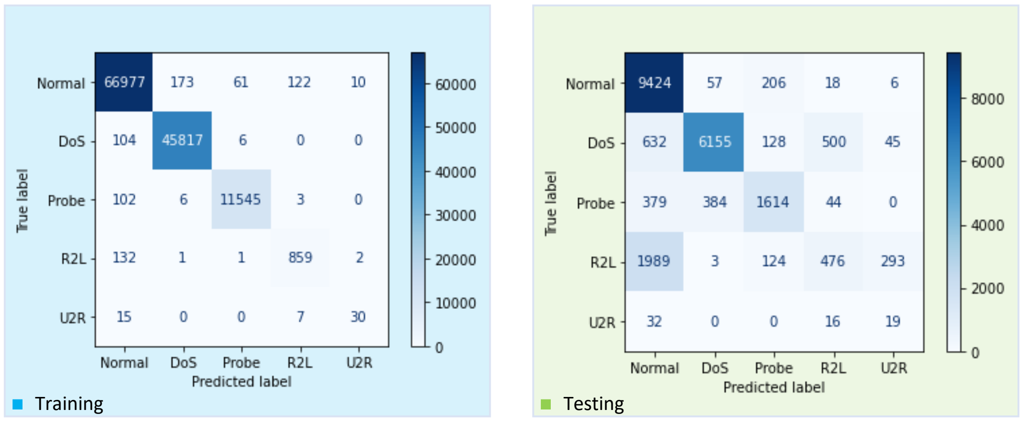


Figure 3.5: Multi-classification confusion matrix of method 2

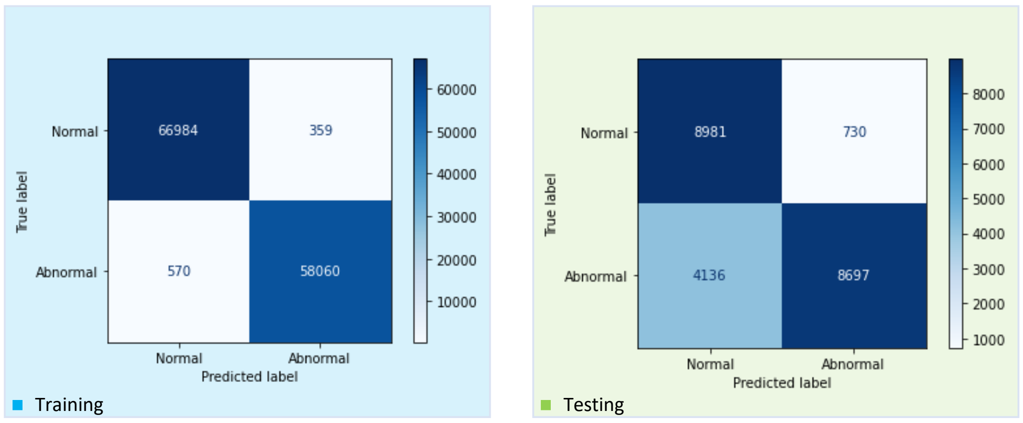


Figure 3.6: Binary-classification confusion matrix of method 2

Table 3.10: The results of the performance measures for method 2

| | Accuracy (%) | | Precision (%) | | Recall (%) | | F-Measure (%) | |
|---|---|---|---|---|---|---|---|---|
| | Training | Testing | Training | Testing | Training | Testing | Training | Testing |
| Multi Classification | 99,41 | 78,45 | 91,31 | 59,44 | 88,45 | 58,20 | 89,86 | 58,81 |
| Binary Classification | 99,26 | 78,41 | 99,26 | 80,35 | 99,24 | 80,12 | 99,25 | 80,23 |

#### 3.5.2.4 Experimental results of the third method

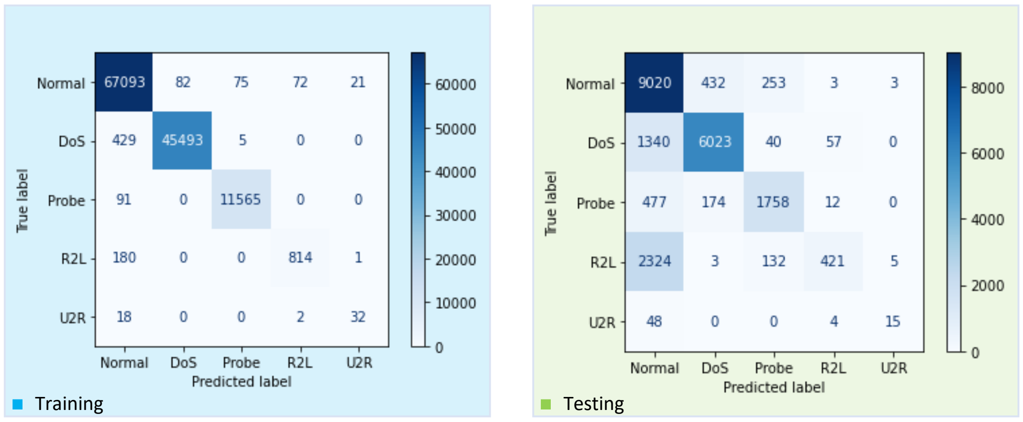


Figure 3.7: Multi-classification confusion matrix of method 3

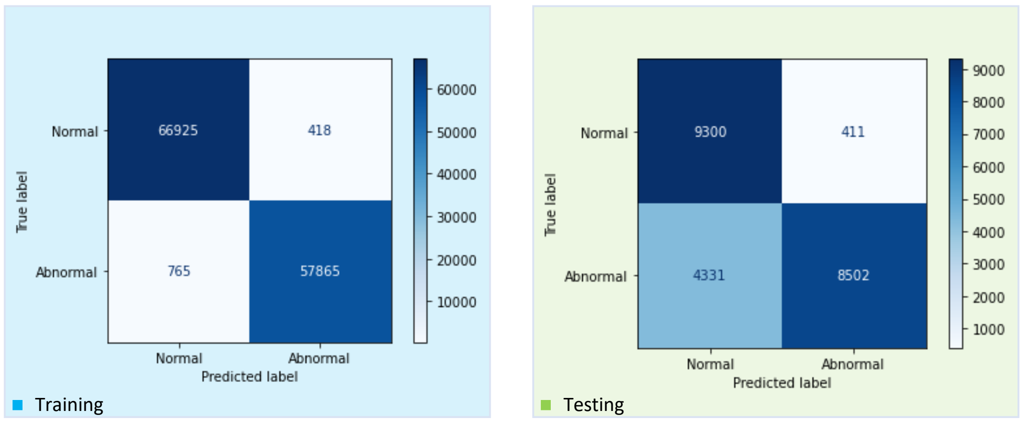


Figure 3.8: Binary-classification confusion matrix of method 3

Table 3.11: The results of the performance measures for method 3

| | Accuracy (%) | | Precision (%) | | Recall (%) | | F-Measure (%) | |
|---|---|---|---|---|---|---|---|---|
| | Training | Testing | Training | Testing | Training | Testing | Training | Testing |
| Multi Classification | 99,22 | 76,45 | 89,79 | 77,90 | 88,24 | 56,63 | 89,00 | 65,58 |
| Binary Classification | 99,06 | 78,97 | 99,07 | 81,80 | 99,03 | 81,00 | 99,05 | 81,40 |

#### 3.5.2.5 Summary of the model evaluation

Table 3.12: Summary of model evaluation

| | | Accuracy (%) | | Precision (%) | | Recall (%) | | F-Measure (%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | Training | Testing | Training | Testing | Training | Testing | Training | Testing |
| Multi-Classification | Method 1 | 99,31 | 75,98 | 95,98 | 76,87 | 83,60 | 55,20 | 89,36 | 64,26 |
| | Method 2 | 99,41 | **78,45** | 91,31 | 59,44 | 88,45 | 58,20 | 89,86 | **58,81** |
| | Method 3 | 99,22 | 76,45 | 89,79 | 77,90 | 88,24 | 56,63 | 89,00 | 65,58 |
| Binary-Classification | Method 1 | 99,41 | 77,07 | 99,39 | 79,39 | 99,41 | 78,91 | 99,40 | 79,15 |
| | Method 2 | 99,26 | **78,41** | 99,26 | 80,35 | 99,24 | 80,12 | 99,25 | **80,23** |
| | Method 3 | 99,06 | 78,97 | 99,07 | 81,80 | 99,03 | 81,00 | 99,05 | 81,40 |

#### 3.5.2.6 Discussion

Depending on the experimental results, we note:

- The superiority of the second and third methods, which rely on training the model using normal samples and abnormal samples, respectively.
- The weakness of the model in detecting R2L attacks, and this is because of the data set, which contains 995 and 2885 samples of this attack for the training set, the testing set, respectively.
- We note that the second method outperforms the third by 2.3% less in classifying the anomaly as normal

Based on this, we adopt the proposed model with the second method as a reference model for our work.

## 3.6 Conclusion

In this chapter, we discussed our model and the experimental results of the various methods used for training it. The experimental results show that our model achieved a classification accuracy of 78.41%, and 78.45%, for 2-class, and 5-class, respectively.

# CONCLUSION

In an open and digitalized era, network security is a primordial issue for every organization. Thus, providing security solutions becomes very urgent and crucial. Intrusion detection systems aim to predict attacks and ensure information security and integrity; they are one alternative among others that protect networks from various dangerous threats.

In this work, we have proposed an anomaly network intrusion detection based system. The model complies with self-taught deep learning framework and uses a convolutional auto-encoder to get a better feature representation of the input data. We found that restricting the training model in the first stage of the auto-encoder to the normal samples only gave fairly good results. This offers to our model more flexibility and compatibility with the most common security policies. The experimental results show that the proposed model gives a good accuracy of 78.41 % for both binary and multi-class classification.

The present work can be further improved in the future. We report some research directions. Firstly, the use of the few-shot learning in the second stage of self-taught learning. Secondly, the use of intrusion embedding with a self-attention mechanism.Finally, the Investigation of collaborative IDS by combining work load and knowledge of multiples IDSs to inspect the network traffic.

# BIBLIOGRAPHY

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for boltzmann machines." In: *Cognitive Science* 9.1 (1985), pp. 147–169. ISSN: 0364-0213. DOI: https://doi.org/10.1016/S0364-0213(85)80012-4. URL: https://www.sciencedirect.com/science/article/pii/S0364021385800124.

[2] Charu C. Aggarwal. *Neural Networks and Deep Learning - A Textbook.* Springer, 2018. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0. URL: https://doi.org/10.1007/978-3-319-94463-0.

[3] AIAI. *Your guide to deep learning.* AI Accelerator Institute | Future of Artificial Intelligence. Mar. 14, 2022. URL: https://www.aiacceleratorinstitute.com/your-guide-to-deep-learning/ (visited on 04/24/2022).

[4] Andrew R. Baker et al. *Snort 2.1 intrusion detection.* 2nd edition. Jay Beale's open source security series. Rockland: Syngress, 2004. ISBN: 978-1-931836-04-3.

[5] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders." In: *CoRR* abs/2003.05991 (2020). arXiv: 2003.05991. URL: https://arxiv.org/abs/2003.05991.

[6] Tom B. Brown et al. "Language Models are Few-Shot Learners." In: (July 22, 2020). arXiv: 2005.14165. URL: http://arxiv.org/abs/2005.14165 (visited on 04/25/2022).

[7] Nassima Chaibi, Baghdad Atmani, and Mostefa Mokaddem. "Deep Learning Approaches to Intrusion Detection: A new Performance of ANN and RNN on NSL-KDD." In: *Proceedings of the 1st International Conference on Intelligent Systems and Pattern Recognition.* ISPR '20: The international conference on Intelligent systems and Pattern recognition. Virtual Event Tunisia: ACM, Oct. 16, 2020, pp. 45–49. ISBN: 978-1-4503-7502-3. DOI: 10.1145/3432867.3432889. URL: https://dl.acm.org/doi/10.1145/3432867.3432889 (visited on 05/19/2022).

[8] *ClearOS OS for your Server, Network, and Gateway Systems.* URL: https://www.clearos.com/ (visited on 05/28/2022).

[9] Roger L Davis, Dwayne Williams, and Chris Crayton. "Principles of Computer Security: CompTIA Security+ and Beyond, Sixth Edition." In: (), p. 1074.

[10] Polash Dey et al. "Comparative Analysis of Recurrent Neural Networks in Stock Price Prediction for Different Frequency Domains." In: *Algorithms* 14.8 (2021), p. 251. DOI: 10.3390/a14080251. URL: https://doi.org/10.3390/a14080251.

[11] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *arXiv:2010.11929 [cs]* (June 3, 2021). arXiv: 2010.11929. URL: http://arxiv.org/abs/2010.11929 (visited on 04/25/2022).

[12] Stuart Dreyfus. "The numerical solution of variational problems." In: *Journal of Mathematical Analysis and Applications* 5.1 (Aug. 1962), pp. 30–45. ISSN: 0022247X. DOI: 10.1016/0022-247X(62)90004-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0022247X62900045 (visited on 04/24/2022).

[13] "Enhanced Network Intrusion Detection using Deep Convolutional Neural Networks." In: *KSII Transactions on Internet and Information Systems* 12.10 (Oct. 31, 2018). ISSN: 19767277. DOI: 10.3837/tiis.2018.10.028. URL: http://itiis.org/digital-library/manuscript/2163 (visited on 05/12/2022).

[14] Kunihiko Fukushima. "Neocognitron for handwritten digit recognition." In: *Neurocomputing* 51 (Apr. 2003), pp. 161–180. DOI: 10.1016/S0925-2312(02)00614-8.

[15] James Graham. *Cyber Security Essentials.* 2010, p. 331. ISBN: 9781439851265.

[16] Xifeng Guo et al. "Deep Clustering with Convolutional Autoencoders." In: *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II.* Ed. by Derong Liu et al. Vol. 10635. Lecture Notes in Computer Science. Springer, 2017, pp. 373–382. DOI: 10.1007/978-3-319-70096-0_39. URL: https://doi.org/10.1007/978-3-319-70096-0%5C_39.

[17] Quentin Hardy. *Jeff Hawkins Develops a Brainy Big Data Company.* Bits Blog. Cad: 1 Section: Technology. Nov. 28, 2012. URL: https://bits.blogs.nytimes.com/2012/11/28/jeff-hawkins-develops-a-brainy-big-data-company/ (visited on 04/10/2022).

[18] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786 (July 28, 2006), pp. 504–507. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1127647. URL: https://www.science.org/doi/10.1126/science.1127647 (visited on 06/11/2022).

[19] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. “A Fast Learning Algorithm for Deep Belief Nets.” In: *Neural Comput.* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: https://doi.org/10.1162/neco.2006.18.7.1527.

[20] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[21] Sepp Hochreiter and Jürgen Schmidhuber. “LSTM can Solve Hard Long Time Lag Problems.” In: *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996.* Ed. by Michael Mozer, Michael I. Jordan, and Thomas Petsche. MIT Press, 1996, pp. 473–479. URL: http://papers.nips.cc/paper/1215-lstm-can-solve-hard-long-time-lag-problems.

[22] *Home - Suricata.* URL: https://suricata.io/ (visited on 05/28/2022).

[23] Ahmad Javaid et al. “A Deep Learning Approach for Network Intrusion Detection System.” In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS).* 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). New York City, United States: ACM, 2016. ISBN: 978-1-63190-100-3. DOI: 10.4108/eai.3-12-2015.2262516. URL: http://eudl.eu/doi/10.4108/eai.3-12-2015.2262516 (visited on 05/21/2022).

[24] Jin Kim et al. “Method of intrusion detection using deep neural network.” In: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp).* 2017 IEEE International Conference on Big Data and Smart Computing (BigComp). Jeju Island, South Korea: IEEE, Feb. 2017, pp. 313–316. ISBN: 978-1-5090-3015-6. DOI: 10.1109/BIGCOMP.2017.7881684. URL: http://ieeexplore.ieee.org/document/7881684/ (visited on 05/09/2022).

[25] Henry J. Kelley. “Gradient Theory of Optimal Flight Paths.” In: *ARS Journal* 30.10 (Oct. 1960), pp. 947–954. ISSN: 1936-9972. DOI: 10.2514/8.5282. URL: https://arc.aiaa.org/doi/10.2514/8.5282 (visited on 04/24/2022).

[26] David Kim and Michael G. Solomon. *Fundamentals Of Information Systems Security.* 2nd. USA: Jones and Bartlett Publishers, Inc., 2013. ISBN: 1284031624.

[27] Yann Lecun. “Gradient-Based Learning Applied to Document Recognition.” In: *PROCEEDINGS OF THE IEEE* 86.11 (1998), p. 47.

[28] Hung-Jen Liao et al. “Intrusion detection system: A comprehensive review.” In: *J. Netw. Comput. Appl.* 36.1 (2013), pp. 16–24. DOI: 10.1016/j.jnca.2012.09.004. URL: https://doi.org/10.1016/j.jnca.2012.09.004.

[29] Marcus A. Maloof, ed. *Machine learning and data mining for computer security: methods and applications*. Advanced information and knowledge processing. OCLC: ocm61529247. London: Springer, 2006. 210 pp. ISBN: 978-1-84628-029-0.

[30] Warren S Mcculloch and Walter Pitts. "A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY." In: (), p. 17.

[31] *NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. URL: https://www.unb.ca/cic/datasets/nsl.html (visited on 03/21/2022).

[32] Baiju NT. *Machine Learning: The Complete History In A Timeline | RoboticsBiz*. Section: Machine Learning. URL: https://roboticsbiz.com/machine-learning-the-complete-history-in-a-timeline/ (visited on 04/24/2022).

[33] *OPNsenseőăa true open source security platform and more - OPNsenseő is a true open source firewall and more*. URL: https://opnsense.org/ (visited on 05/28/2022).

[34] *OSSEC - World's Most Widely Used Host Intrusion Detection System - HIDS*. OSSEC. URL: https://www.ossec.net/ (visited on 05/28/2022).

[35] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.html.

[36] *pfSenseő - World's Most Trusted Open Source Firewall*. URL: https://www.pfsense.org/ (visited on 05/28/2022).

[37] Rajat Raina et al. "Self-taught learning: transfer learning from unlabeled data." In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*. Ed. by Zoubin Ghahramani. Vol. 227. ACM International Conference Proceeding Series. ACM, 2007, pp. 759–766. DOI: 10.1145/1273496.1273592. URL: https://doi.org/10.1145/1273496.1273592.

[38] Shahadate Rezvy et al. "Intrusion Detection and Classification with Autoencoded Deep Neural Network." In: *Innovative Security Solutions for Information Technology and Communications*. Ed. by Jean-Louis Lanet and Cristian Toma. Vol. 11359. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 142–156. ISBN: 978-3-030-12941-5 978-3-030-12942-2. DOI: 10.1007/978-3-030-12942-2_12. URL: http://link.springer.com/10.1007/978-3-030-12942-2_12 (visited on 05/21/2022).

[39] S. Rivard. *Le développement de systèmes d'information: Une méthode intégrée à la transformation des processus, 4e édition*. Presses de l'Université du Québec, 2013. ISBN: 9782760537002. URL: https://books.google.dz/books?id=NVonDwAAQBAJ.

[40] Trevor Thornton Ross. "The synthesis of intelligence–its implications." In: *Psychological Review* 45 (1938), pp. 185–189.

[41] F. Sabahi and A. Movaghar. "Intrusion Detection: A Survey." In: *2008 Third International Conference on Systems and Networks Communications*. 2008 Third International Conference on Systems and Networks Communications. Sliema, Malta: IEEE, 2008, pp. 23–26. ISBN: 978-0-7695-3371-1. DOI: 10.1109/ICSNC.2008.44. URL: http://ieeexplore.ieee.org/document/4693640/ (visited on 04/21/2022).

[42] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers." In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[43] Iqbal H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions." In: *Sn Computer Science* 2.3 (2021), p. 160. ISSN: 2662-995X. DOI: 10.1007/s42979-021-00592-x. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7983091/ (visited on 04/26/2022).

[44] *Security Event Manager - View Event Logs Remotely | SolarWinds*. URL: https://www.solarwinds.com/security-event-manager (visited on 05/28/2022).

[45] Terrence J. Sejnowski and Charles R. Rosenberg. "NETtalk: A Parallel Network That Learns to Read Aloud." In: (1986).

[46] *Sepp Hochreiter's Fundamental Deep Learning Problem (1991)*. URL: https://people.idsia.ch/~juergen/fundamentaldeeplearningproblem.html (visited on 04/25/2022).

[47] *Snort - Network Intrusion Detection & Prevention System*. URL: https://www.snort.org/ (visited on 05/28/2022).

[48] Tuan A Tang et al. "Deep learning approach for Network Intrusion Detection in Software Defined Networking." In: *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). Fez, Morocco: IEEE, Oct. 2016, pp. 258–263. ISBN: 978-1-5090-3837-4. DOI: 10.1109/WINCOM.2016.7777224. URL: http://ieeexplore.ieee.org/document/7777224/ (visited on 05/09/2022).

[49] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set." In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). Ottawa, ON, Canada: IEEE, July 2009, pp. 1–6. ISBN: 978-1-4244-3763-4. DOI: 10.1109/CISDA.2009.5356528. URL: http://ieeexplore.ieee.org/document/5356528/ (visited on 05/24/2022).

[50] Gael Kamdem De Teyou and Junior Ziazet. "Convolutional Neural Network for Intrusion Detection System In Cyber Physical Systems." In: (), p. 7.

[51] Mr Shailesh P Thakare, Prerana Chandurkar, and Maithili S Deshmukh. "Computer Attacks and Intrusion Detection System: A Need Review." In: *Open Access* 6 (2013), p. 12.

[52] Ankit Thakkar and Ritika Lohiya. "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions." In: *Artif. Intell. Rev.* 55.1 (2022), pp. 453–563. DOI: 10.1007/s10462-021-10037-9. URL: https://doi.org/10.1007/s10462-021-10037-9.

[53] François Thill. "Information Security Risk Management." In: *Privacy and Identity Management. Between Data Protection and Security - 16th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Privacy and Identity 2021, Virtual Event, August 16-20, 2021, Revised Selected Papers*. Ed. by Michael Friedewald et al. Vol. 644. IFIP Advances in Information and Communication Technology. Springer, 2021, pp. 17–22. DOI: 10.1007/978-3-030-99100-5_2. URL: https://doi.org/10.1007/978-3-030-99100-5%5C_2.

[54] A. M. Turing. "I.COMPUTING MACHINERY AND INTELLIGENCE." In: *Mind* LIX.236 (Oct. 1, 1950), pp. 433–460. ISSN: 1460-2113, 0026-4423. DOI: 10.1093/mind/LIX.236.433. URL: https://academic.oup.com/mind/article/LIX/236/433/986238 (visited on 04/24/2022).

[55] Ashish Vaswani et al. "Attention Is All You Need." In: *arXiv:1706.03762 [cs]* (Dec. 5, 2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 04/25/2022).

[56] Chuanlong Yin et al. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks." In: *IEEE Access* 5 (2017), pp. 21954–21961. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2762418. URL: http://ieeexplore.ieee.org/document/8066291/ (visited on 05/18/2022).