REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

INTRUSION DETECTION IN IOT NETWORK USING DATAMINING AND FEATURE SELECTION

Hasan Hameed Radhi AL-QARAGHULI

Master's Thesis

Supervisor

Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN

Istanbul, 2022

INTRUSION DETECTION IN IOT NETWORK USING DATAMINING AND FEATURE SELECTION

Hasan Hameed Radhi AL-QARAGHULI

Information Technologies

Master's Thesis

ALTINBAŞ UNIVERSITY

2022

The thesis titled INTRUSION DETECTION IN IOT NETWORK USING DATAMINING AND FEATURE SELECTION prepared by HASAN HAMEED RADHI AL-QARAGHULI and submitted on 13/12/2022 has been **accepted unanimously** for the degree of Master of Science in Information Technology.

Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN

the Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN	Department of Software Engineering, Altınbaş University	_____
Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Serdar KARGIN	Department of Biomedical Engineering, İstanbul Arel University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master's thesis.

Submission date of the thesis to the Institute of Graduate Studies: ___/___/___

I hereby declare that all information and data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Hasan Hameed Radhi AL-QARAGHULI

Signature

DEDICATION

I dedicate this research work to my supervisor who was for guiding me through the whole research work as well as my family for always assisting me in my difficult time.

PREFACE

First and foremost, I would like to thank my supervisor Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work.

ABSTRACT

INTRUSION DETECTION IN IOT NETWORK USING DATAMINING AND FEATURE SELECTION

AL-Qaraghuli, Hasan Hameed Radhi

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Ayça Kurnaz TÜRKBEN

Date: 12/2022

Pages: 74

The Internet of Things, or IoT for short, refers to any physical thing that has been equipped with the required technical capabilities to collect and distribute data through the internet. Individuals are placing themselves at risk of cyberattacks due to the fast-expanding number of security issues connected with the Internet of Things (IoT) and the rapid spread of IoT devices in household contexts. Due to the limited power and processing capabilities of these devices, it may be challenging to design a security solution for Internet of Things devices for this, it is necessary to take an interest in past attacks involving connected objects but also to be able to prevent new types of attacks. This thesis therefore proposes a solution capable of detecting numbers of types of intrusions in connected objects, in use of realizes techniques of tracing for recovery of precise information in loss systems of vigilance and in comparison, with various technical learning to achieve excellent results. Detection capabilities

Keywords: IDS, DDOS, BOTNET, ML, DL

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

AI	:	Artificial Intelligence
FSK	:	Frequency Shift Keying
IoT	:	Internet of Things
PAN	:	Personal Area Network
OFDM	:	Orthogonal Frequency Division Multiplexing
TDMA	:	Time Division Multiple Access

1. INTRODUCTION

1.1 BACKGROUND

The Internet of Things, or IoT for short, refers to any physical thing that has been equipped with the required technical capabilities to collect and distribute data through the internet. Individuals are placing themselves at risk of cyberattacks due to the fast-expanding number of security issues connected with the Internet of Things (IoT) and the rapid spread of IoT devices in household contexts. Due to the limited power and processing capabilities of these devices, it may be challenging to design a security solution for Internet of Things devices [2]. The usage of intrusion detection systems, often known as IDS, may be useful for providing some kind of cyberattack defense (IDS). There are two basic sorts of IDS: those that detect abnormalities and those that identify misuse. For misuse-based filters to identify new attacks, they must first generate a new attack signature before comparing incoming assaults to previously identified attack signatures. These filters will only then be able to recognize fresh assaults. Even while anomaly-based detection, which identifies new risks by observing abnormal traffic, has the ability to uncover new threats, one of the greatest obstacles is ensuring that it does so without creating an excessive number of false positives. The great majority of devices linked to the Internet of Things are constrained and have limited storage, memory, and computing capabilities, making the building of an IDS a difficult task [5].

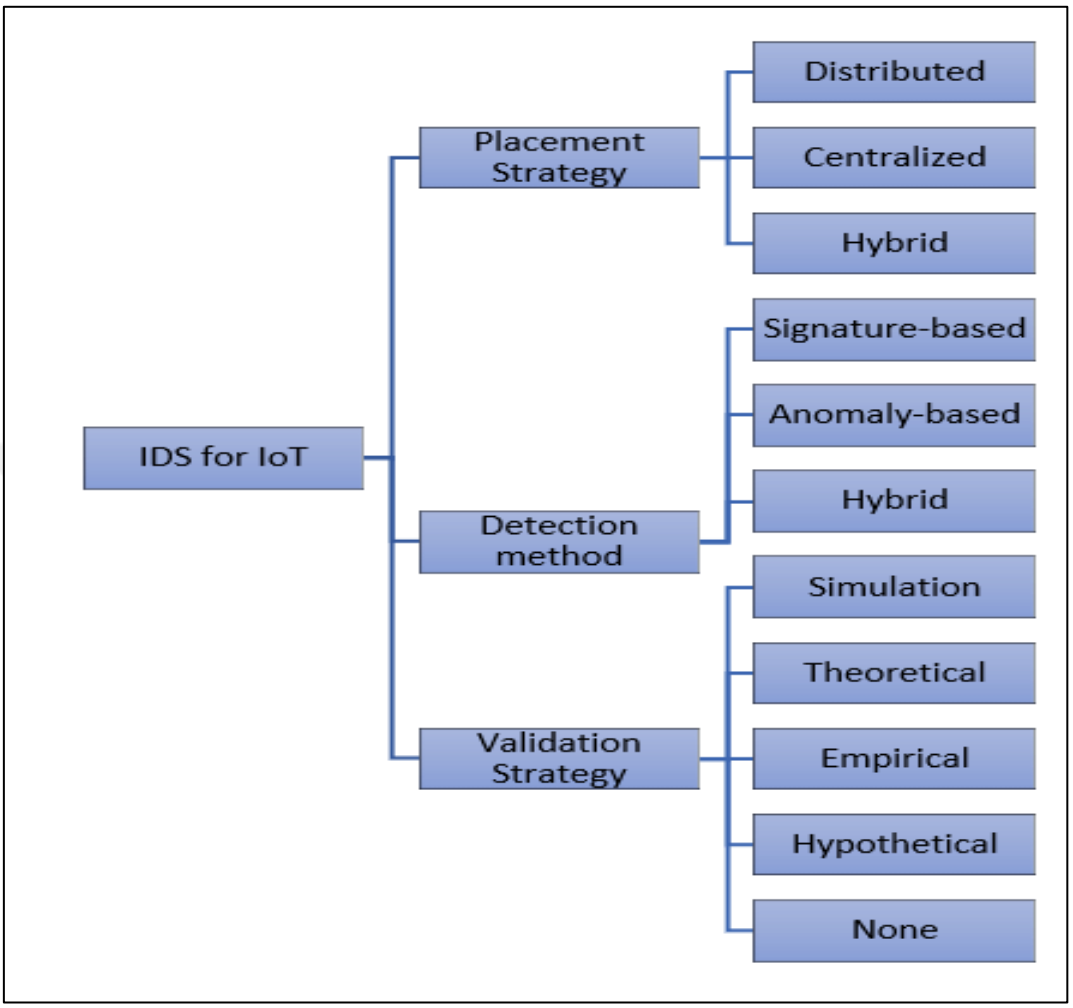

Figure 1.1: Intrusion detection types.

1.2 PROBLEM STATEMENT

Strong competition from manufacturers of connected objects has created a market where many technologies coexist. These objects can take various forms, ranging from alarm or light management systems to ovens or connected toys. They sometimes have extremely limited resources, making it impossible to implement traditional security solutions. The diversity of protocols existing in this ecosystem as well as the absence of an operating system or firmware common to these platforms make the work aimed at securing such objects very dependent on the systems studied. In addition, few manufacturers offer updates for their devices, making discovered vulnerabilities very dangerous for users.

Thus, it becomes difficult to propose security solutions applicable to all objects, and the work carried out in terms of intrusion detection essentially concentrates on network analyses, since there is no need to consider the material resources of the objects nor even the applications which are executed on them. Moreover, although tracing techniques have already been used to detect anomalies on different systems, particularly with the help of system calls made on the monitored machine, few works have focused on all the information available in the events. Thus, the system calls arguments or the various information at the level of the network packets or the processor have hardly been used to detect computer attacks, whereas they can prove invaluable to improve the quality of the detection.

For example, an event related to the launching of a terminal can be benign or harmful for the system depending on the process that is at the origin of this system call. It is therefore appropriate to use all the information collected by the trace points to improve the quality of detection. In addition, few works have proposed tool suites for exploiting traces in a binary format to train machine learning algorithms. Indeed, to achieve such a result, several stages of transformation of the events collected with tracing techniques must be implemented. The challenges to achieve this result are multiple. The first is to determine the event fields useful for training the machine learning algorithms, which can be very different between the various events that can be recorded by the tracer. Another challenge is to harvest these fields efficiently from the formats where the traces are stored, particularly binary formats, and to create other metrics useful for learning algorithms. It is also necessary to convert all these data into vectors of figures which will then be usable for learning the models.

Finally, it is necessary to study the effectiveness of various machine learning approaches developed to improve the detection capabilities for the proposed solution. In fact, each algorithm can give different results depending on the dataset it receives or the use of the model once its training is complete. For example, some algorithms may perform best at classifying sequences of text but may be much less optimal at classifying single words. It is therefore necessary to critically study the behaviour of different learning techniques in the case of intrusion detection on connected objects, with traces relating to the system itself.

1.3 CONTRIBUTIONS

There is no impenetrable computer system, and attacks on these platforms continue to increase in name and intensity every day. At the same time, more and more systems are interconnected, and this phenomenon is accentuated with the advent of the Internet of Things, which is made up of innovative objects but very rarely integrating effective protections against attack of cyber criminals. They threaten their users, whether they are individuals or companies. It is therefore necessary to find solutions capable of effectively securing these objects while respecting their hardware and software constraints. One of the first steps to improving the security of such devices is to be able to detect attack attempts as soon as they occur. For this, it is necessary to take an interest in past attacks involving connected objects but also to be able to prevent new types of attacks. This thesis therefore proposes a solution capable of detecting numbers of types of intrusions in connected objects, in use of realizes techniques of tracing for recovery of precise information in los systems of vigilance and in comparison, with various technical learning to achieve excellent results. Detection capabilities.

1.4 OBJECTIVES

The main problem of this thesis is the following: Is it possible to implement an intrusion detector based on the behaviour of connected objects that is effective and based on tracing and machine learning techniques? To be able to answer this problem, it is necessary to achieve the following objectives:

i. Understand the characteristics of the ecosystem of connected objects.

ii. Develop an architecture allowing the tracking of intelligent objects taking into account their characteristics.

iii. Develop a framework capable of transforming the information contained in the traces into precise information that machine learning algorithm can employ.

iv. Implement and optimize various machine learning algorithms capable of classifying recorded events.

v. Identify and implement attacks on the connected objects studied.

vi. Test the performance of the solution and the different algorithms on this system.

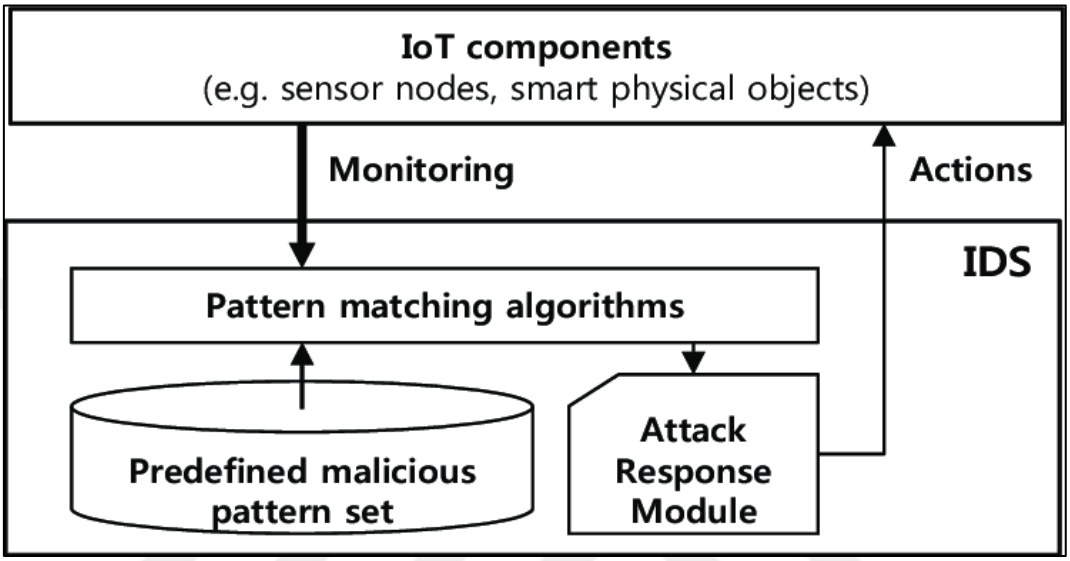

Figure 1.2: IoT based intrusion detection system.

1.5 THESIS ORGANIZATION

This thesis is organized as follows: - In 2nd Chapter of this thesis presents a critical review of the literature on the various concepts useful during this work. The topics of the Internet of Things, intrusion detection systems and different tracking techniques will be discussed. Then, 3rd chapter will present the background methodology adopted in our work to set up and validate the proposed solution. Also, 4th Chapter presents the system in which the developed solution is presented as well as the criticism of its performance. The following chapter evaluates the success of answering the research question. The conclusion of the work as well as the discussion of future work are included in 6th Chapter.

2. LITERATURE REVIEW

The Internet has never connected as many computer systems as today, and the number of objects capable of communicating with each other through this network continues to increase every day. This trend even tends to accelerate with the ever-faster addition of new objects that are constantly diversifying. Although fierce competition between object designers brings a lot of innovation, it also forces manufacturers to reduce the development time of their products and therefore to reduce the security test phase of objects placed on the market. Thus, many vulnerabilities are discovered on smart objects and the associated risks are constantly growing for users. It therefore becomes necessary to be able to secure these different objects, the first step being to be able to detect these malicious activities. Although various works have been proposed, many solutions remain ineffective or very object specific. The following critical review of the literature highlights the different approaches proposed to detect intrusions as well as different techniques that could be used to achieve better results.

2.1 RELATED WORKS

In tests done by Notra et al. (2019) on three common smart home devices, the researchers discovered a lack of encryption and authentication, putting the privacy of the users at risk. Due to the huge diversity of protocols and communication stacks involved, the limited computer capacity, and the vast number of networked devices, IoT security countermeasures cannot be implemented effectively. Therefore, Internet of Things security solutions are needed if enterprises and individuals are to maximize the technology's potential (Sicari et al., 2021).

Multiple efforts are now being done to enhance the security of the Internet of Things. The creation of authentication and data confidentiality techniques, access control inside the IoT network, privacy and trust between users and objects, and the enforcement of security and privacy legislation are a few of these strategies. (Sicari et al., 2021). Even with these precautions in place, networks linked to the internet of things (IoT) remain susceptible to attack from a variety of sources. This is an essential factor to remember. As a result, an additional layer of protection that can identify possible threats is vital. IDSs, or Intrusion Detection Systems, are used for this purpose.

The move from wired to wireless networks may be difficult owing to basic design difficulties, most notably the absence of established infrastructure. Regarding wireless ad hoc networks, the sort of response relies on a number of variables, including the type of intrusion response, the network protocols and applications presently in use, and an individual's level of confidence in the evidence. Among the several possible solutions include reinitializing communication channels between devices, determining which devices have been hacked, and sending an authentication request to every device on the network.

The authors elaborate on seven distinct IDS ideas for MANETs and use the following discussion strategies: Implementing both mobile agent-based detection and distributed anomaly detection. In any scenario, the IDS agent on each mobile node will collect and classify any locally accessible data. In terms of intrusion detection and response, mobile-agent-based detection relies on the employment of mobile agents, while distributed anomaly detection requires input from neighbouring nodes to achieve global detection. Anantvalee and Jie examine the network architecture of IDS on MANETs (2017). The authors examine several possible configurations for systems to detecting intrusions in MANETs, such as a flat-net architecture, a multi-layered network infrastructure, and a mobile intrusion detection agent. In addition, the authors discuss a collaborative and decentralized intrusion detection system (flat and multi-layered network infrastructure).

The authors of the research revealed that practically all of the IDSs they examined were distributed and cooperative, which is not unexpected given that MANETs are characterized by their distributed and cooperative architectures. The authors also provide a taxonomy for recognizing and punishing problematic nodes in MANETs. This taxonomy takes into account the network architecture in addition to the kind of data collected and sent. The major focus of Kumar and Dutta's (2020) examination of MANET intrusion detection approaches is detection algorithms. A tree structure is used to categorize different methods of intrusion detection according to the kind of processing mechanism in use.

The most prevalent methods for detecting intrusions include statistical, heuristic, rule-based, state-based, signature-based, reputation-based, routing information-based, cross-layer, and

graph theory-based methods. The authors classified every investigated intrusion detection method according to the following criteria in great detail: detection method (misuse, anomaly-based, specification, or hybrid), architecture (standalone, distributed and cooperative, mobile agent-based, and hierarchical IDS), detection time (real-time or offline), routing protocol, type of attack addressed, performance, effect of mobility on robustness, flexibility, and scalability, and so on. In addition, research concerns and unanswered questions in the area of MANET intrusion detection are discussed. Managing an environment that is in a constant state of change is a difficult task. Individuals, systems, and networks have shown varied degrees of intrusive and non-invasive behaviours throughout history. The intrusion detection system (IDS) must be capable of self-management and self-configuration in order to keep up with the ever-changing dynamic environment and to respond swiftly to continually changing network hardware and software sources. There are three choices available for the deployment of the IDS agent in WSNs: distributed-centralized, distributed-only, and distributed-only. The vast majority of deployments are distributed-only.

The authors Farooqi and Khan (2019) propose a classification of intrusion detection systems (IDS) for wireless sensor networks (WSNs) based on the deployment method of the IDS agent: (IDS agent is installed in some monitor nodes). Additionally, they analyze the link between the location of the IDS agent in the WSN and its energy use.

A distributed central IDS is a better match for WSNs in terms of power consumption and network topology complexity, according to the results of the study. The three types of intrusion detection systems that may be used to WSNs are abuse detection, anomaly detection, and specification-based detection (Abduvaliyev et al. 2019). This article also examines in depth IDS techniques for WSN structure, with a focus on a number of topics that are still in the early stages of development. Problematically, there are no real-world implementations of intrusion detection systems (IDS) in WSNs. Another issue is the need to provide IDS techniques that are compatible with the Internet of Things (IoT).

In light of this, despite the fact that IDS for WSN has undergone significant development in recent years, several study topics (such as the design of an IDS, the equilibrium between

accuracy and resource consumption, and the improvement of the integration of the underlying mechanism) still require further development. Butun et al. (2014b) conduct a thorough evaluation of the literature on IDS for WSNs. This review incorporates several sources. They present a succinct description of IDSs suggested for MANETs and analyze whether or not these IDSs may be applied to WSNs. Due to the special design constraints of WSNs, there are IDSs that could be used in WSNs without any adjustments (two concepts), IDSs that could be modified significantly (seven ideas), and IDSs that could not be used at all (eight suggestions). In their research, the authors also propose a comparison of WSN IDSs based on the network architecture and detection method of each system. This study focuses on the energy consumption of IDSs as opposed to WSNs since WSNs have relatively low power consumption requirements. According to the findings of Modi and colleagues, cloud computing's availability, secrecy, and integrity have all been compromised (2019).

Cloud-based intrusion detection systems are classified according to IDS technology, detection strategy, and network location (HIDS, NIDS, Hypervisor-based intrusion detection system, and Distributed intrusion detection system). Each recommendation is rated based on its good and negative characteristics, and the obstacles that must be surmounted to make cloud computing a reliable platform for the delivery of Internet of Things services are identified and discussed. When it comes to cloud computing, the bulk of recommended solutions for intrusion detection fall short when it comes to mitigating recurring attacks, such as VM or hypervisor attacks and insider threats. According to Mitchell and Chen (2020), critical infrastructure protection systems (CPSs) are vast, geographically distributed, federated, and heterogeneous life-critical systems comprised of sensors, actuators, control, and networking components.

2.2 CHAPTER CONCLUSION

The authors classify modern IDSs for IPSs according to two design dimensions: the detection method and audit material (host based or network based). These researchers begin by examining in detail the distinctions between ordinary intrusion detection systems and those created specifically for IPSs. These distinctions include a focus on monitoring physical processes, complicated assaults, and legacy technology. Following this is a review of the most current

research on intrusion detection systems (IDSs) for the design of computer intrusion prevention systems (IPS) in terms of attack type, audit features, and dataset quality for the IPS application. In addition, they discuss potential advancements and difficulties pertaining to IDSs for IPSs. IDS solutions that are uniquely adapted to address the challenges provided by the Internet of Things have not yet been the topic of any research (IoT). This survey article examines internet of things-specific IDS installation and detection techniques (IoT). In addition, we discuss the prevalent security risks connected with the IoT and the ways in which IDSs may aid in recognizing and avoiding these dangers. In addition, we present an overview of the key validation techniques utilized in IoT intrusion detection algorithms, as well as a discussion of current research issues and future developments in the field.

3. STATE OF THE ART

In this chapter, we present the work environment used in our research work. We also detail the history of the methodology used to validate the effectiveness of our solution. Thus, we detail the software characteristics of the different platforms used as well as the different open-source projects necessary for the implementation of our solution. The operation of the tools we offer is detailed in the following chapter.

3.1 COMPUTER NETWORKS

In the 1970s, there was a convergence between computers and communications, which resulted in a major change in technology, products, and enterprises, all of which have since devoted themselves to the computer and communications sectors. This change happened due to the compatibility between computers and communications. The emergence of the area of data transmission and computer networks was a direct result of the aforementioned causes. Every communications system and computer network is designed to ease the flow of information. In a model of a data transmission system, the following elements are included:

Transmission systems may be as basic as a transmission line or as complex as an entire network connecting the signal's destination and source. The transmission system sends a signal to the receiver, which receives it, where it undergoes alteration, and is then made appropriate for hand transmission. The data provided by the source is encoded and converted by the transmitter, which subsequently generates electromagnetic signals.

Despite its apparent simplicity, the model that was just shown is quite intricate. Consider the transmission mechanism, interface, signal, and synchronization while developing a communications system. Includes subjects such as communication, data exchange management, error detection and repair, data flow control, addressing and routing, recovery, message format, and network security and management. Before data can be moved between computers and other processing equipment, there is a long procedure that must be accomplished. Consider the scenario in which you try to move a file from one machine to another. In this instance, linking the two systems will need the deployment of a communications network; however, the source

system must also activate a direct data channel or provide identification to the communications network. Before proceeding with the procedure, the destination system must be prepared to accept the data. If there is a manager program on the target system that can accept and save this file for the stated user, there must be a check to identify whether or not there is one, and if there is, one of you must perform a translation operation between the two systems. It has been found that all participating computers must interact effectively. Instead of implementing all communication logic in a single module, the issue is separated into smaller tasks, each of which is completed independently. Each component of a protocol architecture is placed vertically, forming a stack. Each individual layer of the stack is responsible for handling the interdependent activities that must be accomplished in sequential sequence in order to connect with another computer system. The most important functions are often delegated to the layer below, and individuals in charge of the present layer have a tendency to overlook the details of their job. Each layer gives extra services to the layers above it, which it also provides. The optimal situation when designing layers is one in which modifications to one layer have no effect on the others. The three basic categories for categorizing networks are wide area networks (WAN), local area networks (LAN), and metropolitan area networks (MANs). WANs are wide area networks (LAN). The abbreviation MAN stands for "Network." As a direct consequence of the convergence of technology and its possible applications, both groups are having a harder time distinguishing themselves from one another.

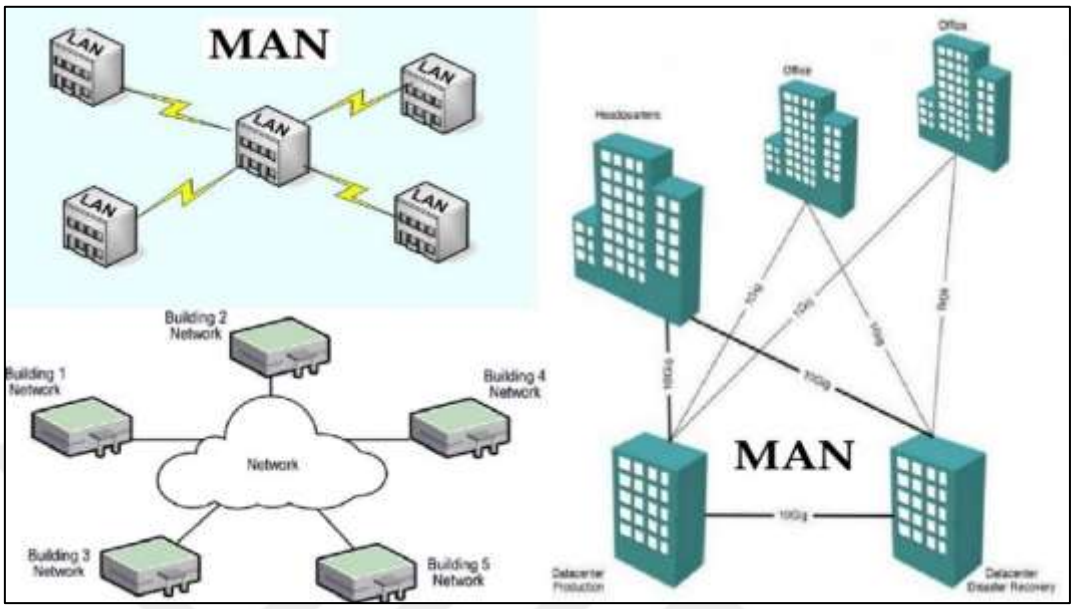

Figure 3.1: LAN WAN and MAN.

3.2 WIRELESS NETWORKS

Wireless local area networks have emerged as the main force in the LAN market in recent years. Wireless local area networks (LANs) are beginning to be acknowledged as a vital addition to wired networks in order to meet the mobile and migrating needs of organizations and to offer coverage in regions where wires cannot be deployed. This realization is starting to occur in corporations. A wireless local area network (LAN) transmits data via an inherently insecure medium, such as radio waves. Wireless local area networks (LANs) have only recently acquired popularity owing to their exorbitant cost, sluggish transmission rates, security risks, and licensing need. Due to the resolution of these issues, wireless local area networks have seen a stratospheric growth in popularity. In order to ease connectivity with other networks, a typical wireless local area network (LAN) will include an Ethernet backbone connecting a number of servers and workstations to one or more bridges or routing devices. Connecting the device to the wireless network is an additional control module (CM) that serves as an interface. When the wireless LAN is attached to the trunk, the control module works as a bridge or routing device. Note that some of the final systems, such as workstations and servers, are autonomous

components that can operate alone. Multiple stations may be considered part of a wireless network if they are administered by a single hub or another user module (UM).

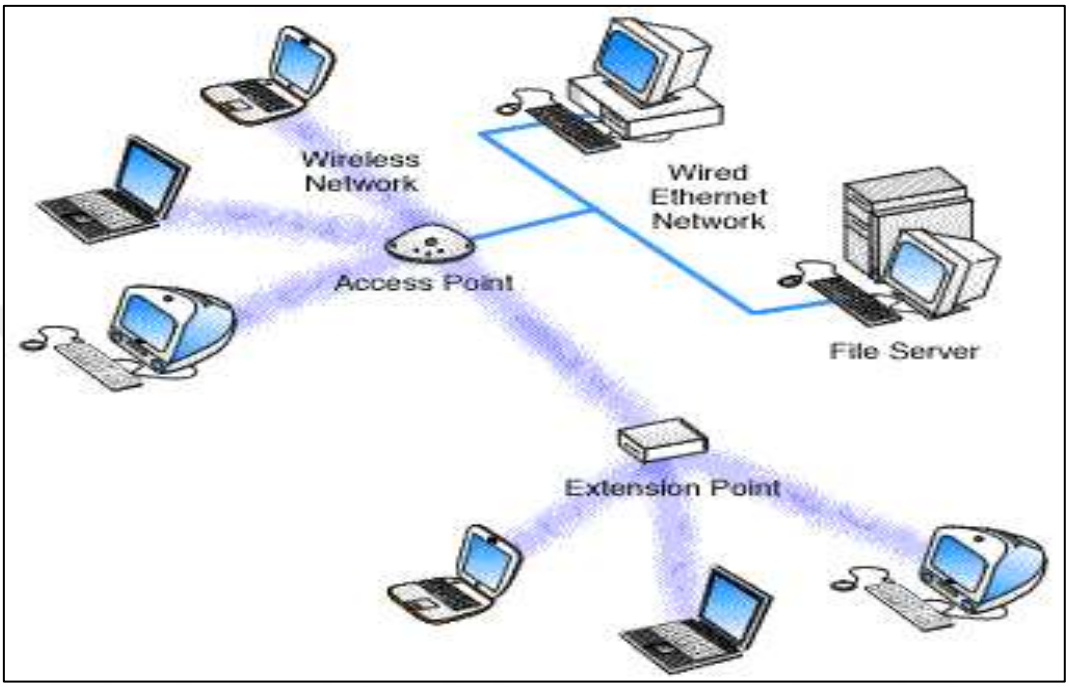

Figure 3.2: Wireless LAN.

3.3 INTERNET OF THINGS (IOT)

3.3.1 History of IOT

Technological developments in recent years have made it possible to imagine a new way of connecting not only people, but also the objects they use. The first machine to implement this idea was a soda dispenser installed at Carnegie Mellon University (CMU) in the computer science department in 1982. By interrogating it remotely via its IP address, it was possible to know if there were any cans left in the machine. The Internet of Things (IoT) or Internet of Things (IoT) was born, although the term was to appear later, in 2002, in an article published in the Forbes newspaper written by Chana R. Schoenberger. Since then, a large number of objects have acquired addresses to communicate and the term IoT has been adopted by them without an exact definition. This is indeed more of an idea than a physical reality. However, this idea has developed: to connect not only people, but things to the Internet. The motivation associated with

everything connected relates to the quantity of recoverable information making it possible to arrive at a realistic modelling of the world at all times. This modelling combined with the analysis of the data shared by these objects leads to the automatic triggering of actions that act on the world through other objects. For example, a set of sensors monitoring the state of road traffic in real time would control the alarm clock of a home to adapt the alarm time according to the affluence. Often heralded as "the new Web revolution", or "Web 3.0", the IoT has it all. However, if the original concept is certainly interesting, today we are far from an ecosystem as automated and unified as the one initially imagined. Currently, the majority of IoT developments consist of adding communication capabilities to an object that initially lacks them.

Figure 3.3: First IOT machine at Carnegie Mellon University (CMU).

This object can then upload data to a server or receive commands to interact with the environment in which it is installed. This does not allow it for the moment to be autonomous and to initiate communications with other objects, on its own, in order to participate in a more complex or generalized treatment. We are therefore seeing a multitude of small environments appear, which we could associate with local networks of objects rather than a real Internet of Things. The only difference with a traditional local network is that a large majority of these objects store their data on servers in the cloud and therefore on the Internet. These servers often

offer complete access to this information, to the detriment of its protection. In addition, if the processing and storage of data in the cloud are now fully exploited and functional, the technologies of interconnection and exchanges between objects are still far too heterogeneous for a future unification to be plausible. Indeed, to allow hundreds of thousands of different objects to communicate, a complete standard would have to be imposed, specifying in particular the communication protocols and the form of the data, or even the hardware architecture of these objects. However, the trends show rather the opposite. This lack of standardization in these means of communication makes it difficult to standardize exchanges and greatly limits the possible interactivity. However, the economic interest associated with the possibility of acquiring information from data collected by hundreds of thousands of objects is pushing manufacturers to develop many connected objects: according to the Institute of Audio-visual and Telecommunications in Europe (IDATE) the number of these objects was estimated at 42 billion in 2015. Indeed, all environments increasingly integrate these different objects to meet specific needs: homes, businesses, industries, cities or public spaces. In conclusion, rather than an Internet of Things, we consider more current the notion of IoT environment or connected environment. This is defined as a bounded environment composed of objects, called connected objects, having the possibility of exchanging information with other objects in the environment directly or via the Internet. The characteristics of these objects, their roles, as well as their means of communication depend on the context in which they are used.

3.3.2 IOT Evolution

All of these environments therefore have many advantages in being composed of multiple connected objects, or even complete connected solutions to ensure a certain level of comfort. When the design of an environment is done by thinking beforehand about the integration of these objects, it is quite possible to provide and offer strong connectivity, which can lead, depending on the use and the needs expressed, to "intelligent" environments. However, the methods of communication, and in particular the evolution of their uses, make today infinitely more complex the realization of a stable connected space, and more generally of an Internet of Things. Indeed, when historically, the majority of data exchanges were carried out via wired connections, a connected object 0was a fixed machine, easy to control and identify.

In addition, the implementation of a wired architecture being costly and restrictive, the number of these machines was more limited. The environments were therefore stable and the different objects of a network more clearly identifiable. However, with the development of Wi-Fi in 1997 and the appearance of mobile communications protocols, allowing Internet access from anywhere, the number of connected machines began to explode. Data exchanges have therefore become more mobile and dynamic, while multiplying exponentially. In addition, in recent years, this expansion in the number of connected objects has also encountered a proliferation of wireless communication methods offered, each offering its own characteristics, as we will see in the following subsection. However, the use of many heterogeneous wireless technologies makes these environments much more unstable, and therefore, difficult to model and understand. Achieving full connectivity in these spaces has therefore become more complex.

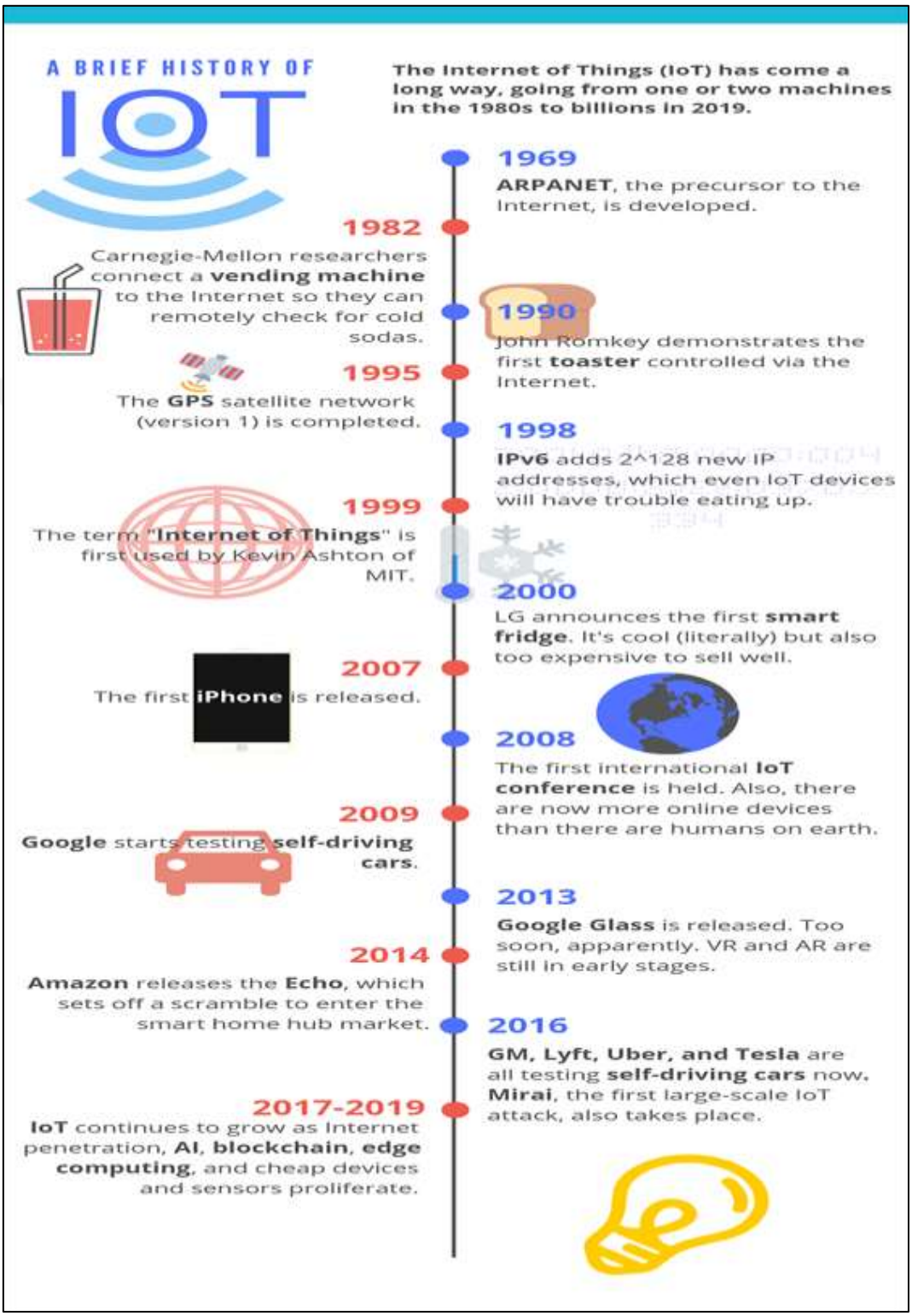

Figure 3.4: Flowchart of IOT evolution.

3.3.3 Protocols and Layers of Communications

This part seeks to give review of this heterogeneity of the means of communication present in the IoT. First, we present the definition of a communication protocol and the principle of standardization, then we detail the differences between wired and wireless protocols, as well as their advantages and disadvantages. We will finally see an overview of those implemented in the IoT.

i. IOT LAYERS

In the early 1970s, the development of experimental networks, in particular with ARPANET was immediately followed by the implementation of heterogeneous networks by computer manufacturers. Each used their own conventions to interconnect their equipment, thus presenting their network architecture. However, the need to interconnect the systems of different manufacturers was quickly felt, and a working group (SC16) was therefore set up by the International Organization for Standardization (ISO). Its objective was to propose a set of standards governing the interconnection of open systems. This group proposed in 1979 an architecture model called OSI model describing the different standards governing the implementation of an interconnection architecture for open systems to communicate information This now well-known model works with a seven-layer structure, allowing in particular to segment the interconnection problem into simple sub-parts. The principle of this model is as follows: each layer N is independent of the layers (N −1) and (N +1) located respectively below and above it. Each entity provides the higher entity with services up to the highest layer, which makes it possible to provide high-level application services distributed over all the machines in the network. The transmission of data takes place via a physical interconnection medium which can be a radio or wired component. This model is embedded on each system in the network, and each layer N communicates with the layer N of the other systems. Each of the layers therefore has a role that is specific to it, defined as follows in the standard:

i. Physical layer: It provides the different procedures allowing the use of the physical interconnection medium. It manages the transmission of physical data on the medium.

ii. Link Layer: Provides the elements for establishing, maintaining, and releasing link-level connections between interconnected systems. It also manages error correction that may occur on the physical link. The messages exchanged between two link layers are called frames.

iii. Network Layer: It provides the functional elements for exchanging information between two machines through a network connection. In particular, manages the routing elements used to transport layer messages through the network.

iv. Transport layer: It provides the services making it possible to manage the exchange between the entities sending and receiving information, without considering the intermediate elements making it possible to bring this data, which are managed by the network layer.

v. Session layer: It provides the elements allowing the exchange of distributed activities while respecting the synchronization and the topology of the network. It manages the elements of sessions established between two entities.

vi. Presentation layer: It provides the elements allowing to standardize the form of the exchanges, to allow the interpretation of the data by the application layer. It manages the display and control of the data structures exchanged.

vii. Application layer: This is the highest layer that provides services to the end user, particularly in the form of applications distributed on the network. The other layers make it possible to support the services of this layer for the user. To communicate and define the specificities and the way in which the services of this layer are rendered, the OSI model also defines the notion of standard or normalized protocol. This notion defines the implementation elements making it possible to provide the services of the associated layer.

Figure 3.5: Abstract representation of the IOT Layers.

A few years after the definition of the OSI model, another very similar architecture was proposed to allow the interconnection of open systems: TCP/IP At the same time, it proposes the definition of a set of standard protocols for implementing this model in systems. The major difference is the consolidation of the presentation and session layers into the application level, thereby reducing the complexity of the five-layer model. This model was quickly adopted, and the TCP/IP protocol stack was integrated everywhere, until today when it is the basis of the operation of the Internet. Subsequently, many protocol standards were defined, making it possible to perform the services of the different layers by meeting specific needs, for example by seeking to reduce energy consumption. The emergence of many wireless protocols in particular, and therefore based on specific low-level / hardware layers (physics, link), has now greatly contributed to making environments composed of connected objects much more complex.

ii. IOT PROTOCOLS

In all the environments previously described, many protocols have gradually imposed themselves, since they offer ease of integration of new objects without modifying the network architecture. The objective of this subsection is therefore to briefly present the wireless protocols used as well as their integration into the previously described connected environments, as well as their interests. We are not interested here in wired protocols, which are already widely studied in the literature and less present in IoT environments. We identify two types of wireless protocols: centralized protocols and decentralized or ad hoc protocols. Generally, centralized protocols require the presence of an access point responsible for transmitting communications. This access point is therefore a central point of the network, through which exchanges pass. To sum up, each message transmitted by a network element is sent to the access point which then takes care of transmitting it to the right recipient, according to the information contained in it. On the contrary, in decentralized or ad hoc protocols each element of the network can exchange information with another element directly. As we will see later, this distinction complicates the observation of exchanges.

i. The best-known family of wireless protocols of the centralized type is undoubtedly Wi-Fi. This family of protocols is centralized, since it is based, in its original version, on the interconnection of the various elements with a point of access. Each element of the network is therefore connected to this access point, and the communication between two of them is carried out by passing the frames through it. This technology, which dates back several decades, is now widely used in the various environments described above. Another feature for this technology is ability to integrate different forms of access control depending on the spaces where it is installed. Indeed, all environments today are made up of Wi-Fi networks, whether companies, with distinctions for example between internal networks reserved for employees or public networks open to visitors, or public spaces, which very often offer free Wi-Fi areas open to everyone. The vast majority of homes today also use this technology. Naturally, connected objects also implement it, which allows them to quickly interconnect with existing Wi-Fi networks. The more recent versions of the Wi-Fi standard now also allow

decentralized alternative operations, but which are still not widely used. This family of protocols often uses the 2.4-2.5 GHz frequency band, but is also present with new specifications on the 5 GHz band. A recent specification (802.11ad) even presents the possibility of communicating using millimetre waves, i.e. by transmitting data on a band exceeding 30 GHz.

ii. Another well-known family of protocols are mobile phone networks, i.e. the protocols used to enable the exchange of data and communications between mobile devices such as telephones. These protocols have evolved through several "generations", which redefine the specifications of the protocol stack. To sum up, antennas are positioned on the territory, then each of them will cover a geographical area with which users within range will be able to communicate through them. The operation of communications is therefore well centralized. The objective of the next 5th Generation is also to converge the mobile infrastructure for connected objects, allowing them to interconnect over long distances These protocols communicate on various frequency bands which depend on the generations. The bands most used in recent systems are the 800-900 MHz, 1500 MHz, 1800 MHz, 2100 MHz and 2600 MHz frequencies.

iii. Still in the rather long-distance protocols, two fairly recent protocols offer a different vision of communications, rather focused on the episodic feedback of remote information. These two protocols are called LoRaWAN and SigFox, and both offer long-range but low-speed communications often categorized as LPWAN (Low-Power Wide-Area Network. The idea of these protocols is to be extremely inexpensive in power, by drastically limiting the flow, and therefore the quantity of information exchanged. For LoRaWAN, all the equipment communicates through gateways within range, on the same principle as mobile communications. The IP protocol is used to allow these gateways to communicate with each other via an application server. As for SigFox, it is a proprietary protocol operating on the same principle as LoRaWAN, with notable differences at the level of the low-level layers used. These protocols communicate via the frequency band of 868 MHz, which is a free-to-use radio band

Decentralized or ad hoc protocols IoT environments based on strong connectivity and high dynamism between the different elements that compose them, ad hoc protocols have interesting characteristics favouring their recent developments. The latter can indeed directly exchange information without going through a gateway that would require a specific infrastructure. With connected objects, several protocols have been designed, including Bluetooth and its Low-Energy or low-consumption variant, Bluetooth Low Energy (BLE). Present for a few years now on mobile phones which used it mainly for the exchange of short-range multimedia data, it has been developed in its low-power version for connected objects. Hundreds of objects therefore implement it, allowing phones equipped with this technology to communicate directly with them. Its operation at the physical layer level is quite particular, since it shares the 2.4-2.5 GHz band with Wi-Fi and other protocols. It uses frequency hopping modulation, which helps to avoid interference.

iv. Today, Bluetooth and BLE topology is often point-to-point. Zigbee is another decentralized protocol stack which offers a mesh network topology, based on the 802.15.4 standard specifying the physical and link layer. The objective is similar to BLE: to provide a low-power protocol for short-range communications. This protocol is widely used in modern home automation, since it makes it quite easy to interconnect a network of sensors in a home. It operates on different bands depending on its country of use, 868 MHz in Europe, 915 MHz on the American continent and Australia, it can also be used worldwide on the 2.4-2.5 GHz band. The examples presented here are only a tiny glimpse of the impressive quantity of protocols proposed for wirelessly interconnecting objects, but allow us to appreciate the great complexity and diversity of existing technologies. In parallel with all the protocols presented here, many of them are proprietary, their specification being unknown to the general public, like SigFox. These are often, as in the early days of open systems interconnection private companies that develop their own communication protocol for the objects they sell.

This is for example the case of Nest Labs, which uses a proprietary protocol called Thread Since the specifications of these protocols are not available to the general public, it is very difficult to understand how they work, leading to many interconnections and communication monitoring problems. To briefly assess the diversity of existing wireless protocols in the world, the Sigidwiki site lists all the identified and unidentified signals as well as more or less precise descriptions of their operation. Counting only the identified signals, there are already 379 different ones.

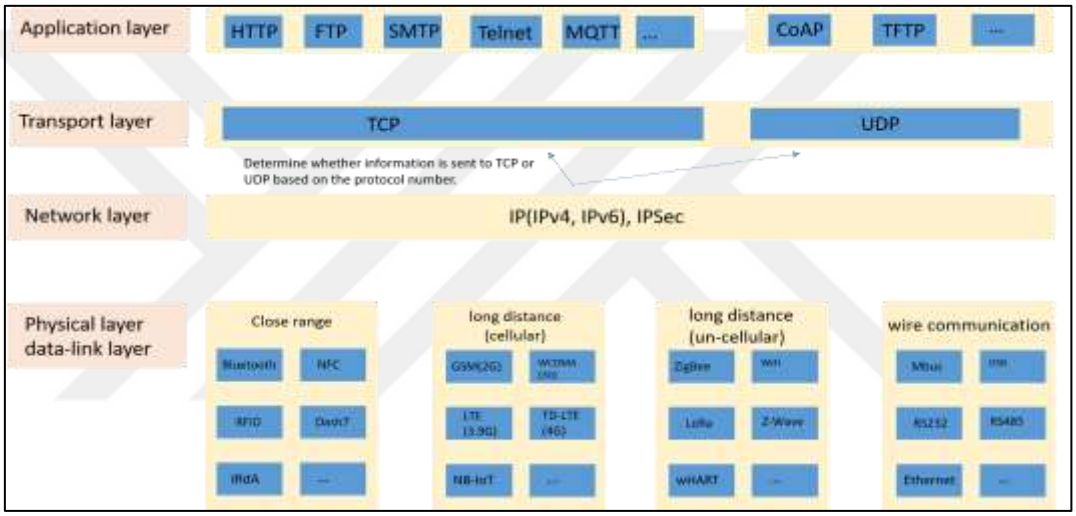

Figure 3.6: IoT protocols at each communication layer.

3.4 INTRUSION DETECTION SYSTEMS

An Intrusion Detection System, or IDS, is a system in charge of identifying, monitoring and responding to suspicious activities when there is communication between two or more machines. It collects data from a system, and when evidence is detected, a response protocol is initiated. We call HIDS an IDS that is executed on host machines (hosts), which monitors incoming and outgoing packets only from the device, alerting its administrator if any anomaly has been detected. HIDS analyzes and audits data provided by the operating system or by applications. Below we will describe how intrusion detection works according to the levels, the difference between IDS and the future trend of these tools.

3.4.1 Intrusion Detection at The Operating System Level

The information collected is extracted from file modifications, file records (logs), low-level operations, system calls... These data are usually difficult to manipulate unless the kernel of the operating system is attacked directly.

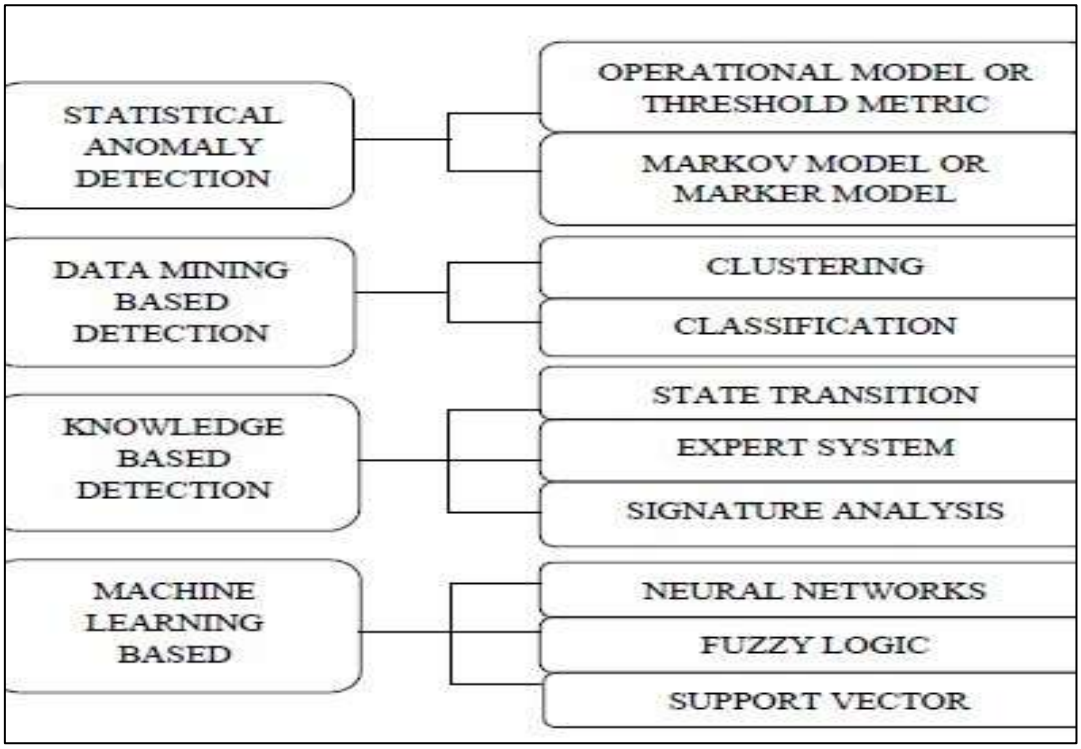

Figure 3.7: Types of intrusion detection systems.

i. Collection of Audited Data

It is essential that the data collected has not been modified, exposed or vulnerable. This is usually taken care of by the integrity of the operating system. The primary utility of the operating system is not intrusion detection, so the IDS is in charge of accessing the data of the operating system to give a use to the collected data. Each operating system has its own auditing mechanisms. SOLARIS, for example, is based on the Kernel BSM (Basic Security Module), which causes each call to be recorded in a log, with the respective user, and allows the compartment to be modified with root access. The Linux operating system has SNARE and LIDS, also at the Kernel level:

i. SNARE: Accesses the system call log files. It has a behavior pattern recognition module, as well as a graphic tool that filters the collected information.

ii. LIDS: It is not an IDS per se, but it facilitates access to the standard control layer. It is also a data collector but acts as file access control. Being a root user does not grant all privileges, so it does not compromise the integrity of the information. There is a log belonging to the operating system that has disadvantages if it is used directly for the extraction of relevant information, such as the storage of information with an unspecified or random format. For this reason, applying an analysis in an automated way, carrying out a statistical study and classifying presumed activities is complicated.

The Microsoft Windows operating system has an auditing system that is divided into three events:

i. System Log: Contains data on windows services and drivers. In addition, it contains system startup and error events. In a network environment, browser events are also generated.

ii. Security Log: Tracks logins and logouts, privilege changes, system startups and system shutdown.

iii. Application Log: These are events generated by applications. The three logs can be viewed through tools such as the Windows Event Viewer and accessible through the Windows32 API.

Windows Vista and above Events	
General Event Descriptions	**General Event IDs**
Account and Group Activities	4624, 4625, 4648, 4728, 4732, 4634, 4735,4740, 4756
Application Crashes and Hangs	1000 and 1002
Windows Error Reporting	1001
Blue Screen of Death (BSOD)	1001
Windows Defender Errors	1005, 1006, 1008, 1010, 2001, 2003, 2004, 3002, 5008
Windows Integrity Errors	3001, 3002, 3003, 3004, 3010 and 3023
EMET Crash Logs	1 and 2
Windows Firewall Logs	2004, 2005, 2006, 2009, 2033
MSI Packages Installed	1022 and 1033
Windows Update Installed	2 and 19
Windows Service Manager Errors	7022, 7023, 7024, 7026, 7031, 7032, 7034
Group Policy Errors	1125, 1127, 1129
AppLocker and SRP Logs	865, 866, 867, 868, 882, 8003, 8004, 8006, 8007
Windows Update Errors	20, 24, 25, 31, 34, 35
Hotpatching Error	1009
Kernel Driver and Kernel Driver Signing Errors	5038, 6281, 219
Log Clearing	104 and 1102
Kernel Filter Driver	6
Windows Service Installed	7045
Program Inventory	800, 903, 904, 905, 906, 907, 908
Wireless Activities	8000, 8001, 8002, 8003, 8011, 10000, 10001, 11000, 11001, 11002, 11004, 11005, 11006, 11010, 12011, 12012, 12013
USB Activities	43, 400, 410
Printing Activities	307

Figure 3.8: Event ID for extracting Logs in Microsoft windows.

ii. Misuse-Based Approaches

Detection systems store a series of attack descriptions. They have their own language to describe the analyzed attacks. STATL is an example of a language that describes an attack as a state machine and transitions. Since it is not possible to describe a system as a state machine given the large number of possibilities, it collects them as multiple variables. The event space has a filter that depends on the mode in which the application is running. Because it is possible for there to be multiple occurrences of the same attack at the same time, STATL has multiple prototype scenarios that have attacks stored in them. The scenario evolution is defined by its transitions, which have associated actions, which are those that describe the events that cause the transition to be taken. STATL has operational semantics that describe instances of scenario prototypes. These prototypes define the global environment, and the instances represent

individual attacks currently in action. The evolution of these instances depend on the type of transitions, which can be consuming, non-consuming or leading:

i. A non-consuming transition is used to represent a step of an attack in action, which does not prevent future occurrences of attacks from arising from the root state of the transition. When a no consuming transition is seized, the root state remains valid, along with the destination state.

ii. In contrast, a consuming transition causes the root state to become invalid, changing the initial attack state.

iii. A leading transition allows to go back (roll-back) to be able to return to the previous states. In the middle of an attack, deleting a file can invalidate the continuation of the file, and thus you can go back to a previous state, like before a file was created. An example would be an attack where the attacker creates a malicious. rehost file in the root (home) directory via ftp, allowing them to gain remote access without having to log in with a password.

iii. Anomaly-Based Approaches

Complementary to the 'misuse-based approach', detection is based on models of normal user behaviour, called 'profiles'. Any deviation from the established profiles is interpreted as attacks. With this procedure, a greater number of unknown attacks are detected, but a high number of false positives are generated. Information is collected from legitimate systems during normal use, and compared to inappropriate use to uncover anomalies. A model is a set of procedures used to evaluate a number of characteristics of a system call (such as the size of a string). A model can operate by learning or detection. In learning mode, the model is trained in normal use, collecting values considered "normal" in the execution of the program. The other way is to return the probability that a system call obtains a valid result. An example model would be the size of a "String": standard input is usually composed of a maximum of one hundred characters, and most consist of human-readable characters (commands). If there is malicious input, it usually appears in the system call arguments. System calls are usually represented by canonical

file names found on the system itself. If, for example, an attempt to open a file fails, and this is reflected in the log of an application, this vulnerability can be used, so that the attacker sends malicious code as an argument in the "open" call.

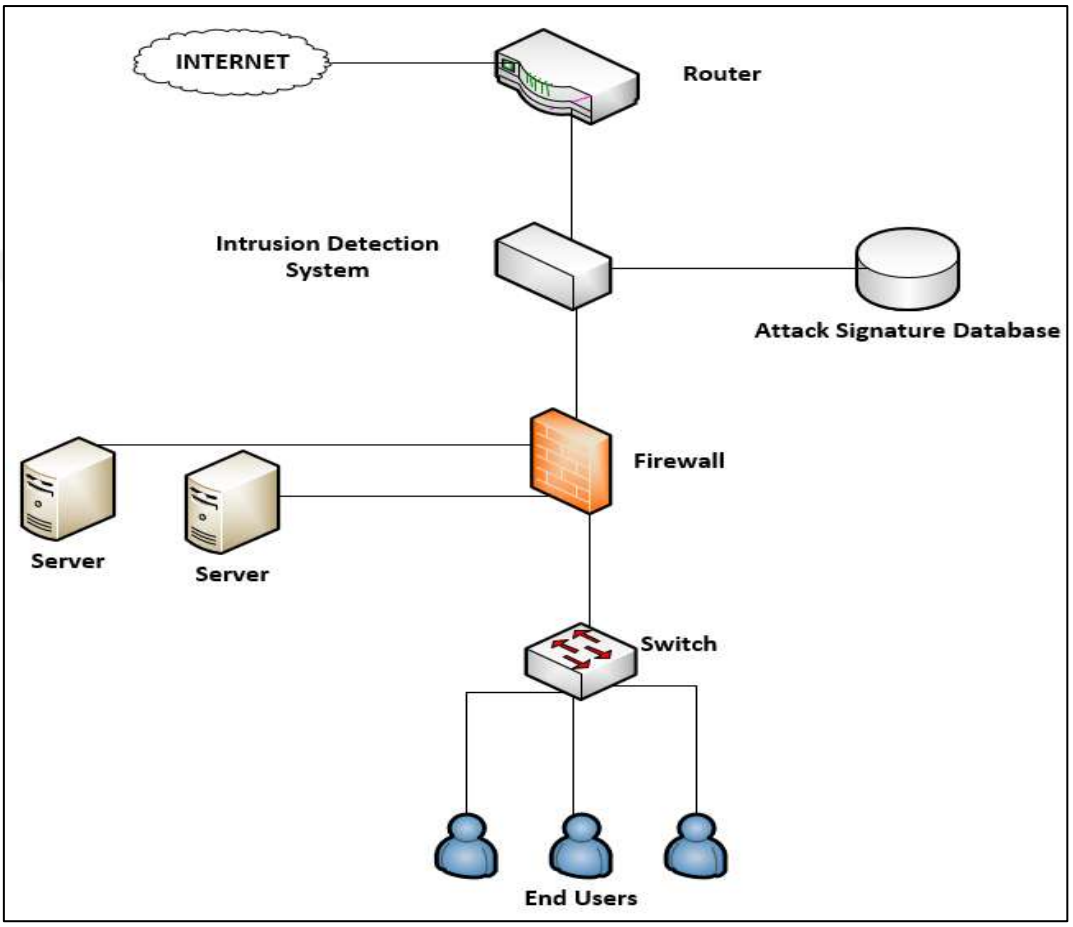

Figure 3.9: Anomaly based Intrusion detection system.

Several hundred bytes in size. The goal is to approximate the lengths of an argument and to detect cases that deviate from normal behaviour. To characterize the lengths of Strings, the mean and variance of the actual (known) String lengths are used. Then, in the detection phase, the values of the system call are compared with the established one, observing if it appears between the legitimate ranges. The character distribution model has the advantage that it cannot be circumvented by some known attempts to hide malicious code within a String. These contain rules that raise an alarm when certain sequences are detected.

An attacker can replace these sequences with instructions that have similar behaviour, and this makes the signature-based model unable to detect the anomaly easily, although character distribution analysis detects the distortion and gives a high anomaly score. The definition of models is complicated, since many of the attacks are usually based on “normal” behaviour. These types of attacks are called "mimicry attacks" that can occur due to design problems or the low quality of the input events. Attackers are often detected because they are unaware that an intrusion detector exists. Otherwise, the attacker tends to behave in a more “normal” way.

iv. Specification-Based Approaches

While learning-based defect detection systems build models by monitoring application traces, the specification model is defined a priori without the need to use dynamic information from the program. Techniques that use specifications are generally not as prone to false alarm reporting as their anomaly-based cousins. That is, given a complete and exact policy, these systems work very well. Unfortunately, the task of producing such a policy for realistic applications and scenarios is not a trivial one. 1.2 Application-Level Intrusion Detection An important source of audit data for host-based intrusion detection systems is information provided directly by applications. The data collected by the application audit gives more extensive and reliable information. Therefore, it is easy to determine which program is responsible for a particular event. Unfortunately, the data provided by the application is very specific, and HIDS has to deal with it on an application-specific basis.

iv. HoneyPot Method

Most operating systems use centralized log files shared between the operating system and auditing applications. Audit application data is also found in application-specific log records. Specifically, the error logs are more important, since they usually collect malicious behaviour. Although the log files have highly detailed information, since they have a different format for each application, the HIDS has to be adapted to each application individually, in addition to the fact that the log only collects information (and in a summarized way) once the operation has already been carried out, so that it cannot act preventively.

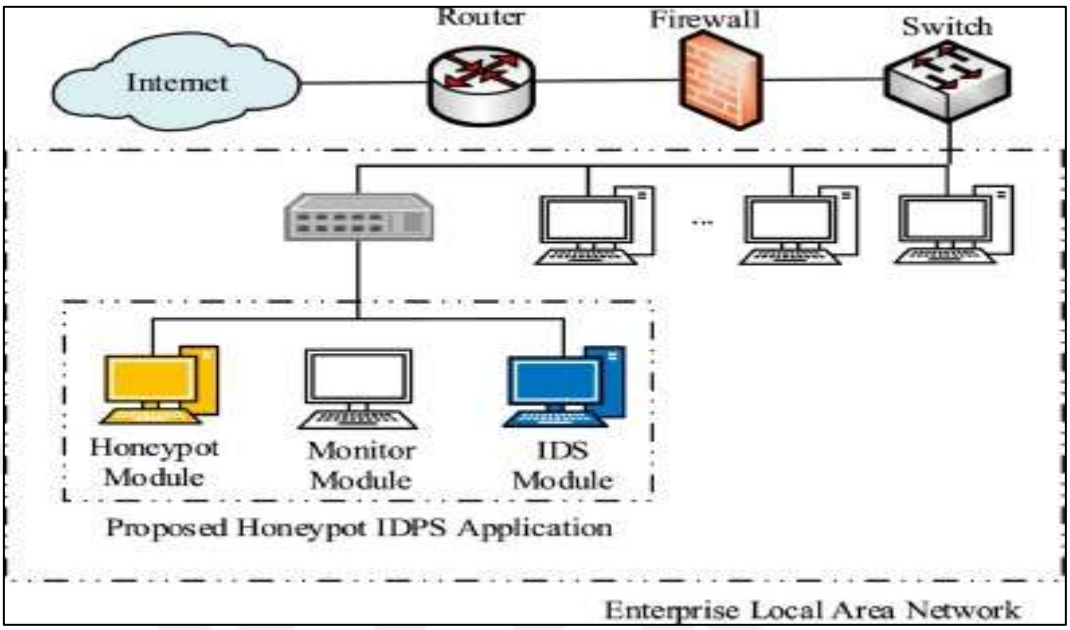

Figure 3.10: Honeypot method for intrusion detection.

In some applications, it is possible to modify the logs so that they collect more information, but the operating system does not always have the necessary information to be able to interpret the operation that has been carried out, and it tends to be very heavy, occupying large amounts of memory. To get rid of some limitations, integrated systems within applications have been proposed that analyze data while the application is running, which acts preventively in a certain way. The more integrated the IDS is with the application, the more information can be collected, and also the fewer false alarms it generates. In order to integrate the IDS with the application effectively, the application must have a suitable interface in the form of an API, or responses to subroutine calls, or simply extend the application if it is an open-source application. An example of an integrated IDS system would be the "Honeypot" system, which deliberately exposes vulnerabilities to lure the attacker. Honeypots are useful for analyzing and studying the behaviour of attackers, as well as distracting them from the main objective. Depending on the level of activity allowed for the attacker, we distinguish two types of honeypots:

i. Low-interaction: They have the advantage of being simple to maintain, but at the same time they are easy for the attacker to detect.

ii. High-interaction: Contrary to low-interaction honeypots, they try to fully emulate the behaviour of an operating system, which allows them to collect a large amount of information to find out the attacker's intentions.

In addition, by not assuming the behaviour of the attacker, they have an open environment that captures all activity, allowing the IDS to learn behaviours that it was not previously aware of. However, these types of honeypots are more complex to develop and maintain.

3.5 DETECTION METHODS FOR INTRUSIONS

The attacker detection module identifies and reveals the identity of the attacking node. This module uses two assessments to determine whether a node is good or attacking. These assessments are based on confidence and reputation calculations in order to detect a sinkhole node. Such assessments are calculated and updated constantly, maintaining the security and integrity of the nodes in the network. The measures for detecting a sinkhole attacking node within the IoT are described below. The first measure is to know the sinkhole attack. This attack advertises wrong routing information to attract network traffic [35], after receiving the traffic it forwards some or no packets to the destination or performs some data manipulation, in addition to injecting false data to compromise other nodes, in order to harm a collection point. The second measure is to define by which characteristic to detect the sinkhole attack. The detection of this attack is complex because it can be “apparently transparent”, however, its effects are quite pronounced [22]. The detection of the attack will be carried out by the packets that may or may not forward to the next hop or destination. The third measure determines the method or technique to detect the sinkhole attack. For that, this proposal adopts an approach based on the reputation of each node that is represented by (R) and by the trust represented by (C), in order to detect sinkhole nodes. In this way, the reputation and trust calculation will determine whether a node is a good or a sinkhole attacker. Therefore, these calculations allow providing greater security and protection to the network nodes. The IoT data forwarder reputation calculation reflects the focus of a trust relationship. Reputation is the opinion or perception that an entity creates through iterations, actions or information according to the equation number (1):

$$i = \frac{12 * \alpha * \beta}{(\alpha + \beta)^2 * (\alpha + \beta + 1)} \tag{3.1}$$

These iterations are directly or indirectly based on past tasks [20]. The Beta distribution is based on two types of satisfactory and unsatisfactory iterations that each node performs within the network. Using the Beta distribution (α,β) one can represent the reputation and trust of IoT nodes. The advantage of using this distribution is that the parameters used are continuously updated to determine the behaviour of a node within the IoT. After obtaining the calculations of the predictions (i, c, d) it is possible to compute the reputation. A node's reputation is calculated from experiences based on directly computed predictions. The reputation calculation is performed taking into account the status (St) of the node that forwarded the IoT data messages. As mentioned before, St Represents how the node behaves in forwarding messages. However, the St and computed predictions (i, c, d) are the input data for using the Dempster-Shafer theory in order to detect and reduce false alarms. A node decides according to the value obtained after applying the Dempster-Shafer theory whether a node is good or an attacker depending on the calculated value.

4. METHODOLOGY, IMPLEMENTATION AND RESULTS

In this chapter, we started with the importance of deep learning for intrusion detection. Then, we presented a definition and a classification of various DL methods. We have also detailed three DL approaches that will be the subject of our work. At the end of the chapter, we cited the different datasets used for the evaluation of IDSs systems based on machine learning.

4.1 DEEP LEARNING FOR CYBERSECURITY

With the availability of large amounts of data from cyber infrastructure, networks, operating systems or information systems and to address cybersecurity challenges, methods and techniques such as machine learning, data mining, statistics and other interdisciplinary capabilities have been exploited.

Deep learning which is part of machine learning could be used for signature based or anomaly detection based IDSs. These classification and prediction methods can be used to detect unusual patterns and behaviors of various cyberattacks that enable real-time cyber response. They have the ability to detect attacks when they have occurred and also the ability to predict potential future attacks. Methods based on deep learning can help overcome the challenges of developing an effective IDS.

On the other hand, the collection of data and network traffic has led to a big-data problem security experts always want better performance IDS which have the highest detection rate and the highest false alarm rate down. Therefore, deep learning approaches that adapt well to a very large amount of data. The latter have been introduced for the detection of network anomalies with the aim of differentiating between normal behaviour and abnormal behaviour to identify harmful or potentially dangerous [45].

4.1.1 Definition of Deep Learning

Deep learning (DL) belongs to a class of machine learning (ML) techniques, it achieves great success in many artificial intelligence (AI) tasks compared to ML algorithms classics. Deep model architectures are relatively recent where many nonlinear information processing steps are

exploited, in which information is processed in hierarchical layers, each receiving and interpreting information from the previous layer for learning representations of data.

i. Functioning

Generally, the architecture of deep networks is organized in layers of neurons for any type of these networks; an Input Layer, one or more Hidden Layers and an Output Layer.
Each pair of neighbouring layers is connected. The connections between them called weights (Weights).
The "neurons" of the same layer generally called "nodes" have no association Deep learning presents itself as an advanced computational system, it is made up of a variety of techniques from the field of machine learning that use a deluge of non-linear neurons (nodes) arranged in several layers of processing that extract and convert entity variable values from the input vector to create multiple levels of abstraction to represent the data. The learning of the DNN is the optimization of the weight parameters and the bias parameter between two neighbouring layers, it evaluates the accuracy of the model and makes it possible to better adapt it to the learning data and its needs. When the model reaches maximum accuracy with optimal parameters, it will be generalized for real data.

ii. Classification of DL Methods

In practice, all deep learning approaches are neural networks, which share some common basic properties. They are all made up of interconnected neurons, they are organized in layers. What differentiates them is the architecture of the network (or the way the neurons are organized in the network) and sometimes the way they are formed. Presented a study and analyzed 10 different deep learning approaches most used for detecting intrusions in cybersecurity. These approaches can be categorized into three models, depending on how they are formed and intended to be used.

i. Deep learning for supervised learning: It is used when target label data is available, it is deep discriminating models namely Deep Neural Networks (DNNs), recurrent neural networks (RNNs), Convolutional neural networks (CNNs).

ii. Deep learning for unsupervised learning: It is used when the input data is not labelled, it is generative models aim to group data according to certain criteria of similarity for the purpose of recognition or synthesis of models, namely the deep belief networks (DBN), deep autoencoders (DA), Restricted Boltzmann machine (RBM) and deep Boltzmann machines (DBM).

iii. Deep learning hybrids: a hybrid combination of these models mentioned above. Unsupervised deep networks could provide excellent initialization on the basis for which discrimination (supervised learning) could be examined.

4.1.2 Some Deep Learning Methods

Deep neural networks are a set of neurons organized in a sequence of interconnected layers. What differentiates them is the architecture of the network (how the network`s neurons are arranged and how they work). Among many implementations of deep learning models:

i. Deep Neural Network (DNN)

These networks refer to a set of neurons organized in a sequence of multiple layers called Multilayer Perceptron's (MLP). They differ from traditional neural networks (Artificial Neural Network) by their depth and the number of layers, nodes (neurons) that make up the network. When an ANN has two o3r more hidden layers, it is known as a deep neural network. They attempt to model data containing complex architectures by combining different nonlinear transformations. Rosenblatt introduced the basic idea of perception in 1958 [40]. Perception computes a single output from multiple real-valued inputs (xi) by forming a linear combination based on its input weights (w), then placing the output through a nonlinear activation function. Mathematically, this can be written as follows:

$$y = \delta(\sum_{n=1}^{n} W_i x_i + b) = \delta(W^T X + b) \tag{4.1}$$

With:

i. *W*: is the vector of the weights.

ii. X: is the vector of the inputs.

iii. b: denotes the bias.

iv. δ: represents the activation function.

A typical multi-layer perception (MLP) network includes a set of source nodes forming the input layer, one or more hidden layers of compute nodes, and an output layer of nodes. The input signal propagates layer by layer on the network. The signal flow of such a network with a hidden layer. DNNs are generally used in supervised learning problems. Model training (learning) means adjusting bias and weight to its optimal environment.

ii. Convolutional Neural Networks (CNNs)

A convolutional Neural Network or CNN is an extension of traditional feed forward networks (FFN) in the context of biological factor inspiration. These were originally investigated for processing images in which repeating patterns can be found - for example, an image with repeating edges and other patterns. CNNs outperform all other classical ML algorithms and make great success in computer vision processing tasks (Computer Vision Tasks), they have wide applications in image and video processing, natural language processing (NLP), recommender systems, etc.

Convolutional networks are particularly efficient thanks to several types of special layers: convolution layers, pooling layers and fully connected layers

Convolution Layers:

The goal of convolution is to extract high-level features. It consists of a set of learner filters (or cores), each represents a certain independent functionality with the input volume. These filters consist of a layer of connection weights, they have a small receiving field (the size of the core), but when passing forward (feed forward), each filter is convolved across the width and height of the input volume, calculating the product of the points between the inputs and the filter values producing a new feature map that better represents the information. As a result, the network learns which filters activate when it. Detects a type of feature that is important and specific to a

certain spatial position in the input. A convolutional layer shares the same convolution kernel, which greatly reduces the number of parameters needed for the convolution operation. A nonlinear activation function will be applied immediately after each convolutional layer. Deep CNNs with the “Rectified Linear Units” activation function

ReLU”

$[F(x) = \max(0, x)]$ Equal (3)

Returns x for all values of x > 0 and returns 0 for all values of x <= 0. Train many times faster than their counterparts with “Tanh Units” Pooling Layers:

After the ReLU transformation, the Pooling operation pools the activation of neurons from one layer into a single neuron from the next layer. The pooling layer works independently on each input entity, it allows to decrease the number of weights or parameters, it is possible to gradually shrink the volume of the representations, which decreases the computational cost in the network, while preserving the most important information. More critical. It also helps to control overfitting.

It can use two different pooling methods:

i. Maximum pooling (Max-Pooling): uses the maximum value of each group of neurons from the previous layer.

ii. Average pooling: uses the average value of each group of neurons from the previous layer.

Pooling is a form of nonlinear subsampling that works similarly to convolution. The pooling kernel convolves over the input volume and divides it into a set of non-overlapping regions, and each sub-region produces a single output value which is the maximum value for Max-Pooling or the average value for Average-Pooling the Pooling layer has no learnable parameters. Because of this, these layers are usually not included in the total number of convolutional network layers.

Fully Connected Layers At the end of a CNN there is one or more fully connected layers.

(Every node in the first layer is connected to every node in the next layer).

They consist in performing a classification based on the features extracted from the convolutions. The final layer contains a Softmax activation function, which generates a probability value from 0 to 1 for each of the class labels that the model tries to predict. In some recent CNNs network architectures, the fully connected layers can be replaced by several average pooling layers. This allows these networks to significantly reduce the total number of parameters and which allows better prevention of over-fitting.

iii. Recurrent Neural Networks (RNNs)

These networks draw their inspiration from how biological neurons behave in the brain of human, these neurons are considered the center of reflection, and sometimes they must memorize certain events for later use before making the decision. Traditional neural networks do not have this property, so the operation of a recurrent neural network (RNN) is motivated by the fact that a human being reason based on the knowledge he has acquired and who 'he previously memorizes. RNNs are Feed-Forward type networks having an internal state (or memory) which takes into account all or part of the data seen previously (already provided to the network), in addition to the data currently seen to adapt their decision. The basic key idea of these networks is the deployment of a recurrent calculation thanks to the loops in the architecture of the network. The network output is a combination of its internal state (memory of inputs) and the last input, at the same time the internal state changes to incorporate this new input data. Due to these properties, recurrent networks are suitable in cases where the presence of a shape is not the only discriminating information but also an order of appearance for example. they are good candidates for tasks that deal with sequential data, such as textual data or data with temporal characteristics. The mathematical description of the memory transfer process is as follows:

$$HT = \delta(Uxt + V\, ht_{-1} + bh) \quad \text{Equation (4)}$$

$$Oh = \delta(Wht + by) \quad \text{Equation (5)}$$

(4.2)

Where:

i. *HT*: is the hidden state at time t.

ii. *xt*: is the input at the same time t.

iii. *U,V,W*: are the weighting, Input-to-Hidden, Hidden-to-Hidden and Hidden-to-Output matrices respectively (known as transition matrices).

iv. *bh*: is the bias value of the hidden state.

v. *by*: is the output bias value.

vi. *Oh*: is the output value at time t.

vii. δ: is a nonlinearity function called activation functions. (either a logistic or tanh sigmoid function) which is a standard scaling tool for condensing very large or very small values into a logistic space, as well as making gradients workable for back-propagation.

A neural network block, examines an xt input and outputs an ot value. A feedback loop occurs at each time step, each hidden state ht contains traces of not only the previous hidden state, but also all those before h_{t-1}as long as memory can persist.

Short Term Memory Units (LSTM)

The RNN network has a long-time step because it takes into account the previous saved state when updating the weight, the gradients as the training gets smaller and smaller and after a few steps the errors have no could not be propagated to the end of the network. There won't be a significant difference in the result, so it can't update the weights. This RNN problem is called Vanishing Gradients. To overcome this problem, a long and short-lived memory (LSTM) architecture was proposed for recurrent neural networks and also additional stages called Gated Recurrent Units (GRUs). These steps have been used to enhance the accuracy and performance of RNNs.

The key idea of the LSTM method is the state of the cell. It has the ability to remove or add information to the cell state. This technique is regulated by structures called gates. These could

be a sigmoid function where a value of 1 means all information passes and a value of 0 means the opposite. The LSTM and GRU architectures work the same way. However, the GRU uses less training parameters and therefore less memories and trains faster than the LSTMs. While LSTM is more accurate on datasets using longer sequence.

LSTM cells are the most effective in retaining useful information during gradient back-propagation. This allowed them to correct the differences between the outgoing predictions and the reference categories by calculating the error gradient for each neuron, going from the last layer to the first.

4.2 SYSTEM OUTLINE

We proposed 3 deep learning models (DNN, CNN, RNN) in cybersecurity allows IDS to deal with these types of network attacks to identify malicious DDoS activities in real network traffic. We first started with a simple DNN model. Then, we implemented the CNN and the RNN for the classification of this type of network attack in order to remedy the problem of detecting the various possible DDoS attacks with a very high detection rate and a negligible alarm rate. The performances of the proposed detection approaches were evaluated taking into account the different evaluation measures of the deep learning algorithms namely, the precision, the recall, the F1 score, the false alarm rate and the detection rate.

4.3 DATA SETS

The data set chosen for this study is CIC-DDoS-2019 data-set, provided by the Canadian Institute of Cybersecurity (CIC), it constitutes data of real network flows with several types of attacks Latest and most popular DDOS. These data are more condensed formats that mainly contain meta information about network connections, where each data (each network stream) groups all the packets which share certain properties in a time window and which do not include a payload. The data set has two versions of data.

This dataset also includes 12 different most up-to-date DDOS attacks, the full dataset contains 50063112 instances, of which 50006249 are DDoS attacks and 56863 are instances of benign (legitimate/normal) network traffic, the number of instances for each type of DDOS attack is

shown in Table 4.1. This data-set also contains 86 characteristics (Features), where 6 of them are labelled and characterized the flow itself, according to Source IP, Source Port, Destination IP, Destination Port, Protocol and Timestamp (attack times), and more than 80 characteristics on the network traffic flow. Dataset data have 2 disadvantages, the first big disadvantage is that they are unbalanced. Instances of benign network traffic represent only 1.13% of the dataset. And also, the percentages between attack classes are varied. The second disadvantage is that the data labelling (class label) CSVs differs between the CSVs of the training data that are generated in the first day of experimentation. And the CSVs of the test data generated in the second day. In addition, this test data has only 5 class types as shown in Table 4.6. What is imposed to modify the labelling of one of the 2 sets before using it and to make the equivalence between the classes of training data and the classes of test data.

Table 4.1: CICDDoS2019: The number of records for each category DDOS attacks in all data.

Class Label	Number of Instances
Benign (legitimate traffic)	56863
DDoS_DNS	5071011
DDoS_LDAP	2179930
DDoS_MSSQL	4522492
DDoS_NetBIOS	4093279
DDoS_NTP	1202642
DDoS_SNMP	5159870

4.4 TAXONOMY OF DDOSS ATTACKS

Distributed Denial of Service (DDos) is an intrusive attempt that is a series of attacks that send a huge amount of packets to a target resource for the purpose of temporarily or permanently disabling the services provided by that resource. A taxonomy of these attacks has been proposed by the authors composed of 2 main categories which are: attacks by reflection and attacks by exploitation [44]**.**

Reflection-based DDoS attacks: These are a series of attacks in which the identity of the attacker is concealed by using botnets to send attack traffic (e.g. HTTP requests) to the victim. These

requests are sent to the reflector servers by the attacker with the source IP address set to the target IP address with the aim of flooding the victim with the response packets. These attacks are carried out using transport layer protocols, namely transmission control protocol (TCP), user datagram protocol (UDP) or a combination of these protocols [44].

i. Exploit-based DDos attacks: This is a series of attacks in which the attacker attempts to directly exploit the remote service. These attacks can be carried out using TCP or UDP transport layer protocols.

Table 4.2: DDoS attacks based on reflection and exploitation.

The attacks	Description
Syn Flood	A denial-of-service attack that exploits the weakness of the TCP connection sequence. It takes place in three stages: the SYN, the SYN-ACK and the ACK the attacker sends a succession of SYN requests to a target server without an ACK response in an attempt to consume its resources.
LDAP	LDAP injection is an attack used to exploit web applications that construct LDAP statements based on user input.
SSDP	In this reflection attack variant, the attacker abuses the Simple Service Discovery Protocol (SSDP) and sends a request with a spoofed source address to a target victim to overload his computing resources with responses from this protocol.
UDP	A DDos attack in which a large number of packets are sent to a victim using the UDP connection without the need for an established connection. This is in order to overload its processing and response capacity which results in limited unavailability for users.
DNS	A reflection attack, exploiting DNS vulnerabilities (Domain Name Service Protocol)
MSSQL	Microsoft Structured Query Language is an attack that allows to execute malicious SQL statements
NetBIOS	A security exploit in the Network Basic Input/Output System (NetBIOS) allows an attacker to see information in the memory of a computer on a network
SNMP	in this variant of o verload attack SNMP protocol attack generates a large amount of traffic that is directed to victims from multiple networks.
UDP lag	An attack used to disrupt the connection between client and server.

Table 4.2: DDoS attacks based on reflection and exploitation "tables continued".

TFTP	This attack exploits the buffer overflow vulnerability in a Trivial File Transfer Protocol (TFTP) server
WebDDoS	WebDDoS is an attack that aims to bring down the target website or slow it down by flooding the network, server or application with fake traffic
NTP	NTP is an amplification attack in which the attacker exploits publicly available NTP servers to overload the target with UDP traffic
PortMap	an attack on TCP or UDP port 111 which is a service used to direct clients to the appropriate port number so they can communicate with the remote procedure call (RPC) service

4.5 DATA PREPARATION

The performance of deep learning methods strongly depends on the quantity and quality of learning data, more data with quality, more precision and good results. In our case, we have a very sufficient mass of data. However, because of the data class imbalance. This data needs to be reduced.

The preparation of the data is about 2 essential operations before processing them. Which are: data reduction and labelling resolution.

4.5.1 The Data Split

The volume of the data set forces us to scale down a significant number of samples from the training and test data.

This mass of data constitutes 11 CSV files of sizes over 22 GB for learning. Thus 7 other CSV files for the size test over 8 GB. The reduction of training and test data is done separately following the same procedure. Reading and preparing data were not easy tasks with this huge amount of data even when we worked on colabe which offers 12 GB of RAM, for this we have:

i. Started to parse and process each CSVs file separately, in order to load the data for some large size files (over 8 GB), we used horizontal splitting of data into 2 or more CSVs files.

ii. Randomly chooses a predefined number of instances for each majority class in the set of instances of this class.

iii. Keep all instances of minority classes, which are collected from multiple files.

iv. Combine the training data plus the test data into 2 single separate files called "Train.csv" and "Test.csv".

v. Generate 3 different subsets of data from training and test data for 3 different experiments. Tables 4.3, 4.4 and 4.5 show the breakdown of these datasets.

4.5.2 The resolution of the labelling

As illustrated in Table 4.6, the labelling of the data is poorly done, because of this we chose to re-label the test data manually using the "Notepad++" editor. The data from the set of CICDDoS_2019 Tests consisting of.

Table 4.3: Subset_1 consists of 6 different DDoS attacks (Dataset_1).

	The Classes concerned	Number of instances for learning	Number of instances for the Test
	BENIGN	56101	17146
	DrDoS_NetBIOS	619700	136729
	DrDoS_MSSQL	619446	157076
Dataset_1	DrDoS_LDAP	619251	150701
(7-Classes)	DrDoS_UDP	618696	150706
	UDP lag	183662	1873
	Syn	790662	150416

Table 4.4: Subsets_2 consists of 2 classes for the detection of attacks DDoS (Dataset_2).

	The Classes concerned	Number of instances for learning	Number of instances for the Test
Dataset_2	BENIGN	56101	17146
(2_Classes)	attack	997054	314716

4.5.3 Data Pre-Processing

In order to build a highly accurate model, it is important to perform exploratory analyzes on the data set and its characteristics. Pre-processing of the dataset is done before it is applied to the deep neural network. The pre-processing steps are as follows:

i. First, the dataset was filtered to remove all redundant rows representing class instances. Then, an analysis was performed to detect any 'NAN' (Not A Number) or 'INF' (Infinite Value) values, these values can be considered as missing values. Deep learning or machine learning algorithms in general treat these values very badly, which directly and negatively affect the performance of the final models. It appears that the data selected as the dataset concerned by this study have several values of 'NAN' for the Flow Bytes column, in order to keep this characteristic and as we have sufficient data, the rows with NAN values or INF have been removed.

ii. The descriptive statistics that summarize the dispersion and distribution of the dataset showed that there are empty columns (its values are always 0), these characteristics do not contain any discriminating information to differentiate the attack classes, on the other hand they can give bad results, these columns are

iii. There are a number of categorical type features in the dataset that need to be encoded. The Flow Packets/s column has been converted to a numeric column. However, other categorical columns like the IP Address and Timestamp columns have been removed. It was considered that these characteristics are related to connection information and

do not represent the properties of DDOS attacks, since the latter can be produced at any time by any machine against any victim machine, for this reason the characteristics 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp' and 'Inbound'. Are also deleted so that only the characteristics of the network traffic (Traffic features) remain.

iv. For the Label column which represents the class of each instance, it has been encoded with a popular technique called "One-Hot-Encoding". The coding will convert rows containing categories to their own column with a value of 1 meaning true (this instance is of this class) or 0 meaning false (this instance is not of this class).

v. When we obtain quality data, where each instance of the classes contains information that well describes its class. The next step before moving on to learning is standardization. The input data must be normalized, this step has an effect in model building by reducing the learning rate, model training converges quickly. It can also have a regularizing effect by reducing generalization error.

vi. The training of data is contained two steps, the first step is training of data and the second step is validation of model. Then the model will be test only on the CICDDos2019 Dataset.

vii. As we mentioned before, these datasets are very unbalanced, we will have bad performances on the minority classes. In our case, the minority class BENIGN is the most important to differentiate normal traffic from malicious traffic containing DDoS attacks. A high true negative rate will reduce the false alarm rate of the system. To deal with this problem, we tried one of the oversampling algorithms: Synthetic Minority Oversampling Technique (SMOTE) (figure 4.2) on the training data in order to increase the size of the minority classes in the 3 experiments.

viii. SMOTE is an oversampling algorithm that creates synthetic observations based on observations of existing minority classes. Dependent on the amount of oversampling needed, SMOTE computes the k nearest neighbours to create synthetic examples.

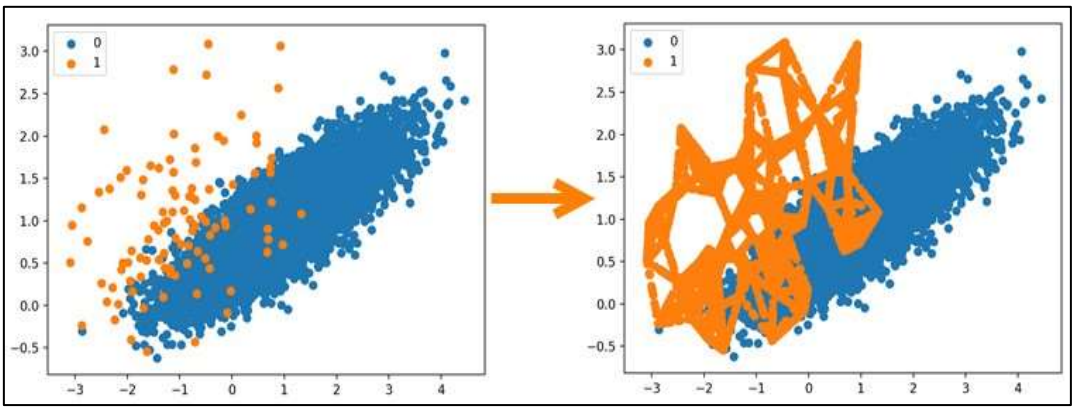

Figure 4.1: Oversampling via SMOTE (Oversampling Technique).

The Scikit learn library also offers an algorithm to handle an unbalanced data set by setting the class weight parameter.

4.6 A DETECTION SYSTEM

Intrusion for the detection of DDoS attacks in Networks Using Deep Learning classification techniques; we have implemented three types of discriminative model based on deep learning which are: the deep neural network (DNN), the convolutional neural network (CNN) and the recurrent neural network (RNN). These models were built and evaluated on 3 subsets of data from CICDDos2019 Data-set for 3 different experiments:

i. A multi-classification (7-classes) with the BENIGN class which designates benign (legitimate) network traffic, and six different attack categories, on the data set illustrated in Table 4.3. We used the training data for model training and validation, then using test data from CICDDos2019 Dataset the sample was assessed.

ii. A binary classification (2 classes) on the data set illustrated in table 4.4. We have grouped both the training and Test data classes into 2 categories: the BENIGN class and the Attack class refers to malicious network traffic that contains all DDoS attacks from both data sets (Training/Test). We used the training data for model training and validation, then the model was evaluated on the test data from CICDDos2019 Data-set. The objective of this experiment is to evaluate the effectiveness and rate of detection of anomalies of DDoS attacks.

iii. A multi-classification (13-classes) with 12 different DDoS attack classes, on the dataset shown in Table 4.5. We used the training data from the CICDDos2019 Data-set for training and model evaluation, this experimentation is for the purpose of testing the system detection efficiency against several types of DDos attacks that have behaviors different and also to assess the ability to determine the kind of attack. At this stage, the system performance relies on the detection rate and the total classification accuracy.

This study is based on the traffic characteristics extracted by CICFlowMeter-V3. We evaluated the rate of false alarm and the detection rate as well as other classification metrics for any approach proposed for the three experiments.

4.6.1 Model Architecture

We have implemented three types of deep network approaches (DNN, CNN and RNN), the architecture of each of these approaches has been modified and improved by trying several combinations of several parameters for each experiment. Nevertheless, these proposed model architectures share some common properties and parameters:

i. The input layers have the same dimensions (number of neurons and Features) in the vector of input for each model.

ii. The activation function used was ReLU, different other functions like tanh and sigmoid have been experimented, but the ReLU always has the best results.

iii. The output layers have the same dimensions as the number of classes, for the multi-class classification the “Softmax” activation function has been chosen. It gives a probability (whose sum is equal to 1) at the output of each neuron, the output neuron with the greatest probability then making it possible to decide that its associated class is the predicted class.

iv. The dropout technique has also been used when the overfitting problem is encountered. This technique involves randomly considering only a percentage of neurons in a layer in order to obtain a generalizable model.

v. The loss function selected was “categorical-cross-entropy” for multi-class classification and “binary-cross-entropy” for binary (normal/attack) classification.

The “Adam” optimizer was used with a Learning Rate of 0.001, instead of the stochastic descending gradient (SGD) optimization algorithm which gave us poor results.

The loss function will measure the deviation between the model predictions and the expected results. Then, the "Adam" optimization algorithm will dictate how to develop the weight of a neural network to reduce the loss, which to the model converges quickly and obtains better predictions with the minimum of error.

4.6.2 Deep Neural Network (DNN)

The proposed architecture of the DNN model has two hidden layers, with an input layer and an output layer (Figure 4.3).

Thanks to these hidden layers which contain a large number of parameters constituting the model, the DNN network can perform an automatic extraction of the corresponding complex characteristics from raw data. This is for the purpose of determining the underlying statistical properties of normal packets and packets of different attacks.

More hidden layers lead to a more complex model, the results may be better,, but they may lead to overfitting. After setting the activation functions, the number of samples per batch and the optimization function. The model converges faster than other CNN and RNN models. This model was trained over more than 10 epochs for the 7-class classification, up to 30 epochs for the 13-class classification.

4.6.3 Convolution Neural Network CNN

1D CNNs were originally investigated for natural language processing using 1D convolution layers. In our study, network traffic events are represented as time series data in 1D form. Flow characteristics are captured in equivalent time periods, but they have different behaviours from millions of benign connections and malicious DDoS attacks. Therefore, we tried to extract spatial discriminatory features by applying CNN 1d.

Initially, we started with a medium-sized CNN using different numbers of convolutional layers, different numbers of filters (16, 32 and 64) with lengths 5, 3 and 2, in order to find the right parameters and the better network structure.

Figure 4.4 shows the evaluation of the accuracy of the CNNs models using several convolutional layers for the 3 experiments. The number of convolution layers needed usually depends on the complexity of the data. The more convolutional layers we used, the better accuracy we got, although after about two or three layers, the gain in accuracy becomes rather stable as well as learning takes a long time.

The efficient version of the "Adam" stochastic gradient descent algorithm was used to optimize the network, and the categorical cross-entropy loss function was used since we are learning a multi-class classification problem.

4.6.4 Recurrent Neural Network (RNN-LSTM)

Starting from the idea that the current record of a network traffic connection can be classified based on the past records of that connection to that network. We tried to extract discriminatory temporal characteristics.

Therefore, we implemented an RNN_LSTM network that classifies network traffic events as time-series data. The proposed model is composed of an input layer with a three-dimensional input [batch size, time step, features], our model has an input of (None, 4, 17), where 4 represents the number of sequences (time step) used, and 17 represents the features of a single sequence. In total there are 4*17 characteristics of the stream used, the model will memorize 17 characteristics at a time at time t. Figure 4.4b shows the evaluation of the accuracy of the RNNs models with different numbers of time steps for the 3 experiments. According to the experiments, dividing the input features into multiple time steps for training increases the accuracy of the models a little, but it doesn't have a huge effect. LSTM cells can both store all input characteristics and use them to differentiate between different records. Moreover, the use of several time steps requires considerable training time.

Four hidden LSTMs layers were used, with 4 dropout layers of different regularization percentages between them and an output layer (figure 4.3b). The dataset can be defined as a list of [xt,yt]. Given an input vector of network traffic events at time t xt = (x1,x2,....,xn_{-1},*y*),(or xn∈R represents the characteristics and y∈R represents its class). LSTM layers add a longer memory module to build deeper models to predict yt. For this, we split the input features into several and time steps so that the model learns from the temporal properties of the data. The first 3 LSTM layers have a return State = True that is to return the last stored state in addition to the output. The activation functions usually recommended for the RNN_LSTM network are the sigmoid, and ReLU. The last layer contains the Soft-max function for multi-class classification.

4.7 MODEL EVALUATION MEASURES

i. Accuracy (Pr): the percentage of DDoS attacks identified as TP attacks among all the examples predicted as an attack, it is given by:

$$Pr = \frac{TP_{Attack}}{TP_{Attack} + FP_{BENIGN}} \tag{4.3}$$

ii. Recall (Rc): The percentage of DDoS attacks identified as TP attacks out of all attacks in the dataset:

$$Rc = \frac{TP_{Attack}}{TP_{Attack} + FN_{Attack}} \tag{4.4}$$

iii. F1-score (F1): the weighted harmonic means of precision and recall (Recall), it is given by:

$$F1 = \frac{2 * (Pr * Rc)}{(Pr + Rc)} \tag{4.5}$$

iv. True Positive Rate (TPR): the number of DDoS attacks identified as attacks divided by the number of DDoS attacks in the dataset, it is given by:

$$TPR = \frac{\sum TP_{Attack}}{\sum DDoSAttaques} \tag{4.6}$$

v. False Positive Rate (FPR): the number of mild cases incorrectly classified as DDoS attacks divided by the total number of mild cases in the dataset, it is given by:

$$FPR = \frac{\sum FP_{BENIGN}}{\sum TraficBenin} \tag{4.7}$$

vi. Confusion Matrix: Is a specific array layout allowing to visualize the performance of an ML algorithm for a classification problem, it is known as the error matrix.

The authors [44] tested 4 common machine learning algorithms using training data and test data from the CICDDoS_2019 Dataset to serve as a baseline reference for a comparative study. They started by selecting the most important features using the library SKLearn, after they chose 4 most common ML algorithms: ID3 (Decision Tree), Random forest, Naive Bayes, and Logistic Regression. The evaluation measures as well as the results obtained are illustrated in the table below 4.5:

Table 4.5: Comparison of results between the proposed DL methods and the reference ML algorithms.

Method	Precision	Recall	F1-Score
DNN	0.85	0.83	0.82
CNN	0.91	0.90	0.89
RNN	0.90	0.89	0.88

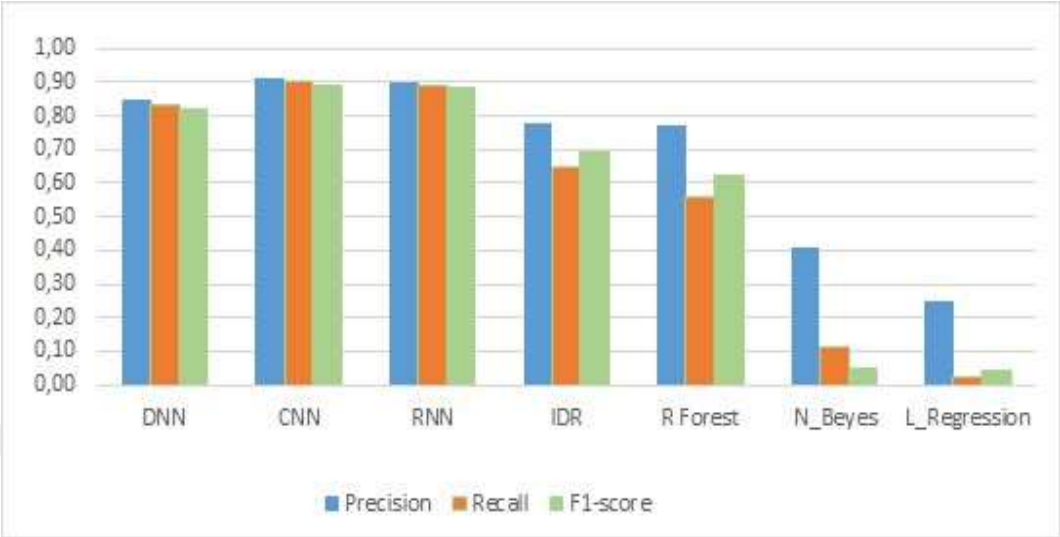

Figure 4.2: Comparison between the proposed DL methods and the ML methods of References.

The DDoS attack detection accuracies of the 3 models DNN, CNN and RNN for the 3 experiments have been illustrated in Table 4.4.

The first experiment Figure 4.8 which illustrates the test results of our DL classifiers (DNN, CNN, RNN) and the reference ML classifiers. It is clear that our DL methods greatly outperform all other ML methods, with a high accuracy and Recall rate.

Among our methods, CNN has the best results with an accuracy of 91%, thanks to its ability to recognize discriminatory patterns for each class, Table 4.4 shows the classification ratio of attacks using the CNN method. As the performance of IDSs systems relies on its ability to differentiate between Normal behaviour from malicious behaviour, the CNN has been well identified. The minority class BENIGN which represents normal traffic with a precision converges to 1 and a recall of 97%. This precision means the number of false alarms is minimal, the recall means that the model is effective in identifying network attacks with a reduced false negative (FN) rate.

Table 4.6: The 7_Classes Classification Report with CNN.

	Accuracy	F1-Score	Recall
Avg-Macro	0.80	0.78	0.78
weighted Avg	0.91	0.89	0.90
BENIGN	0.99	0.98	0.97
DrDoS_NetBIOS	0.94	0.97	1.00
DrDoS_UDP	0.71	0.81	0.93
Syn	1.00	0.99	0.97
DrDoS_LDAP	0.97	0.98	0.99
DrDoS_MSSQL	0.95	0.75	0.61
UDP lag	0.00	0.00	0.00

The CNN model has been well identified also certain types of attacks like DrDoS_LDAP, DrDoS_NetBIOS, DrDoS_UDP and Syn. However, the DrDoS_MSSQL class was misclassified with 61% recall and the worst is the UDP-lag class which was not recognized as an attack class at all, the latter is a minority class and even with different percentages of oversampling its accuracy has not been improved.

The second experiment in this experiment, we tested the effectiveness of our method in detecting DDoS attacks regardless of its types, the detection results showed in Table 4.7. It can clearly be seen that our proposed DL method performed very well. The binary classification with the BENIGN class which represents normal traffic and the Attack class which represents malicious traffic was carried out with a clearly high true positive rate (TPR) which converges towards 1 and a very reduced false negative rate converges towards 0 This means that the proposed models have well identified the normal behaviour of the traffic which makes it possible to reduce the number of false alarms as well as they have well identified the malicious behaviour of the traffic which allows a very high security against this type of network attacks.

Table 4.7: The Binary Classification Report.

	TP	PF	NT	FN	TRP%	RPF%
DNN	301984	140	16814	781	99.74	0.82
CNN	302602	153	16801	163	99.94	0.90
RNN	302571	716	16238	194	99.94	4.22

Figure 4.3 shows the impact of oversampling with SMOTE on the performance of the 3 models for the prediction of the minority class BENIGN. The results of the proposed approaches without using Smote were very good, the BENIGN class was well predicted using small oversampling percentages, the recall results were further improved a bit, however, the use of large oversampling percentages sampling gives us bad results. The Receiver Operating Characteristic (ROC) curve is a performance measure often used for binary classifiers and especially in the case where the classes are unbalanced. It plots the true positive rate against the false positive rate at different classification thresholds, in order to show the trade-off between sensitivity (TPR) and specificity (FPR). Mathematically it is calculated by the area under the curve called Area Under Curve (AUC), In the ideal case we would like to have an Auc = 1, so that the ROC curve passes through the upper left corner of the square at the point (FPR = 0, TPR = 1). The ROC curves allow us to properly assess the accuracy of our classifier with very unbalanced classes where the BENIGN class represents only 5.3% of the data set. We can see that our DL methods perform very well in discriminating between the BENIGN class and the Attack class. The CNN once again outperforms the DNN and RNN with an Auc = 1 for the 2 classes. The third experiment. The best accuracy we got is 93%, this refers to the number of classes involved in this study, thus the imbalance between training and testing data. The normalized confusion matrix below shows that the BENIGN class was predicted well with other attack classes that were predicted well. However, some types of attacks like: DrDoS_DNS, the good discriminating characteristics of the BEGNIN class have been the subject.

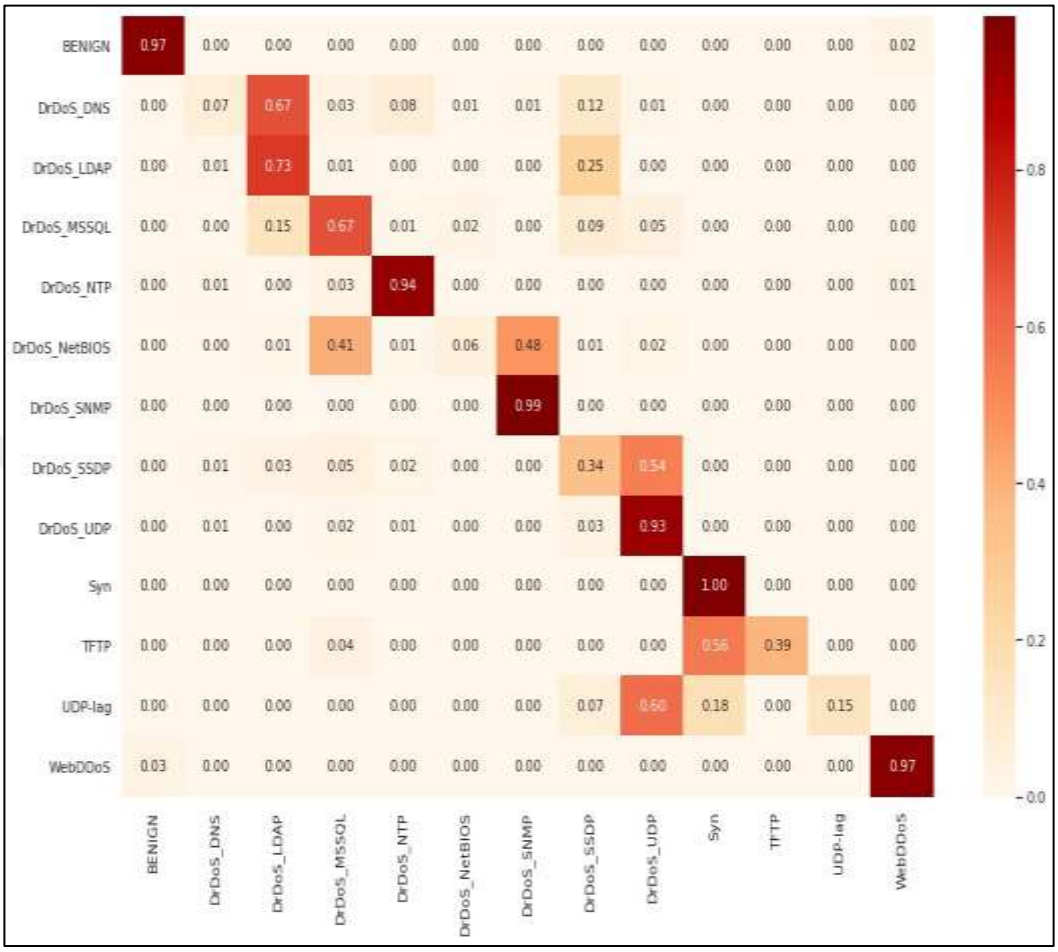

Figure 4.3: CNN Confusion Matrix (13-classes).

5. CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

The protection of one's data is one of the most critical issues of the information community, which is under attack from a large number of viruses, worms, and other dangers. One of these dangers is known as a malware attack, and it consists of a piece of malicious code that is designed to cause damage to computers and networks. Malware assaults are getting more frequent, which means that computer security policy is becoming more vital, and it is very necessary to have a procedure that is well established. In the course of this research, we devised a game in which malicious software and anti-malware are pitted against one another in an effort to maximize the predicted benefit that they get from certain courses of action. The results of the experiments reveal that anti-malware software can effectively identify the kind of malware. Malware causes a great deal of trouble for the information community and is a major cause for worry for the people who utilize it. As a result, malware detection developed into an instrument that was necessary to combat malware. In this study, we make an effort to recognize data that has been modified by malware authors in an attempt to develop an effective malware detection system. The naive bayes classifier serves as the system's foundational component. We test our method with the most common measures, and the results of our experiments demonstrate that our method yields an effective result; it is able to identify malwares with a detection rate that is acceptable; and it is capable of outperforming naive bayes in all circumstances. We would want to augment our model using ensemble classifiers so that it can compete against more than one foe or many varieties of malware.

5.2 FUTURE WORK

For future work, it is possible to employ the use of other heuristic methods not yet used in this problem. Due to the diversity of IOT applications, there are many types of networks with particular characteristics (such as mobile sensors / cluster heads, sensors with different processing capacities and different types of data collection) that can be addressed using clustering with meta-heuristics.

REFERENCES

[1] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Comput. Secur.*, vol. 86, pp. 53–62, 2019, doi: 10.1016/j.cose.2019.05.022.

[2] S. Muller *et al.*, "A training-resistant anomaly detection system," *Comput. Secur.*, vol. 76, pp. 1–11, 2018, doi: 10.1016/j.cose.2018.02.015.

[3] K. Wang, "Network data management model based on Naïve Bayes classifier and deep neural networks in heterogeneous wireless networks," *Comput. Electr. Eng.*, vol. 75, pp. 135–145, 2019, doi: 10.1016/j.compeleceng.2019.02.015.

[4] R. Barnard, *Intrusion Detection Systems, 2nd Edition*. 1988. T. A. Alamiedy, M. Anbar, Z. N. M. Alqattan, and Q. M. Alzubi, "Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, 2019, doi: 10.1007/s12652-019-01569-8.

[5] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *KnowledgeBased Syst.*, vol. 136, pp. 130–139, 2017, doi: 10.1016/j.knosys.2017.09.014.

[6] P. Krishnan, S. Duttagupta, and K. Achuthan, "VARMAN: Multiplane security framework for software defined networks," *Comput. Commun.*, vol. 148, no. July, pp. 215–239, 2019, doi: 10.1016/j.comcom.2019.09.014.

[7] J. Liang, J. Chen, Y. Zhu, F. R. Yu, R. Yu, and F. R. Yu, "A novel Intrusion Detection System for Vehicular Ad Hoc Networks (VANETs) based on differences of traffic flow and position," *Appl. Soft Comput. J.*, vol. 75, pp. 712–727, 2019, doi: 10.1016/j.asoc.2018.12.001.

[8] M. Gajewski *et al.*, "Two-tier anomaly detection based on traffic profiling of the home automation system," *Comput. Networks*, vol. 158, pp. 46–60, 2019, doi: 10.1016/j.comnet.2019.04.013.

[9] X. An, J. Su, X. Lü, and F. Lin, “Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system,” *Eurasip J. Wirel. Commun. Netw.*, vol. 11, p. 249, 2018, doi: 10.1186/s13638-018-1267-2.

[10] E. M. Roopa Devi and R. C. Suganthe, “Improved Relevance Vector Machine (IRVM) classifier for Intrusion Detection System,” *Soft Comput.*, vol. 23, no. 19, pp. 9111–9119, Oct. 2019, doi: 10.1007/s00500-018-3621-z.

[11] C. Callegari, S. Giordano, and M. Pagano, “An information-theoretic method for the detection of anomalies in network traffic,” *Comput. Secur.*, vol. 70, pp. 351–365, 2017, doi: 10.1016/j.cose.2017.07.004.

[12] Y. Liu, L. Zhu, and F. Liu, “Design of Multimedia Education Network Security and Intrusion Detection System,” *Multimed. Tools Appl.*, 2020, doi: 10.1007/s11042-020-08724-w.

[13] I. Ghafir *et al.*, “Detection of advanced persistent threat using machine-learning correlation analysis,” *Futur. Gener. Comput. Syst.*, vol. 89, pp. 349–359, 2018, doi: 10.1016/j.future.2018.06.055.

[14] J. Manan, A. Ahmed, I. Ullah, L. Merghem-Boulahia, and D. Gaïti, “Distributed intrusion detection scheme for next generation networks,” *J. Netw. Comput. Appl.*, vol. 147, p. 102422, 2019, doi: 10.1016/j.jnca.2019.102422.

[15] Y. Al-Hadhrami and F. K. Hussain, “Real time dataset generation framework for intrusion detection systems in IoT,” *Futur. Gener. Comput. Syst.*, vol.108, pp. 414–423, Jul. 2020, doi: 10.1016/j.future.2020.02.051.

[16] C.-R. R. Wang, R.-F. F. Xu, S.-J. J. Lee, and C.-H. H. Lee, “Network intrusion detection using equality constrainedoptimization-based extreme learning machines,” *Knowledge-Based Syst.*, vol. 147, pp. 68–80, 2018, doi: 10.1016/j.knosys.2018.02.015.

[17] W. Li, W. Meng, X. Luo, and L. F. Kwok, “MVPSys: Toward practical multi-view based false alarm reduction system in network intrusion detection,” *Comput. Secur.*, vol. 60, pp. 177–192, 2016, doi: 10.1016/j.cose.2016.04.007.

[18] P. Devan and N. Khare, “An efficient XGBoost–DNN-based classification model for network intrusion detection system,” *Neural Comput. Appl.*, 2020, [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00521-020-04708x.pdf.

[19] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee, and H. Karimipour, “Cyber intrusion detection by combined feature selection algorithm,” *J. Inf. Secur. Appl.*, vol. 44, pp. 80–88, 2019, doi: 10.1016/j.jisa.2018.11.007.

[20] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, “From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods,” *IEEE Commun. Surv. Tutorials Surv. Tutorials*, vol. 20, no. 4, p. 3369, 2018, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8410366/.

[21] B. Kitchenham *et al.*, *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007. X. F. Chen and S. Z. Yu, “CIPA: A collaborative intrusion prevention architecture for programmable network and SDN,” *Comput. Secur.*, vol. 58, pp. 1–19, 2016, doi: 10.1016/j.cose.2015.11.008.

[22] M. M. Rathore, A. Ahmad, and A. Paul, “Real time intrusion detection system for ultra-high-speed big data environments,” *J. Supercomput.*, 2016, [Online]. Available: https://link.springer.com/article/10.1007/s11227-015-1615-5.

[23] C. Azad and V. K. Jha, “Fuzzy min–max neural network and particle swarm optimization-based intrusion detection system,” *Microsyst. Technol.*, vol. 23, no. 4, pp. 907–918, 2016, doi: 10.1007/s00542016-2873-8.

[24] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, “Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,” *Expert Syst. Appl.*, vol. 67, pp. 296–303, 2017, doi: 10.1016/j.eswa.2016.09.041.

[25] X. Zou, J. Cao, Q. Guo, and T. Wen, “A novel network security algorithm based on improved support vector machine from smart city perspective,” *Comput. Electr. Eng.*, vol. 65, no. 3, pp. 67–78, 2018, doi: 10.1016/j.compeleceng.2017.09.028.

[26] E. Kabir, J. Hu, H. Wang, G. Zhuo, H. Wang, and G. Zhuo, “A novel statistical technique for intrusion detection systems,” *Futur. Gener. Comput. Syst.*, vol. 79, pp. 303–318, 2018, doi: 10.1016/j.future.2017.01.029.

[27] W. Ghanem and A. Jantan, “A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm,” Neural Comput. Appl., 2019, [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00521-019-046552.pdf.

[28] A. Yang, Y. Zhuansun, C. Liu, J. Li, and C. Zhang, “Design of Intrusion Detection System for Internet of Things Based on Improved BP Neural Network,” *IEEE Access*, vol. 7, pp. 106043– 106052, 2019, doi: 10.1109/ACCESS.2019.2929919.

[29] D. Moon, S. B. Pan, and I. Kim, “Host-based intrusion detection system for secure human-centric computing,” *J. Supercomput.*, vol.

[30] 72, no. 7, pp. 2520–2536, 2016, doi: 10.1007/s11227-015-1506-9.

[31] S. Ramakrishnan and S. Devaraju, “Attack’s Feature SelectionBased Network Intrusion Detection System Using Fuzzy Control Language,” *Int. J. Fuzzy Syst.*, vol. 19, no. 2, pp. 316–328, 2017, doi: 10.1007/s40815-016-0160-6.

[32] C. Khammassi and S. Krichen, “A GA-LR wrapper approach for feature selection in network intrusion detection,” *Comput. Secur.*, vol. 70, pp. 255–277, 2017, doi: 10.1016/j.cose.2017.06.005.

[33] D. Santoro, G. Escudero-Andreu, K. G. Kyriakopoulos, F. J. Aparicio-Navarro, D. J. Parish, and M. Vadursi, “A hybrid intrusion detection system for virtual jamming attacks on wireless networks,” *Meas. J. Int. Meas. Confed.*, vol. 109, pp. 79–87, 2017, doi: 10.1016/j.measurement.2017.05.034.

[34] L. Li, H. Zhang, H. Peng, and Y. Yang, “Nearest neighbors based density peaks approach to intrusion detection,” *Chaos Solitons & Fractals*, vol. 110, pp. 33–40, 2018, doi: 10.1016/J.CHAOS.2018.03.010.

[35] M. Bouhaddi, M. S. Radjef, and K. Adi, “An efficient intrusion detection in resource-constrained mobile ad-hoc networks,” *Comput. Secur.*, vol. 76, pp. 156–177, 2018, doi: 10.1016/j.cose.2018.02.018.

[36] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. GarcíaTeodoro, and R. Therón, “UGR‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs,” *Comput. Secur.*, vol. 73, pp. 411–424, 2018, doi: 10.1016/j.cose.2017.11.004.

[37] M. Hawedi, C. Talhi, and H. Boucheneb, *Multi-tenant intrusion detection system for public cloud (MTIDS)*, vol. 74, no. 10. Springer US, 2018. G. Bhuvaneswari and G. Manikandan, “An intelligent intrusion detection system for secure wireless communication using IPSO and negative selection classifier,” *Cluster Comput.*, vol. 22, pp. 12429– 12441, 2018, doi: 10.1007/s10586-017-1643-4.

[38] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo, and X. Zeng, “A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network,” *IEEE Access*, vol. 7, pp. 154560–154571, 2019, doi: 10.1109/ACCESS.2019.2948382.

[39] S. C. Sethuraman, S. Dhamodara, and V. Vijayakumar, “Intrusion detection system for detecting wireless attacks in IEEE 802.11 networks,” *IET Networks*, vol. 8, no. 4, pp. 219–232, 2019, doi: 10.1049/iet-net.2018.5050.

[40] D. R. C. Canedo and A. R. S. R. Romariz, “Intrusion Detection System in Ad Hoc Networks with Artificial Neural Networks and Algorithm K-Means,” *IEEE Lat. Am. Trans.*, vol. 17, no. 7, pp. 1109–1115, 2019, doi: 10.1109/TLA.2019.8931198.

[41] E. Anthi, L. Williams, M. Słowi, G. Theodorakopoulos, and P. Burnap, “A Supervised Intrusion Detection System for Smart Home IoT Devices,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9042–9053, 2019, doi: 10.1109/JIOT.2019.2926365.

[42] N. Upasani and H. Om, “A modified neuro-fuzzy classifier and its parallel implementation on modern GPUs for real time intrusion detection,” *Appl. Soft Comput. J.*, vol. 82, p. 105595, 2019, doi: 10.1016/j.asoc.2019.105595.

[43] N. Chouhan, A. Khan, and H. ur R. Khan, “Network anomaly detection using channel boosted and residual learning based deep convolutional neural network,” *Appl. Soft Comput. J.*, vol. 83, p. 105612, 2019, doi: 10.1016/j.asoc.2019.105612.

[44] B. Selvakumar, K. Muneeswaran, S. B, M. K, and B. Selvakumar, “Firefly algorithm-based feature selection for network intrusion detection,” *Comput. Secur.*, vol. 81, pp. 148–155, 2019, doi: 10.1016/J.COSE.2018.11.005.

[45] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Veh. Commun.*, vol. 21, p. 100198, 2020, doi: 10.1016/J.VEHCOM.2019.100198.

[46] A. Nagaraja, B. Uma, and R. kumar Gunupudi, “UTTAMA: An Intrusion Detection System Based on Feature Clustering and Feature Transformation,” *Found. Sci.*, no. 0123456789, 2019, doi: 10.1007/s10699-019-09589-5.

[47] P. Anitha and B. Kaarthick, “Oppositional based Laplacian grey wolf optimization algorithm with SVM for data mining in intrusion detection system,” *J. Ambient Intell. Humaniz. Comput.*, 2019, doi: 10.1007/s12652-019-01606-6.

[48] Q. M. Alzubi, M. Anbar, Z. N. M. Alqattan, M. A. Al-Betar, and ..., “Intrusion detection system based on a modified binary grey wolf optimisation,” *Neural Comput. …*, 2019, [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00521-019-041031.pdf.

[49] J. H. Lee and K. H. Park, “GAN-based imbalanced data intrusion detection system,” *Pers. Ubiquitous Comput.*, 2019, doi: 10.1007/s00779-019-01332-y.

[50] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, “An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset,” *Cluster Comput.*, vol. 23, 2019, doi: 10.1007/s10586-019-03008-x.

[51] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, and Y. Venu Madhav, “A context-aware robust intrusion detection system: a reinforcement learning-based approach,” *Int. J. Inf. Secur.*, no. 1, 2019, doi: 10.1007/s10207-019-00482-7.

[52] I. Dutt, S. Borah, and I. K. Maitra, “Immune System Based Intrusion Detection System (IS-IDS): A Proposed,” *IEEE Access*, vol. 8, pp. 34929–34941, 2020, doi: 10.1109/ACCESS.2020.2973608.

[53] A. S. Alzahrani, R. A. Shah, Y. Qian, and M. Ali, “A novel method for feature learning and network intrusion classification,” *Alexandria Eng. J.*, 2020, doi: 10.1016/j.aej.2020.01.021.

[54] W. Elmasry, A. Akbulut, and A. H. Zaim, “Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic,” *Comput. Networks*, vol. 168, 2020, doi: 10.1016/j.comnet.2019.107042.

[55] E. Mahdavi, A. Fanian, and F. Amini, “A real-time alert correlation method based on code-books for intrusion detection systems,” *Comput. Secur.*, vol. 89, p. 101661, 2020, doi: 10.1016/j.cose.2019.101661.

[56] C. Stylianopoulos, M. Almgren, O. Landsiedel, and M. Papatriantafilou, “Multiple pattern matching for network security applications: Acceleration through vectorization,” *J. Parallel Distrib. Comput.*, vol. 137, pp. 34–52, 2020, doi: 10.1016/j.jpdc.2019.10.011.

[57] Y. Li *et al.*, “Robust detection for network intrusion of industrial IoT based on multi-CNN fusion,” *Measurement*, vol. 154, p. 107450, 2020, doi: 10.1016/J.MEASUREMENT.2019.107450.

[58] W. Zhang, D. Han, K.-C. Li, and F. I. Massetto, “Wireless sensor network intrusion detection system based on MK-ELM,” *Soft Comput.*, Jan. 2020, doi: 10.1007/s00500-020-04678-1.

[59] A. Davahli, M. Shamsi, and G. Abaei, “Hybridizing genetic algorithm and grey wolf optimizer to advance an intelligent and lightweight intrusion detection system for IoT wireless networks,” *J. Ambient Intell. Humaniz. Comput.*, no. 0123456789, 2020, doi: 10.1007/s12652-020-01919-x.

[60] A. N. Jaber and S. U. Rehman, “FCM-SVM based intrusion detection system for cloud computing environment,” *Cluster Comput.*, vol. 6, 2020, doi: 10.1007/s10586-020-03082-6.

[61] S. Sandosh, V. Govindasamy, and G. Akila, “Enhanced intrusion detection system via agent clustering and classification based on outlier detection,” *Peer-to-Peer Netw. Appl.*, 2020, doi: 10.1007/s12083-019-00822-3.

[62] R. B. Benisha and S. R. Ratna, “Detection of data integrity attacks by constructing an effective intrusion detection system,” *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 10, 2020, doi: 10.1007/s12652-020-01850-1.

[63] S. Venkatraman and B. Surendiran, “Adaptive hybrid intrusion detection system for crowd sourced multimedia internet of things systems,” *Multimed. Tools Appl.*, vol. 79, no. 5–6, pp. 3993–4010, 2020, doi: 10.1007/s11042-019-7495-6.