الجـــمـهـوريـــة الجـزائـريـة الديمـقـراطيـة الـشـعـبـيـة

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التـــعـلـيـــم العـــــالي والبـــحث العـــــلـــمـــي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جـــــــــــامعة سعيدة – د. الطاهر مولاي –

University of Saida - Dr. MOULAY TAHAR
Faculty of Technology

A Dissertation Submitted to the Department of Telecommunications in Partial Fulfilment of the Requirements for Degree of Master of Networks & Telecommunications

Presented by: Mr. MOHAMMEDI Houari
Mr. MAHSER Kheireddine

Intrusion Detection in IoT Networks using Deep Learning Algorithms

Defended on **June, 22 2024** in front of the jury composed of:

Dr. BOUYEDDOU Benamar	**MCA**	President
Dr. GUENDOUZ Mohamed	**MCA**	Supervisor
Dr. OUIS. Esma	**MAB**	Examiner

2023 / 2024

Acknowledgment

*We would like to thank first and foremost **ALLAH** the Almighty, who gave us the strength and patience to accomplish our work in the best conditions.*

We thank our family for the sacrifices they have made so that we can complete our studies.

*To our supervisor **Dr. GUENDOUZ Mohamed**; for having agreed to take charge and for his precious advice and his help throughout the work period.*

We would also like to thank the members of the jury for their interest in our work by agreeing to examine our work and enrich it with their proposals.

Finally, our thanks go to all the professors and teaching staff at the University of Saida Dr Moulay Tahar, especially the telecommunications department. who have accompanied us throughout these years of study, to all the students in our class and to anyone who has contributed directly or indirectly to the development of this graduation project.

Dedication

I dedicate this modest work as a sign of expression of all my affection and my great gratitude to:

My dear parents, ***BOUHAFS & ARBIA*** *who gave Me existence, love, support, I'm forever in your debt.*

My brothers ***AHMED, MAAZOUZ*** *and* ***ALI*** *your support has no limit so thank you for always having my back.*

My sisters ***AICHA*** *and* ***KHEIRA*** *for always believing in me, you are my treasures.*

My brother-in-law ***KARIM*** *your support in one of a kind.*

My sister-in-law ***SOUAD*** *thank you for always caring.*

My adorable nieces ***SHAHID*** *&* ***LOJAIN*** *you bring joy to my life.*

My dear friends ***MUSTAPHA, BRAHIM, WISSAM*** *I'm lucky having you as my friends.*

My colleague and Brother ***MAHSER KHEIREDDINE*** *this wouldn't be possible without you; you've always been a kind brother to me so thank you.*

To ***Mr. GUENDOUZ Mohamed****, I thank you for your teachings, patience, advice, and for always bearing with me.*

Houari

Dedication

I am profoundly grateful to those who have made this thesis possible through their unwavering support and encouragement.
Firstly, I would like to express my deepest gratitude to my family. Your constant love, understanding, and sacrifices have been my greatest source of strength. You have supported me in every way possible, and for that, I am forever indebted.
I am also sincerely thankful to our administration: your guidance, resources, and encouragement have been instrumental in the completion of this research. Special thanks to my advisor for his invaluable advice, patience, and constructive feedback throughout this journey.
Lastly, I extend my heartfelt thanks to my friends: your moral support, and belief in my abilities have been crucial. Thank you for standing by me through the challenges and celebrating the achievements with me.
Thanks to my partner in this unforgettable journey for his kindness and seriousness.

This accomplishment would not have been possible without each of you. Thank you.

Kheireddine

Abstract

The internet has become an inseparable part of human life, and the number of devices connected to the internet is increasing sharply. In particular, Internet of Things (IoT) devices have become a part of everyday human life. However, some challenges are increasing, and their solutions are not well defined. More and more challenges related to technology security concerning the IoT are arising. Many methods have been developed to secure IoT networks, but many more can still be developed. One proposed way to improve IoT security is to use deep learning. This research discusses several deep-learning strategies, as well as standard datasets for improving the security performance of the IoT. We developed an algorithm for detecting denial-of-service (DoS) attacks using a deep-learning algorithm. This research used the Python programming language with packages such as scikit-learn, TensorFlow, and Keras. We found that a deep-learning model could increase accuracy so that the mitigation of attacks that occur on an IoT network is as effective as possible.

Keywords: deep learning; Internet of Things; distributed denial-of-service attack; intrusion detection.

Résumé

Internet est devenu un élément indissociable de la vie humaine et le nombre d'appareils connectés à Internet augmente fortement. En particulier, les appareils Internet des objets (IoT) font désormais partie de la vie humaine quotidienne. Cependant, certains défis se multiplient et leurs solutions ne sont pas bien définies. De plus en plus de défis liés à la sécurité technologique concernant l'IoT apparaissent. De nombreuses méthodes ont été développées pour sécuriser les réseaux IoT, mais bien d'autres peuvent encore être développées. L'un des moyens proposés pour améliorer la sécurité de l'IoT consiste à utiliser l'apprentissage automatique. Cette recherche aborde plusieurs stratégies d'apprentissage profond, ainsi que des ensembles de données standard pour améliorer les performances de sécurité de l'IoT. Nous avons développé un algorithme de détection des attaques par déni de service (DoS) à l'aide d'un algorithme d'apprentissage en profondeur. Cette recherche a utilisé le langage de programmation Python avec des packages tels que scikit-learn, TensorFlow. Nous avons constaté qu'un modèle d'apprentissage en profondeur pourrait accroître la précision afin que l'atténuation des attaques qui se produisent sur un réseau IoT soit aussi efficace que possible.

Mots-clés : l'apprentissage en profondeur ; Internet des objets ; attaque par déni de service distribué ; détection d'intrusion.

ملخص

أصبح الإنترنت جزءًا لا يتجزأ من حياة الإنسان، ويتزايد عدد الأجهزة المتصلة بالإنترنت بشكل حاد. على وجه الخصوص، أصبحت أجهزة إنترنت الأشياء (IoT) جزءًا من الحياة البشرية اليومية. ومع ذلك، فإن بعض التحديات آخذة في التزايد، ولم يتم تحديد حلولها بشكل جيد. تظهر المزيد والمزيد من التحديات المتعلقة بأمن التكنولوجيا فيما يتعلق بإنترنت الأشياء. لقد تم تطوير العديد من الطرق لتأمين شبكات إنترنت الأشياء، ولكن لا يزال من الممكن تطوير العديد من الطرق الأخرى. إحدى الطرق المقترحة لتحسين أمان إنترنت الأشياء هي استخدام التعلم الآلي. يناقش هذا البحث العديد من استراتيجيات التعلم العميق، بالإضافة إلى مجموعات البيانات القياسية لتحسين الأداء الأمني لإنترنت الأشياء. لقد قمنا بتطوير خوارزمية للكشف عن هجمات رفض الخدمة (DoS) باستخدام خوارزمية التعلم العميق. استخدم هذا البحث لغة البرمجة بايثون مع حزم مثل scikit-learn و .TensorFlow لقد وجدنا أن نموذج التعلم العميق يمكن أن يزيد من الدقة بحيث يكون التخفيف من الهجمات التي تحدث على شبكة إنترنت الأشياء فعالاً قدر الإمكان.

الكلمات المفتاحية: ؛ تعلم عميق؛ انترنت الأشياء؛ هجوم حجب الخدمة الموزع؛ كشف التسلل.

Table of Contents

Table of Contents

List Of Figures

List Of Tables

List Of Equations

List Of Abbreviations

ARPA: Advanced Research Projects Agency.

ARPANET: Advanced Research Projects Agency Network.

AMQP: Advanced Message Queuing Protocol.

APIDS: Application Protocol-based IDS.

API: Application Programming Interface.

AI: Artificial Intelligence.

BMU: Best Matching Unit.

CDMA: Code-division Multiple Access.

COAP: Constrained Application Protocol.

CNN: Convolutional Neural Networks.

DDS: Data Distribution Service.

DOS: Denial of Service.

DBN: Deep Belief Networks.

FTP: File Transfer Protocol.

FNN: Feed-Forward Neural Networks.

GSM: Global System for Mobile Communications.

GPS: Global Positioning System.

GAN: Generative Adversarial Networks.

HTTP: Hypertext Transfer Protocol.

HIDS: Host Intrusion Detection System.

IEEE: Institute of Electrical and Electronics Engineers.

IoT: Internet of Things.

IoE: Internet of Everything.

IoNT: Internet of Nano Things.

IoMCT: Internet of Mission Critical Things.

IoMT: Internet of Mobile Things.

IP: Internet Protocol.

IPv4: Internet Protocol version 4.

IPv6: Internet Protocol version 6.

ID: Intrusion Detection.

IDS: Intrusion Detection System.

LAN: Local Area Network.

LED: Light-emitting diode.

LR-WPAN: Low-Rate Wireless Personal Area Networks.

LO-WPAN: IPv6 over Low-Power Wireless Personal Area Networks.

LSTM: Long Short-Term Memory Networks.

MQTT: Message Queue Telemetry Transport.

MQTTSA: MQTT Security Assistant.

MitM: Man-in-the-Middle Attacks.

MLP: Multilayer Perceptron.

NIDS: Network Intrusion Detection System.

NLP: Natural language processing.

OSI: Open Systems Interconnection.

PIDS: Protocol-based IDS.

RF: Radio frequency.

RFID: Radio Frequency Identification.

ReLu: Restricted Linear Unit.

RNN: Recurrent Neural Network.

RBFN: Radial Basis Function Network.

RBM: Restricted Boltzmann Machines.

SOM: Self Organizing Maps.

SlowITe: Slow DoS against Internet of Things Environments Attack.

TCP: Transmission Control Protocol.

Tanh: Hyperbolic Tangent.

UDP: User Datagram Protocol.

UTMS: Universal Mobile Telecommunications System.

Wi-Fi: Wireless Fidelity.

Wi-Max: Worldwide Interoperability for Microwave Access.

WLAN: Wireless Local-Area Network.

XMPP: Extensible Messaging and Presence Protocol.

XML: Extensible Markup Language.

General Introduction

Inventors have long dreamed of creating machines that think, a desire that dates back to ancient Greece. Mythical figures such as Pygmalion, Daedalus, and Hephaestus can be interpreted as legendary inventors, with their creations—Galatea, Talos, and Pandora—representing early imaginings of artificial life.

The concept of programmable computers spurred speculation about machine intelligence over a century before such devices were realized. Today, artificial intelligence (AI) is a thriving field with numerous practical applications and active research areas. Intelligent software now automates routine labor, understands speech and images, makes medical diagnoses, and supports scientific research.

Initially, AI rapidly addressed and solved problems that, while intellectually challenging for humans, were relatively straightforward for computers due to their formal, mathematical nature. The real challenge for AI, however, lay in tasks that are easy for humans to perform intuitively but difficult to describe formally, such as recognizing spoken words or identifying faces in images.

Early AI successes were often confined to formal environments, exemplified by IBM's Deep Blue chess-playing system. Despite these achievements, recognizing objects or speech in less structured settings remained difficult for computers. One significant challenge in AI has been capturing informal knowledge in a format accessible to machines. Several projects attempted to hard-code world knowledge into formal languages, but none achieved substantial success.

The advent of machine learning marked a significant shift, enabling computers to tackle problems involving real-world knowledge and make seemingly subjective decisions. Simple algorithms like logistic regression and naive Bayes could recommend cesarean deliveries or filter spam emails, but their performance heavily depended on data representation.

This dependence on representation is a pervasive phenomenon in computer science and daily life. For instance, searching a structured and intelligently indexed data collection can be exponentially faster. In machine learning, the choice of data representation significantly impacts algorithm performance.

AI tasks often involve designing appropriate features for a task and providing them to a simple learning algorithm. For many tasks, such as detecting cars in photographs, identifying the right features is challenging. Representation learning addresses this by using machine learning to discover both the mapping from representation to output and the representation itself. Learned representations frequently outperform hand-designed ones, enabling AI systems to adapt to new tasks with minimal human intervention.

A quintessential representation learning algorithm is the autoencoder, comprising an encoder that converts input data into a different representation and a decoder that reverts it to the original format. Various autoencoders aim to achieve different properties.

Deep learning addresses the central problem in representation learning by introducing representations defined in terms of simpler ones. It allows computers to construct complex concepts from simpler ones, exemplified by the feedforward deep network or multilayer perceptron (MLP). An MLP is a mathematical function mapping input values to output values through a composition of many simpler functions.

Deep learning also enables computers to learn multi-step programs, with each representation layer acting as the state of the computer's memory after executing another set of parallel instructions. Deeper networks can execute more sequential instructions, enhancing power as later instructions can build on earlier results.

The depth of a model can be measured by the number of sequential instructions required to evaluate its architecture and the depth of the graph describing concept relationships. This depth is crucial in deep learning, allowing systems to refine simpler concepts based on more complex ones. Deep learning is currently the most viable approach for building AI systems capable of operating in complex, real-world environments, representing the world as a nested hierarchy of concepts.

The Internet of Things (IoT) phenomenon has been fueled by recent advancements in networking technologies and the widespread availability of various smart gadgets over the past decade. IoT enables physical electronic devices, such as sensors, to connect to the Internet, facilitating data collection and sharing among networked objects. Today, numerous IoT systems are deployed across various industries, including smart farming, industry, transportation, healthcare, and smart cities.

Several communication protocols have been developed to enhance the security and reliability of data exchange among IoT devices, including the Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), Message Queuing Telemetry Transport (MQTT), and Extensible Messaging Presence Protocol (XMPP). Among these, MQTT is the most popular in IoT systems due to its support for low-bandwidth connectivity, minimal memory requirements, and reduced packet loss.

MQTT, a lightweight messaging protocol using a publisher/subscriber architecture, simplifies device-to-device communication. However, this communication model can introduce security vulnerabilities, such as denial of service, identity spoofing, information exposure, privilege escalation, and data tampering.

To address these security concerns, researchers have developed various techniques and methods. Intrusion Detection Systems (IDS) are among the most effective solutions, identifying intrusions by monitoring system activities and distinguishing between legitimate use and attacks. An IDS designed for IoT-based environments must meet strict specifications for minimal processing power, fast response times, and high-volume data processing, making conventional IDS potentially unsuitable for IoT scenarios.

Integrating AI, particularly deep learning, with IoT can enhance the effectiveness of IDS in smart environments. Deep learning's ability to learn complex representations and adapt to new data with minimal human intervention can significantly improve intrusion detection accuracy and efficiency in IoT systems. This synergy between AI and IoT represents a promising frontier for developing robust, intelligent, and secure systems capable of operating in diverse and dynamic real-world environments.

In this project, we will integrate deep learning algorithms with IoT networks to elevate the effectiveness of intrusion detection systems. We will use deep learning algorithms to create an intelligent system that can identify intrusions in an Internet of Things network using the MQTT protocol while accounting for the computing and storage capacities of individual IoT devices.

Our manuscript is divided into four sections, which are categorized as follows:

The broad introduction of this manuscript provides a conceptual overview of our study.

An overview of the idea, forms, architectures, and application domains of Internet of Things networks is given in the first chapter. It also describes the parts that make up IoT networks and how they work. This chapter concludes by discussing the MQTT communication protocol, its main elements and architecture, potential assaults on the protocol, and solutions to lessen such threats.

The state of the art for intrusion detection systems is presented in the second chapter. The chapter provides examples of the types, structures, and specific assaults that can be prevented by IDS, as well as the specifics of its framework.

The specifics of deep learning algorithms are covered in the third chapter. This section lists various commonly used deep learning algorithms.

The foundation of our experiment are presented in the last chapter. It provides a quick overview of the used gear and software. It also gives an explanation of the evaluation measures that were used for the algorithms' assessment as well as the dataset that was used in the experiment.

The recollection concludes with a broad summary that highlights the documented findings of this study as well as the researchers' future objectives.

Chapter I: Internet of Things

I.1 Introduction:

The telegraph's ability to transfer information over long distances by means of a coded signal dates back to the early 19th century, and although there are earlier examples of networked electrical equipment, the Internet of Things actually emerged in the late 1960s. Around that time, a number of well-known researchers started looking into ways to link systems and computers. The network known as ARPANET, which was developed as a precursor to the Internet today by the U.S. Defense Department's Advanced Research Projects Agency (ARPA), is a good example of this effort. Businesses, governments, and consumers started looking into ways to link personal computers (PCs) and other technologies to each other in the late 1970s. By the 1980s, local area networks, or LANs, offered a popular and efficient means of real-time document sharing, data sharing, and other information exchange among a number of PCs. [1]

When the Internet began to expand such capabilities globally in the mid-1990s, scientists and researchers started looking into ways that humans and robots could interact more effectively. The term "the Internet of Things" was first used in a speech by British technologist Kevin Ashton, cofounder of the Auto-ID Center at MIT, in 1999. Ashton started investigating radio-frequency identification (RFID) in 1997 as a technological framework that would enable physical devices to connect via microchips and wireless signals. A more reliable framework for gathering, storing, analyzing, and sharing data was established in a matter of years thanks to cellphones, cloud computing, increases in processing power, and enhanced software algorithms. Simultaneously, advanced sensors emerged that could detect motion, temperature, moisture content, wind direction, sound, light, pictures, vibrations, and a host of other parameters—in addition to having the capacity to geolocate a person or a device. Real-time communication with both digital and physical items is now possible because to these advancements. For instance, one may view the location of an object, such a wallet or bag, by attaching a tracking chip, like an Apple Air Tag, to it. If a digital device is lost or stolen, the same chip that powers it can be used to trace its location. Then, it became feasible to connect people and things in a nearly ubiquitous manner with the broad use of mobile devices like smartphones and tablets and the advent of ubiquitous wireless communication. Consequently, industrial robotics systems, linked storage tanks, and intelligent traffic networks became standard. [1]

IoT development is still ongoing. These days, it can be utilized for a wide range of applications, such as artificial intelligence for extremely complex simulations, sensing systems for identifying contaminants in water sources, and agricultural and animal monitoring systems. For instance, it is now feasible to remotely apply

the ideal amounts of water, fertilizer, and pesticides to crops as well as monitor the whereabouts and health of animals. [1]

Airlines and shipping industries can optimize fleets for maximum loads and efficiencies by accounting for mechanical issues and weather through the use of highly networked systems. Real-time maps and navigation recommendations that redirect and route drivers based on traffic patterns are made available to drivers by the Internet of Things. These solutions save time and money while lowering traffic and pollution. [1]

The Internet of Things (IoT) is the wide range of physical things that have sensors and software installed in order to collect and exchange data across a network and communicate with one another with minimal assistance from humans. The term "Internet of Things" (IoT) refers to the plethora of "smart," computer-like devices that are so widely used today. These "things" include phones, appliances, thermostats, lighting controls, irrigation systems, security cameras, cars, even cities and animals. These devices can communicate with each other or with the Internet via wireless networks. These days, transponders let automobiles pass through tollbooths and pay the cost electronically, smart speakers add things to shopping lists and turn lights on and off, and smart watches track activity and steps. [1]

Complex tasks that are occasionally beyond the capacity of humans are made simpler and more automated via the Internet of Things. There are currently billions of linked devices that make up the Internet of Things. [1]

I.2 Definition of IoT

IoT is a Dynamic global network infrastructure with self-configuring capabilities built on open and compatible communication protocols, where virtual and physical "things" are seamlessly integrated into the information network and frequently communicate user and environment-related data. These "things" have identities, physical characteristics, and virtual personalities. They also use intelligent interfaces.

Let's take a closer look at this definition of IoT to clarify a few words: **Dynamic and Self-Adapting:** Internet of Things (IoT) devices and systems may be able to adjust themselves to changing environments on the fly and take appropriate action according to their operational parameters, the context of their users, or their detected surroundings. Take into consideration, for instance, a surveillance system that consists of several surveillance cameras. Depending on the time of day, the surveillance cameras can switch between standard and infra-red night modes. When motion is detected, cameras have the ability to automatically convert between lower and higher resolution modes, notifying other surrounding cameras to follow suit. In this instance, the monitoring system

is adjusting to the environment and shifting circumstances, such as those that are dynamic.

Self-Configuring: Internet of Things (IoT) devices have the potential to configure themselves, enabling a multitude of devices to collaborate in order to perform a certain purpose (like weather monitoring). With little assistance from the user or physical labor, these devices can autonomously configure themselves (in relation to the Internet of Things architecture), set up networking, and download the most recent software updates.

Interoperable Communication Protocols: Internet of Things (IoT) devices have the ability to communicate with other devices and the infrastructure by supporting several interoperable communication protocols. In the sections that follow, we go over a few of the popular communication models and protocols.

Unique Identity: Every Internet of Things (IoT) device is identified by a unique identifier, which can be an IP address or URI. Intelligent interfaces on Internet of Things devices could be able to communicate with users and their surroundings, adapting to the situation.

Through the use of IoT device interfaces, users can remotely control, configure, and monitor devices as well as query and monitor their status.

Integrated into Information Network: In order to enable communication and data exchange with other devices and systems, Internet of Things (IoT) devices are typically integrated into information networks. IoT devices have the ability to describe themselves (and their attributes) to other devices or user applications. They can also be dynamically discovered in the network, by other devices or by the network itself. For instance, in order for two connected nodes to interact and share data, a weather monitoring node can explain its monitoring capabilities to the other node. IoT systems become "smarter" as a result of integration into the information network because of the combined intelligence of the individual devices. working along with the infrastructure, as a result, it is possible to combine and evaluate data from numerous linked IoT nodes that monitor the weather to provide weather predictions. [2] [3]

I.3 Things in IoT

In the context of the Internet of Things, "things" typically refer to low-power, individually identifiable devices with remote sensing, actuation, and monitoring capabilities. Connected objects (IoT) have the ability to exchange data (directly or indirectly) with other connected objects and applications, gather data from other devices and process it locally, or send it to centralized servers or cloud-based application back-ends for processing. They can also carry out certain tasks locally and other tasks within the IoT infrastructure, depending on temporal and spatial limitations (memory, processing capabilities, communication latencies and speeds, and deadlines).[2]

IoT devices can also come in a variety of forms, such as smart watches, autos, industrial machinery, LED lights, and wearable sensors. Nearly every Internet of Things device produces data in one way or another. This data, when analyzed by data analytics tools, provides insightful knowledge that may be used to direct

additional local or distant actions. For example, sensor data collected by a garden's soil moisture monitoring equipment can be used to determine the best watering schedules. [2]

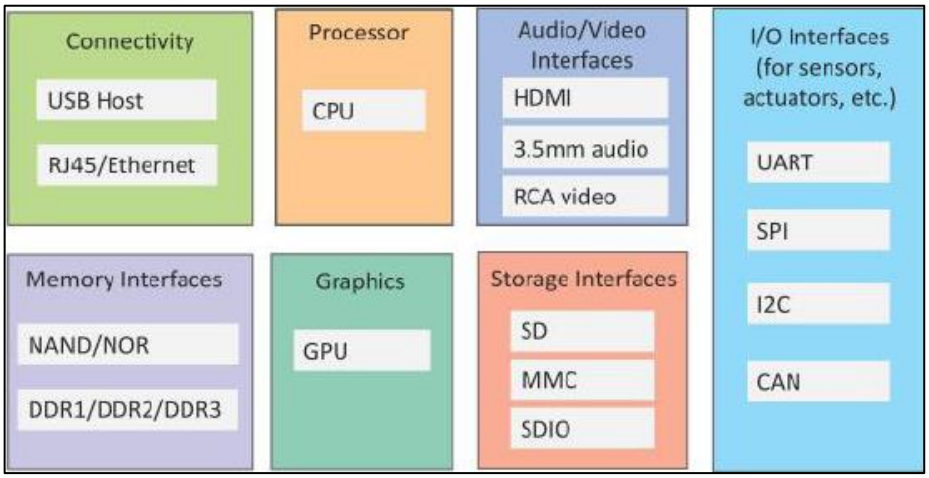


Figure I.1: Generic block diagram of an IoT Device.

I.4 IoT Protocols

I.4.1 Ethernet

IEEE 802.3 is a collection of wired Ethernet standards for the link layer, these standards provide data rates from 10 Mb/s to 40 Gb/s and higher. The shared medium in Ethernet can be a coaxial cable, twisted-pair wire or an optical fiber. The shared medium (i.e., broadcast medium) carries the communication for all the devices on the network, thus data sent by one device can received by all devices subject to propagation conditions and transceiver capabilities. [2]

I.4.2 Wi-Fi

IEEE 802.11 is a collection of wireless local area network (WLAN) communication standards, including extensive description of the link layer. These standards provide data rates from 1 Mb/s up to 6.75 Gb/s. [2]

I.4.3 Wi-Max

IEEE 802.16 is a collection of wireless broadband standards, including extensive descriptions for the link layer (also called Wi-Max). Wi-Max standards provide data rates from 1.5 Mb/s to 1 Gb/s. The recent update (802. 16m) provides data rates of 100 Mbit/s for mobile stations and 1 Gbit/s for fixed stations. [2]

I.4.4 LR-WPAN

IEEE 802,15.4 is a collection of standards for low-rate wireless personal area networks (LR-WPANs). These standards form the basis specifications for high level communication protocols such as ZigBee. LR-WPAN standards provides data rates from 40Kb/s to 250Kb/s. these standards provide low-cost and low-speed communication for power constrained devices. [2]

I.4.5 Mobile Communication (2G/3G/4)

There are different generations of mobile communication standards including second generation (2G including GSM and CDMA), third generation (3G including UTMS and CDMA2000) and fourth generation (4G -LTE). IoT devices based on these standards can communicate over cellular networks. Data rates for these standards range from 9.6Kb/s (for 2G) up to 100Mb/s (for 4G). [2]

I.4.6 IPv4

Internet Protocol version 4 (IPv4) is the most deployed Internet protocol that is used to identify the devices on a network using a hierarchical addressing scheme. IPv4 uses a 32-bit address scheme that allows total of 2^{32} or 4,294,967,296 addresses. As more and more devices got connected to the Internet, these addresses got exhausted in the year 2011. IPv4 has been succeeded by IPv6. The IP protocols establish connections on packet networks, but do not guarantee delivery of packets. Guaranteed delivery and data integrity are handled by the upper layer protocols (such as TCP). [2] [4]

I.4.7 IPv6

Internet Protocol version 6 (IPv6) is the newest version of Internet protocol and successor to IPv4. IPv6 uses 128-bit address scheme that allows total of 2^{128} or 3.4×10^{38} addresses. [2] [5]

I.4.8 6LoWPAN

6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) brings IP protocol to the low-power devices which have limited processing capability, 6LoWPAN operates in the 2.4 GHz frequency range and provides data transfer rates of 250 Kb/s. [2] [6]

I.4.9 TCP

Transmission Control Protocol (TCP) is the most widely used transport layer protocol, that is used by web browsers. TCP is a connection oriented and stateful protocol. While IP protocol deals with sending packets, TCP ensures reliable transmission of packets in-order. TCP also provides error detection capability so that duplicate packets can be discarded and lost packets are retransmitted. The flow control capability of TCP ensures that rate at which the sender sends the data is not too high for the receiver to process. The congestion control capability of TCP helps in avoiding network congestion and congestion collapse which can lead to degradation of network performance. [7]

I.4.10 UDP

Unlike TCP, which requires carrying out an initial setup procedure, UDP is a connectionless protocol. UDP is useful for time-sensitive applications that have very small data units to exchange and do not want the overhead of connection

setup. UDP is a transaction oriented and stateless protocol. UDP does not provide guaranteed delivery, ordering of messages and duplicate elimination. Higher levels of protocols can ensure reliable delivery or ensuring connections created are reliable.[8]

I.4.11 HTTP

Hypertext Transfer Protocol (HTTP) is the application layer protocol that forms the foundation of the World Wide Web. The protocol follows a request-response model where a client sends requests to a server using the HTTP commands. HTTP is a stateless protocol and each HTTP request is independent of the other requests. [9]

I.4.12 CoAP

Constrained Application Protocol (CoAP) is an application layer protocol for machine-to-machine (M2M) applications, meant for constrained environments with constrained devices and constrained networks. Like HTTP, CoAP is a web transfer protocol and uses a request-response model, however it runs on top of UDP instead of TCP. CoAP uses a client-server architecture where clients communicate with servers using connectionless datagrams. [10]

I.4.13 WebSocket

WebSocket protocol allows full-duplex communication over a single socket connection for sending messages between client and server. WebSocket is based on TCP and allows streams of messages to be sent back and forth between the client and server while keeping the TCP connection open. The client can be a browser, a mobile application or an loT device.[11]

I.4.14 MQTT

Message Queue Telemetry Transport (MQTT) is a light-weight messaging protocol based on the publish-subscribe model. MQTT uses a client-server architecture where the client (such as an loT device) connects to the server (also called MQTT Broker) and publishes messages to topics on the server. The broker forwards the messages to the clients subscribed to topics. MQTT is well suited for constrained environment where the devices have limited processing and memory resources and the network bandwidth is low.[12]

I.4.15 XMPP

Extensible Messaging and Presence Protocol (XMPP) is a protocol for real time communication and streaming XML data between network entities. XMPP powers wide range of applications including messaging, presence, data syndication, gaming, multi-party chat and voice/video calls. XMPP allows sending small chunks of XML data from one network entity to another in near real-time. XMPP is a decentralized protocol and uses a client-server architecture,

XMPP supports both client-to-server and server-to-server communication paths. In the context of IoT, XMPP allows real-time communication between IoT devices.[13]

I.4.16 DDS

Data Distribution Service (DDS) is a data-centric middleware standard for device-to-device or machine-to-machine communication. DDS uses a publish-subscribe model where publisher (e.g. devices that generate data) create topics to which subscribers (e.g. devices that want to consume data) can subscribe. Publisher is an object responsible for data distribution and the subscriber is responsible for receiving published data. DDS provides quality-of-service (QoS) control and configurable reliability. [14]

I.4.17 AMQP

Advanced Message Queuing Protocol (AMQP) is an open application layer protocol for business messaging. AMQP supports both point-to-point and publisher/subscriber models, routing and queuing. AMQP brokers receive messages from publishers (e.g. devices or applications that generate data) and route them over connections to consumers (applications that process data), Publishers publish the messages to exchanges which then distribute message copies to queues, Messages are either delivered by the broker to the consumers which have subscribed to the queues or the consumers can pull the messages from the queues.[2]

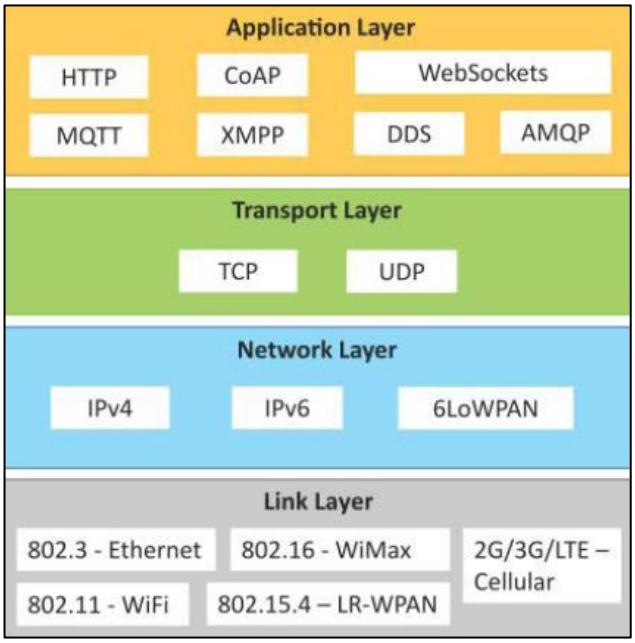


Figure I.2: IoT Protocols.

I.5 Logical Design of IoT

An IoT system's logical design is an abstract depiction of its elements and operations that avoids delving into the finer points of implementation. We go over an IoT system's functional components in this part.[2]

I.5.1 IoT Functional Blocks

A multitude of functional blocks make up an Internet of Things system, giving it the ability to communicate, act, sense, identify, and management. These functional blocks are described as follows [2] :

- **Device:** An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions.
- **Communication:** The communication block handles the communication for the IoT System.
- **Services:** An IoT system uses various types of IoT services such as services for device monitoring, device control services, data publishing services and services for device discovery.
- **Management:** Management functional block provides various functions to govern the IoT system,
- **Security:** Security functional block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
- **Application:** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view or analyze the processed data.

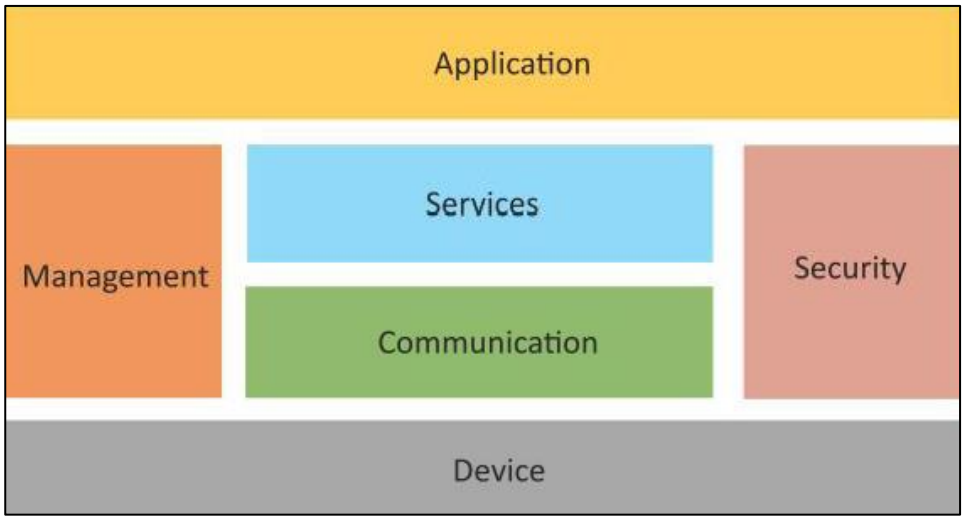


Figure I.3: Functional Blocks of IoT

I.5.2 IoT Communication Models

- **Request-Response communication model:** Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client. [2]

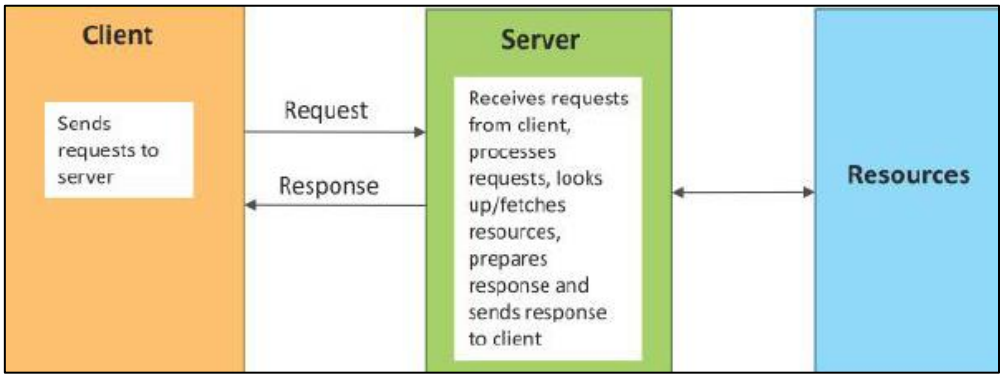


Figure I.4: Request-Response communication model.

- **Publish-Subscribe communication model:** Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers. [2]

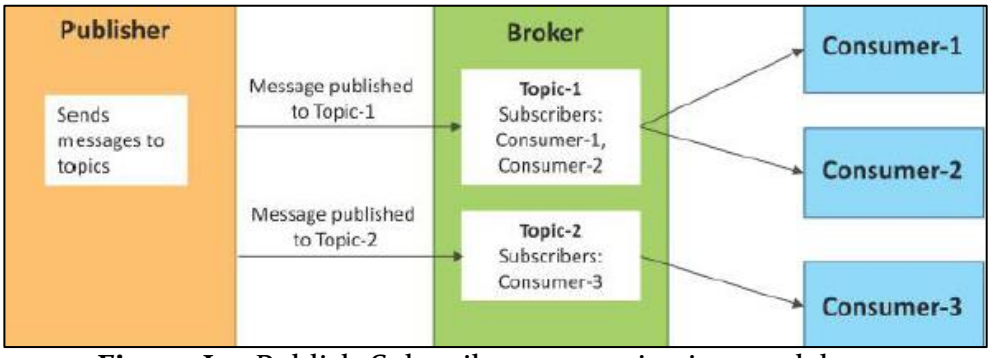


Figure I.5: Publish-Subscribe communication model.

- **Push-Pull communication model:** Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the producers and consumers. Queues also act as a buffer which helps in

situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data. [2]

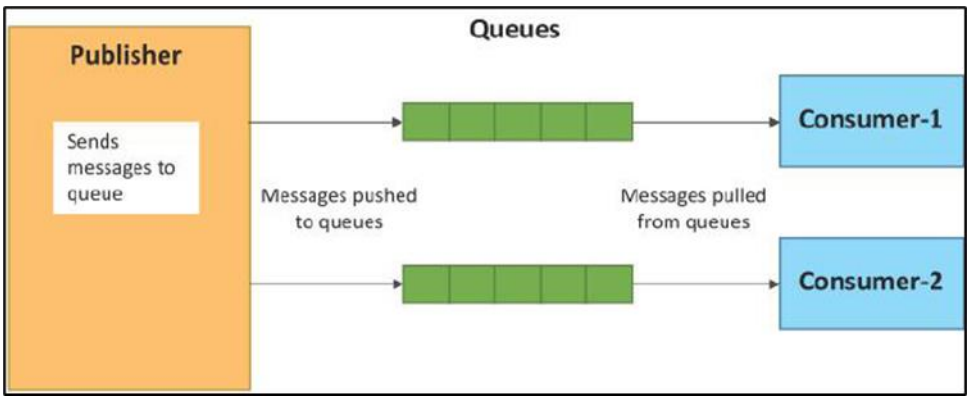


Figure I.6: Push-Pull communication model.

- **Exclusive Pair communication model:** Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server. Once the connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. [2]

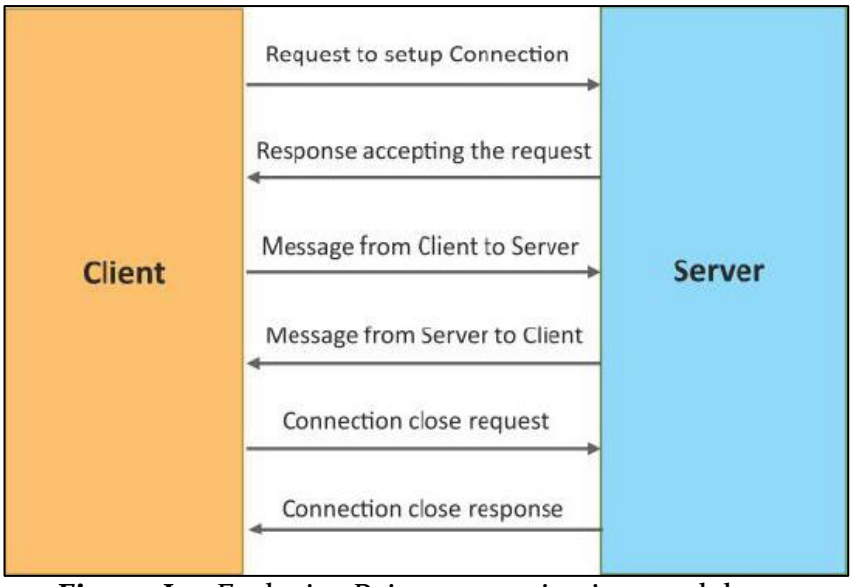


Figure I.7: Exclusive Pair communication model.

I.6 IoT Characteristics

The fundamental characteristics of the IoT are as follows [15] [16] [17]:

I.6.1 Interconnectivity

With regard to the IoT, anything can be interconnected with the global information and communication infrastructure.

I.6.2 Things-related services

Within the limitations of things, the Internet of Things can offer thing-related services like semantic consistency between actual objects and the virtual objects they are connected with, as well as privacy protection. Both the physical and information worlds' technologies will evolve to give thing-related services within the limitations of things.

I.6.3 Heterogeneity

Due to their varied hardware platforms and networks, IoT devices are heterogeneous. They can communicate via various networks with other gadgets or service platforms.

I.6.4 Dynamic changes

Device states fluctuate dynamically, such as whether they are asleep or waking up, connected or not, and in what context they are used, such as location and speed. Furthermore, the quantity of devices may vary on a dynamic basis.

I.6.5 Enormous scale

At least an order of magnitude more devices than those currently connected to the Internet will need to be controlled and communicate with one another. The handling of the produced data and its interpretation for use in applications will be even more crucial. This has to do with effective data processing and data semantics.

I.6.6 Safety

We must remember safety even while we reap the benefits of the Internet of Things. We must design for safety since we are the Internet of Things' producers and users. This covers both the security of our private information and the security of our physical health. The key to securing endpoints, networks, and the data that flows between them is developing a scalable security paradigm.

I.6.7 Connectivity

Network compatibility and accessibility are made possible via connectivity. Connecting to a network is known as accessibility, whereas sharing the capacity to create and use data is known as compatibility.

I.7 Types of IoT Technologies

I.7.1 Internet of Things (IoT)

The Internet of Things (IoT) connects physical objects via the internet, enabling them to identify and manage each other using various sensing devices like RFID and GPS. The IoT board consists of Arduino/Raspberry Pi, RF Module, Sensor Module, Access Point, IoT Server, and Cloud Point.[27]

I.7.2 Internet of Everything (IoE)

The Internet of Everything (IoE) is the new age of IoT, focusing on people, processes, and data, while IoT focuses on physical objects. IoE analyzes real-time data from millions of sensors to support automated processes and integrate industrial policy goals with ecological sustainability, social, and economic issues. It can also be used to make learning new technologies easier for students in educational systems. [27] [30]

I.7.3 Internet of Nano Things (IoNT)

The Internet of Nano Things (IoNT) connects nanoscale objects to communication networks, combining nano components into a single gadget. It differs from the Internet of Things (IoT) as it cannot incorporate nano components. Nanodevices communicate through conventional protocols, with the inbuilt network remotely controlled through a gateway. IoNT applications include gas detection systems and nano-micro interface devices. [27] [29]

I.7.4 Internet of Mission Critical Things (IoMCT)

The Internet of Mission Critical Things (IoMCT) combines sensing, communication, processing, and control to enhance network surveillance. It focuses on managing information sources, devices, and networks individually, reducing human strain on critical missions like combat, border patrol, and search and rescue. [27] [28]

I.7.5 Internet of Mobile Things (IoMT)

Digital devices, such as phones, are becoming increasingly integrated with sensors, allowing for interaction. The main difference between IoT and IoMT lies in context, connectivity, energy availability, and privacy and security. Context refers to the phone's location and current ownership, while connectivity refers to its connection to networks. Mobile charging properties include energy availability. Privacy issues arise from unique phone features and locations, leading to identity mismatch and uniqueness theft.[27]

I.8 IoT Architecture

There is no one widely accepted architecture for the Internet of Things. Various researchers have put forth distinct architectures.

I.8.1 Three- and Five-Layer Architectures

The most basic architecture for the Internet of Things (IoT) is a three-layer architecture consisting of perception, network, and application layers. The perception layer is responsible for sensing and gathering information about the environment, while the network layer connects smart things and devices. The application layer delivers application-specific services to users, defining applications like smart homes and cities. However, this architecture is not

sufficient for research, as it often focuses on finer aspects of the IoT. A five-layer architecture, which includes processing and business layers, is another option, which includes perception, transport, processing, application, and business layers. The business layer manages the entire IoT system, including applications, business models, and user privacy. Another architecture inspired by human brain layers is the human brain, spinal cord, and network of nerves. [18] [20]

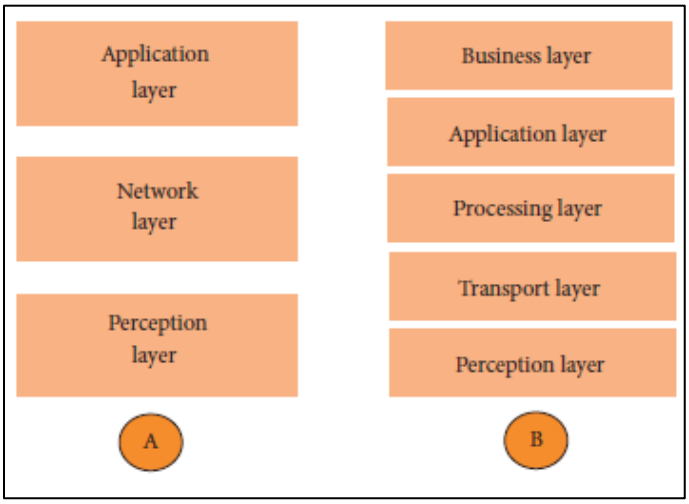


Figure I.8: Architecture of IoT (A: three layers) (B: five layers).

I.8.2 Cloud and Fog Based Architectures

Cloud and fog computing are two types of systems architectures used in IoT devices. Cloud computing is centralized, with applications at the center and smart things below. It offers flexibility, scalability, and services like core infrastructure, platform, software, and storage. Fog computing, on the other hand, involves sensors and network gateways in data processing and analytics. This architecture consists of monitoring, preprocessing, storage, and security layers between physical and transport layers. The monitoring layer monitors power, resources, responses, and services, while the preprocessing layer performs filtering, processing, and analytics of sensor data. Edge computing, on the other hand, adds smart data preprocessing capabilities to physical devices at the edge of the network. The distinction between protocol architectures and system architectures is not clear, and the generic 5-layer IoT protocol stack is used for both architectures.[21] [22]

I.9 IoT Applications

Some useful applications of Internet of Things (IOT) are [23] [24] [25] [26]:

- Connected Health
- Smart City
- Connected Cars
- Smart Home

- Smart Farming
- Smart Retail
- Smart Supply Chain

I.9.1 Connected Health

IoT has numerous applications in healthcare, including remote monitoring, smart sensors, and equipment integration. It can improve physician care and patient safety. IoT can enhance patient engagement and satisfaction by allowing more time for doctor-patient interaction. It offers pocket-friendly solutions for patients and healthcare professionals, empowering them to live healthier lives. Research shows IoT in healthcare will grow massively, enabling personalized health analysis and tailored treatment strategies.

I.9.2 Smart City

Another potent IoT application that is piquing people's interest around the globe is smart cities. Examples of internet of things applications for smart cities include automated transportation, water distribution, smarter energy management systems, smart surveillance, smart urban security, and environmental monitoring. Major issues that city dwellers suffer, such as pollution, gridlock in the streets, and a lack of energy, will be resolved by IoT. When a trash can needs to be emptied, devices like the Smart Belly trash can with cellular communication capability can notify local services. Installing sensors and utilizing online tools allows residents to locate open parking spaces all across the city. In addition, the sensors are capable of identifying general failures, installation problems with the electrical system, and meter manipulation.

I.9.3 Connected Cars

Automotive digital technology is focusing on enhancing the in-car experience with connected cars, which optimize operations, maintenance, and passenger comfort using onboard sensors and internet connectivity. Major brands like Tesla, BMW, Apple, and Google are working on connected car solutions, which consist of multiple sensors, antennas, embedded software, and technologies for consistent, accurate, speedy, and reliable decision-making.

I.9.4 Smart Home

In residential settings, smart homes have emerged as a revolutionary ladder to success, and it is anticipated that soon, smart homes will be as ubiquitous as smartphones. The most significant and effective use of IoT systems that always jumps out is the smart home, which is ranked as the top IOT application across all channels. Over $2.5 billion has been invested in smart home startups, and the figure is still rising. Wouldn't it be wonderful to be able to turn on the air conditioning before you get home or turn out the lights even after you've left?

Even when you're not home, you can provide friends temporary access by unlocking their doors. It should come as no surprise that businesses are developing items to make your life easier and more convenient as IoT takes shape. The largest outlay in a homeowner's life is the cost of house ownership. Products for smart homes are said to save money, energy, and time. Smart home firms, such as Nest, Ecobee, Ring, and August, among others, are poised to become household names and offer a never-before-seen experience.

I.9.5 Smart Farming

IoT applications like smart farming are frequently disregarded. But because farmers tend to a lot of livestock and their farming operations are typically spread out, the Internet of Things can monitor all of this and transform the way farmers operate. But widespread attention to this concept has not yet been received. However, it continues to be one of the IoT applications that is important to consider. Particularly in the nations that export agricultural products, smart farming has the potential to grow into a significant application area.

I.9.6 Smart Retail

Retailers are utilizing IoT solutions to enhance store operations, increase purchases, reduce theft, manage inventory, and improve consumer shopping experiences. This strategy allows retailers to compete with online competitors, regain lost market share, and attract consumers. Smartphones and Beacon technology can enhance in-store interactions, track consumer paths, and improve store layout, allowing for premium product placement in high-traffic areas.

I.9.7 Smart Supply Chain

For a few years already, supply networks have already begun to become more intelligent. Providing solutions for issues like tracking products while they are traveling or on the road, or assisting suppliers in exchanging inventory data, are a few of the well-liked offers. Factory equipment with embedded sensors can exchange data about many factors, including temperature, pressure, and machine utilization, through an Internet of things enabled system. In order to maximize performance, the IoT system can also process workflow and modify equipment settings.

I.10 MQTT Protocol:

The MQTT protocol (Message Queue Telemetry Transport) is a lightweight message queueing and transport protocol. MQTT, as its name implies, is suited for the transport of telemetry data (sensor and actor data). MQTT is very lightweight and thus suited for IoT (Internet of Things) scenarios where sensor and actor nodes communicate with applications through the MQTT message broker. MQTT is a text-based protocol designed for constrained IoT devices and low-bandwidth networks. Positioned in the application layer, it covers all 5th-7th

layers and requires 10 KB of RAM or flash for implementation. It uses TCP connection, but requires an open connection channel.[33]

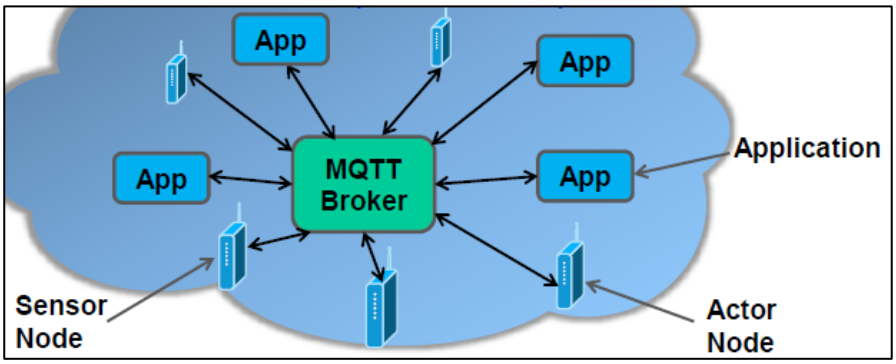


Figure I.9: Communication between sensor, actor nodes and application through MQTT broker.

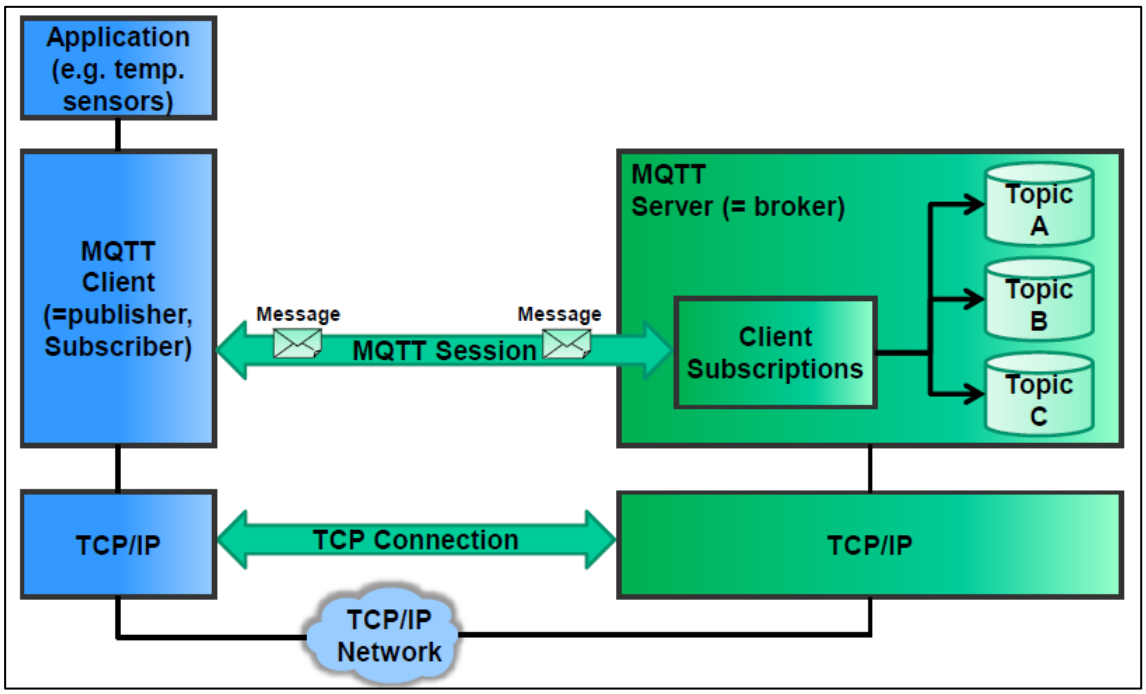


Figure I.10: MQTT Architecture.

I.10.1 MQTT Client (publisher/subscriber)

Clients subscribe to topics to publish and receive messages Thus subscriber and publisher are special roles of a client. [33]

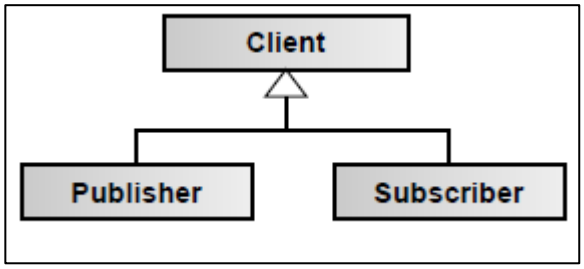


Figure I.11: Client roles.

I.10.2 MQTT Server (broker)

Servers run topics, i.e. receive subscriptions from clients on topics, receive messages from clients and forward these, based on client's subscriptions, to interested clients. [33]

I.10.3 Topic

Technically, topics are message queues. Topics support the publish subscribe pattern for clients. Logically, topics allow clients to exchange information with defined semantics. [33]

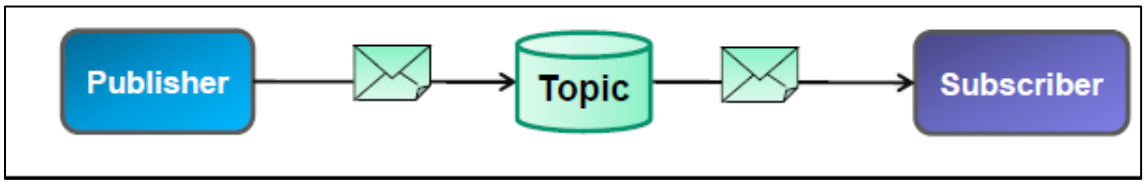


Figure I.12: Topic Example.

I.10.4 Session

A session identifies a (possibly temporary) attachment of a client to a server. All communication between client and server takes place as part of a session. [33]

I.10.5 Subscription

Unlike sessions, a subscription logically attaches a client to a topic. When subscribed to a topic, a client can exchange messages with a topic Subscriptions can be «transient» or «durable», depending on the clean session flag in the CONNECT message. [33]

I.10.6 Message

Messages are the units of data exchange between topic clients. MQTT is agnostic to the internal structure of messages. [33]

I.10.7 MQTT security

Application layer protocols like MQTT have several known and unreported security issues. MQTT's simplicity and scalability allow it to carry data between any IoT device via the application layer protocol, unlike any other protocol. [31] [34] [35]

I.10.7.a Solutions and needs for security in MQTT deployments

- **Authentication:** Making sure that MQTT network nodes may be identified in order to prevent unauthorized access (as producers or subscribers).
- **Access control:** limiting information access to just those nodes that are permitted access.

- **Data integrity:** confirming that there has been no tampering during transmission and that the data received matches the data provided by the source;
- **Confidentiality:** Data privacy and confidentiality must be maintained, and data sniffing must be stopped.

I.10.7.b Attacks and countermeasures

Man-in-the-Middle attack: A man in the middle (MitM) attack is a hacker's attempt to steal personal information from a user or application. It's typically carried out using MQTT protocols, which support two-way handshakes. To prevent MitM attacks, authentication and encryption are required. Attacking strategies include packet injection, session, SSL Stripping, SSL Hijacking, and sniffing. Preventing MitM attacks is crucial. [31] [34] [35]

DoS attack: A denial-of-service attack is a cyber-attack that disrupts a computer's normal operation by overloading or flooding it with requests. It aims to keep the broker busy, making it difficult to manage new incoming connections. Firewalls offer some protection against single-user attacks, but not DDoS attacks. Router access control lists, antivirus software, application protection, and network behavior analysis are developed to prevent such attacks.

Intrusion: Network intrusion refers to unauthorized activity on a computer network, often exploited by hackers using automated programs. Intrusion attacks use protocol ports and "#" commands to obtain sensitive information. MQTT security relies on an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS), which act quickly upon detection, reducing reaction time and ensuring security. [31] [34] [35]

I.11 Conclusion

The Internet of Things (IoT) networks consist of groups of identifiable, smart devices that communicate and interact with each other. These networks possess several characteristics, including scalability, heterogeneity, and safety, among many others. The diverse architectures of IoT networks make them applicable in a wide range of fields, such as healthcare, agriculture, smart homes, smart cities, and beyond. However, one of the primary concerns associated with IoT is security. The open and interconnected nature of these networks makes them particularly vulnerable to cyber-attacks.

To address these challenges, various mechanisms have been developed to protect and defend IoT networks. These mechanisms are designed to ensure the integrity, confidentiality, and availability of the data transmitted within the network. One of the key components in IoT communication is the Message

Queuing Telemetry Transport (MQTT) protocol, which is known for being a lightweight communication protocol.

The MQTT protocol includes its own security mechanisms to safeguard the data exchanged between devices. These security features are crucial because IoT devices often operate with limited computational resources and power, making them susceptible to various types of cyber threats. The protocol employs measures such as authentication, encryption, and secure transmission to protect data. By ensuring that data is transmitted securely, MQTT helps to mitigate the risks of interception and unauthorized access, thereby maintaining the privacy and security of the information within the IoT ecosystem.

In healthcare, IoT networks enable remote patient monitoring, real-time health data analysis, and improved patient care. In agriculture, IoT devices monitor soil moisture, weather conditions, and crop health, leading to increased efficiency and yield. Smart homes benefit from IoT through automation of household tasks, energy management, and enhanced security systems. Smart cities leverage IoT for efficient traffic management, waste management, and improved public services.

Despite the robust security mechanisms provided by protocols like MQTT, the dynamic and evolving nature of cyber threats necessitates continuous advancements in IoT security. Researchers and developers are constantly working on innovative solutions to stay ahead of potential vulnerabilities. This ongoing effort is crucial for ensuring that IoT networks remain reliable and secure, thereby fostering trust and encouraging the widespread adoption of IoT technologies across various sectors.

Chapter II: Intrusion Detection

II.1 Introduction

System administrators had to manually monitor user behavior at first for intrusion detection, but this approach proved to be ad hoc and unscalable. Administrators started using audit logs as a post-event forensic technique to pinpoint security issues in the late 1970s and early 1980s. Software to evaluate this data was created as storage became more accessible. Unfortunately, this research required a lot of time and computing power, often necessitating the nighttime execution of intrusion detection systems. [37]

Real-time intrusion detection systems began to appear in the early 1990s, enabling quick response and assault prevention. System managers were now able to react to threats as they materialized rather than after the fact, which was a huge development. At the moment, the goal of intrusion detection activities is to develop solutions that can be implemented effectively in huge networks. New attack techniques, changing security issues, and the dynamic nature of computer systems are all taken into consideration in these efforts. [37]

Because we use the internet so much in our everyday lives, network security is now the cornerstone of all web services, including online retail purchases and auctions. The purpose of intrusion detection is to find computer assaults by looking through different information records that are seen during network operations. This is regarded as one of the best approaches to handle issues with network security. Data security may be jeopardized by an infiltration via a variety of internet channels. The need for more dependable, efficient, and self-monitoring systems that can function without human intervention has arisen from the quick expansion of networks, faster data transmission speeds, and unexpected internet consumption. It is possible to considerably lower the danger of catastrophic failures in susceptible systems by pursuing such developments. [37]

Systems for detecting intrusions are an essential part of computer network security. They serve as a deterrence as well as an early warning system. These systems may be set up to respond to traffic instantly, cutting out shady connections in accordance with predetermined standards. Many people believe that prevention is even more crucial than detection. It is important to speak with knowledgeable experts who can carry out a network audit in order to guarantee the maximum degree of safety for your networks. These professionals may provide guidance on the optimal defensive stance to take and suggest the finest software for safeguarding your network. [37]

It is impossible to overestimate the significance of strong intrusion detection systems in the linked world of today. Continuous innovation and development in intrusion detection technology are critical as cyber-attacks grow more complex. By doing this, networks are kept safe, data integrity is preserved, and user and stakeholder confidence is maintained. Organizations may establish a robust defense against the constantly changing cyber threat environment by allocating resources towards sophisticated intrusion detection systems and using expert skills.[37]

II.2 What is Intrusion Detection?

To put it simply, it's the persistent efforts to find or identify the existence of invasive activity. When it comes to computers and network infrastructure, intrusion detection (ID) has a far wider application. It encompasses all procedures needed to identify unauthorized usage of computer or network devices. This is accomplished by using software that has been specially created with the express intent of identifying anomalous or unusual activities.[36]

II.3 What Is an Intrusion-Detection System (IDS)?

An intrusion-detection system (IDS) is a tool used to identify, assess, and report unauthorized network activity. It is part of an overall protection system, similar to firewalls, closed doors, alarm systems, and guard dogs. In a warehouse, these technologies can cooperate to prevent network breaches. The implementation of IDSs depends on the location of technology. A network is only as safe as its weakest link, so a layered strategy is essential. A network should have several security layers, each with a distinct purpose, to support the organization's overall security plan. IDSs work at the network layer of the OSI model, while passive network sensors are positioned at choke points. IDSs analyze packets to find specific patterns in network traffic, logging alerts and responding based on recorded data. IDSs use known signatures to recognize potential malicious traffic patterns. [36]

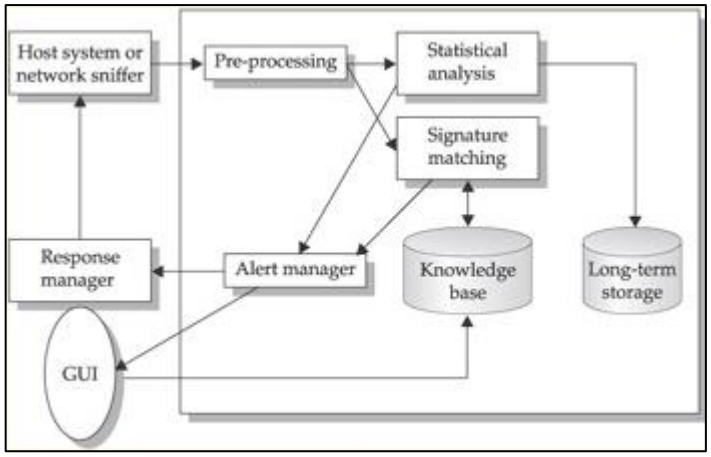


Figure II.1: Standard IDS system.

II.4 Types of IDS Systems

IDSs fall into one of three categories: [36] [38] [40] [42] host-based intrusion-detection system (HIDS), network-based intrusion-detection system (NIDS), and hybrids of the two.

II.4.1 HIDS (Host Intrusion Detection System)

Information gathered from inside a single computer system powers host-based intrusion detection systems. (Remember that host-based IDSs are really a subset of application-based IDSs.)

Because of this advantage, host-based intrusion detection systems (IDSs) can reliably and precisely identify the processes and users engaged in a given operating system assault. Furthermore, since host-based IDSs have direct access to and monitoring of the data files and system activities that are often the focus of assaults, they are able to "see" the results of an attempted attack, in contrast to network-based IDSs.

System logs and operating system audit trails are the two main information sources that host-based intrusion detection systems typically use. Operating system audit trails are more thorough and more secure than system logs since they are often produced at the lowest (kernel) level of the operating system. System logs, on the other hand, are significantly simpler, smaller, and easier to understand than audit trails. A single management console may follow several hosts with the use of a centralized IDS NIST Special Publication on Intrusion Detection Systems management and reporting architecture, which is supported by some host-based IDSs. There are others who produce messages in forms that work with network management systems.

HIDS system will require some software that resides on the system and can scan all host resources for activity; some just scan syslog and event logs for activity. It will log any activities it discovers to a secure database and check to see whether the events match any malicious event record listed in the knowledge base.

II.4.2 NIDS (Network Intrusion Detection System)

Network-based intrusion detection systems make up the bulk of commercial models. By collecting and analyzing network packets, these IDSs identify assaults. One network-based intrusion detection system (IDS) may safeguard several hosts linked to a network segment by monitoring the network traffic impacting those hosts by listening to the network segment or switch. Network-based intrusion detection systems typically include a collection of specialized sensors or hosts positioned across a network.

These devices keep an eye on network traffic, analyze it locally, and report any assaults to a central control panel. The sensors are more readily guarded against attack since their use is restricted to executing the IDS. To make it more difficult for an attacker to locate and detect them, many of these sensors are made to operate in "stealth" mode.

Detecting and categorizing all network traffic from all devices is possible with this kind of intrusion detection system (IDS), which may be employed as a security measure inside a network that is secured.

Unlike a HIDS, a NIDS can see every packet moving across its network, but it cannot confirm the integrity of the contents stored on the devices. It will also "log" any questionable packets.

NIDS system is usually inline on the network, and it analyzes network packets looking for attacks. A NIDS receives all packets on a particular network segment, including switched networks (where this is not the default behavior) via one of several methods, such as taps or port mirroring. It carefully reconstructs the streams of traffic to analyze them for patterns of malicious behavior. Most NIDSs are equipped with facilities to log their activities and report or alarm on questionable events. In addition, many high-performance routers offer NID capabilities.

Table II.1: Network-Based vs. Host-Based Intrusion-Detection Systems.

NIDS	**HIDS**
Broad in scope (watches all network activities)	Narrow in scope (watches only specific host activities)
Easier setup	More complex setup
Better for detecting attacks from the outside	Better for detecting attacks from the inside
Less expensive to implement	More expensive to implement
Detection is based on what can be recorded on the entire network	Detection is based on what any single host can record
Examines packet headers	Does not see packet headers
Near real-time response	Usually only responds after a suspicious log entry has been made
OS-independent	OS-specific
Detects network attacks as payload is analyzed	Detects local attacks before they hit the network
Detects unsuccessful attack attempts	Verifies success or failure of attacks

II.4.3 Hybrid IDS

The features of both NIDS and HIDS are combined in hybrid IDSs. They enable the network and terminals to be watched over. The strategically positioned probes serve as either HIDS or NDS, depending on where they are located. The alarms from all of these probes are then sent to a single system that unifies data from many sources. We now know that hybrid IDS are built on a distributed architecture in which all of the components use a common transmitting format. This facilitates communication and yields more precise notifications.

Combining two or more intrusion detection system methodologies results in a hybrid intrusion detection system. A comprehensive picture of the network system is created by the hybrid intrusion detection system by fusing network data with host agent or system data. Compared to the opposite intrusion detection system, the hybrid intrusion detection system is easier to use. One instance of a hybrid IDS is Prelude.

II.4.4 Protocol-based IDS (PIDS)

An intrusion detection system that is useful for monitoring and analyzing the protocol or protocols that the computer system uses is called a protocol-based intrusion detection system (PIDS). PIDs are usually placed on web servers.

A protocol intelligence and security system, or PIDS, is a system or agent that sits at the front end of a server and monitors and analyzes the communication protocol between a connected device (a user, PC, or system) and the system it is protecting. It also keeps track of the protocol's dynamic behavior and state.

This would normally be used by a web server to keep an eye on the HTTPS protocol stream and comprehend the HTTP protocol in relation to the web server or system that it is attempting to secure.

II.4.5 Application Protocol-based IDS (APIDS)

An intrusion detection system that concentrates its monitoring on a particular application protocol or protocols used by the computer system is known as an application protocol-based intrusion detection system (APIDS). One illustration of APIDS is Secerno.

The Secerno. When in IDS mode, a SQL database security appliance looks for odd movement and generates alarms; it does not really stop possible threats. Alerts can be tailored to particular SQL statement kinds that show how applications communicate with databases. Since the network traffic to the database is duplicated using conventional networking techniques, there is no effect on database traffic.

II.5 Characteristics of IDS

II.5.1 Accuracy

It shows how closely the IDS results coincide with the typical operation of the system under observation. The IDS needs to understand how the system works and distinguish it from invasive activity. A low false positive rate can be used to communicate this trait. [32] [41]

II.5.2 Response Time

This is the maximum speed at which events can be processed in order to minimize latency and enable real-time detection. Additionally, the IDS needs to be able to promptly notify the system administrator of the detection result and/or initiate countermeasures. [32] [41]

II.5.3 Completeness of Detection

All known and undiscovered attacks should be picked up by an ideal IDS. Due to incomplete awareness of the attacks, evaluating this measure is exceedingly challenging. [32] [41]

II.5.4 Fault Tolerance

To stop attempts to circumvent the intrusion detection system, the intrusion detection system itself needs to be resistant to attacks. [32] [41]

II.6 Intrusion Detection Operating Modes

II.6.1 Anomaly Detection

Anomaly-based detection is a technique used by intrusion detection systems (IDS) to identify unusual network traffic, such as malformed IP packets. It uses profiles created by tracking regular activity over time and compares current actions with profile-related thresholds. This method can identify unknown threats, but can create simplified profiles, include harmful activity, and produce false positives. It can identify unidentified attacks and does not always result in aggression.[39]

II.6.2 Signature-based Detection

Signature-based detection is a quick and simple method for evaluating harmful traffic in intrusion detection systems. It relies on known traffic data and is precise but limited in identifying variants of known threats or unknown threats. It cannot monitor complex communications and is not effective for identifying attacks with numerous events. Other methods include using attack signatures, system call sequences, and network traffic patterns.[38] [42]

II.6.3 Specification-based Detection

Specifications are guidelines that specify acceptable behavior models for network elements like routing tables and protocols. They can be statistical rules or manually constructed models. Similar to anomaly-based detection, specifications are set manually by a human expert, allowing for low false positive rates and identifying unidentified attacks. [42]

II.6.4 Behavior after Detection

Two actions can be taken in the event that the IDS detects an attack: an active response or a passive response. This feature is frequently connected to the IDS responses module. [41]

- **Passive response:** In this instance, the IDS's response is restricted to sending the administrator or an archiving system (log files) a warning identifying the attack. The human operator will handle the countermeasures in both scenarios.
- **Active response:** In contrast to the first scenario, automatic defenses will be triggered to stop the attack and restrict its path. Blocking incoming IP addresses or ports, for instance, ends a session or shuts down a computer.

II.6.5 Frequency of Use

This feature is dependent on the IDS analysis module's operational mode. [41]

- **Real-time continuous analysis:** The IDS continuously examines the information flow. When network intrusion detection systems are in this mode, network traffic is examined right away following capture. Any harmful activity found can be immediately dealt with thanks to the ongoing analysis. When the IDS's processing speed surpasses the network's transfer

speed, this mode becomes functional. Real-time analysis is not possible in any other case.

- **Batch Analysis (delayed):** If the IDS processing speed is slower than the dynamics of change in the system being monitored, there are situations where it is better to make detections in a postponed time frame. A Network Intrusion Detection System (NIDS) operating at 100Mbps may be compelled to save traffic and perform analysis in deferred mode if the network is operating at 1Gbps. Similarly, if a HIDS examines system audit logs that are updated on a regular basis, it must do so in accordance with the updates' interval.

II.6.6 Target Monitoring

Systems that monitor targets will report on any alterations or modifications made to certain target items. Typically, a cryptographic procedure is used to accomplish this, computing a crypto-checksum for every target file. Any changes that could affect crypto-checksums, like file alterations or program logons, are reported by the IDS. Through the use of crypto-checksums, Tripwire software will perform target monitoring by instantly notifying users of modifications to configuration files and enabling automatic restoration. This method's primary benefit is that it spares you from having to keep an eye on the target files all the time.[36]

II.6.7 Stealth Probes

Stealth probes use data correlation to look for long-duration attacks, sometimes known as "low and slow" attacks. To find any associated attacks, data is gathered from many sources, characterized, and sampled. Wide-area correlation is another name for this technology, which usually employs a hybrid or combination approach combining various detection approaches in an attempt to identify potentially harmful activities.[36]

II.7 IDS Pros and Cons

The pros of intrusion detection include the following [36]:

- Can detect external hackers as well as internal network-based attacks.
- Scales easily to provide protection for the entire network.
- Offers centralized management for correlation of distributed attacks.
- Provides defense in depth.
- Gives system administrators the ability to quantify attacks.
- Provides an additional layer of protection.

These are the cons [36]:

- Generates false positives and negatives.
- Reacts to attacks rather than preventing them.
- Requires full-time monitoring.
- Requires a complex incident-response process.
- Cannot monitor traffic at higher transmission rates.

- Generates an enormous amount of data to be analyzed.
- Requires highly skilled staff dedicated to interpreting the data.
- Susceptible to "low and slow" attacks.
- Cannot deal with encrypted network traffic.
- It is expensive.

II.8 IDS Architecture

II.8.1 Single-Tiered Architecture

A single-tiered architecture is a type of IDS where components collect and process data themselves, rather than passing it to other components. This architecture offers advantages like simplicity, low cost, and independence from other components. However, it often has components that are not aware of each other, reducing efficiency and functionality. [36]

II.8.2 Multi-Tiered Architecture

A multi-tiered architecture is a system that consists of multiple components that pass information to each other. It is commonly used in intrusion detection systems (IDSs) and includes sensors, analyzers, and a manager. Sensors collect data from various sources, while analyzers monitor intrusive activity on individual hosts. Agents are specialized to perform specific functions, such as examining TCP traffic or FTP connections. When an attack is detected, they send information to the manager component, which performs various functions, such as collecting alerts, triggering a pager, storing information, retrieving relevant information, sending commands, and providing a management console. A central collection point allows for easier analysis of logs, and management consoles enable remote policy changes and parameter erasure. Advantages of a multi-tiered architecture include greater efficiency and depth of analysis, but downsides include increased cost and complexity. [36]

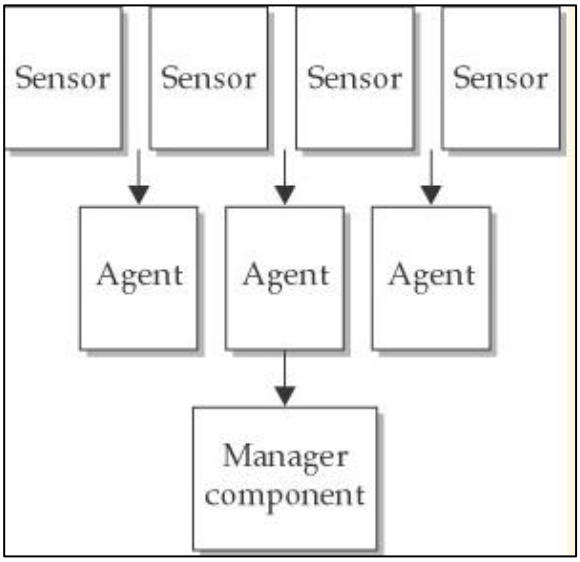


Figure II.2: A multi-tiered architecture.

II.8.3 Peer-to-Peer Architecture

Peer-to-peer architecture is a type of network architecture where information is exchanged between peer components, allowing for intrusion-detection. It is commonly used by cooperating firewalls and routers. This architecture is simple and allows any peer to participate in a group of peer machines, benefiting from each other's information. However, it lacks sophisticated functionality due to the absence of specialized components.

II.9 Intrusion Attacks

II.9.1 System Scanning

When an attacker sends various types of packets to the target network, system scanning may occur. System vulnerabilities and characteristics can be found based on the target's response. These are passive attacks that don't breach or compromise systems. A few tools that can be used for scanning attacks are vulnerability scanners, port scanners, network scanners, port mappers, and port scanners. Various system characteristics that this assault may display include [42] [43]:

- Target topology of the networks.
- The quantity of running hosts on the network.
- Software Version numbers of the server running on the network.
- The host's operating system is now in use.

II.9.2 Denial of Service

DoS attacks happen frequently. They make an effort to impede or stop targeted networks or systems. There are various reasons behind these attacks. DoS attacks caused significant losses for e-commerce businesses because many users were unable to access them at the time of purchase. Dos attacks may result in a number of issues, including unavailable or ineffective services and disruptions in network traffic at the connection interface. The following indicators point to the presence of DoS attacks:

- Performance of the network is unusually slow.
- The particular site is not available.
- An extension of the access time.

II.9.3 Flow Exploitation DoS Attacks

Another name for it is the "Ping of Death" attack. It mostly takes advantage of software bugs in the target system that lead to processing errors or resource exhaustion. This kind of attack sends the target system a lot of ping packets. The system crashes because the target system is unable to handle these unusual packets. CPU time, memory, storage space, space in a dedicated buffer, and network bandwidth are among the several resources that are targeted. Various DoS attack techniques include depleting IDS resources. It would keep sending out alerts and overloading IDS with traffic until it ran out of resources. Consequently, an incomplete event log would be generated.

II.9.4 Flooding DoS Attack

The target receives more information than it can process. When the target system is being attacked, it cannot be patched. A variety of modification strategies can be applied to lessen these kinds of attacks. DDOS attacks, or denial-of-service attacks, are launched by many people. They function as a single, enormous system and are centralized. Therefore, the quickest system can be used to stop it.

II.9.5 System Penetration

System penetration is the unauthorized acquisition of resources, data, or rights within the system. Different software bugs are taken advantage of to take over a system. Their specifics and effects differ. Penetration attacks involve any unauthorized access to or changes to the system's data and resources in order to take advantage of weaknesses in the system. Attackers use a variety of software vulnerabilities to take over the machine in these types of attacks. With the Internet of Things, an attacker can take over a device physically or through an application, giving him the ability to reverse engineer and check for vulnerabilities. Various forms of system intrusion are:

- ***User to Root:*** Target host, completely controlled by local user.
- ***Remote to User:*** An account of target host, managed by the attacker on the network.
- ***Remote to Root:*** Target host, completely controlled by the attacker on the network.
- ***Remote Disk Read:*** An ability to read private data files on target host without authorization of owner by an attacker on network.
- ***Remote Disk write:*** An ability to write private data files on target host without authorization of owner by an attacker on network.

II.9.6 Man-in-the-Middle (MiTM) Attacks

The MiTM attack is another type of assault in which the attacker actively intercepts two nodes' communications without the victims' knowledge. The messages between the nodes are intercepted by the attacker, who may then alter them. Furthermore, as of late, attacking machines are typically a component of a larger network of hacked workstations, or a botnet. The goal of integrity attacks is to change the data or route within the network.

II.9.7 Routing Attacks

The information (messages) exchanged within the framework of the routing protocol is altered or spoof by the attacker in a routing attack. Numerous Internet of Things routing hacks target the RPL protocol, which is a key protocol for Internet-integrated wireless sensor networks.[31]

II.9.8 Application–level Attack

The attacker focuses on the application layer's limitations. For instance, a web server's security flaws or improper server-side controls.

II.9.9 Viruses and Worms

Computer viruses and worms are harmful programs designed to replicate themselves, similar to biological reproduction. They can be classified as worms or viruses based on whether the malicious code requires human intervention to spread to another system. Some viruses/worms have multiple infection mechanisms, such as searching for and emailing to infected email addresses, scanning for unprotected network shares, infecting vulnerable servers, and infecting local and network-accessible files. Pure viruses, like "I Love You" and "SoBig," propagate through email attachments, while pure worms like "Code Red," "Slammer," and "Blaster" actively scan for and infect further vulnerable systems. Currently, worms/viruses have mild actions, such as installing back-door software, installing email engines, defacing websites, conducting distributed denial of service attacks, and logging internet bandwidth. Future threats include data corruption, hardware damage, espionage, and personal information theft.[36]

II.10 Security Mechanisms

One of the key factors in evaluating the system's reliability is security concerns. In the event that these issues are effectively resolved, the system's added value and reliability both rise significantly. If not, reliability is undervalued, compromised, and the system becomes unusable, resulting in a lack of added value. As a result, system security is now a top priority for administrators. They have access to a variety of security tools and techniques, including [41]:

- Data integrity and secrecy guaranteed by encryption techniques.
- Firewalls for network traffic filtering and access control.
- Vulnerability scanners to find system security holes.
- Antivirus software to guard the system from dangerous apps

II.11 Conclusion

Intrusion detection is as crucial to a network system as a burglar alarm is to buildings or houses where valuable information or items are stored. Just as a burglar alarm alerts homeowners to potential break-ins and unauthorized access, an Intrusion Detection System (IDS) monitors a network for any suspicious activity or policy violations. This setup not only detects threats but also actively takes steps to prevent them from causing harm, thereby significantly enhancing the overall security and effectiveness of the system.

A high-quality IDS, offers more than just notifications about potential threats. It can automatically take actions such as blocking malicious traffic, alerting administrators, and logging critical information for further analysis. This proactive approach means that threats are managed in real-time, reducing the risk of data breaches and other security incidents.

IDS technology can be categorized into two main types: Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS).

NIDS monitor network traffic for suspicious activity, providing a broad overview of the entire network's security. In contrast, HIDS focuses on individual devices, monitoring activities such as file modifications, logins, and other critical operations. Some systems combine both NIDS and HIDS to provide comprehensive coverage and enhanced protection.

Implementing an IDS typically involves installing software or deploying hardware sensors across the network or on individual devices. This setup can be tailored to fit the specific needs and infrastructure of an organization. For businesses that do not currently have an IDS in place, it is highly advisable to consider integrating one into their security model or infrastructure. The absence of an IDS leaves a network vulnerable to undetected threats, potentially leading to significant financial and reputational damage.

In conclusion, the proactive defense mechanism provided by IDS, is essential for safeguarding against unauthorized access and potential breaches. By continuously monitoring and responding to threats, an IDS helps maintain the integrity and security of the network, ensuring that valuable information and resources remain protected. For any organization looking to strengthen its security posture, investing in an effective IDS should be a top priority.

Chapter III : Deep Learning

III.1 Introduction

The human brain is the most amazing organ in the body. It determines how we interpret everything that we see, hear, taste, smell, and touch. It allows us to dream, feel emotions, and store memories. Without it, humans would be rudimentary creatures with only the most basic reactions. Our brains are fundamentally what give us intelligence. Even though the newborn brain is barely one pound in weight, it manages to solve puzzles that are above the capabilities of our largest, most potent supercomputers. A few months after birth, newborns are able to distinguish distinct items from their surroundings, recognize their parents' faces, and even distinguish between different voices. In little than a year, they've already acquired an intuitive understanding of natural physics, the ability to track things even when they're partially or totally obscured, and the ability to interpret sounds.

Additionally, by the time they are young children, they have hundreds of words in their vocabularies and a good grasp of syntax. We have long dreamed of creating intelligent machines with minds similar to our own: self-driving vehicles, robotic housecleaning assistants, and disease-detection microscopes. However, in order to create these artificially intelligent robots, we must find solutions to some of the trickiest computing problems we have ever encountered—issues that our brains are currently capable of handling in a matter of microseconds. In order to solve these issues, we will need to create a whole new method of computer programming utilizing methods that have mostly been established in the last ten years. *Deep learning* is a term used to describe this very active area of artificial computer intelligence. During the past few years, deep learning has revolutionized nearly every field it has been applied to, resulting in the greatest leap in performance in the history of computer science. The application of deep learning has made those small, gradual annual improvements a thing of the past — these days, it isn't uncommon to witness improvements of 20 to 30 percent, in months and not years. There's no keeping that kind of success under wraps, which means the media have been filled with references to "artificial intelligence," "machine learning," and "deep learning." These terms are used not only very widely, but most of the time inaccurately and confusingly. With that in mind, this chapter aims to clarify and demystify the distinctions among these technical terms.[45]

III.2 The Story Begins with Artificial Intelligence

John McCarthy, a trailblazing computer scientist, first used the phrase artificial intelligence (AI) in the 1950s. It's a catch-all word for all the techniques and fields that lead to machines displaying intelligence of any kind. This ranges from the expert systems of the 1980s, which were essentially databases of knowledge that had been hardcoded, to the most sophisticated AI systems that are currently in use. Nowadays, almost every software used in almost every industry uses artificial intelligence (AI), even if it's only applied to a few simple manually coded processes. [44]

III.2.1 What Is Machine Learning?

Artificial intelligence's machine learning field studies a machine's capacity to mimic thoughtful human behavior. The creation of algorithms that facilitate

learning from previously collected data is the focus of machine learning. The term "machine learning" was first used in 1959 by Arthur Samuel, who defined it as "the ability for a machine to automatically learn from data, improve performance through experience, and predict things without being explicitly programmed." The process begins with supplying high-quality data, which is then utilized to train our machines by constructing machine learning models based on the data and different techniques. The selection of algorithms is contingent upon the nature of the data at hand and the nature of the work that has to be automated. [44]

III.2.2 Advancing into Deep Learning

Deep learning, also known as deep neural networks, is a subfield of machine learning, which is a subset of AI, as shown in Figure III-1. Deep learning takes inspiration from how the human brain works. What's the difference between deep learning and traditional machine learning? Perhaps the biggest distinction is that deep learning is the first — and currently the only — learning method that is capable of training directly on the raw data. No need for feature extraction with deep learning. In the example of facial recognition, deep learning would be able to dive in and examine the raw pixels of an image, without explicitly being told to pay attention to facial proportions or distance between pupils or other specifics called out by human experts. What's more, deep learning scales well to hundreds of millions of training samples. As the training dataset gets larger and larger, deep learning continuously improves. [44]

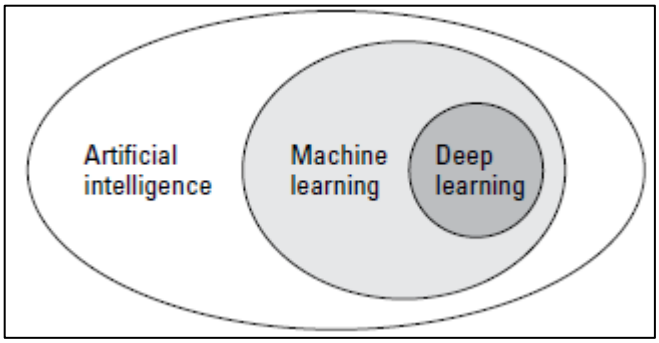


Figure III.1: Deep learning, a subset of a subset of AI.

III.3 Traditional Machine Learning

Artificial intelligence is evolving, with machine learning becoming a more advanced form of AI that allows computers to learn independently. Deep learning is a specific type of machine learning, and understanding it requires a solid understanding of its basic principles. This chapter covers general principles, data training, feature extraction techniques, and training data.[44]

III.3.1 Assembling the Training Data

Machine learning models require data samples, which are essential for their success. For example, a "dog detector" uses a large dataset of images categorized into "dog" and "not dog" classes. Supervised training uses a fully labeled dataset, while unsupervised training uses data without labels. Supervised training typically yields

better results, while unsupervised learning has untapped potential due to the vast amount of unlabeled data available. Both methods are essential for effective machine learning.[44]

III.3.2 Understanding the Importance of Feature Extraction

In traditional machine learning, raw images with or without dogs are used to create labels. However, the machine is aware of these pixels and needs to perform a feature extraction phase to extract predefined properties or features. In the dog detector example, each input sample is represented as a vector of values, each corresponding to a single feature. To identify important features, a domain expert is needed to specify them. For image processing problems, an expert analyzes the problem domain and samples, determining the features to extract. In real-world examples, feature extraction is based on properties of files, such as API or function calls or registry keys used. This process is essential for training models in machine learning.[44]

III.3.3 Learning Algorithms

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? It means by definition "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [45]

III.3.3.a The Task, T

Machine learning is a method that helps solve complex tasks that are too complex for human-designed programs. It involves understanding the principles that underlie intelligence and aims to improve our understanding of tasks. Common machine learning tasks include classification, regression, transcription, machine translation, structured output, anomaly detection, synthesis and sampling, imputation of missing values, denoising, and density estimation.

Classification tasks involve assigning an input to a category, while regression tasks involve predicting numerical values based on input. Transcription tasks convert unstructured data into textual form, while translation tasks convert symbols in one language into another. Structured output tasks involve generating vectors or data structures with important relationships between elements, while anomaly detection involves sifting through events or objects to flag unusual or atypical ones. Synthesis and sampling tasks use machine learning algorithms to generate similar examples from training data, making them useful for media applications like video games. [45]

Denoising tasks involve predicting the conditional probability distribution of a corrupted example from its corrupted version. Density estimation tasks involve learning a probability density or probability mass function on a space, which requires understanding the data structure and cluster examples. However, density

estimation may not always solve all related tasks due to computational intractable operations on the distribution. [45]

III.3.3.b The Performance Measure, P

To evaluate a machine learning algorithm's capability, a quantitative measure of its performance is designed. This measure is specific to the task being performed, such as classification, classification with missing inputs, and transcription. Accuracy is measured by the proportion of examples where the model produces the correct output, while error rate is the proportion of examples where the model produces an incorrect output. For tasks like density estimation, a different performance metric is used, such as the average log-probability assigned to some examples. The choice of performance measure can be challenging due to the complexity of deciding what to measure, or the impracticality of computing the actual probability value assigned to a specific point in space. In such cases, alternative criterion or approximation to the desired criterion is needed. [45]

III.3.3.c The Experience, E

Machine learning algorithms can be supervised or unsupervised based on their learning experience. The Iris dataset, one of the oldest studied, is a collection of measurements of 150 iris plants, each representing a different part of the plant. Unsupervised learning algorithms observe random vectors and attempt to learn the probability distribution p(x), while supervised learning involves observing multiple examples of a random vector and an associated value or vector and learning to predict y from x. Other variants of the learning paradigm include semi-supervised learning, multi-instance learning, and reinforcement learning. Most machine learning algorithms experience a dataset, which can be described in various ways, such as a design matrix. There is no formal definition of supervised and unsupervised learning, but new ones can be designed for new applications. [45]

III.3.4 Training and testing

Machine learning researchers often make mistakes due to contamination between test and train sets, which can lead to skewd results. Contamination can be subtle and can mess up the entire process. For example, if a machine learning model is trained on a dataset of images containing tanks and trees, it may inadvertently learn to detect clouds instead of tanks. This can lead to biased results. Similarly, if a model is trained on malicious files and benign files, it may not accurately classify malicious or benign files. To remedy this, the benign dataset should contain many different files created by different developers, not just Microsoft. It is crucial to ensure that test data is completely separated from train data and that the data is representative of the type and distribution of data encountered in the real world.[44]

III.3.5 Setting aside a validation set

Incorporating test data insights into model training is crucial for effective learning in the real world. Instead of using a test set for training, a validation set is used to measure performance on new data and use the insights for further training and

improvements. The test set remains the ultimate test, replicating real-world conditions. To ensure reliability, measures should be stringent, such as using data from different time periods for training and testing, as new malicious file types appear daily in the real world. [44]

III.4 The Neural Network

III.4.1 The Biological Brain Was the First Real Neural Network

The human brain consists of tens of billions of small processing units known as neurons. These neurons are connected to each other via synapses. You've probably read that the human brain has different regions — such as the visual cortex and auditory cortex — that each perform a certain task. These differences mainly arise from the input each region receives. For example, when the optic nerve transfers signals (the input) from our eyes to a certain region in the brain (the processing area), the neurons in that area learn to process these signals, and form the visual cortex. [44] [48]

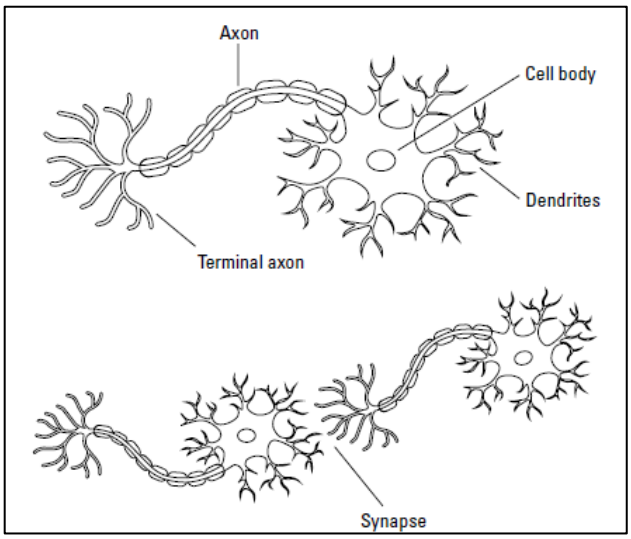


Figure III.2: Making connections in the brain.

We can refer to the neurons as general processing units, which are agnostic of the data they process. The learning process itself takes place when the connection strength between neurons is formed, removed, strengthened, or weakened. In other words, everything humans learn, everything we remember, everything we do, is the result of synaptic activity in the brain. You might consider the cerebral cortex to be the most "interesting" part of our brain, because it's associated with our high-level cognitive capabilities. Mammals are the only animals that have a cerebral cortex.
Why is it that humans are smarter than all other animals? Brazilian neuroscientist Suzana Herculano-Houzel invented a novel method for accurately counting the number of neurons in the brain. Her research suggests that intelligence is correlated with the number of neurons in the cerebral cortex. The higher this number, the

higher the intelligence. An elephant has a brain with a much larger mass, but the human brain's cerebral cortex has a far greater absolute number of neurons. [44]

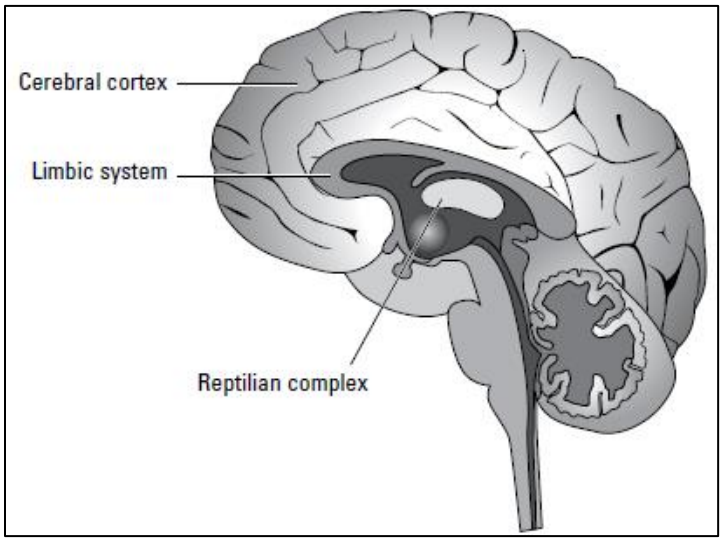
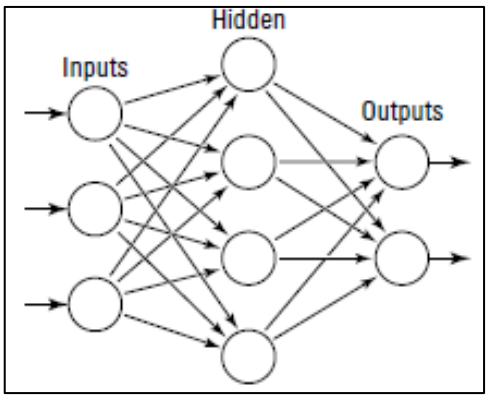


Figure III.3: A few parts of the brain.

III.4.2 Artificial Neural Networks

Artificial neural networks have their origins in 1943 when researchers Warren McCulloch and Walter Pitts proposed a simple model for an artificial neuron. Frank Rosenblatt later created the perceptron, a simple neural network with two layers: the input and output layers. These networks were limited in their learning capabilities. In the late 1960s, researchers discovered that they could expand the capabilities of neural networks by adding hidden layers, creating multilayered neural networks or multilayered perceptron's (MLP). However, these networks could not be trained using conventional mechanisms. In the early 1980s, Paul Werbos and David Rumelhart invented backpropagation, which is still used for training multilayered neural networks today. [44] [48]



. **Figure III.5:**
. Multilayered perceptron.

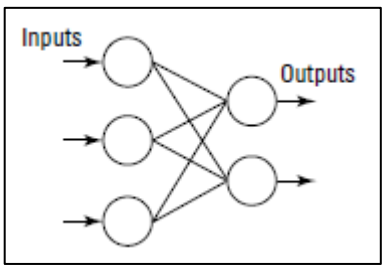


Figure III.4: Connecting neurons in a perceptron neural network.

III.4.3 Training a Neural Network with Backpropagation

Backpropagation is a fundamental principle in neural networks, which helps in recognizing and updating the weights of a neural network. In a training dataset of 10,000 images containing cats and 10,000 images without cats, a neural network is used. The input layer contains 900 neurons, the output layer contains two neurons representing the "no cat" and "cat" classes, and two hidden layers. The weights are initialized randomly and are usually small values around zero.

The neural network learns how to recognize a cat through training. At each point, a training sample is fed into the network, and the training process is done in two stages: feed-forward and backpropagation. The input layer sends values to the next layer, which aggregates the input and passes it through an activation function. The output neurons then fire their results, and the backpropagation algorithm updates the weights of the neural network to improve performance.

The network trains through many iterations over the entire training set, with each pass referred to as an epoch. Gradual updates to the weights are made during the backpropagation phase. After training, the accuracy is tested using a set of samples that were not used during the training. This prevents overfitting and encourages the network to generalize. If the results on the test set are satisfactory, the neural network can be used for real-world prediction. [44]

III.4.4 Feed-Forward Neural Networks

The human brain is composed of multiple neurons, which are organized in layers, such as the human cerebral cortex, which is responsible for most of human intelligence. Information flows from one layer to another until sensory input is converted into conceptual understanding. Artificial neural networks can be constructed by connecting neurons to each other, input data, and output nodes. The bottom layer of the network pulls in input data, while the top layer computes the final answer. The middle layer(s) are called hidden layers, and the parameter vector, θ, is determined by the weights of connections between neurons. These feed-forward networks are the simplest to analyze and are essential for solving complex learning problems. [46]

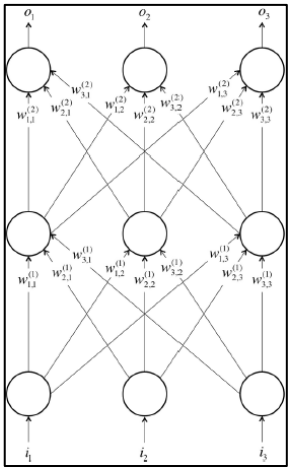


Figure III.6: A simple example of a feed-forward neural network.

III.4.5 Linear Neurons and Their Limitations

Most neuron types are defined by the function f they apply to their logit z. Let's first consider layers of neurons that use a linear function in the form of $f(z) = az + b$. For example, a neuron that attempts to estimate a cost of a meal in a fast-food restaurant would use a linear neuron where a = 1 and b = 0. In other words, using $f(z) = z$ and weights equal to the price of each item, the linear neuron in Figure III-7 would take in some ordered triple of servings of burgers, fries, and sodas and output the price of the combination. Linear neurons are easy to compute with, but they run into serious limitations. In fact, it can be shown that any feed-forward neural network consisting of only linear neurons can be expressed as a network with no hidden layers. This is problematic because, as we discussed before, hidden layers are what enable us to learn important features from the input data. In other words, in order to learn complex relationships, we need to use neurons that employ some sort of nonlinearity. [46]

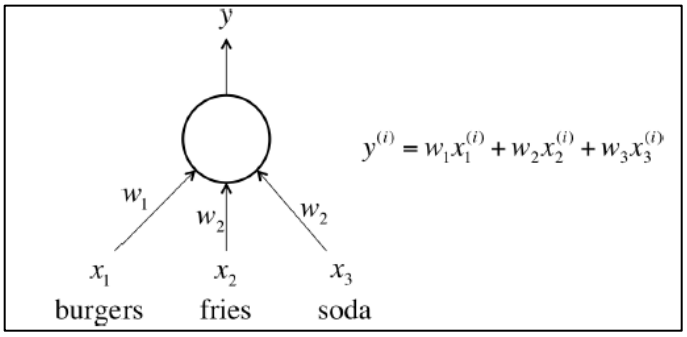


Figure III.7: An example of a linear neuron.

III.4.6 Sigmoid, Tanh, and ReLU Neurons

There are three major types of neurons that are used in practice that introduce nonlinearities in their computations. [46] [47] The first of these is the *sigmoid neuron*, which uses the function:

$$f = \frac{1}{1+e^{-z}} \qquad \textbf{(I.1)}$$

Intuitively, this means that when the logit is very small, the output of a logistic neuron is very close to 0. When the logit is very large, the output of the logistic neuron is close to 1. In-between these two extremes, the neuron assumes an S-shape.

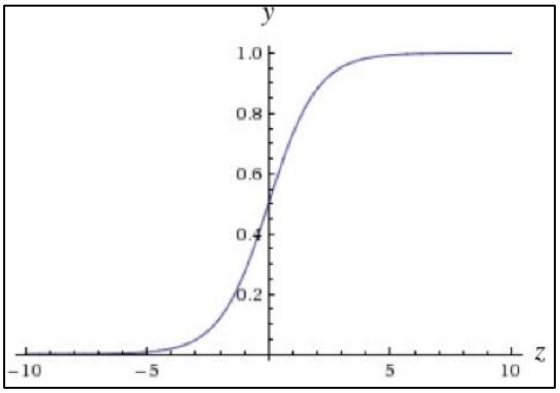


Figure III.8: The output of a sigmoid neuron as z varies.

Tanh neurons use a similar kind of S-shaped nonlinearity, but instead of ranging from 0 to 1, the output of tanh neurons range from –1 to 1. As one would expect, they use f (z) = tanh (z). The resulting relationship between the output y and the logit z is described by Figure III-9. When S-shaped nonlinearities are used, the tanh neuron is often preferred over the sigmoid neuron because it is zero-centered.

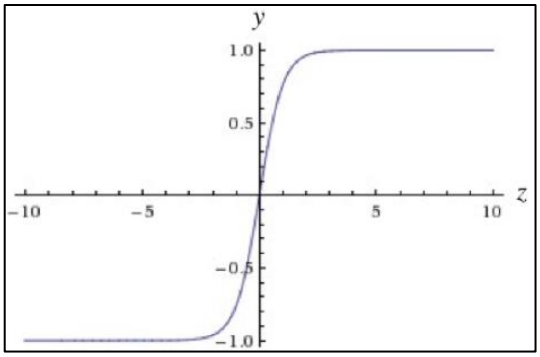


Figure III.9: The output of a tanh neuron as z varies.

A different kind of nonlinearity is used by the *restricted linear unit (ReLU) neuron*. It uses the function $f\ (z) = \ max\ (0, z)$. resulting in a characteristic hockey-stick-shaped response, as shown in Figure III-10.

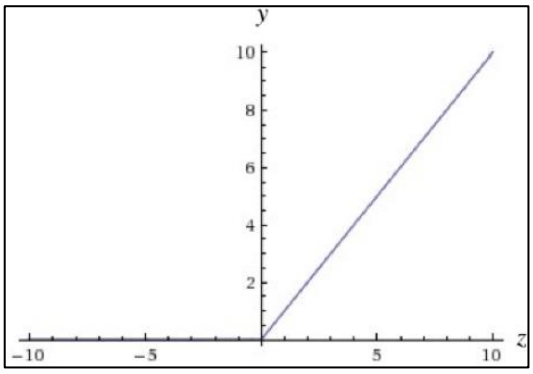


Figure III.10: The output of a ReLU neuron as z varies.

III.4.7 Softmax Output Layers

Oftentimes, we want our output vector to be a probability distribution over a set of mutually exclusive labels. For example, let's say we want to build a neural network to recognize handwritten digits from the MNIST dataset. Each label (0 through 9) is mutually exclusive, but it's unlikely that we will be able to recognize digits with 100% confidence. Using a probability distribution gives us a better idea of how confident we are in our predictions. As a result, the desired output vector is of the form below,

where $\sum_i^9 0p_{i=1} : [p_0 p_1 p_2 p_3 \cdots p_9]$

This is achieved by using a special output layer called a softmax layer. Unlike in other kinds of layers, the output of a neuron in a softmax layer depends on the outputs of all the other neurons in its layer. This is because we require the sum of all the outputs to be equal to 1. Letting z_i be the logit of the i^{th} softmax neuron, we can achieve this normalization by setting its output to:

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad \textbf{(I.2)}$$

A strong prediction would have a single entry in the vector close to 1, while the remaining entries were close to 0. A weak prediction would have multiple possible labels that are more or less equally likely. [46]

III.5 Types of Neural Networks

The neural network spotlighted in the previous section was a simple one. In practice, there are many types of neural networks, used for different tasks. Following are some examples [44]:

III.5.1 Fully connected neural network

This is the simplest form of neural network, in which all the neurons in each layer are connected to all the neurons in the subsequent layer. look at Figure III-11 for a sense of how this plays out. Fully connected networks are popular because they are robust, and because they don't assume anything about the properties of the input. Also note that because all the neurons in each layer are connected to all the neurons in the subsequent layer, the actual position of a neuron within a layer really doesn't matter.

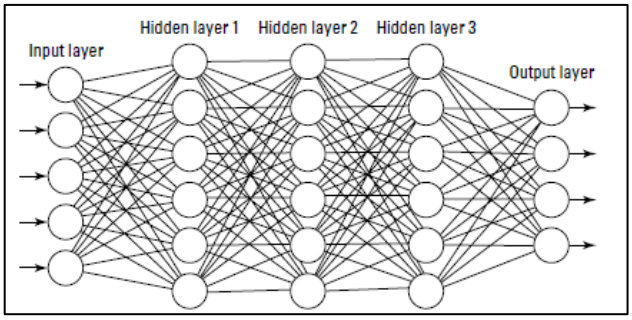


Figure III.11: A fully connected neural network

III.5.2 Recurrent neural network

Neural networks use current inputs for decision-making, particularly in sequential tasks like language understanding. Recurrent neural networks (RNNs) provide an indefinite memory of previous events by adding recurrent connections between neurons in hidden layers. These connections provide weights between neurons in the same layer, providing values in previous time steps. RNNs are useful for presenting sequential data and learning long-term patterns and relationships, with more advanced variants allowing higher accuracy over time.

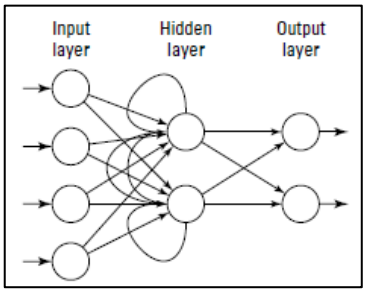


Figure III.12: A recurrent neural network.

III.5.3 Sparsely connected neural network

Sparsely connected neural nets are networks that are not fully connected, with only a portion of neurons between adjacent layers connected. These connections are determined by data properties. One popular variant is the convolutional neural network (CNN), used for computer vision problems. CNNs use a small receptive field, ensuring high correlation between adjacent pixels in real-world images.

III.6 Training deeper neural networks

Deep neural networks, which have a larger number of layers, have been trained using the backpropagation algorithm since the 1990s. However, the gradient vanishing problem has made it difficult to train these networks due to weaker signals. Recent inventions have addressed this issue, allowing researchers to train deeper neural networks with tens of layers and billions of synapses. Deep neural networks allow for a hierarchical pattern learning structure, allowing higher layers to learn and recognize more complex patterns. They also don't require traditional feature extraction, as they can use their deep layers as feature extractors, extracting complex patterns that human experts cannot manually specify. [44]

III.7 Deep Learning Algorithms

Deep learning is a machine learning and artificial intelligence method designed to mimic human brain functions for effective decision-making. It is a crucial data science element that uses predictive modeling and statistics. Deep learning algorithms run through layers of neural networks, pre-trained to serve a task. However, traditional machine learning algorithms struggle to handle structured or unstructured data sets, making deep learning an ideal solution. [49]

III.7.1 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can learn from an input image, assign importance to various aspects, and differentiate between them. It requires less pre-processing than other classification algorithms and can learn filters with training. CNNs have an architecture similar to the connectivity pattern of neurons in the human brain, inspired by the Visual Cortex. They have three main layers: convolutional, pooling, and fully-connected. As the layers increase, the CNN identifies larger elements or shapes until it identifies the intended object. [49]

III.7.2 Long Short-Term Memory Networks (LSTMs)

LSTMs are long-term learning and adaptation neural networks that can remember and recall past data. They are used in time series predictions due to their ability to restrain memory or previous inputs. LSTMs have a chain-like structure consisting of four interacting layers, and can be used in speech recognition, pharmaceutical development, and music loop composition. [49]

III.7.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks or RNNs consist of some directed connections that form a cycle that allow the input provided from the LSTMs to be used as input in the current phase of RNNs. These inputs are deeply embedded as inputs and enforce the memorization ability of LSTMs lets these inputs get absorbed for a period in the internal memory. RNNs are therefore dependent on the inputs that are preserved by LSTMs and work under the synchronization phenomenon of LSTMs. RNNs are mostly used in captioning the image, time series analysis, recognizing handwritten data, and translating data to machines. [49]

III.7.4 Generative Adversarial Networks (GANs)

GANs are deep learning algorithms that generate new instances of data that match training data. They consist of a generator that generates false data and a discriminator that adapts. GANs are used in astronomy, video games, cartoons, human faces, and 3D object rendering. They generate fake data and respond to it as false data, updating the results. [49]

III.7.5 Radial Basis Function Networks (RBFNs)

RBFNs are neural networks used for time-series prediction, regression testing, and classification. They consist of three layers: input, hidden, and output. The input layer uses neurons sensitive to training data, while the hidden layer integrates with the input layer. The output layer uses linear combinations of radial-based data, passing Gaussian functions as parameters. These networks are used for classification, regression testing, and time-series prediction. [49]

III.7.6 Multilayer Perceptron's (MLPs)

MLPs are the base of deep learning technology. It belongs to a class of feed-forward neural networks having various layers of perceptron's. These perceptron's have various activation functions in them. MLPs also have connected input and output layers and their number is the same. Also, there's a layer that remains hidden amidst these two layers. MLPs are mostly used to build image and speech recognition systems or some other types of the translation software.

The working of MLPs starts by feeding the data in the input layer. The neurons present in the layer form a graph to establish a connection that passes in one direction. The weight of this input data is found to exist between the hidden layer and the input layer. MLPs use activation functions to determine which nodes are ready to fire. These activation functions include tanh function, sigmoid and ReLUs. MLPs are mainly used to train the models to understand what kind of co-relation the layers are serving to achieve the desired output from the given data set. See the below image to understand better. [49]

III.7.7 Self Organizing Maps (SOMs)

Teuvo Kohenen invented Self-Organizing Machines (SOMs) to visualize data through artificial neural networks. These machines initialize weights of nodes and

choose random vectors from training data. They examine each node to find relative weights, deciding the Best Matching Unit (BMU). SOMs discover winning nodes over time, reducing them from the sample vector. Multiple iterations are used to ensure no node is missed. Examples include RGB color combinations. [49]

III.7.8 Deep Belief Networks (DBNs)

DBNs, also known as generative models, are used in video and image recognition and motion capture. They are powered by Greedy algorithms and use a layer-to-layer approach, generating weights through a top-down approach. They learn from latent values from every layer using a bottom-up pass approach, drawing samples from visible units and learning from the hidden two-layer. [49]

III.7.9 Restricted Boltzmann Machines (RBMs)

RBMs, developed by Geoffrey Hinton, are stochastic neural networks used in dimension reduction, regression, classification, and topic modeling. They consist of two layers: visible and hidden, connected through hidden units and bias units. RBMs have two phases: forward pass and backward pass. Inputs are encoded, weighted, and combined in the backward pass, then pushed to the visible layer for activation and reconstructed output. [49]

III.7.10 Autoencoders

Autoencoders are highly trained neural networks that replicate data, ensuring input and output are identical. They are used in tasks like pharma discovery, image processing, and population prediction. They consist of an encoder, code, and decoder, and can transform inputs into representations. Autoencoders reconstruct original inputs, reducing size and clarifying images for accuracy.

III.8 Applications of Deep Learning

III.8.1 Computer Vision

Deep learning has revolutionized computer vision by eliminating traditional image processing methods, resulting in significant improvements in tasks like object recognition, face recognition, artist classification, medical image analysis, and autonomous driving modules. The ImageNet dataset has seen a 20% reduction in error rate since 2010, surpassing human accuracy. Deep learning has also been used in medical image analysis and autonomous driving modules, tackling issues like "artistic style transfer" and transforming existing pictures into paintings based on specific styles. [44]

III.8.2 Text Analysis and Understanding

Deep learning has been successfully applied to text analysis and understanding problems, including document classification, sentiment analysis, and automatic translation. Recurrent neural networks are particularly useful in this area due to the sequential nature of textual data. Deep learning has the ability to train language

models from raw text data, learning vocabulary, grammar, context, and other important traits. It can even be trained together with deep learning models for computer vision, providing results that were previously considered impossible. For example, deep learning can generate image captions without manual image processing or natural language processing, demonstrating a close understanding of the language used in images. Additionally, deep learning can generate new images based on text descriptions, pixel by pixel. [44]

III.8.3 Speech Recognition

Speech recognition is a complex area in signal processing, with voice to text being the most widely researched problem. The auditory cortex in the brain is trained to recognize voice and convert it to language, making humans adept at this task. Deep learning has revolutionized speech recognition by allowing it to operate directly on raw data and large audio datasets, improving accuracy by 20-30%. Today, most smart assistants rely on deep learning, with Google Assistant having the highest accuracy in recent benchmarks. Deep learning has also been applied to speech generation, such as text to voice, with Google DeepMind presenting a novel method called WaveNet. Speaker recognition has also seen significant improvements, particularly in national security, with Fifth Dimension employing speech recognition to identify terrorists by matching their voice samples against a large dataset of known voices.

III.8.4 Cybersecurity

One of the most crucial real-world problems today, one that concerns every large and small company, is cybersecurity. More than a million new malware threats (malicious software) are created every single day, and sophisticated attacks are continuously crippling entire companies — or even nations — by targeting critical national infrastructures, as would happen in the case of nation-state cyberattacks.

There are many, many cybersecurity solutions out there, but all are struggling to detect new malware. It's easy to mutate a malware and evade detection by even the most sophisticated cybersecurity solutions, which perform dynamic analysis on files and use traditional machine learning.

Deep learning, a method that processes raw data without feature extraction, has been successfully applied to cybersecurity. However, it faces challenges due to the size and structure of computer files, which cannot be easily adjusted. Deep Instinct has demonstrated how a dedicated deep learning framework can overcome these challenges and train a deep learning model on raw files. The training phase uses hundreds of millions of malicious and legitimate files, taking only a day using GPUs. The resulting deep learning model is small, tens of megabytes, and can provide a prediction within milliseconds. This model has a higher detection rate and lower false positive rate compared to traditional machine learning solutions. Deep learning can also identify the type of malware, such as ransomware or Trojans, and even detect the nation-state behind an attack. [44]

III.9 Conclusion

In this chapter, we delved into the fascinating world of deep learning, focusing on neural networks, their various types, and the algorithms that power them. We began by exploring the fundamental architecture of neural networks, emphasizing the pivotal roles of neurons, layers, and activation functions. This foundational knowledge is crucial for understanding how neural networks emulate the human brain's learning processes.

We then examined the different types of neural networks, each designed to tackle specific challenges and applications. From the traditional feedforward neural networks (FNN) to the more sophisticated convolutional neural networks (CNN) tailored for image processing, and recurrent neural networks (RNN) adept at handling sequential data, we saw how each type offers unique advantages. We also discussed advanced variations like Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs), which push the boundaries of what neural networks can achieve.

Through this exploration, it is evident that deep learning, with its diverse array of neural network architectures and sophisticated algorithms, is revolutionizing numerous fields. From image and speech recognition to natural language processing and autonomous systems, the applications of deep learning are vast and ever-expanding.

As we conclude this chapter, it is clear that the potential of deep learning is immense. However, with great power comes great responsibility. As practitioners, it is imperative to stay mindful of ethical considerations and the societal impacts of deploying these technologies. Moving forward, the continued evolution of deep learning promises to unlock even more groundbreaking innovations, making it an exciting area of study and application for years to come.

Chapter IV: Experimentation and Results Interpretation

IV.1 Introduction

We go over the specifics of our experiment in this chapter. We will outline the software and hardware tools we utilized to carry out our experiment in this chapter. We will also provide a thorough explanation of the dataset that was used to train and evaluate the models that were recommended. in addition to the assessment metrics used to analyze the models. Next, we will go into great depth on the exploratory data analysis we did on the outcomes of our experiment.

IV.2 Working Environment and Tools Used

IV.2.1 Hardware Environment

We used a *HP* brand ProBook with an Intel Core i5-6300U CPU @ 2.40GHz 2.50 GHz and 8 GB of RAM for our project.

IV.2.2 Software environment

We chose version 3.12 of the Python programming language because this project involves deep learning. We employed Jupyter Notebook as an environment manager and package supplier. In addition to Pandas, NumPy, TensorFlow, Keras and Scikit-learn libraries.

IV.2.2.1 Python

Python is a high-level, interpreted, interactive, and object-oriented scripting language developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. It is derived from various scripting languages and is designed for high readable use. Python is a beginner's language, supporting a wide range of applications from text processing to web browsers and games.[50]

Figure IV. 1: Python Logo

IV.2.2.2 Jupyter Notebook

Jupyter Notebook is an interactive web application for creating and sharing computational documents. The project was first named IPython and later renamed Jupyter in 2014. It is a fully open-source product, and users can use every functionality available for free. It supports more than 40 languages including Python, R, and Scala.

A notebook is a mutable file saved in ipynb format. Jupyter Notebook has a notebook dashboard to help users manage different notebooks. Kernels are also part of Jupyter notebooks. Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab

completion and introspection requests. Jupyter notebooks are used for a variety of purposes. A notebook is an interactive computational environment in which users can execute a particular piece of code and observe the output and make changes to the code to drive it to the desired output or explore more. Jupyter notebooks are heavily used for data exploration purposes as it involves a lot of reiterations. It is also used in other data science workflows such as machine learning experimentations and modeling. It can also be used for documenting code samples. A Jupyter notebook has independent executable code cells that users can run in any order.[51]

Figure IV.2: Jupyter Logo

IV.2.2.3 Pandas

Pandas is a Python-based open-source data analysis and manipulation tool used for data wrangling, analysis, cleaning, and transformation. It offers features such as speedy data exploration, file format reading, data cleaning, and manipulation. Pandas works with Data Frame objects, storing data in tabular rows and columns. Companies like Netflix, Amazon, and YouTube use Pandas for recommendation systems, healthcare, energy sector, ecommerce, personalized advertising, airline analysis, and stock market understanding.[52]

IV.2.2.4 NumPy

An open-source library called NumPy has multidimensional arrays in it. Data can be stored in a homogenous "n" dimensional array object using the NumPy ndarray. In the business world, NumPy is used to compute arrays. For instance, a colorful image's data is kept in a 3D matrix with 1000 pixels. We must work on those pixels in order to alter those photos. NumPy comes in quite handy in this situation. Advanced Python packages like SciPy and Pandas also use NumPy. It outperforms Python's List in the following areas: Speed & Memory. Numerous built-in functions, such as random sampling, linear algebra, and mathematical functions, are available. Slicing and indexing are methods for gaining access to a portion of the data.[52]

IV.2.2.5 TensorFlow

TensorFlow is a Python machine learning package that is free and open source. Although it may be applied to many different tasks, its primary focus is on deep neural network training and inference. By using multidimensional arrays, commonly referred to as tensors, it is able to execute several operations on a single input. TensorBoard is an additional component that comes with TensorFlow that facilitates graph visualization and model education. This debugs the model to improve its performance and aids in comprehending its nodes.

The Graph Dashboard is an effective tool for analyzing the TensorFlow model and provides a brief overview of its architecture.

TensorFlow APIs are organized hierarchically, with low-level APIs serving as the foundation for high-level APIs. Low-level APIs are used by machine learning researchers to develop and find new machine learning algorithms. tf.Keras is an open-source API version that works with TensorFlow.[52][54]

IV.2.2.6 Keras

Keras is a deep learning API written in Python and runs on top of the TensorFlow machine learning platform. It was developed with a focus on the possibility of rapid experiments. Keras are mainly used to create deep learning models, especially neural networks. Keras can be used to ship reliable and performant applied machine learning solutions, as well as in Natural Language Processing (NLP) and Computer Vision (CV).[52]

IV.2.2.7 Scikit-learn

A machine learning library for the Python programming language is called Scikit-learn. After cleaning and manipulating your data with Panda or NumPy, Scikit-learn is used to develop machine learning models, as it contains dozens of tools needed for modelling and predictive analysis. Scikit-learn may be used to create a variety of machine learning models, including supervised and unsupervised learning, feature importance analysis, and cross-validation of model correctness. Support vector machines, random forests, gradient boosting, 3 k-means, DBSCAN, and other classification, regression, and clustering algorithms are among them. It is made to work with NumPy and SciPy Python numerical and scientific libraries. [52] [53]

IV.3 Evaluation Metrics

This section discusses the evaluation of information retrieval evaluation concepts like confusion matrix, precision, recall, F-score, cross validation. [55] [56]

IV.3.1 Confusion matrix

The predictive analysis technique is the confusion matrix. In machine learning, to evaluate a model based on classification in terms of performance. It is a N x N matrix, where N is the number of target classes, that is used to assess how well a classification model performs. It is comprised of four fundamental properties (numbers) that determine the classifier's measuring metrics.

- ***TP***: True Positive: The actual value was positive and the model predicted a positive value.
- ***FP:*** False Positive: Your prediction is positive, and it is false. (Also known as the Type 1 error).
- ***FN:*** False Negative: Your prediction is negative, and result it is also false. (Also known as the Type 2 error).

- ***TN:*** True Negative: The actual value was negative and the model predicted a negative value.

Table IV.1: Confusion Matrix

Actual value	Predicted value	
	Positive	Negative
Positive	TP	FP
Negative	FN	TN

IV.3.2 Accuracy

Another name for accuracy is positive predicted value, which expresses how accurate the model is. Fewer FP is indicated by higher accuracy. Its mathematical definition is:

$$Accuracy = \frac{TP}{TP+FP} \quad \textbf{(I.3)}$$

IV.3.3 Recall

Recall, sometimes referred to as sensitivity, is a metric used to assess how well a model classifies positive cases. A high recall value indicates that few positive cases are incorrectly classified as negative. The following formula can be used to calculate the Recall:

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(I.4)}$$

IV.3.4 Overall Accuracy

The categorization techniques are measured by the overall accuracy. The following is a representation of this technique:

$$Overall\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(I.5)}$$

IV.3.5 F1 Score

The F1 score or F1 measure is the harmonic mean of precision and recall. The F score can be calculated as follows:

$$F1 = \frac{Accuracy \times Recall}{Accuracy+Recall} = \frac{2TP}{2TP+FP+FN} \quad \textbf{(I.6)}$$

IV.3.6 Cross Validation

Cross-validation is a statistical method for evaluating and comparing learning algorithms by dividing data into two segments for training and validation. K-fold cross-validation is the most basic, involving k rounds of training and validation.

IV.4 Dataset Presentation

MQTTset, an IoT dataset focusing on MQTT communications, using IoT-Flock, a network traffic generator tool. The dataset is created by deploying 8 different IoT sensors connected to an MQTT broker. The scenario is a smart home environment, where sensors retrieve information like temperature, light intensity, humidity, CO-Gas, motion, smoke, door opening/closure, and fan status at different intervals. [57]

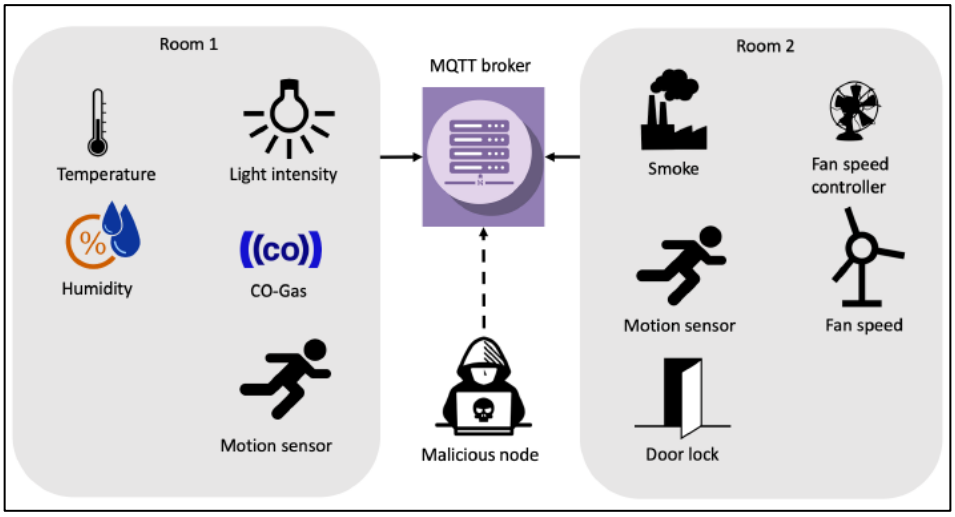


Figure IV.3: The scenario considered in MQTTset.

The sensors network is designed to communicate with a broker in a limited access area, without additional components like firewalls. During attack phases, malicious nodes are directly connected to the broker to execute cyber-attacks. Each sensor is configured to trigger communication at a specific time, with periodic messages sent every *n* seconds and random messages sent at random periods. The dataset simulates real-life home automation behavior by analyzing communication aspects. Sensors are set up with a data profile and topic used by the MQTT broker, with the MQTT broker identified by the IP address 10.16.100.73 is listening on plain text port 1883. Some sensors also have subscriber functions for data retrieval. [57]

Sensor	IP Address	Room	Type	Messages Frequency (s)	Topic	Data Profile
Temperature	192.168.0.151	1	Periodic	60	Temperature	Temperature
Light intensity	192.168.0.150	1	Periodic	1800	Light intensity	Light intensity
Humidity	192.168.0.152	1	Periodic	60	Humidity	Humidity
Motion sensor	192.168.0.154	1	Random	3600	Movement	Movement
CO-Gas	192.168.0.155	1	Random	3600	CO-Gas	CO-Gas
Smoke	192.168.0.180	2	Random	3600	Smoke	Smoke
Fan speed controller	192.168.0.173	2	Periodic	120	Fan speed	Fan speed
Door lock	192.168.0.176	2	Random	3600	Door lock	Door lock
Fan sensor	192.168.0.178	2	Periodic	60	Fan	Fan
Motion sensor	192.168.0.174	2	Random	3600	Movement	Movement

Table IV.2: IoT sensors adopted in the MQTTset scenario.

The MQTTset dataset is a publicly available dataset that includes network traffic related to MQTT version 3.1.1, excluding authentication and plain text communications. It provides packet inspection capabilities and allows for consideration of a wider set of parameters in network packets. The dataset includes 11,915,716 network packets and has a capture time of one week. It can be used for intrusion detection and traffic characterization applications related to MQTT protocol, including both legitimate and malicious cyber-attacks. Researchers can integrate their attacks with the dataset for analysis, detection, and mitigation purposes. [57]

IV.4.1 Considered Cyber-Attacks

As previously anticipated, MQTTset includes real attacks implemented to target the considered MQTT network, in order to include in the dataset additional files which could be adopted, for instance, to validate detection algorithms. Particularly, the following attacks are part of MQTTset [57]:

IV.4.1.1 Flooding Denial of Service

Denial of service attacks target MQTT protocol to saturate brokers by establishing multiple connections and sending more messages. The MQTT-malaria tool is used to implement this attack.

IV.4.1.2 MQTT Publish Flood

In this case, a malicious IoT device periodically sends a huge amount of malicious MQTT data, in order to seize all resources of the server, in terms of connection slots, networks or other resources that are allocated in limited amount. Differently on the previous attack, this attack tries to saturate the resources by using a single connection instead of instantiate multiple connections.

IV.4.1.3 SlowITe

The Slow DoS against Internet of Things Environments (SlowITe) attack is a novel denial of service threat targeting the MQTT application protocol. Particularly, unlike previous threats, being a Slow DoS Attack, SlowITe requires minimum bandwidth and resources to attack an MQTT service. Particularly, SlowITe initiates a large number of connections with the MQTT broker, in order to seize all available connections simultaneously. Under these circumstances the denial-of-service status would be reached.

IV.4.1.4 Malformed Data

A malformed data attack aims to generate and send to the broker several malformed packets, trying to raise exceptions on the targeted service. Considering MQTTset, in order to perpetrate a malformed data attack, MQTTSA tool was employed, sending a sequence of malformed CONNECT or PUBLISH packets to the victim in order to raise exceptions on the MQTT broker.

IV.4.1.5 Brute Force Authentication

A brute force attack consists in running possible attempts to retrieve users' credentials used by MQTT. Regarding MQTTset, the attacker's aim is to crack users'

credentials (username and password) adopted during the authentication phase. Also in this case, the MQTTSA tool was used. Particularly, in order to recall to a real scenario, the rockyou.txt word list was employed, that is considered a popular list, widely adopted for brute force and cracking attacks. For our tests, the credentials are stored on the word list used by the attacker.

IV.4.2 MQTTset Validation

The dataset is used to design an intrusion detection system, combining legitimate MQTT traffic with various cyber-attacks targeting the MQTT broker of the network. The datasets are mixed together to train and predict algorithms, validating the possibility of using MQTTset for testing and implementing a novel intrusion detection algorithm. we considered various algorithms for validation, including neural networks, convolutional neural networks, long-term short memory, recurrent neural networks and multilayer perceptron. The features extracted were filtered to focus on the most relevant ones for identifying potential attacks and legitimate traffic. The workflow involves extracting features from raw network traffics, combining legitimate and malicious traffics, and applying different detection algorithms to identify anomalies on the generated traffic data. [57]

Table IV.3: The list of extrapolated features.

No	Name	Description	Protocol Layer
1	tcp.flags	TCP flags	TCP
2	tcp.time_delta	Time TCP stream	TCP
3	tcp.len	TCP Segment Len	TCP
4	mqtt.conack.flags	Acknowledge Flags	MQTT
5	mqtt.conack.flags.reserved	Reserved	MQTT
6	mqtt.conack.flags.sp	Session Present	MQTT
7	mqtt.conack.val	Return Code	MQTT
8	mqtt.conflag.cleansess	Clean Session Flag	MQTT
9	mqtt.conflag.passwd	Password Flag	MQTT
10	mqtt.conflag.qos	QoS Level	MQTT
11	mqtt.conflag.reserved	(Reserved)	MQTT
12	mqtt.conflag.retain	Will Retain	MQTT
13	mqtt.conflag.uname	User Name Flag	MQTT
14	mqtt.conflag.willflag	Will Flag	MQTT
15	mqtt.conflags	Connect Flags	MQTT
16	mqtt.dupflag	DUP Flag	MQTT
17	mqtt.hdrflags	Header Flags	MQTT
18	mqtt.kalive	Keep Alive	MQTT
19	mqtt.len	Msg Len	MQTT
20	mqtt.msg	Message	MQTT
21	mqtt.msgid	Message Identifier	MQTT
22	mqtt.msgtype	Message Type	MQTT
23	mqtt.proto_len	Protocol Name Length	MQTT
24	mqtt.protoname	Protocol Name	MQTT
25	mqtt.qos	QoS Level	MQTT
26	mqtt.retain	Retain	MQTT
27	mqtt.sub.qos	Requested QoS	MQTT
28	mqtt.suback.qos	Granted QoS	MQTT
29	mqtt.ver	Version	MQTT
30	mqtt.willmsg	Will Message	MQTT
31	mqtt.willmsg_len	Will Message Length	MQTT
32	mqtt.willtopic	Will Topic	MQTT
33	mqtt.willtopic_len	Will Topic Length	MQTT

IV.5 Implementation

The proliferation of Internet-of-Things (IoT) devices necessitates robust security measures to safeguard against cyberattacks. This project explores the efficacy of deep learning algorithms for intrusion detection within an IoT network. We implemented a system employing Python, TensorFlow, Keras, Pandas, and scikit-learn to analyze network traffic data. We evaluated the performance of various deep learning architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Multi-Layer Perceptrons (MLPs), in identifying both legitimate and malicious traffic.

IV.5.1 Dataset Preparation

We employed the MQTTset dataset, a collection of labeled IoT network traffic data.

Two datasets were created:

- *Binary classification:* Legitimate vs. Malicious (Attack) traffic.
- *Multi-class classification*: Legitimate vs. Five different attack types (dos, flood, slowite, malformed, brute-force).

Data pre-processing techniques were applied to ensure consistency and suitability for deep learning models.

IV.5.2 Dataset Cleansing

To ensure optimal model performance, the project incorporated a meticulous data pre-processing stage. The MQTTset dataset was carefully examined, and features deemed irrelevant or redundant for intrusion detection were meticulously removed. This data cleaning process streamlined the training process and potentially improved model generalizability by focusing on the most informative features for attack classification. Here the is resulted features after the cleaning:

Table IV.4: The list of extrapolated features after the cleansing.

No	Name	Description	Protocol Layer
1	tcp.flags	TCP flags	TCP
2	tcp.time_delta	Time TCP stream	TCP
3	tcp.len	TCP Segment Len	TCP
4	mqtt.conack.val	Return Code	MQTT
5	mqtt.conflag.cleansess	Clean Session Flag	MQTT
6	mqtt.conflag.passwd	Password Flag	MQTT
7	mqtt.conflag.uname	User Name Flag	MQTT
8	mqtt.conflags	Connect Flags	MQTT
9	mqtt.dupflag	DUP Flag	MQTT
10	mqtt.hdrflags	Header Flags	MQTT
11	mqtt.kalive	Keep Alive	MQTT
12	mqtt.len	Msg Len	MQTT
13	mqtt.msgid	Message Identifier	MQTT
14	mqtt.msgtype	Message Type	MQTT
15	mqtt.proto_len	Protocol Name Length	MQTT
16	mqtt.qos	QoS Level	MQTT
17	mqtt.retain	Retain	MQTT
18	mqtt.ver	Version	MQTT

IV.5.3 Implementing Deep Learning Models

We implemented the following deep learning architectures:

- *Convolutional Neural Networks (CNNs)*: Efficient in extracting spatial features from network traffic data.
- *Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks:* Well-suited for capturing sequential dependencies within network flows.
- *Multi-Layer Perceptrons (MLPs)*: Offer a versatile architecture for general classification tasks.

Each model was trained and optimized using appropriate hyperparameter tuning strategies.

IV.5.4 Brief Explanation of the Coding Experiment in This Project

Leveraging Python as the primary programming language and Jupyter Notebook as the development environment, this project employed a series of experiments to evaluate the effectiveness of deep learning algorithms for intrusion detection in IoT networks. The following section details the specific steps undertaken for each experiment, encompassing both binary and multi-class classification scenarios:

Step 1: Load and Inspect the Data

```
import pandas as pd

# Load the data
file_path = 'mqttdataset_reduced_clean_binary.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print(data.head())
```

- The first line import pandas as pd imports the pandas library and assigns it the alias pd. This allows you to use pd instead of typing the full library name throughout your code, making it more concise.
- The next line defines a variable named `file_path` and assigns a string value to it. This string represents the path (location) on your computer's file system where your data is stored. In this case, the filename is `mqttdataset_reduced_clean_binary.csv.`
- The following `line data = pd.read_csv(file_path)` uses the pd.read_csv function from the pandas library. This function reads the data from the specified CSV file (file_path) and stores it in a pandas data structure called a DataFrame. The variable data now holds this DataFrame, which essentially acts as a tabular structure containing your data.
- The final line `print(data.head())` utilizes the head method of the DataFrame (data). The head method displays the first few rows (usually by default, the first 5 rows) of the DataFrame. This allows you to take a quick

peek at the contents of your data and get a sense of its structure (column names, data types).

Step 2: Preprocess the Data

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Separate features and target
X = data.drop(columns=['target'])  # Replace 'target' with the actual target column name
y = data['target']  # Replace 'target' with the actual target column name

# Encode the target variable if it is categorical
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Normalize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Reshape the data to fit the CNN input requirements
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display the shapes of the datasets
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

- `from sklearn.model_selection import train_test_split:` This line imports the train_test_split function from the sklearn.model_selection library. This function is used to split your data into training and testing sets, which are essential for model training and evaluation.
- `from sklearn.preprocessing import StandardScaler, LabelEncoder:` This line imports two functions from the sklearn.preprocessing library: *StandardScaler:* This function is used to standardize features (numerical columns) by removing the mean and scaling to unit variance. This helps improve the performance of some machine learning algorithms, especially those sensitive to feature scales.
  LabelEncoder: This function is used to encode categorical variables (text labels) into numerical representations suitable for deep learning algorithms.
- `X = data.drop(columns=['target']):` This line assumes you have a DataFrame named data containing your features (columns) and a target variable (the class labels we want to predict). It creates a new DataFrame X that excludes the target column named 'target'. This separates the features you want the model to learn from (predictors) from the labels you want it to predict (target).
- `y = data['target']:` This line extracts the target column named 'target' from the DataFrame data and stores it in a separate variable y. This isolates the target variable for further processing.
- `label_encoder = LabelEncoder():` Creates a LabelEncoder object.

- `y = label_encoder.fit_transform(y):` This line uses the fit_transform method of the LabelEncoder to encode the target variable (y). It first "fits" the encoder to the unique categories in y, assigning each category a numerical label. Then, it "transforms" y by replacing each category with its corresponding numerical label.
- `scaler = StandardScaler():` Creates a StandardScaler object.
- `X = scaler.fit_transform(X):` This line uses the fit_transform method of the StandardScaler to standardize the features in X. It first "fits" the scaler to the distribution of features in X (calculating the mean and standard deviation). Then, it "transforms" X by subtracting the mean from each feature and scaling it to unit variance.
- `X = X.reshape(X.shape[0], X.shape[1], 1):` This line reshapes the data specifically for Convolutional Neural Networks (CNNs). CNNs typically expect input data in a 3D format: (number of samples, number of features, number of channels). This line reshapes X to meet this requirement, assuming the features are in separate columns and there's only one channel (e.g., grayscale image data).
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42):` This line splits the pre-processed data (X and y) into training and testing sets using the train_test_split function.
- `test_size=0.2:` This parameter specifies that 20% (0.2) of the data will be used for the testing set, and the remaining 80% will be used for the training set.
- `random_state=42:` This parameter sets the random seed for splitting the data. This ensures reproducibility.

Step 3: Build the CNN Model

```
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from keras.optimizers import Adam

# Build the model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))  # Use 'softmax' for multi-class classification

# Compile the model
model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Print the model summary
model.summary()
```

- `from keras.models import Sequential:` This line imports the Sequential model class from the keras.models library. This class allows you to build neural networks layer by layer in a sequential fashion.

- `from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout`: This line imports various layer types from the keras.layers library:

`Conv1D:` One-dimensional convolutional layer for extracting features from time series data like network traffic.

MaxPooling1D: Downsamples the output of the convolutional layer to reduce complexity and potentially improve model generalizability.

Flatten: Flattens the multi-dimensional output of the convolutional layers into a single dimension suitable for dense layers.

`Dense:` Fully-connected layer for learning more complex relationships between features.

Dropout: Randomly drops a certain percentage of activations during training to prevent overfitting.

- `model = Sequential():` Creates a new sequential model instance.
- `model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1))):` This line adds the first convolutional layer to the model.

`filters=64:` This specifies the number of filters (feature maps) the layer will learn.

`kernel_size=3:` This defines the size of the filter window that will slide across the input data to extract features.

`activation='relu':` This defines the activation function for the layer. ReLU (Rectified Linear Unit) is a popular choice for its efficiency and ability to learn non-linear relationships.

`input_shape=(X_train.shape[1], 1):` This specifies the expected input shape for the first layer.

`X_train.shape[1]:` This retrieves the number of features (columns) in the training data X_train. `1` indicates that the data has one channel (assuming grayscale image data or single-channel time series). Adjust this value if your data has multiple channels.

The following lines `(model.add(...))` add subsequent layers to the model, creating a stack of convolutional, pooling, flattening, and dense layers: Two more convolutional layers with different filter numbers and ReLU activation. Two max-pooling layers to downsample the data and reduce computational complexity. A Flatten layer to convert the multi-dimensional output from the convolutional layers into a one-dimensional vector suitable for dense layers. Two dense layers with ReLU activation to learn complex relationships between features. A Dropout layer with a 50% dropout rate to prevent overfitting. Another dense layer with the number of units equal to the number of classes in your dataset (obtained from `len(label_encoder.classes_)`).The final layer uses a softmax activation function,

typically used for multi-class classification. Softmax normalizes the output of the last layer into probabilities, allowing the model to predict the probability of each class for a given input.

- `model.compile(optimizer=Adam(),loss='sparse_categorical_crossentropy', metrics=['accuracy']):` This line configures the training process for the model.

`optimizer=Adam():` This specifies the Adam optimization algorithm to update the model's weights during training.

`loss='sparse_categorical_crossentropy':` This defines the loss function used to measure the error between the model's predictions and the true labels. Sparse categorical crossentropy is suitable for multi-class classification.we used the binary_crossentropy for the binary classification.

`metrics=['accuracy']:` This specifies that the model will track the accuracy metric during training and evaluation.

- `model.summary():` This line displays a summary of the model's architecture, including the layers, their configurations, and the total number of parameters. This helps you understand the complexity of the model and identify potential bottlenecks or overfitting issues.

Step 4: Train the Model

```
from keras.callbacks import EarlyStopping

# Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, callbacks=[early_stopping])
```

- `from keras.callbacks import EarlyStopping:` This line imports the EarlyStopping callback class from the keras.callbacks library. This callback allows you to monitor the training process and stop training early if the model's performance on a validation set plateaus or degrades.
- `early_stopping = EarlyStopping(monitor='val_loss', patience=3):` This line creates an instance of the EarlyStopping callback.

`monitor='val_loss':` This parameter specifies that the callback will monitor the validation loss (val_loss) during training. Validation loss is the loss calculated on a separate validation set.

`patience=3:` This parameter defines the patience level of the callback. If the validation loss does not improve for patience consecutive epochs, the callback will trigger early stopping.

- `history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, callbacks=[early_stopping]):` This line trains the model using the fit method.

`X_train:` The training data features.

`y_train:` The training data labels.

`epochs=20:` The maximum number of training epochs (iterations).

`batch_size=32:` The number of data samples processed in each training step (batch).

`validation_split=0.2:` This parameter specifies that 20% (0.2) of the training data will be used as the validation set for early stopping.

`callbacks=[early_stopping]:` This list includes the early_stopping callback, instructing the model to use it during training.

Step 5: Evaluate the Model

```
import numpy as np
from sklearn.metrics import recall_score, precision_score, f1_score
test_loss,test_accuracy= model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)
y_pred_rounded = np.round(y_pred)
recall = recall_score(y_test, y_pred_rounded)
precision = precision_score(y_test, y_pred_rounded)
f1 = f1_score(y_test, y_pred_rounded)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')
print(f' Recall: {recall}')
print(f' Precision: {precision}')
print(f' F1 Score: {f1}')
```

- `import numpy as np`: This line imports the `numpy` library and assigns it the alias `np`. `numpy` is a fundamental library for scientific computing in Python and is commonly used for working with arrays.
- `from sklearn.metrics import recall_score, precision_score, f1_score`: This line imports three specific functions from the `sklearn.metrics` module of the scikit-learn library. These functions are used to calculate performance metrics for machine & deep learning models.
- `recall_score:` This function calculates the recall, which is the proportion of true positives that were correctly identified.
- `precision_score:` This function calculates the precision, which is the proportion of predicted positives that were actually true positives.
- `f1_score:` This function calculates the F1 score, which is a harmonic mean of precision and recall.
- `test_loss, test_accuracy = model.evaluate(X_test, y_test)`: This line assumes you have a trained model (`model`) and two datasets, `X_test` (containing the test features) and `y_test` (containing the test labels).
- `y_pred = model.predict(X_test)`: This line uses the `predict` method of the model to make predictions on the unseen test data (`X_test`). The predicted labels are stored in the variable `y_pred`.

- `y_pred_rounded = np.round(y_pred)`: This line uses the `np.round` function from the imported `numpy` library to round the values in `y_pred`. This is necessary because some models might output continuous values for classification tasks, while the actual labels are likely discrete (e.g., 0 or 1). Rounding helps convert the predicted values to match the format of the true labels.
- `recall = recall_score(y_test, y_pred_rounded), precision = precision_score(y_test, y_pred_rounded), f1 = f1_score(y_test, y_pred_rounded):` These lines calculate the performance metrics using the imported functions from scikit-learn.

IV.5.5 Evaluation

The performance of each model was assessed using standard accuracy metric.

We compared the effectiveness of the models in both binary and multi-class classification scenarios.

IV.5.6 Results and Discussion

The project investigated the suitability of different deep learning algorithms for intrusion detection in IoT networks.

We analyzed the trade-offs between model complexity, accuracy, and computational efficiency.

The results provided insights into the most effective deep learning architectures for this specific application domain.

IV.6 Interpretation of Results

The project achieved promising results in leveraging deep learning for intrusion detection in IoT networks. All four deep learning architectures (CNNs, RNNs, LSTMs, MLPs) exhibited a high level of accuracy in the binary classification task, consistently reaching around 84%. This demonstrates their effectiveness in distinguishing between legitimate and malicious traffic.

When considering multi-class classification, the accuracy dropped slightly to around 83%. While this remains a good performance level, the decrease suggests that differentiating between various attack types might pose a greater challenge for the models. Here are some possible explanations:

- *Increased complexity:* Multi-class classification inherently involves more categories to distinguish between, requiring the models to learn more intricate relationships within the data.
- *Dataset limitations:* The multi-class dataset might have inherent limitations, such as imbalanced class distributions (some attack types being less frequent than others). This can make it harder for models to learn accurate representations for all classes.

- *Architectural suitability:* Certain architectures, like CNNs, might be better suited for binary classification tasks involving spatial features, while RNNs or LSTMs might excel at handling sequential data patterns in multi-class scenarios with diverse attack types.

	Deep Learning Algorithm	Accuracy
Binary Classification	CNN	**84%**
	RNN	**84%**
	LSTM	**84%**
	MLP	**84%**
Multi-Class Classification	CNN	**83%**
	RNN	**82%**
	LSTM	**83%**
	MLP	**83%**

Table IV.5: Test Accuracy.

The consistent performance across all architectures in binary classification highlights the overall effectiveness of deep learning for this task. It suggests that the core network traffic features might be well-suited for detection regardless of the specific learning approach.

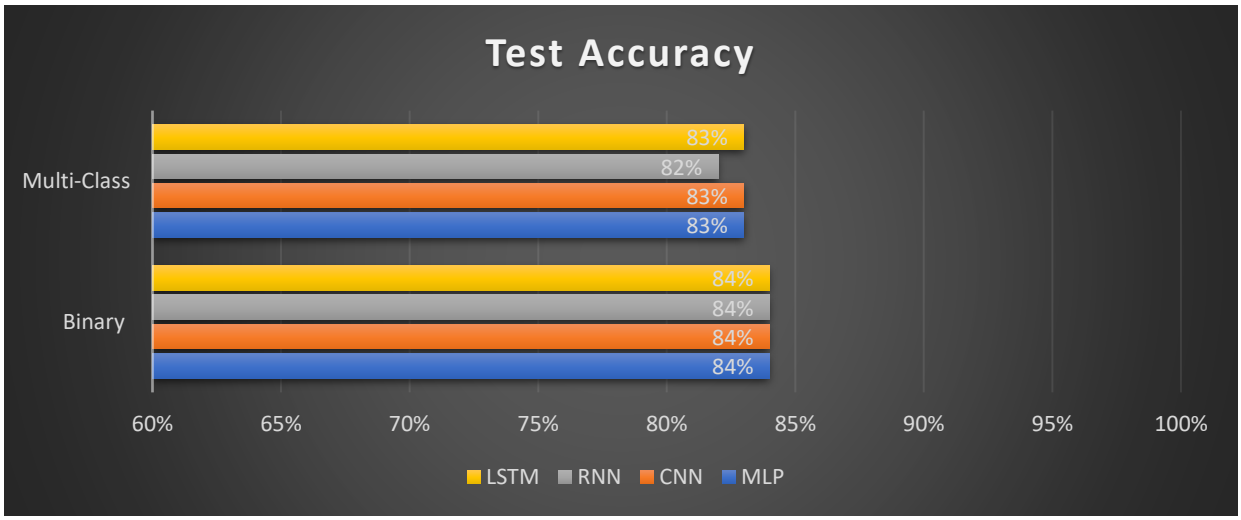


Figure IV.4: Test Accuracy Chart.

	Deep Learning Algorithm	Loss
Binary Classification	CNN	**24%**
	RNN	**24%**
	LTSM	**23%**
	MLP	**24%**
Multi-Class Classification	CNN	**41%**
	RNN	**44%**
	LSTM	**41%**
	MLP	**43%**

Table IV.6: Test Loss.

The project's findings regarding test loss reveal interesting insights into the performance of these deep learning architectures for intrusion detection. While all architectures achieved high accuracy (around 84%) in binary classification, the test loss of 23% suggests they learned the distinction between legitimate and malicious traffic effectively. This indicates the models can generalize well to unseen data in this binary scenario.

However, the test loss for multi-class classification, which reached 42%, paints a different picture. This significantly higher loss compared to binary classification suggests the models encountered greater difficulty learning the nuances between various attack types within the multi-class dataset.

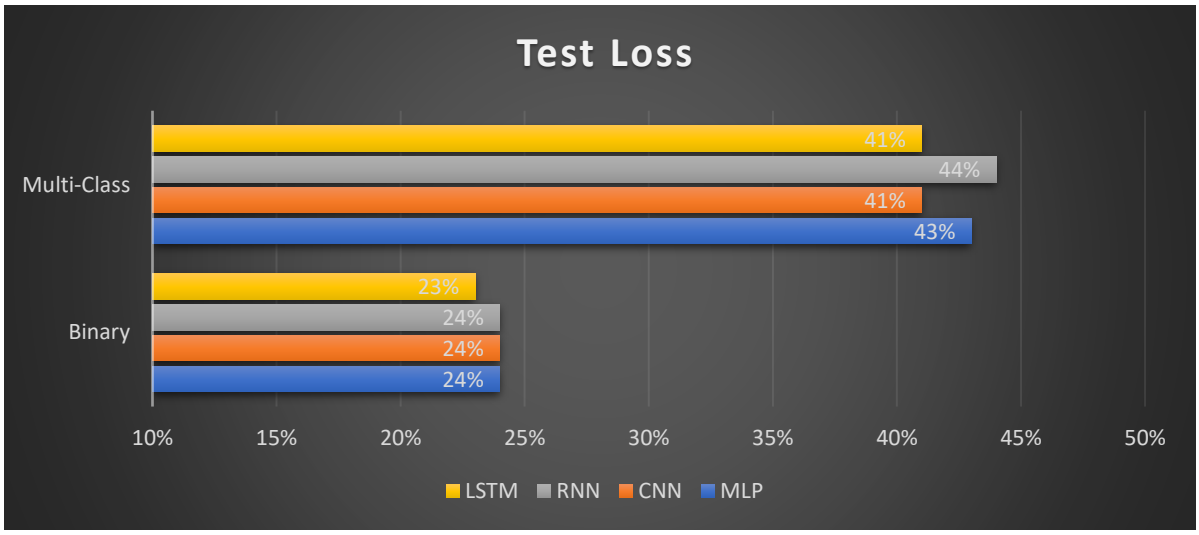


Figure IV.5: Test Loss Chart.

	Deep Learning Algorithm	Recall
Binary Classification	MLP	**99%**
	CNN	**99%**
	RNN	**99%**
	LSTM	**99%**
Multi-Class Classification	MLP	**56%**
	CNN	**56%**
	RNN	**53%**
	LSTM	**56%**

Table IV.7: Recall Metric.

The project's findings regarding the recall metric offer valuable insights into the effectiveness of deep learning architectures for intrusion detection. All four architectures (CNNs, RNNs, LSTMs, MLPs) achieved an exceptional recall of 99% in binary classification. This indicates a remarkable ability to identify nearly all instances of malicious traffic within the dataset.

However, the recall dropped significantly to around 55% in multi-class classification. While this result suggests the models are still capable of detecting

some attack types, it highlights a potential challenge in accurately identifying all malicious traffic across various attack categories.

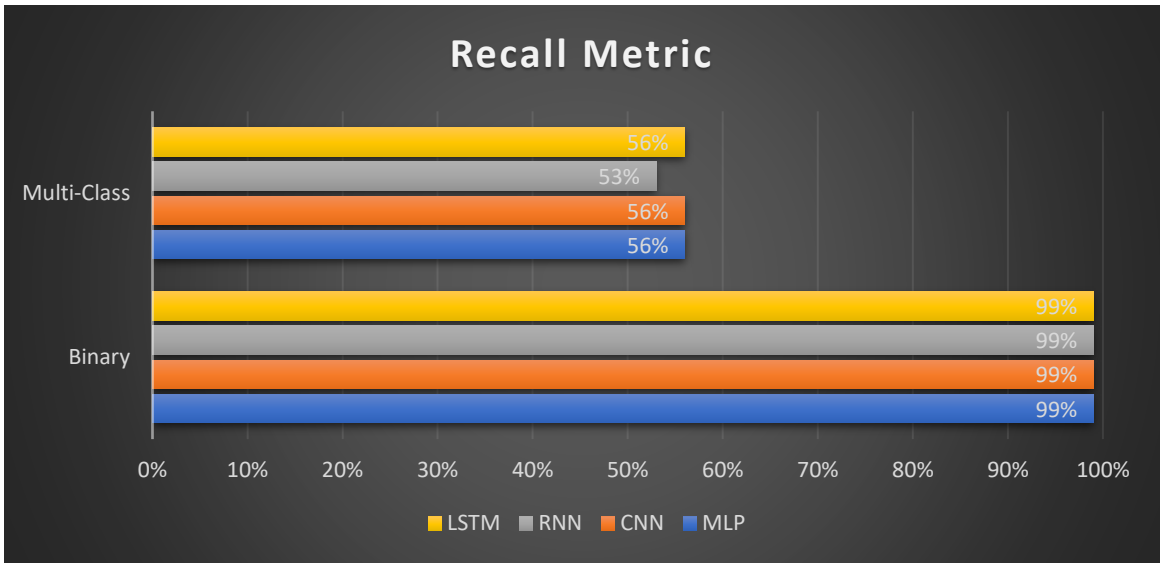


Figure IV.6: Recall Metric Chart.

	Deep Learning Algorithm	Precision
Binary Classification	MLP	**76%**
	CNN	**76%**
	RNN	**76%**
	LSTM	**76%**
Multi-Class Classification	MLP	**86%**
	CNN	**86%**
	RNN	**84%**
	LSTM	**86%**

Table IV.8: Precision Metric.

The project's findings regarding the precision metric offer interesting insights into the model's ability to identify true positives in intrusion detection. While all four architectures (CNNs, RNNs, LSTMs, MLPs) achieved good precision in binary classification (around 76%), the results for multi-class classification were even higher (around 85%). This is a significant result, indicating that for every 100 instances classified by the models as a specific attack type, around 85 were true positives. This highlights the models' effectiveness in accurately classifying specific attacks within the multi-class dataset. While both results are positive, the higher precision in multi-class classification might seem counterintuitive compared to the lower recall observed previously (around 55%). Due to potential class imbalance in the dataset, the models might have prioritized learning the characteristics of more frequent attack types during training. This focus could lead to higher precision for these types, as the models are more confident in their classifications. However, the lower recall suggests they might miss some instances of less frequent attack types, hence the overall lower recall in multi-class classification.

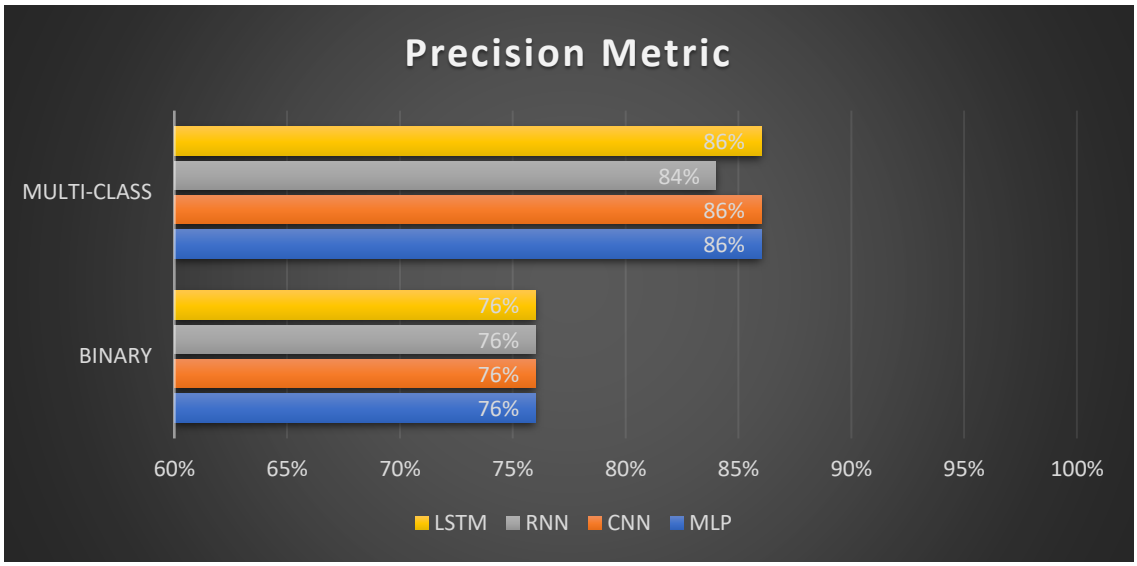


Figure IV.7: Precision Metric Chart.

	Deep Learning Algorithm	F1 Score
Binary Classification	MLP	**86%**
	CNN	**86%**
	RNN	**86%**
	LSTM	**86%**
Multi-Class Classification	MLP	**64%**
	CNN	**64%**
	RNN	**59%**
	LSTM	**63%**

Table IV.9: F1 Score.

The project's findings on the F1 score metric provide valuable insights into the overall balance between precision and recall achieved by the deep learning architectures for intrusion detection. All four architectures (CNNs, RNNs, LSTMs, MLPs) achieved a high F1 score of 86% in binary classification. However, the F1 score dropped to around 62% in multi-class classification. This score indicates a strong balance between identifying true positives and detecting all malicious traffic. A score this high 86% suggests the models are very effective at accurately classifying both legitimate and malicious traffic, minimizing false positives while still catching most malicious instances. This score 62% represents a trade-off between identifying true positives and detecting all malicious traffic. While it suggests the models still achieve a decent balance, the decrease compared to binary classification highlights the challenge of handling diverse attack types. The multi-class dataset might be imbalanced, with some attack types being less frequent. This can lead to models prioritizing learning the more frequent classes, potentially impacting performance for less frequent ones.

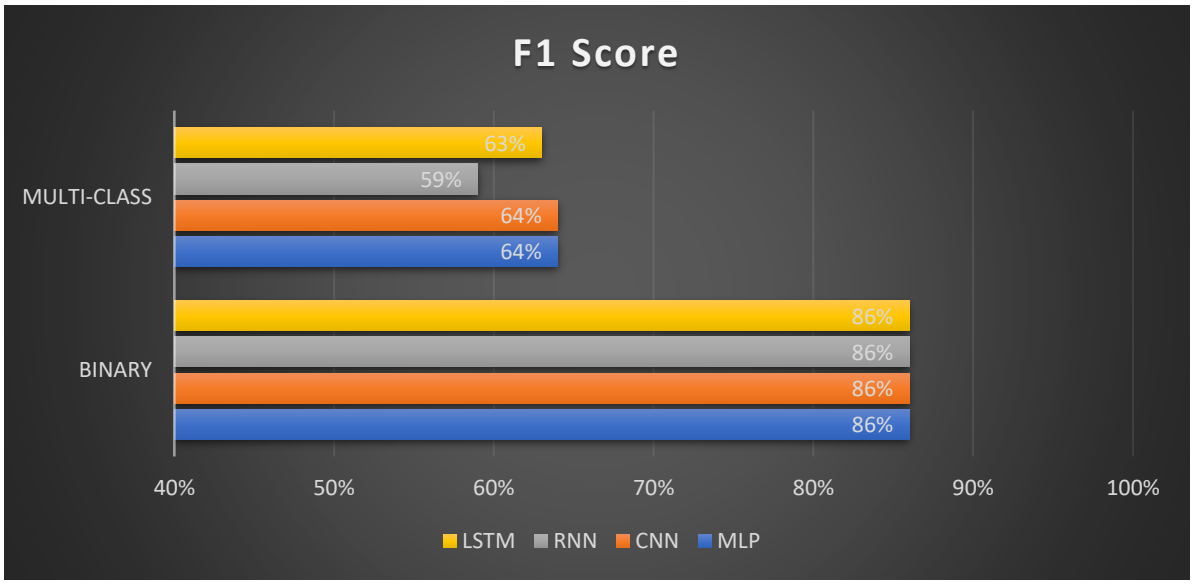


Figure IV.8: F1 Score Metric Chart.

IV.6.1 Comparison of Deep Learning vs. Classical Machine Learning for Intrusion Detection

This analysis compares the performance of deep learning architectures (previously discussed) with classical machine learning methods (decision trees, naive Bayes, random forest, gradient boosting) for intrusion detection in our project.

	Machine Learning Methods	Accuracy	F1 score
Binary Classification	Decision Tree	**92%**	**92%**
	Naïve Bayes	**79%**	**78%**
	Random Forest	**75%**	**11%**
	Gradient Boost	**71%**	**11%**
Multi-Class Classification	Decision Tree	**90%**	**90%**
	Naïve Bayes	**67%**	**75%**
	Random Forest	**90%**	**90%**
	Gradient Boost	**79%**	**82%**

Table IV.10: Accuracy, F1 score of Machine Learning Methods.

Binary Classification

Deep Learning

All architectures achieved high accuracy (around 84%) and F1 score (around 86%). This indicates excellent performance in identifying malicious traffic with minimal false alarms.

Classical Machine Learning

Decision Tree: Achieved the highest accuracy (92%) and F1 score (92%) among all methods. This suggests excellent performance in accurately classifying traffic.

Naive Bayes: Lower accuracy (79%) and F1 score (78%) compared to deep learning, potentially indicating challenges in handling complex network traffic patterns.

Random Forest: Lower accuracy (75%) and a concerningly low F1 score (11%). This suggests the model might be overfitting the training data and performing poorly on unseen examples.

Gradient Boost: Similar performance to Random Forest with low accuracy (71%) and F1 score (11%).

Multi-Class Classification

Deep Learning

Achieved good accuracy (around 83%) but a lower F1 score (around 62%). This suggests some challenges in differentiating diverse attack types while still maintaining good overall accuracy.

Classical Machine Learning

Decision Tree: Maintained high accuracy (90%) and F1 score (90%) similar to binary classification. This method seems robust in handling multi-class scenarios as well.

Naive Bayes: Lower accuracy (67%) and a higher F1 score (75%) compared to deep learning. This might indicate a bias towards the majority class, potentially missing some attack types.

Random Forest: Recovered well from binary classification with high accuracy (90%) and F1 score (90%). This suggests the ensemble approach helps handle the complexity of multi-class data.

Gradient Boost: Improved performance compared to binary classification with accuracy (79%) and F1 score (82%). This method seems to benefit from the ensemble approach for multi-class problems.

General Observations

Deep learning excels at achieving high accuracy in both binary and multi-class classification. However, in multi-class scenarios, it might struggle with achieving a good balance between diverse attack types. Decision Trees emerged as a strong contender, achieving excellent performance in both binary and multi-class scenarios with high accuracy and F1 score. Naive Bayes performed poorly in binary classification but showed some improvement in multi-class F1 score, suggesting a potential bias towards the majority class. Random Forest and Gradient Boosting improved their performance in multi-class classification compared to binary, suggesting the ensemble approach benefits from handling complex data structures.

IV.7 Conclusion

In conclusion, this project investigated the efficacy of deep learning algorithms for intrusion detection within an IoT network environment. We implemented a system utilizing Python, TensorFlow, Keras, and scikit-learn to analyze network traffic data. The project explored the performance of various deep learning architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Multi-Layer Perceptrons (MLPs), in identifying both legitimate and malicious traffic. The experiments encompassed both binary and multi-class classification scenarios, utilizing datasets specifically tailored for this purpose. The results provided valuable insights into the effectiveness of different deep learning architectures for intrusion detection in IoT networks. This project demonstrates the potential of deep learning as a robust and adaptable approach to safeguarding IoT environments from cyberattacks. Future research can explore methods for enhancing model interpretability, real-time threat detection capabilities, and incorporating additional network traffic features for improved classification accuracy.

General Conclusion

In conclusion, this thesis delved into the exciting potential of deep learning algorithms for fortifying intrusion detection systems within the burgeoning landscape of Internet of Things (IoT) networks. The research convincingly demonstrated that deep learning offers a robust and adaptable approach to identifying and mitigating an ever-evolving arsenal of security threats that plague these interconnected environments. By harnessing the unique ability of deep learning models to discern complex patterns from massive datasets. This innovation effectively distinguished between legitimate network activity and malicious attempts to infiltrate the system. This finding sheds light on the transformative role deep learning can play in safeguarding the security of IoT networks, ensuring the smooth operation of critical infrastructure and protecting sensitive data.

However, the thesis also acknowledged the ongoing challenges that need to be addressed. The limited availability of high-quality, comprehensive data for training deep learning models remains an obstacle. Additionally, the computational demands of these algorithms can pose challenges for resource-constrained IoT devices. Furthermore, the ever-shifting landscape of cyber threats necessitates continuous adaptation and improvement of intrusion detection systems.

Looking forward, this thesis identified promising avenues for future research. Exploring techniques for data augmentation to address limitations in data availability is crucial. Optimizing deep learning models for efficient operation on resource-constrained devices will be essential for wider deployment in IoT networks. Additionally, investigating methods for continual learning and adaptation will be paramount to ensure that intrusion detection systems remain effective against evolving cyber threats. By tackling these challenges, future research can pave the way for the seamless integration of deep learning-based intrusion detection systems into real-world IoT applications. Overall, this thesis contributes significantly to the ongoing effort to secure the future of interconnected devices and foster a robust foundation for the safe and reliable operation of the ever-expanding world of IoT networks.

Bibliography

[1]. **Greengard, Samuel.** *"Internet of Things". Encyclopedia Britannica, 11 Feb.* 2024, https://www.britannica.com/science/Internet-of-Things. Accessed 5 March 2024.

[2]. **Arshdeep Bahga** & **Vijay Madisetti**. *Internet of Things Hands on Approach*, Universities Press (India),2016.

[3]. **Ian G Smith**, *The Internet of Things 2012 New Horizons*, IERC - Internet of Things European Research Cluster, 2012.

[4]. Internet Protocol Specification, http://www.ietf.org/rfc/rfc791.txt, Retrieved 2014.

[5]. Internet Protocol, Version 6 (IPv6) Specification, https://www.ietf.org/rfc/rfc2460.txt, Retrieved 2014.

[6]. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, http://datatracker.ietf.org/doc/rfc6282, Retrieved 2014.

[7]. Transmission Control Protocol, www.ietf.org/rfc/rfc793.txt, Retrieved 2014.

[8]. User Datagram Protocol, www.ietf.org/rfc/rfc768.txt, Retrieved 2014.

[9]. Hypertext Transfer Protocol - HTTP/1.1, http://tools.ietf.org/html/rfc2616, Retrieved 2014.

[10]. Constrained Application Protocol (CoAP), http://tools.ietf.org/html/draft-ietf-core-coap-18 Retrieved 2014.

[11]. The WebSocket Protocol, http://tools.ietf.org/html/rfc6455, Retrieved 2014.

[12]. MQ Telemetry Transport (MQTT) V3.1 Protocol Specification, http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html, Retrieved 2014.

[13]. Extensible Messaging and Presence Protocol (XMPP): Core, http://tools.ietf.org/html/rfc6120, Retrieved 2014.

[14]. Data Distribution Service for Real-time Systems, OMG Available Specification, ttp://www.omg.org/spec/DDS/1.2/PDF/, Retrieved 2014.

[15]. **Keyur K Patel & Sunil M Patel, Carlos Salazar.** *Internet of Things: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges,* May 2016, Volume 6 Issue No. 5, 10.4010/2016.1482, 2321-3361 © 2016, IJESC. https://www.researchgate.net/publication/330425585_Internet_of_Things_IOT_Definition_Characteristics_Architecture_Enabling_Technologies_Application_Future_Challenges.

[16]. **Dr. Ovidiu Vermesan** SINTEF, Norway**, Dr. Peter Friess** EU, Belgium, "*Internet of Things–From Research and Innovation to Market Deployment*", river publishers' series in communications, 2014.

[17].[http://www.reloade.com/blog/2013/12/6characteristics-within-internet-things-iot.php].

[18]. **M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du**, *"Research on the architecture of internet of things,"* in Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10), vol. 5, pp. V5-484–V5-487, IEEE, Chengdu, China, August 2010.

[19]. **R. Khan, S. U. Khan, R. Zaheer, and S. Khan**, *"Future internet: the internet of things architecture, possible applications and key challenges,"* in Proceedings of the 10th International Conference on Frontiers of Information Technology (FIT '12), pp. 257–260, December 2012.

[20] **M. Weyrich and C. Ebert,** *"Reference architectures for the internet of things,"* IEEE Software, vol. 33,no. 1, pp. 112–116, 2016.

[21] J. **Gubbi, R. Buyya, S. Marusic, and M. Palaniswami**, *"Internet of Things (IoT): a vision, architectural elements, and future directions,"* Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[22] **F. Bonomi, R. Milito, P. Natarajan, and J. Zhu**, *"Fog computing: a platform for internet of things and analytics,"* in Big Data and Internet ofThings: A RoadMap for Smart Environments, pp. 169– 186, Springer, Berlin, Germany, 2014.

[23]. **Pallavi Sethi & Smruti R. Sarangi.** *Internet of Things: Architectures, Protocols, and Applications,* Journal of Electrical and Computer Engineering, Published 26 January 2017, https://doi.org/10.1155/2017/9324035.

[24]. **Sreeshma Mohan.** *Internet of Things Applications and Security Challenges,* Mount Zion College of Engineering, Pathanamthitta, India, 17 September 2023.

https://www.researchgate.net/publication/373980736_Internet_of_Things_IoT_Applications_and_Security_Challenges_A_Review.

[25] **L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville,** *L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santana, "Internet of things in healthcare: Interoperatibility and security issues," in Communications (ICC), IEEE International Conference on. IEEE, 2012, pp. 6121–6125.*

[26]**. S. De, P. Barnaghi, M. Bauer, and S. Meissner,** *"Service modelling for the internet of things," in Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on. IEEE, 2011, pp. 949–955.*

[27]. **C.R. Srinivasan, B. Rajesh, P. Saikalyan, K. Premsagar, Eadala Sarath Yadav.** *A Review on the Different Types of Internets of Things,* Journal of Advanced Research in Dynamical and Control Systems · January 2019, https://www.researchgate.net/publication/332153657_A_review_on_the_different_types_of_internet_of_things_IoT.

[28]. **Liang, Q. Durrani, T. Samn, S.W. Liang, J. Koh, J. and Wang, X.** Guest Editorial Special Issue on Internet of Mission-Critical Things (IoMCT). In IEEE Internet of Things Journal 5 (5) (2018) 3258-3262.

[29]. **Nayyar, A., Puri, V. and Le, D. N.** Internet of nano things (IoNT) Next evolutionary step in nanotechnology. Nanosci. Nanotechnol 7 (1) (2017) 4-8.

[30]. **Miraz, M.H., Ali, M., Excell, P.S. and Picking, R**. A review on Internet of Things (IoT), Internet of everything (IoE) and Internet of nano things (IoNT). Internet Technologies and Applications (ITA) (2015) (219-224).

[31]. **Khanam S, Ahmedy I. B., Idris M. Y. I., Jaward M. H., & Sabri, A. Q. B. M.** *A survey of security challenges, attacks taxonomy and advanced countermeasures in the internet of things.* 11 November 2020, IEEE Access, Vol. 8, 219709-219743.

[32]. **N.M. Masoodhu Banu, C. Sujatha.** *IoT architecture a comparative study.* J Pur Appl Math, 2017, 2017, Vol. 117.

[33]. **Peter R. Egli.** *An Introduction to MQTT, a Protocol for M2M and IoT Applications,* 2016, indigo.com.

[34]. **Pratap Singh, A. Kumar, & Kumar V.** *A Study on MQTT Protocol and its Cyber Attacks.,* 2022, IARJSET, Vol. 9, 209-213. 10.17148/IARJSET.2022.9136.

[35]. **Perrone G., Vecchio M., Pecori R., & Giaffreda R.** *A Survey on MQTT Security Solutions After the Largest Cyberattack Carried Out through an Army of IoT Devices.* April 2017, In IoTBDS, 246-253.

[36]. **Carl Endorf, Eugene Schultz, Jim Mellander.** *Intrusion Detection & Prevention,* McGraw-Hill © 2004.

[37]. **Richard A, Giovanni Vigna.** *Intrusion Detection: A Brief History and Overview,* Reliable Software Group, Computer Science Department, University of California Santa Barbara, 2002.

[38]. **Lisong Pei, Jakob Schütte, Carlos Simon.** *Intrusion Detection Systems,* 2007-10-07.

[39]. **Jabez J, Muthukumar B.** *Intrusion Detection System (IDS) Anomaly Detection using outlier detection approach.* 2015, Procedia Computer Science.

[40]. **Bace, R. G., & Mell, P.** *Intrusion detection systems.* 2001.

[41]. **Zaidi, A.** *Recherche et détection des patterns d'attaques dans les réseaux IP à hauts débits (*Doctoral dissertation, Université d'Evry-Val d'Essonne). 2011.

[42]. **Kumar B, T. Raju, P. Ratnakar, M. Baba & Sudhakar N.** *Intrusion detection system-types and prevention.* 2013.

[43].Intrusion detection system for DoS attack in cloud**. Samani, Mishti D**., et al. 2016, International Journal of Applied Information Systems., Vol. 10.

[44]. **Eli David.** *Deep Learning for Dummies, Deep Instinct Special Edition,* 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey.

[45]. **Ian Goodfellow, Yoshua Bengio, Aaron Courville.** *Deep Learning, www.deeplearningbook.org*.

[46]. **Nikhil Buduma, Nicholas Locascio.** *Fundamentals of Deep Learning Designing Next-Generation Machine Intelligence Algorithms,* 2017, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[47]. **Nair, Vinod, and Geoffrey E. Hinton.** "Rectified Linear Units Improve Restricted Boltzmann Machines" Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010. Journal of Neurophysiology 20.4 (1957): 408-434.

[48]. **Alexander I. Galushkin.** *Neural Networks Theory,* Springer-Verlag Berlin Heidelberg 2007.

[49]. https://www.javatpoint.com/deep-learning-algorithms

[50]. **T. Mothilal.** *Python. Definition: History of Python.* July 2019.

[51]. https://domino.ai/data-science-dictionary/jupyter-notebook

[52]. **Samira Gholizadeh.** *Top Popular Python Libraries in Research*. 2022,J Robot Auto Res 3(2), 142-145.

[53]. **Hao, J., & Ho, T. K.** (2019). *Machine learning made easy: a review of scikit-learn package in python programming language.* Journal of Educational and Behavioral Statistics, 44(3), 348-361.

[54]. Google's Intro to TensorFlow. Available from: https://developers.google.com/machine-learning/crash-course/first-stepswith-tensorflow/toolkit.

[55]. **Karimi, Zohreh.** *Confusion Matrix.* 2021.

[56]. **Dalianis, Hercules.** (2018). *Evaluation Metrics and Evaluation.* 10.1007/978-3-319-78503-5_6.

[57]. **Ivan Vaccari, Giovanni Chiola, Maurizio Aiello, Maurizio Mongelli & Enrico Cambiaso**. *MQTTset, a New Dataset for Machine Learning Techniques on MQTT.* Intelligent and Adaptive Security in Internet of Things. 18 November 2020.