RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ 20 AOÛT 1955 SKIKDA, ALGÉRIE

FACULTÉ DES SCIENCES

Département d'Informatique

THÉSE DE DOCTORAT

Présenté pour l'obtention du diplôme de **DOCTORAT** de 3ème cycle

Spécialité : Informatique

Option : Réseaux et Systèmes distribués

Par :

Hayette Zeghida

Thème

Méthodes Intelligentes pour la Sécurisation de l'Internet des Objets (IoT)

Jury :

Président	Mazouzi Smaine	Professor	University 20 Août 1955, Skikda, Algeria
Examinateur	Boulehouache Soufiane	MCA	University 20 Août 1955, Skikda, Algeria
Examinateur	Brahimi Said	MCA	University 8 may 1945, Guelma, Algeria
Rapporteur	Mehdi Boulaiche	MCA	University 20 Août 1955, Skikda, Algeria
Co-Rapporteur	Ramadane Chikh	MCB	University 20 Août 1955, Skikda, Algeria

2025

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY 20 AOUT 1955 SKIKDA, ALGERIA
FACULTY OF SCIENCES
Department of Computer Sciences

DOCTORAL DISSERTATION

Presented for obtaining the 3rd cycle **DOCTORATE** diploma
Major : Computer Science
Option : Computer Networks and Distributed Systems

By :
Hayette Zeghida

Theme

Intelligents Methods for Securing the Internet of Things (IoT)

Jury :

President	Mazouzi Smaine	Professor	University 20 Août 1955, Skikda, Algeria
Examiner	Boulehouache Soufiane	MCA	University 20 Août 1955, Skikda, Algeria
Examiner	Brahimi Said	MCA	University 8 may 1945, Guelma, Algeria
Rapporteur	Mehdi Boulaiche	MCA	University 20 Août 1955, Skikda, Algeria
Co-Rapporteur	Ramadane Chikh	MCB	University 20 Août 1955, Skikda, Algeria

2025

Acknowledgements

In the Name of ALLAH, the Most Beneficent,
First and foremost, I am deeply grateful to Allah, the Most Magnificent and the Most Grateful, for His guidance and blessings, granting me the opportunity, strength, and perseverance to undertake and complete this PhD dissertation.
Many people have contributed to this work, too many to mention by name. Nevertheless, I would like to take this opportunity to express my heartfelt gratitude to my parents (my father and my mother, may ALLAH have mercy on her).
I want to express my heartfelt thanks to my husband, children, siblings, and family members for their unwavering support and encouragement throughout this journey.
I would like to express gratitude to my supervisors for their invaluable guidance and support throughout the completion of this work, my supervisor, Dr. Mehdi Boulaiche, and my co-supervisor, Dr. Ramdane Chikh.
I would like to thank Prof. Ahmed Patel, Prof. Ana Luiza Bessa Barros, Prof. Alwi M Bamhdi, and Dr.Djamel Zeghida for their help and support.
I am also thankful to my dear friends and colleagues for their encouragement.
I would like to thank the examiners.
Additionally, I extend my gratitude to all those who have contributed to this work, though they may not be individually mentioned.

Hayette Zeghida.

Dedication

I would like to dedicate this thesis to :

My **Mother, May ALLAH have mercy on her,**
My **Father**,
My **Husband** ,
My **Children,**
All My **Family.**

Hayette Zeghida.

Abstract

The Internet of Things (IoT) represents a broad network of interconnected devices, ranging from vehicles and household devices to various physical objects equipped with sensors, software, and connectivity. These devices are designed to gather, share, and respond to data, enabling smooth interactions with their surroundings, other systems, and people. By supporting intelligent and automated decision-making, IoT has brought transformative changes to areas like smart homes, urban infrastructure, industrial operations, and healthcare, fostering greater efficiency, innovation, and convenience.

To enable communication between devices, IoT leverages various protocols. Among these, the Message Queuing Telemetry Transport (MQTT) protocol plays a pivotal role in many IoT systems. MQTT is a lightweight, publish-subscribe messaging protocol specifically designed for efficient communication in resource-constrained environments. However, despite its widespread adoption, MQTT lacks robust built-in security features, rendering it vulnerable to a range of cyber threats, including unauthorized access, data interception, and denial-of-service attacks. These security concerns are particularly critical as IoT networks continue to expand and integrate into essential infrastructure, increasing their exposure to potential risks.

This thesis explores the IoT ecosystem with a focus on the MQTT protocol, addressing its security weaknesses and emphasizing the need for effective protective mechanisms. In particular, it proposes the integration of Intrusion Detection Systems (IDS) enhanced with Machine Learning (ML) and Deep Learning (DL) techniques as a robust solution for safeguarding MQTT-based IoT environments. Traditional security methods often fall short in such contexts due to the limited computational resources of IoT devices. Therefore, this research develops lightweight, data-driven IDS frameworks capable of real-time traffic analysis to detect anomalies and prevent intrusions.

By leveraging ML and DL algorithms—including advanced models such as Generative Adversarial Networks (GANs) the proposed IDS solutions aim to proactively identify suspicious behavior and mitigate emerging threats. The outcome is a resilient and intelligent security framework that strengthens the privacy, reliability, and trustworthiness of MQTT communication. Ultimately, this work contributes to advancing the state of IoT cybersecurity by delivering practical, adaptive, and scalable IDS strategies for protecting interconnected devices in an increasingly digital world.

Keywords: *Cyber-Attack, Cyber- Security, Deep Learning (DL), Generative Adversarial Network (GAN), Intrusion Detection System (IDS), Internet of Things (IoT), Message Queuing Telemetry Transport (MQTT), Machine Learning (ML).*

Résumé

L'Internet des objets (IoT) représente un vaste réseau d'appareils interconnectés, allant des véhicules et appareils électroménagers à divers objets physiques équipés de capteurs, de logiciels et de connectivité. Ces appareils sont conçus pour collecter, partager et répondre aux données, permettant ainsi des interactions fluides avec leur environnement, d'autres systèmes et des personnes. En prenant en charge une prise de décision intelligente et automatisée, l'IoT a apporté des changements transformateurs dans des domaines tels que les maisons intelligentes, les infrastructures urbaines, les opérations industrielles et les soins de santé, favorisant ainsi une plus grande efficacité, innovation et commodité.
Pour permettre la communication entre les appareils, l'IoT exploite divers protocoles. Parmi ceux-ci, le protocole Message Queuing Telemetry Transport (MQTT) joue un rôle central dans de nombreux systèmes IoT. MQTT est un protocole de messagerie léger de publication-abonnement spécialement conçu pour une communication efficace dans des environnements aux ressources limitées. Cependant, malgré son adoption généralisée, MQTT ne dispose pas de fonctionnalités de sécurité intégrées robustes, ce qui le rend vulnérable à toute une série de cybermenaces, notamment les accès non autorisés, l'interception de données , les attaques par déni de service et d'autres. Ces problèmes de sécurité sont particulièrement critiques à mesure que les réseaux IoT continuent de se développer et de s'intégrer dans les infrastructures essentielles, augmentant ainsi leur exposition aux risques potentiels.
Cette thèse explore l'écosystème IoT en se concentrant sur le protocole MQTT, en abordant ses faiblesses de sécurité et en soulignant la nécessité de mécanismes de protection efficaces. Elle propose notamment l'intégration de systèmes de détection d'intrusions (IDS) optimisés par des techniques d'apprentissage automatique (ML) et d'apprentissage profond (DL) comme solution robuste pour la protection des environnements IoT basés sur MQTT. Les méthodes de sécurité traditionnelles sont souvent inefficaces dans de tels contextes en raison des ressources de calcul limitées des appareils IoT. Par conséquent, cette recherche développe des infrastructures IDS légères et basées sur les données, capables d'analyser le trafic en temps réel pour détecter les anomalies et prévenir les intrusions.
En exploitant des algorithmes d'apprentissage automatique et d'apprentissage profond, notamment des modèles avancés tels que les réseaux antagonistes génératifs (GAN), les solutions IDS proposées visent à identifier proactivement les comportements suspects et à atténuer les menaces émergentes. Le résultat est une infrastructure de sécurité résiliente et intelligente qui renforce la confidentialité, la fiabilité et la fiabilité des communications MQTT. En fin de compte, ce travail contribue à faire progresser l'état de la cybersécurité de l'IoT en fournissant des stratégies IDS pratiques, adaptatives et évolutives pour protéger les appareils interconnectés dans un monde de plus en plus numérique.

Mots clés : *Cyberattaque, cybersécurité, apprentissage profond (DL), réseau antagoniste génératif (GAN), système de détection d'intrusion (IDS), Internet des objets (IoT), Message Queuing Telemetry Transport (MQTT), apprentissage automatique (ML).*

ملخص

يمثل إنترنت الأشياء IoT شبكة واسعة من الأجهزة المترابطة، بدءًا من المركبات والأجهزة المنزلية إلى مختلف الأشياء المادية المجهزة بأجهزة استشعار وبرامج وإمكانية الاتصال. تم تصميم هذه الأجهزة لجمع البيانات ومشاركتها والاستجابة لها، مما يتيح تفاعلات سلسة مع محيطها والأنظمة الأخرى والأشخاص. من خلال دعم اتخاذ القرارات الذكية والآلية، جلب إنترنت الأشياء تغييرات تحويلية في مجالات مثل المنازل الذكية والبنية التحتية الحضرية والعمليات الصناعية والرعاية الصحية، مما عزز الكفاءة والابتكار والراحة.

لتمكين الاتصال بين الأجهزة، يستفيد إنترنت الأشياء من بروتوكولات مختلفة. من بين هذه البروتوكولات، يلعب بروتوكول نقل بيانات طابور الرسائل MQTT دورًا محوريًا في العديد من أنظمة إنترنت الأشياء. MQTT هو بروتوكول مراسلة خفيف الوزن لنشر الاشتراك مصمم خصيصًا للاتصال الفعال في البيئات المحدودة الموارد. ومع ذلك، على الرغم من اعتماده على نطاق واسع، يفتقر MQTT إلى ميزات أمان مدمجة قوية، مما يجعله عرضة لمجموعة من التهديدات السيبرانية، بما في ذلك الوصول غير المصرح به واعتراض البيانات وهجمات رفض الخدمة وغيرها. تعتبر هذه المخاوف الأمنية بالغة الأهمية بشكل خاص مع استمرار شبكات إنترنت الأشياء في التوسع والتكامل مع البنية التحتية الأساسية، مما يزيد من تعرضها للمخاطر المحتملة.

تستكشف هذه الأطروحة منظومة إنترنت الأشياء مع التركيز على بروتوكول MQTT، ومعالجة نقاط ضعفه الأمنية، والتأكيد على الحاجة إلى آليات حماية فعالة. وتقترح، على وجه الخصوص، دمج أنظمة كشف التسلل (IDS) المُعززة بتقنيات التعلم الآلي (ML) والتعلم العميق (DL) كحلٍّ فعّال لحماية بيئات إنترنت الأشياء القائمة على MQTT .غالبًا ما تفشل أساليب الأمن التقليدية في مثل هذه السياقات بسبب محدودية الموارد الحاسوبية لأجهزة إنترنت الأشياء. لذلك، يطور هذا البحث أطر عمل خفيفة الوزن لأنظمة كشف التسلل، قائمة على البيانات، وقادرة على تحليل حركة المرور في الوقت الفعلي للكشف عن الشذوذ ومنع التسلل.

من خلال الاستفادة من خوارزميات التعلم الآلي والتعلم العميق - بما في ذلك النماذج المتقدمة مثل شبكات التوليد الخصومة (GANs) ،تهدف حلول أنظمة كشف التسلل المقترحة إلى تحديد السلوك المشبوه بشكل استباقي والحد من التهديدات الناشئة. النتيجة هي إطار عمل أمني مرن وذكي يعزز خصوصية وموثوقية وموثوقية اتصالات MQTT .في نهاية المطاف، يُساهم هذا العمل في الارتقاء بمستوى الأمن السيبراني لإنترنت الأشياء من خلال تقديم استراتيجيات كشف تسلل عملية وقابلة للتكيف وقابلة للتطوير لحماية الأجهزة المترابطة في عالم رقمي متزايد.

كلمات مفتاحية: الهجوم السيبرانيٍ، الأمن السيبراني، التعلم العميق *DL*، شبكة الخصومة التوليدية *GAN*،نظام كشف التسلل *IDS*، إنترنت الأشياء *IoT* ، نقل القياس عن بعد في قائمة انتظار الرسائل *MQTT*، التعلم الآلي *ML*.

Contents

List of Figures

List of Tables

List of Abbreviations

AAA	Authentication, Authorization, and Accountability
AAE	Adversarial Auto-Encoder
ACC	Accuracy
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
ANOVA	ANalysis Of VAriance
AR	Association Rule
ARP	Address Resolution Protocol
AUC	Area Under the ROC Curve
CIA	Confidentiality, Integrity, and Availability
CL-GAN	Conditional Generative Adversarial Network
CNN	Convolutional neural networks
CPS	Cyber-Physical System
D	Discriminator
DAE	Deep Autoencoder
DCPS	Data-Centric Publish-Subscribe
DL	Deep Learning
DoS	Denial of Service
DDoS	Distributed Denial of Service
DDS	Data Distribution Service
DLRL	Data-Local Reconstruction Layer
DNS	Domain Name System
DT	Decision tree
DTLS	Datagram Transport Layer Security
EL	Ensemble Learning
EML	Elite Machine Learning algorithms
F1-S	F1 Score
FDS	Flooding Denial of Service
FN	False Negative
FP	False Positive
G	Generator
GA	Genetic Algorithm
GAN	Generative Adversarial Networks
GBM	Gradient Boosting Machine
GRU	Gated Recurrent Units
HTTP	Hyper Text Transfer Protocol

IDS	Intrusion Detection System
IoT	Internet of Things
ITU	International Telecommunication Union
IT	Information Technology
IERC	European Research Cluster on the Internet-of-Things
KNN	K-Nearest Neighbor
LAN	Local Area Network
LDA	Linear Discriminant Analysis
LR	Logistic Regression
LSTM	Long Short-Term Memory
M2M	Machine-to-Machine
MCC	Matthews correlation coefficient
MITM	Man-In-The-Middle
ML	Machine Learning
MLP	Multilayer Perceptrons
MQTT	Message Queuing Telemetry Transport
NB	Naive Bayes
NIDS	Network Intrusion Detection Systems
NN	Neural Network
NPL	Natural Language Processing
UDP	User Datagram Protocol
OASIS	Organization for the Advancement of Structured Information Standards
PREC	Precision
RBF	Radial Basis Function
REC	Recall
RF	Random Forest
RNN	Recurrent Neural Network
ROC	A Receiver Operating Characteristic
SDN	Software-Defined Networking
SMO	Sequential Minimal Optimization
SMOTE	Synthetic Minority Over-sampling Technique
SPEC	Specificity
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
WSN	Wireless sensor network
XGBoost	eXtreme Gradient Boosting

List of Contributions

1. Zeghida, H., Boulaiche, M., & Chikh, R. (2023). Securing MQTT protocol for IoT environment using IDS based on ensemble learning. *International Journal of Information Security*, *22*(4), 1075-1086.

2. Zeghida, H., Boulaiche, M., & Chikh, R. (2023, May). Detection of DoS Attacks in MQTT Environment. In *International Conference on Intelligent Systems and Pattern Recognition* (pp. 129-140). Cham: Springer Nature Switzerland.

3. Zeghida, H., Boulaiche, M., Chikh, R., Bamhdi, A. M., Barros, A. L. B., Zeghida, D., & Patel, A. (2025). Enhancing IoT cyber attacks intrusion detection through GAN-based data augmentation and hybrid deep learning models for MQTT network protocol cyber attacks. *Cluster Computing*, *28*(1), 58.

General Introduction

The Internet of Things (IoT) has revolutionized modern technology by enabling extraordinary connectivity between everyday objects. Initially designed to improve Machine-to-Machine (M2M) communication, IoT has quickly become essential to various industries, including industrial automation, healthcare, smart cities, and transportation. The basic idea behind IoT is straightforward yet transformative: linking a broad range of devices - from sensors and wearables to vehicles and home devices - through the internet, allowing them to exchange data and communicate in real-time. This connectivity improves efficiency, innovation, and convenience by facilitating autonomous decision-making and enhancing human-machine interactions [8].

The origins of IoT can be traced back to the early 1980s when the concept of a "smart" object network began to take form. One of the first practical examples of IoT was the development of an Internet-connected Coca-Cola vending machine at Carnegie Mellon University in Pennsylvania, United States, in 1982, which provided real-time information on the availability of drinks [9]. However, it was not until the early 2000s that IoT gained significant momentum, driven by the availability of affordable sensors, advances in wireless communication, and the widespread adoption of the internet. The notion of "ubiquitous computing" proposed by Mark Weiser in the 1990s also laid the intellectual foundation for the IoT by advocating for the seamless integration of computing devices into everyday objects.

Today, IoT networks are made up of a wide range of interconnected smart devices, from basic sensors to advanced embedded systems. These devices communicate using various protocols, with the Message Queuing Telemetry Transport (MQTT) protocol becoming one of the most widely used in IoT applications. MQTT is a lightweight, publish-subscribe messaging protocol designed for efficient, real-time communication in environments with limited bandwidth and resources. Its simplicity and effectiveness make it ideal for devices with limited processing power, memory, or network capacity, making it a popular choice for IoT systems in different sectors [10].

The rapid growth of IoT devices and the widespread use of MQTT for communication have created significant security challenges. MQTT, designed to optimize performance and minimize resource use, lacks strong built-in security features, making it vulnerable to cyber threats. These vulnerabilities threaten the confidentiality, integrity, and availability of data within IoT networks. As IoT continues to expand to critical infrastructure, ensuring the security of MQTT-based communication has become an essential concern for developers and users alike [11].

The need for securing MQTT in IoT environments is further exacerbated by the resource constraints inherent in many IoT devices. Traditional security mechanisms, such as encryption and authentication protocols, often incur high computational costs that are incompatible with the limited processing power and memory available on IoT devices. One promising

approach involves using Machine Learning (ML) and Deep Learning (DL) models in Intrusion Detection Systems (IDS) to detect abnormal behavior in IoT traffic in real-time. These models offer additional protection by continuously monitoring networks for signs of malicious activities [12].
This thesis examines the security of the MQTT protocol within IoT ecosystems, investigating how ML and DL-based methods can be utilized to identify and address security threats. It is guided by the central question: **How can IoT communication protocols be secured against the growing number of threats?**
The objective of this research is to contribute to the creation of more robust and secure IoT networks, ensuring the protection of data privacy and reliability in the communication between devices, particularly in the face of evolving cyber risks. By applying advanced intrusion detection techniques and strengthening the MQTT security framework, this study aims to offer valuable insights into securing the core of IoT communication, thereby enabling the safe and efficient realization of the benefits of this transformative technology.

1. **Thesis Contributions** This thesis presents significant contributions to advancing the security of the MQTT protocol within IoT ecosystems. By leveraging state-of-the-art methodologies and focusing on real-world challenges, the research addresses critical gaps in IoT security and proposes innovative solutions. The key contributions are as follows:

    - **Development of a Balanced Dataset for IoT Intrusion Detection:** A novel, balanced binary dataset is generated from the publicly available MQTTset, which specializes in IoT systems. This dataset undergoes extensive preprocessing, including scaling, encoding, handling missing values, and feature reduction—processes often overlooked in existing research but essential for improving model performance. This preprocessing ensures that the dataset is not only balanced but also optimized for effective model training.
    - **Improved Intrusion Detection through Ensemble Learning:** The research demonstrates the efficacy of ensemble learning techniques in generating classification models from the preprocessed and balanced dataset. Experimental results show that these ensemble methods outperform single learning techniques, leading to improved intrusion detection system efficiency in MQTT-based IoT environments.
    - **Innovative Data Augmentation with Generative Adversarial Networks (GANs):** To address class imbalance in the dataset, GANs are employed as an unsupervised learning technique for data augmentation. This ensures a more balanced distribution across dataset classes, enhancing the IDS's ability to detect threats across diverse attack scenarios.
    - **Design and Implementation of an ML and DL-Based IDS for MQTT Security:** The research introduces and implements novel IDSs leveraging ML and DL models to enhance the security of MQTT-based IoT networks. These models are employed to proactively detect and mitigate security threats by identifying anomalous behavior within IoT traffic.
    - **Comprehensive Evaluation and Validation:**The research includes a thorough evaluation of the proposed methods using IoT scenarios and datasets. Perfor-

mance metrics such as accuracy and efficiency or others are assessed to validate the effectiveness of the solutions in enhancing MQTT security.

- **Contribution to Knowledge and Future Research Directions:** This research delivers a comprehensive framework to enhance the reliability and security of MQTT communication in IoT networks. It provides actionable insights for researchers while establishing a foundation for future advancements in safeguarding the backbone of IoT systems through innovative and resource-efficient methods.

2. **Thesis Organization** Apart from the General Introduction, this thesis is structured into six chapters and a general conclusion, each addressing a critical aspect of IoT security and MQTT communication challenges. The organization is as follows:

   - **Chapter 1: Internet of Things and MQTT Protocol: State of the Art**: This chapter provides an overview of the IoT and its foundational aspects, including its definition, architecture, and associated security challenges across its three layers. The chapter also explores IoT attacks (active and passive) and discusses application-layer protocols with a focus on the MQTT protocol. It concludes with an analysis of MQTT vulnerabilities and publicly available datasets.
   - **Chapter 2: Machine Learning and Deep Learning Models:** The second chapter delves into the fundamentals of ML and DL models. It outlines ML techniques, applications, and models, including ensemble learning approaches, followed by an introduction to DL and its applications in IoT security. Additionally, it discusses classifier evaluation metrics, providing a foundation for the practical implementation of these techniques in later sections.
   - **Chapter 3: ML and DL Techniques for Cybersecurity in MQTT Environments: A Review of Related Work :** This chapter reviews the existing literature on intrusion detection systems that utilize ML and DL or GAN techniques to secure MQTT-based IoT environments.
   - **Chapter 4: Securing MQTT Protocol for IoT Environments Using IDS Based on Ensemble Learning (First Contribution):** This chapter introduces the first major contribution of this thesis; the development of an IDS using ensemble learning models. It details the preprocessing techniques applied to the MQTT dataset, including features engineering and data balancing. The proposed approach is evaluated through experiments, demonstrating its effectiveness in enhancing IDS performance compared to single learning models.
   - **Chapter 5: Detection of DoS Attacks in the MQTT Environment (Second Contribution):**The fifth chapter focuses on detecting denial-of-service (DoS) attacks in MQTT-based IoT systems. It outlines the methodology, including preprocessing steps and proposed detection models. Experimental results showcase the efficacy of the approach in mitigating DoS attacks while addressing resource constraints.
   - **Chapter 6: Hybrid Deep Learning IDS Based on Generative Adversarial Network for Detecting Cyberattacks Against the MQTT Protocol (Third Contribution):** The final contribution of this thesis is presented in this chapter, which introduces a hybrid IDS combining deep learning models and

GAN-based data augmentation. This chapter highlights the use of GANs to generate balanced datasets and discusses the proposed framework's ability to detect cyberattacks with high accuracy. Comparative analyses against existing works and validation results are provided to demonstrate the framework's superiority.

- **General Conclusion:** The thesis concludes by summarizing the key contributions, emphasizing the practical insights and innovations introduced in securing MQTT communication. It also highlights directions for future research in addressing emerging IoT security challenges.

Part 1 : State of Art

Chapter 1

Internet of Things and MQTT Protocol

1.1 Introduction

The IoT has quickly evolved as a transformational technology, changing the way people interact with the physical world through connected devices and smart systems. IoT enables seamless data exchange between sensors, actuators, and computing systems, facilitating applications across various industries and enabling a wide range of applications across industries. As the number of connected devices increases dramatically, so do the challenges associated with managing, securing, and optimizing these networks.

One of the key enablers of IoT is the communication protocol, which ensures that devices can efficiently exchange information in real time. Among the various protocols developed for this purpose, the MQTT protocol has gained significant traction due to its lightweight nature and suitability for resource-constrained environments. Originally designed for remote monitoring in the oil and gas industry [13]. MQTT has evolved into a cornerstone of IoT communications, particularly in scenarios where bandwidth, power consumption, and network reliability are critical concerns.

This chapter provides a comprehensive overview of the IoT landscape, starting with the fundamental definitions and security considerations that are crucial to understanding the ecosystem. We then dive into the layered architecture of the IoT, highlighting the security challenges in the perception, network, and application layers. As cyberattacks increasingly target IoT systems, we explore both passive and active attacks that pose significant threats to IoT deployments.

Subsequently, the chapter shifts focus to the protocols that operate at the IoT application layer, with particular emphasis on the MQTT protocol. We will examine MQTT's architecture and its vulnerabilities. The discussion is further enriched by the presentation of commonly used datasets for evaluating MQTT security solutions.

This chapter aims to equip readers with a detailed understanding of the current state of IoT and MQTT, laying the foundation for exploring advanced security strategies and applications in subsequent sections.

1.2 IoT Definition

The term "Internet of Things", or "IoT" for short was coined by Kevin Ashton in 1999 [8], refers to a network of interconnected cyber-physical systems that are linked to the internet and possess the ability to sense and interact with the physical environment autonomously. This concept involves connecting everyday objects and devices to the internet, allowing seamless communication and data sharing with minimal human intervention. The IoT envisions a dynamic global network consisting of both physical and virtual entities - such as sensors, people, software, and various devices - that communicate through unique addressing protocols to provide a wide range of services. The evolution of IoT, driven by advancements in IPv6 technology, aims to overcome the limitations of IPv4, transforming the internet and facilitating the connection of potentially 70 billion or more smart devices. This widespread adoption of IoT is often described as the "Second Economy" or the "Industrial Internet revolution" [14].
In 2005, the International Telecommunication Union (ITU) expanded the concept of IoT and proposed four key technologies necessary for its realization: intelligent embedded technology, Radio Frequency IDentification (RFID), sensor technology, and nanotechnology [15]. Although IoT has made significant progress, it is still in its early stages, and a universally accepted standard for its full implementation remains elusive [16]. Tan & Sidhu [17] describe IoT as a computing environment comprising multiple RFID-embedded entities that communicate to deliver intelligent Information Technology (IT) services. According to IERC; the European Research Cluster on the internet of things, IoT is a global network infrastructure that is self-configuring (Figure. 1.1), relies on interoperable communication protocols, and integrates physical and virtual 'things' with identities, attributes, and virtual personalities into the information network [18].

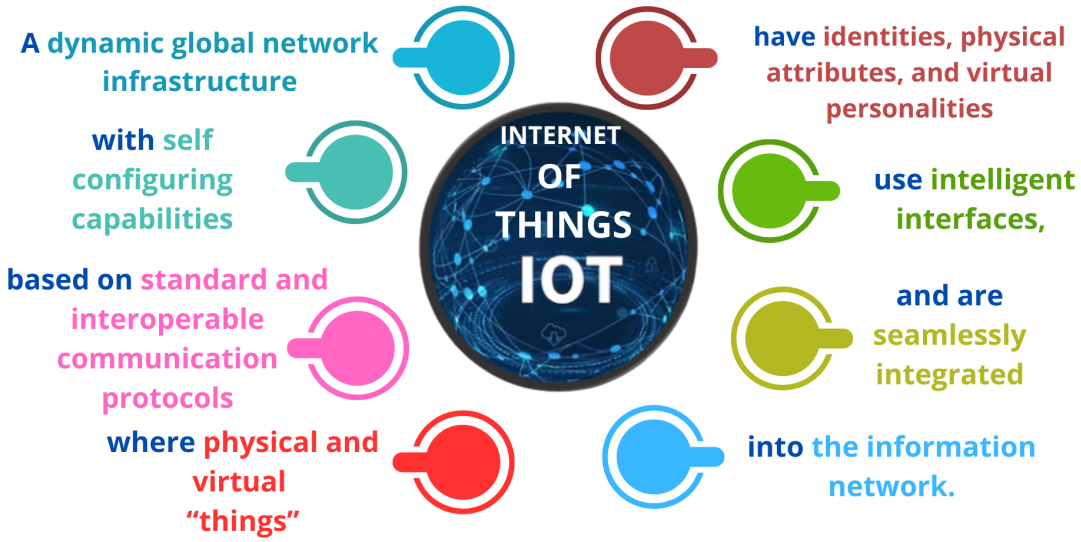


Figure 1.1: IoT definition

A critical aspect of IoT is the correlation between "things" in both the cyber and physical realms [15] as Figure 1.2 shows. These "things" can be classified into two categories: physical

and cyber. Physical things refer to tangible objects like people, vehicles, or devices, as well as their associated behaviors (e.g., running, driving). Events in the physical world are shaped by the actions and interactions of these objects, such as a vehicle remaining stationary in a parking space due to traffic conditions or weather. Cyber things, in contrast, are abstract entities existing in the digital space, including code and data. Actions in the cyber domain involve data processing, such as information transmission, and services refer to tasks that are either offered by or to a thing to achieve specific goals. This interplay between physical things and cyber things exemplifies the seamless connectivity and functionality within the IoT ecosystem.

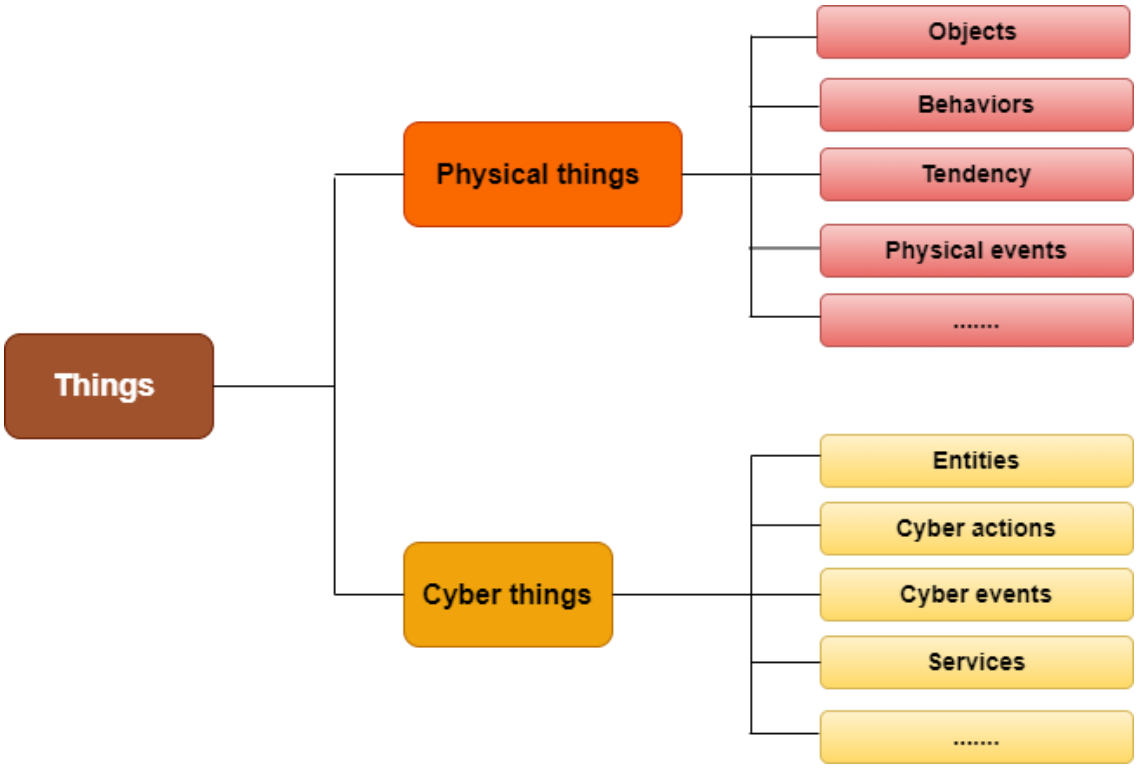


Figure 1.2: Classification of physical and cyber things

1.3 IoT Security Considerations

The IoT holds transformative potential across various sectors, yet its communication infrastructure is plagued by inherent security vulnerabilities. The rapid proliferation of IoT devices exacerbates these challenges, highlighting the urgent need for robust security measures to protect networks from cyber threats. Security in IoT systems is essential at every architectural level, made more complex by the blurred network boundaries typical of IoT configurations. Key security considerations fall into four domains: network security, which safeguards communication channels and data confidentiality; application security, which fortifies IoT software against unauthorized access and data breaches; endpoint security, which protects devices from unauthorized manipulation and ensures operational integrity; and data security, which focuses on protecting sensitive information throughout its lifecycle, with a strong emphasis on privacy [19]. These challenges are further compounded by IoT networks' diverse and dynamic nature, where nodes vary in capabilities, protocols, and security require-

ments, making secure connections difficult to achieve. Ensuring data safety in IoT systems hinges fundamentally on upholding the CIA triad - Confidentiality, Integrity, and Availability -. These security traits are crucial for the secure operation of the network. Complementing this, a comprehensive set of security controls, collectively called AAA, comprising Authentication, Authorization, and Accountability, plays a crucial role in protecting the system's CIA features. These measures are indispensable in strengthening the principles and safeguarding the key aspects of the CIA within the system. CIA and AAA (Figure 1.3) collectively create a robust security posture essential for the effective and safe operation of IoT systems [20, 21].

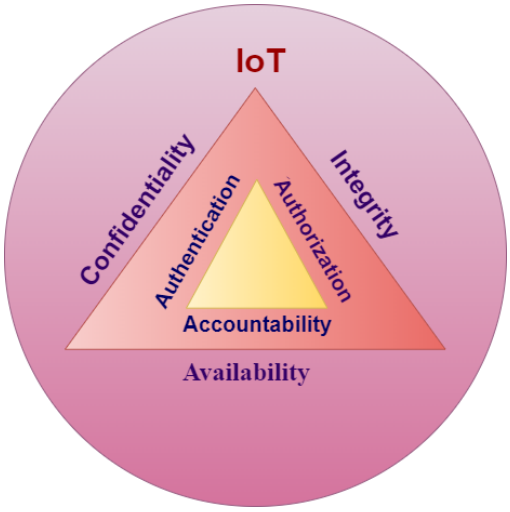


Figure 1.3: IoT security considerations

- **Authentication:** This is the process of confirming the legitimacy of the communication's source.
- **Authorization:** Ensures that services and resources are accessed solely by authorized devices or users.
- **Accountability:** This entails taking responsibility for actions and duties, as well as implementing security policies. In cases of repudiation, accountability procedures help identify the responsible party.
- **Confidentiality:** This involves ensuring discretion and safeguarding data secrecy from unauthorized third parties.
- **Integrity:** Preserving the integrity of information requires the recipient to authenticate that the received communications remain unaltered during delivery or broadcast.
- **Availability:** Refers to the ability to access information or resources in an accurate and reliable form. Ensures that authorized users can use various IoT services to counteract DoS attacks, safeguarding uninterrupted access.

1.4 IoT Architecture

Integrating diverse networks in the IoT presents significant challenges in establishing reliable connections between dynamic and constantly evolving nodes. As depicted in Figure 1.4,

the IoT architecture is broadly divided into three layers: the perception (sensing) layer, the network (transportation) layer, and the application layer. Each layer addresses specific functionalities and security concerns, showcasing a layered approach designed to tackle the complex security challenges inherent in the IoT ecosystem.

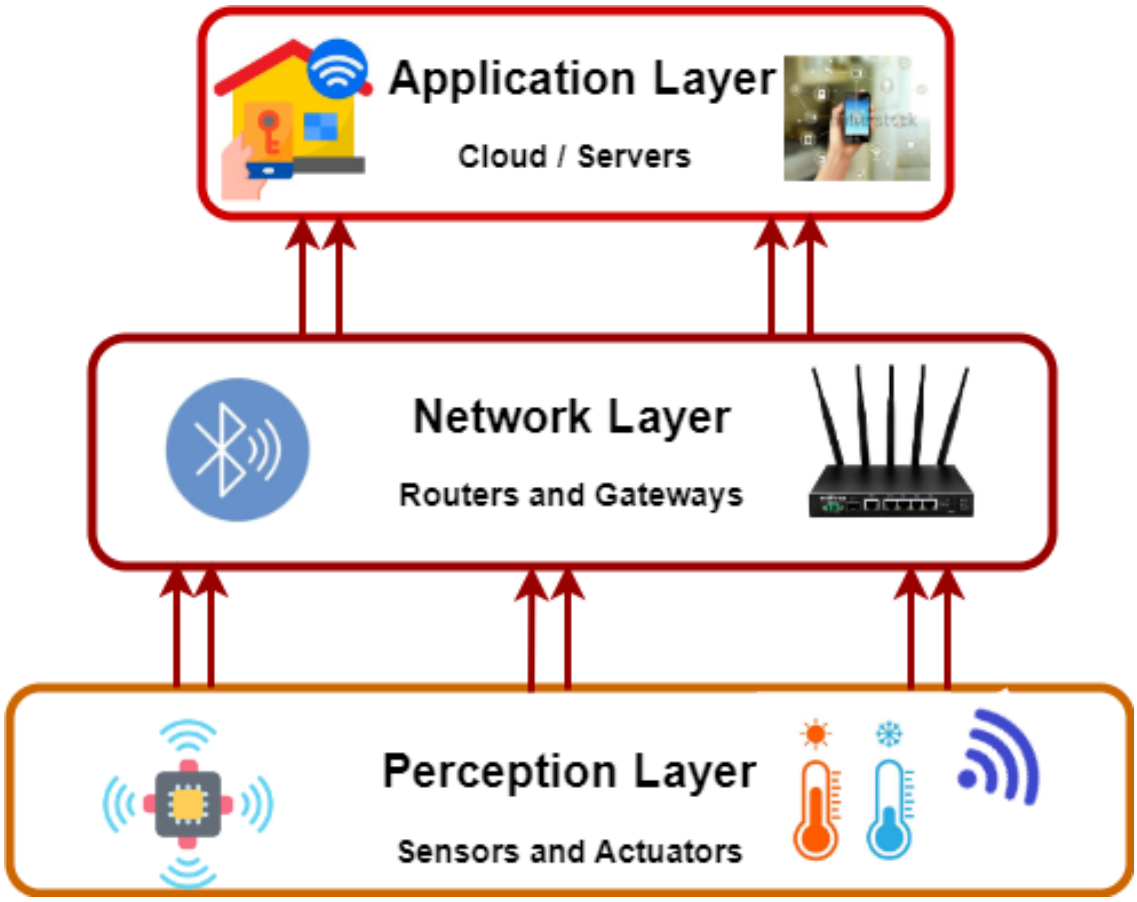


Figure 1.4: The Three-layer IoT architecture

This section discusses the three layers and some of the most prominent security concerns currently faced by the IoT scene in each layer.

1.4.1 Perception Layer

The perception layer, also known as the sensing layer, acts as the foundation of the IoT architecture, connecting the physical and digital worlds. It consists of diverse devices, such as sensors, RFID tags, readers, smart cards, and other data-gathering instruments, all designed to acquire and send real-world information. This layer detects environmental changes, collects real-time data, and sends them to upper layers for processing and decision-making. It offers a wide range of applications, including temperature, humidity, location, and motion tracking, allowing for smooth integration into smart settings [22].

1.4.2 Security Issues at the Perception Layer

Sensor nodes play a crucial role in the IoT ecosystem, responsible for information transmission, collaboration, and integration tasks. , the security challenges within sensor networks are intricate and multifaceted, given their battery-operated nature and limited security defenses. These nodes are particularly vulnerable to malicious code, such as self-contained worms, which infiltrate networks without needing other files, making them difficult to detect and mitigate. Another critical issue is the exploitation of tag defects in RFID systems, where insufficient security measures enable intruders to bypass authentication, forge credentials, and access sensitive data. Strengthening security at this layer is essential to protect data integrity and ensure the reliable operation of IoT ecosystems [19].

1.4.3 Network Layer

The network layer in the IoT is a pivotal component encompassing computers and both wired and wireless networks. This layer's primary function is to facilitate data transmission within the IoT infrastructure. Particularly in the case of wireless networks, nodes exhibit a high degree of mobility, allowing them to connect or disconnect from the network without prior confirmation. While this flexibility is a fundamental aspect of wireless networks, it simultaneously exposes them to heightened security threats. The dynamic nature of node mobility raises concerns about potential security breaches, emphasizing the need for robust security measures at the network layer [22].

1.4.4 Security Issues at the Network Layer

The network layer is essential for the security and operation of IoT systems, managing the transmission and processing of large amounts of data from various devices. It handles data broadcasting and multicasting while ensuring data integrity and privacy. However, security concerns such as unauthorized access, hacking, and DoS attacks are prevalent, especially given the limited processing power and memory of IoT devices. These vulnerabilities are particularly concerning in sensitive applications like smart cities, where data related to traffic management, energy distribution, and public safety must be protected. To address these risks, advanced cryptographic techniques are recommended to enhance data security and privacy, ensuring confidentiality, integrity, and user authentication [19], [23]. Safeguarding these aspects is crucial as IoT networks become increasingly integrated into essential services.

1.4.5 Application Layer

The application layer in an IoT system is key to facilitating intelligent processing and decision-making. It manages device connectivity, identification, and control, utilizing technologies like cloud computing to process data and execute smart actions as needed. This layer underpins various applications, from industrial monitoring to smart grids and advanced systems, making it a cornerstone of IoT operations. However, its broad scope and varied uses make it vulnerable to risks such as malicious software and data congestion, potentially affecting system safety and reliability [19].

1.4.6 Security Issues at the Application Layer

The application layer in IoT plays a vital role in enabling user interaction with diverse services, from smart homes to critical industrial systems. However, its complexity and exposure to external threats make it highly susceptible to security challenges. Malicious software, such as malware or ransomware, poses significant risks, potentially compromising sensitive data and service availability. Furthermore, the integration of numerous applications and devices places a growing demand on QoS, as many services rely on frequent, small data transfers for upgrades and synchronization. These sessions can increase network latency and accessibility issues if not securely managed, leading to performance bottlenecks and vulnerabilities to attacks such as DoS attacks [24]. Ensuring robust security at this layer requires advanced encryption, rigorous authentication protocols, and efficient management of network capacity to maintain seamless communication and safeguard information integrity. Addressing these challenges is essential to ensure the reliability and secure operation of interconnected IoT devices and applications.

1.5 IoT Attacks

Understanding technical terminology related to IoT security is essential for tackling the challenges posed by these threats. This knowledge helps individuals and organizations comprehend the complexities of securing IoT devices, enabling the development of structured frameworks to categorize and mitigate attacks. Such understanding also facilitates agile responses to evolving threats, empowering stakeholders to protect connected devices and their data effectively.

IoT security threats can be broadly categorized into **physical** and **cyber** attacks. Physical attacks directly damage IoT devices, such as mobiles, cameras, sensors, and routers, causing service disruption through hardware compromise. Cyber-attacks, which exploit digital network vulnerabilities, manipulate user data via methods like eavesdropping, data dropping, tampering, or destruction [25]. These attacks are further categorized into **passive attacks**, which involve unauthorized data access without altering its content, and **active attacks**, which involve intentional manipulation or disruption of data transmission, affecting device functionality and security.

For a visual summary, Figure 1.5 outlines the types of attacks targeting IoT devices. Additionally, common IoT-specific attacks are discussed in the following sections.

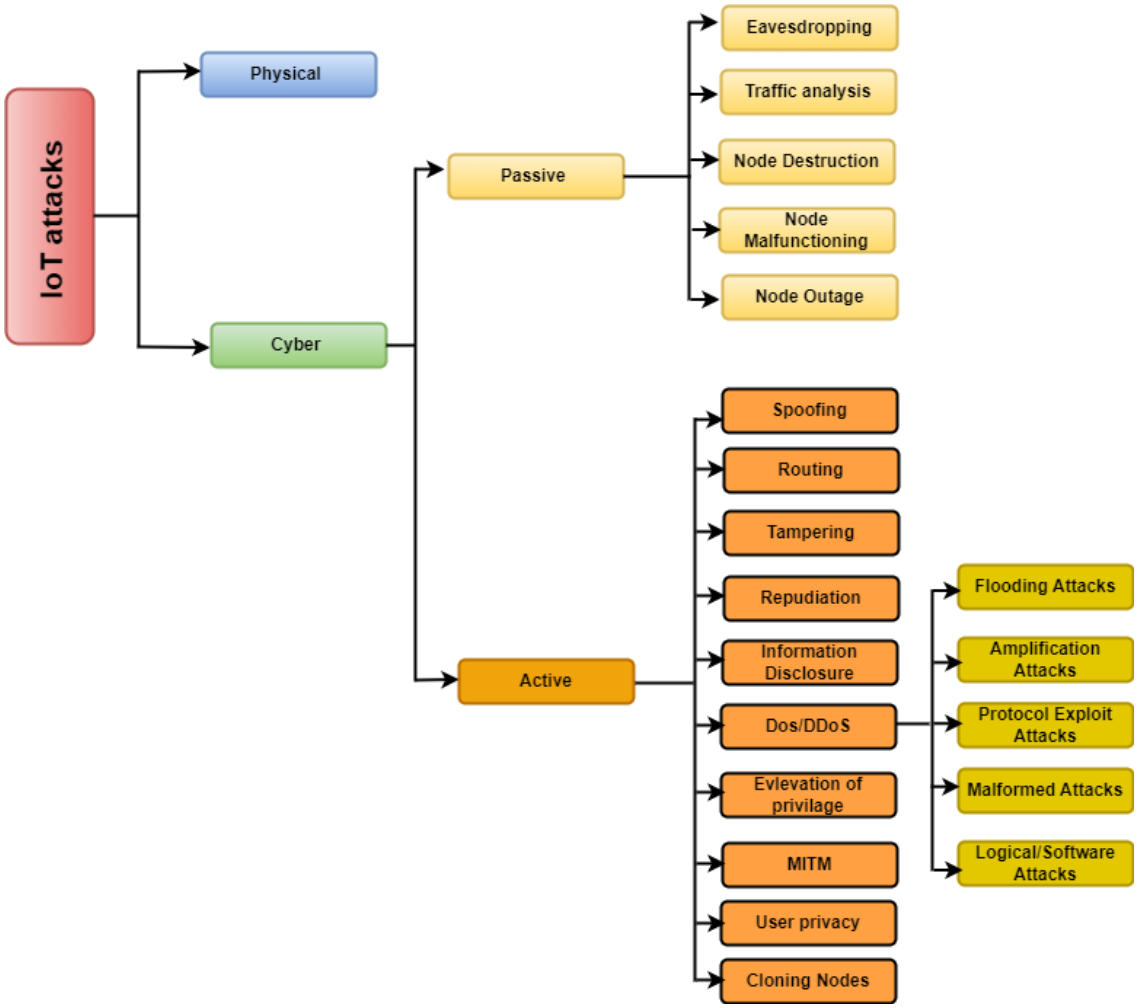


Figure 1.5: Taxonomy of IoT attacks

1.5.1 Passive attacks

Passive IoT attacks are stealthy threats where adversaries secretly gather information from users or devices without their knowledge or consent, prioritizing data confidentiality. These attacks are difficult to detect as they emit no radio signals and do not interfere with the network's normal functions. Common forms include eavesdropping, traffic analysis, and exploiting vulnerabilities like node malfunction, tampering, destruction, or outages [26].

1.5.1.1 Eavesdropping

Eavesdropping is a covert attack in which communication data is intercepted without authorization. It exploits vulnerabilities in the communication infrastructure to access sensitive information, such as node locations, message identities, timestamps, and unencrypted data, which can be misused or sold, causing significant harm to victims [27].

1.5.1.2 Traffic analysis

Traffic analysis attacks involve examining network traffic patterns, such as packet timing, size, and source, to infer sensitive information without accessing data content. These attacks

can expose communication patterns, entity relationships, and vulnerabilities, compromising security and privacy. Countermeasures include encryption and traffic obfuscation [26].

1.5.1.3 Node Destruction

Node destruction involves disabling a network node through physical means, such as electrical surges or physical force, disrupting network operations by removing a critical component of the communication infrastructure [28].

1.5.1.4 Node Malfunctioning

Node malfunctioning occurs when a network node experiences operational issues triggered by diverse factors such as malfunctioning sensors, energy depletion, or other DoS attacks. This passive attack disrupts the node's functionality, impacting overall network performance [28].

1.5.1.5 Node Outage

Node outage occurs when a node fails to perform its functions, disrupting network operations. In cases like the failure of a cluster head in a heterogeneous network, robust protocols are needed to mitigate effects, such as electing new cluster heads or rerouting network communication paths [25].

1.5.2 Active attacks

Active attacks involve an adversary accessing a system to introduce or modify data. These attacks compromise IoT device security through methods such as intervention, interruption, and modification. Common types include spoofing, routing attacks, tampering, repudiation, information disclosure, DoS/DDoS (Distributed Denial of Service), elevation of privilege, Man-in-the-Middle attacks, user privacy, and cloning nodes [29].

1.5.2.1 Spoofing

Spoofing [30],[31], also known as impersonation attacks, is a tactic where attackers obtain authentication credentials to gain unauthorized access, often via theft, eavesdropping, or phishing. It threatens IoT security through methods like IP spoofing, which manipulates IP headers to conceal identities and launch DDoS attacks; ARP (Address Resolution Protocol) spoofing, which exploits ARP vulnerabilities to spread fake messages in a Local Area Network (LAN); and DNS (Domain Name System) spoofing, which alters DNS server configurations to redirect data and potentially access or manipulate sensitive information within the IoT network.

1.5.2.2 Routing

Routing attacks exploit vulnerabilities in routing protocols to manipulate or replay routing information, leading to incorrect network traffic [32]. Examples include sinkhole attacks, where a malicious node presents a false optimal route; selective forwarding, where malicious nodes drop genuine packets; and blackhole attacks, where nodes appear as the best routes but discard data [33], [34]. Wormhole attacks involve two malicious nodes tunneling packets

to present a shorter route falsely. Replay attacks involve retransmitting or delaying data to gain unauthorized access [30]. These attacks emphasize the need for strong security measures in IoT networks.

1.5.2.3 Tampering

Tampering attacks in IoT include device tampering and data tampering. Device tampering involves the theft or misuse of unattended IoT devices, while data tampering refers to altering stored or transmitted data, threatening its integrity. Both emphasize critical security challenges in IoT, emphasizing the importance of robust protections against unauthorized access and modification [30].

1.5.2.4 Repudiation

Repudiation refers to situations where devices carry out malicious actions and then deny their involvement, often leaving no trace for identification. For example, a device might spread a virus across a network without any evidence linking it to the attack. This type of attack is particularly threatening to systems that cannot track and record unauthorized actions, making it difficult to attribute and hold devices accountable. To address this challenge, robust monitoring and recording mechanisms are essential to maintain network integrity and security, especially in IoT environments. [35].

1.5.2.5 Information Disclosure

Information disclosure involves unauthorized access to information by attaching snooping devices, intercepting network channels, or physically accessing devices [30]. Techniques like the Probe method [36],[37] exemplify this, where attackers scan connections (e.g., via port scanning) to identify vulnerabilities. Beyond unauthorized access, information disclosure includes the leakage of sensitive data, such as in side-channel attacks [38], which exploit indirect channels like power usage or electromagnetic signals to extract information.

1.5.2.6 Dos/DDoS

Unlike DoS attacks, which are launched from a single source, Distributed Denial of Service or DDoS attacks, as outlined by Sonar and Upadhyay [39], involve numerous compromised devices (often botnets) flooding a target system with excessive traffic from various locations. This makes DDoS attacks more damaging and harder to counter than single-source DoS attacks. Kolias et al. [40] point out that DDoS attacks are prevalent in IoT, particularly in social IoT contexts like smart cities, where their distributed nature poses significant challenges. As noted by Zlomislić et al. [41] and Prabadevi & Jeyanthi [42], DDoS attacks employ various tactics, each designed to exploit system vulnerabilities and overwhelm the target. These types of DDoS attacks are as follows:

- **Flooding Attacks:** This category involves bombarding the victim's system with a massive influx of packets, predominantly UDP (User Datagram Protocol) or ICMP (Internet Control Message Protocol) packets, depleting network bandwidth. Botnets are often used to amplify their effect, making them highly disruptive.

- **Amplification Attacks:** These exploit reflection mechanisms and IP source spoofing to overwhelm a victim by redirecting response packets from reflector servers. Examples include smurf and fraggle attacks [42], as well as DNS and SSDP amplification attacks [43].
- **Protocol Exploit Attacks:** This category exploits vulnerabilities in different protocols to transform small queries into a massive number of requests, thereby slowing down or crashing the victim's server(s). Examples include SYN flood, TCP reset, and water torture attacks.
- **Malformed Attacks:** These attacks rely on the use of malformed network packets, such as utilizing the same IP address for both source and destination addresses [42].
- **Logical/Software Attacks:** This category pertains to attacks on application protocols. The "Ping of Death" [39], for instance, involves sending a simple fragmented ICMP request packet larger than the maximum IP packet size, causing the victim to fail in reassembling it. In the Teardrop attack, the adversary sends two fragments that do not reassemble based on the offset value of the packet.

1.5.2.7 Elevation of Privilege

Elevation of privilege attacks involve gaining unauthorized access or escalating existing privileges to compromise a device or service, posing serious security risks when attackers are treated as trusted entities [30]. These attacks primarily take two forms: User-to-Root (U2R) and Remote-to-Local (R2L). In U2R attacks, an attacker with a standard user account escalates privileges to gain root (superuser) access, bypassing normal restrictions and potentially compromising system security. In contrast, R2L attacks occur when an attacker without any initial account exploits vulnerabilities to obtain local user access, often through password guessing or exploiting system weaknesses [36].

1.5.2.8 Man-In-The-Middle

A Man-In-The-Middle (MITM) attack involves an attacker intercepting and potentially altering communications between two systems without detection, compromising the integrity and confidentiality of the exchange [30], [31]. Common forms include Address Resolution Protocol (ARP) Cache poisoning, where the attacker associates their MAC address with a legitimate IP address to intercept communication; DNS spoofing, which redirects victims to malicious sites by altering DNS responses; and session hijacking, where the attacker takes control of an established session. Other tactics include ICMP redirect, manipulating traffic routing, and port stealing, where the attacker controls a network port to intercept data. These diverse methods underscore the importance of robust security measures to protect communication channels from MITM attacks.

1.5.2.9 User Privacy

User privacy, as discussed in [30] and [44], represents a critical aspect of cybersecurity. Unlike traditional breaches that involve direct access to sensitive data, privacy can be compromised by analyzing metadata and traffic patterns. Hackers can infer user activities and behaviors without accessing explicit information, making this a subtle yet significant threat. Protecting

user privacy requires comprehensive security measures that safeguard not just the data itself, but also the contextual information linked to user interactions, addressing the full scope of a user's digital footprint.

1.5.2.10 Cloning Nodes

Cloning Nodes, as discussed in [45] and [46], involves the covert replication of a node in a system or network. The attacker captures the original node's credentials and features, gaining control to manipulate the system. With the cloned node, they can inject false information, disable critical functions, and compromise system integrity. This type of attack is especially dangerous when the owner is unaware, allowing the cloned node to spread malicious activities across the network. The complexity of cloning attacks underscores the need for robust security measures to detect and prevent them.

1.6 IoT Application Layer Protocols

The IoT allows physical and virtual objects to collect data from their surrounding environments and enables smooth communication among diverse devices. However, due to the inherent heterogeneity of these devices, creating a universal protocol to accommodate all types is challenging. Consequently, multiple protocols are employed at the IoT application layer to ensure reliable communication and support various functionalities. This chapter aims to explore and discuss the primary application layer protocols utilized within the IoT ecosystem.

- **Hyper Text Transport Protocol (HTTP)**: Is a well-established web communication protocol initially created by Tim Berners-Lee and later formalized as a standard by the IETF and W3C in 1997. It follows a request-response architecture commonly used in RESTful web services, where data is exchanged through Universal Resource Identifiers (URIs). While HTTP is widely used for internet-based applications, its reliance on TCP and text-based message formatting introduces higher latency and overhead compared to other lightweight IoT protocols like MQTT and CoAP. Additionally, HTTP does not natively provide QoS mechanisms, requiring additional layers for reliable messaging. Despite these limitations, HTTP remains relevant in IoT applications that require integration with web services, cloud platforms, and secure communication using TLS/SSL [47].
- **Constrained Application Protocol (CoAP):** CoAP emerged as a pivotal internet utility protocol meticulously tailored for devices grappling with limited resources, especially over low-bandwidth networks. It facilitates machine-to-machine (M2M) communication and is tailored for IoT environments leveraging the Hypertext Transfer Protocol (HTTP) protocols. CoAP uses the UDP protocol for standard implementation, making it well-suited for devices with limited resources. CoAP incorporates Datagram Transport Layer Security (DTLS) to ensure secure data transfer, enhancing its reliability in IoT systems [48].
- **Advanced Message Queuing Protocol (AMQP):** represents a sophisticated message-oriented middleware protocol designed to ensure reliable and secure information exchange within IoT systems [49]. It incorporates fundamental elements responsible for

managing the routing and storage of messages within a broker service, adhering to rigorously defined policies [48]. The protocol's architecture comprises three core components: Exchange, which serves as the initial point for routing messages from publishers to designated queues; Message Queue, a repository for the temporary storage of messages awaiting processing; and Binding, which defines the parameters of the connection between the exchange and message queue, thereby regulating the flow of messages. AMQP's meticulously crafted framework underscores its pivotal role in establishing a robust and efficient communication infrastructure, integral to the dynamic landscape of IoT systems.

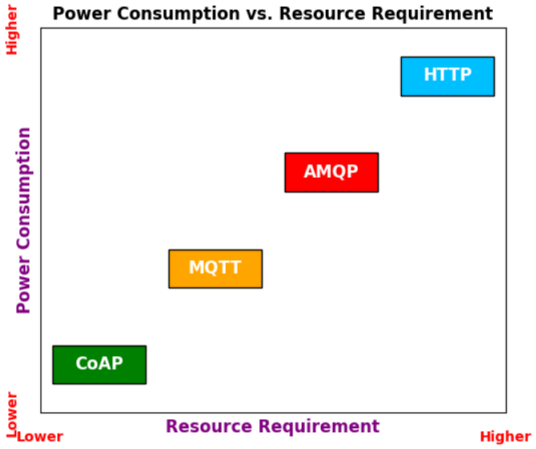
- **Data Distribution Service (DDS):**The DDS protocol uses a publish-subscribe mechanism and is a dynamic and versatile solution for scalable, real-time, and accurate data distribution in IoT systems. Renowned for its efficiency and interoperability, DDS employs multicasting to ensure high-quality QoS, making it integral to seamless information flow. The protocol spans diverse platforms, from low-footprint devices to cloud computing, emphasizing bandwidth consumption and adaptability within the IoT ecosystem. The DDS-IoT protocol architecture comprises two core layers: Data-Centric Publish-Subscribe (DCPS), which ensures efficient data delivery to subscribers and serves as the backbone of information dissemination, and Data-Local Reconstruction Layer (DLRL), an interface enabling distributed data sharing among IoT devices. These layers collectively enhance collaboration and interaction within IoT networks. DDS's comprehensive capabilities establish it as a foundational protocol, delivering robust, scalable, and real-time communication across IoT applications and platforms [48].

- **Message Queuing Telemetry Transport Protocol (MQTT) :** The MQTT protocol, widely used in the IoT ecosystem, is a consequential messaging protocol that saw its inception through a collaborative effort by Andy Stanford-Clark from IBM and Arlen Nipper of Arcom in 1999 [50]. Specifically tailored for M2M communication and remote tracking within IoT environments, MQTT has emerged as a cornerstone in the seamless exchange of information. At its core, MQTT primarily collects data from various devices, objects, and gadgets, fostering a cohesive network linking these entities to software packages and middleware [48].
  MQTT's architecture revolves around three key components: the subscriber, the publisher, and the broker. The publisher, responsible for generating and disseminating data, communicates this information to subscribers through the intermediary, known as the broker. This orchestrated exchange ensures the secure and reliable flow of information within the IoT ecosystem. Moreover, the broker plays a pivotal role in reinforcing security by meticulously scrutinizing the authorization of both publishers and subscribers, thereby establishing a robust foundation for the secure transmission of data over the MQTT protocol. The intricate dynamics of MQTT underscore its significance in orchestrating a cohesive and secure communication framework within the dynamic landscape of IoT systems.

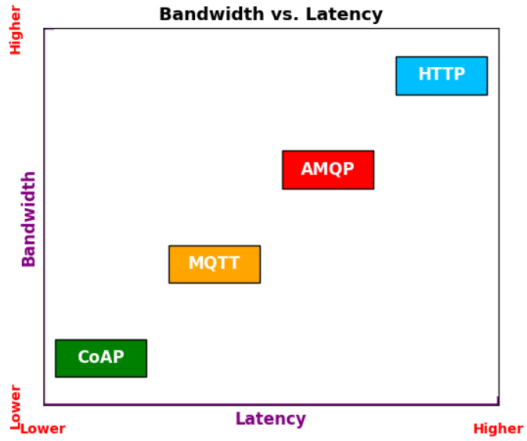
1.7 Protocol Selection

Selecting an appropriate IoT messaging protocol requires careful consideration of various factors such as standardization, message overhead, power efficiency, resource constraints,

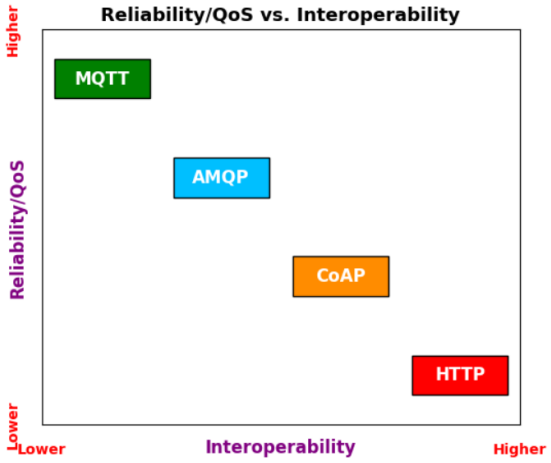
bandwidth, latency, security, QoS, and interoperability. According to a study by Nitin Naik [47] as Figure 1.6 shows, MQTT and CoAP are the most suitable lightweight protocols for IoT applications. However, due to the need for scalable and reliable message distribution, MQTT stands out due to its publish-subscribe architecture, which enables seamless communication across multiple devices. Furthermore, its support for multiple QoS levels ensures reliable data transmission under varying network conditions. Another significant advantage of MQTT is its extensive support in various programming languages, including C, Java, Python, and JavaScript. It allows for flexible development across different platforms, whether on computers or mobile devices. Considering these benefits, our study specifically focuses on the MQTT protocol, which will be thoroughly explored in the following sections.



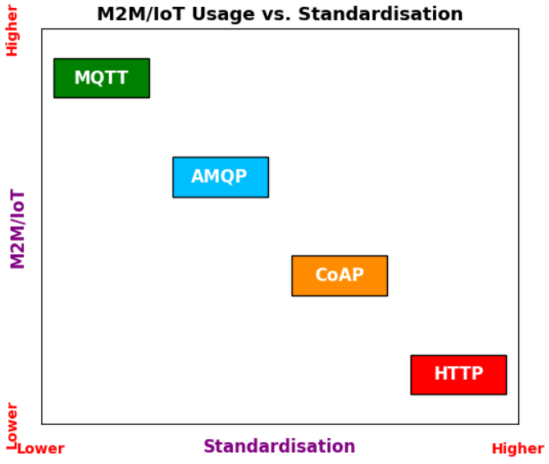
(a) Power Consumption vs. Resource Requirement



(b) Bandwidth vs. Latency



(c) Reliability/QoS vs. Interoperability



(d) M2M/IoT Usage vs. Standardization

Figure 1.6: Comparison of IoT Protocols Based on Key Performance Metrics

1.8 MQTT Architecture

The MQTT protocol, recognized for its lightweight design, is a widely adopted open standard endorsed by the Organization for the Advancement of Structured Information Standards (OASIS). It utilizes a publish/subscribe communication model as delineated in Figure 1.7

[3], making it particularly effective for M2M communication like smart homes, industrial automation, and healthcare monitoring.. Operating over TCP, MQTT ensures reliable, lossless, and bidirectional communication. The protocol is structured around topics, allowing users to publish messages under specific topics. Clients subscribed to those topics receive the corresponding messages, facilitating organized and targeted communication.
The MQTT architecture is based on essential components: publishers, subscribers, and brokers. Publishers transmit messages about specific topics, subscribers receive them, and brokers act as intermediaries, managing the distribution of messages between publishers and subscribers. These elements are integral to the protocol's efficient functioning [51]:

1. **Publishers:** Are MQTT clients responsible for sending messages to the MQTT broker. These messages contain data or information that other clients may be interested in. Publishers choose a topic under which they publish their messages. A temperature sensor could act as a publisher in a smart home system. The sensor continuously measures the temperature in a room and publishes this data on a specific topic. For instance, the temperature sensor might publish the temperature data on a topic like "Home/Living-Room/temperature ".

2. **Subscribers:** are MQTT clients who express their interest in receiving messages on specific topics. They subscribe to one or more topics of interest. When a publisher sends a message to a subscribed topic, the subscriber receives and processes that message. In our previous example, a smartphone app controlling the temperature may act as a subscriber, subscribing to the "Home/Living-Room/temperature" topic to receive and execute temperature commands.

3. **The broker:** Serves as a central intermediary in the MQTT network, managing the exchange of messages between publishers and subscribers. It receives messages from publishers, categorizes them by topic, and distributes them to the appropriate subscribers. For example, in a smart home system, when a temperature controller (publisher) sends commands to the topic "Home/Living-Room/temperature," the broker ensures these commands reach all subscribers interested in controlling the living room temperature. The broker maintains a registry of connected clients and their subscribed topics, enabling efficient communication. In essence, the broker is the hub connecting all MQTT clients within the network.

4. **Topics:** Are hierarchical categories under which messages are organized within the MQTT protocol. Topics enable a structured approach to communication, allowing clients to filter and receive only the information they are interested in. Topics are represented as strings and can include slashes to create a hierarchy. For instance, in a smart home system, a topic like "Home/Living-Room/temperature" implies a hierarchy where "Home" is the top-level category, " Living-Room" is a subcategory, and "temperature" is the specific piece of data being transmitted. Clients can subscribe to specific topics to receive messages relevant to their needs.

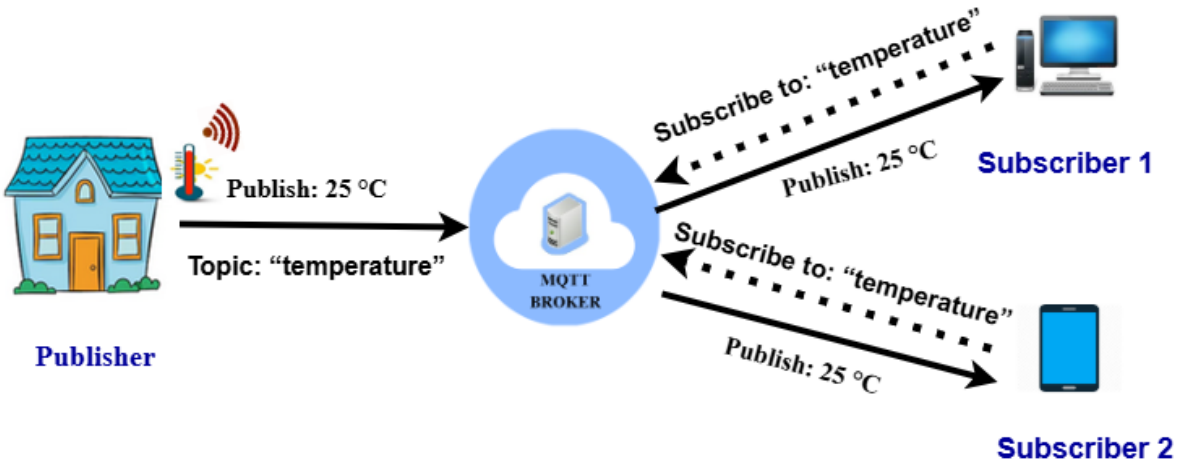


Figure 1.7: MQTT architecture

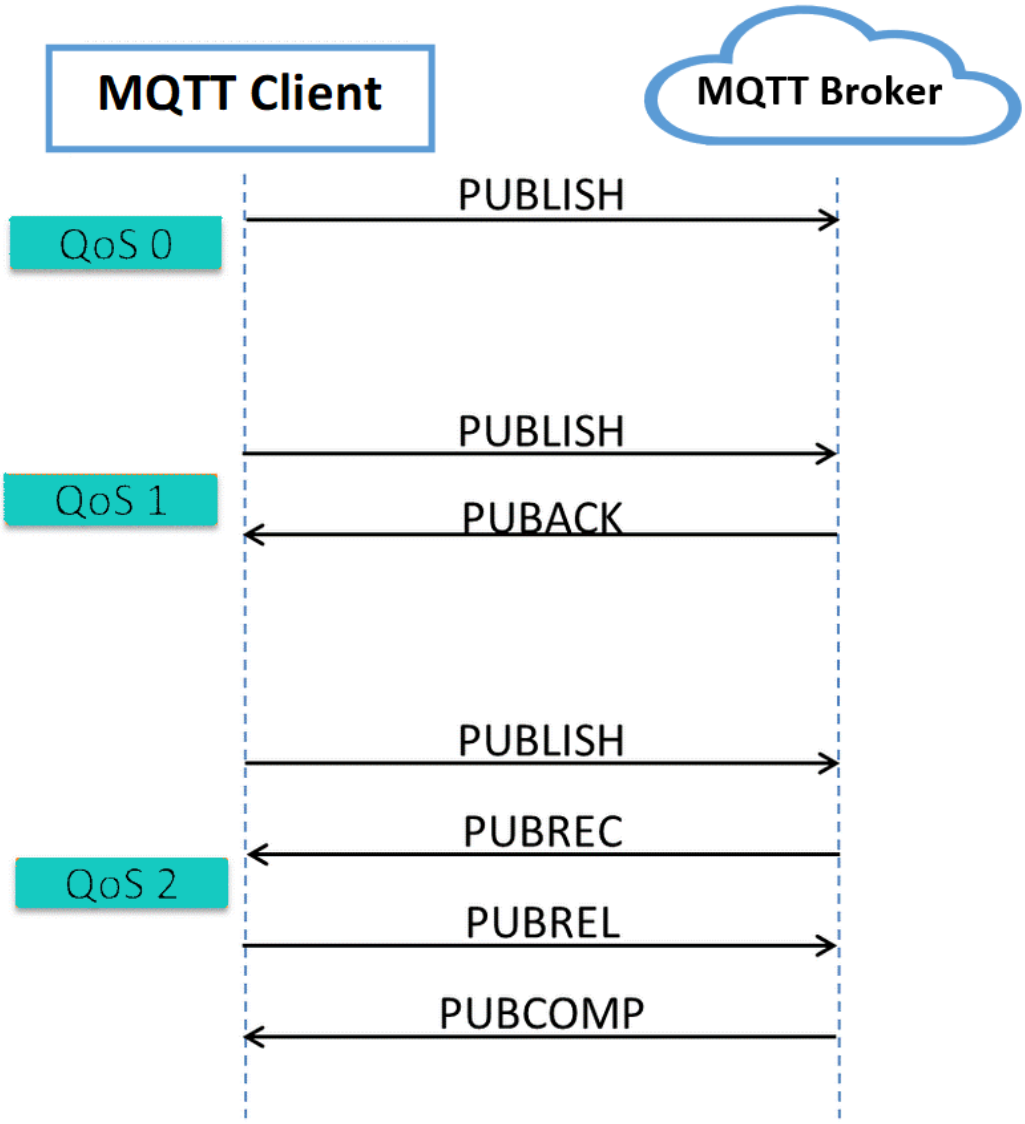
The MQTT protocol is designed with a simple and resource-efficient architecture, facilitating seamless communication between clients and the central broker. One key feature of MQTT is its implementation of Quality of Service or QoS for message delivery. QoS defines the reliability of message transmission, ensuring that connected devices can receive and process messages based on predefined delivery guarantees. The protocol offers three distinct QoS levels, which determine how many times a message will be delivered, ranging from zero to multiple deliveries, depending on the selected QoS level.

1. **QoS Level 0 (At most once):** This level operates with the least assurance, dispatching messages at most once without any confirmation or overhead. While it provides swift transmission, it lacks a delivery guarantee, making it suitable for scenarios where speed takes precedence over reliability.

2. **QoS Level 1 (At Least Once):** This level involves the transmission of messages through a single PUBLISH message exchange. In the absence of acknowledgment (PUBACK), the sender has the option to resend the PUBLISH message, albeit with the potential drawback of message duplication. This level strikes a balance between reliability and efficiency.

3. **QoS Level 2 (Exactly Once):** The highest level of delivery assurance, it guarantees that a message is delivered exactly once. This is achieved through a four-step handshake process between the sender and receiver, consisting of the PUBLISH, PUBREC, PUBREL, and PUBCOMP messages (Figure 1.8). While offering the utmost reliability, QoS Level 2 introduces significant overhead, making it best suited for applications where precise and secure message delivery is critical.

The choice of QoS level in MQTT depends on the specific needs of the IoT application, balancing speed with reliability. A lower QoS level may be suitable when fast transmission is prioritized over guaranteed delivery, while a higher QoS level is better when message delivery certainty is crucial. Figure 1.8 visually outlines the three QoS levels, helping users tailor their communication strategy to the unique requirements of their IoT application [51].



PUBLISH = Send , PUBACK =Publish Acknowledgment, PUBREC =Publish Received, PUBREL=Publish Release, PUBCOMP =Publish Complete

Figure 1.8: MQTT QoS level

1.9 MQTT Vulnerabilities

MQTT clients communicate with brokers via TCP/IP port 1883 for unencrypted exchanges, while port 8883 is used for encrypted communication through SSL/TLS (Secure Sockets Layer / Transport Layer Security) protocols. Despite the efficiency of the protocol in terms of low power consumption and minimal bandwidth usage, security remains a critical concern. The primary focus of MQTT developers on optimizing performance has come at the expense of robust security measures, leaving the protocol vulnerable to various threats [4]. This prioritization of efficiency over security has introduced several vulnerabilities, making the MQTT environment susceptible to potential attacks and compromising the integrity of

communication within IoT systems.
Andy et al. [52] attribute the weaknesses of the protocol to several factors, such as :

- **Resource-Constrained Devices:** Certain devices with limited resources, such as RAM and ROM memories, may struggle to implement the TLS encrypted protocol due to its high demands. These memory limitations make it difficult for devices to handle most security mechanisms, especially those involving complex computations.
- **Number of Devices:** The vast number of IoT devices makes managing them very difficult, which can negatively impact key security aspects such as confidentiality, integrity, and availability.
- **Lack of Security Awareness:** Developers may prioritize functionality over security, exploiting users' lack of knowledge about security concerns. Users may also ignore changing default usernames and passwords or neglect software updates, which can weaken the overall security of the device.

According to Burange et al. [53], the vulnerabilities in the MQTT protocol arise from the following:

- **Transmission of Plain Text Information:** MQTT messages are sent without encryption, often containing log-in credentials. As a result, unauthorized individuals can access, read, and modify the payload, compromising the confidentiality of the data.
- **Open Ports:** MQTT typically uses port 8883 for encrypted communication and port 1883 for unencrypted traffic. These open ports can be exploited to carry out attacks across networks.

Raikar and Meena [54] identify additional vulnerabilities:

- **Public and Accessible Brokers:** Attackers often use the Shodan search engine to collect information about connected devices worldwide. They target publicly available MQTT brokers to either extract data, inject false information, or even initiate a DoS attack against these brokers.
- **Wildcard Utilization:** The risk of data theft increases with the use of wildcards in topic names. For example, using the wildcard "#" can allow users to subscribe to all available topics.

1.10 Datasets

In most cases, IoT networks are susceptible to various forms of attacks, which prompts the examination and analysis of network behavior in the presence of these attack scenarios. During this process, benign and malicious data are gathered and employed to train ML, DL, and other IDS algorithms for the development of robust IDSs. The following datasets are among the most commonly used.

1. **N-BaIoT:** The N-BaIoT dataset, as described [55], contains network traffic data from various IoT devices, including doorbells, a thermostat, a baby monitor, security cameras, and a webcam. It features benign and malicious traffic from Mirai and BASHLITE botnet attacks, offering a valuable resource for studying and detecting IoT-based botnet threats.

2. **ToN_IoT:** The ToN_IoT dataset [56] captures data from IoT and Industrial Internet of Things (IIoT) environments, including telemetry, operating systems, and network interactions across edge, fog, and cloud layers. It provides a comprehensive resource for studying intrusion detection in large-scale IoT networks.

3. **Bot-IoT :** The Bot-IoT dataset, available on [57], is a synthetic dataset designed for network forensics research using ML and DL techniques. It includes five IoT scenarios simulating devices such as weather stations, smart fridges, motion-activated lights, garage doors, and smart thermostats. The dataset features three categories of attacks: information-gathering (e.g., port scans, OS fingerprinting), DoS (e.g., TCP, UDP, HTTP attacks, including DDoS), and information theft (e.g., keylogging, data theft). These attacks, commonly associated with botnets, are represented by over 72 million packet capture (PCAP) records, with information theft attacks having the fewest records [58].

4. **IoTNID:** The IoT Network Intrusion Dataset (IoTNID) [59] was created using two real devices: a camera and a speaker. It includes various attack scenarios, such as reconnaissance, MiTM, DoS, and Mirai attacks. Most attack packets, except for those related to the Mirai attack, were captured using the Nmap tool, while Mirai attack packets were generated using a laptop.

5. **IoT-23:** The IoT-23 dataset [60] was carefully curated using three real IoT devices: a Philips HUE smart LED light, an Amazon Echo, and a Somfy smart door lock. It includes 20 malware scenarios and three benign scenarios, with the malware scenarios exposed to various botnet attacks such as Mirai, Gafgyt, and Torii. The dataset underwent detailed manual analysis to extract and distinguish features related to both benign and attack traffic.

6. **MedBIoT:** The MedBIoT dataset [61] simulates a medium-sized network with 80 virtual devices and 3 real devices, such as a switch, light bulb, lock, and fan. It was exposed to attacks from three botnets: Mirai, BASHLITE, and Torii. The dataset is designed to provide data for intrusion detection focused on botnet activities.

7. **MQTT-IoT:** The MQTT-IoT dataset [62] is based on the MQTT protocol and created in a simulated environment with 12 IoT sensors across four attack scenarios and one benign scenario. Its main purpose is to support intrusion detection using machine learning techniques.

8. **MQTTset:** The MQTTset [63] crafted to facilitate the application of ML techniques within MQTT networks. The simulation involves eight diverse sensors, covering temperature, light, humidity, carbon monoxide (CO) gas, motion, smoke, door, and fan, utilized to emulate five different MQTT network attacks. This dataset distinguishes itself by excluding certain features like source and destination IP addresses, port addresses, and communication times, among others, commonly present in other datasets. Instead, it predominantly emphasizes MQTT-specific features.

9. **Edge-IIoTset:** The Edge-IIoTset dataset [64] explores IoT and IIoT networks, featuring data from 12 devices, such as sound sensors and motors, across seven layers. It includes testing under 15 attacks grouped into five categories, providing a robust resource for security research.

10. **CICIoT2023:** The CICIoT2023 dataset [65] is a comprehensive IoT resource, featuring data from 105 devices and 33 attacks categorized into seven groups. It includes diverse IoT technologies like Zigbee and Z-wave, making it one of the largest and most inclusive datasets for IoT security research.

1.11 Conclusion

This chapter provided an overview of the IoT ecosystem. It covered the definition of IoT and highlighted key security challenges. The discussion on IoT architecture outlined its three layers and their vulnerabilities, while the section on IoT attacks distinguished between passive and active threats affecting data integrity, privacy, and system availability. Additionally, the chapter summarized commonly used application layer protocols, with a particular focus on the MQTT protocol an efficient device communication protocol, which is the core of our study.

The MQTT protocol plays a pivotal role in IoT communication, excelling in applications such as industrial automation, remote sensing, and agriculture. Its adaptability to low-bandwidth networks and support for asynchronous messaging make it a cornerstone of IoT systems. Recent advancements, including MQTT 5.0, have expanded its functionality, enabling more sophisticated IoT implementations. However, despite its advantages, challenges like security and interoperability remain critical considerations.

The next chapters will explore ML and DL techniques that power modern attack detection mechanisms. It will also review related work in the field, tracing the evolution and efficacy of various approaches. This discussion will bridge theoretical foundations and practical applications, providing insights into advancements in cybersecurity.

Chapter 2

Machine Learning and Deep Learning Methodologies

2.1 Introduction

The IoT is a cornerstone of the digital age, enabling seamless data collection and transmission across interconnected networks and fostering a globally unified digital ecosystem. This interconnected framework has inspired extensive research, particularly on the challenges of securing and preserving the privacy of IoT data. Recent scholarly efforts have highlighted the adoption of ML and DL methodologies as transformative tools for enhancing the detection and mitigation of IoT-related cyber threats.

As illustrated in Figure 2.1, ML and DL are pivotal domains within Artificial Intelligence (AI), significantly shaping the technological landscape. ML focuses on designing algorithms that enable computers to learn from data and make well-informed decisions. These capabilities have revolutionized various sectors through advancements in predictive analytics, automated decision-making, and the detection of patterns in diverse datasets [66]. Within ML, DL represents a specialized subfield that leverages neural networks with multiple layers to model complex patterns in massive datasets. Applications of DL span fields such as autonomous systems, natural language processing, and image and speech recognition, achieving remarkable outcomes by emulating the human brain's structure and functionality [67].

This chapter delves into ML and DL's core principles, methodologies, and applications. It explores fundamental ML models, such as logistic regression, k-nearest neighbors, and support vector machines, alongside the role of Ensemble Learning techniques, including bagging, boosting, and stacking, in enhancing model performance and reliability. Additionally, it examines DL models, including convolutional neural networks, recurrent neural networks, and generative adversarial networks.

This chapter aims to comprehensively understand how ML and DL methodologies address real-world problems by integrating theoretical insights with practical implementations. This exploration will illuminate the transformative potential of these technologies across various domains, paving the way for future innovations.

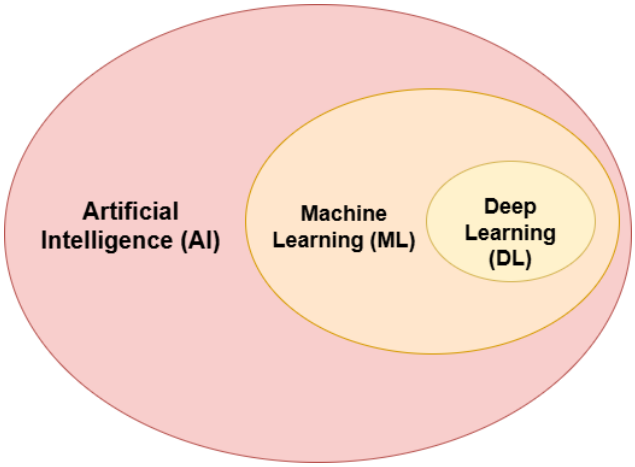


Figure 2.1: Artificial Intelligence Subfields

2.2 Machine Learning (ML)

2.2.1 Definition

Machine learning is a branch of artificial intelligence that involves developing algorithms and models that enable computers to learn from data and make predictions or decisions without being specifically programmed for every task [29]. ML emerged in the 1940s as researchers explored methods to enable computers to learn from data. Its growth accelerated in the 1990s with advances in computing power and the availability of large datasets, leading to the development of more sophisticated algorithms and practical applications [68].
This field is typically categorized into three main types: supervised learning (using labeled data for predictions), unsupervised learning (identifying patterns in unlabeled data) and reinforcement learning (learning through rewards and penalties). The widespread relevance of ML is demonstrated by its use in various sectors, including recommendation systems, autonomous vehicles, cybersecurity, and more [69].

2.2.2 Machine Learning Techniques

To detect sophisticated attacks on IoT devices and develop effective defense strategies, ML offers various approaches, including supervised learning, unsupervised learning, and reinforcement learning. These techniques can improve network security by addressing authentication, access control, anti-jamming, offloading, and malware detection [29, 70]. Each ML approach has its specific application within the security of IoT. Figure 2.2 illustrates the different ML algorithms used to secure IoT systems.

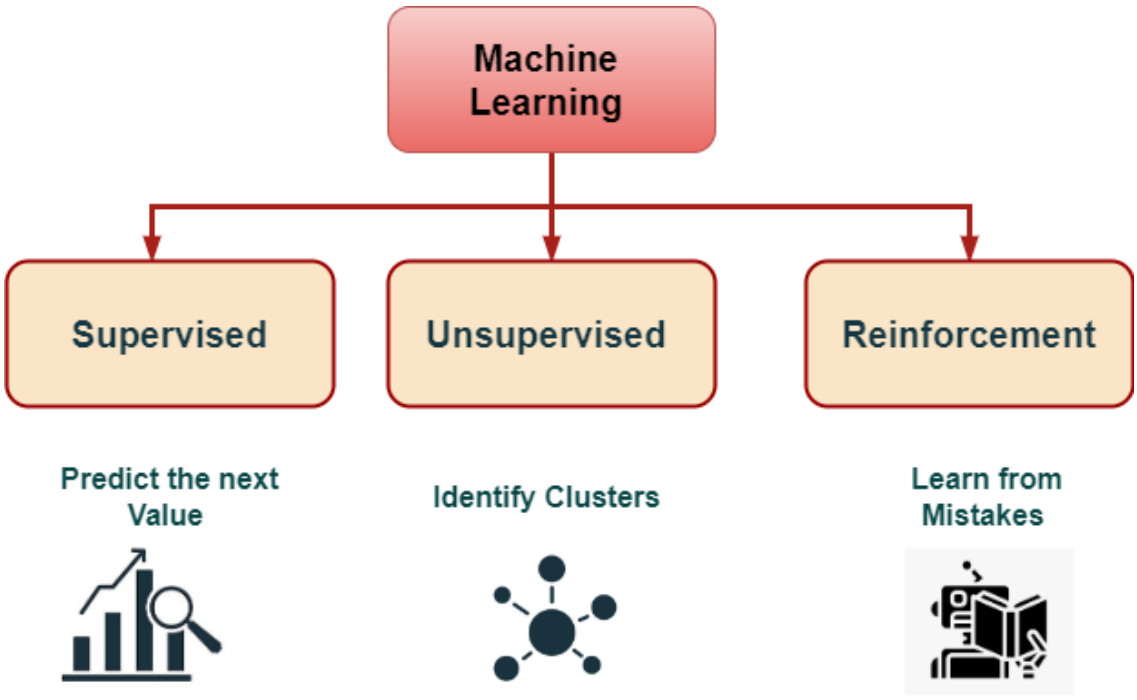


Figure 2.2: Machine Learning Subfields

2.2.2.1 Supervised Learning :

It is a key ML technique that uses labeled training data to build predictive models and make decisions. By pairing input features with corresponding target labels, the aim is to identify and generalize patterns in the data. This process is similar to a teacher guiding a student's learning, helping the model align inputs with the correct outputs. Supervised learning primarily consists of two types: classification models, which categorize data into classes, and regression models, which predict continuous values. Despite the complexity of the algorithms, the focus is on ensuring clear, interpretable relationships between inputs and outputs [71, 70, 72].

A) **Classification :** It involves training a model to predict discrete or categorical output labels for new data based on labeled training data. The model assigns instances to predefined categories, such as weather conditions (sunny, rainy) or animal types (cat, dog, fish). Common real-world applications include email spam detection, image recognition, and sentiment analysis. Supervised learning algorithms used for classification include Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), Random Forest (RF), and Association Rule (AR).

B) **Regression :** It focuses on predicting continuous or numerical output values based on labeled training data. It is used for tasks such as predicting property prices or stock values, using input features like historical data and market signals. While supervised learning can provide highly accurate predictions, it faces challenges such as the difficulty and cost of acquiring labeled data and the risk of overfitting, where models perform well on training data but poorly on new, unseen data. Common regression algorithms include Decision Tree (DT), Neural Network (NN), and Ensemble Learning (EL).

2.2.2.2 Unsupervised Learning :

Unsupervised learning is an ML approach where algorithms discover patterns, structures, or relationships in unlabeled data without relying on input-output pairings. The algorithm autonomously explores the data to identify meaningful groupings or patterns. This learning domain encompasses various tasks, including clustering, anomaly detection, and latent variable learning. Since no labels are provided, it isn't easy to verify model performance, making evaluation more challenging. Unsupervised learning is typically divided into two main types:

A) **Clustering:** This technique divides a dataset into separate groups or clusters. The core principle here is that data points grouped within a particular cluster exhibit a significant level of similarity or closeness compared to those in other clusters.

B) **Dimensionality Reduction:** This approach focuses on reducing the number of input features or variables in a dataset. The objective of dimensionality reduction is to retain the most crucial aspects of the data by transforming a high-dimensional dataset into a more concise representation with fewer dimensions.

Applications of unsupervised learning span various domains, such as anomaly detection, data compression, and visualization, underscoring its versatility. One of the primary benefits of unsupervised learning is its aptitude to unveil obscured patterns or relationships within data. These nuances might remain undiscovered if one were to rely exclusively on labeled data. Moreover, unsupervised learning emerges as an invaluable tool when labeled data is scarce or the process of obtaining such labeled data proves to be prohibitively expensive.

2.2.2.3 Reinforcement Learning (RL):

The key components of this learning paradigm include [73]:

A) **Agent**: refers to the active entity constantly interacting with the surrounding environment. The state of the environment influences its behavior, and it adapts its subsequent actions based on the feedback received.

B) **Environment**: This encapsulates the external system or context with which the agent engages. As the agent exhibits certain behaviors or takes actions, the environment responds, often by providing a reward signal indicative of the action's outcome.

C) **Reward Function**: This function plays a crucial role by translating each combination of state and action into a reward value, signifying the effectiveness or consequence of the agent's chosen action.

This type of learning emerges as a unique approach where an agent learns to make decisions and navigate an environment to maximize long-term rewards. Through continuous interaction with the environment, the agent receives feedback, which helps refine its decision-making strategy or policy. The goal is to create a policy that links specific states in the environment to optimal actions. Over time, the agent improves its policy through iterative learning from feedback. There are two main approaches within RL:

- Value-Based Methods: Focus on evaluating the value of individual states or state-action combinations to create an optimal value function for maximizing long-term rewards.

- Policy-Based Methods: Focus on optimizing the policy itself, determining the best actions to take in various states to maximize cumulative rewards.

Reinforcement learning is versatile and applicable in various fields, such as robotics, personalized medicine, natural language processing, and resource allocation. Its adaptability allows it to make informed decisions in complex, dynamic environments where the best action may not always be obvious.

2.2.3 Machine Learning Applications

Over the years, ML has evolved into a ubiquitous technology, deeply integrated into various applications across various industries. Its impact spans from simple tasks like image and speech recognition to complex functions such as fraud detection and autonomous vehicle navigation. ML's versatility underscores its current widespread use and potential for further expansion. Researchers and innovators continue to explore new ways to integrate this transformative technology into even more applications [67].
The scope and utility of ML [74] encompass a wide spectrum, including but not limited to:

- **Speech and Image Recognition**: equipped with advanced algorithms, can decipher and recognize entities within images or discern spoken words, converting them into understandable and actionable data.
- **Natural Language Processing (NLP)**: This branch leans on ML to unravel the intricacies of human language. Doing so facilitates operations like sentiment analysis, which gauges emotional undertones in text or translation, bridging linguistic barriers.
- **Predictive Modeling**: With the prowess to analyze historical data, ML models can forecast future occurrences. This can range from predicting consumer behavior trends to making informed projections about stock market movements.
- **Anomaly Detection**: ML algorithms can flag deviations or anomalies within data sets through keen observation and pattern recognition. This becomes especially critical in scenarios like fraud detection or identifying potential network security breaches.
- **Robotics**: ML breathes intelligence into robots, enabling them to perform various tasks. This includes recognizing objects, navigating through terrains, or even manipulating items.
- **Health Care**: A sector where precision is paramount, ML is being harnessed to revolutionize medical processes. From sifting through vast medical records to identifying diseases to crafting personalized treatment regimens, its potential in healthcare is vast.
- **Energy Management**: ML enhances energy management by predicting consumption patterns, adjusting smart appliances, and detecting inefficiencies. AI-driven systems in smart homes regulate heating, cooling, and lighting, reducing waste and promoting sustainability by balancing power grids.
- **Environmental Monitoring**: Our planet's well-being is a shared concern and ML aids in monitoring its health. It aids in analyzing vast amounts of environmental data, facilitating better understanding and predictions, such as in climate modeling.

2.2.4 Machine Learning Models

Machine learning models as described by "Batta Mahesh" [75], are advanced algorithms that learn from data to make decisions or predictions. These models integrate mathematical and statistical techniques to identify patterns, extract insights, and apply knowledge to practical scenarios. Each algorithm has unique characteristics, contributing to the diverse range of methods within the ML field. This section provides an overview of the most widely used ML algorithms.

2.2.4.1 Logistic Regression (LR)

Logistic regression is a fundamental method in statistical analysis for examining the relationship between a binary dependent variable (y) and one or more predictor variables (x). It predicts the probability of a binary outcome using a logistic function, which models the likelihood of the dependent variable belonging to a particular category [76]. LR is valued for its simplicity, ease of interpretation, and resilience against overfitting, especially when dealing with datasets of lower dimensionality. It is particularly useful when probabilistic predictions are needed in real-world applications, as it provides probability scores for outcomes (Figure 2.3).

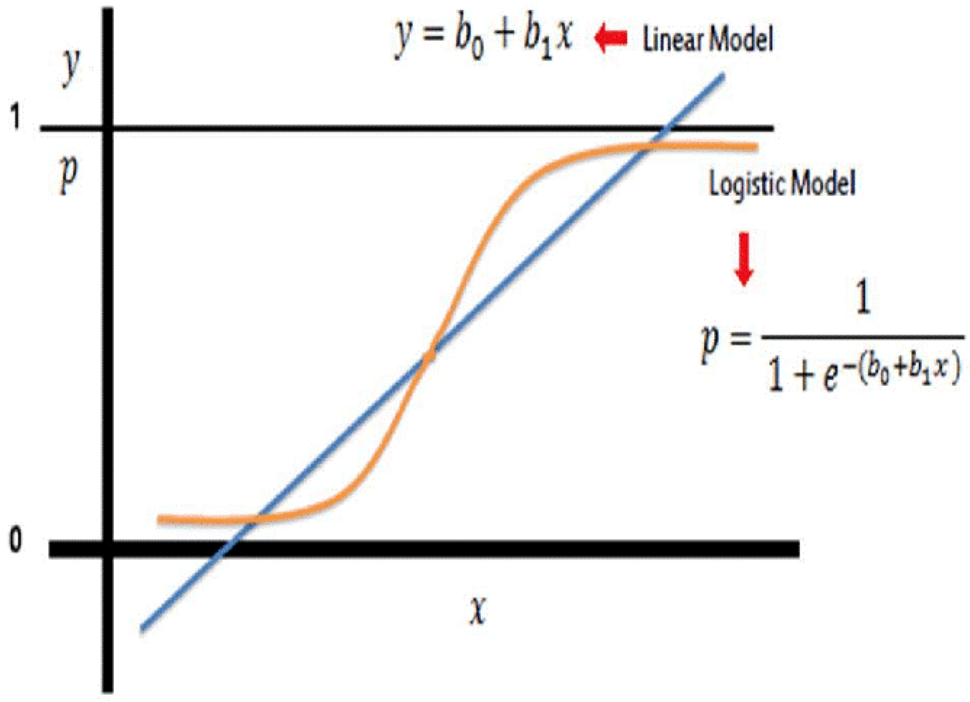


Figure 2.3: Logistic regression Model (Appiah et al. 2019) [1]

However, LR has assumptions and limitations. It assumes a linear relationship between the log odds of the dependent variable and the independent variables, which may not always hold in real-world data. Additionally, LR requires a large sample size for reliable predictions and may struggle with many categorical predictors. The logistic function (sigmoid), which

is the foundation of LR, is described by Equation 2.1, where it produces the probability of a given outcome [1].

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

LR can be applied in different formats: binary, multinomial (multiple categories), and ordinal (categories with a logical order). The model makes predictions by applying a decision boundary, such as the threshold of 0.5 in binary LR, which classifies outcomes into distinct categories as shown in Equation 2.2.

$$Y = \begin{cases} A, & \text{if } f(X) \geq 0.5 \\ B, & \text{otherwise} \end{cases} \tag{2.2}$$

2.2.4.2 Decision Trees (DT)

Decision Trees (DT) are a widely embraced algorithm known for their simplicity [2]. They employ a tree-like structure to make predictions and decisions by partitioning data based on features. Starting at a root node, the process recursively branches into child nodes for possible outcomes, continuing until specific criteria are met. The terminal nodes, or "leaves," provide the final prediction values, as illustrated in Figure 2.4. There are two primary types of DTs:

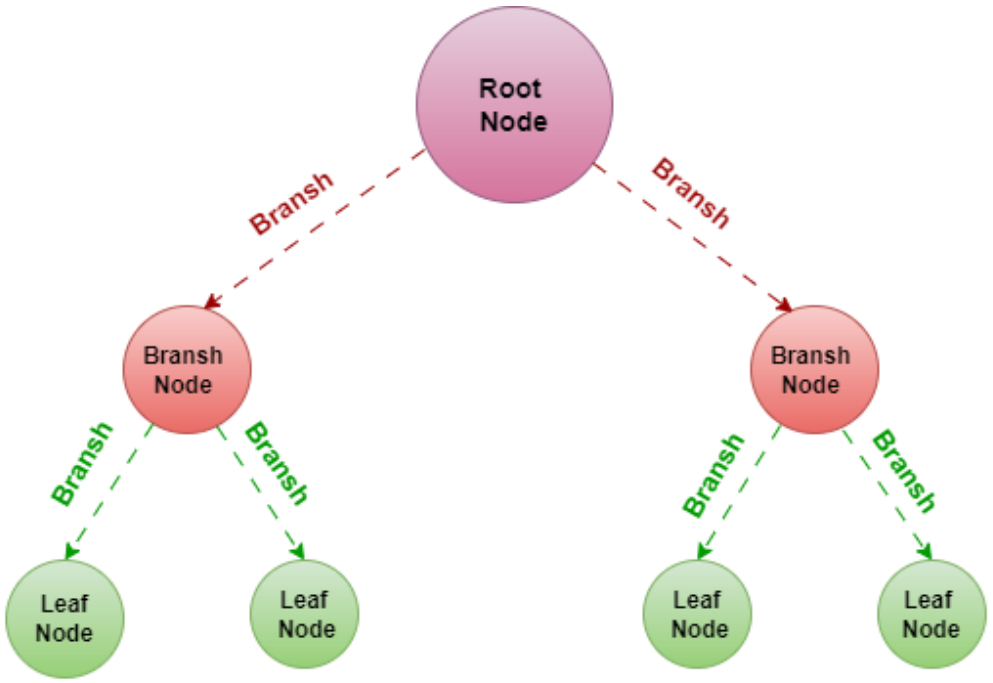


Figure 2.4: Decision Trees Model [2]

- **Classification Trees:** These are aptly utilized for categorical or qualitative response variables. Each leaf in the tree corresponds to a specific class, delineating the classification for the given set of input features.

- **Regression Trees:** Come into play when dealing with numeric or quantitative response variables. In this scenario, each leaf in the tree represents a numerical value that elucidates the regression prediction.

DTs are valued for their transparency, interpretability, and minimal data preprocessing requirements. They handle missing values effectively, accommodate nonlinear relationships as nonparametric models, and aid in feature selection, with top nodes often indicating the most influential features. However, DTs are prone to overfitting, particularly with complex or noisy datasets, which can hinder their generalizability. Pruning and careful tuning are necessary to address this limitation, emphasizing the importance of a thoughtful approach to their implementation.

2.2.4.3 K-nearest Neighbors (KNN)

The KNN algorithm classifies new data points by identifying the 'K' closest points from the training dataset and assigning a label based on the majority class of its neighbors [77]. It is effective in both binary and multiclass classification, using strategies like majority voting or distance-weighted voting to determine the label. Known for its simplicity and effectiveness [78], KNN relies on distance measures to evaluate the similarity between instances, such as the L2 norm (Euclidean distance) in Equation 2.3 and the L1 norm (Manhattan distance) in Equation 2.4 [79].

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.3}$$

$$\|x\|_1 = \sum_{i=1}^{n}|x_i - y_i| \tag{2.4}$$

The KNN algorithm determines the label of a new instance by finding its 'K' nearest neighbors in the training data and using majority voting. A small 'K' makes the model sensitive to noise, while different 'K' values can change the classification. Figure 2.5 shows sample KNN boundaries, with two classes represented by red and yellow colors, and an unknown instance in green. The label of the unknown instance changes based on different 'K' values, leading to varying classifications [80].

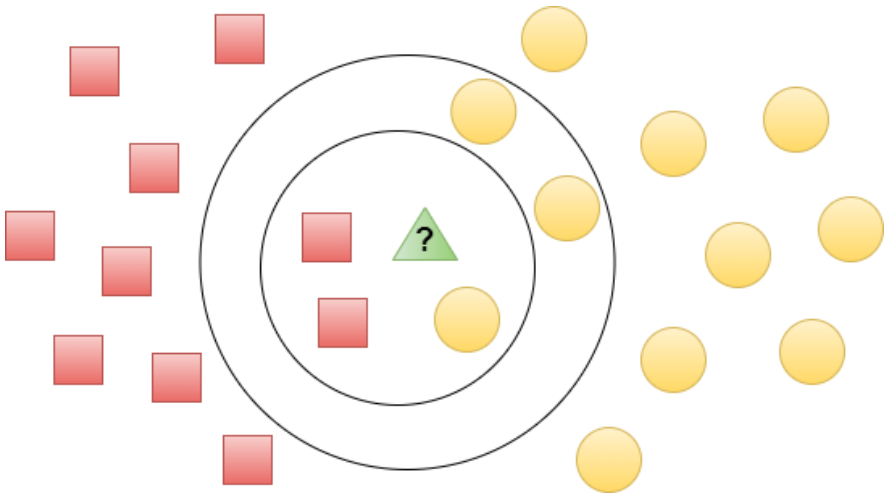


Figure 2.5: K-nearest Neighbors Model

The KNN algorithm is simple, versatile, and handles nonlinear relationships, but it struggles with high dimensionality, sparsity, and sensitivity to the choice of 'K' and distance metric.

2.2.4.4 Support Vector Machine (SVM)

The SVM is a powerful supervised ML technique primarily used for pattern recognition and classification tasks [81]. The goal of training an SVM model is to construct an optimal hyperplane in a higher-dimensional space that effectively separates the classes. For binary classification [82], the SVM optimization problem can be mathematically formalized as outlined in 2.5 and 2.6.

$$\min_{\mathbf{w}\in\mathbb{R}^n} \|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max(0, 1 - y_i f(x_i)) \tag{2.5}$$

$$f(x_i) = \mathbf{w}^T\mathbf{x_i} + b \tag{2.6}$$

In the SVM optimization equation, $\|\mathbf{w}\|$ represents the norm of the weight vector, which corresponds to the margin width that SVM aims to maximize. The parameter 'C' controls the trade-off between maximizing the margin and minimizing classification errors. The term $max(0, 1 - y_i f(x_i))$ is the loss function, penalizing points on the wrong side of the margin. Geometrically, the hyperplane divides the feature space, with its position relative to data points determined by the sign and magnitude of $y_i f(x_i)$. Points with a value greater than 1 lie beyond the margin with no loss, points equal to 1 are on the margin edge and points less than 1 are on the wrong side, increasing the loss.
When data cannot be separated linearly, SVMs use kernel functions to project input features into a higher-dimensional space where separation is possible. SVM kernels include linear, Radial Basis Function (RBF), polynomial, and sigmoid kernels [83]. These kernels and their decision boundaries [84], shown in Figure 2.6 , help SVMs handle complex, non-linearly separable datasets.

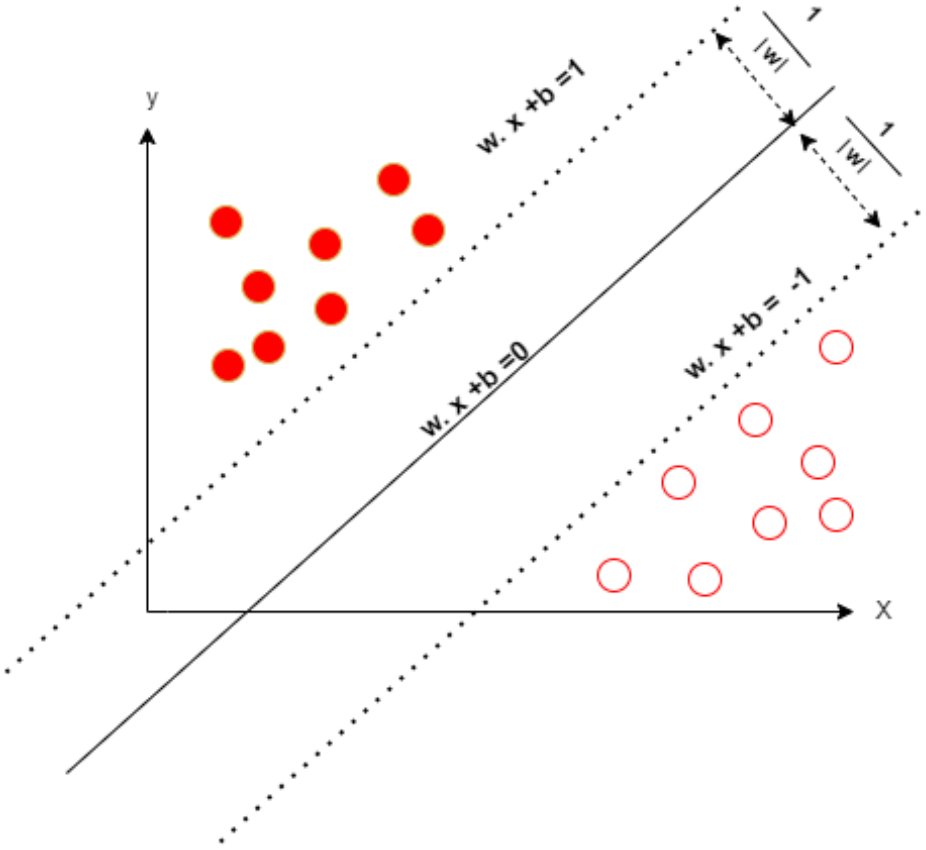


Figure 2.6: SVM Model

2.2.5 Ensemble Learning Model (EL)

Ensemble learning is an ML technique that combines multiple models, leveraging their individual strengths to create a more accurate, stable, and robust composite model, similar to how a symphony orchestra achieves harmony through diverse instruments working together. The strength of an ensemble lies in its diversity, utilizing the unique strengths of each model to mitigate weaknesses and enhance overall predictive power, especially for complex problems. Among the ensemble techniques, several stand out due to their effectiveness and widespread adoption in the field [85]. These include:

2.2.5.1 Bagging

Bagging, short for bootstrap aggregating, reduces variance and prevents overfitting by creating multiple random training data samples with replacement. This process generates diverse datasets, enhancing the model's generalization and robustness [3].
Bagging trains multiple independent base models on varied bootstrap samples, allowing each to learn unique patterns. It then combines their predictions, creating a consensus for more reliable and accurate outcomes. In classification tasks, bagging employs a voting system where each model votes for a predicted class, and the class with the majority votes becomes the final prediction (Figure 2.7). This method minimizes the influence of outlier predictions,

enhancing stability and accuracy [86]. Among the EL techniques, we distinguish the random forest and extra trees models.

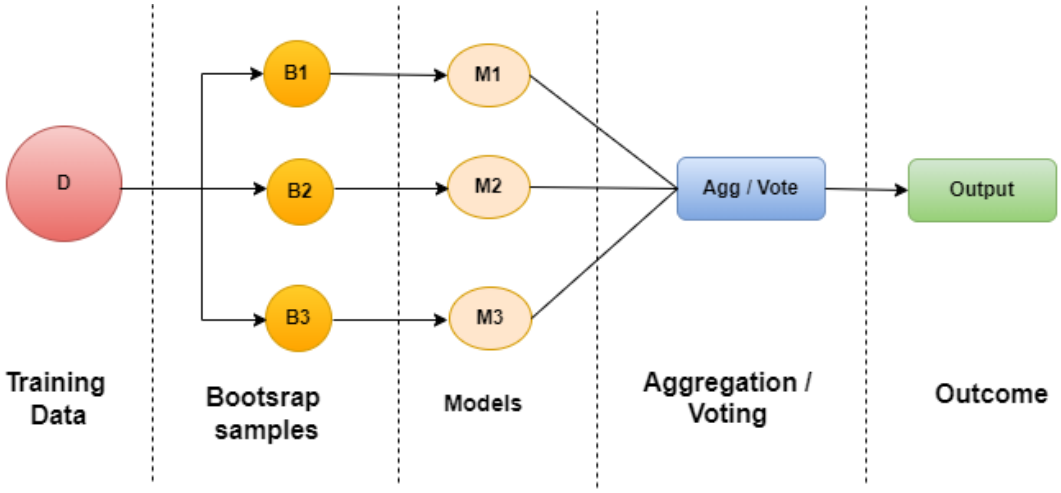


Figure 2.7: Bagging technique Model

However, bagging has limitations, including high computational demands during training and aggregation and significant memory requirements to store base models and predictions. Despite these challenges, its effectiveness in producing stable, accurate models makes it a valuable approach for ML [87].

- **Random Forest (RF):** The RF algorithm is a robust EL technique that combines bagging and feature randomness to create accurate predictive models for classification and regression tasks. It builds multiple decision trees, each trained on a bootstrap sample of the data, ensuring exposure to different aspects of the dataset. Additionally, RF introduces randomness at each split by selecting a random feature subset, allowing the trees to capture diverse patterns. The final prediction is made by averaging the outputs of all trees, resulting in a more accurate model than any individual tree [87]. The RF algorithm can be mathematically represented as an ensemble of individual estimators, where each tree, denoted as $f_i(X)$, predicts based on the feature space X. In regression tasks, the final prediction is the average of all tree predictions, as expressed by the equation:

$$F_{RF}(X) = \frac{1}{n}\sum_{i=1}^{n} f_i(X)$$

  where $F_{RF}(X)$ is the predicted value, and n is the number of trees in the forest. This averaging process results in more stable and reliable predictions than individual trees. In classification tasks, each tree "votes" for a class, and the final class prediction is determined by the majority vote, often weighted by tree accuracy or reliability [87].

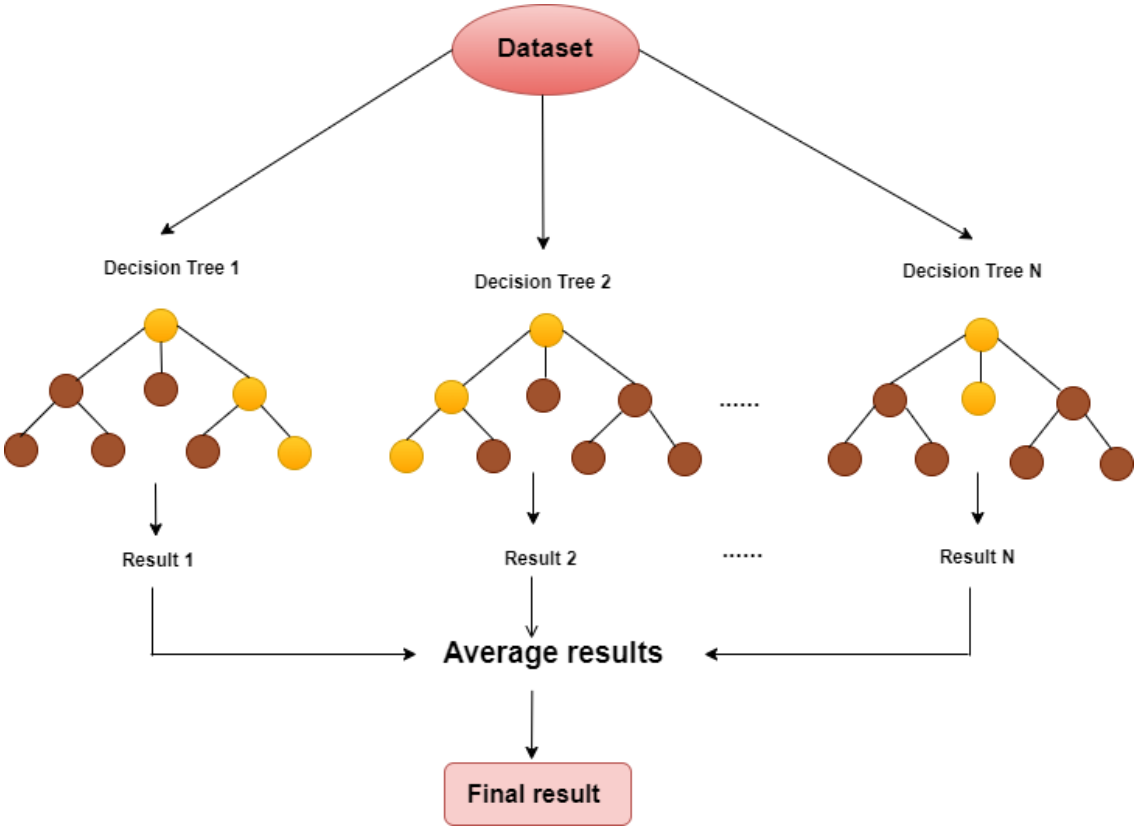


Figure 2.8: Random Forest Model

Moreover, the RF algorithm includes an Out-Of-Bag (OOB) error estimate as an internal evaluation mechanism. During training, about one-third of the data is left out for each tree, forming the OOB data. This omitted data is used to assess the tree's performance, providing a built-in cross-validation process [88].

- **Extra Trees:** Short for Extremely Randomized Trees is an EL technique similar to RF, but with increased randomness during tree construction. Unlike RF, which selects optimal thresholds for splits, Extra Trees randomly chooses cut-points for each feature and selects the best among them for splitting. This heightened randomness helps improve the model's resilience against overfitting.
  In Extra Trees (Figure 2.9), each tree is built using the original dataset without resampling, unlike traditional bootstrapping. At each split, a random value is selected for every feature, increasing randomness in both feature selection and threshold determination. This approach reduces the correlation between trees, lowering variance and improving prediction accuracy. In classification tasks, predictions are made by majority voting, where the class receiving the most votes from the trees becomes the final prediction.

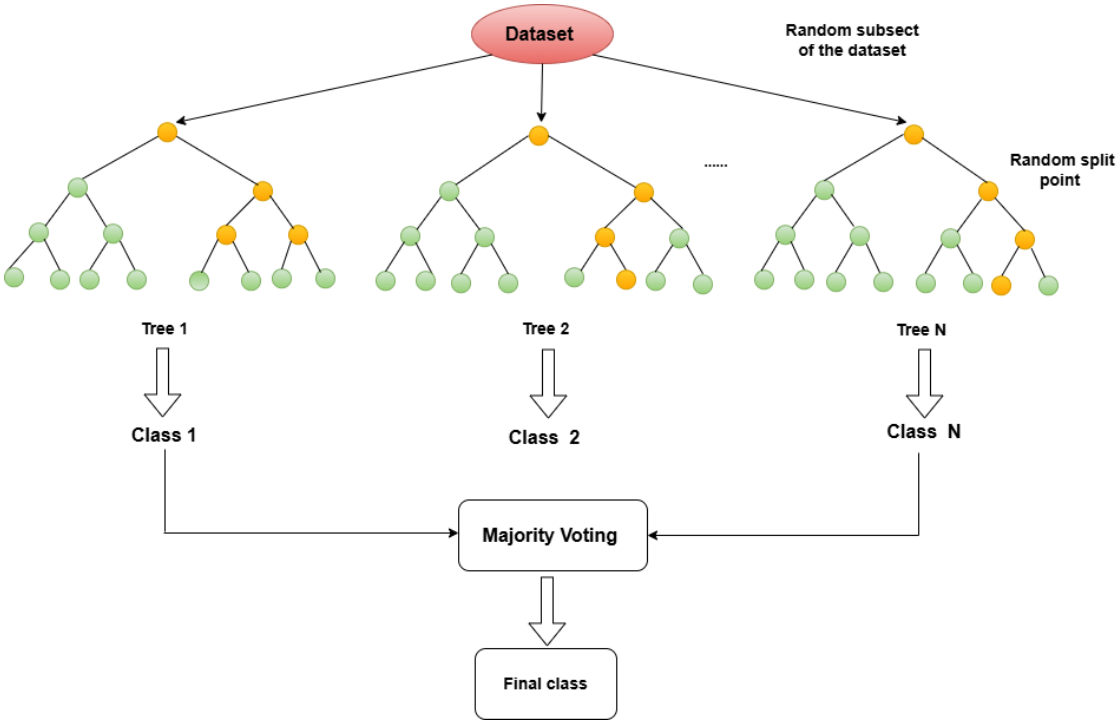


Figure 2.9: Extra Trees Model

Extra Trees is a powerful tool for handling large, feature-rich datasets, effectively reducing overfitting through its increased randomness. This randomness allows it to create flexible decision boundaries, while the random selection of cut-points speeds up training, especially in high-dimensional spaces. However, Extra Trees can introduce bias, requiring careful tuning of parameters such as tree count and depth to balance bias and variance. Despite this, it remains popular due to its simplicity, ease of use, and strong performance[89].

2.2.5.2 Stacking

Stacked generalization (or stacking) is an advanced EL technique that improves prediction accuracy by combining multiple models. Several base models (level-0) are trained on the same dataset using diverse algorithms (e.g., KNN, DT, NN), providing a range of predictive insights. Instead of using the base models' predictions directly, they are used as input for a meta-learner (level-1), which learns how to combine them to improve overall performance. This method not only refines predictions but also reduces the risk of overfitting by leveraging multiple learning algorithms [90]. The architecture of stacking, illustrated in Figure 2.10, allows for flexible integration of various learning models, making it a robust choice for tackling complex predictive tasks where single models might falter due to their inherent biases or limitations [86].

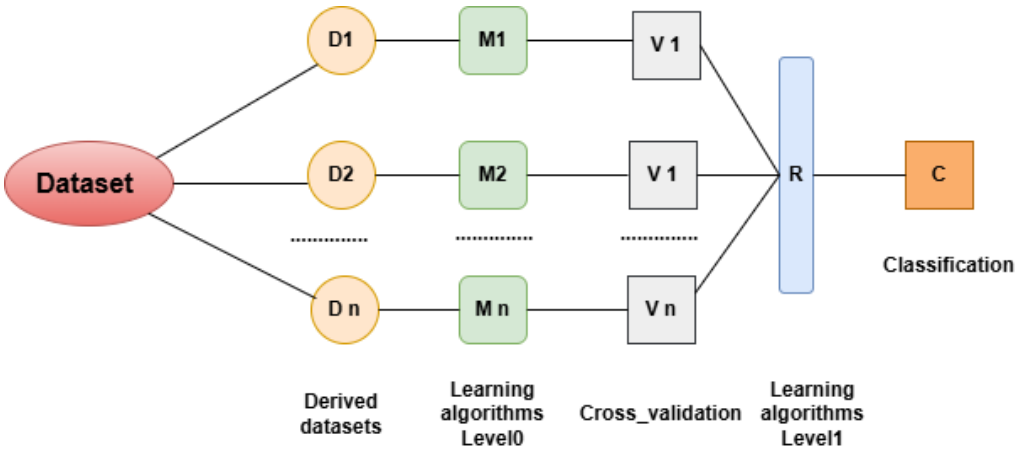


Figure 2.10: Stacking Scheme

2.2.5.3 Boosting

Boosting stands out as a formidable EL technique in ML, engineered to elevate prediction accuracy by orchestrating a sequential fusion of basic and feeble learning algorithms. These modest learners, often in the form of DTs, individually possess a knack for making predictions slightly better than random chance. The crux of boosting lies in its iterative approach: it learns from the missteps of its forerunners, forging a chain of models that rectify the errors of their predecessors. This iterative refinement endeavors to craft a robust final model endowed with markedly diminished bias and variance, a concept vividly depicted in Figure 2.11 [86].

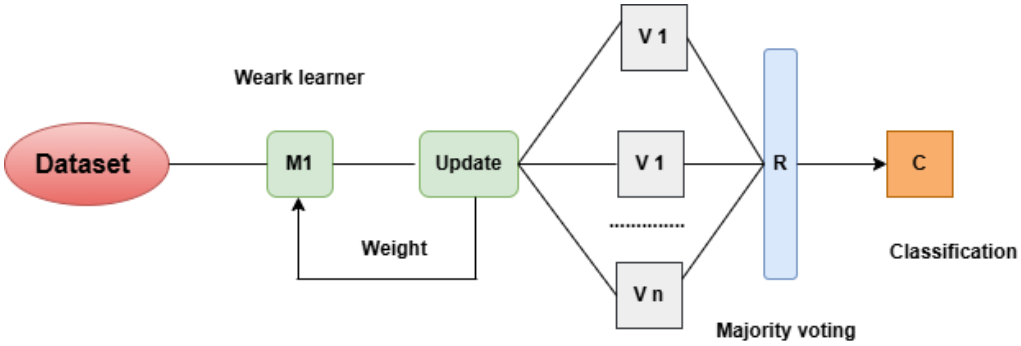


Figure 2.11: Boosting Scheme

Boosting algorithms include several techniques, such as AdaBoost (Adaptive Boosting), Gradient Boosting Machine (GBM), XGBoost (eXtreme Gradient Boosting), and LightGBM.

2.3 Deep learning (DL)

2.3.1 Definition

Deep Learning is a subfield of AI and ML that enables models to learn from data in a way that mimics the human brain's neural networks. DL has shown remarkable performance

in applications such as image and speech recognition, natural language processing, predictive analytics, and attack detection. At its core, DL relies on artificial neural networks, where neurons are connected in layers, processing and transmitting signals. Weights between neurons are adjusted during learning to improve predictions. DL has revolutionized problem-solving across various domains. Similarly to the ML field, DL includes three main techniques: supervised, unsupervised, and reinforcement learning. [66].

2.3.2 Deep Learning Applications

DL has been extensively implemented in varied fields, fundamentally transforming how complex issues are tackled and facilitating informed decision-making. This segment will delve into a selection of the most notable implementations of DL and analyze their influence on different fields.

- **Computer Vision:** A major application of DL is computer vision, particularly in image classification, which automatically categorizes images for searchability or tagging. For example, DL can detect human faces by recognizing facial features like edges and lips. Other key uses include object recognition and video analysis, essential for technologies like self-driving cars, robotics, theft detection, and movie rating systems [91].

- **Natural Language Processing (NLP):** DL has greatly advanced NLP, enabling machines to better understand, interpret, and generate human language. DL has improved applications like machine translation, sentiment analysis, language modeling, and text summarization. This progress has enhanced customer service, marketing, legal document analysis, and powered language-based apps, chatbots, virtual assistants, and content recommendation systems.[66].

- **Recommendation Systems:** DL has transformed recommendation systems by automating the learning of complex patterns from raw data. DL models excel at capturing non-linear relationships between users and items, allowing for personalized recommendations based on subtle user preferences and item characteristics. When combined with collaborative filtering, DL enhances the accuracy of recommendations for products, movies, music, and more. Companies like Netflix, Amazon, and Spotify use these technologies to improve user experience and engagement. [92].

- **Speech Recognition and Synthesis:** DL has revolutionized speech recognition and synthesis, enabling accurate transcription and natural-sounding speech. Some models process and generate speech with high precision, powering applications such as virtual assistants, real-time translation, and improved human-computer interaction. [93, 66]. DL is now integral to systems like Google Translate and ChatGPT.

- **Healthcare:** DL has transformed healthcare by improving medical image analysis for disease detection, diagnosis, and treatment planning. It has also played a key role in drug discovery, personalized medicine, and predictive healthcare by analyzing large datasets, including electronic health records and genomic data. DL's ability to process vast amounts of information has led to significant advancements in healthcare technology, enhancing patient care and overall healthcare quality [94].

- **Autonomous Vehicles:** DL plays a crucial role in developing autonomous vehicles. It is used for object detection, scene understanding, path planning, and decision-making. It helps vehicles recognize key elements such as objects, pedestrians, and road signs, while also supporting vehicle control tasks like steering, acceleration, and braking. DL enables self-driving cars to learn driving behavior from observations, allowing them to navigate complex environments, identify obstacles, and make real-time decisions for safer and more efficient transportation [95].

- **Gaming and Robotics:** Deep reinforcement learning, which combines DL and reinforcement learning, is widely used in gaming and robotics. It allows agents such as game characters and robots to learn from experience and make optimal decisions in complex situations. This approach has been applied in areas like game playing, robot control, and motion planning, leading to the development of more advanced and intelligent systems [96].

- **Medical Applications:** DL has transformed medical applications by providing advanced tools for disease diagnosis, treatment planning, and medical image analysis. DL helps classify diseases, differentiate stages, and identify complex patterns in medical images. It has shown high accuracy in tasks like diagnosing ocular diseases (e.g., diabetic retinopathy), analyzing histopathology images for cancer detection, and designing cancer radiotherapy. With ongoing advancements, DL holds great potential to improve healthcare outcomes and patient care [66].

- **Educational Systems:** DL is transforming educational systems by improving the analysis and use of student data. Using advanced neural networks, DL models can process large volumes of academic data to identify complex patterns and relationships across text, images, and structured data. In student feedback analysis, DL helps automatically classify sentiments, identify key themes, and extract meaningful insights from unstructured text. Integrating DL with linguistic knowledge and sentiment analysis makes educational systems more intelligent and adaptive, enhancing learning environments [97].

- **Cybersecurity:** DL is becoming a key tool in cybersecurity, improving anomaly detection, malware identification, spam filtering, and phishing prevention by learning patterns from network traffic and system logs. DL models excel at detecting new threats, providing real-time responses, and addressing data scarcity with semi-supervised learning, by processing large datasets and adapting to evolving cyber threats. The future of DL in cybersecurity involves more unsupervised and human-like learning to overcome challenges like labeled data limitations and improve system adaptability, offering significant potential for proactive threat management in the digital world [98].

These are just a few examples of the numerous applications of DL in various domains. As the field continues to evolve and new advancements are made, we can expect DL to play an increasingly significant role in addressing complex challenges and driving innovation across industries.

2.3.3 Deep Learning Models

There are many well-known DL models, each tailored for specific tasks. Due to the rapid development of DL, new networks and architectures emerge frequently, which may be beyond the scope of this article. This section covers several popular DL models.

2.3.3.1 Multi-Layer Perceptron (MLP):

The MLPs are a key type of feedforward Artificial Neural Network (ANN) used in supervised DL. They consist of an input layer, an output layer, and hidden layers that process data. MLPs generate predictions using activation functions such as ReLU, Tanh, Sigmoid, and Softmax. Optimization techniques such as Stochastic Gradient Descent and Adam are used for training. While tuning MLPs hyperparameters can be computationally intensive, MLPs excel at learning complex, non-linear patterns in data, making them a powerful tool in DL applications [94].

2.3.3.2 Convolutional Neural Networks (CNN):

the CNNs are supervised DL algorithms primarily used for visual data processing, such as images and videos. They consist of convolutional layers (for feature extraction), pooling layers (for dimension reduction), activation functions (for non-linearity), fully connected layers (for feature combination), and an output layer (for predictions). The CNNs automatically learn features from data, making them highly effective in tasks like image classification and object detection. As a result, CNNs are widely used in fields like computer vision, speech processing, and natural language processing. [99].

2.3.3.3 Recurrent Neural Network (RNN):

The RNNs are supervised DL models designed to handle sequential data by incorporating loops that allow the network to remember previous inputs and calculations. This makes RNNs ideal for tasks where the sequence and context of data are important, such as time series analysis, speech recognition, and natural language processing. RNNs can process variable-length input sequences, capturing temporal patterns and correlations. To address challenges like long-term dependency tracking and the vanishing gradient problem, variations such as Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks have been developed. [100].

2.3.3.4 Stacking Automatic Encoders Technique (SAE):

It is an unsupervised DL technique that uses multiple layers of autoencoders to learn hierarchical data representations. Each autoencoder encodes and reconstructs input data to minimize reconstruction error, with the output of one layer feeding into the next. This iterative process helps the model capture complex patterns, leading to more abstract representations in higher layers. Variations like denoising, contractive, sparse, and convolutional autoencoders offer specific advantages, such as robustness to noise, stability, sparsity, and improved image processing. This technique is valuable in deep learning applications for automatically learning intricate features in data [101, 102].

2.3.3.5 Generative Adversarial Networks (GANs):

GANs are unsupervised DL consisting of two neural networks: the Generator (G) and the Discriminator (D), which are trained together through a competitive process. The generator creates fake data from random noise, aiming to produce samples that resemble real data. The discriminator acts as a binary classifier, distinguishing between real and fake data and providing feedback to the generator. This adversarial training loop improves both networks, with the generator producing more realistic data and the discriminator becoming better at identifying fakes. GANs are widely used in applications like image and video generation, data augmentation, anomaly detection, and drug development, revolutionizing how data synthesis and generation problems are approached [103].

2.4 Classifier evaluation metrics

The performance of ML and DL is investigated to see how well classification algorithms can learn to profile IoT devices on the network, detect wireless attacks, and classify the type of such attacks when the corresponding network activity data was used to train and evaluate the classification model. When deciding whether a packet is malicious or harmless, the classification is compared to the training dataset, and four outputs are generated [104], as shown in Figure 2.12.

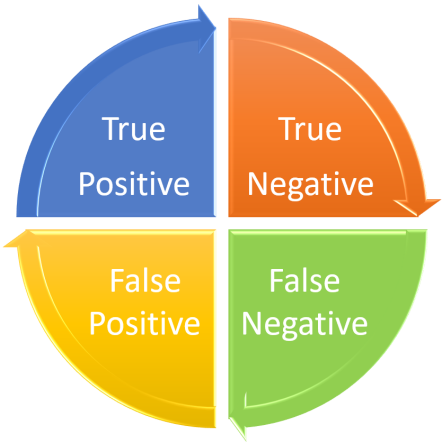


Figure 2.12: The Outputs Generated by the ML Classifier

- **True Positive (TP):** The packet that is predicted to be benign is really benign.
- **True Negative (TN):** The packet predicted to be malicious is really malicious.
- **False Positive (FP):** The Packet is predicted to be benign, but it is malicious.
- **False Negative (FN):** the packet that is predicted to be malicious but benign.

The goal of evaluating the performance of a classifier is to maximize all measures, which range from 0 to 1. Therefore, larger numbers indicate greater categorization performance [104]. ML provides many indicators for assessing a classifier's effectiveness. Some of the most essential performance measures shown in Figure 2.13 [71] are described as follows:

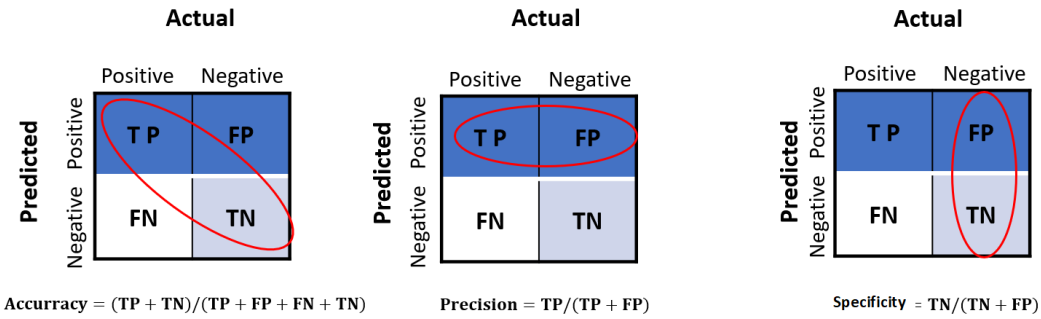


Figure 2.13: Accuracy, precision, and specificity performance metrics

- **Accuracy (ACC):** It evaluates how well a classifier can identify intrusions or assaults from an IDS standpoint. In other words, ACC is calculated by dividing the number of valid predictions (true positive and true negative) by the total number of predictions produced by the model [105].

$$\text{Accuracy (ACC)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.7}$$

- **Precision (Positive Predictive Value/ PR):** Is officially defined as the ratio between the number of true positives and the number of true positives plus the number of false positives [71]. Precision is computed as follows:

$$\text{Precision (PREC)} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.8}$$

- **Recall (Sensitivity):** Determines the percentage of harmful packets that were properly detected [104]. It is computed as follows:

$$\text{Recall (REC / SENS)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.9}$$

- **Specificity (True Negative Rate):** Often known as true negative rate, it is a metric that evaluates a model's ability to detect negative situations properly [71]. It is computed as follows:

$$\text{Specificity (SPEC)} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.10}$$

- **F1 Score(F1-S) :** Also called F-measure, combines precision and recall. A high F1 means a model correctly identified attack traffic while decreasing false positives and negatives. It's a harmonic mean of precision and recall. It calculates in the following way [105]:

$$\text{F1 Score ((F1-S)} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.11}$$

- **Model Loss:** In ML, model loss measures how much the model's predictions differ from the target values. Minimizing this loss during training improves model accuracy.

Different tasks use specific loss functions, and cross-entropy is typically used for classification. Lower loss values generally mean better model performance [106, 107]. In classification tasks, The model loss is quantified using the following equation :

$$\text{Cross-entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik}) \tag{2.12}$$

Where:

N is the number of samples,

K is the number of classes,

y_{ik} is 1 if the true label of the sample i is class k (one-hot encoded),

$\hat{y}_{ik}$ is the predicted probability that sample i belongs to class k.

- **ROC Curve:** A Receiver Operating Characteristic curve (ROC curve) is a graph that shows how well a classification model performs across all classifications and thresholds. This curve displays two parameters, where the True Positive Rate (TPR) is a function of the False Positive Rate (FPR) in this curve. We must pick a threshold to categorize the input. An output value over that threshold indicates a positive class, whereas an output value below that threshold indicates a negative class [71].

- **AUC (Area Under the ROC Curve):** The AUC curve is a model comparison statistic metric. As its name indicates, it measures the area beneath the whole ROC curve (Figure2.14). This variable sums up the model's performance across several categorization thresholds. This parameter helps us determine which training model is the most accurate at predicting classes. Put another way, it aids classification model ranking and sorting [71]. AUC is calculated in the following way [108] :

$$\text{AUC} = \int_0^1 \frac{\text{TP}}{\text{TP}+\text{FN}}\, d\left(\frac{\text{FP}}{\text{FP}+\text{TN}}\right)$$

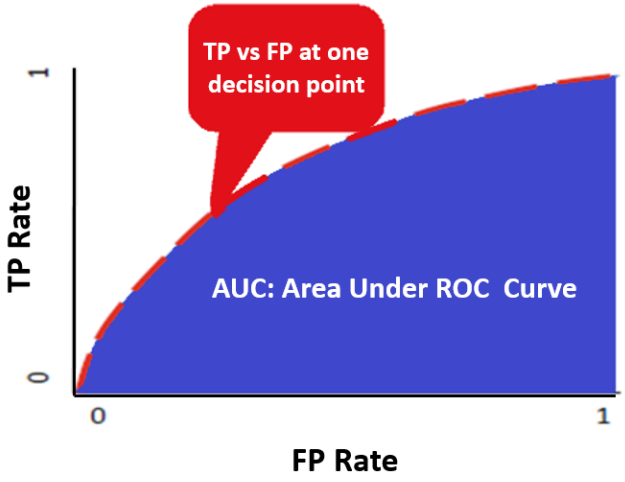


Figure 2.14: A typical ROC / AUC curve .

- **Matthew's Correlation Coefficient (MCC):** It is a statistical metric used to evaluate models. It provides a high score when predictions perform well across all four categories of the confusion matrix (true positives, false negatives, true negatives, and false positives), considering the size of the dataset's positive and negative components. The MCC can be calculated using the following mathematical equation [3]:

$$MCC = \frac{(TP * TN - FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.13}$$

2.5 Conclusion

In conclusion, this chapter delved into the fundamental concepts of ML and DL Methodologies. We first explored the broad field of ML, including its definition and various techniques employed. Additionally, we examined practical applications and different models utilized in ML, with a specific focus on ensemble learning models.
Furthermore, we delved into the realm of DL, providing a definition and discussing its applications across different domains. We also explored various DL models utilized in contemporary applications. After that, we presented the most renowned metrics employed to assess ML algorithms' performance.
Overall, this chapter was a foundational introduction to ML and DL techniques. It set the stage for further exploration and understanding of these intricate and powerful techniques and the justifications for their applications in the IoT domain.

Chapter 3

Machine Learning and Deep Learning Models for Cybersecurity in MQTT Environments: A Review of Related Work

3.1 Introduction

The IoT has transformed device communication, bringing not only unparalleled convenience and efficiency but also significant security challenges that demand robust measures. IDSs powered by ML and DL offer critical solutions for IoT security. These systems effectively detect known and unknown threats by adapting to new data patterns and identifying anomalies, surpassing traditional security methods. They enable real-time monitoring, proactive threat responses, and enhanced accuracy by analyzing large data volumes and complex patterns. The MQTT protocol, a key IoT messaging standard, highlights the need for such advanced IDS to secure IoT environments comprehensively.
In this section, we survey recent advancements in IDS that utilize ML, DL, and GAN methods to address cybersecurity challenges within IoT environments where the MQTT protocol is employed. Our focus is on innovative methods developed between 2020 and 2024 that aim to enhance the security of IoT networks by mitigating vulnerabilities related to the MQTT protocol. By citing these emerging approaches, we provide insights into the evolving landscape of IDS solutions tailored for MQTT-based IoT ecosystems.

3.2 Machine Learning-Based Intrusion Detection System for MQTT Environment

ML enhances IDS by improving their ability to recognize both known and novel threats. ML-based IDS can adapt to evolving threats and process large volumes of data in real time, leading to more effective threat detection and streamlined security management.
Vaccari et al. [6] developed the MQTTset dataset to support intrusion detection research focused on the MQTT protocol widely used in IoT environments. The dataset combines legitimate IoT traffic with simulated cyber-attacks targeting the MQTT broker. To validate

its utility, the authors designed an IDS and used the dataset to train and test various ML algorithms, including NN, RF, NB, DT, GB, and MLP. Their evaluation demonstrated the effectiveness of the MQTTset dataset in advancing the development and testing of IDS for IoT networks.

Hindy et al.[109] introduced the "MQTT-IoT-IDS2020" dataset, which encompasses benign and malicious scenarios specific to MQTT-based networks. Their study evaluates the efficacy of six ML algorithms (LR, KNN, DT, RF, SVM, and NB) for detecting MQTT-based attacks in IDSs. They explored features at three levels of abstraction: packet-based, unidirectional, and bidirectional flows. The findings indicate that the proposed ML models adequately meet the IDS needs of MQTT-based networks. Additionally, the study highlights the importance of employing flow-based features to distinguish MQTT-based attacks from benign traffic, while packet-based properties are sufficient for identifying traditional networking attacks.

Liu et al. [110] developed a Bayesian rule learning-based IDS for MQTT protocols, focusing on enhancing intrusion detection. They processed network traffic data, discretized continuous variables to handle non-normal distributions, and used the Hill-Climbing algorithm to build an optimal Bayesian network structure. By integrating domain knowledge through blacklists and whitelists, they refined the network and improved learning efficiency. This Bayesian network effectively captured conditional dependencies between features, enabling accurate detection and classification of intrusions in MQTT networks.

Ahmad et al. [111] proposed an approach for enhancing network security in IoT environments. The authors introduce feature clusters based on Flow, MQTT, and TCP using the UNSW-NB15 data set, aiming to address challenges like over-fitting, dimensionality, and data imbalance. By applying supervised ML algorithms such as RF, SVM, and ANN to these clusters, the study achieves high classification accuracy in binary and multi-class scenarios. Notably, the research demonstrates that the proposed feature clusters lead to improved accuracy and reduced training time compared to existing ML-based approaches.

Pokhrel et al.[112] proposed a method combining data collection, pre-processing, and SMOTE for dataset balancing to improve the detection of botnet-based DDoS attacks in IoT networks. They used feature engineering to evaluate ML methods such as KNN, NB, and MLP, providing insights into enhancing detection accuracy and efficiency. Their findings emphasized the role of feature reduction in boosting model performance and the effectiveness of SMOTE and artificial neural networks in achieving class balance in real-time data.

Sultan et al. [113] emphasized the critical role of suitable data communication protocols in IoT, focusing on the MQTT protocol and its vulnerability to cyber threats like man-in-the-middle (MiTM) attacks. They evaluated five ML models (ANN, NB, XGB, DT, KNN) on an open-source dataset, assessing metrics such as accuracy, precision, recall, F1 score, and processing times, mindful of IoT devices' resource limitations. All models achieved over 95% accuracy, with XGB performing best at 99.6% accuracy, albeit with longer training times.

siddharthan et al. [114] developed an intelligent IDS for IoT networks using Elite Machine Learning algorithms (EML) and a lightweight protocol to manage time constraints. The authors created the SENMQTT-SET dataset comprising normal scenarios and attacks on subscribers and brokers. They used an ensemble multi-view cascade feature generation algorithm to generate optimized features for intrusion detection. They evaluated the dataset using several ML algorithms (LR, KNN, SVM, NB, DT, RF, GB), and they selected the best model using EML, achieving an accuracy of over 99% for attack detection in MQTT-IoT networks.

Zeghida et al. [3] proposed a novel approach to improve IDS efficiency in IoT environments

by creating a balanced binary dataset from MQTT data, addressing imbalanced data challenges. They applied EL techniques, including bagging, boosting, and stacking, to enhance IDS performance. Their method achieved up to 95% detection accuracy and F1 score, with an MCC exceeding 90%, highlighting the effectiveness of dataset balancing and ensemble learning in strengthening IoT network security against MQTT-based attacks.
Musleh et al. [115] conducted a study on implementing an IDS that combines feature extraction with ML and DL algorithms. They used ML techniques such as KNN, SVM, and RF alongside feature extraction methods, including DenseNet and VGG-16 transfer learning, to improve intrusion detection accuracy in IoT networks. Experimental validation on the IEEE Dataport image dataset showed promising results, with the stacked model combined with VGG-16 achieving 98.3% accuracy. These findings highlight the effectiveness of their approach in enhancing IoT security and detecting malicious traffic.
Javed et al. [116] proposed a novel two-layered IDS for smart home IoT devices, integrating edge and cloud computing to enhance detection accuracy and efficiency. The system embeds an XGBoost-based IDS within a smart thermostat for real-time detection of attacks like DoS and MITM, achieving 97.59% accuracy. A more comprehensive XGBoost IDS is deployed in the cloud to detect various attacks, including DDoS, port scanning, and ransomware. They used the IDSH dataset, containing real-time data from a Raspberry Pi-based adversary, and compared various ML algorithms (RF, DT, MLP, ANN, LSTM, Conv1d), demonstrating the effectiveness of embedding IDS in IoT devices while leveraging cloud resources for broader security coverage.
Rakha et al. [117] proposed a hybrid model for IoT-enabled smart towns using the MQTT-IoT-IDS2020 dataset, aiming to improve the efficiency and security of smart cities. The model combines IoT technologies with IDS to address urbanization challenges. They applied various ML classifiers, including LR, KNN, NB, DT, Adaboost, and RF, to evaluate the model's performance. The results showed that the hybrid model outperformed traditional methods in precision, recall, F1 score, and accuracy, effectively identifying cyber threats and enhancing the security of IoT-enabled smart cities.

The table below summarizes the previous related work based on ML classifiers.

Table 3.1: Summary of Related Work Based on Machine Learning

Study	Year	ML used	Data sets	Attacks	Evaluation Metrics
Vaccari et al. [6]	2020	NN, RF, NB, DT, GB, MLP	MQTTset	Bruteforce, DoS, Flood, Malformed, SlowITe	ACC, F1-S
Hindy et al. [109]	2020	LR, k-NN, DT, RF, SVM, NB	MQTT-IoT-IDS2020	Aggressive scan, UDP scan, Sparta SSH brute-force, MQTT BF	ACC, Precision, REC, F1-S

Continued on next page

Table 3.1 – continued from previous page

Study	Year	ML used	Data sets	Attacks	Evaluation Metrics
Liu et al. [110]	2021	Bayesian networks	Data recorded	Sniffing DoS, PDoS, Replay Attack, FDIA, Botnet	ACC, REC , F1-S
Ahmad et al. [111]	2021	RF, SVM, ANN	UNSW-NB15	Exploits, Reconnaissance, DoS, Generic, Shellcode, Fuzzers, Analysis, Backdoor, Worms	ACC, Confusion matrix
Pokhrel et al. [112]	2021	KNN, NB, MLP	BoT-IoT	DDOS	ACC, PREC, REC, F1-S, AUC
Sultan et al. [113]	2022	ANN, NB, XGB, DT, KNN	Open-source dataset	MiTM	ACC, PREC, REC, F1-S
siddharthan et al. [114]	2022	LR, KNN, SVM, NB, DT, RF, GB	SEN-MQTTSET dataset	DoS, Flooding	ACC,F1-S
Zeghida et al. [3]	2023	Ensemble Learning (bagging, boosting, stacking)	MQTTset	Flooding denial of service, MQTT Publish flood, SlowITe, Malformed data, Brute force authentication	ACC, F1-S, MCC
Musleh et al. [115]	2023	KNN, RF, SMO, VGG-16, DenseNet, stacking	IEEE Dataport image	Malicious traffic	ACC, PREC, REC,F1-S
Javed et al. [116]	2024	XGBoost, RF, DT, MLP, ANN, LSTM, Conv1d	TON_IoT Dataset	DoS, DDoS, Port scanning, MITM, XSS/injection, Brute force, Backdoor, Ransomware	ACC, PREC, REC,F1-S

Continued on next page

Table 3.1 – continued from previous page

Study	Year	ML used	Data sets	Attacks	Evaluation Metrics
Rakha et al. [117]	2024	LR, KNN, NB, DT, Adaboost, RF, Hybrid model	MQTT-IoT-IDS2020	-	ACC, PREC, REC, F1-S, AUC

3.3 Deep Learning-Based Intrusion Detection System for MQTT Environment

DL enhances IDSs by improving accuracy, efficiency, and adaptability. It excels at identifying complex and unknown threats, supports continuous self-learning, and enables real-time analysis of high-dimensional data, reducing risks and simplifying security management. These advantages make DL-integrated systems highly effective for network protection, with significant research advancing this field.

Ferrag et al. [118] conducted a comprehensive analysis of DL techniques for intrusion detection in cybersecurity. They categorized 35 prominent cyber datasets into seven groups based on 15 characteristics, providing valuable insights into IDS data. Using real traffic datasets like CSE-CIC-IDS2018 and Bot-IoT, they evaluated DL models' performance and compared them to traditional ML methods, focusing on false alarm rate, accuracy, and detection rate as key metrics.

Mosaiyebzadeh et al. [119] proposed a deep learning-based Intrusion Detection System utilizing DNN, CNN-RNN-LSTM, and LSTM models. They trained their models using the MQTT-IoT-IDS2020 dataset, a public dataset containing MQTT attacks. The authors evaluated the performance of their DL-based network IDS using standard metrics such as accuracy, precision, recall, F1 score, and weighted average. The results demonstrated high accuracy and F1 score in detecting MQTT attacks, with an average accuracy of 97.09% and an F1 score of 98.33%.

In 2021, Imtiaz and H. Mahmoud [120] proposed a deep learning-based approach to enhance cybersecurity in IoT environments. They developed a model using 1D, 2D, and 3D convolutional neural networks to analyze malicious network traffic at the packet level and extract spatial features. Their methods demonstrated exceptional performance, achieving over 99% accuracy on several datasets, including BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, IoT-23, IoT-DS-1, and IoT-DS-2.

Alzahrani and Aldhyani [121] introduced an IDS using AI algorithms to address growing cybersecurity threats in IoT networks. They assessed the performance of KNN, Linear Discriminant Analysis (LDA), CNN, and CNN-LSTM models in detecting intrusions within the MQTT protocol IoT environment. Testing on a dataset with various attacks, they found the CNN-LSTM model to be the most effective, achieving an accuracy of 98.94%.

Naser et al. [122] proposed a hybrid deep learning model that integrated CNN and LSTM techniques. This model was specifically designed to detect cyber-attacks in wireless sensor networks (WSNs). By combining CNN and LSTM, the hybrid model aimed to enhance the classification of various attack types in the MQTT2020 dataset, achieving superior predictive performance compared to traditional models that utilize either CNN or LSTM alone. Inte-

grating these two techniques significantly improved the accuracy of cyber-attack detection in WSNs, which achieved more than 95% .
Zeghida et al. [4] proposed a deep learning-based IDS to identify malicious behavior during communication between IoT devices using the MQTT protocol. The authors focused on addressing the vulnerabilities of the MQTT protocol, particularly concerning DoS attacks. They used single and hybrid DL classifiers (LSTM, CNN, GRU, CNN-RNN, CNN-LSTM, CNN-GRU), and their experimental results demonstrate the proposed approach's effectiveness with an accuracy rate exceeding 99% and a very small loss rate.
Otoom et al. [123] developed a DL model to detect brute force attacks in MQTT-IoT networks using the MQTT-IoT-IDS2020 dataset. After preprocessing to create Bi-flow and Uni-flow feature sets, their model achieved over 99% accuracy in distinguishing normal traffic from attacks.
Dandapat and Mondal [124] proposed an IDS combining a Genetic Algorithm (GA) for feature selection with a CNN for classification, optimizing accuracy and efficiency by focusing on the most relevant features. Their model achieved classification accuracy rates of 85.50% for packet features, 99.51% for uni-directional flow features, and 99.33% for bi-directional flow features, surpassing existing solutions. The system also excelled in metrics like precision, sensitivity, specificity, and false positive rate, making it suitable for real-time IoT network deployment.
Saiyed and Al-Anbagi [125] introduced DEEPShield, a deep ensemble learning system designed to detect high- and low-volume DDoS attacks in resource-constrained environments. Combining CNN and LSTM architectures, DEEPShield achieved high detection accuracy while minimizing processing time and resource usage. Optimized for edge computing through unit pruning, the system maintained accuracy while reducing complexity. Validated on the HL-IoT dataset and others like HICT, ToN-IoT, CICIDS-17, and ISCX-12, DEEPShield demonstrated adaptability, achieving 90% accuracy, up to 40% faster operation, and 30% reduced memory usage, outperforming existing models.
The Table 3.2 below summarizes the recent related work based on DL techniques.

Table 3.2: Summary of Related Work Based on Deep Learning

Study	**Year**	**DL used**	**Data sets**	**Attacks**	**Evaluation Metrics**
Ferrag et al. [118]	2020	RNN, DNN, CNN, Restricted Boltzmann Machines, Deep Belief Networks, Deep Boltzmann Machines, Deep Autoencoders	CSE-CIC-IDS2018, Bot-IoT	DoS, R2L, U2R, Probe, DDoS, OS, Service Scan, Keylogging, Data exfiltration, Botnet	ACC, PREC, REC, F1 score, False Alarm Rate, ROC

Continued on next page

Table 3.2 – continued from previous page					
Study	**Year**	**DL used**	**Data sets**	**Attacks**	**Evaluation Metrics**
Mosaiyebzadeh et al. [119]	2021	DNN, CNN-RNN-LSTM, LSTM	MQTT-IoT-IDS2020	Aggressive scan, UDP-scan, MQTT brute-force	ACC, PREC, REC,F1-S
IMTIAZ and H. Mahmoud [120]	2021	CNNs (1D, 2D, 3D)	BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, IoT-23, IoT-DS-1, IoT-DS-2	BoT-IoT, MQTT attacks, IoT-23 attacks, IoT NI attacks	ACC, PREC, REC,F1-S
Alzahrani and Aldhyani [121]	2022	KNN, LDA, CNN, CNN-LSTM	MQTTset	Brute-force, flooding, malformed packet, SlowITe	ACC, PREC, REC, F1-S, Support, Time, Weighted average
Naser et al. [122]	2022	CNN, LSTM, CNN-LSTM	MQTTset	Not specific	ACC, loss
Zeghida et al. [4]	2023	LSTM, CNN, GRU, CNN-RNN, CNN-LSTM, CNN-GRU	MQTT dataset	DoS	ACC, F1-S, Model loss
Otoom et al. [123]	2023	DL	MQTT-IoT-IDS2020	Brute force	ACC, PREC, REC, F1-S, FPR
Dandapat and Mondal [124]	2024	Genetic Algorithm (GA), CNN	MQTT-IoT-IDS2020	Aggressive scan, UDP-scan, MQTT brute-force attack	ACC, F1-S, FP, PREC, SENS, SPEC, MCC
Saiyed and Al-Anbagi [125]	2024	LSTM-CNN	HL-IoT, HICT, ToN-IoT, CICIDS-17, ISCX-12	DDoS	ACC, FP, PREC,F1-S

3.4 Generative Adversarial Networks-Based Intrusion Detection System for MQTT Environment

The integration of GAN into IDS marks a major step forward in cybersecurity. By utilizing adversarial learning, GANs generate synthetic attack data to improve detection model ro-

bustness and simulate complex cyberattacks. Recent research has focused on enhancing IDS frameworks with GAN-based models for more effective anomaly detection.

Shahriar et al. [126] proposed a GAN-based Intrusion Detection System (G-IDS) designed to address issues like imbalanced or missing data in network security datasets. Using the NSL KDD-99 dataset to model a Cyber-Physical System (CPS). The G-IDS demonstrated superior performance compared to standalone IDS models in attack detection and exhibited improved training stability.

Hara et al.[127] presented IDS on the NSL-KDD dataset that uses semi-supervised learning with the Adversarial Auto-Encoder(AAE). The proposed semi-supervised learning technique achieved accuracy equivalent to existing ML methods by reducing labeled data since it required less training data. Their methodology achieved an 82.78% identification rate while requiring only 1.0 % labeled data.

Duy et al. [128] proposed DIGFuPAS, a framework that generates attack samples to bypass ML-based IDS in Software-Defined Networking (SDN) environments using a black-box approach. DIGFuPAS utilizes WGAN to improve GAN training and ensures that attack traffic retains functional features for effective adversarial attacks. Tested on the NSL-KDD and CICIDS2018 datasets, it was shown to reduce the detection rate of black-box IDSs, causing misclassification in SDN networks. Additionally, DIGFuPAS serves as a tool to assess and enhance the robustness of IDS systems by enabling repeated retraining with generated attack traffic.

Hou et al. [129] explored the security of ML-based algorithms for identifying IoT devices and introduced IoTGAN, an attack technique designed to manipulate IoT device traffic to evade detection. IoTGAN uses a replacement model to mimic the target identification model in a black-box environment and trains a manipulative model to introduce adversarial perturbations into the traffic. Experiments showed that IoTGAN effectively disrupts IoT device identification.

Park et al. [130] developed an enhanced AI-based network IDS that addresses the data imbalance issue in Network Intrusion Detection Systems (NIDS). By using GANs to generate artificial data, their approach improved classification accuracy, particularly for less common attack categories. The method demonstrated significant accuracy improvements on benchmark datasets such as NSL-KDD and UNSW-NB15.

Ding et al. [131] proposed a hybrid model for abnormal traffic detection that combines Deep Autoencoder (DAE) and RF algorithms within a GAN framework. The DAE uses the Back Propagation (BP) algorithm for optimal dimensionality reduction, improving training efficiency. The RF algorithm constructs decision nodes based on the Gini coefficient and random sampling, aggregating results from multiple sub-trees for final classification. The model achieved high accuracy, with 99.8% in binary classification and 99.6% in multi-class classification experiments.

Boppana & Bagade [132] introduced GAN-AE, an unsupervised model combining GAN and autoencoders to detect unknown intrusions in MQTT IoT applications. Tested against other unsupervised models like autoencoders, One-Class SVM, and Isolation Forest, GAN-AE outperformed them, achieving a high accuracy and F1 score of 97% .

Mustapha et al. [133] investigated GANs' role in generating traffic resembling legitimate data to test ML/DL-based DDoS detection methods. They initially developed an LSTM-based detection model, which performed well against standard DDoS attacks. However, it struggled against adversarial DDoS traffic created by GANs. To address this, the researchers enhanced the detection model, achieving 91.75% to 100% accuracy in identifying GAN-generated ad-

versarial DDoS traffic.
Li et al. [134] developed HDA-IDS, a Hybrid DoS Attack Intrusion Detection System that combines signature-based and anomaly-based methods to detect both known and unknown attacks. For signature detection, it employs a stacking method using outputs from multiple ML models, while anomaly detection leverages a novel Conditional Generative Adversarial Network (CL-GAN) to identify zero-day attacks by analyzing temporal and data series properties. HDA-IDS demonstrated superior performance, achieving accuracy rates of 99.63% on NSL-KDD, 98.53% on Bot-IoT, and 98.75% on CICIDS2018 datasets.
Zeghida et al.[5] presented a significant advancement in enhancing IDS for IoT networks by addressing the challenges of imbalanced datasets. Their work utilized GAN techniques to generate high-quality synthetic data, creating balanced datasets that improve the detection accuracy of cyber attacks on the MQTT protocol. The authors propose three hybrid deep learning models—CNN-RNN, CNN-LSTM, and CNN-GRU—which demonstrated superior performance in multi-class classification and reduced false positives where the accuracy rate achieved 99 %, and the recall and precision achieved 100% in their three suggested hybrid models.
The Table 3.3 below summarizes the recent related work based on generative adversarial networks.

Table 3.3: Summary of Related Work Based on Generative Adversarial Networks

Study	**Year**	**Data Sets**	**Attacks**	**Evaluation Metrics**
Shahriar et al. [126]	2020	KDD'99 dataset	DoS, User to Root (U2R), Remote to Local (R2L), Probing Attack (PA)	Precision, Recall, F1 score
Hara et al.[127]	2020	NSL-KDD	Probe, DoS , U2R, R2L	Accuracy, False Negative Rate (FNR), False Positive Rate (FPR)
Duy et al. [128]	2021	NSL-KDD, CICIDS2018	DDoS, DoS, Botnet, Brute Force, Web attack, Heartbleed, Infiltration, U2R, R2L	Detection Rate (DR), F1 score, Evasion Increase Rate (EIR)
Hou et al. [129]	2021	UNSW	Traffic Manipulation, Adversarial Perturbations	Identification Rate, Spoofing Rate

Continued on next page

Table 3.3 – continued from previous page				
Study	**Year**	**Data Sets**	**Attacks**	**Evaluation Metrics**
Park et al. [130]	2022	NSL-KDD, UNSW-NB15, IoT Data Set, Real Data Set	XSS, DDoS, Brute Force, Injection, Trojan, Backdoor, Command and Control communications , Port Scanning	Accuracy, Precision, Recall, F1 score
Ding et al. [131]	2022	InSDN	DoS, DDoS, Web Attacks, R2L, Malware, Probe, U2R	Accuracy, Recall, Precision, Confusion Matrix (CM)
Boppana & Bagade [132]	2023	MQTT-IoT-IDS2020, Custom-built dataset	Aggressive Scan, UDP Scan, Sparta SSH Brute-force, Brute-force, Malformed packet, Invalid Subscribe/Publish, TCP Syn Flood, Port Scanning	Accuracy, Precision, Recall, F1-S
Mustapha et al. [133]	2023	CIC-DDoS2019, CICIDS-2017	DDoS	Accuracy, F1-S, ROC-AUC Curve, Confusion Matrix (CM)
Li et al. [134]	2024	CICIDS2018, Bot-IoT, NSL-KDD	DoS, Botnet	Accuracy, Precision, Recall, F1-S, Confusion Matrix (CM)
Zeghida et al. (2024)	2024	MQTTset	Malaria, DoS, MQTTFlood, SlowITe, Malformed, Brute-force	Accuracy, Model loss, Recall, Precision, Confusion Matrix (CM)

3.5 Conclusion

In conclusion, IDSs based on ML and DL technologies represent a forward-thinking and intelligent strategy for securing IoT networks. These systems leverage advanced analytics and adaptive algorithms to mitigate security risks, safeguard sensitive data, and ensure the seamless operation of interconnected IoT devices. This chapter has offered a comprehensive exploration of the latest advancements in IDS, with a focus on ML, DL, and GAN techniques, each contributing uniquely to strengthening IoT security, particularly within the context of the MQTT protocol.

ML-based IDS solutions remain pivotal, offering flexible and scalable security measures.

These systems excel in utilizing algorithmic efficiency to identify and neutralize threats, making them indispensable for real-time detection in dynamic, fast-evolving environments.
DL-based IDS has further advanced detection capabilities by uncovering complex data patterns and correlations that traditional ML methods may overlook. The ability of DL models to process vast amounts of data with exceptional accuracy positions them as critical tools for combating sophisticated and constantly evolving threats in IoT networks.
GAN-based IDS introduces a groundbreaking dimension to intrusion detection by harnessing the adversarial properties of GANs. These systems not only simulate malicious activities to train more robust detection models but also address challenges like imbalanced datasets by generating high-quality synthetic data. This dual capability enhances the resilience and robustness of security frameworks, enabling them to counteract novel attack vectors effectively.
Collectively, these innovations highlight the dynamic evolution of cybersecurity, where traditional methods are augmented by cutting-edge technologies to confront increasingly complex threats. The insights presented in this chapter underscore the necessity of continuous research and development, fostering ongoing innovation to secure IoT ecosystems in an ever-changing threat landscape.

Part 2: Contributions

Chapter 4

Securing MQTT protocol for IoT environment using IDS based on ensemble learning (First contribution)

4.1 Problem Statement

Traditional security techniques frequently fail to handle the specific issues caused by the dynamic nature of IoT settings. As a result, a sophisticated IDS is required to detect anomalies specific to MQTT traffic effectively.

4.2 Introduction

This chapter introduces our first contribution entitled "Securing MQTT protocol for IoT environment using IDS based on ensemble learning". This approach addresses the security risks associated with the privacy and confidentiality of low-power IoT devices using MQTT. To counter these threats, modern IDSs have been designed to strengthen the security of the MQTT protocol at the broker level.

Conventional detection methods that rely on a single ML technique often fall short due to the complexity and diversity of attacks. This chapter proposes an intrusion detection model that utilizes EL techniques, trained on a recent public dataset with various MQTT attacks. EL is an advanced ML technique that involves training multiple ML models independently on random subsets of the training data and then combining their predictions to achieve a final result.

We focused on balancing the data; for training and testing, we created a balanced binary MQTT dataset from the MQTTset dataset, instead of the commonly used imbalanced datasets in previous studies. In addition to traditional ML models (DT, RF, NB, MLP), we introduced three well-known EL methods: bagging, boosting, and stacking. EL principles enhanced predictive performance by leveraging the strengths of multiple distinct models. To our knowledge, this is the first application of ensemble learning methods to this MQTT dataset. The experimental results demonstrated that our EL-based network IDS improved detection accuracy and F1 score to 95%, with the MCC value exceeding 90%.

4.3 Methodology and Experimental Setup

4.3.1 Dataset Description

The MQTTset data set used in our study was developed by Vaccari et al. [6] and was publicly released on Kaggle in 2020. It serves as a vital resource for IDS research in the IoT domain, specifically focusing on detecting threats in the MQTT protocol. The data set simulates a smart home environment, it was collected from eight MQTT sensors monitoring factors such as humidity, carbon monoxide levels, light intensity, temperature, smoke, motion, fan operation, door locks, and speed control. Sensor data was transmitted at varying intervals based on the unique behavior of each sensor. This dataset contains both legitimate data and five distinct types of attack, including:

1. **Flooding Denial of Service (FDS)** aims to disrupt service availability for legitimate users.
2. **MQTT Publish Flood** which seeks to overwhelm system resources by using a single connection rather than multiple ones.
3. **SlowITe**, a new type of DoS attack.
4. **Malformed Data** involves creating and sending multiple corrupted packets to the broker.
5. **Brute force authentication** involves making as many attempts as possible to retrieve the user's login information.

The original dataset was divided into two subsets, with 70% allocated for training and 30% for testing. Our experiments aimed to determine whether traffic was legitimate or abnormal, without specifying the type of attack. To facilitate this, the data was reclassified into two categories: normal (legitimate) and abnormal (attack), enabling a binary classification approach. This reclassification yielded a balanced training set comprising 115,822 attacks and 115,824 legitimate instances. Similarly, the test set included 49,639 legitimate entries and 49,651 attack entries. The dataset includes 33 features, as detailed in Table 4.1.

4.3.2 Development Platforms

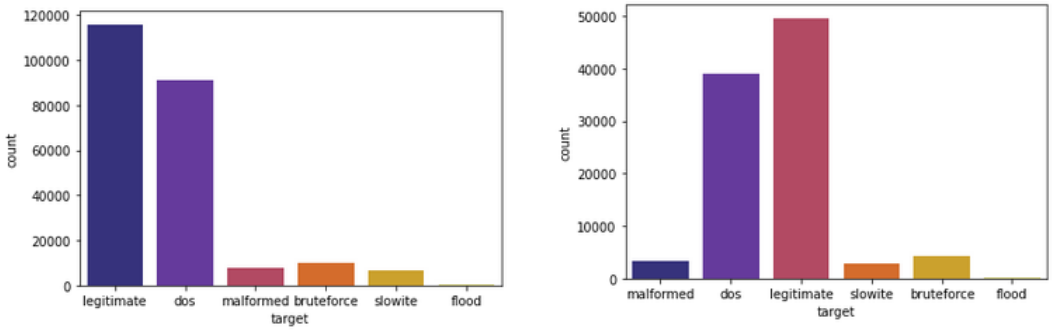
In our study, the proposed model was developed and implemented on the Google Colab platform, which provides access to a free GPU with 12 GB of capacity. Google Colab is a cloud-based platform widely recognized for its robust computational capabilities, interactive environment, and seamless integration with popular ML libraries. Its ability to facilitate collaboration and eliminate the need for local hardware setup makes it particularly advantageous for artificial intelligence research. Python was selected as the programming language because of its extensive use in ML and data analysis, as well as its rich ecosystem of libraries. Specifically, libraries such as Pandas and NumPy were employed for data preprocessing, while TensorFlow, Keras, and Scikit-learn were used for implementing and fine-tuning the algorithms. These frameworks were chosen for their scalability, ease of use, and support for building and evaluating ML models. This combination of tools and platforms ensured an efficient and flexible development process tailored to the requirements of the study.

No	Name	Description	Protocol Layer
1	tcp.flags	TCP flags	TCP
2	tcp.time_delta	Time TCP stream	TCP
3	tcp.len	TCP Segment Length	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	Message Length	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

Table 4.1: Table of MQTT and TCP features [6]

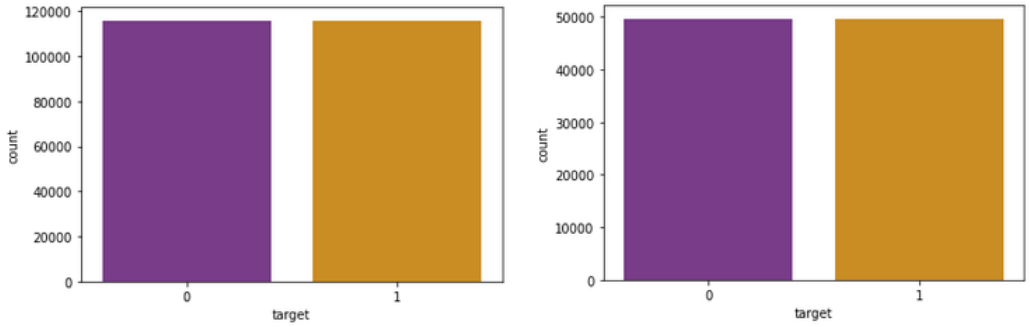
4.3.3 Preprocessing Data and Feature Engineering

The data preprocessing phase is a critical step in preparing raw data for ML applications, ensuring a balanced and high-quality dataset. Its primary goal is to eliminate faulty, inconsistent, or irrelevant data while enhancing data integrity to achieve fair and accurate results. Effective preprocessing also mitigates common issues such as overfitting, often caused by noisy or biased data. Key tasks include identifying data types, handling missing or duplicate values, and addressing imbalanced datasets to ensure uniformity in data representation.
In our study, Python was used to preprocess a complex dataset, starting with data profiling. This involved analyzing the dataset's quality by gathering essential statistics, such as whether features were numerical or categorical and whether they were normalized. Data profiling also assessed the balance between training and testing data to ensure representativeness. Additionally, we identified and removed empty or redundant features, such as columns with a single unique value, which contribute little to ML performance.
After profiling, data cleaning involved removing irrelevant features. For instance, the 'mqtt.conflags' feature was dropped due to a significant imbalance between the training and test sets, while the 'mqtt.msg' feature was excluded because of its large size and limited utility. We then transformed categorical features, such as 'tcp.flags' and 'mqtt.hdrflags' into numerical values using normalization and encoding techniques to ensure compatibility with ML algorithms that require numerical inputs.
Another key step was simplifying the target variable by reducing its classes from six to two. The original dataset represented various types of network activity, including both legitimate and attack data. To streamline the classification problem, we converted the target variable into a binary format: legitimate data was labeled as 0, and all attack data as 1, using a scaler function. This binary classification aligned with our goal of detecting any malicious activity without focusing on specific attack types.
Figures 4.2 and 4.1 illustrate this transformation. Figure 4.1 shows the original target vari-



(a) Train data (b) Test data

Figure 4.1: Data visualization before the preprocessing phase [3]



(a) Train data (b) Test data

Figure 4.2: Data visualization after the preprocessing phase [3]

able with six classes, while Figure 4.2 displays the post-processed target with two categories: 115,822 instances of attack data labeled as 1 and 115,824 instances of legitimate data labeled as 0 in the training set. This reduction simplified the classification task and improved the model's ability to distinguish between normal and malicious activities.
In summary, this comprehensive preprocessing phase resulted in a clean, balanced, and fully numerical dataset, optimized for machine learning. These improvements enhanced the model's performance, accuracy, and reliability, ensuring a smooth and effective validation process in the subsequent stages of training and testing.

4.3.4 Proposed Approach

The proposed approach in this study aims to enhance the security of the MQTT protocol in IoT environments by implementing an IDS based on EL techniques. The research focuses on a binary classification task to analyze MQTT protocol network traffic, distinguishing between normal and abnormal traffic patterns. The primary objective is to improve the efficiency of the MQTT protocol's IDS by leveraging a range of ML and EL models to detect attacks targeting IoT devices.
The models used include DT, RF, NB, MLP, Bagging, AdaBoost, HistGradientBoost, XGBoost, and Stacking. This study emphasizes the use of EL approaches, which combine multiple classifiers to harness their complementary strengths and mitigate various error sources, resulting in improved accuracy and reliability for intrusion detection. By preprocessing the

dataset and implementing advanced algorithms, the study aims to achieve superior detection performance.
To evaluate the effectiveness of the proposed models, key metrics such as the ACC, the F1 score, and the MCC are used. These metrics provide a comprehensive assessment of the models' performance and their suitability for securing MQTT-based IoT environments.

4.3.5 Ensemble Learning Models

To protect an IoT system from attacks, the detection mechanism must accurately identify malicious behaviors. In this section, we introduce our novel method, which utilizes EL models to detect attacks and defend the system. Various EL algorithms were employed for data analysis and validation, and the details of these techniques are explained below.

1. **Bagging:** is an EL technique that trains multiple models on randomly selected subsets of the training data (Figure 4.3) and combines their predictions through a voting mechanism. The final output is determined by averaging the predictions on the testing sets. This approach improves model accuracy while minimizing the risk of increased variance. Bagging can be implemented in Python using the sklearn.ensemble library [135]. Figure 4.4 illustrates a code snippet for creating a bagging ensemble classifier with a base estimator and its associated hyperparameters [135].

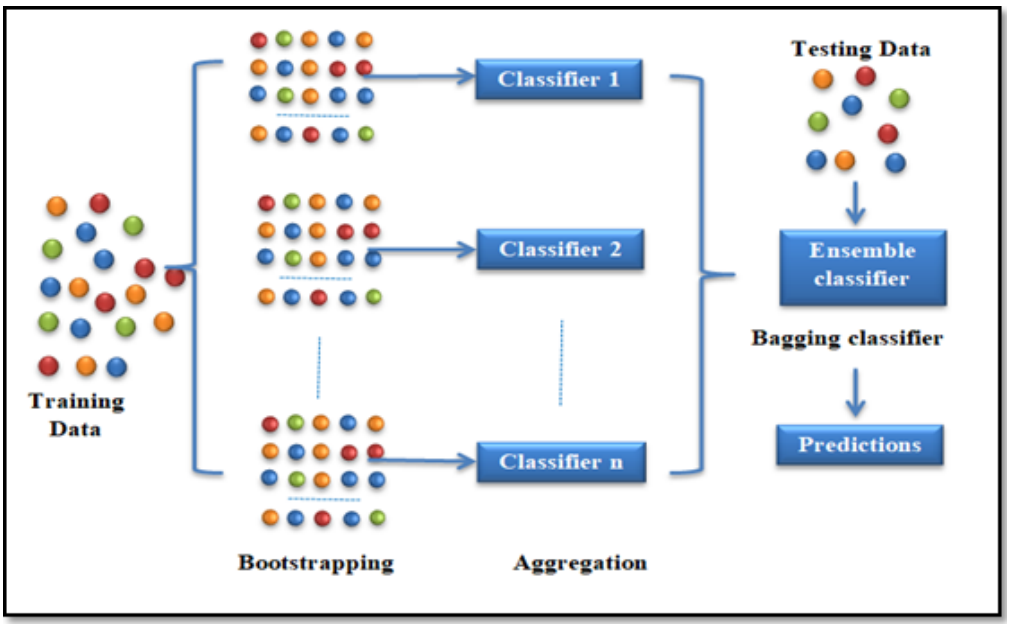


Figure 4.3: Bagging Classifier [3]

- **base_estimator**: This is the algorithm that should be used for each random subset of the dataset. In our instance, we applied the decision tree algorithm.
- **n_estimators**: The total number of base estimators used in training by the ensemble. We selected 500 estimators.
- **max_samples**: shows the number of data samples that will be utilized to train each base estimator. We selected 100 samples.

```
from sklearn.ensemble import BaggingClassifier
model_Bagging=BaggingClassifier(base_estimator
=DecisionTreeClassifier(random_state=42),
n_estimators=500, max_samples=100, bootstrap =True ,random_state=42).
model_Bagging.fit(X_train, y_train)
y_pred = model_Bagging.predict(X_test)
```

Figure 4.4: Bagging Classifier Code[3]

- **bootstrap**: a Boolean parameter that determines whether replacement is used (True case) or not (False case) while drawing the samples. We have "True" for it in our code.
- **random_state**: This parameter is used to determine the random seed that the algorithm will use while training on a different subset of data that is picked at random. In our experience, the value 42 indicates that the same random number will be utilized each time the method is executed to generate the same data subsets.

2. **Boosting:** The goal of this classifier is to develop multiple models by training several weak models in sequence. Each subsequent model aims to correct the errors of the previous one (Figure 4.5). As a result, this process produces a set of complementary models where the strengths of others balance the weaknesses of some. To achieve this classification, we utilized three primary boosting algorithms: AdaBoost, histogram-based gradient boosting, and the XGB Classifier model. The code snippets in Figure 4.6 demonstrate how to build these classifiers with updated hyperparameters (max_iter and max_leaf_nodes) [135].

   - **max_iter**: The maximum number of boosting iterations. For binary classification, the maximum number of trees can be set. In our scenario, we selected 500 iterations.
   - **max_leaf_nodes**: The maximum number of leaves that each tree can have. In our code, we specified 100 leaves.

3. **Stacking:** Stacking is a strategy for EL that uses a meta-classifier to merge many classification models. Initially, individual classifiers are trained on the whole training set. The meta-features of the various classifiers are then utilized as input to the meta-classifier. The final forecast is calculated using a classifier (see Figure 4.8). Stacking enables the use of the strongest characteristics of each estimate (base classifiers) as input for a final estimator (final classifier) [135]. In our contribution, we layered the final model using the estimator's list of decision trees (dt), random forests (rf), multi-layer perceptrons (mlp), and naive bayes. And for the final_estimator hyper parameters (which are a classifier that will be used in the base estimator's combination), we picked "Logistic Regression" approach. The code in Figure 4.7 demonstrates how to build a stacking classifier using these hyper parameters:

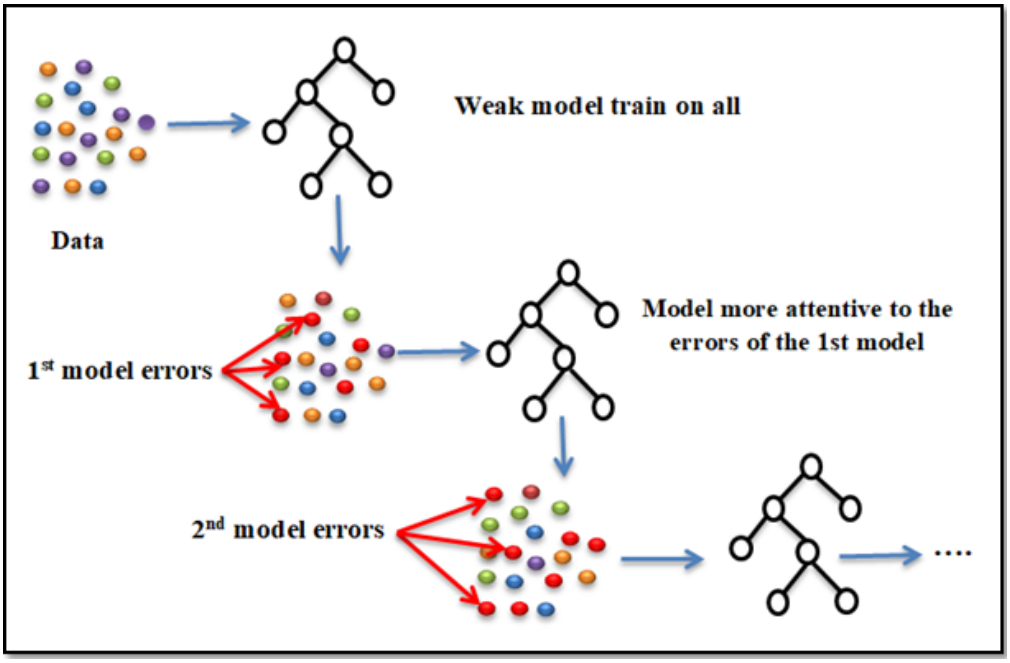


Figure 4.5: Boosting Classifier [3]

```
from sklearn.ensemble import
AdaBoostClassifier
model_AdaBoost = AdaBoostClassifier(
n_estimators=100,random_state=42)
model_AdaBoost.fit(X_train, y_train)
y_pred = model_AdaBoost.predict(X_test)
```

```
from sklearn.ensemble import
HistGradientBoostingClassifier
model_HGB = HistGradientBoostingClassifier(
max_iter =500, max_leaf_nodes =100,
random_state=42)
model_HGB.fit(X_train, y_train)
y_pred =model_HGB.predict(X_test)
```

```
from xgboost import XGBClassifier
model_XGB = XGBClassifier()
model_XGB.fit(X_train,y_train)
y_pred = model_XGB.predict(X_test)
```

Figure 4.6: AdaBoost, Histogram-based Gradient Boosting, and XGBClassifier Codes [3]

4.4 Results and Discussion

4.4.1 Performance Comparison

This section shows the results of our selected models. Table 4.2 compares the accuracy, F1 score, MCC, and execution time of each selected approach for the experiment conducted by Vaccari et al. in [6] on the MQTT dataset to the outcomes of our suggested system.

```
from sklearn.ensemble import
StackingClassifier
from sklearn.linear_model import
LogisticRegression
estimator_list = [ ('dt',dt), ('rf',rf),
('mlp',mlp),('nb',nb) ]
model_stack = StackingClassifier(estimators=
estimator_list, final_estimator=
LogisticRegression(solver='liblinear',
random_state=42))
model_stack.fit(X_train,y_train)
y_test_pred = model_stack.predict(X_test)
```

Figure 4.7: Stacking Classifier code [3]

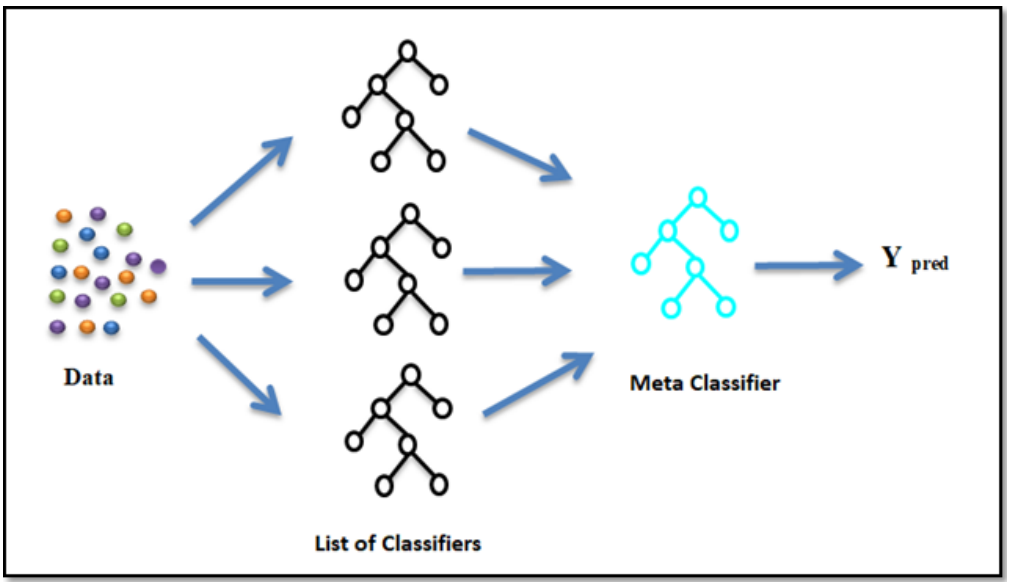


Figure 4.8: Stacking Classifier[3]

Our analysis highlights significant improvements in model performance and efficiency. For example, the DT algorithm achieved a remarkable increase in accuracy, rising from 91.59% to 94.55%, and a corresponding enhancement in the F1 score, improving from 91.40% to 94.54%. These performance gains were accompanied by a dramatic reduction in execution times, with training time decreasing from 148.81 seconds to just 0.29 seconds, and test time dropping from 2.30 seconds to 0.01 seconds. These results underscore the efficiency and effectiveness of our implementation.

Similarly, the RF model exhibited substantial performance improvements, with accuracy increasing from 91.59% to 95.38% and the F1 score advancing from 91.40% to 95.37%. Execution times were also significantly optimized, with training time reduced from 2298.27 seconds to 9.70 seconds and test time from 125.85 seconds to 0.81 seconds. These reductions in execution time further emphasize the computational efficiency of our proposed algorithm compared to the findings of Vaccari et al.

In contrast, the NB algorithm, although showing improvements in accuracy from 64.38% to 80.42% and F1 score from 68.72% to 79.64%, still underperformed compared to other meth-

Table 4.2: Results obtained from the MQTT dataset.

Models	**Vaccari et al [6]**					**Our Work**				
	Accuracy%	F1 score%	MCC%	Train time (s)	Test time (s)	Accuracy%	F1 score%	MCC%	Train time(s)	Test time(s)
Decision tree	91.59	91.40	-	148.81	2.30	94.55	94.54	89.52	0.29	0.01
Random forest	91.59	91.40	-	2298.27	125.85	95.38	95.37	90.96	9.70	0.81
Naïve bayes	64.38	68.72	-	85.28	13.78	80.42	79.64	66.12	0.16	0.04
Multilayer perceptron	90.38	90.18	-	5714.48	27.28	84.55	84.18	72.64	43.88	0.10
Bagging	-	-	-	-	-	95.38	95.37	90.96	23.52	8.30
AdaBoost	-	-	-	-	-	95.38	95.37	90.96	17.05	1.93
HistGradientboost	-	-	-	-	-	95.38	95.37	90.96	10.11	0.73
XGB Classifier	-	-	-	-	-	95.38	95.37	90.96	5.83	0.12
Stacking	-	-	-	-	-	95.43	95.42	90.97	250.05	0.94

ods. Its minimal training time, which decreased from 85.28 seconds to 0.16 seconds, suggests that faster training does not necessarily correlate with superior performance. This highlights the importance of a comprehensive training phase to optimize model efficacy, which is a key focus of our proposed approach.
Further analysis of the MLP reveals a decline in both accuracy from 90.38% to 84.55% and F1 score from 90.18% to 84.18%, yet it achieved substantial reductions in training time from 5714.48 seconds to 43.88 seconds and test time from 27.28 seconds to 0.10 seconds. This indicates that while the model may be less accurate, it benefits from improved efficiency in terms of execution time.

Ensemble learning methods, including Bagging, AdaBoost, HistGradientboost, XGB Classifier, and Stacking, consistently outperformed traditional algorithms. Stacking, in particular, achieved the highest accuracy and F1 score, both exceeding 95%, and an MCC greater than 90%. However, it is important to note that Stacking involves the training of multiple models, resulting in the longest training time. Despite this, the overall performance improvement validates the effectiveness of EL techniques. The XGB Classifier, while delivering results comparable to the RF in accuracy, F1 score, and MCC, offered better training time efficiency. The DT method, on the other hand, provided the best training and test times, at 0.29 seconds and 0.01 seconds respectively.
The confusion matrices shown in Figure 4.9 further substantiate the efficacy of our model in improving the number of true positives and true negatives, indicating a robust ability to identify samples correctly. However, the presence of false positives and false negatives suggests areas for further refinement to enhance overall accuracy.
In summary, our proposed scheme demonstrates significant improvements in model performance and computational efficiency. The effective use of ensemble learning techniques, combined with robust dataset preparation and pre-processing, has notably enhanced the detection methods' performance metrics. Ensemble models' ability to integrate multiple approaches effectively addresses various error sources and mitigates overfitting, thus improving the overall efficiency and reliability of the Intrusion Detection System.

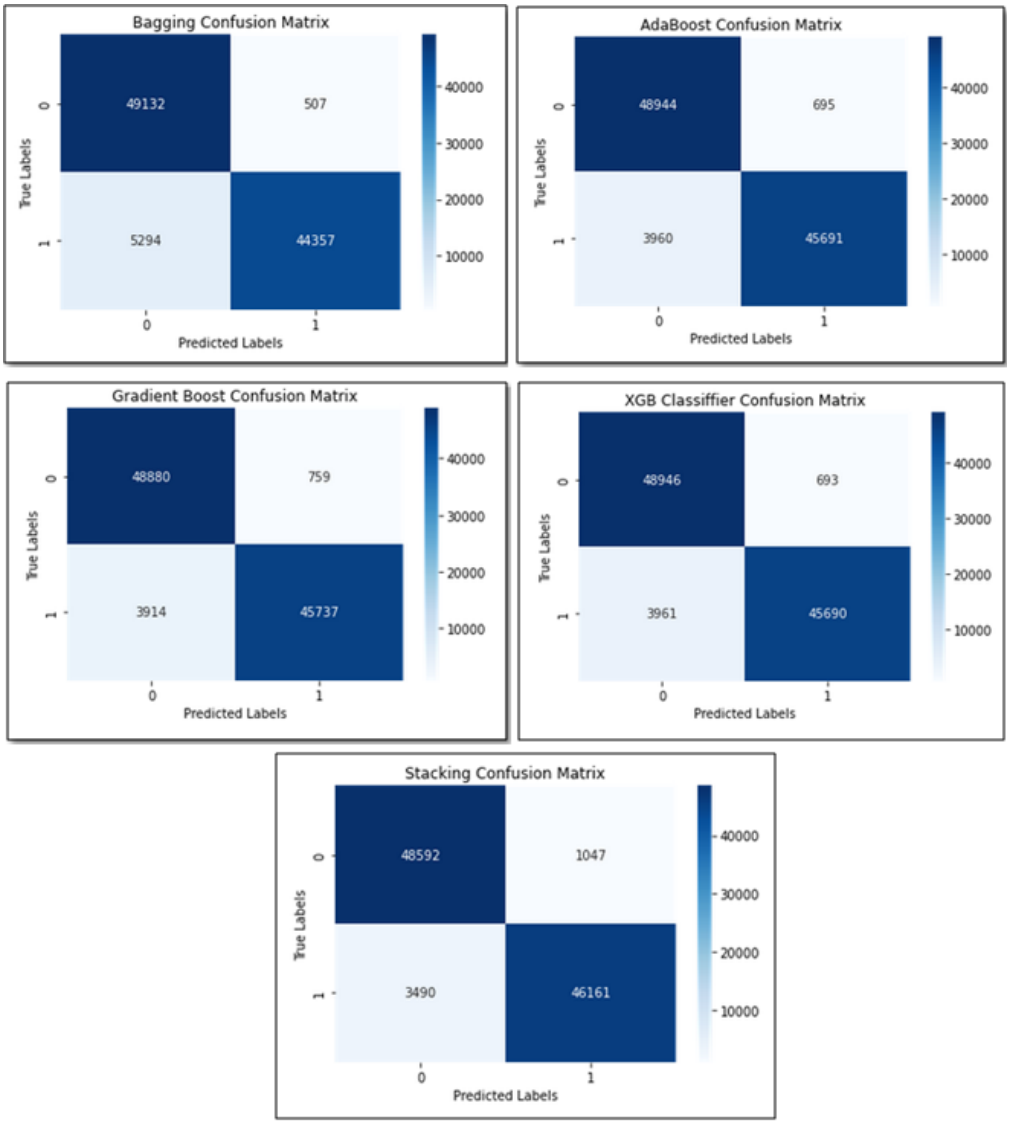


Figure 4.9: Confusion Matrices of Ensembles learning models [3]

4.4.2 Issues and Challenges

Despite the significant improvement in results achieved with our methodology compared to those reported by Vaccari et al. [6], particularly in terms of accuracy and execution time, several challenges were encountered. The data pre-processing phase required meticulous examination and refinement to produce high-quality data and, consequently, reliable outcomes. Additionally, the EL methodology demands robust and rapid processing capabilities, which is why we utilized Google Colab to achieve these results. The EL technique is less interpretable, and predicting the outputs of the aggregated model poses a significant challenge. The insights derived from the individual models within the ensemble are not easily comprehensible to the user. Furthermore, selecting appropriate models for the EL approach can be problematic, as an incorrect choice may result in lower accuracy compared to using a single model.

4.5 Conclusion

In this chapter, we introduced a novel approach to securing MQTT protocol traffic within IoT ecosystems, employing a robust ML and EL framework to enhance performance. We developed and assessed a variety of ML algorithms, specifically DT, RF, NB, MLP, Bagging, AdaBoost, HistGradientBoost, XGB Classifier, and Stacking to classify MQTT traffic as either normal or abnormal. To facilitate this, we proposed generating a balanced binary dataset from the MQTTset dataset, which was used to train and test IDSs designed for IoT environments. Our experiments demonstrated that the proposed models significantly improve IDS efficiency, achieving accuracy and F1 score exceeding 95%, with MCC values above 90%, while also reducing execution time. Notably, the XGB Classifier ensemble learning model exhibited superior performance compared to others in terms of both training and testing times. Ensemble Learning, as a powerful ML technique, consistently delivers high-quality results across various applications by combining the predictions of multiple models to enhance accuracy and robustness. As ML technology evolves and access to new data and computing resources expands, EL remains a promising and dynamic area of research with exciting future developments.

Chapter 5

Detection of DoS attacks in the MQTT environment (Second contribution)

5.1 problem Statement

The MQTT protocol exhibits significant security vulnerabilities, including plain text data transmission and the reliance on unencrypted communication ports, which expose MQTT systems to various cyber threats, particularly DoS attacks. Existing IDSs often utilize traditional ML techniques that may not effectively counter these evolving threats. Therefore, there is a critical need for a novel deep learning-based intrusion detection system that can accurately identify and mitigate malicious activities in MQTT communications, ensuring the security and reliability of IoT services.

5.2 Introduction

The widespread adoption of IoT has significantly improved machine-to-machine communication, with MQTT emerging as a preferred protocol due to its lightweight nature and low resource demands. However, its increasing use in critical sectors such as healthcare and manufacturing has raised serious security concerns. The inherent vulnerabilities of MQTT, including unencrypted data transmission and reliance on exposed ports, make it susceptible to cyber threats, particularly DoS attacks. These risks are compounded by the lack of awareness of security among users and developers. To address these issues, this chapter explores the implementation of a deep learning-based intrusion detection system to detect and mitigate DoS attacks on MQTT brokers. The proposed approach, tested on an open-source MQTT dataset, demonstrates superior performance, achieving over 99% accuracy and an F1 score exceeding 98%, thereby contributing to the development of robust security measures for MQTT-based IoT systems.

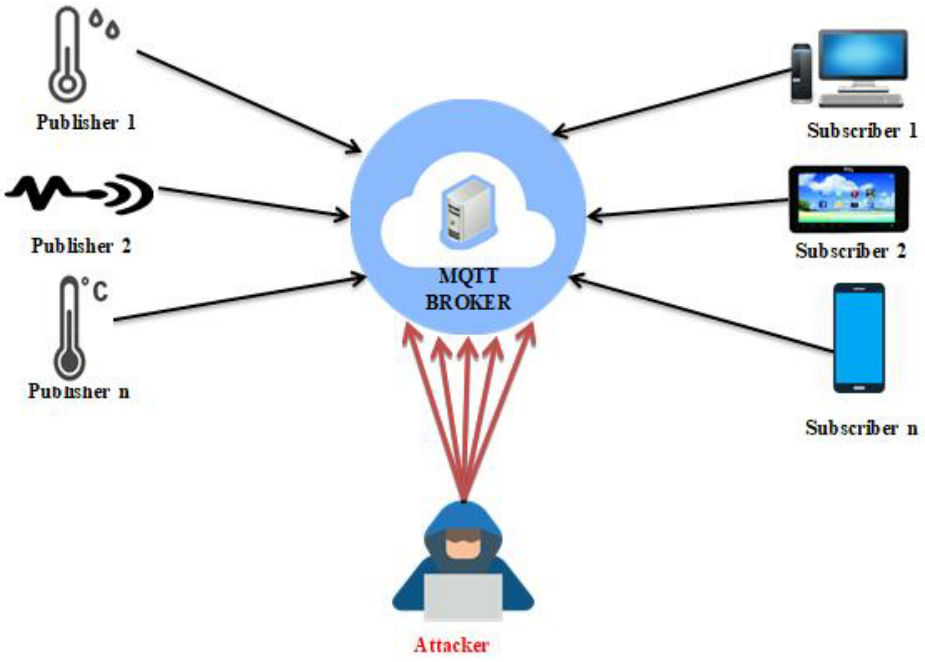


Figure 5.1: MQTT Denial of Service attack[4]

5.3 Methodology and Experimental Setup

5.3.1 Dataset Description

Ghannadrad, A. [7] created the MQTT dataset utilized in our study; this data effectively aids in detecting and classifying DoS assaults in MQTT sensor networks. The dataset was created by replicating a smart home setting with eight IoT sensors, which enabled legal and malicious traffic development. He used Wireshark to capture network activity, which enabled him to collect a large amount of flow-level data. This data was then processed to extract important properties, yielding an initial dataset including 44 flow-level characteristics. He then utilized the ANalysis Of VAriance (ANOVA) feature selection approach to reduce the dataset, resulting in a more concentrated version with the top ten characteristics for classification. The resulting dataset was labelled for binary classification, which distinguishes between malicious and lawful flows, as well as multi-value classification, which identifies legitimate traffic and multiple DoS attack types, which are Heavy-Flood, Fast-Flood, and Connect-Flood. The top 10 characteristics from each of the two datasets are described in Tables 5.1 and 5.2.

5.3.2 Preprocessing Data and Feature Engineering

The preprocessing and feature engineering phase is crucial to transform raw data into a format suitable for ML algorithms. We utilized the Python programming language on the Colab platform to preprocess our dataset. The preprocessing phase starts with loading a dataset that contains network traffic features, which provide insights into network behavior and help differentiate between normal and malicious activities.
The dataset used in our study comprises two types of data: binary data with 1,000 rows and

Feature	**Description**	**ANOVA Score**
State-CON	Reported transaction is active	1091.797346
State-FIN	Reported transaction is closed	773.240791
DstRate	Destination pkts per second	268.743461
SrcRate	Source pkts per second	252.760566
Rate	Pkts per second	172.975403
TcpRtt	TCP connection setup round-trip time	133.061976
AckDat	The time between the SYN-ACK and the ACK packets	130.633730
SynAck	The time between the SYN and the SYN-ACK packets	120.424206
Load	Bits per second	32.709531
Dur	Record total duration	31.573044

Table 5.1: Binary Classification: Top-10 Best Features dataset [7]

Feature	**Description**	**ANOVA Score**
Flgs-e *	Both Src and Dst loss/retransmission	1032.465791
Flgs-e	Ethernet encapsulated flow	448.155430
DstLoss	Destination pkts retransmitted or dropped	400.635048
State-FIN	Reported transaction is closed	252.980545
Flgs-e s	Src loss/retransmissions	248.745935
pLoss	Percent pkts retransmitted or dropped	207.513853
Load	Bits per second	159.877985
State-CON	Reported transaction is active	159.767769
DstPkts	Src $\rightarrow$ Dst packet count	158.117338
DstBytes	Dst $\rightarrow$ Src transaction bytes	146.474182

Table 5.2: Multi-value classification: Top-10 Best Features dataset [7]

11 columns and multi-value data with 960 rows and 11 columns.
To prepare the data, it is first divided into two parts: the input features (the first 10 columns) are stored in a DataFrame ("X"), while the 11^{th} column, the "Class" column, serves as the target variable stored in "Y", indicating whether the traffic is normal or malicious. Since the target variable is categorical, one-hot encoding is applied to convert it into a binary matrix. In the binary data, the traffic types are encoded as 0 for malicious and 1 for legitimate. For multivalue data, the four traffic types -Heavy-Float, Fast-Float, Connect-Float, and Legitimate- are encoded as 0, 1, 2, and 3, respectively.
Subsequently, the data is split into training and test sets using a 70-30 split, allowing the model to be trained on 70% of the data and tested on the remaining 30% to evaluate performance. The data is also shuffled to prevent ordering bias. These preprocessing steps ensure that the model receives clean, well-structured input, enhancing its ability to learn underlying patterns in network traffic and improving its accuracy in identifying anomalies. This approach is essential for improving data quality and ensuring the model's generalizability across various network traffic scenarios.

5.3.3 Proposed approach

Our proposed approach involves the application of several DL models, including single DL, such as LSTM, CNN, GRU, and hybrid DL, such as CNN-RNN, CNN-LSTM, and CNN-GRU architecture, for binary and multi-classification tasks. Each model trains on its respective data and leverages its unique structure to handle sequential data, aiming to identify patterns contributing to accurate classification.

- **LSTM model :** The first model, an LSTM, was selected for its ability to capture long-term dependencies in sequence data, and two variants were developed for binary and multi-class classification tasks. The binary classification model consisted of an LSTM layer with 32 units, a 50% dropout layer to prevent overfitting, and a dense output layer with 2 units using a sigmoid activation function. It was compiled with binary cross-entropy loss and the Adam optimizer with a learning rate of 0.001. The multi-class classification model had a similar architecture but included an additional dense layer with 4 units and a softmax activation function in the final layer, paired with categorical cross-entropy loss. Both models used the Adam optimizer and included an early stopping mechanism to monitor validation loss and prevent overfitting.

- **CNN model :** This model, typically used for image processing, was adapted in this approach to extract local patterns from sequential data. Two CNN models were developed for binary and multi-class classification tasks. The binary classification model followed a "Sequential" architecture, starting with a 1D convolutional layer with 32 filters and a kernel size of 2, using ReLU activation. This was followed by a max pooling layer, another convolutional layer with the same configuration, and a flattening layer. The output was passed through a dense layer with 8 units before a final dense layer with 2 units and a sigmoid activation function to produce binary outputs. The model was compiled with the binary cross-entropy loss function. For multi-class classification, a similar architecture was used, but with 64 filters in the convolutional layers and a softmax activation function in the output layer to handle four classes. This model is compiled with categorical cross-entropy. Both models were trained using the Adam optimizer with a learning rate of 0.001.

- **GRU model:** This model, a simplified variant of the LSTM, was used due to its computational efficiency while still capturing temporal dependencies in data. Two GRU models were developed for binary and multi-class classification tasks. The binary classification model features a sequential architecture with a GRU layer of 32 units, followed by a 50% dropout layer to prevent overfitting and two dense layers. The final predictions were made via a sigmoid activation function, with two output units. The model was compiled using binary cross-entropy loss and the Adam optimizer with a learning rate of 0.001. For multi-class classification, a similar GRU architecture was used, with an additional dense layer containing 8 units and a ReLU activation function before the output layer, which used softmax for multi-class predictions. This model was compiled with categorical cross-entropy loss.

- **CNN-RNN model:** The hybrid CNN-RNN model combines convolutional and recurrent layers to benefit both feature extraction and temporal modeling. For binary

classification, the model started with a Conv1D layer of 32 filters to extract local features, followed by batch normalization to stabilize training. A SimpleRNN layer with 20 units captured temporal dependencies, and dropout is applied for regularization. The output was processed through two dense layers: one with ReLU activation and another with sigmoid activation for binary classification with two outputs. The model was optimized using the Adam optimizer and binary cross-entropy loss.
For multi-class classification, the architecture was adjusted to handle four output classes. This version used a Conv1D layer with 64 filters, batch normalization, and a SimpleRNN layer with 32 units to capture temporal relationships. The fully connected layers consisted of two dense layers with 32 and 16 units, using ReLU activation, and the final layer was a dense layer with 4 units and a softmax activation for class probabilities. This model is compiled with categorical cross-entropy loss. Both models use early stopping to prevent overfitting by monitoring validation loss. This hybrid approach, combining CNN for feature extraction and RNN for sequential modeling, is well-suited for tasks requiring both spatial and temporal data analysis.

- **CNN-LSTM model:** It is a hybrid architecture combining CNN and LSTM networks for time-series classification, utilizing both spatial and temporal feature extraction. The model started with a 1D Convolutional layer ("Conv1D") using 32 filters with a kernel size of 3 to capture local features, followed by an "AveragePooling1D" layer to reduce dimensionality and retain key information. A "Dropout" layer with a 0.01 rate was applied to prevent overfitting. The output of the CNN layers is passed through two LSTM layers with 128 and 32 units, respectively. The first LSTM layer outputs the entire sequence to allow the second LSTM to process temporal dependencies. The CNN-LSTM combination was followed by a "Flatten" layer to convert the sequence data for the fully connected ("Dense") layers. For binary classification, the output layer used a "sigmoid" activation, while "softmax" was employed for multi-class classification. The model was compiled using the "Adam" optimizer with a learning rate of 0.001 and used "binary_crossentropy" or "categorical_crossentropy" loss functions, depending on the task. This architecture effectively captures local patterns via CNN and long-term temporal dependencies via LSTM, making it suitable for sequence classification tasks.

- **CNN-GRU model :** The proposed CNN-GRU model combines CNN with GRU for sequential data processing. In the binary classification version, the architecture started with a 'Conv1D' layer with 32 filters to extract local features, followed by a 'MaxPooling1D' layer to down-sample and reduce dimensionality. A dropout layer was applied to prevent overfitting. The sequence was processed by two GRU layers, with the first GRU layer (64 units) configured to return sequences, allowing for temporal dependency capture, and the second GRU layer (32 units) further processing the output. The output was flattened and passed through a dense layer with 32 units and ReLU activation, followed by another dropout layer. The final output layer used a sigmoid activation for binary classification, and the model was optimized with the Adam optimizer and binary cross-entropy loss.
  The architecture mirrored the binary model for multi-class classification but was adjusted for four categories. It used a final output layer with four units and a softmax activation to output probabilities for the four classes, with categorical cross-entropy loss used for training. Overall, the CNN-GRU model effectively combines CNNs for feature

extraction and GRUs for capturing temporal dependencies, making it well-suited for both binary and multi-class classification tasks.

5.4 Results and Discussion

While we utilized the same dataset as Ghannadrad, A. [7], our algorithms and evaluation metrics differ significantly.
In Table 5.3, we compare the results of binary classification between our work and the findings of Ghannadrad, A. [7]. Notably, Ghannadrad evaluated three traditional ML algorithms -RF, SVM, and KNN- with F1 score of 0.99, 0.84, and 0.99, respectively. These results indicate that these algorithms perform exceptionally well in terms of F1 score, particularly RF and KNN.
In contrast, our approach leverages DL models, including single DL models such as LSTM, CNN, GRU, and hybrid DL ones such as CNN-RNN, CNN-LSTM, and CNN-GRU. Among these, LSTM achieved an accuracy of 0.993 and an F1 score of 0.993. This suggests that the LSTM model excels in capturing the temporal dynamics of the data, making it effective for binary classification tasks. The GRU model also performed well, achieving an accuracy of 0.993 and an F1 score of 0.983. These results highlight the strength of GRUs in handling sequential data while maintaining high-performance metrics. However, the results for the

Models	**Ghannadrad, A. [7]**	**Our Work**		
	F1 score	**Accuracy**	**F1 score**	**Model loss**
RF	0.99	/	/	/
SVM	0.84	/	/	/
KNN	0.99	/	/	/
LSTM	/	0.993	0.993	0.044
CNN	/	0.753	0.780	0.493
GRU	/	0.993	0.983	0.072
CNN-RNN	/	0.746	0.775	1.022
CNN-LSTM	/	0.986	0.989	0.027
CNN-GRU	/	0.980	0.980	0.011

Table 5.3: Results of binary classification.

CNN model show moderate performance in binary classification tasks. This model achieved an accuracy of 0.753, an F1 score of 0.780, and a model loss of 0.493. These values indicate that while CNN can learn the features necessary to differentiate between the two classes, its performance was somewhat lower compared to other DL models like LSTM, GRU, or even the hybrid models (CNN-LSTM, CNN-GRU). The relatively higher model loss suggests that the CNN struggled with generalization, possibly due to a lack of sequential memory, which is important for capturing temporal dependencies in the data.
The hybrid models (CNN-RNN, CNN-LSTM, and CNN-GRU) demonstrated varied effectiveness. The CNN-LSTM model exhibited the highest performance, with an accuracy of 0.986, an F1 score of 0.989, and a very low model loss of 0.027, indicating strong learning capabilities for both spatial and temporal features. The CNN-GRU also achieved impressive results, with accuracy and F1 score of 0.980, and a notably low model loss of 0.011, highlighting its efficiency in capturing temporal dependencies while keeping the model simple

and less prone to overfitting. On the other hand, the CNN-RNN model showed significantly lower performance, with an accuracy of 0.746, an F1 score of 0.775, and a high model loss of 1.022. This suggests that combining a CNN with a Simple RNN was insufficient for accurately capturing the temporal relationships needed for binary classification, leading to poorer generalization compared to the CNN-LSTM and CNN-GRU hybrids.

The multi-class classification results in the Table 5.4 reveal significant differences in model performance. Traditional ML models for Ghannadrad, A. [7], such as RF, SVM, and KNN, reported by Ghannadrad, achieved relatively high F1 scores of 0.91, 0.86, and 0.95, respectively. This demonstrates the solid performance of classical methods for multi-class classification. However, these models lack metrics in our evaluation framework.
In DL approaches, GRU excelled, with an accuracy of 0.975, an F1 score of 0.919, and a low model loss of 0.149, indicating its proficiency in capturing temporal dependencies and managing multi-class distinctions effectively. The LSTM model also performed well, with an accuracy of 0.906, an F1 score of 0.912, and a moderate model loss of 0.305, suggesting its suitability for this type of data, though slightly less efficient compared to GRU.
The CNN-LSTM hybrid performed comparably to standalone LSTM, with an accuracy and F1 score of 0.913, and a model loss of 0.242, highlighting its capability to effectively integrate both spatial and temporal data features. The CNN-GRU also performed well, achieving an accuracy of 0.854 and an F1 score of 0.855, with a model loss of 0.388, though slightly less effective than CNN-LSTM.

Models	**Ghannadrad, A. [7]**	**Our Work**		
	F1 score	**Accuracy**	**F1 score**	**Model loss**
RF	0.91	/	/	/
SVM	0.86	/	/	/
KNN	0.95	/	/	/
LSTM	/	0.906	0.912	0.305
CNN	/	0.663	0.618	0.723
GRU	/	0.975	0.919	0.149
CNN-RNN	/	0.566	0.317	1.062
CNN-LSTM	/	0.913	0.913	0.242
CNN-GRU	/	0.854	0.855	0.388

Table 5.4: Results of multi-classification.

Conversely, the standalone CNN and the hybrid CNN-RNN models exhibited poor performance in the multi-class scenario. The CNN model had an accuracy of 0.663 and an F1 score of 0.618, indicating its limitations in multi-class contexts, likely due to its inability to capture temporal patterns. The CNN-RNN had the worst performance, with an accuracy of 0.566, an F1 score of 0.317, and the highest model loss of 1.062, suggesting that Simple RNNs, even when combined with CNNs, struggle significantly with multi-class classification. Overall, the results show that models incorporating GRU or LSTM, either standalone or in hybrid formats tend to perform best in both binary and multi-class classification, while RNN or CNN models fall short in capturing the complexity of data classes.
The confusion matrices shown in Figure 5.2 for the LSTM and GRU models provide insight-

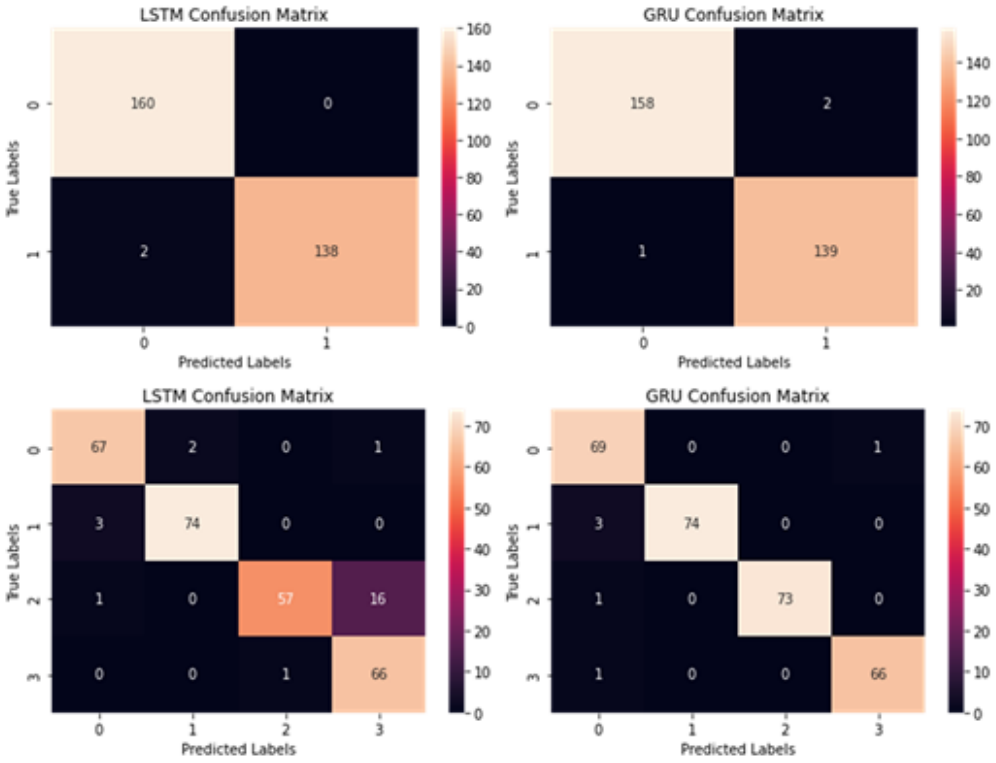


Figure 5.2: Confusion matrices of LSTM and GRU models [4]

ful analysis of their performance across binary and multi-class classification tasks. In the binary classification scenario, both models performed exceptionally well; the LSTM achieved 160 true positives and 138 true negatives, with minimal false positives (0) and false negatives (2), indicating high accuracy. The GRU model also demonstrated strong results with 158 true positives and 139 true negatives, experiencing only slightly more false positives (2) and false negatives (1). In the multi-class classification task, the GRU outperformed the LSTM, achieving high true positive rates and no false positives across all classes, showcasing its effectiveness in differentiating between categories. The LSTM, while performing well with classes 1 and 3, struggled with class 2, exhibiting a higher false positive rate of 16. Overall, the GRU's consistent performance and low error rates suggest it is a more robust choice for this dataset, while the LSTM's results highlight areas for potential improvement, particularly in handling multi-class scenarios.

The loss plots in Figure 5.3 for the LSTM and GRU models provide valuable insights into their training performance across binary and multi-class classification tasks. In the binary classification scenario, the LSTM model demonstrates effective learning with a significant decline in training loss, stabilizing at approximately 0.1, while the test loss, though slightly higher, indicates minimal overfitting. The GRU model follows a similar trend, showing a rapid decrease in both training and test loss; however, it experiences a noticeable spike in test loss around the 60th epoch, suggesting a temporary increase in prediction error. In the multi-class classification task, both models exhibit a consistent decrease in loss, with the LSTM stabilizing around 0.4 for both training and test losses, indicating a good fit. The GRU also shows comparable performance, with its training and test losses remaining closely

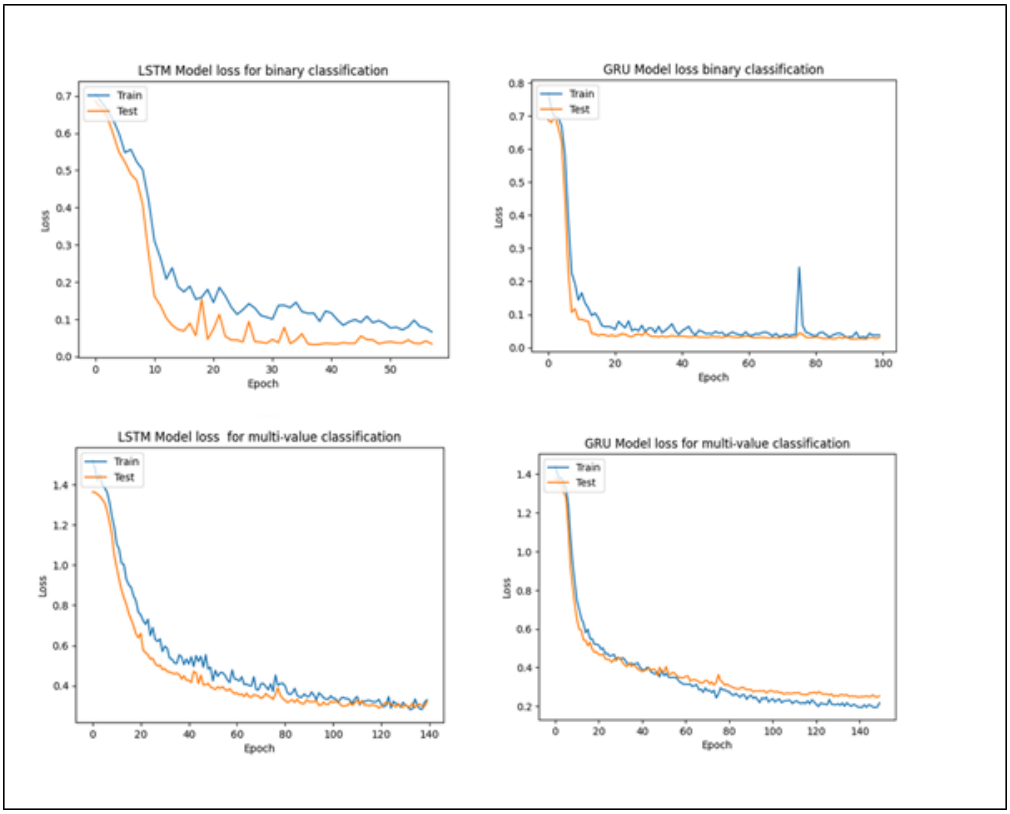


Figure 5.3: Model Loss of LSTM and GRU models [4]

aligned, highlighting effective learning and generalization. Overall, both models demonstrate strong capabilities in handling the classification tasks, with the GRU displaying slightly greater robustness in maintaining stable loss values to 0.2 throughout the training process. Using DL techniques for MQTT attack detection presents significant advantages over traditional ML approaches, particularly in improving detection accuracy and reducing false positives. DL models can automatically learn sophisticated features directly from raw data, eliminating the need for manual feature extraction inherent in traditional ML methods. This capability allows DL techniques to create more accurate detection models, as they can learn from past attacks and identify patterns to preemptively detect similar threats. Moreover, DL models can effectively capture complex patterns and temporal dependencies, enhancing the detection of sophisticated attacks. However, it is essential to acknowledge that DL models typically require substantial amounts of training data and computational resources, which can pose challenges in implementation. Overall, while deep learning offers robust solutions for ensuring the security of MQTT networks, careful consideration must be given to resource requirements.

5.5 Conclusion

In conclusion, the IoT represents a paradigm shift in connectivity, enabling seamless interactions among various services, devices, and systems through numerous applications and protocols. Among these, the MQTT protocol stands out due to its lightweight architecture and open-source nature, making it particularly suitable for environments with constrained

resources, such as those typical in IoT applications characterized by low power consumption, limited computational capabilities, and restricted bandwidth. However, securing the MQTT protocol remains a paramount concern, especially given its susceptibility to vulnerabilities, particularly DoS attacks.

Our study concentrated on detecting and analyzing DoS attacks as a significant threat within MQTT environments. We employed a range of DL algorithms, including LSTM, CNN, GRU, CNN-RNN, CNN-LSTM, and CNN-GRU, to enhance the detection and categorization of DoS attacks within MQTT sensor networks. The empirical results indicated that these models not only improved detection accuracy but also demonstrated a robust capability to adapt to the evolving threat landscape.

Overall, our work underscores the critical importance of advancing security measures in MQTT environments to ensure the integrity and reliability of IoT systems in an increasingly interconnected world.

Chapter 6

Enhancing IoT cyber attacks intrusion detection through GAN-based data augmentation and hybrid deep learning models for MQTT network protocol cyber attacks (Third contribution)

6.1 Problem Statement

A critical challenge in developing an effective IDS for MQTT environments is the issue of unbalanced datasets, where the representation of normal or attack classes is disproportionately skewed. This imbalance can lead to biased model training, resulting in decreased detection rates for minority classes and an overall reduction in the efficacy of IDS. Furthermore, existing methodologies often fail to adequately address the need to generate synthetic data to balance these datasets, thus limiting the performance of detection algorithms. To address these pressing issues, we propose integrating generative adversarial networks to create balanced datasets, coupled with developing hybrid deep learning algorithms aimed at enhancing the detection capabilities of IDS in MQTT-based IoT environments.

6.2 Introduction

IoT systems, particularly those utilizing the MQTT protocol, are highly vulnerable to cyber attacks due to the protocol's lack of strong security features. As IoT traffic continues to increase, the development of effective IDS solutions becomes essential to detect and mitigate threats such as DDoS attacks, MITM attacks, and unauthorized data access, ensuring the security and reliability of IoT environments.
Traditional IDS approaches often face challenges due to unbalanced datasets, where normal traffic is overrepresented compared to various attack types. This imbalance leads to biased model training, reducing detection rates for less common attack classes and overall IDS effectiveness. Furthermore, current methods frequently fail to generate sufficient synthetic data

to balance these datasets, thereby limiting the performance of detection algorithms.
Addressing these critical challenges, we proposed a comprehensive and innovative approach that integrates GAN to generate synthetic data, thereby creating balanced datasets for training. By leveraging the capabilities of GANs, we aimed to enhance the diversity and representation of attack scenarios within the training data, ultimately improving the robustness of the IDS. Furthermore, we introduced three advanced hybrid DL algorithms—CNN-RNN, CNN-LSTM, and CNN-GRU—that synergistically combined the strengths of two DL models to enhance the detection of MQTT attacks. These models were designed to capture both spatial and temporal patterns in the data, enabling more accurate identification of anomalies and threats [5].
Our experimentation demonstrated that a GAN-assisted approach combined with hybrid DL techniques can significantly enhance IDS detection capabilities in MQTT-based IoT environments. By addressing the limitations of existing methodologies, the proposed framework offers a robust solution for data generation and model training. This advancement contributes significantly to improved cybersecurity and reliable threat detection in the rapidly evolving IoT landscape.

6.3 Generative Adversarial Networks

Generative Adversarial Networks or GANs, introduced by Goodfellow et al. in 2014, are designed to address challenges in generative modeling. GANs consist of two main components: a Generator (G) and a Discriminator (D). The generator creates samples intended to mimic real data, while the discriminator's job is to distinguish between real and generated samples. These components work in opposition, engaging in a minimax game to iteratively improve their performance, ultimately enhancing the quality of the generated data by learning the underlying distribution of the training set [103].
Mathematically, the GAN framework can be expressed as follows [103]:

1. **Generator (G):** It introduces random noise to produce data instances that resemble the training data. Represented as $G(z;\theta_g)$, where θ_g are the network parameters, it aims to generate from random noise vector "z" samples "x" that are indistinguishable from real data, thereby minimizing the discriminator's ability to identify them as fake.

2. **Discriminator (D):** This is denoted as $D(z;\theta_d)$, where θ_d represents its parameters. It assesses the likelihood that a sample "x "originates from the real data distribution rather than the generator. The discriminator outputs a scalar value indicating this probability. Its primary function is to examine the generated data and differentiate between genuine and synthetic instances, determining whether a particular sample seems real or fake.

3. **Objective Function:** GAN training involves a minimax game between G and D. The discriminator aims to maximize its accuracy in labeling real and generated data, formalized by (6.2), while the generator seeks to minimize this probability by creating realistic samples, represented by (6.3). The overall GAN objective is given by (6.1).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{6.1}$$

$$\max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] \tag{6.2}$$

$$\min_G V(D,G) = \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{6.3}$$

Given:

- x represents real data samples.
- $p_{\text{data}}(x)$ is the data distribution.
- z represents the input noise vector sampled from a prior distribution $p_z(z)$.
- $D(x)$ represents the discriminator's estimate of the probability that x is real data.
- $G(z)$ represents the generated data from the noise z.

Through the adversarial process, GANs gradually enhance the generated data, producing highly realistic outputs that are hard to distinguish from real data.

4. **Training Procedure:** The training process alternates between updating the discriminator by maximizing $V(D,G)$ concerning θ_d, and updating the generator by minimizing $V(D,G)$ with respect to θ_g. This adversarial training continues until G produces samples that cannot be distinguished from real data. The GAN training process is illustrated in the following algorithmic form:

Algorithm for Adversarial Modeling Framework
Input:

- Data x
- Noise distribution $p_z(z)$
- Number of iterations N
- Number of steps for optimizing k_D
- Learning rates α_D and α_G

Initialize:

- Generator $G(z;\theta_g)$
- Discriminator $D(z;\theta_d)$

Procedure:

for each iteration $t = 1$ to N: **do**
  for each step $j = 1$ to k: **do**
    Sample a batch of real data $\{x^{(i)}\}$ from $p_{\text{data}}(x)$
    Sample a batch of noise $\{z^{(i)}\}$ from $p_z(z)$
    Generate a batch of fake data $\{G(z^{(i)};\theta_g)\}$
    Compute the discriminator loss:

$$L_D = -\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x^{(i)};\theta_d) + \log(1 - D(G(z^{(i)};\theta_g);\theta_d))\right]$$

Update the discriminator parameters using gradient descent:

$$\theta_d \leftarrow \theta_d - \alpha_D \nabla_{\theta_d} L_D$$

end for
Sample a batch of noise $\{z^{(i)}\}$ from $p_z(z)$
Compute the generator loss:

$$L_G = -\frac{1}{m} \sum_{i=1}^{m} \log D(G(z^{(i)}; \theta_g); \theta_d)$$

Update the generator parameters using gradient descent:

$$\theta_g \leftarrow \theta_g - \alpha_G \nabla_{\theta_g} L_G$$

end for
End Procedure

Alternative Objective for Generator:

- If $\log(1 - D(G(z); \theta_d))$ saturates:

$$L'_G = -\frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}; \theta_g); \theta_d))$$

- Update generator parameters using:

$$\theta_g \leftarrow \theta_g - \alpha_G \nabla_{\theta_g} L'_G$$

Note: This algorithm uses alternating updates for θ_d and θ_g, with k steps of optimizing D followed by one step of optimizing G.

End Algorithm

Figure 6.1 illustrates the GAN's architecture. GANs are designed to generate high-quality samples that closely mimic real data distributions by optimizing a minimax objective function. The competitive interaction between the generator and discriminator networks fosters continuous enhancements, resulting in realistic and varied outputs. This effectiveness is especially prominent in image generation, where GANs achieve impressive outcomes. Building upon this method, we utilize GANs for generating sequential tabular data to help balance our dataset. Implementing the GAN framework adds variability to sequence data, effectively mitigating data imbalances

6.4 Methodology and Experimental Setup

This study presents an IDS specifically tailored for detecting MQTT attacks, as illustrated in Fig. 6.2. The proposed methodology involves several key phases. It starts with a comprehensive preprocessing step, where raw data is cleaned by removing rows with missing values.

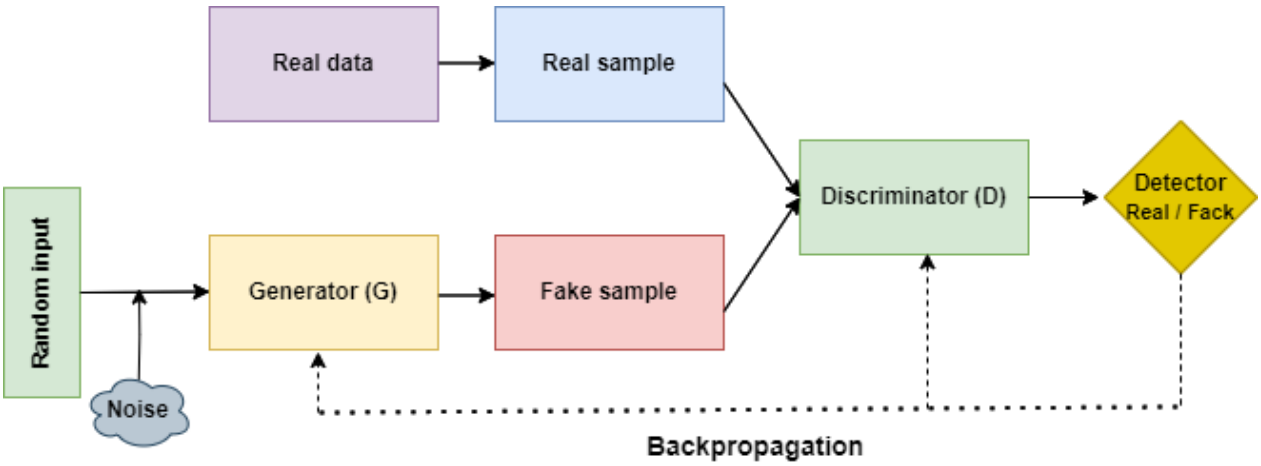


Figure 6.1: Overview of the generative adversarial network architecture [5]

This is followed by encoding and scaling processes, and then by applying Principal Component Analysis (PCA) to improve data precision. A significant part of our approach is the use of a GAN technique to generate additional samples, ensuring a balanced dataset. The refined dataset is then divided into training and testing sets. Next, three distinct hybrid DL models (CNN-RNN, CNN-LSTM, and CNN-GRU) are employed for multiclass classification. These models, trained on both original and augmented MQTT datasets, are specifically developed to detect and classify various MQTT attack types using the designated test subsets [5].

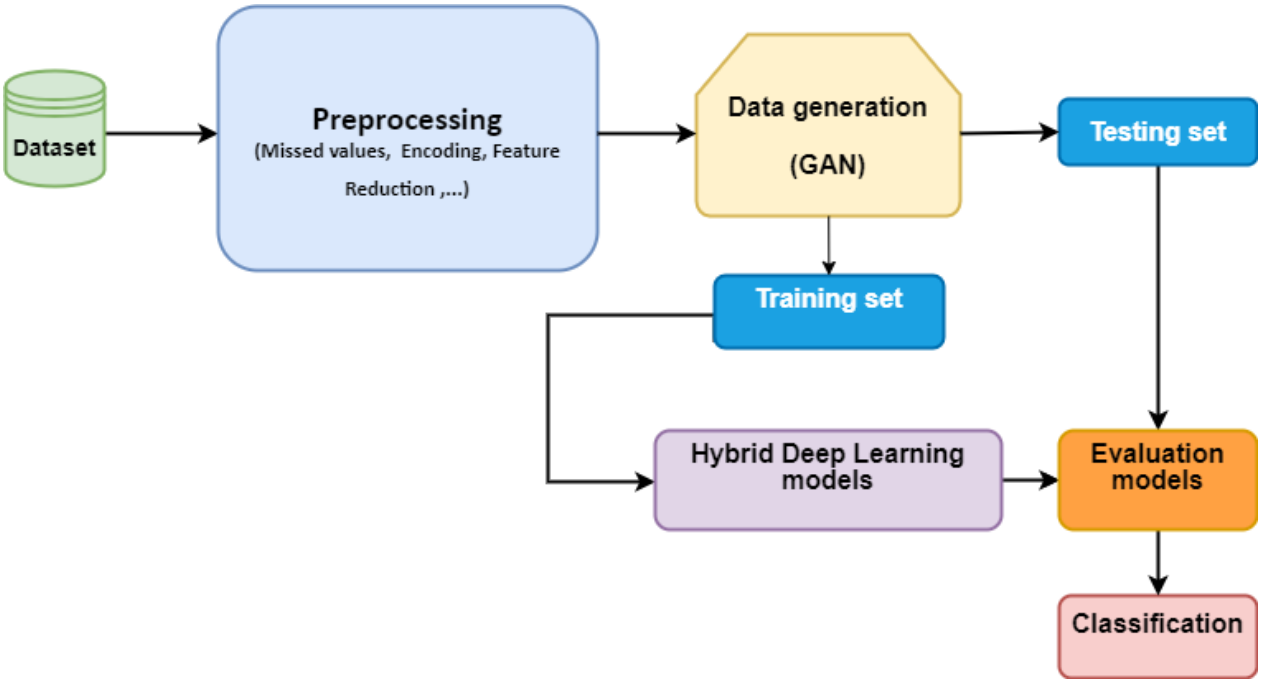


Figure 6.2: Cyber attack detection training model schema [5]

6.4.1 Dataset Description

A dataset is fundamental for developing a classifier. Our research chose the MQTTset dataset developed by Vaccari et al. [6] due to its precise representation of real-world IoT networks, making it highly suitable for analyzing MQTT data. The dataset used in our study has been previously detailed in Section 4.3.1.
The creators of this dataset employed several ML algorithms to evaluate its performance. Tables 6.1 and 6.2 show the results for both unbalanced and balanced datasets. The balanced dataset was constructed by duplicating each threat instance.
Due to the differences in data distribution, the balanced dataset demonstrated lower performance metrics compared to the unbalanced one. In the context of IDS and ML, having a balanced dataset is crucial to ensure that the model receives equal exposure to all classes (including both legitimate traffic and various types of attacks). A balanced dataset allows the algorithm to learn to effectively differentiate between classes, leading to a more accurate and robust model overall.
In contrast, an unbalanced dataset, where one class (e.g., legitimate traffic) significantly outweighs others (such as malicious traffic or specific attack types), may cause the algorithm to become biased towards the majority class. This bias can lead to higher accuracy and F1 score for the dominant class but may result in decreased performance for minority classes.
We attribute the superior results from the unbalanced dataset to the fact that the balanced version was created by repeating existing threats instead of generating new, distinct ones. This repetition made it easier for the classifier to quickly recognize and predict outcomes.
To address this, our study employed a GAN technique to generate new, nearly realistic data without duplication, making it harder to distinguish it as synthetic and thereby enhancing the performance of the classifier.

Table 6.1: Obtained results from the MQTT dataset with unbalanced dataset [6]

Algorithm	Accuracy	F1 score
Neural network	0.993	0.993
Random forest	0.994	0.994
Naïve bayes	0.988	0.990
Decision tree	0.978	0.985
Gradient boost	0.991	0.992
Multilayer perceptron	0.947	0.964

Table 6.2: Obtained results from the MQTT dataset with balanced dataset [6]

Algorithm	Accuracy	F1 score
Neural network	0.904	0.902
Random forest	0.916	0.914
Naïve bayes	0.644	0.687
Decision tree	0.916	0.914
Gradient boost	0.880	0.873
Multilayer perceptron	0.904	0.902

6.4.2 Preprocessing Data and Feature Engineering

The data preprocessing phase aims to extract relevant features that characterize normal behaviour and detect malicious activities. This phase is crucial because the selected features significantly impact the performance of the applied algorithm. To achieve a consistent dataset, we applied four different preprocessing steps to each class file [5]:

1. **Missed values and irrelevant features:** Some data may contain null values or irrelevant features, which could compromise the experiment's outcomes. Thus, it is essential to eliminate these to ensure successful results.

2. **Encoding:** Label encoding is a technique used to convert categorical values within a dataset. For MQTT attack categories, it assigns a unique integer value ranging from 0 to n-1 for each category, where n indicates the total number of distinct categories for that feature.

3. **Feature reduction:** PCA is a technique used to reduce dimensionality while preserving as much variance in the data as possible. It aids in eliminating redundant features and streamlining the model. In our research, we applied PCA to the original MQTTset data to determine a set of input features that accurately represent the data distribution. Given the extensive number of features linked to each data sample in the MQTTset, we utilized PCA to identify the key features for the training process and remove irrelevant ones, ultimately reducing the feature set to a final total of 10.

4. **Feature Scaling:** This technique is used to normalize the diverse range of features. Normalization ensures features have a uniform scale, addressing issues caused by differing magnitudes, units, and ranges. For ML algorithms relying on Euclidean distance, standard scaling (mean = 0, variance = 1) is essential to enhance performance and ensure effective operation.

6.4.3 Data Generation Through GAN Framework

Our dataset exhibits unbalanced classes. For instance, there are 165,463 samples in the 'legitimate' class, while the 'Flood' class contains only 613 samples. To address this issue, one effective approach is to create new data using a GAN technique. The proposed GAN framework consists of two components: the G and the D, we created a G to produce malicious messages and a D to distinguish between real messages and malicious ones [5].

The Generator class: It comprises three main layers: a dense layer, an activation layer that employs ReLU, and a dropout layer as shown in Figure 6.3. To prevent overfitting during training, the dropout layer randomly disables neurons with a probability of 0.1. This setup ensures that that the model performs inll in both training and testing datasets. The input data are fed into the generator's initial layer, which consists of 128 neurons, and each subsequent dense layer doubles this number. The final output layer of the generator corresponds to the feature count of the original dataset.

The discriminator class: It consists of four block layers, each with a dense layer, a ReLU activation, and a dropout layer, except for the first block (dense + ReLU) and the last block (dense + sigmoid). The dense layers start with 512 neurons, halving in each subsequent layer. This architecture forms the generator and discriminator in the GAN class. The GAN is trained separately for each class file using an Adam optimizer, batch size of 64, learning

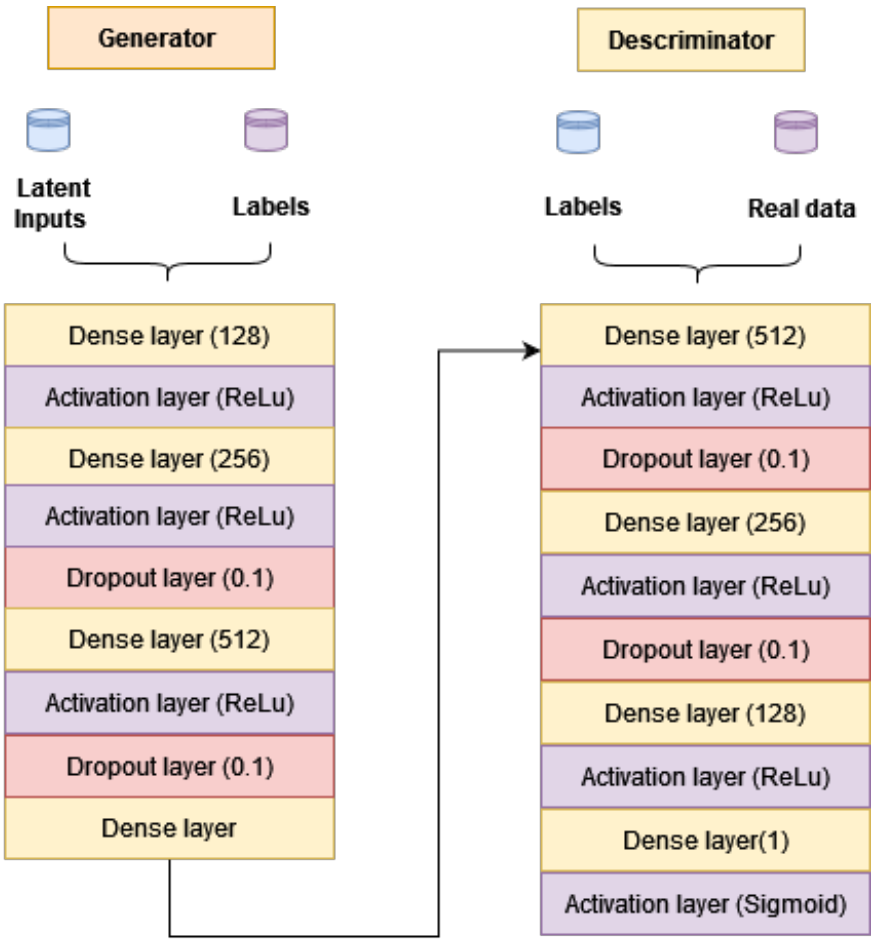


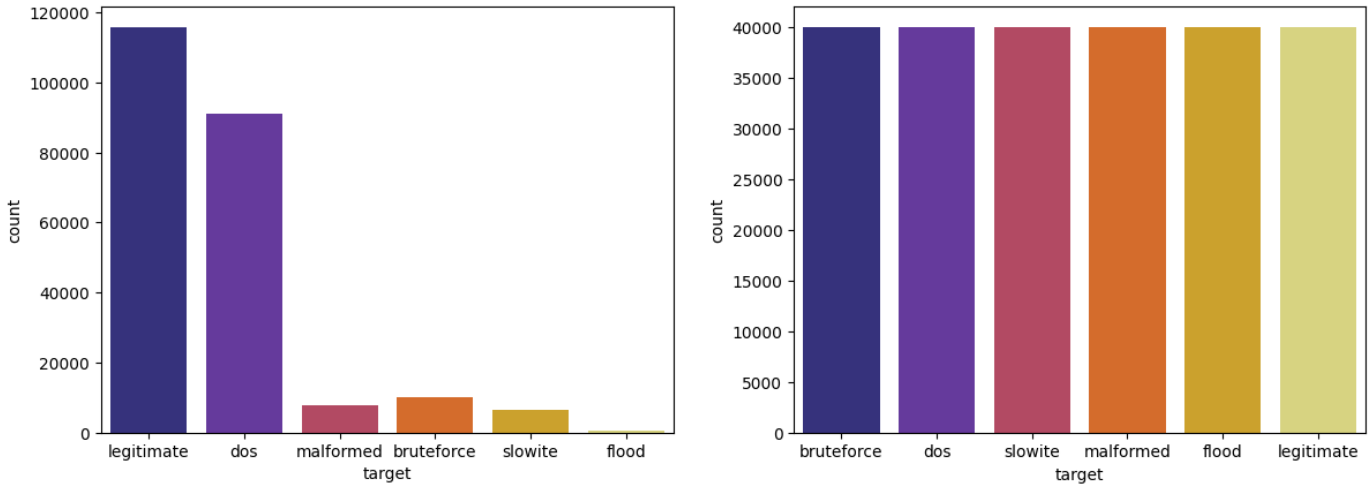
Figure 6.3: Utilizing GAN for generating new data [5]

rate of 0.00001, noise shape of 32, and 350 epochs.
Figure 6.3 illustrates how our approach incorporates two models of artificial neural networks. After the data generation phase, we ensure that the dataset is balanced, as shown in Figure 6.4.

6.4.4 Proposed approach

In our study, we introduced three novel hybrid DL architectures that take advantage of the convolutional and recurrent neural networks mentioned earlier. These architectures are referred to as CNN-RNN, CNN-LSTM, and CNN-GRU. We evaluated hybrid DL models both before and after the GAN process to illustrate the effectiveness of the data generation technique in classifying cyber attacks as follows:

1. **CNN-RNN model:** This DL architecture combines CNNs to extract local features and RNNs to capture long-term dependencies [136]. A 1D convolutional layer with ReLU activation and batch normalization processes the input, followed by a simple RNN layer and two dense layers. The final dense layer uses softmax activation to classify the data into six predefined classes. Batch normalization between the CNN and RNN layers enhances training speed and stability, ensuring effective categorization of



(a) Before (b) After

Figure 6.4: Data samples of each class before and after preprocessing [5]

features as genuine or spurious. The architecture of the CNN-RNN model is represented in Figure 6.5

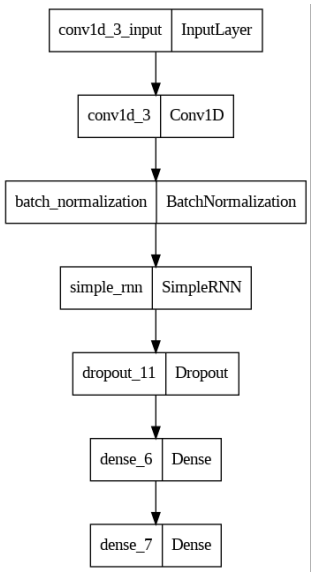


Figure 6.5: CNN-RNN Model [5]

2. **CNN-LSTM model:** This model, as shown in Figure 6.6, is a hybrid DL architecture that combines CNNs with LSTM networks [137]. The model starts with a 1D convolutional layer using a ReLU activation function to extract local features from the input sequences. This is followed by an average pooling layer (pool size of 2) and a dropout layer to reduce overfitting. Two LSTM layers are added to handle longer sequences and mitigate the vanishing gradient problem. After the LSTM layers, dropout and flattening layers are included. The model ends with two dense blocks: the first containing a dense layer (ReLU activation) and dropout, and the second with a dense layer (softmax activation) for classification into six predefined classes. Combining convolu-

tional and pooling layers enhances feature extraction from long sequences, improving the transition to LSTM layers and overall computational efficiency.

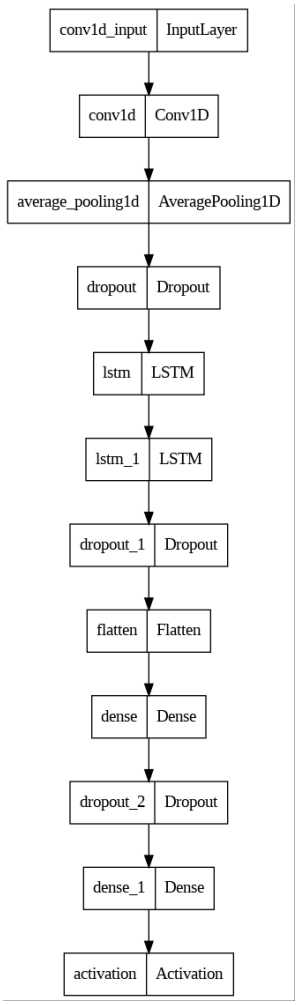


Figure 6.6: CNN-LSTM Model [5]

3. **CNN-GRU model:** This model, shown in Fig. 6.7, is a hybrid architecture combining CNNs with GRUs. It starts with two 1D convolutional layers followed by ReLU activation, max-pooling, and dropout layers to reduce overfitting. Two GRU layers, each with a dropout layer, process the sequence data. The output is flattened and passed through two dense blocks: the first with a ReLU activation and dropout, and the second with a softmax activation for classification into six predefined classes. This architecture efficiently extracts features from long sequences, optimizing computational efficiency and reducing processing time.

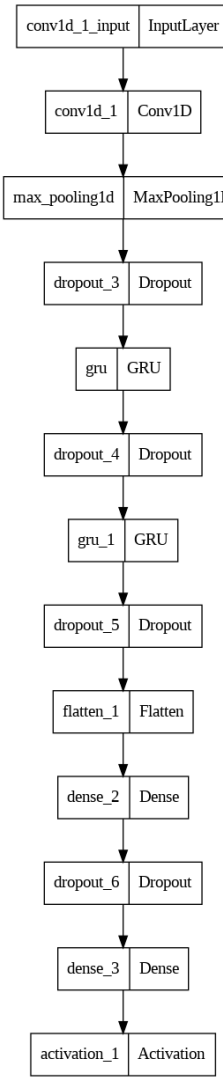


Figure 6.7: CNN-GRU model [5]

6.5 Results and Discussion

The three proposed models, CNN-RNN, CNN-LSTM, and CNN-GRU, are developed for multiclass classification using Python and the Keras library. The models used the Adam optimizer, with a batch size of roughly 2050, and categorical cross-entropy was chosen as the loss function. Each model was trained over ten epochs. To evaluate the proposed IDSs, we used two datasets: the original MQTTset dataset (dataset1) and a composite dataset that combines original and GAN-generated data (dataset2). The data was split, with 70% used for training and 30% for testing. To thoroughly evaluate the performance of our hybrid DL models, we carefully chose five key metrics: Accuracy, F1 score, model Loss, precision, and recall. As shown in Table 6.3, the data generation approach applied to MQTT significantly enhances the detection rate [5].

In the first experiment using dataset1, the classification accuracy and F1 score for all proposed models were 83% and 61%, respectively. The model loss values were 0.415 for CNN-RNN, 0.409 for CNN-LSTM, and 0.410 for CNN-GRU. Precision showed slight varia-

Table 6.3: Obtained results from the MQTT dataset.

	Dataset 1			Dataset 2		
Metrics /DL models	CNN-RNN	CNN-LSTM	CNN-GRU	CNN-RNN	CNN-LSTM	CNN-GRU
Accuracy	0.83	0.83	0.83	0.99	0.99	0.99
F1 score	0.61	0.61	0.61	0.99	0.99	0.99
Model loss	0.415	0.409	0.410	0.005	0.001	0.002
Recall	0.55	0.55	0.55	1.00	1.00	1.00
Precision	0.86	0.87	0.81	1.00	1.00	1.00

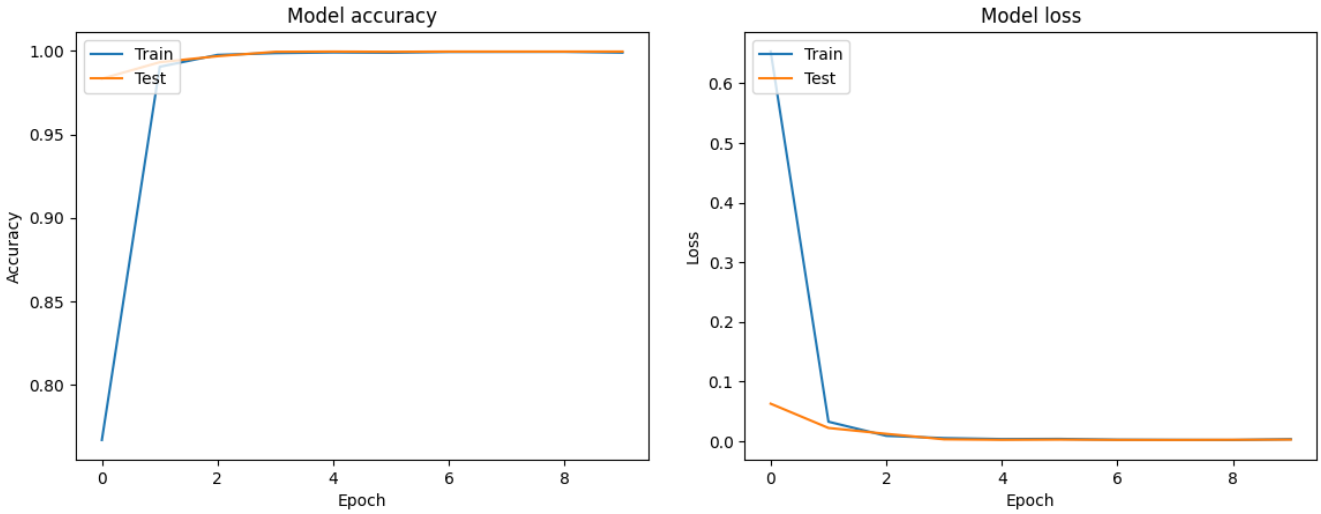


Figure 6.8: CNN-LSTM accuracy and loss after GAN.

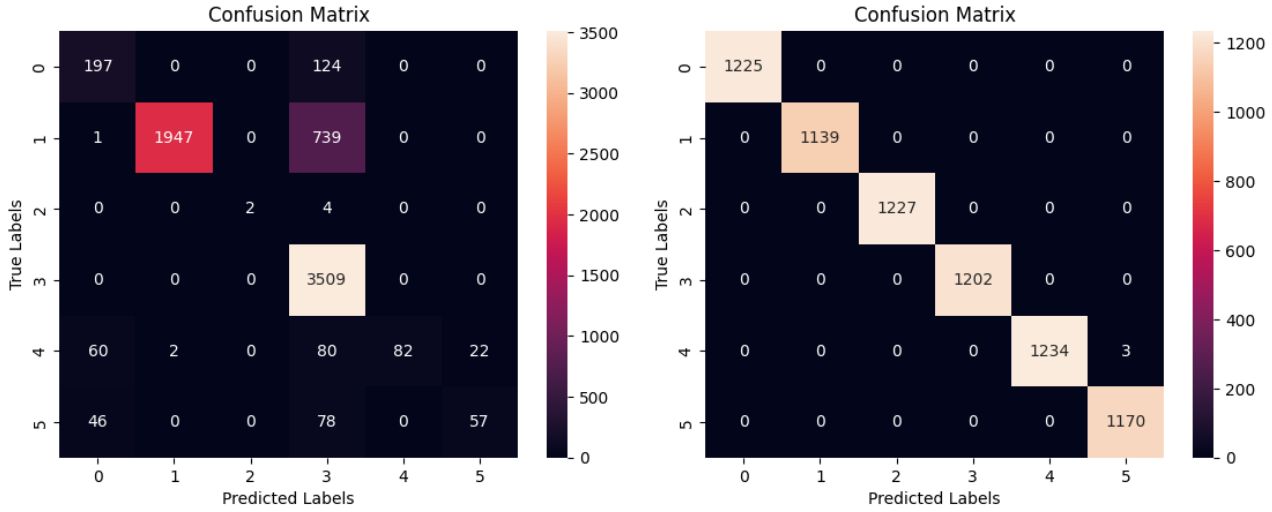
tion, with CNN-RNN at 0.86, CNN-LSTM at 0.87, and CNN-GRU at 0.81, indicating the proportion of correct positive identifications, where CNN-LSTM had a slight advantage. All models had a recall of 0.55, meaning they successfully identified 55% of true positive cases among all actual positives. These results could be due to the dataset being unbalanced and relatively small, which makes it challenging to distinguish between different classes effectively.

In the second experiment using the balanced dataset2, we evaluated the impact of data generation and augmentation on the performance of the proposed models. Unlike the results from dataset1, all models achieved a nearly perfect accuracy of 99% with dataset2, demonstrating a significant improvement and indicating that the models could correctly classify almost all instances. Similarly, the F1 score for all models was 99%, reflecting an excellent balance between precision and recall, as well as high overall performance. The loss values for dataset2 were notably low: 0.005 for CNN-RNN, 0.001 for CNN-LSTM, and 0.002 for CNN-GRU, indicating minimal prediction errors, with CNN-LSTM having the lowest loss. All models attained a recall of 100%, meaning they correctly identified all true positives, while precision was also 100%, showing that every positive prediction was accurate. Figure 6.8 presents the accuracy and model loss after applying GAN for the CNN-LSTM model.

In our study, we employed two distinct confusion matrices to evaluate the performance of our CNN-LSTM model in detecting various MQTT attack types, as shown in Fig. 6.9. These

attack types include 'Bruteforce', 'Dos', 'Flood', 'Legitimate', 'Malformed', and 'Slowrite' coded by 0,1,2, 3, 4, 5 respectively. Initially, before applying GAN for data augmentation, the confusion matrix depicted a scattered distribution of classifications. For example, in the case of the 'Bruteforce' attack type, while 197 instances were correctly identified, 124 were mistakenly classified as 'Legitimate'. Similarly, for 'Dos', 1,947 instances were correctly labeled, but 739 were misclassified as 'Flood'. This pattern persisted across different attack types, with 'Malformed' frequently being misclassified among various categories, leading to only 82 correct classifications out of many attempts.

However, a stark improvement in the confusion matrix was evident post-GAN implementa-



(a) Before (b) After

Figure 6.9: CNN-LSTM confusion matrix before and after GAN
0='Bruteforce',1='Dos', 2='Flood',3='Legitimate',4='Malformed', 5='Slowite'.

tion. The classifications became predominantly diagonal, indicating substantially improved accuracy in predictions. For 'Bruteforce', 1,225 instances were correctly labeled, with no misclassifications into other categories. This trend was consistent across all attack types; for example, 'Flood' had 1,227 correct classifications, and 'Legitimate' had 1,202. The only minor exception was in the 'Malformed' category, where three instances were misclassified as 'Slowrite'.

Overall, incorporating the GAN has effectively enhanced the model's ability to distinguish between different classes of traffic, drastically reducing misclassification and improving overall accuracy. This improvement reflects the GAN's contribution to more robust feature extraction or improved training data, resulting in more reliable and precise predictions.

6.5.1 Comparative results analysis against published research works

The field of IDS has seen the development of various methodologies. To illustrate this diversity, Table 6.4 provides a comprehensive comparison of accuracy and error rate metrics for several IDS models documented in other literature. This comparison highlights the performance of our CNN-LSTM model alongside other established algorithms, offering insights into the relative strengths and weaknesses of each approach. The results show that our CNN-LSTM model outperforms all others, achieving an accuracy of 99.00% and a low error rate

of 0.10%. This exceptional performance underscores the model's effectiveness in intrusion detection and contributes to the ongoing efforts to optimize IDS technologies and enhance cybersecurity.

Table 6.4: Comparative results analysis against published research works.

Reference	Accuracy	Error rate
GAN [138]	93.34	5.12
Neural Network [6]	90.44	-
RF [6]	91.59	-
Naive Bayes [6]	64.39	-
Decision Tree [6]	91.60	-
Gradient Boost [6]	87.96	-
Multilayer Perceptron [6]	90.39	-
CNN-LSTM [139]	88.55	0.16
CNN-LSTM (Our Model)	99.00	0.10

In comparison, the GAN model by Prabakaran et al. [138] achieved an accuracy of 93.34% and an error rate of 5.12%, which, though strong, highlights a notable performance gap with our CNN-LSTM model. Similarly, models from Vaccari et al. [6], such as the RF and DT algorithms, demonstrated competitive accuracy rates of 91.59% and 91.60%, respectively, but still fell short of our model's performance. Traditional methods like NB [6] showed significantly lower accuracy at 64.39%, indicating its limitations in intrusion detection.
Additionally, the gradient boosting and MLP models [6] achieved accuracy scores of 87.96% and 90.39%, respectively. These results reinforce the trend that while various algorithms perform reasonably, the CNN-LSTM model stands out in both accuracy and error rate.
The CNN-LSTM model reported by Alzahrani et al. [139] achieved a commendable accuracy of 88.55% and a low error rate of 0.16%, but these results are still significantly lower than our model's 99.00% accuracy and 0.10% error rate, suggesting room for improvement in Alzahrani et al.'s approach. Enhancements in model architecture, training methodologies, and feature engineering could help bridge this gap, particularly considering the potential influence of data imbalance on their results. These findings emphasize the need for continued advancements in ML techniques to address cybersecurity challenges, highlighting the importance of robust data handling for optimizing model performance.
The considerable reduction in error rate in our model reflects its robustness and practical applicability in real-time intrusion detection scenarios, positioning our CNN-LSTM model as a strong candidate for advancing intrusion detection technologies in cybersecurity.

6.6 Conclusion

Our study has investigated the essential role of IDS in protecting IoT networks from a growing range of cyber threats. Our research emphasized the significant challenges posed by unbalanced datasets, which limited the effectiveness of conventional detection methods. To address these issues, we have proposed an innovative approach utilizing GANs to generate high-quality synthetic data that mitigates data imbalance. The integration of hybrid DL techniques, such as CNN-RNN, CNN-LSTM, and CNN-GRU, has yielded promising results in improving the accuracy and reliability of attack detection.

Our findings suggest that the application of GANs not only enhanced the representation of various attack types but also boosted the overall performance of IDS in multi-class classification scenarios.
Ultimately, this research contributes to the ongoing pursuit of developing robust and adaptive security solutions for IoT ecosystems. By tackling the challenges associated with data scarcity and imbalance, we aspire to advance more effective intrusion detection mechanisms capable of evolving in tandem with emerging cyber threats, thereby safeguarding the integrity and security of interconnected devices.

General Conclusion

In conclusion, the rapid expansion of the IoT has introduced transformative benefits across a variety of sectors, enabling more efficient and interconnected systems. However, this growth has been accompanied by significant security challenges, particularly in the communication protocols that underpin IoT networks. Among these, the MQTT protocol stands out due to its lightweight, efficient design, which is ideal for resource-constrained IoT devices. Despite its advantages, MQTT lacks inherent security features, leaving IoT systems vulnerable to a wide range of cyber threats.

This dissertation has presented a comprehensive investigation into the security challenges associated with MQTT in IoT networks and proposed advanced solutions to address these challenges. By leveraging ML, DL, and GAN techniques. This research has demonstrated novel and resource-efficient approaches for intrusion detection, threat mitigation, and overall system robustness.

The key contributions of this dissertation include the development of a balanced dataset derived from MQTT-specific traffic, the implementation of ensemble learning models to improve IDS efficiency, and the innovative use of GAN-based data augmentation to overcome class unbalance issues and get data balanced and unbaised. Moreover, the introduction of hybrid DL models combining for example CNNs with RNNs or CNNs with LSTM has enabled robust detection of complex attack patterns in MQTT environments.

Through extensive experimentation and validation, the proposed approaches have been shown to significantly enhance the performance and accuracy of IDS in detecting a variety of cyber attacks, including DoS, bruteforce, flood, and other attacks. Our research has bridged a crucial gap between theoretical advancements and practical applications.

This work provides valuable insights for researchers aiming to fortify IoT systems against evolving threats. The proposed methodologies not only improve the reliability, security, and privacy of MQTT communication but also establish a foundation for future research in developing intelligent, adaptive, and resource-efficient IoT security frameworks. As IoT continues to expand, the findings of this thesis will contribute to enabling a safer and more resilient connected world.

References

[1] Prince Appiah, Thierry Oscar Edoh, and Jules Degila. Predicting elderly patient behaviour in rural healthcare using machine learning. In *IREHI*, pages 92–97, 2019.

[2] Harsh H Patel and Purvi Prajapati. Study and analysis of decision tree based classification algorithms. *International Journal of Computer Sciences and Engineering*, 6(10):74–78, 2018.

[3] Hayette Zeghida, Mehdi Boulaiche, and Ramdane Chikh. Securing mqtt protocol for iot environment using ids based on ensemble learning. *International Journal of Information Security*, 22(4):1075–1086, 2023.

[4] Hayette Zeghida, Mehdi Boulaiche, and Ramdane Chikh. Detection of dos attacks in mqtt environment. In *International Conference on Intelligent Systems and Pattern Recognition*, pages 129–140. Springer, 2023.

[5] Hayette Zeghida, Mehdi Boulaiche, Ramdane Chikh, Alwi M Bamhdi, Ana Luiza Bessa Barros, Djamel Zeghida, and Ahmed Patel. Enhancing iot cyber attacks intrusion detection through gan-based data augmentation and hybrid deep learning models for mqtt network protocol cyber attacks. *Cluster Computing*, 28(1):58, 2025.

[6] Ivan Vaccari, Giovanni Chiola, Maurizio Aiello, Maurizio Mongelli, and Enrico Cambiaso. Mqttset, a new dataset for machine learning techniques on mqtt. *Sensors*, 20(22):6578, 2020.

[7] Ali Ghannadrad. Machine learning-based dos attacks detection for mqtt sensor networks. 2020.

[8] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[9] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[10] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 141:199–221, 2018.

[11] Tara Salman and Raj Jain. A survey of protocols and standards for internet of things. *arXiv preprint arXiv:1903.11549*, 2019.

[12] Ding-Chau Wang, Ray Chen, and Hamid Al-Hamadi. Reliability of autonomous internet of things systems with intrusion detection attack-defense game design. *IEEE Transactions on Reliability*, 70(1):188–199, 2020.

[13] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE, 2008.

[14] Alma Oracevic, Selma Dilek, and Suat Ozdemir. Security in internet of things: A survey. In *2017 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2017.

[15] Huansheng Ning. *Unit and ubiquitous internet of things*. CRC press, 2013.

[16] Euihyun Jung, IlKwon Cho, and Sun Moo Kang. An agent modeling for overcoming the heterogeneity in the iot with design patterns. In *Mobile, Ubiquitous, and Intelligent Computing: MUSIC 2013*, pages 69–74. Springer, 2014.

[17] Weng Chun Tan and Manjit Singh Sidhu. Review of rfid and iot integration in supply chain management. *Operations Research Perspectives*, 9:100229, 2022.

[18] European Research Cluster on the Internet of Things. About iot. https://www.internet-of-things-research.eu/about_iot.htm. Accessed: 2024.

[19] Mayuri A Bhabad and Sudhir T Bagade. Internet of things: architecture, security issues and countermeasures. *International Journal of Computer Applications*, 125(14), 2015.

[20] Arthur AZ Soares, Yona Lopes, Diego Passos, Natalia C Fernandes, and Débora C Muchaluat-Saade. 3as: Authentication, authorization, and accountability for sdn-based smart grids. *IEEE Access*, 9:88621–88640, 2021.

[21] Mehdi Boulaiche. Survey of secure routing protocols for wireless ad hoc networks. *Wireless Personal Communications*, 114(1):483–517, 2020.

[22] Changxiang Shen, HuangGuo Zhang, Dengguo Feng, ZhenFu Cao, and JiWu Huang. Survey of information security. *Science in China Series F: Information Sciences*, 50(3):273–298, 2007.

[23] Malik Zaib Alam, Faheem Reegu, Arshad Ahmad Dar, and Wasim Ahmad Bhat. Recent privacy and security issues in internet of things network layer: A systematic review. In *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, pages 1025–1031. IEEE, 2022.

[24] Mario Weber and Marija Boban. Security challenges of the internet of things. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 638–643. IEEE, 2016.

[25] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer networks*, 57(10):2266–2279, 2013.

[26] Ismail Butun, Patrik Österberg, and Houbing Song. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys & Tutorials*, 22(1):616–644, 2019.

[27] Emrah Atilgan, Ilker Ozcelik, and Esra N Yolacan. Mqtt security at a glance. In *2021 International Conference on Information Security and Cryptology (ISCTURKEY)*, pages 138–142. IEEE, 2021.

[28] Petros Spachos, Ioannis Papapanagiotou, and Konstantinos N Plataniotis. Microlocation for smart buildings in the era of the internet of things: A survey of technologies, techniques, and approaches. *IEEE Signal Processing Magazine*, 35(5):140–152, 2018.

[29] Syeda Manjia Tahsien, Hadis Karimipour, and Petros Spachos. Machine learning based solutions for security of internet of things (iot): A survey. *Journal of Network and Computer Applications*, 161:102630, 2020.

[30] Ahmad W Atamli and Andrew Martin. Threat-based security analysis for the internet of things. In *2014 International Workshop on Secure Internet of Things*, pages 35–43. IEEE, 2014.

[31] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, 18(3):2027–2051, 2016.

[32] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2-3):293–315, 2003.

[33] Hamid Bostani and Mansour Sheikhan. Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, 98:52–71, 2017.

[34] Linus Wallgren, Shahid Raza, and Thiemo Voigt. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, 9(8):794326, 2013.

[35] Michael Mc Grath. Threat modelling for legacy enterprise applications. 2013.

[36] Nelcileno Araújo, Ruy De Oliveira, Ailton Akira Shinoda, Bharat Bhargava, et al. Identifying important characteristics in the kdd99 intrusion detection dataset by feature selection using a hybrid approach. In *2010 17th International Conference on Telecommunications*, pages 552–558. IEEE, 2010.

[37] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. Cyber scanning: a comprehensive survey. *Ieee communications surveys & tutorials*, 16(3):1496–1519, 2013.

[38] Shahid Anwar, Zakira Inayat, Mohamad Fadli Zolkipli, Jasni Mohamad Zain, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Victor Chang. Cross-vm cache-based side channel attacks and proposed prevention mechanisms: A survey. *Journal of Network and Computer Applications*, 93:259–279, 2017.

[39] Krushang Sonar and Hardik Upadhyay. A survey: Ddos attack on internet of things. *International Journal of Engineering Research and Development*, 10(11):58–63, 2014.

[40] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2015.

[41] Vinko Zlomislić, Krešimir Fertalj, and Vlado Sruk. Denial of service attacks, defences and research challenges. *Cluster Computing*, 20:661–671, 2017.

[42] B Prabadevi and N Jeyanthi. Distributed denial of service attacks and its effects on cloud environment-a survey. In *The 2014 International Symposium on Networks, Computers and Communications*, pages 1–5. IEEE, 2014.

[43] Christian Rossow. Amplification hell: Revisiting network protocols for ddos abuse. In *NDSS*, pages 1–15, 2014.

[44] Muhammad Rizwan Asghar, György Dán, Daniele Miorandi, and Imrich Chlamtac. Smart meter data privacy: A survey. *IEEE Communications Surveys & Tutorials*, 19(4):2820–2835, 2017.

[45] L Sujihelen, C Jayakumar, and C Senthil Singh. Detecting node replication attacks in wireless sensor networks: Survey. *Indian Journal of Science and Technology*, 8(16), 2015.

[46] Wafa Ben Jaballah, Mauro Conti, Gilberto Filè, Mohamed Mosbah, and Akka Zemmari. Whac-a-mole: Smart node positioning in clone attack in wireless sensor networks. *Computer Communications*, 119:66–82, 2018.

[47] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017.

[48] Bill Glover and Himanshu Bhatt. *RFID essentials*. " O'Reilly Media, Inc.", 2006.

[49] Priya Saini. Design and requirements of iot based smart buildings. 2021.

[50] Edgar Miguel Felício Oliveira da Silva. A multi-criteria framework to assist on the design of internet-of-things systems. 2020.

[51] Victor Seoane, Carlos Garcia-Rubio, Florina Almenares, and Celeste Campo. Performance evaluation of coap and mqtt with security support for iot environments. *Computer Networks*, 197:108338, 2021.

[52] Syaiful Andy, Budi Rahardjo, and Bagus Hanindhito. Attack scenarios and security analysis of mqtt communication protocol in iot system. In *2017 4th International conference on electrical engineering, computer science and informatics (EECSI)*, pages 1–6. IEEE, 2017.

[53] Anup Burange, Harshal Misalkar, and Umesh Nikam. Security in mqtt and coap protocols of iot's application layer. In *Communication, Networks and Computing: First International Conference, CNC 2018, Gwalior, India, March 22-24, 2018, Revised Selected Papers 1*, pages 273–285. Springer, 2019.

[54] Meenaxi M Raikar and SM Meena. Vulnerability assessment of mqtt protocol in internet of things (iot). In *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pages 535–540. IEEE, 2021.

[55] Wael Dheeb, Mahdi Dridi, Swarup Ghosh, Abdelouahid Guerar, and Slim Sahnoun. Detection of IoT botnet attacks: N-baiot. https://archive.ics.uci.edu/dataset/442/detection+of+iot+botnet+attacks+n+baiot, 2018.

[56] Nour Moustafa. Ton_iot datasets. https://dx.doi.org/10.21227/fesz-dm97, 2019.

[57] IEEE DataPort. Bot-iot dataset. https://ieee-dataport.org/documents/bot-iot-dataset, 2019.

[58] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.

[59] IEEE DataPort. Iot network intrusion dataset (iotnid). https://ieee-dataport.org/open-access/iot-network-intrusion-dataset, 2019.

[60] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. Iot-23: A labeled dataset with malicious and benign iot network traffic (version 1.0.0). Zenodo, 2020.

[61] Medbiot data. https://cs.taltech.ee/research/data/medbiot/, 2020.

[62] IEEE DataPort. MQTT IoT IDS2020: Mqtt internet of things intrusion detection dataset. https://ieeedataport.org/open-access/mqtt-iot-ids2020-mqtt-internet-things-intrusion-detection-dataset, 2020.

[63] Kaggle user: cnrieiit. MQTTSet dataset. https://www.kaggle.com/cnrieiit/mqttset, 2020.

[64] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning. https://dx.doi.org/10.21227/mbc1-1h68, 2022.

[65] University of New Brunswick. IoT 2023 Dataset. https://www.unb.ca/cic/datasets/iotdataset-2023.html, 2023.

[66] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[67] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[68] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine learning from theory to algorithms: an overview. In *Journal of physics: conference series*, volume 1142, page 012012. IOP Publishing, 2018.

[69] Dhruti Sharma and Devesh Jinwala. Functional encryption in iot e-health care system. In *Information Systems Security: 11th International Conference, ICISS 2015, Kolkata, India, December 16-20, 2015. Proceedings 11*, pages 345–363. Springer, 2015.

[70] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*, pages 506–509, 2017.

[71] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. Machine learning in iot security: Current solutions and future challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1686–1721, 2020.

[72] Michael W Berry, Azlinah Mohamed, and Bee Wah Yap. *Supervised and unsupervised learning for data science*. Springer, 2019.

[73] Deepanshu Mehta. State-of-the-art reinforcement learning algorithms. *International Journal of Engineering Research and Technology*, 8:717–722, 2020.

[74] Krunal Bhavsar, Vrutik Shah, and Samir Gopalan. Machine learning: a software process reengineering in software development organization. *International Journal of Engineering and Advanced Technology*, 9(2):4492–4500, 2020.

[75] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[76] Hyeoun Park. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43(2):154–164, 2013.

[77] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. In *2019 international conference on intelligent computing and control systems (ICCS)*, pages 1255–1260. IEEE, 2019.

[78] Jesús Maillo, Isaac Triguero, and Francisco Herrera. A mapreduce-based k-nearest neighbor approach for big data classification. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 2, pages 167–172. IEEE, 2015.

[79] Daniel T Larose and Chantal D Larose. *Discovering knowledge in data: an introduction to data mining*, volume 4. John Wiley & Sons, 2014.

[80] Javed Asharf, Nour Moustafa, Hasnat Khurshid, Essam Debie, Waqas Haider, and Abdul Wahab. A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics*, 9(7):1177, 2020.

[81] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[82] Ulaş Çaydaş and Sami Ekici. Support vector machines models for surface roughness prediction in cnc turning of aisi 304 austenitic stainless steel. *Journal of intelligent Manufacturing*, 23:639–650, 2012.

[83] Preeti Mishra, Vijay Varadharajan, Uday Tupakula, and Emmanuel S Pilli. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials*, 21(1):686–728, 2018.

[84] Sourish Ghosh, Anasuya Dasgupta, and Aleena Swetapadma. A study on support vector machine based linear and non-linear pattern classification. In *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, pages 24–28. IEEE, 2019.

[85] Yuzhen Zhang, Jingjing Liu, and Wenjuan Shen. A review of ensemble learning algorithms used in remote sensing applications. *Applied Sciences*, 12(17):8654, 2022.

[86] Agostino Di Ciaccio, Giovanni Maria Giorgi, et al. Deep learning for supervised classification. *RIVISTA ITALIANA DI ECONOMIA, DEMOGRAFIA E STATISTICA*, 70(1):157–166, 2016.

[87] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. *Ensemble machine learning: Methods and applications*, pages 157–175, 2012.

[88] Haidong Fu and Kexin Qi. Evaluation model of teachers' teaching ability based on improved random forest with grey relation projection. *Scientific Programming*, 2022:1–12, 2022.

[89] Yingli Lou, Yunyang Ye, Yizhi Yang, Wangda Zuo, Gang Wang, Matthew Strong, Satish Upadhyaya, and Chris Payne. Individualized empirical baselines for evaluating the energy performance of existing buildings. *Science and Technology for the Built Environment*, 29(1):19–33, 2023.

[90] Stamatios-Aggelos N Alexandropoulos, Christos K Aridas, Sotiris B Kotsiantis, and Michael N Vrahatis. Stacking strong ensembles of classifiers. In *Artificial Intelligence Applications and Innovations: 15th IFIP WG 12.5 International Conference, AIAI 2019, Hersonissos, Crete, Greece, May 24–26, 2019, Proceedings 15*, pages 545–556. Springer, 2019.

[91] M Gheisari, F Ebrahimzadeh, M Rahimi, M Moazzamigodarzi, Y Liu, PK Dutta Pramanik, MA Heravi, A Mehbodniya, M Ghaderzadeh, MR Feylizadeh, et al. Deep learning: Applications, architectures, models, tools, and frameworks: A comprehensive survey., 2023. *DOI: https://doi. org/10.1049/cit2*, 12180:581–606.

[92] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[93] Andrew L Maas, Peng Qi, Ziang Xie, Awni Y Hannun, Christopher T Lengerich, Daniel Jurafsky, and Andrew Y Ng. Building dnn acoustic models for large vocabulary speech recognition. *Computer Speech & Language*, 41:195–213, 2017.

[94] Iqbal H Sarker. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, 2021.

[95] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2020.

[96] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics*, 2023.

[97] Asad Abdi, Gayane Sedrakyan, Bernard Veldkamp, Jos van Hillegersberg, and Stéphanie M van den Berg. Students feedback analysis model using deep learning-based method and linguistic knowledge for intelligent educational systems. *Soft Computing*, 27(19):14073–14094, 2023.

[98] Samaneh Mahdavifar and Ali A Ghorbani. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347:149–176, 2019.

[99] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[100] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*, 2017.

[101] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[102] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[103] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[104] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019.

[105] Rasheed Ahmad and Izzat Alsmadi. Machine learning approaches to iot security: A systematic literature review. *Internet of Things*, 14:100365, 2021.

[106] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[107] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[108] Abhishek Verma and Virender Ranga. Machine learning based intrusion detection systems for iot applications. *Wireless Personal Communications*, 111(4):2287–2310, 2020.

[109] Hanan Hindy, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, and Xavier Bellekens. Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset). In *International Networking Conference*, pages 73–84. Springer, 2020.

[110] Qi Liu, Hubert B Keller, and Veit Hagenmeyer. A bayesian rule learning based intrusion detection system for the mqtt communication protocol. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–10, 2021.

[111] Muhammad Ahmad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using unsw-nb15 data-set. *EURASIP Journal on Wireless Communications and Networking*, 2021:1–23, 2021.

[112] Satish Pokhrel, Robert Abbas, and Bhulok Aryal. Iot security: botnet detection in iot using machine learning. *arXiv preprint arXiv:2104.02231*, 2021.

[113] Ali Bin Mazhar Sultan, Saqib Mehmood, and Hamza Zahid. Man in the middle attack detection for mqtt based iot devices using different machine learning algorithms. In *2022 2nd International Conference on Artificial Intelligence (ICAI)*, pages 118–121. IEEE, 2022.

[114] Hariprasad Siddharthan, Thangavel Deepa, and Prabhu Chandhar. Senmqtt-set: An intelligent intrusion detection in iot-mqtt networks using ensemble multi cascade features. *IEEE Access*, 10:33095–33110, 2022.

[115] Dhiaa Musleh, Meera Alotaibi, Fahd Alhaidari, Atta Rahman, and Rami M Mohammad. Intrusion detection system using feature extraction with machine learning algorithms in iot. *Journal of Sensor and Actuator Networks*, 12(2):29, 2023.

[116] Abbas Javed, Amna Ehtsham, Muhammad Jawad, Muhammad Naeem Awais, Ayyaz-ul-Haq Qureshi, and Hadi Larijani. Implementation of lightweight machine learning-based intrusion detection system on iot devices of smart homes. *Future Internet*, 16(6):200, 2024.

[117] Muhammad Allah Rakha, Inam Ullah Khan, Mariya Ouaissa, Mariyam Ouaissa, Muhammad Yaseen Ayub, et al. Hybrid model for iot-enabled intelligent towns using the mqtt-iot-ids2020 dataset. In *Cyber Security for Next-Generation Computing Technologies*, pages 159–176. CRC Press, 2024.

[118] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[119] Fatemeh Mosaiyebzadeh, Luis Gustavo Araujo Rodriguez, Daniel Macêdo Batista, and Roberto Hirata. A network intrusion detection system using deep learning against mqtt attacks in iot. In *2021 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2021.

[120] Imtiaz Ullah and Qusay H Mahmoud. Design and development of a deep learning-based model for anomaly detection in iot networks. *IEEE Access*, 9:103906–103926, 2021.

[121] Ali Alzahrani and Theyazn HH Aldhyani. Artificial intelligence algorithms for detecting and classifying mqtt protocol internet of things attacks. *Electronics*, 11(22):3837, 2022.

[122] Shaymaa Mahmood Naser, Yossra Hussain Ali, and Dhiya Al-Jumeily OBE. Deep learning model for cyber-attacks detection method in wireless sensor networks. *Periodicals of Engineering and Natural Sciences*, 10(2):251–259, 2022.

[123] Ahmed Fawzi Otoom, Emad E Abdallah, et al. Deep learning for accurate detection of brute force attacks on iot networks. *Procedia Computer Science*, 220:291–298, 2023.

[124] Asimkiran Dandapat and Bhaskar Mondal. Design of intrusion detection system using ga and cnn for mqtt-based iot networks. *Wireless Personal Communications*, 134(4):2059–2082, 2024.

[125] Makhduma F Saiyed and Irfan Al-Anbagi. Deep ensemble learning with pruning for ddos attack detection in iot networks. *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[126] Md Hasan Shahriar, Nur Imtiazul Haque, Mohammad Ashiqur Rahman, and Miguel Alonso. G-ids: Generative adversarial networks assisted intrusion detection system. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 376–385. IEEE, 2020.

[127] Kazuki Hara and Kohei Shiomoto. Intrusion detection system using semi-supervised learning with adversarial auto-encoder. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8. IEEE, 2020.

[128] Phan The Duy, Nghi Hoang Khoa, Anh Gia-Tuan Nguyen, Van-Hau Pham, et al. Digfupas: Deceive ids with gan and function-preserving on adversarial samples in sdn-enabled networks. *Computers & Security*, 109:102367, 2021.

[129] Tao Hou, Tao Wang, Zhuo Lu, Yao Liu, and Yalin Sagduyu. Iotgan: Gan powered camouflage against machine learning based iot device identification. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 280–287. IEEE, December 2021.

[130] Cheolhee Park, Jonghoon Lee, Youngsoo Kim, Jong-Geun Park, Hyunjin Kim, and Dowon Hong. An enhanced ai-based network intrusion detection system using generative adversarial networks. *IEEE Internet of Things Journal*, 10(3):2330–2345, 2022.

[131] Shanshuo Ding, Liang Kou, and Ting Wu. A gan-based intrusion detection model for 5g enabled future metaverse. *Mobile Networks and Applications*, 27(6):2596–2610, 2022.

[132] Tej Kiran Boppana and Priyanka Bagade. Gan-ae: An unsupervised intrusion detection system for mqtt networks. *Engineering Applications of Artificial Intelligence*, 119:105805, 2023.

[133] Ali Mustapha, Rida Khatoun, Sherali Zeadally, Fadlallah Chbib, Ahmad Fadlallah, Walid Fahs, and Ali El Attar. Detecting ddos attacks using adversarial neural network. *Computers & Security*, 127:103117, 2023.

[134] Sifan Li, Yue Cao, Shuohan Liu, Yuping Lai, Yongdong Zhu, and Naveed Ahmad. Hda-ids: A hybrid dos attacks intrusion detection system for iot by using semi-supervised cl-gan. *Expert Systems with Applications*, 238:122198, 2024.

[135] Scikit learn Developers. Ensemble methods. https://scikit-learn.org/stable/modules/ensemble.html, 2023. Accessed: 14-09-2023.

[136] Behnaz Bahmei, Elina Birmingham, and Siamak Arzanpour. Cnn-rnn and data augmentation using deep convolutional generative adversarial network for environmental sound classification. *IEEE Signal Processing Letters*, 29:682–686, 2022.

[137] Yuhuang Hu, Adrian Huber, Jithendar Anumula, and Shih-Chii Liu. Overcoming the vanishing gradient problem in plain recurrent networks. *arXiv preprint arXiv:1801.06105*, 2018.

[138] P Prabakaran, R Mohana, and S Kalaiselvi. Enhancing the cyber security intrusion detection based on generative adversarial network. elem. educ. *Online*, 20:7401, 2021.

[139] Mohammed Y Alzahrani and Alwi M Bamhdi. Hybrid deep-learning model to detect botnet attacks over internet of things environments. *Soft Computing*, 26(16):7721–7735, 2022.