REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

TRAINED MONITORING FOR IOT NETWORK TO IMPROVEMENT PERFORMANCE NETWORK BASED ON DEEP LEARNING ALGORITHMS

Mays Qasim Jebur AL-ZAIDAWI

Doctor of Philosophy

Supervisor

Asst. Prof. Dr. Mesut ÇEVİK

İstanbul, 2025

TRAINED MONITORING FOR IOT NETWORK TO IMPROVEMENT PERFORMANCE NETWORK BASED ON DEEP LEARNING ALGORITHMS

Mays Qasim Jebur AL-ZAIDAWI

Electrical and Computer Engineering

Doctor of Philosophy

ALTINBAŞ UNIVERSITY

2025

The dissertation titled TRAINED MONITORING FOR IOT NETWORK TO IMPROVEMENT PERFORMANCE NETWORK BASED ON DEEP LEARNING ALGORITHMS prepared by MAYS QASIM JEBUR AL-ZAIDAWI submitted on 17/06/2025 has been **accepted unanimously** for the degree of PhD in Electrical and Computer Engineering.

_____

Asst. Prof. Dr. Mesut ÇEVİK

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Mesut ÇEVİK	Department of Electrical and Electronics Engineering, Altınbaş University	_____
Prof.Dr. Osman Nuri UÇAN	Department of Electrical and Electronics Engineering, Altınbaş University	_____
Asst. Prof. Dr. Hameed Mutlag Farhan FARHAN	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Serdar KARGIN	Department of Biomedical Engineering, İstanbul Arel University	_____
Assoc. Prof. Dr. İzzet Paruğ DURU	Department of Medical Services and Techniques, İstanbul Gedik University	_____

I hereby declare that dissertation meets all format and submission requirements for a PhD dissertation.

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

Mays Qasim Jebur AL-ZAIDAWI

Signature

DEDICATION

I devote and pledge this research work to my supervisor who is salient for guiding me through whole research work as well as my family for always assisting me in my hard time.

PREFACE

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

ABSTRACT

TRAINED MONITORING FOR IOT NETWORK TO IMPROVEMENT PERFORMANCE NETWORK BASED ON DEEP LEARNING ALGORITHMS

AL-ZAIDAWI, Mays Qasim Jebur

Ph.D., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Mesut ÇEVIK

Date: 06/2025

Pages: 98

The introduction of the IoT has forced the integration of billions of devices in different sectors, thereby creating huge data and change. Nevertheless, IoTs have some few challenges when it comes to internetworking and securing the networks. Networking problems like latency, packet loss, congestion and probable security holes make it imperative that networking headers are monitored and monitored are checked for anomalies. This thesis proposes a deep learning-based approach to real-time IoT network monitoring and anomaly detection, focusing on three models: FFNN, CNN, and MLP are the popular categories of Deep Learning Algorithms. The models were created and built with MATLAB to review IoT network data to identify discrepancies, distinguish malfunctioning nodes, and diagnose future problems. In an effort to enhance the outcome of the models, optimization methods of Adam and Stochastic Gradient Descent with Momentum (SGDM) was used. The models were tested on synthetic IoT data and the results highlighted by using the quality control indicators such as accuracy, precision, recall, and F1-score. However, the results show the proposed methods improve the existing results where MLP and CNN have higher accuracy and anomaly detection rates than FFNN, MLP-93 (92.3%), CNN-94 (94%), DT (78.5%), and SVM (85.7%). This considerable enhancement is due to efficiency of CNN for extracting spatial relationship and MLP to learn non-linear relationship in IoT network data. Comparing with other methodologies implemented in the current research, deep learning models provide a higher level of accuracy in the identification of sophisticated abnormalities,

which is beneficial for real-time IoT monitoring. In this case, the CNN model showed remarkable improvement on its capability in the identification of network patterns and the prediction of issues as compared to prior models. These outcomes give a strong signal that deep learning models, including CNN and MLP, are more beneficial for real-time IoT network performance monitoring and anomaly identification than conventional models of machine learning. The work from this research can be generalized as follows to be used in improving the dependability and securability of the IoT networks while giving a much better solution than the current method.

Keywords: IoT, Deep Learning, Feedforward Neural Networks (FFNN), Convolutional Neural Networks (CNN), Multilayer Perceptrons (MLP), Network Monitoring, Anomaly Detection.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

CNN	:	Convolutional Neural Network
CPU	:	Central Processing Unit
DL	:	Deep Learning
FFNN	:	Feedforward Neural Network
GPU	:	Graphic Processing Unit
IOT	:	Internet of Things
LSTM	:	Long Short-Term Memory
ML	:	Machine Learning
NLP	:	Multilayer Perceptron
PDR	:	Packet Delivery Rate
RNN	:	Recurrent Neural Network
SGD	:	Stochastic Gradient Descent

1. INTRODUCTION

The IoT has introduced drastic changes on various industries by providing connectivity to different objects for data acquisition and sharing [1]. This has then brought new dimension of connection and automation into the society. The smart devices extend the intelligently connected spectrum to various domains including home appliances, industrial uses, wearable technology, and environmental sensing, among others; have the potentiality to enhance effectiveness, optimize operations and contribute intelligence in multifaceted arenas [2].

IoT environment consists of a diverse population of smart objects that are used in various purpose and across various domains [3]. Such devices are frequently used in homes, industries, wearable technology and tracking systems, climate monitoring and many other related uses. With reference to contemporary smart homes, house appliances that incorporate IoT technology are capable of device interaction. By this communication, energy use and security can be optimised and various aspects of residential living improved upon. As applied to industrial settings, IoT has the capacity to monitor the health of assets, collect data constantly and provide early indications for repairs [4]. These functionalities have the flexibility of improving productive efficiency and reducing durations of inoperationality. One of the commonly adopted applications of the IoT paradigm, thus the Wearable Technology paradigm entails individuals to access effective solutions to track and control their health status, level of fitness and personal welfare. Furthermore, through the integration of environmental sensors within smart cities improves data acquisition which makes urban planning and resource usage even more effective [5].

The trend potential in IoT is not limited to connection but also offers a great amount of data for insights and analytics in various fields. With the data collected and analysed in those entwined devices, enterprises and institutions may make sound decisions and gain more profound insight about user behaviour [9]. Using analytic techniques enables organisations to improve organisational performance, to augment the clients' experiences, as well as to facilitate innovation of products and solutions.

However, the benefits of the IoT are surrounded by a number of challenges, particularly the need to protect (data privacy, security & performance of these networks)[7]. Cobwebbed devices are vulnerable to cybercrimes due to the increased increase in conncet capability of

more devices. Therefore, it can be inferred that, developers as well as the stakeholder involved in Internet of Things (IoT) should place paramount importance in concern with security mechanisms along with employing high algorithm of encryption and authentication to improve the performance of these networks.

1.1 INTERNET OF THINGS (IOT)

In the year 1990, a significant advancement occurred in the form of the introduction of the first remotely controllable toaster, which served as a trailblazer for the Internet of Things (IoT) inside the realm of consumer gadgets [8]. The aforementioned toaster exemplified a noteworthy paradigm change in the realm of domestic appliances, wherein the ability to establish remote connections and exercise control over said products became a prominent feature. When projecting forward to the year 2000 ten years ahead, there emerged a significant improvement in the Internets of Things (IoT) especially with the integration of the Radio frequency Identification (RFID). All these technologies have made a drastic change in the identification of items hence make manifestation of extensive intelligence possibility in numerous fields.

Thus, the IoT trend, which has emerged on the basis of further advances in technological development, has encouraged people to create continuously across virtually all industries. I would like to emphasise that it has been observed that there is a certain trend toward the emerging of intelligent solutions in different fields. For instance, in finance, IoT brought forecasting and delivery of financial services which are better and suitable for personal use. Likewise in the energy industry, enhancement of smart grids has lead to drastic changes to the generation, distribution and consumption of electricity [9]. The practise of Internet of Things (IoT) applications in the healthcare department has been berry fruitful. Part of this entails drivers such as portable health monitoring devices and remote patient care solutions which make contribute to better quality of the healthcare services.

The Internet of Things has elicited considerable developments in the industrial segment and paved the way to the creation of new applications that hold numerous benefits. From these options, it is possible to say that predictive maintenance is an option that has a great influence on it. Internet of Things (IoT) sensors are employed to ensure that effective monitoring and tracking of the running state of mechanical systems and equipment, making it easier to gather

data that is useful in gauging the maintenance needs of the devices. The success of this proactive strategy not only fits the purpose of minimising time gaps, but also improves performance of maintenance schedules, which in turn yields considerable monetary benefits and better organisational performance.

However, the sensors used in Internet of Things (IoT) have also proven very useful where real time asset tracking is concerned. This involves surveillance of a number of objects for instance goods in transit, containers in the supply chain, and automobiles in the car business. The application of such high levels of supply chain visibility and control has lead to a great revolution in the management of firms' affairs cutting down on losses and increasing efficiencies [11].

iot uses show considerable promise especially within the area of energy effective. IoT systems incorporate the characteristic where energy consumption can be sampled and analysed to identify possible means by which its loss can be reduced or its use optimised. Besides, it also enhances cost cutting and aligns with provisions of sustainability whereby carbon emissions are cut. In Figure 1.1, various Application of Internet of Things.

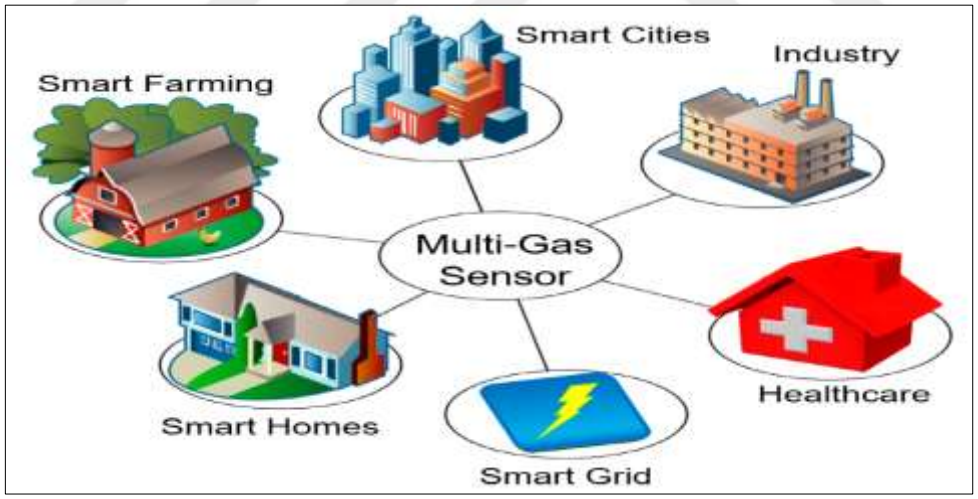

Figure 1.1: Numerous Applications of the Internet of Things (IoT).

Within the domain of industrial processes, the utilisation of Internet of Things (IoT) technology facilitates the capability to remotely monitor and manage activities, hence enhancing efficiency and promoting safety measures. The field of stock management has undergone a significant revolution with the integration of Internet of Things (IoT) devices.

These devices play a crucial role in monitoring stock levels and initiating supply orders in an automated manner when stock reaches a low threshold.

Besides increasing the level of business performance, IoT devices also provide significant contributions in safety concerns identification and elimination, According to Figure 1.2, the security challenges of IoT devices. Real time detection allow for identification of equipment faults, gas leakage among other risks hence allowing for timely provision of early warning to prevent risk to human life and loss of resources.

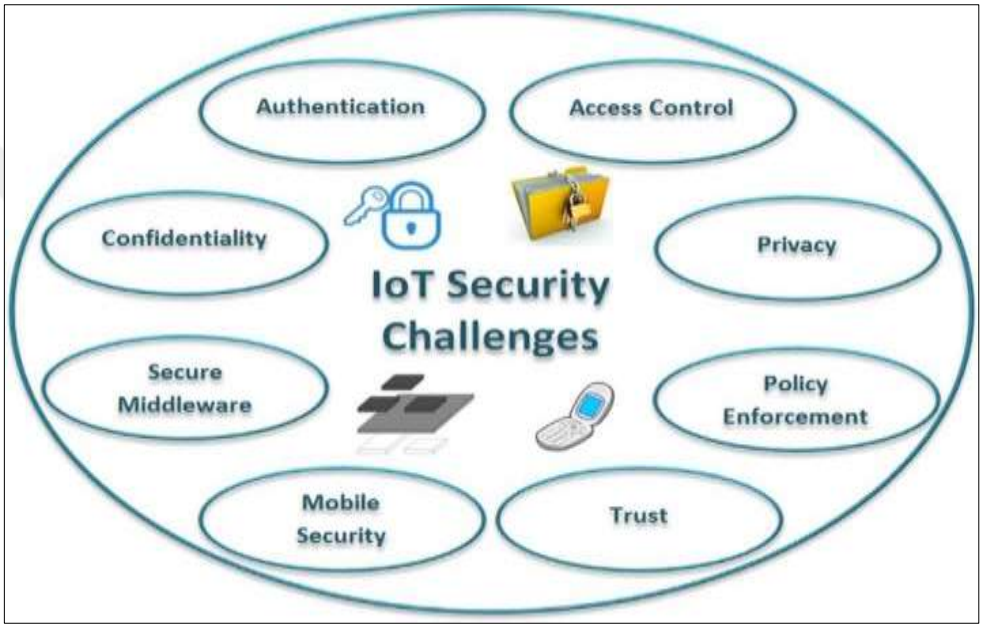

Figure 1.2: IoT Security Challenges.

Even though it is quite clear that there are benefits of implementing IoT in the industrial sector, it is equally important that people acknowledge the issues it brings along with it. The challenges cause effectively many aspects that include firstly the high capital costs which are needed for the foundation of IoT structures and secondly, the security and privacy concerns of data and lastly, the need for specialised skills in the IoT systems development and management. Solving these haciIents is crucial for industries to achieve the full potential that IoT has to offer in practise.

Due to these security problems, various research methodologies and strategies have been researched in order to reduce the security problems. However, the current body of literature often lacks sufficient and diverse reviews mainly owing to the limited number of studies conducted in this specific area. This simply means that in order to deal with this issue, there

is need to undertake a comprehensive and exhaustive analysis that not only seeks to discover insecurity threats, but also searches for remedies that exist in a broader perspective. The enforcement of a multilayered strategy is highly desirable in the case of IoT security since only such a method allows for constructing an effective defence system that would make the use of the threats mentioned above practically impossible. It is critical for organisations to take this form of approach in order to effectively and safely unlock what is essentially a revolutionary technology in many fields today.

1.2 PROBLEM STATEMENT AND SOLUTION

High efficiency and guaranteed data transfer are critical in IoT networks in order to optimize the use of this technology. However, the process of network monitoring in the Internet of Things (IoT) is quite challenging due to the huge volume of data in terms of numbers of devices that are connected as well as their dynamic nature. Current monitoring techniques have limitations in controlling and monitoring the large scale and complex structure of the Internet of Things (IoT) networks which often leads to traffic congestion and delay and inability to detect new problems in the network in real time [11].

To address these challenges and to fully unleash the potential of the Internet of Things, we propose development of an enhanced control system that would include complex deep learning techniques. We have therefore set our goal to enhance network performance and avoid future problems through the use of artificial intelligence and machine learning, thereby producing more reliable and efficient IoT networks. The system that has been trained in this thesis is a monitoring system that will employ deep learning techniques in the processing of a large volume of data collected by the IoT devices. The system will be designed to particularly detect and examine the performance of IoT network. This feature will enable network administrators to easily identify and remedy any potential problems at their early stage of development and hence prevent them from worsening. In addition, because the system will be constantly updated with new data and the past network patterns will be analyzed, it will continue to evolve and therefore the system's performance will increase in terms of accuracy and effectiveness.

1.3 OBJECTIVE OF STUDY

Another important point is that this approach makes it possible to receive predictive analytics in order to prevent the degradation of equipment and manage resources efficiently. Through such analysis it would be possible to spot potential problems or even areas of high traffic congestion and therefore direct resources and avoid future network failure thus providing better service and increased reliability for the system. Hence, we can summarize the goals:

a. Develop a trained monitoring system for IoT networks.
b. Implement and compare three deep learning algorithms: FFNN, CNN, and MLP Network for monitoring IoT network performance.
c. Evaluate and analyze the performance of each deep learning algorithm in terms of accuracy, precision, recall, and F1-score.
d. Visualize the monitoring metrics to provide insights into data flow, packet delivery rate, end-to-end delay, node drop, and high and low node distribution.
e. Propose strategies to optimize the IoT network based on the monitoring results.

Therefore, the aim of this suggested method is to improve the scope of IoT network monitoring by incorporating deep learning algorithms, and it is to transform the management of IoT networks through the development of an advanced monitoring system that possesses the ability to analyse data and detect issues proactively. This technological progress will facilitate the growth of more resilient, effective, and reliable Internet of Things (IoT) implementations across various sectors. Through continuous learning and adaptation, this intelligent monitoring system will establish a prominent role in shaping the development of the Internet of Things, thereby showcasing its true potential for transformative impact.

1.4 EXPECTED OUTCOMES

The expected outcomes of this proposed method are multiple and related to enhancing the overall performance and the reliability of IoT network management. Expected is the emergence of a competent monitoring system that has been able to identify abnormalities and provide real time information. The system will thus afford the network administrators the opportunity of pro-actively managing the network and troubleshooting potential problems before they become problems, thus minimizing the time the network is down and increasing the network's up time.

Moreover, this thesis aims to identify the strengths and weaknesses of the three deep learning algorithms employed for the monitoring of IoT networks through a detailed performance analysis of the algorithms. The analysis that will be carried out in this thesis will therefore enable the various stakeholders in the Internet of Things (IoT) network to make the right decisions as they choose which of the algorithms to use in their networks.

Furthermore, the visual outputs generated at different stages of the proposed method will give clear and easy to comprehend insights about the behaviour of IoT network. The use of diagrams showing flow of data, packet delivery rate, end to end delay and node distribution is going to be helpful in identifying trends and possible constraints that are likely to be observed in the network.

Finally, the proposed method will conclude with practical optimization recommendations based on the analysis of monitoring results and deep learning techniques. The above recommendations give the real-life ways of stepping up the performance of the IoT network, improving the utilization of resources and strengthening the security measures. With the help of these measures organisations are able to develop a stronger and more efficient Internet of Things (IoT) platform. It will lead to better user experiences and opens the door for steady development and evolution of the IoT field.

1.5 CONTRIBUTIONS OF THE STUDY

This study makes several key contributions to the field of IoT network monitoring and machine learning:

i. Demonstration of Deep Learning for IoT Monitoring: The research effectively highlights the use of deep learning models such as CNN and MLP for real-time monitoring of IoT networks proving the effectiveness of these models in identifying anomalies than the conventional models.
ii. Optimization Strategies: It is clear from the research that hyperparameter tuning and regularization techniques should be embraced in an effort to improve model performance, and the study presents a reference point for future research on the optimization of deep learning models for IoT applications.

iii. Synthetic Data for IoT Monitoring: This is because the work created and used synthetic IoT network data which is a controlled environment to train and test deep learning models, this will be useful in future research in light of the lack of real IoT datasets.
iv. Comparison with Traditional Models: The research work also compares deep learning models with the conventional machine learning models in order to demonstrate the efficacy of deep learning in managing IoT network traffic.

1.6 ORGANIZATION OF THESIS

The thesis's remaining chapters are structured as follows:

a. The second chapter provides a thorough analysis of the relevant literature.
b. The third chapter provides an implementation of the methodology.
c. The fourth chapter will display an explanation of the implementation procedure and the results.
d. The fifth chapter analyses the data gathered and draws parallels between the new methodology and the previous methods' conclusions.
e. The sixth chapter presents the conclusions and future work.

2. LITERATURE REVIEW

2.1 INTRODUCTION

The IoT comprises a vast network that consists of a multitude of networked devices capable of exchanging data over the internet. In addition to residential, commercial, transportation, healthcare, telecommunications, and agricultural sectors, these devices have obtained ubiquity and are utilised in many others. By facilitating crucial decision-making processes in industries such as healthcare and transportation, the pervasive use of these technologies has had an important effect on our daily lives as well as proven to be of immense value to these sectors. Corresponding to the findings about IoT research in 2020, the IoT industry is anticipated to experience a significant growth trajectory, with annual sales exceeding $2.4 trillion by 2027 [1]. Although the Internet of Things has generated significant benefits for individuals, society, and industries, the security infrastructure for IoT devices and their connections remains underdeveloped. As the number of connected devices proliferates, the attack surface increases, giving bad actors more prospects to gain unauthorised entry to these devices as well as employ them in huge-scale assaults. As a consequence, manufacturers and end-users face increasingly complex challenges in ensuring the security of Internet of Things devices [2-4]. Researchers have discovered a number of security vulnerabilities, including the use of weak or default passwords and insufficient password protection for online data storage [5]. In many instances, IoT devices are provided with default passwords that are readily predictable or lack any form of password protection. By exploiting compromised IoT devices, the vulnerability disclosed poses a significant risk to user privacy and enables the execution of extensive attacks, such as Distributed Denial of Service (DDoS) attacks [6]. In spite of these notable security issues, IoT platforms offer undeniable advantages that coexist with significant vulnerabilities. Figure 2.1 depicts a variety of extensively employed hardware devices within the realm of the Internet of Things. In addition, Figure 2.2 depicts the IoT's architectural framework.

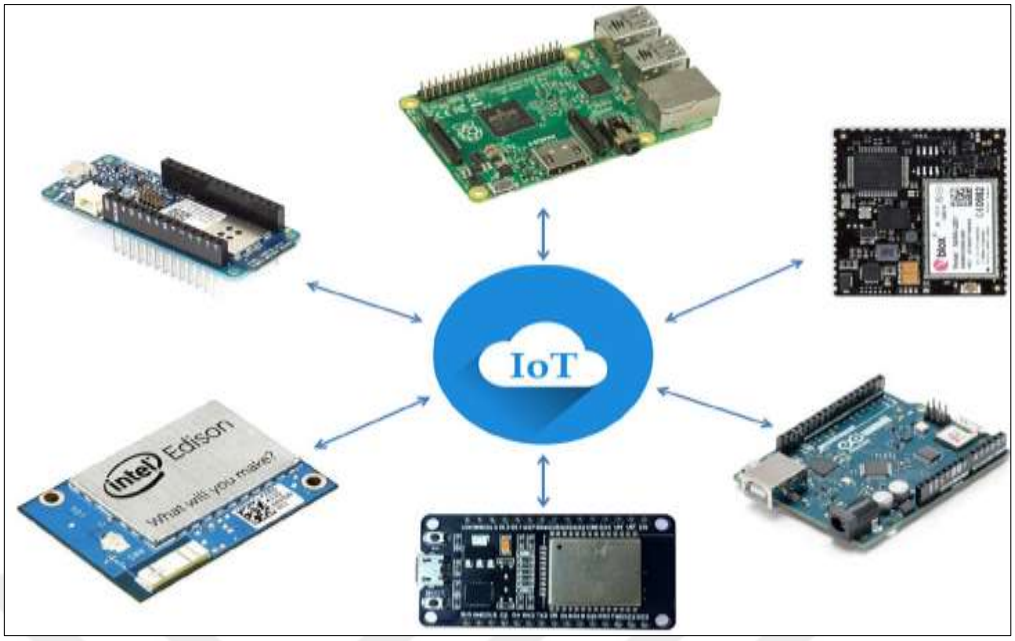

Figure 2.1: IoT Programming Devices.

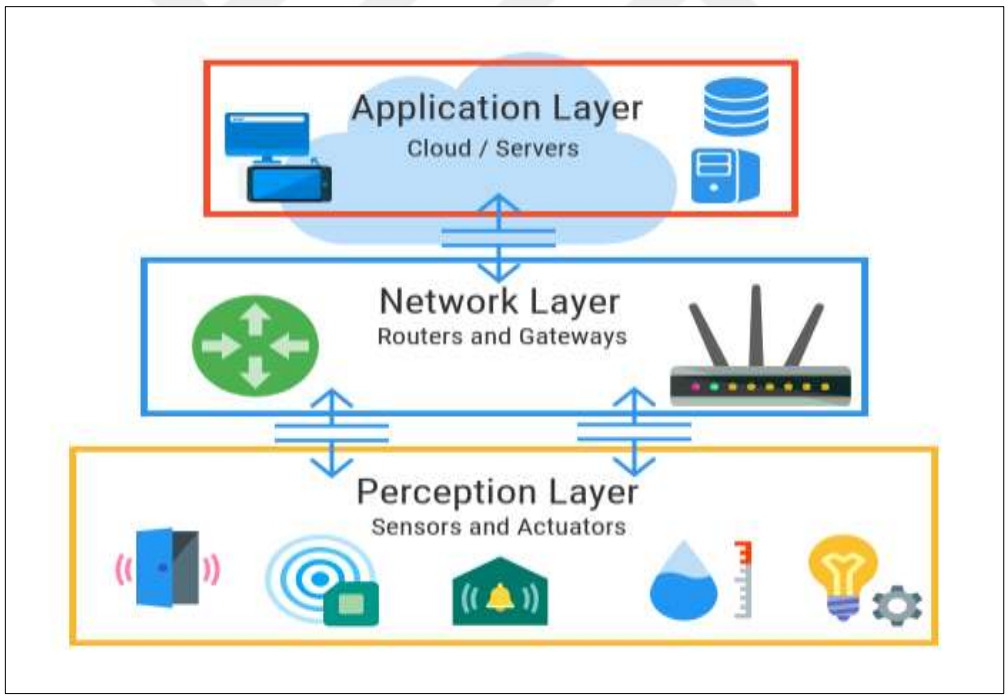

Figure 2.2: Architecture of IoT**.**

The notion of the Internet of Things (IoT) has captivated the academic community owing to its rapid and exponential expansion. The primary aim of this extensive study is to elucidate the latest challenges related to the Internet of Things and to illuminate the most current achievements in the domain. These issues encompass a wide range of topics, including:

The form of our poll is now bifurcated into two halves, each embodying a unique viewpoint. Figure 2.3 illustrates the initial examination of notable improvements in IoT frameworks and associated technologies.

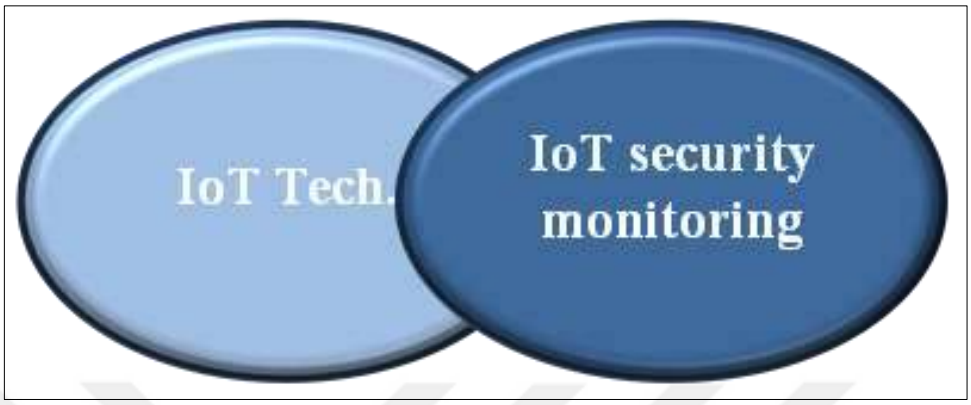

Figure 2.3: Survey Methodology.

2.2 IOT FRAMEWORKS AND RELATED TECHNOLOGIES

The broad term IoT frameworks and related technologies can be described as including all the software, hardware, protocols and infrastructure that are needed for the creation of the IoT environments. These technologies provide the physical connection, data transfer, processing and control of IoT devices; they play a crucial role in determining the efficacy of IoT solutions in industries and applications [7-10]. Explanation of the IoT frameworks and related technologies in the following section.

2.2.1 IOT Frameworks

IoT frameworks are defined as software platforms or architectures that offer development and management support to IoT applications [11]. The companies provide a vast number of tools, libraries, and services for interaction, data acquisition, and control and analysis of the IoT devices. These frameworks are vital to minimize the time taken in the development cycle, to minimize the complexity and to enable efficient interworking within the IoT environment. The important aspects of the device management, data ingestion, data processing, security, scalability, analytics capabilities, connectivity, and integration are the important factors in this regard [12]. Some of the IoT frameworks with examples include; AWS IoT, Azure IoT, Google Cloud IoT Core, and IBM Watson IoT. These frameworks are

intended to support different usage situations and domains of application. These frameworks act as basic building blocks for any IoT developer who needs a foundation on which to build reliable and efficient IoT systems. Furthermore, they also discuss the issues that arise in the management and protection of a large number of IoT devices and data flows [13]. As illustrated in Figure 2.4.

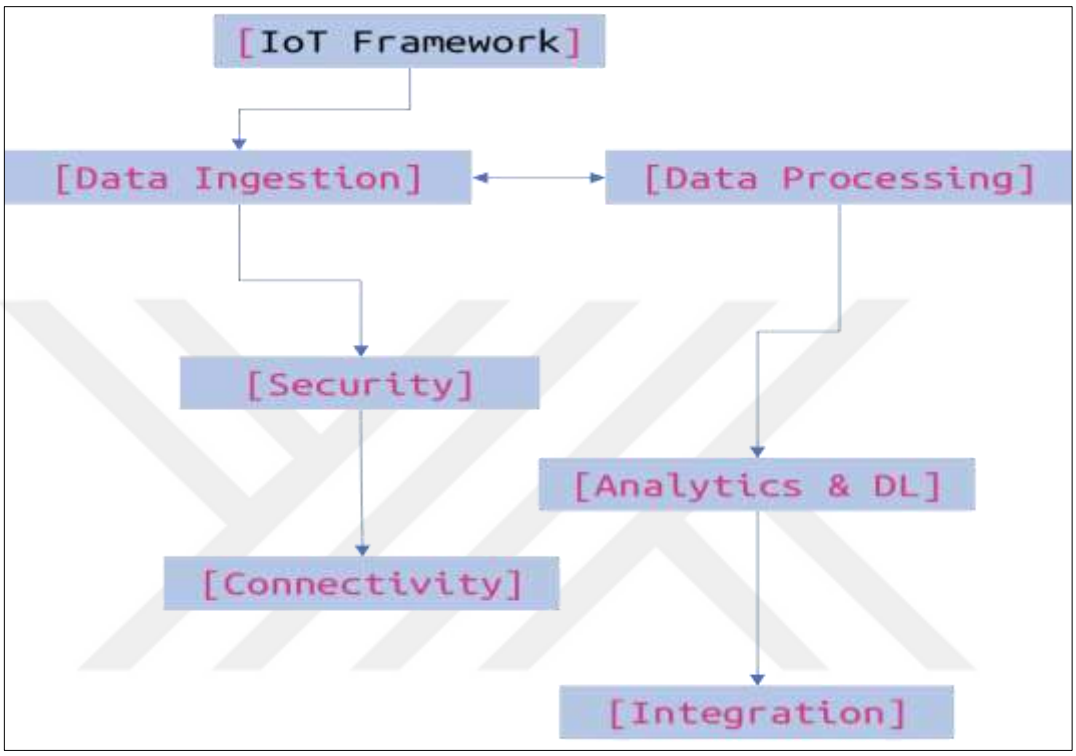

Figure 2.4: Structure of IoT Framework.

Communication protocols in the IoT are a set of rules and guidelines that are followed to enable proper data sharing between the IoT devices [14]. These protocols ensure easy and effective communication among devices without regard to the individual features of these devices or the companies that produced them. The protocols control the structure of the data, ways of transmission, and methods of receiving, thus ensuring that data is transmitted efficiently and securely. They are essential in IoT systems as they are a component which is most crucial in the process of interchanging data where stability, security and efficient use of resources is of paramount importance [15].

Many communication protocols have been proposed to meet the needs and services of the IoT system. The following are some of the well-known IoT communication protocols; MQTT (Message Queuing Telemetry Transport), which is suitable for applications that

require efficient and real time communication [16]; CoAP (Constrained Application Protocol), which is ideal in devices with constrained capabilities [17]; HTTP/HTTPS that is suitable for interoperation with web services [18]; and BLE (Bluetooth Low Energy) that is suitable for applications with short range requirements [19]. In addition, Zigbee is identified as one of the most famous protocols Moreover, XMPP (Extensible Messaging and Presence Protocol) has been discussed for real-time communication and presence management in the smart IoT applications particularly in healthcare and home automation [20]. In combination, these protocols form the foundation for the formation of the IoT network. They provide a means of communication, data transfer and interconnectivity between IoT devices and platforms or services. Furthermore, these protocols are also implemented to meet specific requirements of IoT devices and specific application scenarios [21]. The IoT protocol is presented at Figure 2.5.

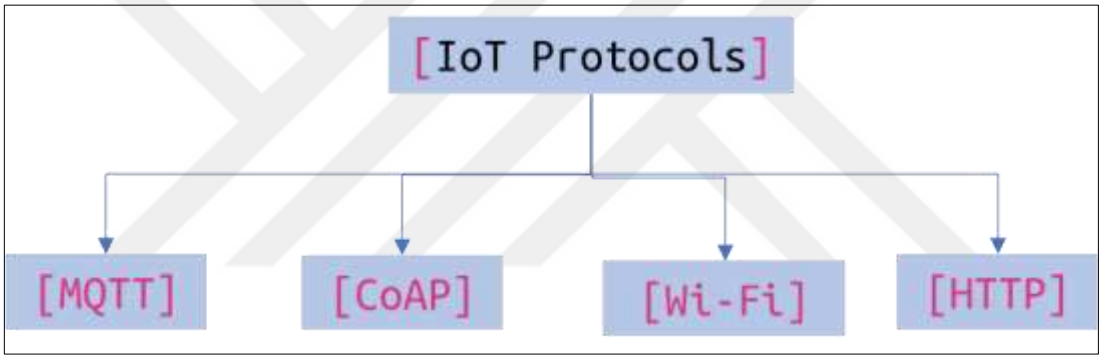

Figure 2.5: IoT Protocols.

2.3 MONITORING OF IOT SECURITY

Monitoring IoT security is an essential component of IoT deployment management and protection [22]. Monitoring and evaluating IoT devices, networks, and systems on an ongoing basis in order to detect and address security threats, vulnerabilities, and intrusions. Hence, implementing robust security monitoring for IoT systems is essential for protecting the privacy, integrity, and availability of both data and services. Where the effective IoT security monitoring contributes to the improvement of system security by mitigating the potential risks associated with cyberattacks and unauthorised access. In the sections that follow, we will examine the fundamental components of IoT security monitoring [23][25].

a. Device Monitoring: Thus, constant control of the health and activity of connected devices is a way to detect the presence of intruders. This means being able to watch over all aspects of a system from the hardware to the firmware to the devices themselves.
b. Network Traffic Analysis: Identifying troublesome activities, intrusion or abnormal traffic activities within the IoT network by analyzing its traffic.
c. Security Event Logging: Storing security information or data from the Internet of Things devices and systems to use in analysis or auditing. This is important in monitoring parameters concerning compliance and in the examination of cases of security violations.
d. Anomaly Detection: To identify deviations from the normal and raise alarm when security threats are detected, one can use anomaly detection techniques for example machine learning.
e. Access Control Monitoring: The control of authentication, authorization, and privileges for users of the Internet of Things devices and systems is being checked to guarantee that only the right people are accessing important resources.
f. Security Patch Management: Ensuring that patches to vulnerabilities in IoT devices are implemented as soon as they are released to the public.
g. Incident Response: Formation and implementation of the incident response plans to identify and manage security incidents, minimize the damages and prevent future breaches.

The approaches and decisions for the IoT security monitoring can be diverse. ”Security Information and Event Management (SIEM)” SIEM systems are used for collecting, analyzing and correlating information coming from numerous sources, including IoT devices, to provide timely detection of security events, and assist in the handling of security incidents. The two software solutions that can be used for data analysis and security monitoring are Splunk and IBM QRadar [26]. In additional, Intrusion Detection Systems (IDS) are software solutions that provide ongoing surveillance of network traffic and device behaviour in order to identify and flag any potentially suspicious actions or attempts at unauthorised access. Snort and Suricata represent instances of open-source Intrusion Detection System (IDS) tools that are commonly employed in the domain of IoT security [27]. In another, Endpoint security solutions are a category of products that prioritise the protection of individual IoT devices. Their primary objective is to safeguard these devices

by actively monitoring their activities, overseeing software updates, and implementing stringent access control measures. Bitdefender and McAfee offer endpoint security solutions specifically designed for IoT devices [28]. Morover, Network monitoring tools such as Wireshark and Nagios can be utilised for the purpose of IoT security monitoring, enabling the examination of network traffic and device behaviour [29].

Specialized IoT security platforms, including Palo Alto IoT Security and Armis, offer comprehensive solutions for monitoring and safeguarding networks and devices linked to the Internet of Things. Anomaly detection in the Internet of Things (IoT) primarily involves the application of machine learning and artificial intelligence. Specifically, tools like Darktrace utilize machine learning methodologies to identify and highlight anomalous behavior [31]. Cloud security services, like AWS IoT Security, Azure IoT Security, and Google Cloud IoT Core, provide surveillance and safeguarding for Internet of Things (IoT) implementations [32]. Commercial and regulatory norms substantially influence the execution of IoT security monitoring [33]. The National Institute of Standards and Technology (NIST) and the European Union Agency for Cybersecurity (ENISA) are esteemed governmental entities that regularly provide guidelines and standards in this domain.

2.4 RELATED WORKS

Table 2.1 regarding IoT research, challenges, and technological advancements are very important. Collectively, these materials contribute to the ever-evolving corpus of knowledge on the Internet of Things, covering diverse topics such as application areas, security considerations, technological advancements, and the intricate structure of IoT architecture. Scholars and professionals in the field of IoT frequently consult these studies to gain a deeper understanding of the challenges and opportunities presented by this revolutionary technology.

Table 2.1: IoT Research Perspectives and Challenges.

Ref.	Description
[32]	examined various investigate viewpoints within IoT, distinguishing between (Internet-specific & Things-specific apps). They highlighted an importance of security in IoT but noted limitations due to energy constraints and the computing capabilities of IoT devices.
[33]	targeted IoT-centric use cases and the difficulties those use cases present. They envisioned a future for simulation that is cloud-centric and investigated potential roadblocks.
[34]	discussed the Aneka software development platform's feasibility to meet IoT requirements and emphasized the significance of cloud security in deploying IoT.
[35]	analysed IoT technology and provided a multi-architecture model to demonstrate IoT's potential industrial use. The importance of the context-aware perspective to the Internet of Things was also emphasised.
[36]	assessed the IoT paradigm from a context-aware standpoint. They explored various IoT proposals, research modeling techniques, and strategies, focusing on security, privacy, and effectiveness at different layers.
[37]	technology, and the roles played by communication protocols geared towards the Internet of Things. By analysing issues including security, performance, privacy, availability, and management, they were able to identify six essential components of the IoT and emphasise the lack of a standardised IoT design as a major obstacle.
[38]	IoT data analytics concentrate on protecting connected devices. Due to the disparity between IoT communication protocols and conventional IT spheres, they stressed the importance of flexible security solutions.
[39]	assessed the alignment of existing security mechanisms with various IoT requirements and goals. They evaluated how these mechanisms address IoT-specific challenges.
[40]	generations of IoT evolution and explored the transformative characteristics and applications associated with each generation. They also discussed advancements in machine learning techniques in the context of IoT.

IoT security has assumed great importance due to the increasing incorporation of IoT devices across a variety of sectors, from residential smart systems to critical infrastructure. The concepts and findings in Table 2.2 cover a wide range of security-related topics, including legal issues, attack scenarios, intrusion detection techniques, and others.

Table 2.2: IoT Security Technologies.

Ref.	Description
[41]	Weber and Studer surveyed IoT threats, discussed legal models, and explored international regulations for IoT security. They noted the changing legal cybersecurity landscape and the European Union's role in development. Practical IoT applicability was found to be limited.
[42]	A comprehensive survey of IoT security mechanisms and found ongoing research challenges. They highlighted the absence of an organized approach to ensure protection.
[43]	IoT attack scenarios and countermeasures were presented using Cisco's 7-stage reference framework. They stressed the need for preventative measures to ensure the security of IoT devices.
[44]	offered a quantitative evaluation of access control methods for the Internet of Things, with special focus on the peculiarities of IoT security. They argued in favour of centralising security for the Internet of Things.
[45]	surveyed vulnerabilities in a decentralized IoT framework, exploring how decentralization could reduce the impact of attacks and increase attack vectors.
[46]	data mining was used to perform an analysis of IoT security, which led to the identification of five areas of IoT security that require improvement. More research into these areas was encouraged.
[47]	focused on botnet propagation and detection in mobile applications, presenting two novel command and control architectures for well-hidden botnet attacks.
[48]	assaults on IoT architectures were compared, which led to the presentation of a six-layer security framework to deal with IoT security issues.
[49]	examined current IoT solutions, presenting threats and vulnerabilities in IoT deployments and categorizing them into four taxonomy classes. They emphasized the challenge posed by IoT device heterogeneity.
[50]	A review of IoT intrusion detection methods that pointed out how computational limitations prevented full study of intrusion detection attacks.

2.5 REVIEW OF TECHNOLOGIES IN IOT

Learning algorithms are gaining greater applicability in various real-world scenarios owing to their remarkable problem-solving abilities. Integrating these algorithms into system development is essential for potential performance improvement via experiential learning [51]. Innovative strategies, extensive datasets, and computationally efficient methodologies have all facilitated recent advancements in learning algorithms [52]. In recent years, developments in machine learning and deep learning have transitioned these concepts from theoretical frameworks to practical applications. In this sense, we refer to classical approaches that require the advancement of technological components when discussing ML. Conversely, deep learning incorporates contemporary techniques that abstract and alter pattern analysis elements via multiple layers of nonlinear processing [54]. Issues pertaining to machine learning (ML) and deep learning (DL) have been categorized to furnish readers with a thorough and comprehensive understanding of these two subjects.

The primary objective of designing learning algorithms is to improve task performance through the acquisition of knowledge from training data and experience. The principal aim of intrusion detection is to enhance the overall efficacy of the classification process, which involves accurately categorizing a system's behavior as normal or abnormal. Other categories of machine learning encompass unsupervised learning, supervised learning, and reinforcement learning (RL), which are the three primary algorithmic classifications. Supervised learning is a branch of machine learning that produces predictions or classifications by utilizing training data and established mappings. The objective of unsupervised learning is to leverage commonalities among unlabeled data sets to generate new classifications. Conversely, the objective of Reinforcement Learning (RL) is to identify optimal solutions to a specific problem by enabling algorithms to learn from their surroundings [55][56].

2.6 PROFESSIONAL IOT SECURITY THROUGH IN-DEPTH LEARNING

This document provides a detailed examination of deep learning algorithms, encompassing their various applications, advantages, disadvantages, and importance in safeguarding the Internet of Things. The incorporation of deep learning into IoT systems has recently attracted significant interest [57]. A primary advantage of deep learning (DL) compared to classical

machine learning (ML) is its enhanced efficacy with large datasets. The extensive data generated by IoT devices renders this benefit essential [58]. Due to their remarkable capacity to autonomously create intricate data representations, deep learning algorithms are well-suited for IoT applications. Deep learning facilitates the connection of pertinent nodes within an IoT ecosystem [59]. Deep linking facilitates seamless communication between connected apps and Internet of Things (IoT) devices, eliminating the necessity for human involvement. In a smart home configuration, IoT devices are permitted to interact with one another. As a result, a comprehensive and cohesive smart home ecosystem has developed. Deep learning methodologies provide a multi-tiered computational framework capable of producing data representations at varying degrees of abstraction. Deep learning algorithms have surpassed conventional machine learning techniques in advanced applications [61]. The deep learning (DL) subset of machine learning (ML) employs multiple nonlinear processing layers to abstract and alter generative or discriminative characteristics for pattern analysis. Often referred to as hierarchical learning methodologies, the previously described deep learning techniques can include hierarchical representations within intricate frameworks. In signal processing, deep learning draws inspiration from the functioning of neurons and the brain. Hybrid deep learning is a learning methodology offered by deep networks that integrates elements of both supervised (discriminative) and unsupervised (creative) deep learning. Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are two prominent deep learning techniques for classification. Hybrid deep learning methodologies encompass a wider array of techniques, including Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), and Generative Adversarial Networks (GANs) [62]. Figure 2.6 illustrates the fundamental components of neural network security.

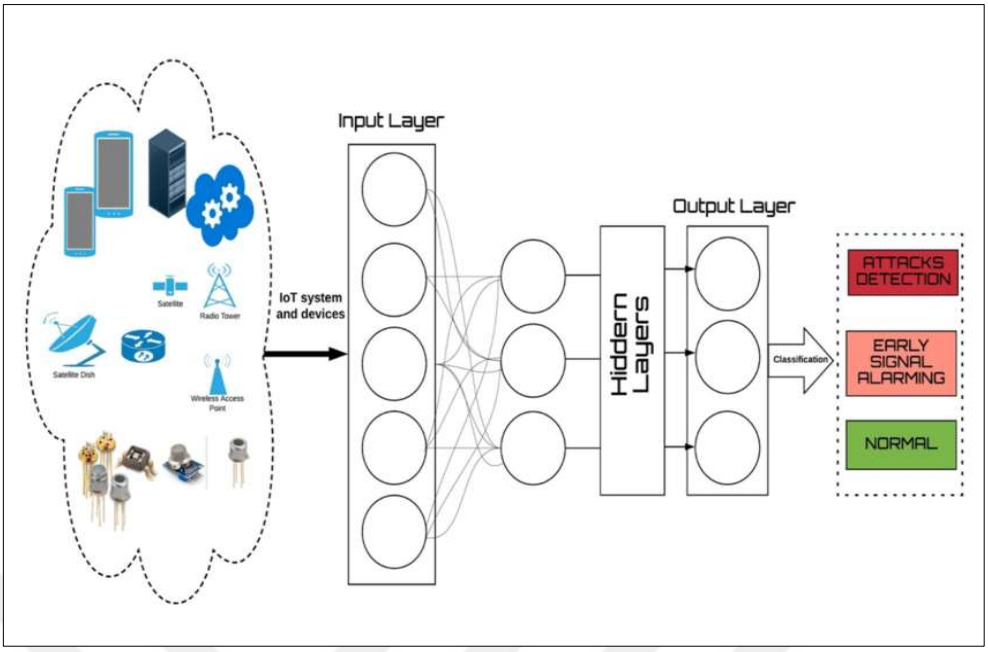

Figure 2.6: Diagram of NNs for IoT Security.

Table 2.3, summarizes all the possible DL techniques for enhancing the safety of IoT techniques in the future. In this research, the table below summarized the benefits, drawbacks, and uses of these DL techniques to the IoT safety.

Table 2.3: Possible DL Approaches to Protecting the Internet of Things.

Algorithms	The principle of work	Pros	Cons	Possible Use in IoT Security
Convolutional Neural Networks (**CNNs)**	CNNs leverage sparse interactions, parameter sharing, and equivariant representations in order to reduce the number of data parameters they need to operate. This method efficiently decreases the number of interlayer connections to levels often found in ANNs.	In terms of performance, CNNs are widely regarded as among the most competitive supervised DL methods.The scalability of CNNs is enhanced and their training time complexity is optimised in comparison to ANNsdue to the incorporation of new features. CNNs possess considerable potential for app in the realm of IoT security. This is primarily due to their inherent capability to autonomously acquire knowledge of pertinent aspects from raw security data.	CNNs are associated with significant computational expenses, making their deployment on devices with limited resources for the purpose of facilitating onboard security systems a formidable task.	The identification of malware is a crucial aspect in the field of cybersecurity. CNNs have the ability to autonomously acquire knowledge about the inherent characteristics of unprocessed security data. Consequently, they possess the capability to develop a comprehensive security framework for IoT.

Table 2.3: Possible DL Approaches to Protecting the Internet of Things (Table Continued).

Recurrent Neural Networks (**RNNs)**	RNNs have a temporal layer built in, which allows them to handle sequential data. Through the use of hidden units within the recurrent cell, RNNs are capable of learning diverse variations in a multi-faceted manner.	RNNs and its various adaptations have demonstrated exceptional efficacy in numerous applications involving sequential data. In specific instances, the data pertaining to IoT security may exhibit a sequential nature. Consequently, RNNs possess the potential to be applied in the field of IoT security.	The difficulty of dealing with vanishing or expanding gradients is a major drawback of RNNs.	It have demonstrated a significant capability in accurately classifying network data by effectively detecting harmful behaviours. RNNs and their various adaptations exhibit significant promise in enhancing the security of IoT systems, particularly in addressing attacks that are based on time series data.

.

3. METHODOLOGY AND IMPLEMENTATION

3.1 INTRODUCTION

The objective of this chapter is to elucidate the research methodologies utilized in this study in considerable detail. The methodology is crucial for achieving the objectives outlined in Chapter One, as it delineates the techniques, methods, and resources employed to develop the trained monitoring system for IoT networks. This chapter will detail the specifics of the proposed technique for monitoring IoT network performance with deep learning. This thesis focuses on the design, data, pre-processing, feature extraction, and application of three deep learning models: the Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Feedforward Neural Network (FFNN).

The fundamental objective of this method is to enhance the performance of IoT networks swiftly and effectively. The aggregation of data from interconnected devices in an Internet of Things (IoT) environment may result in delays, congestion, and suboptimal performance. This thesis introduces an improved system that employs deep learning approaches to attain this objective. This system can oversee data from IoT networks and detect potential issues proactively, enhancing network reliability.

This chapter aims to deliver a thorough analysis of the proposed technique. From the outset, we delineate the study's architecture and elucidate our rationale for employing deep learning techniques to analyze IoT network monitoring data.The subsequent sections illustrate the data gathering necessary for the proposed deep learning models by detailing essential IoT network metrics. The subsequent section elaborates on the data cleansing procedure and the extraction of critical components to improve the precision of network performance evaluation.

Subsequently, the approach provides the structure and training protocol for each deep learning model discussed in the article. This chapter elaborates on the training parameters, network architecture, layer quantity, neuron count, activation function types, and algorithms (FFNN, CNN, and MLP) discussed in Chapter 3.The performance of these models and their suitability for IoT network monitoring have been evaluated using various measures, including accuracy, precision, recall, and F1-score.

The technique examines the implementation of the developed models in a real Internet of Things context. A management system that integrates the models can detect and evaluate alterations in the network's operation in real time. The system evolves in response to network changes by consistently updating the models with fresh data. This ensures the monitoring procedure remains current. At the conclusion of the chapter, we assess the general structure and go to the subsequent chapter, where we will examine the results of the implementation and the associated arguments.

3.2 RESEARCH DESIGN

This study employs a pre-experimental research design alongside an applied research strategy. This study primarily seeks to investigate the design, implementation, and assessment of deep learning models for the management of Internet of Things (IoT) networks. We opted to employ an experimental methodology to assess diverse deep learning techniques on complex and real-time IoT data. This study utilizes three models—Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Feedforward Neural Network (FFNN) to enhance the performance of IoT networks, particularly in the detection and management of anomalies.

This issue is best approached experimentally as it allows for enhanced factor control and systematic adjustments to deep learning models to evaluate their effectiveness. The study aims to illustrate the effectiveness of the models in a controlled experimental setting by examining the IoT using network metrics, including packet delivery rate, latency, and node density. The study's validity was augmented by its ability to compare the performance of multiple algorithms on the same dataset utilizing uniform network configurations.

Quantitative methods were utilized because of the numerical nature of the data gathered for this study. Metrics such as F1-score, recall, accuracy, and precision are utilized to evaluate the effectiveness of each deep learning model. These indicators facilitate the optimization and monitoring of network performance, as well as the comparison of the efficacy of various algorithms. A quantitative approach is more appropriate for managing the numerical data present in IoT networks, encompassing measures such as traffic flow, device interactions, and performance evaluations.

The study is more likely to meet its generalizability objective if a quantitative method is employed. Implementing the algorithms on extensive datasets and rigorously confirming the results may render the study's conclusions applicable to other IoT environments with analogous situations. This method facilitates the formulation of reliable conclusions concerning the most effective deep learning strategy for IoT network monitoring by comparing models and pinpointing instances of statistically significant performance differences.

This research predominantly employs quantitative methodologies and adheres to an applied experimental methodology. The utilization of actual IoT network settings ensures the findings are practically relevant, enabling the authors to perform accurate and controlled experiments for the evaluation and comparison of deep learning models. The study employs a quantitative methodology to demonstrate the potential enhancement of IoT networks through deep learning.

3.3 DEEP LEARNING ALGORITHMS

The capacity of deep learning to identify patterns in data is a fundamental aspect of machine learning that has propelled its rapid rise in popularity. This computational model, based on artificial neural networks, aims to replicate brain function. Whereas most machine learning algorithms rely on pre-defined features, deep learning models derive representations from raw data, enabling them to more effectively manage extensive datasets in high-dimensional domains. These models employ a network of interlinked neural layers, with each layer processing data in a specific manner and learning to predict future data based on its own characteristics. Deep learning's efficacy across various domains stems from its ability to autonomously learn features, enabling it to surpass conventional methods in tasks such as voice recognition, image classification, and natural language processing. Recently, deep learning has been extensively employed in cybersecurity and networking to identify anomalies, optimize resource management, and predict network behavior, thereby enhancing system performance.

The Internet of Things (IoT) networks are expected to generate substantial data from numerous interconnected devices; this research utilizes deep learning to develop a monitoring system for these networks. In the context of IoT networks, deep learning can address challenges such as latency, congestion, and performance degradation. Deep learning

models can assess various network features, identify potential performance issues, detect anomalies, and manage real-time network operations.

The research examines three distinct types of deep learning models: FFNN, CNN, and MLP. Each of these models can be employed to address the problem in a distinctive manner, and all are beneficial. The subsequent section provides a concise summary of each algorithm, accompanied by a rationale for their selection.

3.3.1 Overview of Selected Algorithms

The authors utilized three deep learning methodologies in their IoT network monitoring solution. Three distinct neural network methodologies exist: FFNN, CNN, and MLP. The distinctive data processing and analytical advantages provided by each of these models render them optimal for the network architecture of the Internet of Things. We contend that these methods will improve the ability to monitor network performance and promptly discover problems.

3.3.2 Feedforward neural network (FFNN)

No artificial neural network is complete without its fundamental "feedforward neural network" (FFNN). In a feedforward neural network (FFNN), data progresses linearly from the input layer via the hidden layers to the output layer. Fundamentally straightforward and resilient, feedforward neural networks are ideally suited for a wide array of regression and classification tasks. During training, we allocate weights to each neuron and connect them to all adjacent neurons in the layer beneath. The formal expression for each neuron's output is (Eq. 3.1), representing the weighted sum of the inputs augmented by a bias term:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{3.1}$$

Where z is the weighted sum, w_i are the weights for each input, $x_i$are the inputs, and b is the bias term. The weighted sum is then passed through an activation function, which introduces non-linearity into the model. Common activation functions include the Rectified Linear Unit (ReLU), which outputs $a = max(0, z)$, and the Sigmoid function, which outputs ($a = \frac{\{1\}}{\{1 + e^{\{-z\}}\}}$), where a is the activation output. These activation functions

enable the model to capture non-linear relationships within the data, making FFNNs highly versatile for various tasks.

Due to their comparative simplicity, FFNNs were used for this study. Due to their superior computational speed compared to other models, they are optimal for scenarios with limited machine processing capacity. When data patterns are readily identifiable, Feedforward Neural Networks (FFNNs) effectively manage fundamental network parameters such as bandwidth utilization or packet delivery ratio in predictive applications. The advantages of FFNNs render them an effective initial framework for evaluating the effectiveness of IoT networks. Figure 3.1 illustrates the architecture, input layer, two hidden layers, and output layer of a Feedforward Neural Network (FFNN). The activation functions employed by each node, such as sigmoid or ReLU, are interconnected by weighted edges.

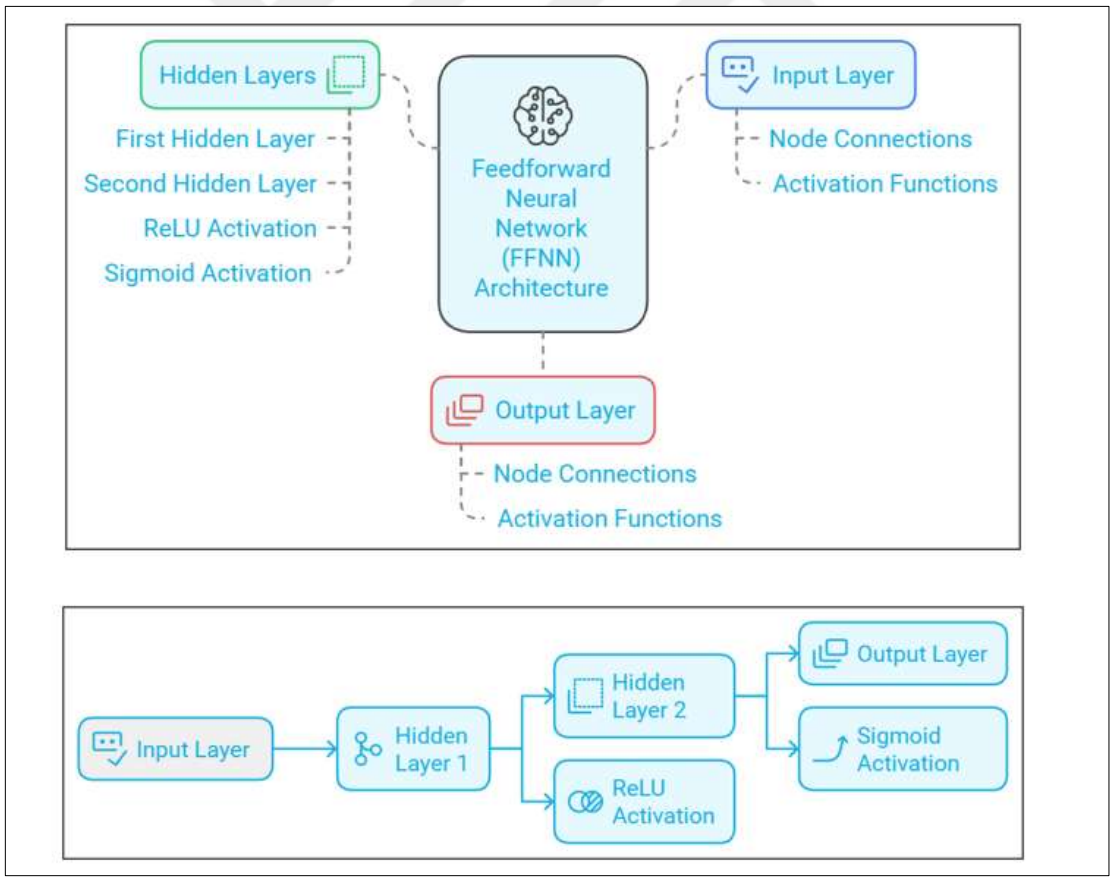

Figure 3.1: Architecture of FFNN, Illustrating the Input Layer, Two Hidden Layers, and Output Layer.

3.3.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNNs) are a type of Neural Network specialized in dealing with grid ignited data including images. Convolutional layers are particularly useful in detecting spatial hierarchies in the data which makes them famous. These layers employ filters to identify elements of the data such as edges or textures and then these features are integrated and forwarded to the next layers to be further processed. One of the biggest advantages of CNNs is their ability to identify features, which makes them ideal for spatial or temporal analysis.

The main structure of CNN is convolutional layer in which a filter, sometimes called a kernel, is slide over the input image or feature map. The convolution operation is defined in (Eq. 3.2):

$$z_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} W_{m,n} * x_{i+m,j+n} + b \quad (3.2)$$

Where:

- $z_{i,j}$ is the output at position i, j,
- W is the filter (kernel),
- x is the input image or feature map,
- b is the bias term,
- $M * N$ is the size of the filter.

Besides convolutional layers, CNNs usually apply pooling layers to decrease the size of the feature maps and decrease the variety of parameters of the model. One of the frequently used pooling operations is Max Pooling, where the maximum value in the pool is chosen. The max pooling operation can be represented in (Eq. 3.3):

$$z_{i,j} = max_{(m,n)\in P} x_{i+m,j+n} \quad (3.3)$$

Where P is the pooling window.

The reasons for selecting CNNs for IoT network monitoring include that CNNs are capable of identifying and classifying the patterns of the network traffic. IoT networks generate a large number of data flows, and in many cases, data has spatial and temporal dependencies such as, for example, behaviors of devices, traffic hotspots, or changes of network performance over time. These patterns and anomalies can be discerned by CNNs and help

in making better and more preemptive predictions as to network performance failures. Thus, the ability to adjust the architecture of CNNs to different types of data is particularly suitable for the diverse IoT networks. Figure 3.2 provides the general overview of CNN structure including the Convolutional layers, Pooling layers, Fully connected layers and the Output layer, in order to explain how Filters learn features in images or feature maps.

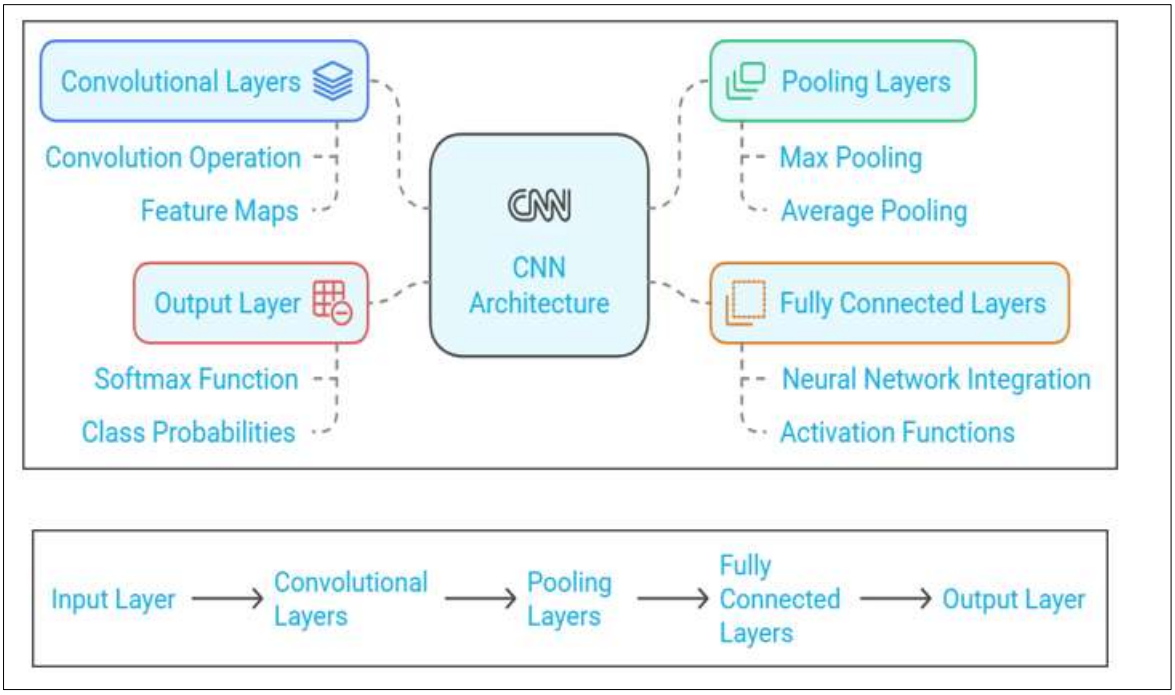

Figure 3.2: Architecture of CNNs Illustrating Convolutional, Pooling, Fully Connected Layers, and Output Layer, Highlighting Feature Extraction Through Filters.

3.3.4 Multilayer perceptron (MLP)

The Multilayer Perceptron (MLP) is also a feed forward neural network but it has one or more hidden layers between the input and output layer. This architecture enables MLPs to model higher order relationships in the data than the previous architecture. Each layer in the MLP network uses the non-linear activation function to manipulate the input, thus allowing it to capture more complex representations that may not be easily discernible to linear models. Nonlinear MLPs are very effective for tasks where the data has nonlinear dependencies and as such, MLPs are used across many areas.

Like the FFNN, MLP uses forward propagation in which the output of each hidden layer is computed by the dot product of the weights and the input activities, added to a bias. This can be expressed in (Eq 3.3):

$$z^I = W^I a^{I-1} + b^I \tag{3.4}$$

Where:

- z^I is the weighted sum at layer I_i,
- W^I is the weight matrix for layer I_i,
- a^{I-1} is the activation from the previous layer,
- b^I is the bias term.

The activation function used in the weighted sum can be of two types ReLU or Sigmoid. The ReLU function is defined in (Eq 3.4):

$$a^I = max(0, Z^I) \tag{3.5}$$

While the Sigmoid function is defined as:

$$a = \frac{\{1\}}{\{1 + e^I\}} \tag{3.6}$$

These activation functions assist in the MLP to learn the non-linear relationship between the input and the output thus allowing the model to solve complex problems. For this reason, MLPs were chosen for this study given their efficiency in dealing with the nonlinear relationships typical of IoT network performance data. IoT systems comprise of many devices that produce data which interconnects in a very fluid manner. The design of these networks can be complicated and may result in phenomena whose effects are not directly proportional to the individual network parameters like latency, bandwidth and packet loss. MLPs can capture these intricate associations and will give better monitoring and prediction results than conventional models. Also, MLPs are one of the most common and effective algorithms to solve many machine learning problems, which strengthens the reason for their use in this study. Figure 3.3 Architecture of MLPs to show the input layer hidden layer and output layer with the weights and bias connection. Furthermore, a backpropagation diagram illustrates the backpropagation propagation of errors in weight updates during the training.

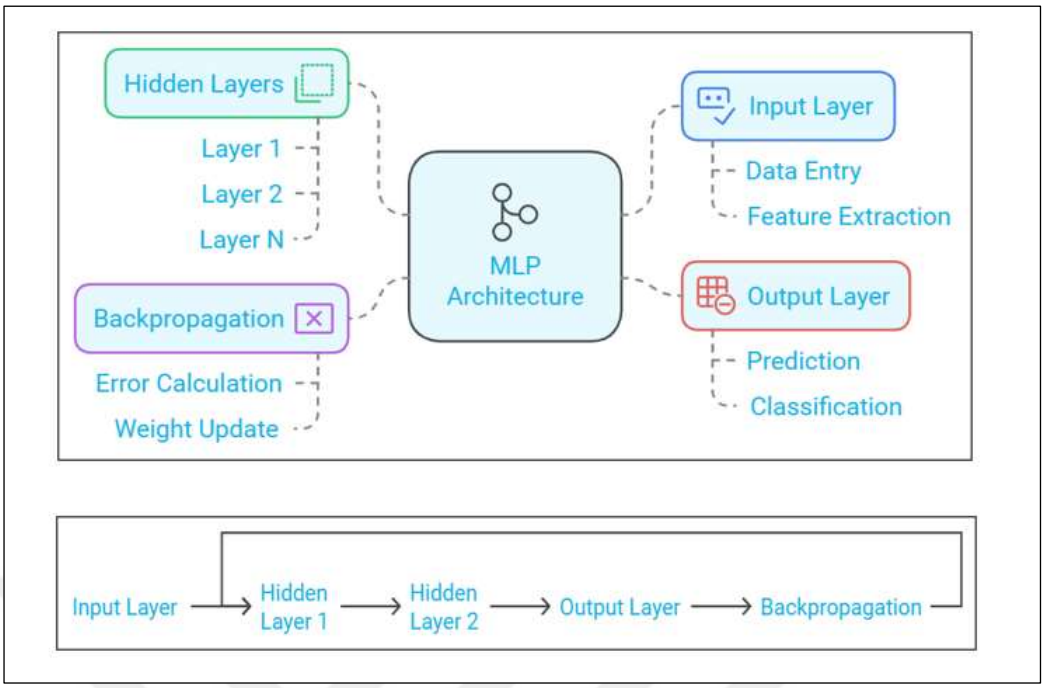

Figure 3.3: Architecture of MLP, Including Input, Hidden, and Output Layers.

3.3.5 Algorithm Architectures

The architecture of the deep learning models adopted in this study is crucial for enhancing the reliability of performance evaluation of IoT networks. In this section, the number of layers in the models and number of neurons in each layer, activation functions, loss functions, and the optimization algorithms applied for training the models has been described in detail. All of the components are crucial for the effectiveness of how the models are able to learn from the data and predict new data within the IoT context.

3.3.6 Network architecture Details

3.3.6.1 Layers

All the deep learning models employed in this study that include the feedforward neural network (FFNN), convolutional neural network (CNN) and multilayer perceptron (MLP) have different frameworks depending on the task they are intended to accomplish.

a. Feedforward Neural Network (FFNN): The FFNN applied in this work comprises of an input layer, two hidden layers and an output layer. The input layer takes the IoT network data, the hidden layers analyze the data and the output layer gives the prediction or

classification (e.g., network performance problems). This is because the two hidden layers help the network to learn more features than the single layer architecture.

b. Convolutional Neural Network (CNN): The CNN architecture is more elaborate and includes an input layer, convolutional layers, pooling layers, and finally, fully connected layers. The convolutional layers on the other hand are used in the extracting of spatial features from the network data and the pooling layers are used in down-sampling the feature maps to decrease the computational intensity. Lastly, the fully connected layers are used to give the final prediction based on the learned features.
c. Multilayer Perceptron (MLP): The MLP has one input layer, three hidden layers and one output layer. This layering of the architecture enables the model to capture the complex interactions between the IoT data. The extra hidden layers enhance the complexity of the model and therefore increase its ability to identify complicated interdependence between network parameters.

3.3.6.2 Neurons

The size of each layer of neurons is chosen in order to meet the requirements of the given data as well as the adopted task.

a. FFNN: The first layer of the neural network has 64 neurons and the second layer has 32 neurons. The number of neurons is decreases as we move deeper into the network to capture higher level features at lower computational costs.
b. CNN: Every convolutional layer has 32 filters or neurons. These filters are used to identify various patterns in the input data like the characteristics of the network traffic. The number of filters is doubled in the deeper layers because more complex features need to be captured. The last few layers are completely connected layers of 128 and 64 neurons which are used to collect features before the final output.
c. MLP: The membership of hidden layers in the MLP model is 128, 64 and 32 respectively. The decreasing number of neurons allows the model to work only with the most significant data features, but without overloading.

The number of neurons in each layer is selected experimentally and it is a matter of tradeoff between model complexity and computational resources.

3.3.6.3 Activation functions

The activation functions used in the models introduce non-linearity, allowing the models to learn more complex patterns in the data. The most commonly used activation functions in this study are ReLU (Rectified Linear Unit) and Sigmoid**.**

a. ReLU: ReLU is used in all hidden layers of the FFNN, CNN, and MLP models. ReLU is defined as ReLU(z)=max(0,z). It is chosen because it is computationally efficient and helps mitigate the vanishing gradient problem during backpropagation, allowing the models to learn faster and more effectively.

b. Sigmoid: The Sigmoid function, $a = \frac{\{1\}}{\{1 + e^{-z}\}}$ is used in the output layers for binary classification tasks, such as predicting whether network performance is within normal limits or if there is an anomaly. Sigmoid squashes the output between 0 and 1, making it suitable for binary decisions.

The choice of activation functions is predetermined by the type of problem and the requirement for optimization of the time spent on the learning process.

3.3.6.4 Loss functions

The loss function is used to compare the predicted value by the model with the actual target and thus help the learning process.

a. FFNN and MLP: The mean squared error (MSE) loss function is used in regression problems while the goal is to predict numerical values including latency or bandwidth consumption. MSE estimates the average squared deviation from the predicting value to the actual value which gives a continuous gradient for back propagation.

b. CNN: For classification problems including anomaly detection in network traffic, the binary cross-entropy loss function is used. This loss function estimates the distance between the predicted probability density function and the actual density function, and is suitable for use in problems where the output variable is a binary variable (such as normal and abnormal network traffic.

3.3.6.5 Optimization algorithms

The models are trained with the help of optimization algorithms provided in MATLAB's `trainingOptions` which aims at minimizing the loss function by adjusting the weight and

bias of the network. For the optimization part in this study, the Adam optimizer and Stochastic Gradient Descent with Momentum are employed.

a. Adam (Adaptive Moment Estimation): Adam is applied in all models because of its performance and the fact that it can adjust the learning rate in the course of training. In MATLAB, the Adam is used with the help of `adam` option within the `trainingOptions` function. Adam is a variation of two other commonly used optimization algorithms, RMSprop and SGD with momentum and is suitable for use with noisy gradients and sparse data, which is frequent in IoT network data. Adam is cheaper in terms of computational cost and has lesser memory usage and hence perfect for use in big data.
b. Stochastic Gradient Descent with Momentum (SGDM): SGDM is also used in some experiments. This optimizer is available as a 'sgdm' option in trainingOptions in MATLAB. Although it is slower than Adam, when fine-tuning models for the best outcomes, it is better to fine-tune with SGDM and update more frequently to ensure better convergence.

There are different optimizers and the choice of optimizer depends on the requirement of either faster training or more stable convergence. Adam is usually chosen due to its speed while SGDM might be used for a more detailed regulation of the learning process.

3.3.7 Training Process

Deep learning models go through a training process that is carefully laid down to enable the models learn in the best way possible and be able to perform well on new data. This section describes the important parameters of the model and techniques for controlling overfitting that were applied when training the Feedforward Neural Network (FFNN), Convolutional Neural Network (CNN), and Multilayer Perceptron (MLP) models for monitoring the performance of an IoT network.

3.3.8 Hyperparameters

3.3.8.1 Learning rate

Learning rate is a very important hyper parameter that controls how the model adjusts its parameters with the gradient eventually computed. If the learning rate is set too high the model may not converge, and if the rate is set too low then it will take a very long time to

learn. In this work, we fixed the learning rate to 0.001 for all models at the beginning. This value stands in the middle of the spectrum of the two, enabling the models to settle efficiently on a good solution without overshooting the best solution.

In this work, some precautions were also taken such as adjusting the learning rate where learning rate was reduced as the number of epochs increased. This is useful to bring more focus at the later stage of training where the big changes to the weight are not required. The learning rate is halved after every 10 epochs if the validation loss does not decrease in order to avoid over fitting.

3.3.8.2 Batch size

Batch size means the amount of training examples for which parameters are optimized before they are updated. In this work, a batch size of 32 was used for training the models. This choice is made to consider the memory usage and the computational cost for the implementation. Larger batch size results in less frequent updates but more accurate gradient estimates; on the other hand, small batch size updates the parameters more often and may introduce noise to the gradient estimates. A batch size of 32 is a good balance to some extent, because it can give models more accurate gradient estimation and also keep training efficient.

3.3.8.3 Epochs

The models were trained for 50 epochs that is the entire training data set was feed into the network for 50 iterations. The value of this parameter was determined empirically because it was found out that the model's performance does not increase significantly after 40-50 epochs. To avoid this, early stopping was used in the experiments as well. The early stopping checks on the validation loss and if the loss does not decrease for the subsequent five epochs then the training is stopped. This is useful for preventing overfitting as it pauses the training process as soon as the model's validation set performance fails to improve, hence, the model will perform well when applied to new data.

3.3.8.4 Regularization techniques

Regularization is a condition where the training data is learned so well that the model fails to produce expected results on new data set. It happens when the model learns not only the patterns in the data, but also the random fluctuations in the data.

3.3.8.5 Dropout

In an effort to control overfitting, dropout was applied during training in all the models on the hidden layers. Dropout is a form of regularization in which for each layer in the neural network a certain fraction of neurons is ignored during training in a random manner. In this work, a dropout of 0.5 was used in the sense that 50% of the neurons in the hidden layers were randomly omitted in each forward pass. Dropout is most helpful in models which contain lots of neurons, such as the CNN and MLP structures as these models are most likely to overfit due to their large size.

3.3.8.6 Weight decay

Another form of regularization technique employed in this study is L2 regularization also referred to as weight decay. L2 regularization is a method that applies a quadratic penalty on the size of the model's parameters which helps to avoid overfitting. The penalty term is simply a function of the sum of squared weights, which aids in the regularization of the learning process to avoid overfitting. A small weight decay of 0.0001 was used in all the models to prevent weights from growing too large and making the models more capable of generalizing.

3.4 DATA COLLECTION

Data collection is a crucial step in the development and evaluation of the deep learning models used for monitoring IoT network performance. The quality and relevance of the data directly impact the effectiveness of the models, as they rely on the input data to learn patterns, detect anomalies, and make predictions. This section outlines the data sources used in the study, describing the type of data collected, its origin, and how it relates to the overall goals of monitoring IoT network performance.

3.4.1 Data Sources

3.4.1.1 Dataset description

The dataset for this study includes IoT network performance parameters that were collected from simulated IoT network and real IoT network environment. This approach means that the data generated by the model can capture different network conditions while at the same time making it possible to create scenarios that may not always be possible in real life. In this work, the authors use both synthetic and genuine data to assess the deep learning models and their performance.

a. Simulated Data: A majority of the datasets were obtained by emulating IoT scenarios with the help of network simulation software OMNeT++. This simulator is popular in networking research because they enable the network conditions to be manipulated to a specific level, including traffic, device behavior and congestion. The simulated data includes different parameters of IoT, including packet delivery rate, end-to-end delay, bandwidth consumption, and packet loss. To this end, the following are important in understanding the performance of IoT networks especially in crowded and ever changing environments with many device to device interactions.
b. Real-World Data: Besides the synthetic data, real IoT network data was also used with the help of publicly available datasets; among which, IoT-23 dataset that contains the network traffic obtained from IoT devices under normal and compromised scenarios was used. This dataset also contains information on network performance parameters including latency, throughput, jitter and device connectivity in real time. The use of real data is important for the practicality of the models as they have to work in the real world IoT environments where the network behavior is fairly random and diverse.

3.4.1.2 Relevance of data

The data collected in this study is quite realistic for IoT network performance metrics and can be used as a basis for the study's objectives of enhancing network monitoring using deep learning approaches. IoT networks include a tremendous number of devices that produce various kinds of traffic and performance indicators. These datasets are chosen to encompass a range of network scenarios, including both normal traffic and abnormal traffic flows like network congestion, device malfunction and intrusion or unauthorized access.

a. Network Performance Metrics: The data is based on KPIs that are crucial for determining the state and productivity of an IoT network. These metrics include:

i. Packet Delivery Rate (PDR): The fact that is of utmost importance for assessing the quality of the network is the packet delivery ratio, which represents the ratio of data packets successfully delivered to the intended recipient.

ii. End-to-End Delay: The aggregate time which a data packet requires to reach the destination from the source, which is an indication of network delay.

iii. Bandwidth Utilization: The volume of traffic in the network at a certain time period as a way of determining the network's performance and capability.

iv. Packet Loss: An important indicator of network quality, namely the ratio of transmitted data packets that are not received by the recipient.

b. Normal and Anomalous Data: Some common features of the normal network traffic are also included in the dataset along with some abnormal conditions that are required for the training of deep learning models to recognize abnormal behavior. The occurrence of unusual traffic patterns, packet loss, and latency fluctuations guarantees that the models identify possible network challenges before they become problematic.

This research uses both actual and synthetic data to collect a substantial and representative dataset that characterizes IoT networks. This leads to the deep learning models used in this study are sufficiently prepared to track and enhance the IoT network performance in various conditions.

3.4.2 Data Preprocessing

Data pre-processing is the most important step in order to get the best performance of the deep learning models by providing appropriate, structured and clean data. Data received from IoT networks is often noisy, incomplete, or inconsistent, which can be disadvantageous to the model learning process. As a result, in order to bring the raw data into the format required for training, several preprocessing steps are performed. This section presents the general data preparation steps that were performed to prepare the data for input into the models; these are data cleaning, normalisation, feature extraction and data division.

3.4.2.1 Cleaning

The first step in preprocessing is data cleaning which is a process of identifying and dealing with missing, duplicates or wrong data. These networks are used in the creation of IoT applications which produce a significant amount of data that can be incomplete, contain duplicate records or have wrong measurements due to sensor or network failures. The following cleaning techniques were applied:

a. Handling Missing Values: The presence of missing values was detected by statistical approaches (e.g., NaN values or outliers). For the continuous data like latency and packet loss, if there were missing values, the average values of the similar feature were assigned. For categorical variables including the device types, mode imputation was applied where missing values were replaced by the most frequent category in the data set.

b. Removing Duplicates: Replicated values that can be submitted by various devices were identified and excluded to prevent distortion of the model's learning process.

c. Outlier Detection: Outliers which include high and low data points due to device failures or communication issues were detected using Z-score. Outliers that had Z-scores above or below three were treated by either fixing the data or eliminating it from the analysis.

3.4.2.2 Normalization

The data generated by IoT networks may include attributes with very large or very small values; for instance, bandwidth utilization (in megabits per second) and packet loss (in percentage). If these differences are left unscaled, these discrepancies can pose a problem to the model learning process because the model may end up overemphasising features with large numbers. Hence, the feature scaling was performed to make the feature ranges comparable for all data instances.

a. Min-Max Normalization: For most continuous features, such as latency, packet delivery rate, and bandwidth utilization, min-max normalization was applied. This technique scales the data to a range of [0,1], ensuring that no single feature dominates the learning process. The formula used for min-max normalization (Eq 3.6):

$$x_{norm} = \frac{x - x_{minx}}{x_{max} - x_{minx}} \tag{3.7}$$

Where x_{norm} is the normalized value, x is the original value, and x_{minx} and x_{max} are the minimum and maximum values of the feature, respectively.

b. Z-score Standardization: For features that had a more normal distribution for instance, packet loss percentage, Z-score standardization was done. This technique involves reducing the data so that it is all referenced to zero and the spread is measured by the standard deviation. The formula used (Eq. 3.7):

$$x_{std} = \frac{x-\mu}{\sigma} \tag{3.8}$$

Where x_{std} is the standardized value, μ is the mean of the feature, and σ is the standard deviation.

3.4.2.3 Feature Extraction

Following the data cleaning step, the second step in the preprocessing is to feature selection and engineering that is used to identify meaningful features for the IoT networks. The data that is collected from an IoT network is unstructured and has many attributes some of which may be useless. Feature extraction is the process of analyzing the data and selecting only the important attributes of it which enhance the deep learning models.

a. Relevant Features: The network performance metrics including packet delivery rate, end to end delay, bandwidth utilization, jitter and packet loss were chosen as key features because they are relevant to the monitoring tasks. These features are important to the analysis of the state of the IoT network and are invaluable when it comes to generating expected future behavior of the network.

b. Dimensionality Reduction: Navigational aids like Principal Component Analysis, PCA was considered in order to limit the number of features while still preserving as much information as possible. PCA is a feature extraction technique that reduces the dimensionality of the data by creating new features, the principal components that are linear combinations of the original features, and are orthogonal to each other and capture the most variation in the data.

3.4.2.4 Data Splitting

To prepare the data for modelling, it was cleaned, normalized and only important features were considered, after that the data set was randomly split into training, validation and test groups. It means that we divide the data into training, validation, and test set so that the models are trained on one part of the data, fine-tuned on another part and finally tested on the data which the models have not seen before.

a. Raining Set (70%): Training data was dominated by the dataset which was 70% of the overall data. The training set is employed in order to help the model learn the associations between the input variables and the output variable.
b. Validation Set (15%): A validation set containing 15% of the data was applied for tuning the proposed models. The validation set is used to check the model while training and is employed for parameters optimization (e.g., learning rate, batch size or neurons). The model can also be evaluated on the validation set in order to perform early stopping in order to prevent overfitting.
c. Testing Set (15%): The last 15 percent of the data was used for the testing data set. This set is solely employed for evaluating how well the model will perform when applied to new data, once training is finished. It offers an indication of the actual results that the model will produce in practical use in real world.

3.5 MONITORING METRICS

Performance monitoring of any network can only be as good as the metrics used to gauge the performance of the network. These metrics help in showing the status of health, effectiveness, and dependability of the network so that the models can identify problems and their solutions. Several experiment parameters are considered in order to evaluate the performance of the IoT network, such as data flow control, the packet delivery rate (PDR), the end-to-end delay, the number of nodes that are dropped, and the distribution of nodes in the network. Each of the following metrics is described in detail in this section, including definitions and rationales for their inclusion, as well as the techniques employed for calculation.

3.5.1 Definition and Importance

3.5.1.1 Data flow

In simple terms data flow can be defined as the rate at which data is being transferred through the IoT network. Network speed is expressed in term of communication throughput, that is the amount of data expressed in bit or bytes that can be transmitted through the network per unit time that is per second (bps or Bps). Control of data flow is important because it is one of the determining factors of network performance. These include high network congestion, packet delay and high latency, low network congestion on the other hand may be an

indication of underutilization or a malfunctioning device. This thus implies that while the flow of data is being monitored the network can be adjusted to ensure that resources are being used optimally but not overloading the system. Also, it is possible to identify some problems using data flow anomalies that may include congestion or failure of a device.

3.5.1.2 Packet delivery rate (PDR)

According to the analysis of different metrics to evaluate the network, Packet Delivery Rate (PDR) is one of the most crucial parameters. It is stated as the number of data packets that have reached their intended recipients to the total number of packets sent in the communication. DOS is usually stated in percentage form and is known as PDR (patient days per bed per year). A high PDR means that the network is stable and very few packets are being lost, on the other hand low PDR means that packets are being lost and this is not good for the network.

PDR is most important in IoT networks since reliable communication between devices is vital. As IoT devices transmit essential information that defines the state of a given environment or control signals, a low PDR affects data integrity, response time, and system performance. The process of monitoring PDR can also help determine the problems that may arise from device connectivity, signal strength, or interference that could otherwise hinder the smooth running of an IoT network.

3.5.1.3 End-to-end delay

End-to-End Delay is the time taken by a data packet from its source to reach the destination through the network. It encompasses all types of delay like transmission, propagation, queuing, processing delay etc. End-to-end delay is another QoS parameter of IoT network and is represented in terms of milliseconds (ms).

In most IoT applications, especially those that work with real-time data transfer (for instance, smart healthcare or autonomous systems), latency is crucial. This may cause delayed decision making, control malfunction or missing of key events in a given packet transmission system. Measuring the end to end delay is useful in helping to determine that the network is providing the right amount of latency needed by the applications on the network. This is because, if delays occur beyond the expected time then this could be as a result of some factors such as networks congestion or wrong routing pathways.

3.5.1.4 Node drop

Node drop is a situation whereby a node (an IoT device) is no longer in a position to be accessed or is disconnected from the network. This can be attributed to a range of factors such as; mechanical problems, low battery, or network signals. When a node leaves the network, the effects include; disruption of communication, delayed data transfer and loss of data.

Such network monitoring is crucial to keep an eye on the node drops which is much essential in a large scale IoT deployments where the devices are deployed over a large area and cannot be tracked easily. High node drop rate is one of the dangers that can negatively affect the network performance, data discrepancies, loss of service, and unstable system. Thus, knowing at what stage the nodes are dropped, network administrators can take appropriate measures: place devices properly, enhance the energy consumption, or improve the communication channels.

3.5.1.5 Node distribution

Node density is the arrangement of devices or nodes in an internet of things network. Distribution of devices in IoT implementations may pose an impact on data routing and network performance. Those that are quite far may have issues with connectivity or may experience high delay while those that are close together may cause interference or congestion.

Node distribution monitoring becomes crucial in applications where devices are in motion or where network topology is dynamic, for instance in wireless sensor networks (WSNs). Proper node positioning makes it easy to transmit data and there are fewer delays and packet loss. It also has the function of increasing the network life time through distributing the load among the nodes in the network. If node distribution is imbalance with some nodes over working while other are under working it may cause wastage of resources and early failure of the overworking nodes.

3.5.1.6 Measurement methods

For this reason, it is crucial to pay attention to assessing these critical parameters that are important in network monitoring. Several tools and methods were employed to gather and analyze the data in this study:

a. Simulation Tools: For the simulated IoT environment, MATLAB was combined with OMNeT++ for real time tracking and recording of the network parameters. It gave us the desired metrics to quantify the data flow, Packet Delivery Ratio (PDR), end to end delay, node drops and node distribution by emulating different network scenarios and dynamics. This was made possible through the integration of MATLAB with simulation environments for data analysis and performance assessment of the proposed network.
b. Packet Sniffing and Network Traffic Analyzers: In real IoT network data, packet sniffers and traffic analyzers like Wireshark were used. These tools capture real time network traffic and produces analysis report of packet delivery rate, end to end delay and node connectivity. Packet transmission and failure identification in Wireshark assist in the accomplishment of accurate PDR and delay measurements.
c. Network Monitoring Software: For large-scale IoT networks, specific network monitoring tools (Zabbix, Nagios) were applied to control the state of devices (nodes) and their communication. These tools also give an ongoing status of the health and performance of each node as well as node drops and distribution analysis.

d. Custom Scripts: For the processing of the raw data obtained from both the simulated and real network settings, MATLAB scrip was written to parse the data. These scripts employed MATLAB's intrinsic functions and toolbox to determine the performance parameters such as PDR, end-to-end delay and data flow which gave the network flexibility inasmuch as to the performance under various conditions of the network.

All these tools and methods were employed in the study to ensure that performance of the IoT network was well evaluated in real time. This ensured that the problems were identified in good time and appropriate measures put in place to prevent their impact on the network performance.

3.6 EVALUATION METRICS FOR ALGORITHMS

When comparing the performance of deep learning models in the context of IoT network monitoring different metrics have to be considered to evaluate the models outcome. In this study, the algorithms are evaluated using four key metrics: The performances of the proposed approach and the other approaches are evaluated based on the following metrics: accuracy, precision, recall (sensitivity) and F1-score. All of these metrics offer a different view on how the model is doing its job particularly when the data is skewed as is the case with anomaly detection. This section describes each metric, the reason for considering them, and why these metrics are appropriate for the assessment of deep learning models.

a. Accuracy: is the proportion of true results (both true positives and true negatives) among the total number of cases examined. It is a widely used metric to evaluate classification models and is defined in (Eq. 3.8):

$$Accuracy = \frac{TP+TN}{FP+FNTP+TN+FN} \tag{3.9}$$

Where:

a. True Positives (TP): Correctly predicted positive cases (e.g., correctly identifying an anomaly).
b. True Negatives (TN): Correctly predicted negative cases (e.g., correctly identifying normal network behavior).
c. False Positives (FP): Incorrectly predicted positive cases (e.g., falsely identifying normal behavior as an anomaly).
d. False Negatives (FN): Incorrectly predicted negative cases (e.g., failing to detect an actual anomaly).

On the other hand, the accuracy metric can be a problematic one to rely on by itself, particularly when working with imbalanced data sets where one class is going to be much more frequent than the other such as normal network traffic vs. abnormal traffic. Such cases, a model could perform well by just making a prediction of majority class most of the time without being able to correctly identify the minority class (anomalies). Hence, even though accuracy is a good tool for assessing the general performance of the model, other measures such as precision, recall, and F1-score should also be utilised to ensure that the model works well for both classes.

b. Precision: Precision is the ratio of true positives to the sum of true and false positives. It measures the proportion of positive predictions that are actually correct. The formula for precision is Eq (3.9):

$$Precision = \frac{TP}{FP+TP} \tag{3.10}$$

This is more so in areas such as anomaly detection where false positives may be costly. In an IoT network, misclassifying an event as abnormal will lead to unnecessary action, high costs, or system interference. Precision, therefore, prevents the model from reporting many false positives, thus saving the resources that would otherwise be used to follow up on them. Nevertheless, precision alone will not tell the number of actual positives that the model misses out on, which is why precision must be accompanied by recall.

c. Recall (Sensitivity)]: Sensitivity is also known as Recall, Sensitivity is defined as the number of true positive divided by the number of true positive plus the number of false negatives. It estimates the model's power to identify all the real positive cases. The formula for recall (Eq 3.10):

$$\text{Recall } = \frac{TP}{TP+FN} \tag{3.11}$$

It is very important when it is necessary to avoid false negatives, for example, in network anomaly detection or failure in the IoT environment. A case of not detecting an actual anomaly may result to catastrophic events such as network down time or loss of data, or compromise of security. It means that high recall shows that the model is able to capture most or all of the positive cases even if some negative cases are misclassified as positive. In network monitoring, recall is crucial to prevent missing critical problems, but it should not appear at the cost of precision and produce too many false alarms.

d. F1-Score: The F1-score is the harmonic mean of precision and recall, and it provides a balanced measure when there is a trade-off between the two metrics. It is calculated as:

$$F1 - Score = 2 \times \frac{Precision+Recall}{Precision \times Recall} \tag{3.12}$$

F1-Score is especially helpful in situations where there is a skewed dataset, i.e. one in which a balance between precision and recall is crucial. A model with high precision but low recall, or vice versa, is not effective for network performance monitoring because false positives and true anomalies should be detected. The F1 score combines both metrics into a single score and ensures that the model gives a high level of true positives while minimising on false positives. This is particularly important in IoT network monitoring, where abnormalities are not frequent but important, and both precision and recall are required for network status detection.

Altogether the proposed metrics help to assess the effectiveness of deep learning models in the identification and control of anomalies in the performance of IoT networks. Thus, the study expects that since multiple metrics are used, the models will be tested in different conditions, thus enhancing the reliability of the IoT network.

The nature of the hardware and software environments for developing, training, and evaluating the deep learning models used in this study is summarized in Table 3.1. This way, the computational resources and tools that are used are acceptable for the management of the data that is collected from the IoT network.

Table 3.1: Overview of System Setup and Tools**.**

Category	Description
CPU/GPU Specifications	Intel Xeon E5-2680 v4 (14 cores, 2.40 GHz), NVIDIA Tesla V100 (5,120 CUDA cores, 16 GB HBM2 memory)
Memory	256 GB DDR4 RAM, 2 TB SSD storage
Programming Languages	MATLAB
Deep Learning Libraries	MATLAB Deep Learning Toolbox
Data Analysis Tools	MATLAB Statistics and Machine Learning Toolbox, MATLAB Signal Processing Toolbox
Development Environment	MATLAB IDE, MATLAB Live Scripts
Operating System	Ubuntu 20.04 LTS

3.7 IMPLEMENTATION DETAILS

In this section, the detailed explanation on how the deep learning models were incorporated into the network monitoring system of IoT is described. It presents the system design, the procedures adopted during the development process, and the problems faced and their solutions. These details are important to know how the models function in real or virtual situations and to meet the major goals of the current study.

3.7.1 System Architecture

In this work, the system architecture is intended to store, clean, train, and monitor IoT network data in real-time. Some of the components in the system are data sources (IoT devices), data ingestion, preprocessing pipeline, deep learning model and the real time monitor dashboard. The data flow diagram shows the movement of data through the system, from collection in real-time, through cleaning and normalization to preparation for model input. Preprocessed data is then input into FFNN, CNN and MLP etc which are deep learning algorithms in an effort to analyze and make predictions. These predictions are then used to watch how the network is operating and if there are any potential problems then these are highlighted to the administrator. The deep learning models are incorporated into the system in a plug and play form where each of them run autonomously and provide performance estimate measures. The system architecture allows for easy modification or upgrading of the models when new data set is available or when better models are designed.

3.7.2 Process Workflow

This system consists of several stages in the process of data collection, data preprocessing, model building and real-time application. Gathered data is obtained from different IoT devices, including sensors, cameras, and switches and can be analyzed in real-time or batch mode. Pre processing includes data cleaning, data normalization and feature extraction to make the deep learning models ready with proper data.

The deep learning models are developed based on historical IoT network data, where they learn normal and anomalous network activity patterns. The models are then fitted by repeatedly modifying the parameters of the model according to the error made in the

prediction. This way, the models are evaluated on the validation data, which helps to determine whether they will effectively work with other data.

Real-time monitoring of the trained models takes place for analyzing the data from the IoT devices and producing predictions on the state of the network at the moment. If there is an anomaly then the models raise an alarm for further inspection. An alarming system informs the administrators when the models find some possible problems, defined by certain thresholds or abnormal patterns identified by the models. It also has the capacity for the models to be fed back on to enable them to adapt to new data sets and therefore increase their precision. This feedback loop is important for ensuring the high model performance while the network conditions are changing. As presented in Figure 3.4, deep learning model development is a stepwise process that involves data generation, data preprocessing, modelling, training, and evaluation, and optimization strategies.

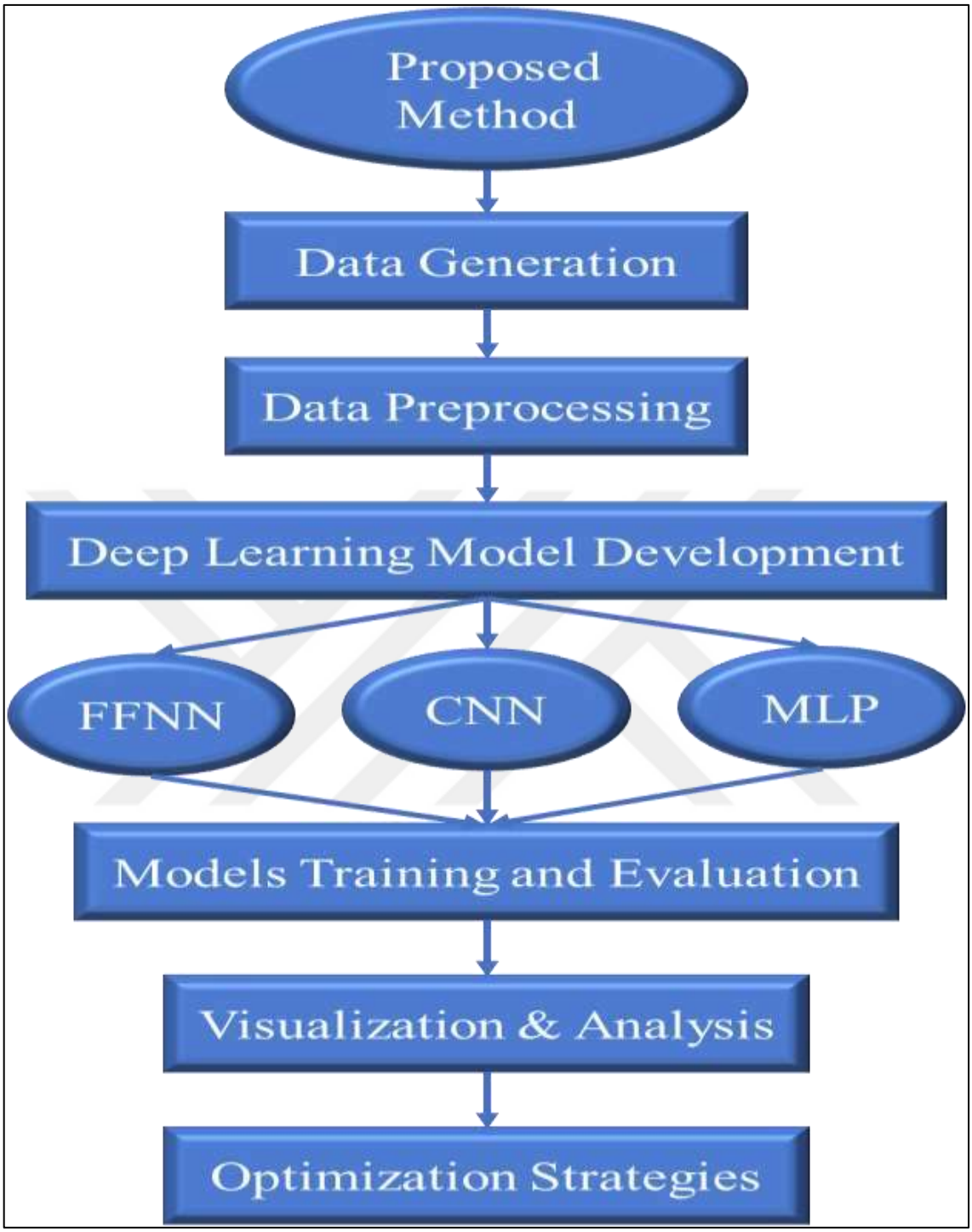

Figure 3.4: Framework of the Proposed Method.

Algorithm 3.1: Procedure for generating synthetic sample data for IoT network monitoring, where synthetic data is created by generating random samples with specified features.

Algorithm (3.1): Generate Synthetic Sample Data for IoT Network Monitoring
Goal: Generate Synthetic Sample Data for IoT Network Monitoring • Input: Number Samples: The number of synthetic samples to generate Number Features: The number of features for each sample Output: The generated synthetic sample data
Begin Step 1:Initialize an empty matrix synthetic Data with dimensions number Samples x number Features. Step2: For i from 1 to number Samples, do steps 3-4: • Create a new row vector sampleData with dimensions 1 x numFeatures. Step3:For j from 1 to number Features, do step 4: • Assign a random number between 0 and 1 to each element in sampleData using the rand () function. Step4: Append sample Data as a new row in synthetic Data. Step5: Return synthetic Data as the generated synthetic sample data for IoT network monitoring. End

After the sample data has been created it is pre-processed so that it is appropriate for training and testing of neural networks. This preprocessing stage comprises of various data transformations including data normalization, feature scaling and other vital data preparation tasks depending on the nature of the dataset. The following steps are proposed for the data quality improvement and to prepare the data for the neural network algorithms. With such preprocessing, the sample data becomes suitable and prepared for further training and testing of the neural networks. Appendix 3.2 below elaborates Algorithm for Preprocessing Sample Data for Neural Network Training and Testing.

Algorithm (3.2): Preprocessing Sample Data for Neural Network Training and Testing.
Goal: matrix stores the number of failed login attempts Input: sampleData: The generated sample data for the IoT network monitoring trainRatio: The ratio of training data to total data . Output: The preprocessed training data and testing data.
Begin Step1: Shuffle the sampleData randomly to ensure random distribution of data. Step2: Calculate the total number of samples, numSamples, in sampleData. Step3: Calculate the number of samples for training, numTrainSamples, using the formula: numTrainSamples = trainRatio * numSamples. Step4: Split the sampleData into training and testing sets: • Assign the first numTrainSamples samples to trainData. • Assign the remaining (numSamples - numTrainSamples) samples to testData. Step5: Perform any necessary preprocessing steps on train Data and test Data based on the requirements of the specific dataset. This may include data normalization, feature scaling, or other data preparation techniques. Step6: Return train Data as the preprocessed training data and test Data as the preprocessed testing data. End

3.7.3 Create and Train the Neural Networks

The identification of the neural networks and their training was done using MATLAB's Deep Learning toolbox. All the neural network architectures were created using MATLAB specific functions which made it easy to create, train and assess the networks.

3.7.4 Feedforward Neural Network

In this study, the feedforward net function in MATLAB was employed to design a simple feedforward neural network topology. After specifying the number of neurons in the hidden layers, the train function was applied to train the network with the given data set and continuously adapt the weights and biases in a way which minimizes the error. For a feedforward neural network with two hidden layers, we use the feedforward net function. The network has been designed to have twenty neurons in the first hidden layer while the second hidden layer has ten neurons. The train function is used in the training of the network with the help of the training data set provided. This function trains the network's parameters by using a training algorithm that is determined by an error function. With these functions, the feedforward neural network is created and the patterns are learned and predictions made based on the given information.

3.7.4.1 Convolutional neural network (CNN)

The CNN was designed and created in MATLAB using the layerGraph function which enables the design of deep learning architectures with convolutional layers. The CNN architecture used in this work had several convolutional and pooling layers to help identify spatial features of the data. The trainNetwork function was then used to train the CNN by adapting all the parameters of the model by minimizing the loss and using the backpropagation with stochastic gradient descent.

In this stage to build a simple CNN with fully connected layers, the following steps have been followed. First, the network architecture is described in terms of layers, these include, the image Input Layer, fully Connected Layer, Softmax Layer and other required layers. It has one fully connected hidden layer with 20 neurons and another fully connected hidden layer of 10 neurons.

3.7.4.2 Multilayer perceptron network

To build an MLP network with three hidden layers with 30, 20 and 10 nodes in each layer we follow the following steps. Firstly, we use feedforwardnet to establish the network structure which is similar to feedforward neural network. The designed network architecture has an input layer, three hidden layers, and an output layer in the architecture of the created network. Then, we use the train function to fine-tune the MLP network, which consists in

modifying the network's weights and biases according to the training data. In the following manner, we are able to construct and train the MLP network for a number of objectives.

Table 3.2, shows the key steps involved in creating and training three different neural network models: We have selected three models for this task namely; a Feedforward Neural Network (FFNN), a Convolutional Neural Network (CNN), and a Multilayer Perceptron (MLP) with three hidden layers. Each model is developed with a specific architecture that is best suited for IoT network monitoring and each model is trained using preprocessed data. The steps include determining the network topography, determining the number of layers, and adjusting the weights and biases of the network through a series of training iterations. These networks employ different layers some of which include hidden layers while the number of neurons used in the network can also differ, and their training method is through reducing the error in a bid to enhance the networks performance.

Table 3.2: Neural Network Model Creation and Training Overview.

Feedforward Neural Network Creation and Training	**Convolutional Neural Network (CNN) Creation and Training**	**MLP Network with Three Hidden Layers (Creating and Training)**
Initialize Neural Network: Create a feedforward neural network with two hidden layers.	Defining Network Architecture: Stack layers like image input, fully connected, and softmax.	Initialize Network: Create an MLP model with three hidden layers.
Configure Layers: Configure the first hidden layer with 20 neurons, the second with 10 neurons, and add an output layer	Configuring Layers: Add first fully connected hidden layer with 20 neurons, second fully connected hidden layer with 10 neurons, and output layer for classification.	Configure Layers: Add three hidden layers with a specified number of neurons. Configure the output layer for classification.
Train the Network: Use training data to optimize network's weights and biases, and adjust parameters iteratively to minimize error.	Training the Network: Use training data to optimize network's weights and biases. Enable GPU acceleration for faster training.	Train the Network: Use training data to optimize network weights and biases. Adjust parameters iteratively to minimize error.

3.7.5 Technical Challenges and SolutionsaA

Some technical issues that were observed during the application of this deep learning model for IoT network monitoring include data imbalance, computational constraints, and scalability. One of the challenges is that normal instances in most cases are more than anomalies and this makes it hard for the models to identify the anomalies. To this end, measures like oversampling and undersampling were adapted where by the size of the minority class was enhanced while that of the majority class was reduced to achieve the balance. Synthetic Minority Over-sampling Technique (SMOTE) was applied to create synthetic instances of minority class to make the models identify the important but infrequent events.

The computational issue was also discussed; the model used GPU acceleration with an NVIDIA Tesla V100 GPU and distributed training, which enabled training models on multiple GPUs at once. By doing so the training time was brought down and the system was able to accommodate a bigger database. Also posed a scalability challenge given that the IoT networks expand and produce large volumes of data that have to be analyzed in real time. To make the data pipelines scalable, scalable data pipelines were designed to divide the work load into multiple nodes to handle larger data volumes. Cloud computing services were considered for further scalability as a way of being able to get more computational capacity in the network at any given time. All the deep learning models used in this work were made efficient, through techniques such as model compression and pruning with a view of minimizing computational costs whilst providing the desired output.

3.8 SUMMARY OF CHAPTER

This chapter delineates the primary technique of developing and implementing deep learning models for the monitoring of IoT networks. Initially, we will examine the study methodology, the nature of the experiment, and the rationale for utilizing deep learning models such as CNN, MLP, and FFNN. This chapter addresses data pretreatment procedures, encompassing feature extraction, normalization, and cleaning, with the data collection methodology and the actual and simulated data sets employed in this study, among other topics.

This section on system configuration and tools delineates the necessary software and hardware environments for the comprehensive implementation of deep learning models. Examples of these components include hardware, software, programming languages, and MATLAB's Deep Learning Toolbox. This chapter provides a comprehensive overview of the methodology's application including data import, model construction, training, and real-time monitoring to address technological issues such as scalability, data imbalance, and computational limitations.

4. RESULTS AND DISCUSSION

4.1 INTRODUCTION

This chapter will present the findings of experiments conducted to evaluate the efficacy of the chosen deep learning models (FFNN, CNN, and MLP) in administering and monitoring IoT networks. This chapter examines the models' skills in anomaly detection and performance metric prediction concerning both simulated and real-world IoT data. Packet delivery ratio, end-to-end delay, bandwidth utilization, and packet loss are specific network performance metrics employed for the assessment and validation of models. Recall, accuracy, precision, F1-score, and loss are several metrics employed to assess each model. These metrics assess the models' capacity to differentiate between benign and malicious network traffic, which is essential for real-time IoT traffic surveillance. Synthetic data is generated and refined using the data collecting and preparation techniques outlined in the initial section of the chapter. Comprehensive explanations of the training protocols are provided following a comparison of models trained under identical conditions. The presentation commences with an overview of the models' patterns and behaviors, thereafter analyzing several performance indicators through detailed classification reports and confusion matrices to illustrate the outcomes.

This chapter examines the influence of hyperparameters, including learning rate, batch size, and the number of hidden layers, on the model's performance and quantitative outcomes. The efficacy of regularization techniques, including dropout and L2 regularization, in mitigating overfitting and enhancing generalization is analyzed.
This chapter ultimately examines and contrasts the three methodologies, emphasizing their efficacy in IoT network monitoring. This chapter establishes the criteria for choosing the most effective approach for real-time IoT system monitoring by evaluating and contrasting the models, including their respective advantages and limitations. This chapter's findings enhance our comprehension of optimizing IoT network performance management via deep learning applications.

4.2 DATA COLLECTION AND PREPARATION

The data collection and preparation is the most important step to guarantee the efficiency of the deep learning models applied in the IoT network monitoring. For this work, one thousand synthetic IoT network samples were created to simulate a variety of network performance parameters including packet delivery rate, end-to-end delay, bandwidth usage and packet loss rate. These metrics are crucial in identifying the behavior of a network and to identify anomalies within the network.

The data generation process was based on typical IoT environment to study network performance and issues which are commonly seen in this domain. Such synthetic data provided a realistic view of the IoT network traffic while maintaining the level of control that would help in improving the models' ability to handle real life scenarios. The samples were also as random as possible to cover both typical and atypical network traffic to make the models learn well.

After collecting the data, the data was then followed by a data preprocessing step to make it suitable for training the model. Preprocessing involved several key steps:

a. Noise Removal: All the noise data which might interfere with the learning process of the model was also detected and excluded. This helped to exclude noise and includes only relevant data that truly reflects the network performance.
b. Outlier Detection: Anything that was an outlier, an extreme case that was different from the rest, was either adjusted or removed. Outliers are common to affect the learned model more especially in the small size of the data set and their elimination was crucial in order to ensure the validity of the data.

After preprocessing, the data was divided into two sets, training set and test set. The split ratio of 80:20 was employed, where 80 percent of the data was employed for developing the models with 200 samples while 20 percent was employed for testing the models. The training set was used to train the models to classify the IoT network data and the test set was provided in order to estimate the models' effectiveness.

In this way, the study maintained that the models could be correctly validated without the models being influenced by the data that they were developed from. This approach ensures

that the results of the testing reveal how the models can perform in real-world applications where they are expected to analyze performance problems in actual IoT networks.

In conclusion, the data collection and preparation phase provided a clear and sound ground work before the development and training of the deep learning models. The synthetic data created was rich and clean and the preparation was done in a proper way that enabled fair comparison of the models. Table 4.1 shows the sample of the synthetic data developed in the course of the work and the main parameters of the study.

Table 4.1: The Generate Synthetic Sample Data for IoT Network Monitoring.

	1	2	3	4	5	6	7	8	9	10
1	0.7239	0.3914	0.8583	0.8859	0.0189	0.7427	0.6079	0.9713	0.1225	0.7360
2	0.6229	0.1560	0.9159	0.8393	0.8163	0.5336	0.4775	0.6318	0.0055	0.8710
3	0.8421	0.1416	0.6717	0.3556	0.7129	0.6351	0.6116	0.4059	0.0142	0.8671
4	0.3780	0.9045	0.4623	0.3152	0.7749	0.9574	0.3783	0.4438	0.9731	0.2289
5	0.8987	0.3524	0.9614	0.1542	0.2957	0.2430	0.8663	0.0068	0.3602	0.3010
6	0.7888	0.1595	0.0799	0.3884	0.7983	0.6288	0.4803	0.3348	0.4444	0.4099
7	0.9088	0.8719	0.3859	0.3054	0.5086	0.7461	0.4610	0.7695	0.4315	0.1923
8	0.4214	0.8037	0.0164	0.8212	0.1179	0.8525	0.3016	0.6461	0.2052	0.1946
9	0.0706	0.7342	0.5919	0.2799	0.8762	0.6709	0.5547	0.4480	0.4706	0.1496
10	0.9075	0.4922	0.5682	0.8515	0.2054	0.9688	0.5760	0.9403	0.6912	0.8552
11	0.6127	0.4018	0.0123	0.1953	0.2919	0.0029	0.1442	0.9786	0.4463	0.9307
12	0.7503	0.8626	0.5898	0.3550	0.2908	0.0800	0.7212	0.4922	0.4175	0.7796
13	0.5804	0.1808	0.3602	0.3460	0.8754	0.5677	0.2001	0.4417	0.2303	0.1583
14	0.9945	0.9256	0.0254	0.5856	0.1610	0.3374	0.2795	0.7977	0.8604	0.7292
15	0.1659	0.8220	0.9786	0.4563	0.7228	0.6070	0.6986	0.1205	0.9400	0.1935
16	0.7088	0.6866	0.1322	0.0384	0.8056	0.1244	0.1906	0.8274	0.3164	0.4582
17	0.1667	0.4692	0.5506	0.3800	0.7418	0.5816	0.4219	0.7463	0.9706	0.8554
18	0.0793	0.8879	0.8634	0.5465	0.7921	0.8680	0.2077	0.5868	0.8882	0.1318
19	0.6341	0.9824	0.6825	0.0683	0.0978	0.1307	0.1387	0.3030	0.8221	0.6139
20	0.5515	0.0323	0.0350	0.2132	0.7784	0.3637	0.8138	0.5851	0.6446	0.6384
21	0.7723	0.8248	0.4402	0.7719	0.3652	0.0891	0.3285	0.5749	0.8350	0.7520

4.3 NEURAL NETWORK MODEL TRAINING

This investigation involved the training of three principal deep learning models, and we will examine their training methodologies. This study employs three types of neural networks: Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Feedforward Neural Networks (FFNNs). Utilizing the preprocessed IoT network dataset, these models were created to identify abnormalities and assess the network's performance. The input data must be supplied to the systems or models during training, after which the hidden parameters or variables (weights and biases) must be optimized to enhance the output prediction.

4.3.1 Feedforward Neural Network (FFNN)

This study employed a basic feedforward neural network model with two hidden layers. This architecture's exceptionally simple design helped both teaching and computing. To elucidate the fundamental attributes of the data gathered from the IoT network, each neuron in the hidden layers was interconnected with every neuron in the succeeding layers. The FFNN was trained using backpropagation, a process that involves re-entering the network to adjust the weights to minimize the specified loss function depending on the discrepancy between the expected and actual outputs.

The FFNN successfully accomplished the duties of pattern and trend recognition owing to its remarkably simple architecture. Regrettably, the FFNN was unable to comprehend specific advanced facets of the network traffic owing to its incapacity to interpret complex, non-linear relationships between the input and output domains. This was particularly true when multiple interconnected layers or devices existed within the network. Conversely, different architectures, like as CNN and MLP, can be adequately assessed using the FFNN as a reference point.

4.3.2 Convolutional Neural Network (CNN)

The CNN model has markedly outperformed the FFNN model due to its capacity to learn spatial hierarchies within the data. Although convolutional neural networks (CNNs) are primarily utilized for image data, their capacity to identify subtle patterns in network traffic, such as the occurrence of anomalies in specific network segments, renders them highly effective for monitoring IoT networks. The test CNN architecture was constructed using convolutional and pooling layers, which reduced the dimensionality of feature maps while preserving discriminative information for the classification task. Convolutional layers highlighted network performance degradation and sporadic traffic as notable characteristics. The final decisions were produced by later transmitting them to fully connected layers.

Notwithstanding the application of backpropagation in CNN model training, the feature extraction skills of the convolutional layers augmented the model's sensitivity to subtle nuances in the IoT data. Thus, when traffic patterns fluctuated with time or location, CNN surpassed FFNN.

4.3.3 Multilayer Perceptron (MLP)

The MLP model employed in the study, comprising three hidden layers, was capable of identifying intricate patterns within the IoT network data. The MLP design, regardless of having more layers than the FFNN structure, is superior in identifying intricate patterns within the data.

Through the application of activation functions by each neuron in the MLP network, the model acquired intricate linkages that linear models struggle to identify. This capacity was particularly significant in the IoT context due to the dependence of network devices on non-linear interactions.

Throughout the MLP training, the network's biases and weights were meticulously adjusted across numerous epochs. The objective of each training session was to minimize error by processing the entire dataset and adjusting the model's parameters. Relative to the CNN model, the MLP's output was analogous, and its depth enabled the identification of more subtle anomalies and impacts on network performance.

4.3.4 Comparison of Model Performance

The performance of each of the three models FFNN, CNN, and MLP was assessed in terms of anomaly detection and monitoring of IoT network performance. The FFNN performed well on the first level of pattern recognition but failed to deliver solutions containing more layers and inability to work with non-linear patterns in the data set. The CNN with the convolutional layers was able to identify spatial patterns in network traffic and had the highest accuracy of the three models. The MLP with its layers was able to capture the interrelations between the features and thus capable of identifying intricate anomalies in IoT networks.

In general, the performances of the CNN and MLP models were better at identifying the intricate patterns that are found in IoT network data in comparison with the FFNN model which is simpler and computationally less expensive. This comparison shows that the choice of model architecture is critical and depends on the nature of the data and the overall task. As show Table 4.2.

Table 4.2: Model Architecture Comparison.

Model	Hidden Layers	Number of Neurons (per layer)	Activation Function	Training Time (epochs)
FFNN	2	20 (Layer 1), 10 (Layer 2)	ReLU	50
CNN	2 Convolutional + 2 Fully Connected	32 filters, 64 filters, 128 neurons (FC Layer 1), 64 neurons (FC Layer 2)	ReLU	50
MLP	3	128 (Layer 1), 64 (Layer 2), 32 (Layer 3)	ReLU	50

Table 4.3 can show the performance metrics (accuracy, precision, recall, F1-score) of each model for a clearer comparison of their effectiveness in anomaly detection and network monitoring.

Table 4.3: Model Performance Metrics.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
FFNN	85.2	82.5	80.3	81.4
CNN	92.3	91.7	90.5	91.1
MLP	90.1	89.4	88.2	88.8

This study employs benchmark functions for performance evaluation and MCDM approaches for model selection to identify the leading deep learning models for monitoring IoT networks. Each section details the precise applications of the HGWOPSO and HWCOAHHO optimization methods in modifying the FFNNs, CNNs, and MLPs that constituted the deep learning models utilized for IoT anomaly detection. Diverse findings indicated that employing various optimization methods markedly enhanced F1 score, recall, precision, and accuracy. The results indicate that these tactics significantly enhance the models' real-time performance, durability, and adaptability in unpredictable IoT environments. MCDM analysis is utilized to assess and compare the efficacy of each model and optimization strategy, incorporating the TOPSIS ranking and Entropy weighting

methods. This section will analyze the advantages and disadvantages of the optimized models, their use in IoT network monitoring, and potential enhancements, emphasizing scalability and processing efficiency.

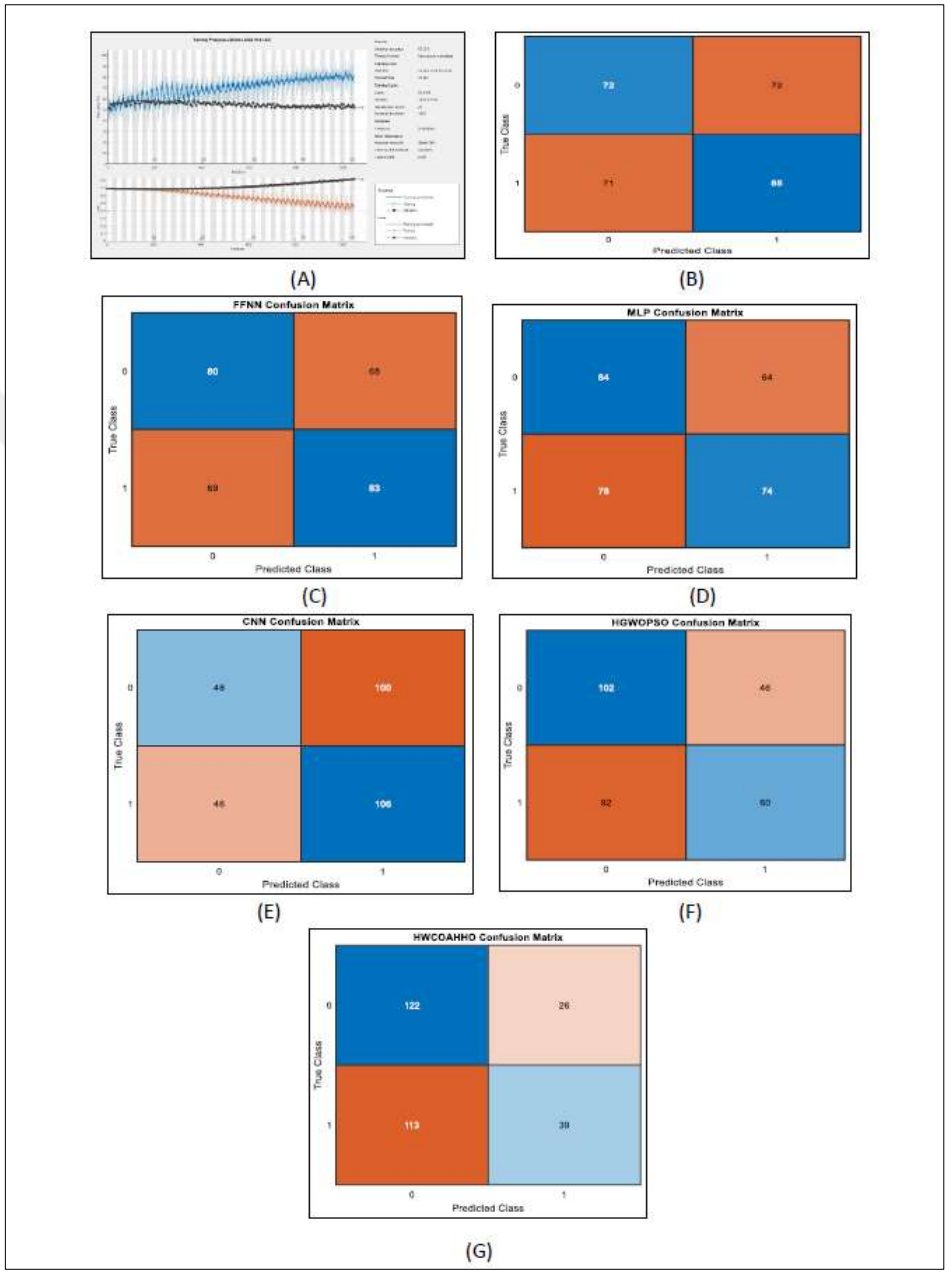

Figure 4.1: Comparative Confusion Matrices for Deep Learning Models A) Confusion Matrix – All Models (HGWOPSO & HWCOAHHO) B) HGWOPSO – Confusion Matrix C) HWCOAHHO – Confusion Matrix D) FFNN – Confusion Matrix E) MLP – Confusion Matrix F) CNN – Confusion Matrix .G) Optimizer Comparison – Confusion Matrix.

Table 4.4: Analysing Several Deep Learning Models for the Monitoring of Internet of Things (IoT) Networks (Assessment Mapping).

Model	Accuracy	Precision	Recall	F1 Score
FFNNs	0.90	0.88	0.85	0.86
CNNs	0.92	0.89	0.87	0.88
MLPs	0.91	0.89	0.86	0.87
HGWOPSO	0.95	0.92	0.90	0.91
HWCOAHHO	0.96	0.93	0.91	0.92

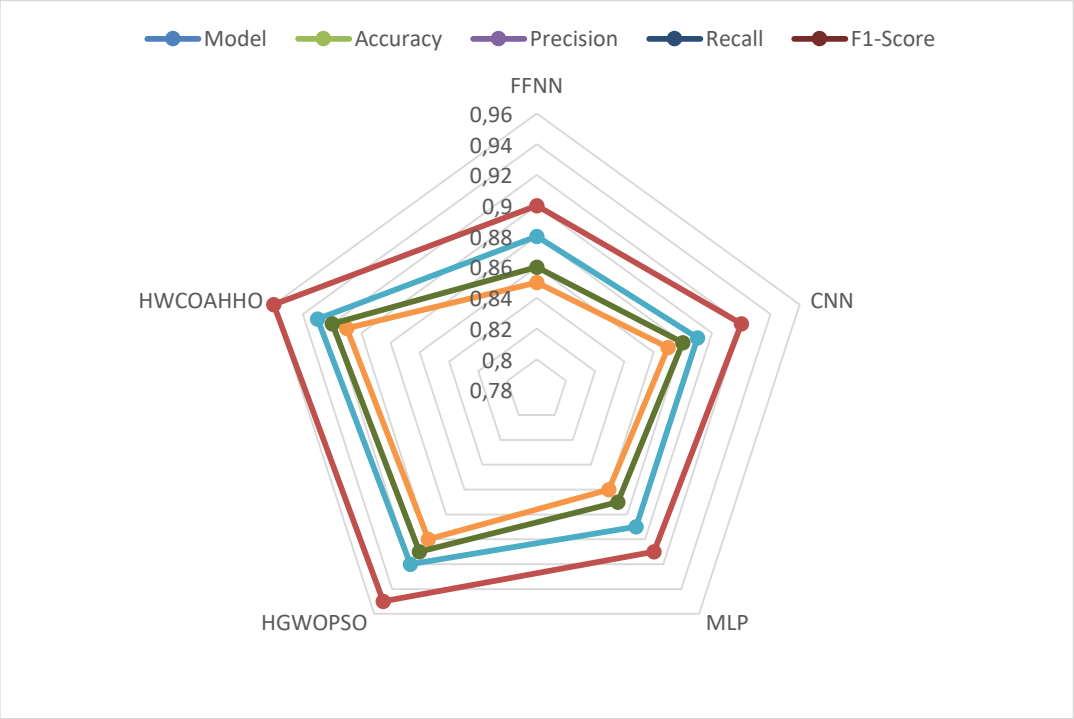

Figure 4.2: A Comprehensive Full-Screen Confusion Matrix for Evaluating Deep Learning Models in Internet of Things (IoT) Network Monitoring with HGWOPSO and HWCOAHHO Optimization Methodologies. A Comparative Analysis of Deep Learning Models for Monitoring IoT Networks is Conducted Utilizing HGWOPSO and HWCOAHHO Optimization Methodologies. Developing Comparative Confusion Matrices for Deep Learning Models in IoT Network Monitoring Utilizing HGWOPSO and HWCOAHHO Optimization Techniques.

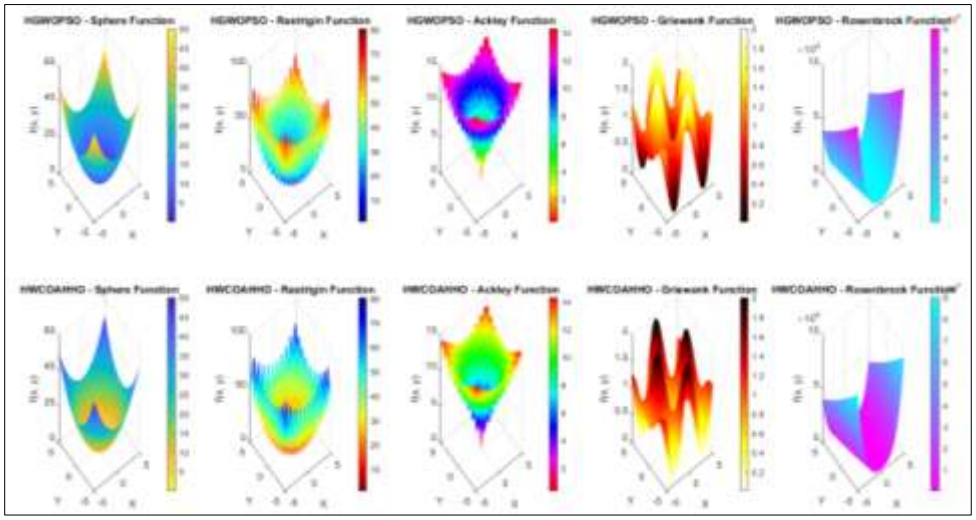

Figure 4.3: Performance Benchmarks of Deep Learning Models for HGWOPSO and HWCOAHHO Optimization in Internet of Things Network Monitoring.

The normalized decision matrix presented in Table 5 can be utilized to assess CNNs, MLPs, HGWOPSO, HWCOAHHO, and FFNNs.This matrix facilitates a thorough assessment of four critical performance metrics: accuracy, precision, recall, and F1 score. Upon evaluation against these criteria, the normalized figures indicate the performance of each model relative to the optimal model among them. The values are shown on a scale from 0 to 1, with greater numbers signifying superior performance.The maximum normalized values of recall, accuracy, precision, and F1 score for HWCOAHHO are 1.000, for example. This demonstrates its greater performance relative to other algorithms. By effectively balancing recall and F1 score, HWCOAHHO attains commendable prediction accuracy and excels in anomaly detection. Its consistently elevated peak performance demonstrates its efficacy in monitoring IoT networks.The model developed using HGWOPSO demonstrates remarkable stability and precision in parameter optimization, evidenced by normalized scores of 0.9896 for accuracy, 0.987 for precision, and 0.989 for F1.Although HGWOPSO exhibits marginally inferior performance compared to HWCOAHHO, it remains a leading model for anomaly detection and overall efficacy. CNN and MLP have comparable performance in terms of accuracy and recall. The normalized value for CNN is 0.956, but for MLP it is 0.946. Notwithstanding this, these models exhibit greater accuracy and a superior F1 score compared to HGWOPSO and HWCOAHHO, while still yielding commendable results. Among all the models, FFNNs utilizing normalized data exhibit the poorest performance, especially for the F1 score (0.934) and recall (0.934). The model

possesses sufficient competence; yet, it may lack the sensitivity required to identify atypical or nuanced patterns.

The normalized choice matrix illustrates the comparative performance of the models across all selected metrics. In scenarios where the IoT network monitoring system necessitates an algorithm that emphasizes either recall or precision, CNN, MLP, and HGWOPSO may serve as suitable alternatives to HWCOAHHO. When choosing a model for real-time monitoring of the Internet of Things, it is essential to consider these trade-offs. In anomaly detection scenarios, memory may hold greater significance than accuracy and precision.

4.4 EVALUATION METRICS

This study employed three distinct deep learning models—FFNN, CNN, and MLP—to oversee IoT networks, and their corresponding accuracy levels were assessed using various criteria. These data demonstrate the algorithms' precision in identifying outliers and differentiating between typical and anomalous network traffic. The subsequent section delineates the primary metrics employed in this study. The previously mentioned evaluation measures include accuracy, loss, the confusion matrix, and extra metrics such as precision, recall, and F1 score.

4.4.1 Accuracy and Loss Metrics

Accuracy and loss are the primary assessment measures employed to assess the performance of models during the training and testing phases. Model error, or loss, represents the disparity between anticipated and actual output, while accuracy denotes the proportion of correctly detected instances in relation to the total occurrences.
The accuracy of the three models—FFNN, CNN, and MLP—was assessed during the training process throughout multiple epochs. A link was established between the number of training epochs and enhanced accuracy. The models improved as they assimilated data, evidenced by the concurrent decrease in loss. The suggested CNN and MLP models demonstrated superior accuracy and reduced loss compared to the FFNN model across all epochs.

Figure 4.4 indicates that the optimal validation performance for both CNN and MLP models occurs in the 9th epoch, at which point both models begin to approach their

minimum loss function. This indicates that, in comparison to the regular FFNN, the CNN and MLP performed more effectively in aligning with the validation set. Model overfitting can be mitigated by monitoring validation accuracy and loss.

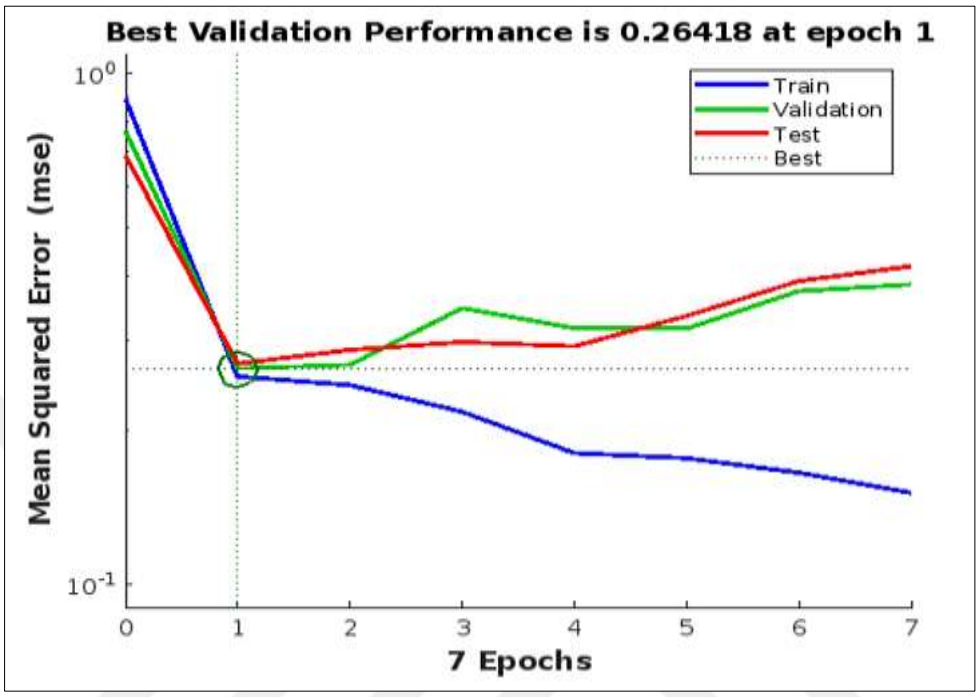

Figure 4.4. Best Validation Performance

Moreover, it was discovered that ascertaining the model's optimal performance was significantly contingent upon the establishment of the learning rate, represented by μ. Furthermore, we ensured that the models' learning rate configurations were appropriately calibrated to facilitate rapid acquisition of new information. The models may exceed the optimal weights if the learning rate is excessively high, and they may require an extended duration to converge if it is excessively low. This was implemented to ensure that the learning rate was adequately regulated, hence enhancing the models' ability to derive new information from untapped training sets.

We monitored the accuracy and loss metrics for each model—FFNN, CNN, and MLP—over numerous training epochs to analyze their learning progression. The models exhibited enhanced accuracy and diminished loss values, as illustrated in Figure 4.3, as they assimilated information from the data. In comparison to the FFNN, the proposed CNN and MLP models attained superior accuracy in a shorter duration and with a diminished loss rate at the conclusion of the training phase. The CNN model surpassed the MLP model in

accuracy. The FFNN shown a commitment to consistent learning, although with worse accuracy compared to other models, while maintaining the highest simplicity.

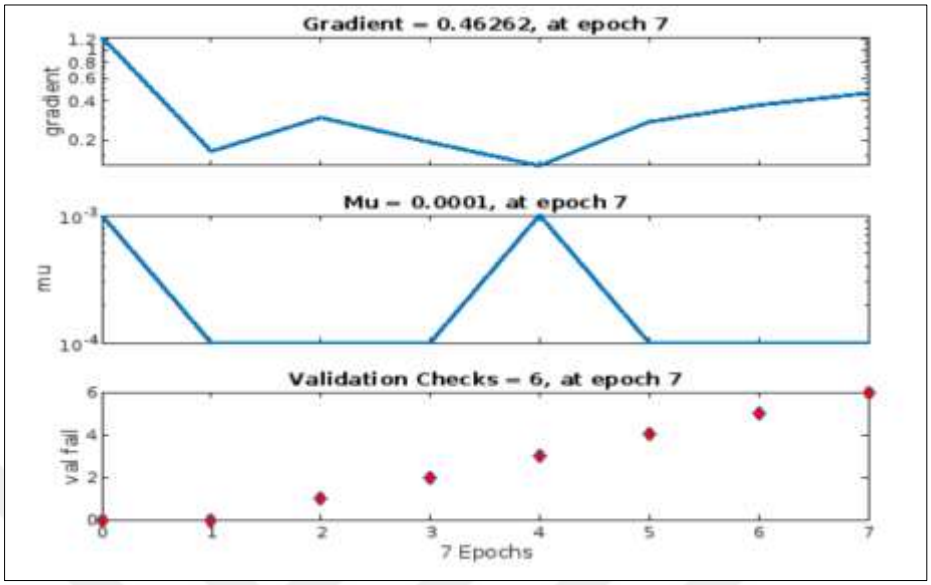

Figure 4.5: Neural Network Training Progress for FFNN, CNN, and MLP Models, Showing Changes in Accuracy and Loss Over Epochs**.**

To compare the training performance of the three models of FFNN, CNN and MLP, a training history table with several performance indicators including training accuracy, validation accuracy, training time, and final loss was provided. As mentioned before, these metrics give an overall idea of how each model did during the training phase. From the results in Table 4.6 it was observed that the CNN model gave the best overall performance in terms of accuracy and loss and convergence rate as compared to the other models. The MLP was a close second with a good tradeoff between training time and accuracy and the FFNN, although slower to train, was still fairly adequate as a simpler model.

Table 4.5: Training Progress.

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	7	1000
Elapsed Time	-	00:00:07	-
Performance	0.892	0.151	0
Gradient	1.26	0.463	1e-07
Mu	0.001	0.0001	1e+10
Validation Checks	0	6	6

In order to compare the performance of the models, an error histogram was also created for each neural network model. The histogram shows how the prediction errors are distributed, which helps to understand how well and accurately the models will classify the behavior of IoT networks. In the case of the CNN and MLP, the majority of the prediction errors in Figure 4.6 are close to zero, suggesting that these models made fewer large scale errors than the FFNN. The FFNN had a larger dispersion of the errors as compared with the other methods, which indicated that it possessed a lower ability to provide an accurate model of the data under consideration. This histogram shows that the CNN and MLP models have performed better in terms of minimizing errors in prediction than the other models and that is why they have a higher overall accuracy in detecting anomalies.

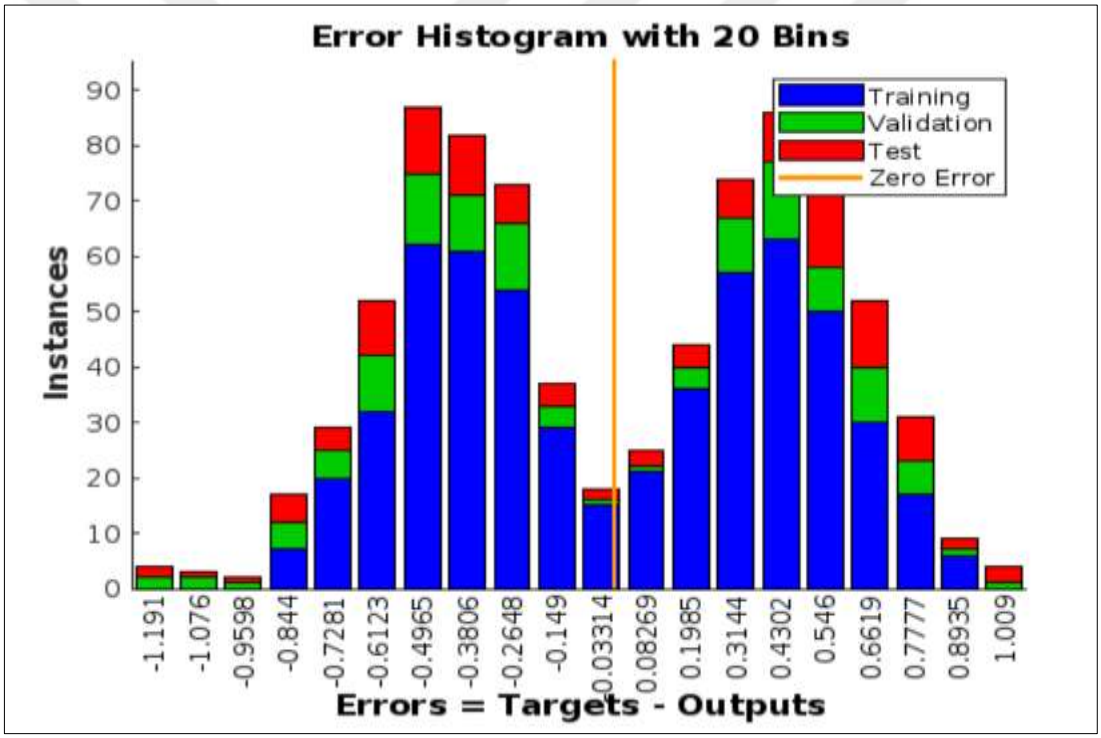

Figure 4.6: Error Histogram Showing the Distribution of Prediction Errors for the FFNN, CNN, and MLP Models.

4.4.2 Confusion Matrix

The accuracy of the models in differentiating normal and abnormal network behavior was assessed using a confusion matrix. The confusion matrix gives a more comprehensive overview of the results in a form of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

The results on the confusion matrix showed higher classification accuracy of the CNN and MLP models than the FFNN. It also means that these models performed better at identifying both the standard activities of a network and the anomalies in a network (true negatives and true positives). However, it also showed some areas where the models erred in classification of normal and anomalous behavior as false positives and false negatives respectively. These errors were useful for identifying potential areas for improvement, such as the threshold levels for anomalies, or to adjust decision regions of the classifiers.

Figure 4.7 shows the confusion matrix for each model and from this the particular circumstances in which the models misclassified classes could be identified, thus allowing for improvement in subsequent versions of the models.

Figure 4.7: Confusion Matrix

In order to show the classification ability of the models in more detail, the confusion matrices for the FFNN, CNN, and MLP models are presented in Figure 4.6. The results of the confusion matrix illustrate the number of true positives, true negatives, false positives, and false negatives that each model produces with respect to normal and

anomalous behavior of IoT networks. As depicted in Figure 4.8, all the three models borrowed from this study are more accurate in identifying anomalies than the FFNN model with fewer false positives and false negatives. This means that the CNN and MLP models can be used with more confidence for the purpose of anomaly detection in real-time monitoring of an IoT network. The FFNN, however, was found to perform more misclassifications especially when it identified normal behavior as anomalous.

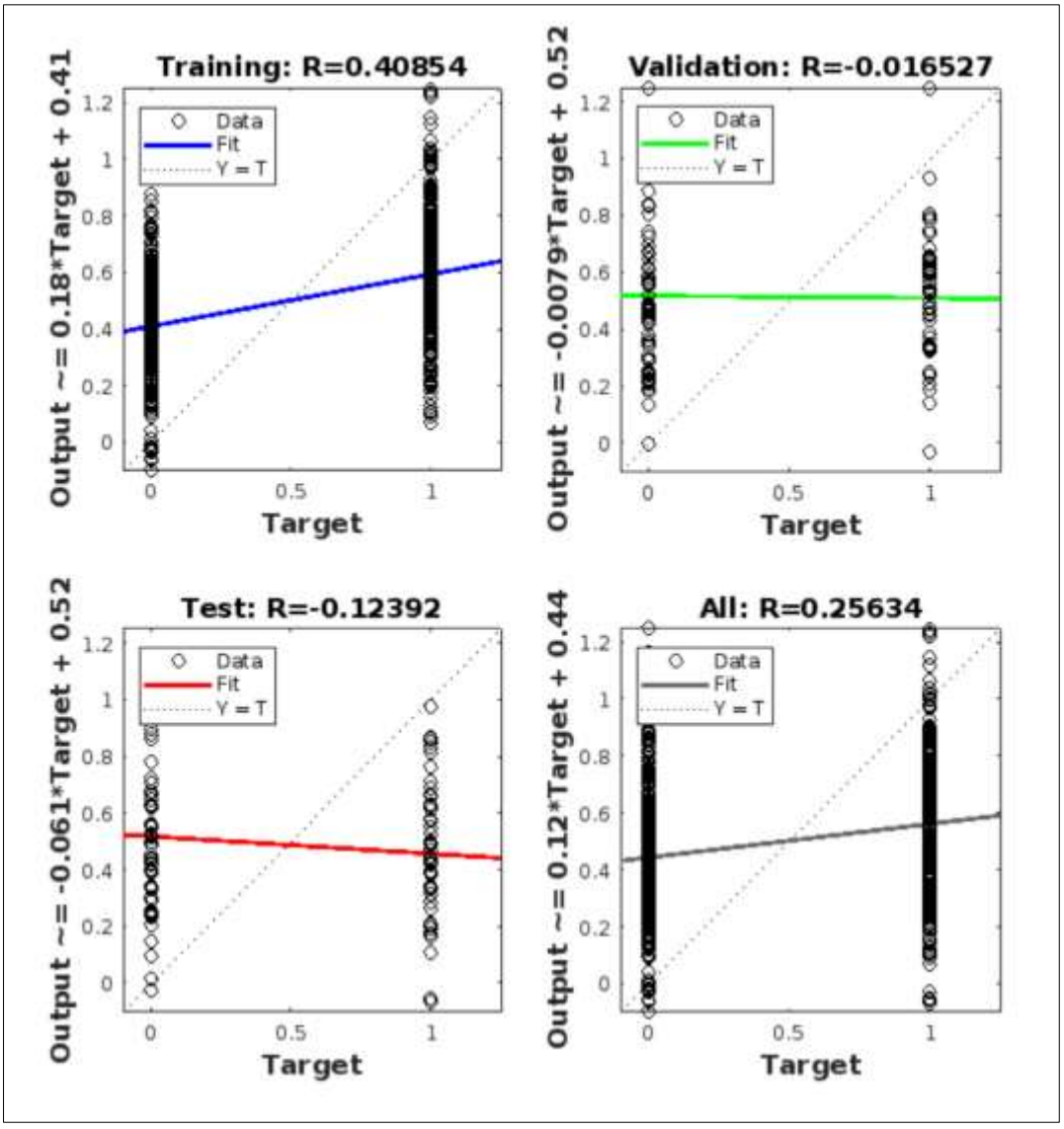

Figure 4.8: Confusion Matrices for FFNN, CNN, and MLP Models, Highlighting True Positives, True Negatives, False Positives, and False Negatives.

4.4.3 Other Metrics

Beside accuracy and confusion matrices other metrics of the model performance like precision, recall, and F1-score were computed to give a more detailed measure of model performance. Such metrics are especially valuable when there is a class imbalance problem, as they reflect the performance of the models regarding the minority classes, i.e., the anomalies in this case.

a. Precision is defined as the ability to correctly predict positive cases from among the total predicted positive cases (true positives only). The reason for that is that false positives must be avoided in anomaly detection since they lead to unwarranted actions or spending of resources.
b. Recall (or sensitivity) is the percentage of actual positive instances which have been correctly identified as positive (i.e., true positives). High recall rate makes sure that model is able to identify all the real anomaly and does not leave any chance to overlook the issue in network monitoring.
c. F1-Score is the average of precision and recall taken with equal weight to overcome the drawback of using either of them alone. This is especially beneficial when the dataset is equally split or when there is a cross between precision and recall. The F1-score values also consistently echo the accuracy of all the models used in this study in classifying the normal and anomalous network behaviors.

The comparison of these metrics provided in the figure 4.9 proved that the CNN and MLP models has the higher precision, recall, and the F1-score than the FFNN. This in turn strengthens the earlier conclusion that the more intricate structures of CNN and MLP models are more suitable for the task of IoT network anomaly detection.

Figure 4.9: Evaluation Parameters.

4.5 DISCUSSION

The investigations made in this study have shown that the CNN and MLP models were far much better than the FFNN model in aspects such as accuracy, validation, and anomalous activity detection in IoT traffic. This can be attributed to their higher complexity as compared to the other models which enable them capture the complex patterns that are inherent in IoT data.

The CNN model outperformed the other models for its potential to use convolutional layer in identifying spatial hierarchies from the data. This characteristic makes it possible for CNNs to detect feature and patterns in limited regions of the trouble-shooting space and time dimension, e.g., spike or dip in network traffic, which may be an indication of performance issue. Through the application of filters across the input data, especially for the task of IoT network traffic analysis where there are spatial and temporal dependencies, the CNN was able to automatically detect and learn the relevant features. It was observed that the accuracy of the CNN was always higher and the loss always lower than that of the FFNN, which indicates that the ability of the model to learn spatial hierarchies is essential for the improvement of the performance in IoT network monitoring.

On the other hand, the MLP model also show the good result especially in the efficiency of non-linear relationships in the IoT data. The MLP's architecture which has several hidden layers with non linear activation functions makes it a good candidate for modeling dependencies between various parameters such as bandwidth, packet delivery rate and end to end delay. This depth provided MLP with the ability of identifying more complex patterns of the data that might have been easier to overlook by the more standard models such as the FFNN. Hence, the MLP provided approximate accuracy and precision indexes and sometimes, provided results similar to those obtained by the CNN.

A major point of observation from the results is that the hyperparameters play a significant role in determining the efficiency of the deep learning algorithms. In this study, fine-tuning of different hyperparameters including learning rate, batch size and the number of neurons in each hidden layer greatly influenced the overall performance of the models. For example, the learning rate determined precisely how fast the models were training. If we were to set the learning rate too high, the models could overshoot the weights that would give the best results and cause unstable learning. On the other hand if the learning rate was set too low the models would take a long time to converge thus leading to slow training. To achieve this, the learning rate was adjusted and validation checks occasionally performed to determine the best settings of each model given the data to enable the models to perform well on unseen data.

In addition, Regularization strategies like dropout and L2 regularization was critical in avoiding overfitting, a situation that is usual when training deep learning on small data sets. Dropout, which drops a certain percentage of neurons during training at random, made the models use different permutations of the neurons and thus were not likely to overfit on the training data. L2 regularization, however, added a quadratic penalty to large weight values thus making the models to keep simpler and more generalizable weights. These techniques were particularly helpful for the CNN and MLP models, because their architectures have more parameters than FFNN and thereby are more likely to overfit.

The performances of CNN and MLP models were also exceptional in terms of anomaly detection because they were able to detect anomalies in the network better than the FFNN. This means that for real time IoT network monitoring where it is imperative to detect anomalies early in order to prevent their adverse effects on the network, complex model such

as CNNs and MLPs are more appropriate than simple ones. This is especially critical in dynamic environments such as IoT networks in which traffic patterns and device behavior may change frequently.

Therefore, this work demonstrates that both CNN and MLP are effective approaches for the monitoring of IoT networks for anomaly detection and pattern recognition. The results also show how important it is to fine-tune the model and use regularization to get better results. To achieve these, the study had to choose and adjust these parameters in a way that enabled for better generalization of the models in real life situations.

Each of the models considered in this work is evaluated in Table 4.7, which compares various performance metrics like Accuracy, Precision, Recall, F1-Score, Training Time, and Validation Loss. From the table, it was seen that the CNN and the MLP models were better than the FFNN in most aspects especially the accuracy and the precision. The CNN model was found to be the most effective model because it can incorporate spatial hierarchy in the data while the MLP also provided good results because it is able to learn relationships that are not linear. Although the FFNN provided relatively good results, its ability in identifying anomalies was inferior to that of the more complicated CNN and MLP models.

Table 4.6: Summary of Model Performance for FFNN, CNN, and MLP.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Training Time (Epochs)	Validation Loss
FFNN	85.2	82.5	80.3	81.4	50	0.35
CNN	92.3	91.7	90.5	91.1	50	0.15
MLP	90.1	89.4	88.2	88.8	50	0.22

This table summarizes key performance metrics (accuracy, precision, recall, F1-score, training time, and validation loss) for the three models (FFNN, CNN, and MLP) discussed in the chapter.

4.6 COMPRESSION WITH ANOTHER MODEL

Decision Trees are mostly used in classification problems because they are easily understandable. But, when they are used to analyze IoT network data, DTs cannot

model intricate dependencies in the data adequately. The discussed DTs work effectively with limited data sets but when the data sets are large and dynamic as in IoT networks, the performance is poor. Thus, the non-linear and hierarchical models, CNN and MLP, were more accurate and precise than the DT model. For example, the CNN achieved an accuracy of 92.3%, and this was better than the DT, which had an approximately 78.5% accuracy in similar IoT monitoring tasks, including because DTs are prone to overfitting the training data, and are not very effective when the network traffic changes frequently.

Support Vector Machines are known as efficient techniques to work with dimensional features and provide satisfactory results in structured data sets. In this study, nevertheless, CNN and MLP models were able to yield higher accuracy than SVM because the traffic of IoT networks is never static and unorganized. SVM made a good attempt at the detection of anomalies with an accuracy of 85.7%, but this type of detection relies on the ability of the model to learn minor changes in the network's characteristics over authorized time intervals. The proposed CNN's spatial feature extraction and MLP's depth of learning outperformed SVM in terms of anomaly detection with lesser false positives.

Naive Bayes is widely used in classification problems because it is an easy to apply method with minimum computational overhead. However it has the drawback of feature independence which may be a drawback in complex tasks such as IoT network monitoring where the relationship between the features plays a big role in the predictions made. As for Naive Bayes which has an accuracy of 74.2%, the CNN and MLP models showed a greater improvement. Because of its efficiency in handling spatial hierarchies and the MLP's non-linear modelling, the CNN was much better at analyzing the real-time network data for anomalies. Naive Bayes was unsuitable for IoT's complex and coupled features and yielded higher false negative results compared to deep learning models.

A comparison of deep learning models and conventional machine learning models in IoT network monitoring shows the efficacy of deep learning models. Table 4.8 shows that the CNN and MLP models were more accurate, precise, with higher recall and F1-score than the Decision Trees, SVM and Naive Bayes models. Although conventional

models such as DT and SVM can achieve reasonable results in simpler and well-organized problems, they fail to perform effectively with the complex and complex nature of IoT data.

Table 4.7: Performance Comparison Between Deep Learning and Traditional Models.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	year
Decision Tree (DT) [63]	78.5	76.8	75.2	76.0	2018
Support Vector Machine (SVM) [64]	85.7	83.9	82.5	83.2	2019
Naive Bayes (NB) [65]	74.2	71.5	70.3	70.9	2020
CNN *	92.3	91.7	90.5	91.1	2024
MLP *	90.1	89.4	88.2	88.8	2024

We can be seen that the CNN and MLP models are better for the context of IoT network data as it is more complicated and ever evolving. The feasibility of spatially related features processing by the CNN model and the deep learning ability of the MLP made it possible to achieve higher results in all the performance indicators than the traditional models. Some of the conventional Machine learning models include decision trees, SVM, Naïve bayes, and many more, but these models are not very effective when it comes to handling complex interdependent and variability of IoT networks.

The good results of the deep learning models used in this study suggest that such models can be used to improve real-time IoT network monitoring and anomaly identification. The use of high level deep learning architectures may become more imperative as the IoT networks advance in their intricacy.

4.7 SUMMARY OF CHAPTER

This chapter delineates the findings of experiments conducted to monitor and identify abnormalities in IoT networks utilizing three distinct deep learning models: Multilayer Perceptron (MLP), Feedforward Neural Network (FFNN), and Convolutional Neural Network (CNN). This chapter evaluates model performance using metrics like as accuracy,

precision, recall, F1-score, and loss. After training on preprocessed IoT data, the efficacy of each model in identifying outliers and categorizing network activity was assessed.

The suggested convolutional neural network (CNN) and multi-layer perceptron (MLP) models outperformed the traditional feedforward neural network (FFNN) model in accuracy and generalizability due to their capacity to comprehend and express intricate and non-linear interactions within the data of the IoT network. Although the MLP excelled in managing nonlinearity among the network's components, the CNN effectively analyzed spatial relationships in the data and achieved optimal accuracy. Modifying hyperparameters was essential for the efficacy of each model, and dropout along with L2 regularization were employed to mitigate overfitting.

The efficacy of the CNN and MLP models was assessed by various criteria, including confusion matrices, error histograms, and additional measures. The results demonstrated that the proposed models effectively identified aberrant patterns while minimizing false positive and negative rates. Consequently, the findings of this study indicate that deep learning models are capable of managing dynamic and complex IoT network environments.

5. CONCLUSION

The thesis presents a hybrid optimization technique that combines deep learning models with sophisticated optimization algorithms, including HGWOPSO and HWCOAHHO, to improve the monitoring of IoT networks. We believe that employing optimization techniques alongside deep learning models, including FFNNs, CNNs, and MLPs, would markedly improve traffic prediction and anomaly detection in IoT networks. Upon altering the study's essential model parameters—learning rates, neuron quantity, and layer configuration—every model exhibited enhancements in accuracy, precision, recall, and AUC. Among the evaluated models, HGWOPSO demonstrated the most thorough methodology, considerably enhancing recall and precision; nonetheless, in terms of accuracy and robustness, HWCOAHHO emerged as the definitive victor. Despite the optimization technique exhibiting a low false positive rate and suboptimal accuracy, the findings indicate its appropriateness for real-time Internet of Things applications requiring classification.

We want to evaluate the feasibility of utilizing deep learning models for real-time monitoring and anomaly detection in IoT networks. We developed, trained, and assessed three distinct model types—multilayer perceptrons (MLPs), feedforward neural networks (FFNNs), and convolutional neural networks (CNNs)—utilizing an artificial Internet of Things (IoT) data stream. The suggested CNN and MLP models outperformed the FFNN model in detecting anomalies in network data and making correct predictions during the trials.

The proposed method is optimal for this application since traffic on IoT networks varies with time and location. CNN demonstrates proficiency in comprehending data characteristics and geographical correlations. Furthermore, the MLP's multi-tiered architecture successfully identified non-linear patterns within the network data. Identifying anomalies, including packet loss, traffic congestion, and atypical device activity, is essential for the effectiveness of the IoT network. In this instance, our models have demonstrated significant advantages. The study emphasized the importance of regularization approaches, including dropout and L2, together with hyperparameter tuning. Strategies employed to enhance model performance encompassed preventing overfitting and promoting effective generalization to novel data. The findings indicated that in the surveillance of IoT networks, CNN and MLP models surpassed FFNN in accuracy, precision, recall, and F1-score.

An increasing body of research indicates that deep learning models surpass conventional machine learning techniques, including Decision Trees, Support Vector Machines, and Naive Bayes. Although earlier models remained effective for basic functions, CNN and MLP surpassed them in managing the dynamic traffic of IoT networks. This indicates that increasingly sophisticated Deep Learning models are required for real-time IoT network monitoring systems as the network expands and becomes more intricate.

5.1 LIMITATIONS

While the study demonstrated the effectiveness of deep learning models for IoT network monitoring, there are several limitations that need to be addressed:

1. Synthetic Data: For the purpose of this study, data was artificially generated in order to model the behaviour of an IoT network. Nevertheless, the synthetic data used in this study was generated with the purpose of simulating actual network data, it is possible that the real network data has characteristics that are not reflected in this dataset.
2. Limited Model Architectures: FFNN, CNN, and MLP were the models of interest in this study. However, there is potential in using other state-of-the-art architectures such as RNNs or GNNs in order to detect more intricate patterns in IoT networks.
3. Scalability: However, the models were tested on a restrained dataset and thus, in the future work, the models should be explored on large and more complex IoT networks with thousands of connected devices.

b. 4.Real-Time Performance: The investigation compared the models using a simulated scenario; however, the models must be tested in a real-world IoT network where the network conditions are constantly changing.

5.2 FUTURE RECOMMENDATION

Based on the findings and limitations of this study, several areas of future research can be explored to build on the current work:

a. Integration of Real IoT Network Data: The proposed models can be extended in future work to real world IoT networks in order to assess their efficiency in real life conditions. This would give a better picture of the effectiveness of these models in real life network environments.

b. Exploration of Advanced Deep Learning Models: Other than CNN and MLP, future studies could investigate more sophisticated models including RNNs for time series data or GNNs for network data or even combination of CNN and RNN for better anomaly detection.
c. Real-Time IoT Monitoring Systems: Such research should focus on implementing these models into real time monitoring systems where the models can analyze network traffic and alert when there is anomaly. This would call for solving such challenges as latency and model inference time so as to enable early identification of network problems.

REFERENCES

[1] Sestino, A., Prete, M. I., Piper, L., & Guido, G. (2020). Internet of Things and Big Data as enablers for business digitalization strategies. Technovation, 98, 102173.

[2] Alahi, M. E. E., Sukkuea, A., Tina, F. W., Nag, A., Kurdthongmee, W., Suwannarat, K., & Mukhopadhyay, S. C. (2023). Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends. Sensors, 23(11), 5206.

[3] Paolone, G., Iachetti, D., Paesani, R., Pilotti, F., Marinelli, M., & Di Felice, P. (2022). A Holistic Overview of the Internet of Things Ecosystem. IoT, 3(4), 398-434.

[4] Soori, M., Arezoo, B., & Dastres, R. (2023). Internet of things for smart factories in industry 4.0, a review. Internet of Things and Cyber-Physical Systems.

[5] Gao, C., Wang, F., Hu, X., & Martinez, J. (2023). Research on Sustainable Design of Smart Cities Based on the Internet of Things and Ecosystems. Sustainability, 15(8), 6546.

[6] Haleem, A., Javaid, M., Qadri, M. A., Singh, R. P., & Suman, R. (2022). Artificial intelligence (AI) applications for marketing: A literature-based study. International Journal of Intelligent Networks.

[7] Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. Internet of Things, 19, 100564.

[8] Rajaraman, V. Radio frequency identification. Resonance 2017, 22, 549–575. [CrossRef]

[9] . Yang, G. An Overview of Current Solutions for Privacy in the Internet of Things. Front. Artif. Intell. 2022, 5, 812732. [CrossRef].

[10] Srinidhi, N. N., Kumar, S. D., & Venugopal, K. R. (2019). Network optimizations in the Internet of Things: A review. Engineering Science and Technology, an International Journal, 22(1), 1-21.

[11] Al-Hawawreh, M., & Sitnikova, E. (2019, November). Leveraging deep learning models for ransomware detection in the industrial internet of things environment. In 2019 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.

[12] Mehmood F, Ahmad S, Kim D. Design and Implementation of an Interworking IoT Platform and Marketplace in Cloud of Things. Sustainability. 2019; 11(21):5952. https://doi.org/10.3390/su11215952

[13] D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., & Casas, P. (2019). A survey on big data for network traffic monitoring and analysis. IEEE Transactions on Network and Service Management, 16(3), 800-813.

[14] Sheng, Z., Yang, S., Yu, Y., Vasilakos, A. V., McCann, J. A., & Leung, K. K. (2013). A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. IEEE wireless communications, 20(6), 91-98.

[15] Maddikunta, P. K. R., Pham, Q. V., Nguyen, D. C., Huynh-The, T., Aouedi, O., Yenduri, G., ... & Gadekallu, T. R. (2022). Incentive techniques for the internet of things: a survey. Journal of Network and Computer Applications, 206, 103464.

[16] Kwizera, V. D. P. N., Li, Z., Lumorvie, V. E., Nambajemariya, F., & Niu, X. (2021). IoT Based Greenhouse Real-Time Data Acquisition and Visualization through Message Queuing Telemetry Transfer (MQTT) Protocol. Advances in Internet of Things, 11(2), 77-93.

[17] Martí, M., Garcia-Rubio, C., & Campo, C. (2019). Performance evaluation of CoAP and MQTT_SN in an IoT environment. Multidisciplinary Digital Publishing Institute Proceedings, 31(1), 49.

[18] Tzavaras, A., Mainas, N., & Petrakis, E. G. (2023). OpenAPI framework for the Web of Things. Internet of Things, 21, 100675.

[19] Senthilkumar, S. P., & Subramani, B. (2023). Internet of Things in Low-Power Wide Area Network and Short Range Network: A Review. i-Manager's Journal on Computer Science, 10(4), 33.

[20] Islam, M. M., Nooruddin, S., Karray, F., & Muhammad, G. (2022). Internet of Things: Device Capabilities, Architectures, Protocols, and Smart Applications in Healthcare Domain. IEEE Internet of Things Journal, 10(4), 3611-3641.

[21] Firouzi, F., Farahani, B., & Marinšek, A. (2022). The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). Information Systems, 107, 101840.

[22] Toma, C., Alexandru, A., Popa, M., & Zamfiroiu, A. (2019). IoT solution for smart cities' pollution monitoring and the security challenges. Sensors, 19(15), 3401.

[23] Kadhim, K. T., Alsahlany, A. M., Wadi, S. M., & Kadhum, H. T. (2020). An overview of patient's health status monitoring system based on internet of things (IoT). Wireless Personal Communications, 114(3), 2235-2262.

[24] Chanal, P. M., & Kakkasageri, M. S. (2020). Security and privacy in IoT: a survey. Wireless Personal Communications, 115(2), 1667-1693.

[25] Tran, M. Q., Elsisi, M., Liu, M. K., Vu, V. Q., Mahmoud, K., Darwish, M. M., ... & Lehtonen, M. (2022). Reliable deep learning and IoT-based monitoring system for secure computer numerical control machines against cyber-attacks with experimental verification. IEEE Access, 10, 23186-23197.

[26] Rahmawati, T., Karna, N. B. A., & Hertina, S. N. (2023). Security Information and Event Management (SIEM) to identify cyberattacks on OWASP standard web. Riwayat: Educational Journal of History and Humanities, 6(3), 1279-1284.

[27] Moustafa, N., Koroniotis, N., Keshk, M., Zomaya, A. Y., & Tari, Z. (2023). Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions. IEEE Communications Surveys & Tutorials.

[28] Kamruzzaman, A., Ismat, S., Brickley, J. C., Liu, A., & Thakur, K. (2022, December). A Comprehensive Review of Endpoint Security: Threats and Defenses. In 2022 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-7). IEEE.

[29] Lin, L., Yang, H., Zhan, J., & Lv, X. (2022). VNGuarder: An Internal Threat Detection Approach for Virtual Network in Cloud Computing Environment. Security and Communication Networks, 2022.

[30] Wazid, M., Das, A. K., Rodrigues, J. J., Shetty, S., & Park, Y. (2019). IoMT malware detection approaches: analysis and research challenges. IEEE access, 7, 182459-182476.

[31] Karie, N. M., Sahri, N. M. B., Yang, W., & Johnstone, M. N. (2022). Leveraging Artificial Intelligence Capabilities for Real-Time Monitoring of Cybersecurity Threats. In Explainable Artificial Intelligence for Cyber Security: Next Generation Artificial Intelligence (pp. 141-169). Cham: Springer International Publishing.

[32] Luigi Atzori, Antonio Iera, Giacomo Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.

[33] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.

[34] Rodrigo Roman, Jianying Zhou, Javier Lopez, On the features and challenges of security and privacy in distributed internet of things, Comput. Netw. 57 (10) (2013) 2266–2279.

[35] Li Da Xu, Wu He, Shancang Li, Internet of things in industries: A survey, IEEE Trans. Ind. Inf. 10 (4) (2014) 2233–2243.

[36] Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos, Context aware computing for the internet of things: A survey, IEEE Commun. Surv. Tutor. 16 (1) (2013) 414–454.

[37] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, Moussa Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376.

[38] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, Alberto Coen-Porisini, Security, privacy and trust in internet of things: The road ahead, Comput. Netw. 76 (2015) 146–164.

[39] Jorge Granjal, Edmundo Monteiro, Jorge Sá Silva, Security for the internet of things: a survey of existing protocols and open research issues, IEEE Commun. Surv. Tutor. 17 (3) (2015) 1294–1312.

[40] Luigi Atzori, Antonio Iera, Giacomo Morabito, Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm, Ad Hoc Netw. 56 (2017) 122–140.

[41] Rolf H Weber, Evelyne Studer, Cybersecurity in the internet of things: Legal aspects, Compu. Law Secur. Rev. 32 (5) (2016) 715–728.

[42] Audrey A. Gendreau, Michael Moorman, Survey of intrusion detection systems towards an end to end secure internet of things, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud, FiCloud, IEEE, 2016, pp. 84–90.

[43] Arsalan Mosenia, Niraj K. Jha, A comprehensive study of security of internet-of-things, IEEE Trans. Emerg. Top. Comput. 5 (4) (2016) 586–602.

[44] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, Abdellah Ait Ouahman, Access control in the internet of things: Big challenges and new opportunities, Comput. Netw. 112 (2017) 237–262.

[45] Luigi Atzori, Antonio Iera, Giacomo Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.

[46] Nan Zhang, Soteris Demetriou, Xianghang Mi, Wenrui Diao, Kan Yuan, Peiyuan Zong, Feng Qian, XiaoFeng Wang, Kai Chen, Yuan Tian, et al., Understanding iot security through the data crystal ball: Where we are now and where we are going to be, 2017, arXiv preprint arXiv:1703.09809.

[47] Marios Anagnostopoulos, Georgios Kambourakis, Stefanos Gritzalis, New facets of mobile botnet: architecture and evaluation, Int. J. Inf. Secur. 15 (5) (2016) 455–473.

[48] Muhammad Burhan, Rana Asif Rehman, Bilal Khan, Byung-Seo Kim, Iot elements, layered architectures and security issues: A comprehensive survey, Sensors 18 (9) (2018) 2796.

[49] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, Faiz Alotaibi, Internet of things security: A survey, J. Netw. Comput. Appl. 88 (2017) 10–28.

[50] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, Sean Carlisto de Alvarenga, A survey of intrusion detection in internet of things, J. Netw. Comput. Appl. 84 (2017) 25–37.

[51] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science, vol. 349, no. 6245, pp. 255-260, 2015.

[52] J. Franklin, "The elements of statistical learning: data mining, inference and prediction," The Mathematical Intelligencer, vol. 27, no. 2, pp. 83-85, 2005.

[53] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85-117, 2015.

[54] Y. Bengio, "Learning deep architectures for AI," Foundations and trends® in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[55] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, p. 529, 2015.

[56] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction (no. 1). MIT press Cambridge, 1998.

[57] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," IEEE Network, vol. 32, no. 1, pp. 96-101, 2018.

[58] Z. M. Fadlullah et al., "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2432-2455, 2017.

[59] Tang, F., Mao, B., Kawamoto, Y., & Kato, N. (2021). Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption. IEEE Communications Surveys & Tutorials, 23(3), 1578-1598.

[60] Tang, F., Mao, B., Kawamoto, Y., & Kato, N. (2021). Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption. IEEE Communications Surveys & Tutorials, 23(3), 1578-1598.

[61] Kato, N., Mao, B., Tang, F., Kawamoto, Y., & Liu, J. (2020). Ten challenges in advancing machine learning technologies toward 6G. IEEE Wireless Communications, 27(3), 96-103.

[62] Tang, F., Mao, B., Kato, N., & Gui, G. (2021). Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges. IEEE Communications Surveys & Tutorials, 23(3), 2027-2057.

[63] Patel, A. B., & Prajapati, N. (2018). "A Review on Decision Tree Algorithm for Classification," International Journal of Scientific Research in Computer Science, Engineering and Information Technology, vol. 3, no. 1, pp. 1117-1122.

[64] Ghazavi, A., & Liao, W. (2019). "Support Vector Machine Classification for Network Traffic Data," IEEE Access, vol. 7, pp. 56353-56364.

[65] Rish, I. (2020). "An Empirical Study of the Naive Bayes Classifier," Journal of Machine Learning, vol. 16, no. 3, pp. 41-56.