REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

A NEW IOT SECURITY FRAMEWORK USING HYBRID DEEP LEARNING TECHNIQUES

Amjed Sabbar Kokaz KOKAZ

Doctor of Philosophy

Supervisor

Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN

İstanbul, 2025

A NEW IOT SECURITY FRAMEWORK USING HYBRID DEEP LEARNING TECHNIQUES

Amjed Sabbar Kokaz KOKAZ

Electrical and Computer Engineering

Doctor of Philosophy

ALTINBAŞ UNIVERSITY

2025

The dissertation titled A NEW IOT SECURITY FRAMEWORK USING HYBRID DEEP LEARNING TECHNIQUES prepared by AMJED SABBAR KOKAZ KOKAZ and submitted on 20/06/2025 has been **accepted unanimously** for the degree of PhD in Electrical and Computer Engineering.

Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN	Department of Software Engineering, Altınbaş University	_____
Prof. Dr. Osman Nuri UÇAN	Department of Electrical and Electronic Engineering, Altınbaş University	_____
Asst. Prof. Dr. Hameed Mutlag Farhan FARHAN	Electrical and computer Engineering, Altınbaş University	_____
Asst. Prof. Dr. Zeynep ALTAN	Department of Software Engineering, İstanbul Beykent University	_____
Asst. Prof. Dr. Serdar KARGIN	Department of Biomedical Engineering, İstanbul Arel University	_____

I hereby declare that this dissertation meets all format and submission requirements of 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.

Amjed Sabbar Kokaz KOKAZ

Signature

DEDICATION

First, I would like to Allah Almighty for the power of mind, health, strength, guidance knowledge and skills to complete this study. This thesis is wholeheartedly dedicated to my beloved father, who have been my source of inspiration, he tells me that" every success in your life will be best gift for me". To my mother, who have been supporting me with the kind and pure love.

ABSTRACT

A NEW IOT SECURITY FRAMEWORK USING HYBRID DEEP LEARNING TECHNIQUES

KOKAZ, Amjed Sabbar Kokaz

Phd., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Ayça KURNAZ TÜRKBEN

Date: 06 / 2025

Pages: 106

Connecting systems, apps, data management, operations, the Internet of Things builds a network, continuously supports organizations while also opening up new avenues for cyberattacks. IoT security is currently seriously threatened by illicit downloading and virus attacks, which have the potential to compromise private data and harm a company's reputation and finances.

Here we describe a hybrid deep learning optimization strategy for detecting and averting assaults in Internet of Things environments. We build a cybersecurity warning system index by first identifying and quantifying pertinent aspects, and then we assess the situation. We employ bio-inspired approaches to increase the efficiency of an Intrusion Detection System (IDS) by lowering the dimensionality of the data and eliminating noisy inputs.

One such method that improves IDS effectiveness is the Grey Wolf Optimization (GWO) algorithm, which can identify both typical and anomalous network congestion. By using different pre-processing techniques, we have enhanced the intelligent initialization step and made sure that informative features are there right away. To minimize underlying data characteristics in a large data environment, To find and confirm index components, integrate

Whale and Grey Wolf optimization with a deep learning strategy in simulation to prevent attacks. TensorFlow is a deep neural network that uses system software plagiarism detection to classify software that has been copied.

Our suggested strategy for assessing cybersecurity threats in IoT offers better classification results than current approaches, as evidenced by experimental data. Therefore, we use Whale and Grey Wolf Optimization (WGWO) in combination with a deep convolutional network for efficient attack avoidance in IoT.

Keywords: WGWO, IOT, Deep Learning, Machine Learning, Connected Devices Security.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

AMQP : Advanced Messaging Queuing Protocol

AMQP : Advanced Message Queuing Protocol

ANN : Artificial Neural Network

AUC : Area Under the Curve

FS : Feature Selection

LSTM : Long Short-Term Memory

SASL : Simple Authentication And Security Layer

SVE : State Vector Estimation

1. INTRODUCTION

1.1 GENERAL INFORMATION

The world's population is living in more ease than before because to internet technological advancements made possible by the quick development of data and telecommunications infrastructure. Nonetheless, women's security of information and proper protect are seriously threatened by the growing quantity and variety of cyberattacks. As a fundamental pro-tec security precaution, firewalls are widely utilized and deployed in locations such as government buildings, military installations, etc. since they are difficult to manually alter and take time to detect new types of attacks. Because of this, specialists in network security have created a unique technique for identifying and controlling unusual intrusion detection systems (IDSs) in networking. The first stage in creating a solid solution is feature selection (FS), which involves figuring out which element is most important. This is a crucial stage that appears to directly affect how successful the intrusion detection is. Putting in place an intrusion detection system on the network may help allay worries about Internet of Things application security. IoT devices constantly monitor all incoming and outgoing internet data. After that, the IDS compiles this information and searches for any indications of potential hacks. It uses two different approaches for threat identification: anomaly-based and signature-based. When recorded events match the rules set by successful assaults, the signature-based IDS notifies users of potential dangers. On other side, an anomaly-based approach bases a simulation on the normal behavior of the state. Using this method, the attack might then be located by contrasting observed and taught behavior[1,2]**.** These methods' primary flaw is their limited coverage of processes behavior, which is inadequate for Internet of Things networks that include a wide range of component types. Additionally, the accuracy of attack detection is decreased since different types of devices may result in different types of network activities. Deep learning is advancing across several sectors, suggesting that intrusion detection systems (IDS) built on top of it can overcome current challenges and enhance identification and mitigation accuracy. Compared to some other supervised learning methods, the deep learning model is significantly more effective at extracting complex and non-linear characteristics from network data. Keeping the labeled datasets current is not feasible due to the time and effort-consuming nature of labeling.

Furthermore, even though they are essential to raising the accuracy of IDS detection, high quality dataset for IDS training is hard to come by. Any company that operates needs electronic data, and a person may be able to afford to part with their personal data even though it is very valuable to them. In the modern digital age, information security has become essential, and there needs to be a practical defense against any danger. Thus, risk reduction and cybersecurity are crucial for data-driven or information-dependent enterprises. The protection of electronic information resources by technology, human action, and regulations is known as cybersecurity. It is becoming more difficult for cybercriminals to avoid security measures, which raises concerns about the safety of important digital assets. Selecting a thorough strategy that is suitable for the risky circumstance is essential. This makes it simpler to anticipate and reduce cybersecurity dangers by utilizing the newest, most innovative technique. The kinds and targets of attacks may also influence which model is best.

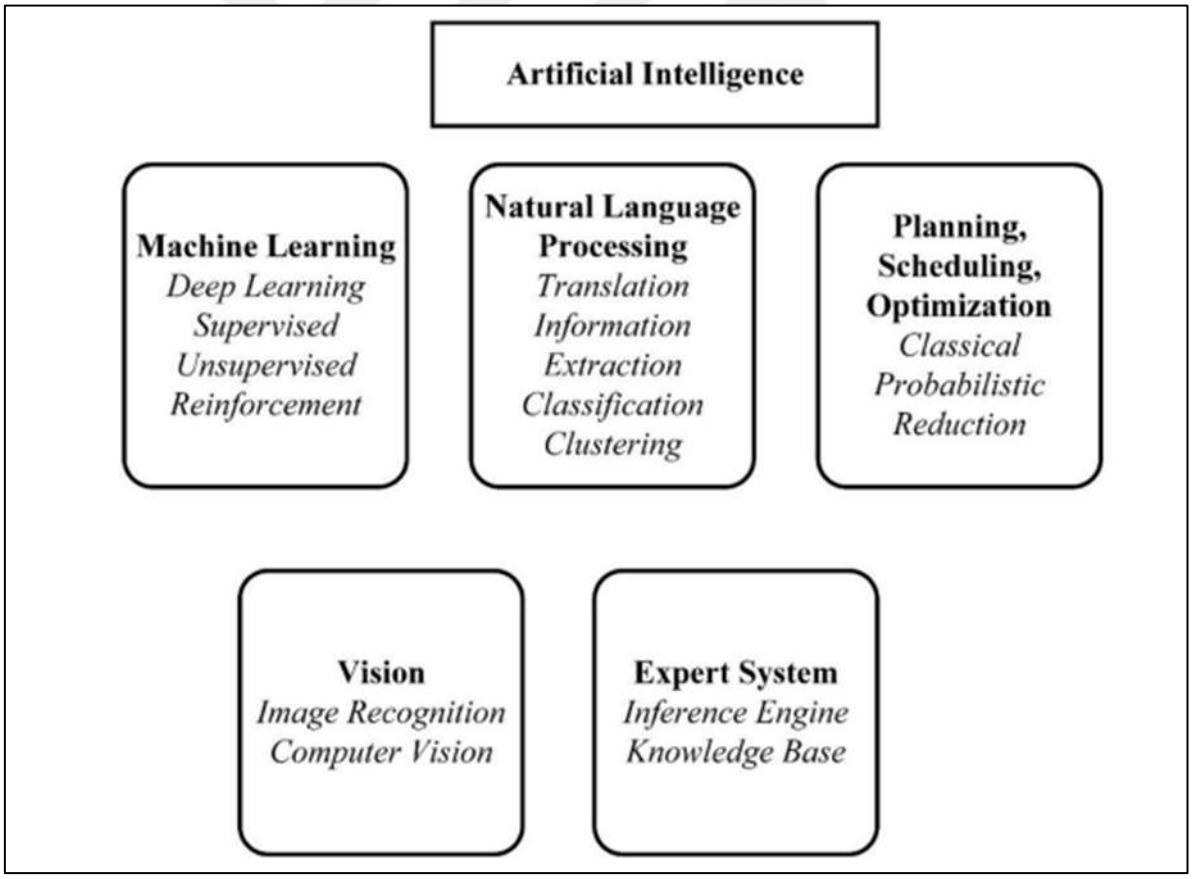

Figure 1.1: AI Taxonomy.

Fig. 1.1 shows the broad categorization of artificial intelligence approaches. Intelligent systems and cryptography are related to several fields. It's probable that sophisticated models for cyber threat identification, malware categorization, and mobile network intrusion detection were developed using deep learning technology. Advanced cybersecurity and security solutions are required, together with a secure federated learning environment, for AI models to lower risks and increase data protection[3]. Machine learning and natural processing language are only two of many new disciplines that artificial intelligence has brought forth. This does not imply, then, that using the system's capabilities has less risks now. Numerous advancements in the field of mobile systems benefit the next generation of In comparison, modern days networking is getting quicker and more dependable. The need for ongoing risk management system growth is underscored by the possibility of both known and undiscovered dangers. However, the possibility of both known and unidentified risks emphasizes how important it is to keep creating risk management systems. Unknown risks highlight the necessity of expanding the application of current risk management strategies. Consequently, cyber security decision-makers continue to face a challenging challenge even with the multitude of frameworks available for safeguarding an organization's resources against cyberthreats. Mobile virus may be found in financial services, fraudulent applications, and cryptocurrency mining. One of the most common risks to mobile phone networks is Trojan horses. Mobile applications have become more appealing to criminal activities due to patching procedures being used by hackers as a means of infecting them. Appropriate security solutions seem inadequate given the faster network transmission speed brought upon by new mobile technology. Artificial intelligence has made remarkable strides, leading to the development of advanced technologies that are enhancing the security of critical services, many of which depend on mobile networks[4]. One key innovation is the Internet of Things (IoT), a vast network of devices connected via wireless, Bluetooth, and near-field communication (NFC) technologies. IoT devices are extensively used in various sectors such as military, agriculture, healthcare, security, and smart home gadgets like thermostats and refrigerators. These devices utilize IP protocols to efficiently send data, enabling quick and autonomous communication without human intervention. But there are a lot of ways that IoT devices affect a person's life, and IoT security vulnerabilities pose a

serious threat to personal safety. It's possible that the sophisticated attacks of today will be too much for the intrusion detection systems in place. Therefore, in order to evaluate the risk of IoT infiltration and protect against threats and attacks, a sparse convolute net was employed in this work. IoT devices feature centralized design, extensive data collecting, connections in both physical and virtual environments, and a number of detrimental environmental repercussions. These characteristics enable the Internet of Things, but they also provide a means for hostile actors to listen in on discussions. IoT systems can be weaponized and compromised for distributed denial of service (DDoS), targeted code injection, and man-in-the-middle attacks. A major factor influencing data transit throughout the system is the possibility of malicious actors directly affecting Internet of Things devices. In order to avoid intermediary errors in a distributed system, it is more crucial to maintain and preserve IoT security. Then, by segmenting IoT devices, tracking, examining, and enforcing policies, as well as by securing the entire network and taking immediate automated action in the event that the system is compromised, the safety risks are reduced. Through the use of the Intrusion Detection System (IDS), IoT security was achieved. In order to accurately anticipate dangerous transmissions, these four primary categories continually monitor software packages, wireless networks, and networks. To address the aforementioned problems, several deep learning techniques have been put forth. Instead of using signatures to identify infections, data mining approaches aim to create a data-driven model based on predefined criteria. The two most popular methods for feature extraction are the steady and transient malware classification strategies. Source code retrieval can be used to obtain pertinent components from executable code, such as byte sequence or text properties. Outside of runtime settings, static analysis has been carried out. Malware must be run in a controlled environment where its activity can be tracked, and hostile behavior can be detected in order to carry out a performance simulation. Experts in the field might create useful components for a malware area identification tool that could be used for both dynamic and static evaluations[5].

1.2 PROBLEM STATEMENT

Now that the Internet of Things is growing exponentially and cybersecurity threats continue to rise, the challenge for advanced, adaptive threat detection has never been more needed.

Current strategies often do not work, considering that the IoT networks are dynamic and that the threat landscape keeps changing continuously. In this regard, our research work proposes a novel hybrid architecture that combines the working characteristics of Convolutional Neural Networks with Long Short-Term Memory networks, specifically tailored for improved accuracy and efficiency of IoT threat detection. Some of the significant challenges we need to cater to are adaptability to dynamic IoT environments, which calls for designing detection models robust enough to quickly adapt to emergent threats and the continuously changing IoT environment. For this, deep-dive analysis of the complex data streams in IoT networks and the creation of algorithms capable of rapidly identifying unusual activity patterns within everyday communication will be necessary. Our models should thus be very flexible and responsive to assure real-time threat detection and mitigation. Validation Through Dynamic Experimentation: We verify the effectiveness of our techniques dynamically by experimentation on multiple datasets, which helps our model generalize effectively and stay flexible; the result is an excellent reduction in rates of false positives and negatives with high performance. We verify our model in several different real-world situations to guarantee the robustness and reliability of the model for real-time applications. Resource Utilization Improvements: Optimal resource trade-offs need to be achieved between high-performance detection and the resource-constraint nature of IoT devices. We aim to develop efficient, resource-based detection systems that are highly practical and accurate in real-world environments with constrained resources. Thus, the solutions we aim to develop should be effectively deployable on an extensive range of IoT devices by our algorithms' optimization so that they minimize computational demands without loss of robustness in threat detection[6].

1.3 RESEARCH CONTRIBUTION

We have achieved an essential advance in IoT security by developing a new powerful approach that amalgamates the best features of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks into a unified hybrid model. Such a novel model has furnished a robust framework for detecting and analyzing complex threat patterns within large-scale and diversified IoT environments, which will enhance our power toward security threat detection and effective response mechanisms. Innovative Hybrid CNN-

LSTM Architecture: This advanced architecture perfectly marries the spatial feature extraction ability of Convolutional Neural Networks (CNNs) and the ability to model temporal sequences with the Long Short-Term Memory (LSTM) network. In this sense, it helps us better detect and analyze elaborate patterns of threats in the IoT environment and hugely outperforms earlier models. Our model is based on a hybrid model to give the best performance and reliability in safeguarding an IoT network. Advanced Optimization Techniques We have utilized advanced optimization techniques to optimize our model for efficiency and effectiveness. Techniques such as PCA, model quantization, and pruning enhance the model to process data; this makes it possible for deployment within resource-constrained IoT systems. These include optimizations toward increased scalability and practicality, making our approach more practical and applicable where all other solutions presented previously failed. We test our approach against three different datasets, namely IoT23, N-BaIoT, and CICIDS2017, which gives flexibility to the broad spectrum of IoT scenarios and the threat environment it can address. These are broad datasets used during model training to secure their robustness and adaptability. All the extensive tests carried out covered 14 categories of attacks, and the results derived indicated that our model was doing better than existing methods in terms of both effectiveness and accuracy. The given comprehensive validation testifies to the significant impact of our research on enhancing IoT security[7].

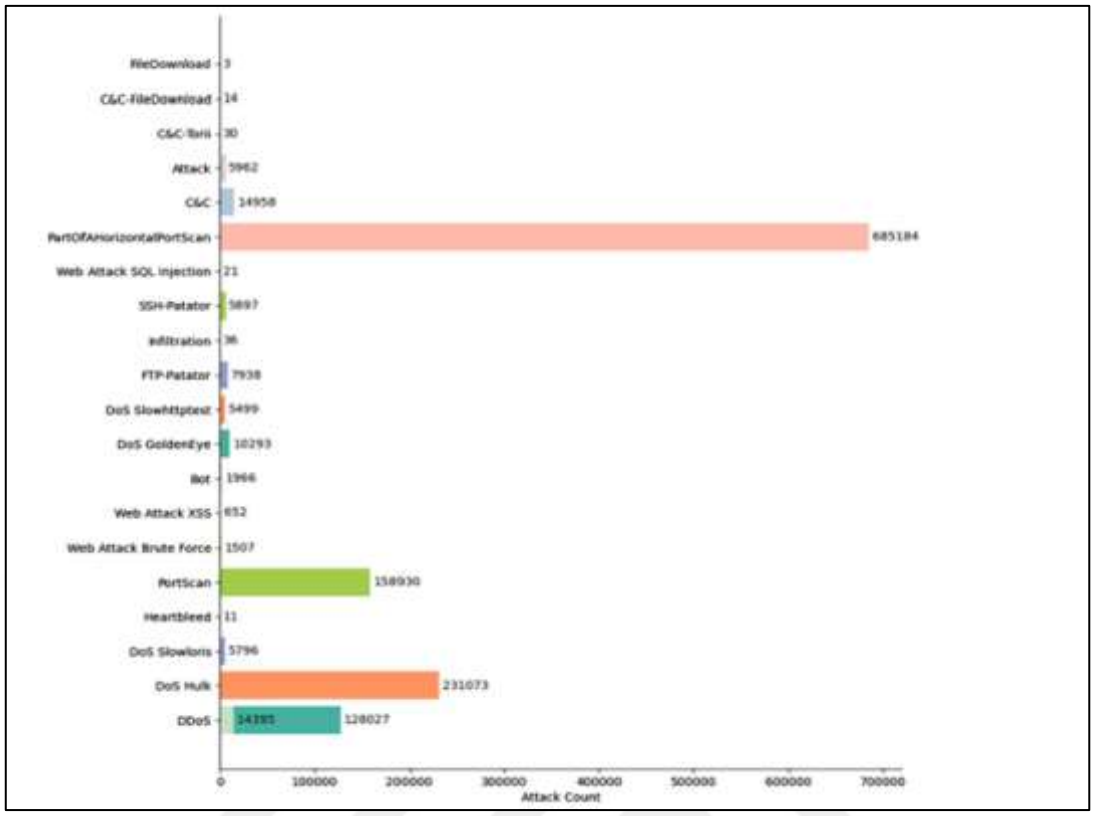

Figure 1.2: IoT Security Attacks and Their Frequencies Addressed in This Research.

1.4 DEEP LEARNING AND MACHINE LEARNING-BASED IOT SECURITY SOLUTIONS

The Internet of Things is a vast collection of physical objects which are interconnected using wired or wireless technology. These devices have minimal computing and storage capabilities. These devices can collect, manipulate the data, and transmit over the Internet because they are embedded with sensing capabilities, acting capabilities, and software other than electronics. IoT technology permeates many sectors, including energy, retail, healthcare, agriculture, and military. More and more intelligent gadgets are carrying sensitive data, such as location, health information, and contact details—making the security of IoT devices a growing primary concern. The thing is, keeping it secure in such a vast, complex system is challenging. IoT gadgets often work in an open environment and have minimal resources in terms of memory, bandwidth, battery life, and processing power. This leads to challenges in creating complex security algorithms. The IoT is wholly connected and dependent on other systems, which makes it much more vulnerable to attacks than

regular computing platforms. It opens up new paths for attack. In these large networked systems, the traditional security methods, including encryption, access control, and identification, tend to be ineffective because there is usually an intrinsic imperfection associated with each element. Deep learning and machine learning are two great avenues to boost IoT security. Machine learning utilizes many algorithms that learn from past experiences and improve their performance. Deep learning is a subset of ML, which helps in multiple layers for abstracting and altering features. This will prove that they can exceed the traditional security measures in dynamic networks by training models to detect and protect from broad spectrums of attacks, including new or emerging threats[8].

a. Briefly explain the different levels of IoT infrastructure.

b. Graphically illustrate the overview of security issues, categories of threats, and likely assaults per each infrastructure component of the IoT system

c. Carry out a thorough analysis of frameworks and ML and DL classifiers that provide security for the IoT environment at every tier.

4. Talk about the gaps in the research and upcoming difficulties to help shape the future course of the study.

1.5 IOT INFRASTRUCTURE AND ATTACKS

The architecture of the Internet of Things (IoT) is composed of three fundamental levels: the application layer, the network layer, and the perception layer. These layers work together to enable seamless communication and functionality within IoT systems. Figure 3 illustrates this structure, highlighting the distinct roles and interactions of each layer[9].

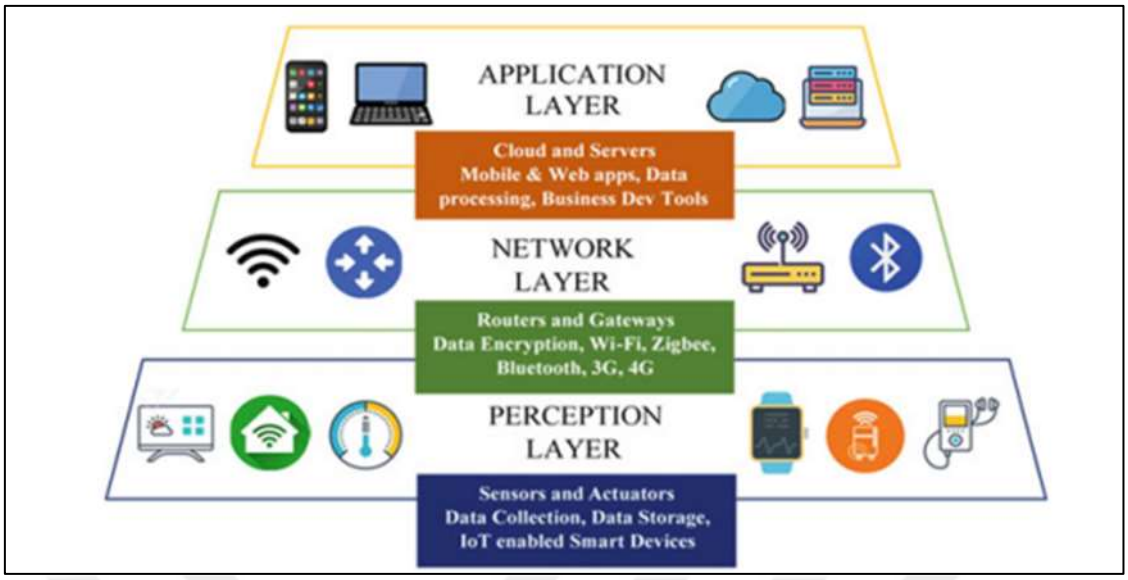

Figure 1.3: IoT Architecture.

Every layer is important: the application layer analyzes and uses the data to carry out specified activities, the network layer manages data transmission, and the perception layer collects data using sensors and devices. This multi-layered configuration does, however, come with a drawback. Attacks can happen at any layer, or even at several layers at once, making it difficult to identify and mitigate them.

1.6 PERCEPTION INTERFACE

The hardware elements that transmit and receive data are under the control of the perceptual layer, sometimes referred to as the physical layer. This covers gadgets and sensors that make use of Bluetooth, Zigbee, and RFID, among other communication protocols. These connections are made possible by protocols such cellular networks.

1.7 NETWORK INTERFACE

The transmission medium that links devices to intelligent services is the network layer. It makes use of IPv6, Wi-Fi, 3-5G, GSM, and other protocols. In addition, this layer houses local servers and clouds that store and process data, serving as a middleman between the perception layer and the rest of the network. The middleware makes it easier for various IoT

devices to cooperate, find each other, and become aware of each other and their immediate IoT surroundings[10].

1.8 WEB/APPLICATION INTERFACE

The web/application interface plays an important role with the Internet of Things (IoT) providing user access through web and mobile-based software applications. This layer supports various smart services, including:

a. Smart Grids: Enhancing the efficiency and reliability of electricity distribution.

b. Smart Transportation: Improving traffic management and vehicle communication.

c. Smart Agriculture: Optimizing farming practices through data-driven insights.

d. Smart Homes and Buildings: Automating and securing living spaces.

e. Smart Healthcare: Enabling remote monitoring and personalized medical care.

1.9 THREATS IN IOT

Cyberattacks and physical attacks are the two main categories of dangers in the IoT environment. Depending on the attack's nature, these risks may affect one or more system levels[11].

1.9.1 Cyberattacks

Cyberattacks involve infiltrating a system to target various IoT nodes connected to a wireless network. The aim of these attacks is to alter, remove, steal, or destroy data to manipulate user information. Generally, there are two types of cyberattacks[12]:

a. Active Attacks: These attempt to modify or interfere with the system's regular activities by means of direct involvement.

b. Passive Attacks: These involve eavesdropping or data interception and concentrate on gaining unwanted access to information without changing the way the system operates.

1.9.2 Physical Attacks

Conversely, physical attacks entail causing harm to the IoT devices directly. Particularly at risk are gadgets like routers, sensors, cell phones, and cameras. These assaults damage the devices physically, rather than necessarily breaking into the network, to disrupt services.

These kinds of attacks emphasize how crucial it is to have strong security measures in Internet of Things environments in order to safeguard the network's physical and digital components. Figure 4 presents these ideas.

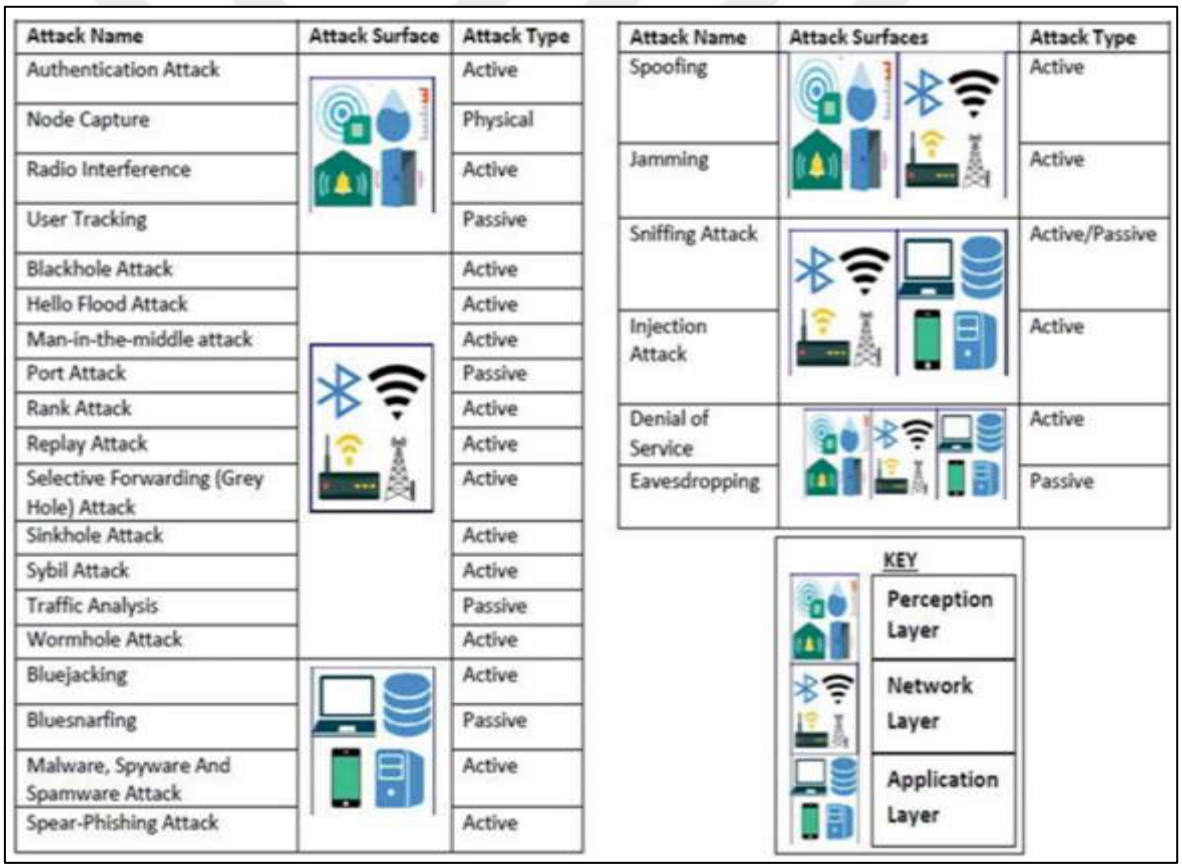

Figure 1.4: Threats in IoT.

1.10 REVIEW OF SOLUTIONS

1.10.1 Perception Interface

Wang et al. introduced an innovative authentication method that leverages the multidimensional properties of radio channels to enhance accuracy. This system employs an extreme learning machine model to detect spoofing attempts. Their research demonstrated how deep learning can be applied to develop authentication systems that utilize Wi-Fi signals from IoT devices, such as smart TVs and air conditioners, to gather physiological and behavioral characteristics of individuals. These characteristics are extracted by a deep neural network, which then generates a unique Wi-Fi fingerprint for each user, significantly improving the precision of user identification[13]

Physical layer authentication was tackled as a convex problem by Fang et al., who used an adaptive solution based on kernel least mean square. By employing RF-PUF frameworks along with neural networks to verify wireless nodes in real-time, they improved the security of Internet of Things networks. Liao et al. combined deep neural networks (DNN) with data augmentation to increase the accuracy of multi-user authentication. A centralized solution was presented by Namvar et al. to handle jamming assaults in IoT networks with constrained resources. In order to prevent jamming and interference, Aref et al. created a system of numerous wireless autonomous cognitive radios (WACRs) employing spectrum knowledge and RL-based algorithms. Han et al. increased the effectiveness of Q-learning algorithms by combining deep CNN with RL. In order to filter out poisoning attacks, Baracaldo et al. suggested a supervised machine learning model that used data provenance. Meanwhile, Ozay et al. used SVMs and KNNs to detect false data injection attacks in smart grids, outperforming more conventional state vector estimation (SVE) methods in their work. In order to reduce the possibility of eavesdropping, Hong et al. suggested a model for secure medical information transmission that uses deep reinforcement learning (DRL) and a machine-learning-based antenna design for IoT communications with ambient backscatter technology.

1.10.2 Network Interface

Security at the network layer is highly significant because IoT devices are highly vulnerable to attack. Researchers have come out with several innovative ways of predicting and detecting the following threats[14]**:**

a. Sybil Attack Prediction: Zhang and his team have developed an independent system using artificial recurrent neural networks with Long Short Term Memory (LSTM) architecture that can successfully predict any Sybil attacks. The current system developed can take advantage of the data processing capability of the LSTM architecture, which will help it capture various patterns and other anomalies that could indicate a Sybil attack. By making the model learn from the history of the network data in real-time, it can then predict the possible incidents of Sybil attack, thus making the IoT network security more robust.

b. Sybil Attack Detection in WSN: Saraswathi and team prepared a model for support vector regression to solely detect Sybil attacks for a wireless sensor network (WSN) integrated with IoT applications.

c. KNN-Based Rank Attack Detection: Neerugatti and Reddy offered a technology for searching rank attack—identification and mining; their proposed work uses the K-nearest neighbor technique.

d. Combination of PCA with Fuzzy C-Means Clustering: Deng et al. made a significant step in Sybil attack detection through a wise combination of feature selection-oriented Principal Component Analysis (PCA) with Fuzzy C-Means Clustering.

e. Detection of Wormhole Attacks: Shukla had worked at proactively developing strategies that enhance security in IPv6 low-power wireless personal area networks. The model proposed includes the three new machine-learning models, which are applied in the detection of wormhole attacks: a decision tree-based intrusion detection system, unsupervised k-means clustering intrusion detection systems, and a sophisticated dual-level hybrid system. These contributions all come under a more secure network environment[15].

These developments highlight the need for advanced security measures to protect IoT networks against diverse attack types. A few innovative approaches that researchers have come up with include:

a. DRL-Based Routing Protocol: Guo et al. developed a DRL-based routing protocol to optimize decision-making policies and defend against attacks.

b. Machine Learning in Wormhole and Rank Assault Detection: The framework developed by Fatima-Tuz-Zahra et al. for the detection of wormhole and rank assaults is based on the use of machine learning.

c. Deep Neural Network for IoT Networks: Yavuz et al. developed a deep-neural-network DNN model over a simulated 1000-node IoT network to recognize version number attacks, hello floods, and decreasing rank assaults.

d. Combining Deep Belief Networks with Optimization: Sai Srinivas and Manivannan added an optimized rider, who incorporated the deep belief network, to minimize HELLO flooding attacks.

e. ML/DL-Based Botnet Detection: Kim et al. developed a (machine-learning/deep-learning) framework for detecting botnet flooding attacks in IoT systems, using datasets from the Mirai and Bashlite botnets on smart home devices.

f. ANN for Man-in-the-Middle Attacks: Skjellum created an artificial-neural-network to recognize man-in-the-middle attacks and data tampering in IoT devices.

g. Linear Regression for DDoS and Blackhole Attacks: Amouri et al. proposed using linear regression to identify distributed-denial-of-service (DDoS) and blackhole attacks[16]

h. Low-Cost ML Algorithms for DDoS Detection: Doshi and his team tackled the challenge of detecting DDoS assaults in a cost-effective manner. They utilized low-cost machine learning algorithms, leveraging flow-based and protocol-agnostic traffic data, to identify these attacks at intermediate network nodes, demonstrating a practical and efficient approach to network security.

These cutting-edge techniques demonstrate the ongoing efforts to enhance IoT security and protect networks from a wide range of cyber threats.

Preventing DDoS Attacks: Hodo et al. trained a multilayer perceptron to counter DDoS attacks, while Saied et al. utilized an artificial neural network (ANN) for real-time detection of both known and unknown DDoS attacks. Nobakht et al. took a different approach, using classifiers like logistic regression and RBF-kernel SVM to create an intrusion detection system specifically for home devices. Smart Healthcare Security: In the smart healthcare domain, Kamel and Elhamayed showcased the effectiveness of convolutional neural networks in detecting various attacks such as selective forwarding, version, wormhole, and sinkhole attacks, all while optimizing power consumption in healthcare equipment. Thamilarasu and Chawla developed an intelligent intrusion detection system (IDS) that integrates network virtualization with a deep neural network classifier, achieving high recall and precision rates in detecting multiple network attacks. Identifying Impersonation Attacks: Aminanto and Kim introduced an unsupervised technique to detect impersonation attacks in wireless networks without the need for labeled data. Building on this, Aminanto et al. enhanced the detection capabilities by training a neural network on mixed features, combining weighted and stacked feature selection to better identify impersonation attacks[17].

Hybrid Intrusion Detection for IoT Systems: Bostani and Sheikhan created a hybrid solution combining anomaly and specification-based techniques to detect intrusions in IoT systems. Their approach was particularly effective in identifying sinkhole and selective-forwarding attacks in low-power wireless personal area networks.

These diverse approaches highlight the ongoing efforts to enhance security measures and protect IoT networks from a wide range of cyber threats.

1.11 CHALLENGES ARISING FROM DATA USED IN IOT

a. Availability of the Real World Data: Due to privacy issues and limitations on data sharing, accessing real-world IoT data can be difficult.

b. Data Preprocessing: To ensure high-quality data, the raw data must be thoroughly cleaned and formatted.

c. Data Augmentation: To build a more reliable model, data augmentation is required for small datasets.

d. Data Fusion: It is important but difficult to combine data from several, heterogeneous sources.

e. Privacy and Data Protection: In Internet of Things systems, preserving user privacy and guarding against data breaches are critical.

1.12 CHALLENGES WITH MACHINE-LEARNING AND DEEP-LEARNING ALGORITHMS

Algorithms for Machine-Learning and Deep-Learning are frequently extremely specialized and designed to address certain issues. When applied to a similar situation, a model that works well on one problem could not produce acceptable results. Neural networks have come a long way. There are serious safety problems when using applications like Internet banking and driverless cars, which demand real-time inputs and outputs with low latency. Real-world systems encounter delays not just from sensors and actuators but also from feedback systems and real-time inference, which present challenges for the efficiency of ML and DL algorithms.

Furthermore, hackers have used the quick advancements in ML and DL to their advantage to carry out complex attacks. Studies have demonstrated that cryptography systems can be cracked using ML (via Support Vector Machines, SVMs) and DL approaches. Deep Neural Networks (DNNs) can be tricked by perturbations, while Recurrent Neural Networks (RNNs) can be trained to learn decryption methods. IoT systems have been targeted by Generative Adversarial Networks (GANs) and other ML models[18].

1.13 CONSTRAINTS ON RESOURCES

IoT devices frequently have constrained computational power, storage capacity, and other resources, which makes it difficult to use intelligent strategies. Although technology such as

GPUs can enhance algorithm performance and speed up computational operations, they also use a lot of power from batteries. Strong security measures are required for IoT devices and the produced data, as demonstrated by cyberattacks like the Mirai attack against low-power, resource-constrained devices.

1.14 TRADE-OFF WITH SECURITY

In certain situations, IoT device operation must come first and security protocols second. Although this might mean sacrificing security in an emergency, an IoMT system should always be available. It can make a big difference if a patient can easily access one of the many implanted medical devices, for that matter, maybe between life and death. Striking the right balance between device security and patient safety is very important in this respect. Safety and security should both be one of the first considerations fundamentally at the time of design. While the Internet of Things affords many advantages through the amalgamation of the physical and virtual worlds, it also brings considerable security risks. It is, therefore, essential to secure IoT from a scenario of breach that could compromise private and sensitive data. As ML and DL are used to provide enhancements in the security features of IoT, self-learning, and self-optimization, have become a must to combat potential threats. This paper reviews the architecture of IoT security and explores how machine learning (ML) and deep learning (DL) can be used to protect IoT systems. It begins with a thorough examination of the IoT architecture, followed by an in-depth analysis of various security risks and challenges. The research delves into layer-by-layer attacks and threats to the IoT architecture and provides a comprehensive review of the latest ML and DL methodologies used to safeguard IoT systems[19].

2. LITERATURE REVIEW

2.1 PROTECTING INDUSTRIAL IOT DEVICE CONNECTIVITY

The Internet was once intended to connect computers, but it is now used to connect billions of items, such as energy meters, smartphones, actuators, and sensors. A new Internet revolution is about to begin because of increasing the growth exponentially in the number and kinds of web and wireless connected devices as well as the software applications running on this emerging IoT in the ensuing ten years. Currently, 99 percent of the physical objects with the potential to connect to the network are not[20].

The Internet has undergone many stages of development:

a. Phase One (1960–1980): Mainframe computers dominated this era, with researchers and academics being the primary Internet users.

b. Phase Two (1981–2012): The Internet's user base increased significantly as more people and businesses were able to use it.

c. Phase Three (2012–present): IoT is a technological endeavor that aims to establish a global network by connecting all devices.

Figures 2.1 and 2.2 illustrate the projected increase in connected devices and the anticipated number of gadgets per person in the coming years. In an Internet of Things (IoT) environment, objects, including people and devices, are assigned unique identifiers and can exchange data across a network without direct human interaction. The massive expansion of IPv6's address space has significantly accelerated the growth of IoT. With IPv6, it's conceivable that every atom on Earth could have its own IP address, and there would still be enough addresses left for over 100 more Earths. This tremendous expansion presents businesses with a fantastic opportunity to assist governments and other organizations in managing the convergence of digital and physical worlds[21].

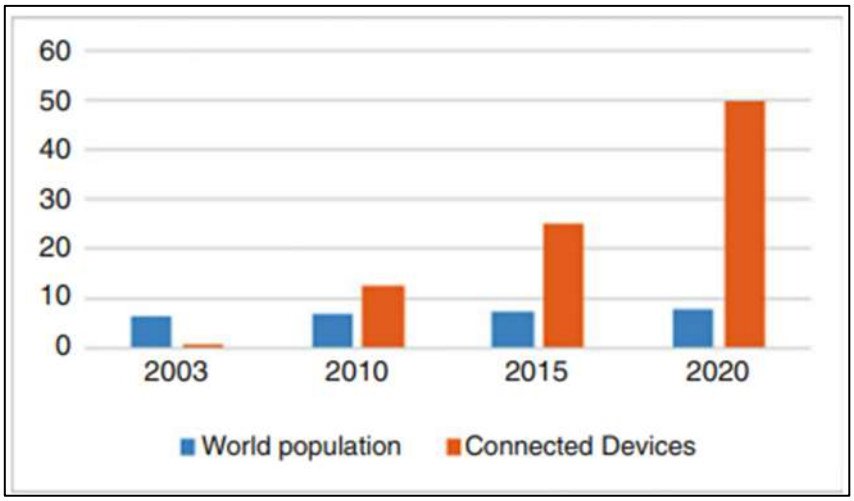

Figure 2.1: Projected Number of Connected Devices.

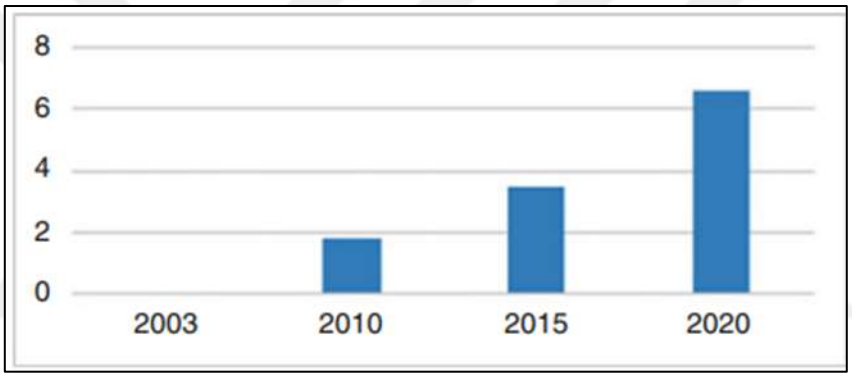

Figure 2.2: Expected Number of Connected Devices Per Person.

2.2 KEEPING IOT DEVICE CONNECTIVITY SECURE

For IoT, which is developing quickly, device connectivity security is essential. IoT creates an integrated ecosystem by connecting everything from energy meters to washing machines and cellphones. Large volumes of private information, including a person's location, may be made public due to this connectedness. If this information is viewed without authorization, there are serious concerns to data security and privacy. While there are many advantages to sharing data between devices, strict security measures are also needed to preserve data availability, confidentiality, and integrity. Real people will suffer direct consequences if computers or other devices fall victim to attacks. For example, consider the terror of a car

with brakes disabled while moving at high speed on the highway. This is where securing such connections is vital and growing so with increased challenges.

a. Resource Consumption: Other resource-intensive cryptographic algorithms may be complex to implement, given that most IoT devices have limited memory and processing capability.

b. Unreliable connectivity: Many Internets of Things devices rely on weak connectivity. It becomes hard to ensure that security updates have been propagated to all associated devices when connections are erratic.

c. User-Friendly Security: Security protocols should be user-friendly and intuitive. It is essential to ensure increased consumer adoption, so the devices should be easy to install and maintain.

Next, we will discuss other relevant IoT industrial connectivity protocols, which we have covered in the introduction - the significance and the challenges of securing communication between devices. Other essential security issues like digital signatures, encryption, authentication, authorization, and infrastructure security will be discussed in Subsection 1.3. Lastly, how to improve the device connection security of the IoT industry will be proposed.

We can ensure the benefits of IoT are realized without risking the safety and privacy of users by understanding and addressing these challenges. In our hyperconnected world, the security of the IoT is not only a technical necessity but also a critical element in ensuring the safety of people and their data[22].

2.3 PROTOCOLS CONNECTIVITY FOR IOT

All devices of the Internet of Things ecosystem use different protocols for communication with each other. This section emphasizes some of the most popular industrial Internet-of-Things (IIoT) protocols. Often in the industry, there is a mixture of modern and legacy equipment, each with differing real-time requirements and out-of-date protocols. One of the most critical problems is making secure communication possible for limited devices in the IIoT. These devices usually lack crucial system resources, such as memory, bandwidth, or

secure storage (e.g., disk space). In addition to that, being sometimes without keyboards and screens, they cannot quite conveniently run standard protocols. As a matter of fact, due to all these limitations, one can anticipate that various protocols will remain current in industrial IoT in the near future. Securing IIoT data is critical to unlocking its full potential. In such a way, the security of data stored is ensured along with data transport; data integrity and availability are maintained. These protocols allow convenient and easy data sharing and communication between devices in an industrial IoT system.

2.3.1 Transport for Message Queue Telemetry - MQTT

MQTT is a general-purpose messaging protocol for the exchange of messages between a number of clients, using a central broker. More specifically, MQTT is very well suited for real-time data messaging because the clients maintain an always-on, full-duplex TCP connection to the broker. However, MQTT lacks native support to tag messages with types and metadata, so all clients must know the message format in advance.

Developed by IBM and Eurotech, MQTT is a lightweight TCP-based publish/subscribe protocol. Even with this simplicity, it is reliable in providing protection against packet loss during client disconnections, and it enables bi-directional transmission of the messages; therefore, adding new message producers or consumers is simple. The protocol size is small, and the resource requirements are minimal, measured in kilobytes. Being an ideal protocol for low-power devices, MQTT provides tiny headers and avoids polling for near-real-time event alerting and network efficiency. The efficiency thereby trickles down to lower costs on the network as compared to HTTP. The other benefit of an MQTT server is that it can handle up to one million connected devices or users with only one server. Additionally, new functionalities within an MQTT application can be added effortlessly without modifying the current code. The protocol has also been designed so that its implementation is easy for developers to learn and use. Even though MQTT does not enforce many security measures, it implements widely used security protocols like SSL or TLS for transit security. The protocol is a multi-layered security architecture, and each of its layers covers a different class of security risks:

a. Network Layer: It provides an assured connection between the client and the broker through VPNs or physically secure networks[23].

b. Transport Layer: For secure data privacy, encryption to the data is applied by employing TLS/SSL Encryption.

c. Application Layer: The devices are authenticated through client IDs and username/password credentials. The authorization for a device shall rely on how the broker implements it.

Further, the capability to perform application-layer encryption of payloads allows MQTT to ensure secure communication between devices without using all of the transport layer's encryption capabilities. It is this multi-pronged approach towards security that ensures MQTT will forever be a dependable, lightweight protocol for Internet of Things communication[24].

2.3.2 Advanced Messaging Queuing Protocol (AMQP)

This consequently translated into developing the Advanced Message Queuing Protocol (AMQP) as a sophisticated protocol to enable message passing between applications. It is a modern standard designed by technologists for business organizations to ensure that resource sharing is possible and straightforward in both new and old applications where interaction is mediated by core middleware in the form of AMQP between the businesses, applications, and shared resources, resulting in business collaboration and infrastructure development. AMQP can be thought of as the bridge between the content of a message and the methods used to get it where it needs to go. AMQP is a secure, complete, open, standardized, interoperable, and reliable protocol. AMQP is supporting several broker architectures including peer-to-peer messaging. This ensures that messages are secured, global addressing is supported, and interoperability with standards such as JMS and WCF is maintained. It provides an effective wire protocol and abstracts message transfer processing from broker architectures and management. The protocol itself does not include layers of security. Still, it has dependencies on other security standards, which are widely used for the same purpose, like TLS for the encryption of data. Apart from that, AMQP defines a Simple Authentication

and Security Layer (SASL), which permits the protocol to negotiate a mutually agreed-upon authentication mechanism, ensuring secure and authenticated communication. In other words, AMQP provides a basis for the interconnection of business and technology across time and space, ensuring security, reliability, and effectiveness in message exchange within a dynamically changing digital space[25].

2.3.3 Constrained-Application-Protocol - CoAP

The Constrained Application Protocol is thus best suited for IoT, which is made for dependable and effective communication between servers and clients. In other words, the CoAP protocol assures the state of data and consistently allows its flow compared to other event-based protocols. CoAP servers and clients send and receive UDP packets to enable seamless communication. CoAP is excellent at content negotiation and discovery and makes it very easy for devices to communicate and share information. CoAP is considered ideally suitable for machine-to-machine applications, more particularly those with devices and networks of constrained capability. CoAP is a RESTful protocol that directly translates to constrained devices and networks while providing the simplest possible way of mapping to and from HTTP without trying to replace it or, for that matter, to provide any form of HTTP compression. CoAP is an open standard by the Internet Engineering Task Force, IETF. In most cases, it is used within closed automation networks. It supports various transport protocols like TCP, SMS, and UDP and has features like asynchronous subscription and integrated discovery. Since the development of CoAP was aimed at servicing the IoT, it integrates with proxy infrastructures very well and can also map legacy protocols to CoAPCoAP's efficiency and versatility make it indispensable in satisfying modern requirements for IoT communication. CoAP is a UDP-based protocol and uses IPsec (Internet Protocol Security) and DTLS (Datagram Transport Layer Security) to realize security. To reach the security services requirement, CoAP can operate with four security modes: NoSec, PreSharedKey, RawPublicKey, and Certificate, making it a robust and flexible alternative for safe communication for the Internet of Things[26].

a. NoSec setting: This parameter disables DTLS but allows using it in conjunction with other security mechanisms like IPsec.

b. Pre-shared Key Mode: Here, peers or groups of peers are authenticated with pre-shared keys when DTLS is enabled.

c. Respublika Mode: This supports the use of an asymmetric key without a certificate so that it is DTLS-enabled.

d. Certificate Mode: X.509 certificates are used along with asymmetric keys, with the DTLS being switched on.

However, DTLS is not built to handle multicast communication, which is essential for the CoAP protocol and forms a cornerstone in IoT scenarios; only unicast communication is provided. The present DTLS transport supports only IPv4 and not IPv6, whereas a fine-grained topic-specific setup is impossible because of the security being applied at the transport level. There are, however, several drawbacks to the use of DTLS. In the first place, CoAP mandates the use of multicast communications, and this is something that DTLS is not capable of dealing with. In the second place, CoAP does not align well with DTLS security properties. Thirdly, retransmitting all the messages in flight whenever one message is lost in transmission could be very inefficient. Even so, IPsec is not the only alternative, and DTLS is the prominent CoAP security protocol in other cases and applications[27].

2.4 SECURITY OF CONNECTIVITY PROTOCOLS

There exist many security concerns of the protocols for device connection in the Industrial IoT. Some of the major problems include those to be analyzed and detailed in this subsection. These are authentication, authorization, encryption, digital signatures, infrastructure security, etc. Most IoT protocols use conventional IT security solutions to solve the issues that include the following fundamental security measures based on generally acknowledged norms: • Transport Layer Security, TLS, that ensures security over a network connection. • IPsec/VPN (Internet Protocol Security/Virtual Private Network):

a. It provides strong encryption of each IP packet for secure IP communications, authenticating, and hence providing robust security for data transmitted over the network.

b. SSH (Secure Shell): Secure network service access over insecure networks.

c. Secure File Transfer Protocol, or SFTP, is the secure transfer method.

d. Safe Boot Loader and Auto Fallback: Safely boots the system, with the ability to automatically fall back into a secure state if necessary.

e. Filtering: Filters through only permitted traffic and restricts access.

f. HTTPS: Hypertext Transfer Protocol Secure: Secured network communication.

g. SNMP v3 (Simple Network Management Protocol version 3): This version is an improved one that contains previous properties such as encryption and increased security features to better network management and to enhance the security against unauthorized access. • Encryption and Decryption: This is associated with data security as it converts the data into a secure format accessible only to permitted users. • Datagram Transport Layer Security, or DTLS, provides security to datagram-using application programs[28].

2.4.1 The Authentication

Authentication is the process of verifying someone's identity. After this, a user can access system resources by proving who he is. In the domain of the Internet of Things, it is essential to have an authentication process followed up with authorization to ensure security within applications. Let us now get accustomed to some IoT authentication processes that are commonly carried out:

2.4.2 MQTT Protocol

A client can be identified in MQTT using a client digital certificate, user ID, or unique client identity. The client sends a password to the server of MQTT, or the server validates the client certificate using the SSL protocol. After authentication, the server shall allow access to its resources based on the identification of the client. On the other hand, the client uses SSL to confirm the server's certificate. For instance, MQTT supports encryption on both the application and transport layers. At the transport layer, TLS (Transport Layer Security) is

used to verify the digital certificates of both the client and the server. Usually, an authentication application layer applies a combination of a client username and password. However, if transport layer encryption is not used, users and passwords are passed along as clear text, which can potentially be intercepted. Unique identification, such as a MAC address of the device or a 36-character UUID, is given to each of the MQTT clients. It even supports client ID-based authentication, in which even the password and login are unnecessary[29].

2.4.3 AMQP Protocol

The Simple Authentication and Security Layer (SASL) is employed by the Advanced Message Queuing Protocol (AMQP) for client authentication. Generally, it utilizes the PLAIN method, which requires a username and password. However, this approach is susceptible to man-in-the-middle attacks if SSL is not implemented. To improve security, it is advisable to disable PLAIN authentication on the broker when SSL is not in use.

2.4.4 XMPP Protocol

SASL is also utilized by the Extensible Messaging and Presence Protocol (XMPP) to offer a variety of authentication methods. This adaptability permits the use of secure techniques such as SCRAM-SHA-1 or SCRAM-SHA-1-PLUS. For enhanced security when using client certificates, the EXTERNAL method is recommended. If less secure methods are necessary, it is prudent to log all authentication activities to detect potential attacks[30].

2.4.5 CoAP Protocol

There isn't a set authentication procedure for CoAP (Constrained Application Protocol). Instead, security options are used to build authentication systems on top of the protocol. Interoperability may suffer from a lack of standardized solutions.

2.4.6 Authorization

Authorization determines who is allowed to access which resources. Unlike some other protocols, MQTT doesn’t have built-in authorization features. Instead, MQTT servers, which function as publish/subscribe brokers[31].

To handle a large number of clients efficiently, MQTT servers often group clients based on profiles. Without proper permissions, a client might publish or subscribe to any message type. Therefore, it's crucial to implement precise restrictions to control what kinds of messages a client can send and receive. This requires customizable topic permissions on the broker's end.

Regarding authorization, MQTT brokers utilize topic permissions to establish policies that control which messages clients can publish and subscribe to. Under the MQTT 3.1.1 protocol, if a client tries to publish without proper authorization, the broker will disconnect the client instead of notifying them. To ensure security, it is advisable to restrict client publishing to authorized topics by including the client ID in the permissions. This approach can also be applied to manage topic subscriptions effectively[32].

MQTT operates over TCP, and by default, its communication is unencrypted. To secure these communications, many MQTT systems use Transport Layer Security (TLS), which is the recommended approach. However, implementing MQTT with TLS can lead to increased CPU usage, particularly for low-power devices, although the impact on the MQTT broker is typically minimal. To enhance TLS performance, session resumption can be utilized, allowing the reuse of an already-negotiated TLS session upon reconnecting to the server, thereby avoiding a full TLS handshake. It's important for organizations to be aware that not all TLS libraries support every method of session resumption, which can involve either session IDs or session tickets.

For devices with limited resources, establishing a new TLS connection can consume several kilobytes of bandwidth and require tens of kilobytes of memory, depending on the ciphers used. To reduce the TLS overhead, it's beneficial to maintain long-lived client connections. However, factors such as frequent reconnections, lack of session resumption, limited device resources, and the need to publish numerous small MQTT messages can increase the TLS overhead. Therefore, organizations should evaluate the TLS overhead in their MQTT implementation to assess the feasibility of using MQTT with TLS. If TLS proves impractical, alternatives such as TLS-PSK ciphers, which avoid resource-intensive public key operations, or encrypting the payloads of published messages can be considered.

Although TLS-PSK ciphers are not widely used, they can be a viable option in some scenarios.

Using TLS is a best practice for all IoT protocol implementations, provided the CPU and bandwidth overhead is manageable. Properly configured TLS can prevent eavesdropping, ensuring secure communication. To enhance security for application-level authentication and authorization, organizations should adopt the latest version of TLS, which helps mitigate the risks associated with older versions. Other protocols, including HTTP, should also be communicated over encrypted channels, and SSL V3 and earlier versions should be avoided due to security flaws like the POODLE attack[33].

One effective way to prevent man in the middle attacks is to validate the client's X.509 certificate from the broker. Avoid using self-signed server certificates with permissive client implementations, as this often leads to certificates not being validated, increasing the risk of man in the middle attacks. While they may be more expensive, they ensure security, especially when using protocols over WebSocket's, as modern browsers require trusted certificates for WebSocket connections.

Using client certificates offers several benefits, including verifying the identity of MQTT clients. This allows invalid clients to be disconnected before they connect to the broker, saving resources. However, managing client certificates can be complex and costly due to the provisioning and revocation processes. If the organization owns the devices, provisioning client certificates as part of firmware updates is simpler. For devices not owned by the organization, provisioning can be more challenging. It's essential to manage the client certificate lifetime, particularly without a public key infrastructure (PKI)[34].

All organizations shall use secure cipher suites when implementing protocols, as many cipher suites have been compromised. Encryption prevents eavesdropping and data interception by preventing plain-text data transport, while digital signatures confirm the message's legitimacy. SSL is used for signature and encryption, offering more secure encryption than SASL. While CoAP provides a simple authorization mechanism, it cannot differentiate between various clients' access privileges.

2.5 ENCRYPTION

2.5.1 End-to-End Encryption (E2E)

The E2E method makes sure that the application data is encrypted the entire way, preventing the broker from accessing the data. The only people who can decode the data are trusted clients who have the decryption key. This approach works especially well for firms who don't use TLS or whose authentication and permission systems aren't set up correctly. Without the decryption key, an attacker is unable to decrypt the data, even if they manage to intercept the protocol packets. When appropriate authentication and authorization procedures are absent, E2E encryption is advised to safeguard application data from attackers and other customers. It is independent of broker implementation. Trusted subscribers receive the content in plain text because it is sent over a TLS-secured communication channel. To implement client-to-broker encryption, companies need a broker plug-in capable of decrypting messages in real-time.

2.5.2 Encryption Mechanisms

Two widely used mechanisms-symmetric and asymmetric encryption-are typically used for encryption of data.

Asymmetric encryption, also known as public/private key encryption, uses a private key to decrypt data that has been encrypted with a public key. Only the corresponding private key can decrypt data encrypted with its public key. This method is ideal when the private key is available only to a select group of trusted subscribers. For added security, it is recommended that each topic have its own set of public and private keys when using asymmetric encryption[35].

Conversely, Symmetric encryption uses the same key for both data encryption and decryption. For secure IoT environments, this approach is advised since it is simpler to install. Giving each topic its own key while using symmetric encryption is a smart practice. In this manner, communications from different subjects cannot be decrypted by an attacker who manages to obtain access to one key. We may guarantee a higher level of security in our data transmissions by comprehending and using these encryption techniques effectively.

2.5.3 Payload Encryption

Payload encryption safeguards application-specific data at the application level, providing end-to-end encryption for that data. This means the actual content of the communication (the payload) is encrypted and secured, while the message's metadata remains accessible. Unlike other encryption methods, payload encryption is fully tailored to the application's needs and is not restricted by IoT protocol specifications[36].

Because binary data is more compact than text, protocols with binary payloads, like MQTT, don't require converting the encrypted message to a text format, saving bandwidth. By ensuring that message brokers don't require any unique tools to decode data, this technique permits them to route messages to subscribers without having to deal with the decryption process. When payload encryption is used, it is crucial to confirm that brokers do not try to decrypt the data. In the event that broker-side decryption is required, custom plugins can be created.

SSL is the encryption method used in AMQP. In order for SSL encryption to function, This certificate must be trusted by clients. The client's certificate has to be verified by a CA and accepted by the broker if client authentication is also necessary. These certificates are usually kept in a database that requires a password to access.

Strong end-to-end security for application data is provided by payload encryption, which is especially helpful for resource-constrained devices that are unable to employ TLS. For messages carrying sensitive data, it provides an additional degree of protection. However, secure key provisioning for the clients is necessary for the encryption and decryption of data, which can be resource-intensive for IoT devices with limited resources. It's crucial to remember that payload encryption just secures the data[37].

2.5.4 TLS vs. Payload Encryption

It's important to understand that Transport Layer Security (TLS) operates at the transport layer, while payload encryption functions at the application layer. This means you can use both TLS and payload encryption together for enhanced security. While TLS secures the entire communication channel, payload encryption specifically protects the content of

application messages from eavesdroppers or untrusted clients. Using payload encryption alone can still be beneficial, especially for organizations with constrained devices that might not handle TLS well. In such cases, payload encryption ensures that application data isn't sent in plain text, even if the overall communication channel isn't secure. However, there are risks. If the communications aren't sent over a secure channel like TLS, an attacker may be able to replay them in full or in part. The best protection is thus provided by combining the two approaches. Selecting an encryption method that successfully strikes a balance between performance and security is advised because the encryption and decryption processes might be resource intensive. This guarantees that the encryption procedure protects the data effectively without unduly taxing the device[38].

2.5.5 Checksums, Message Authentication Code Algorithms (MACs), and Digital Signatures

Making sure the messages transmitted to the broker have the correct data is essential in an IoT system with a large number of untrusted customers or clients beyond our control. If the broker does not employ TLS, this becomes even more crucial. Organizations can guarantee that the contents of the communications cannot be altered by implementing appropriate data integrity checks. A digital signature, MAC (Message Authentication Code), or content checksum can all be found in communication packets. Typically, these are included in the payload and added as a computed "stamp." The packet's integrity can be confirmed by the recipient by recalculating this stamp. In this context, "stamp" refers to a generic phrase that encompasses checksums, MACs, and digital signatures. It is recommended that the stamp contain simply the payload and the message topic[39].

The following three methods are frequently employed while making stamps:

a. Hash/Checksum Algorithms: CRC and MD5.

b. MAC: Including HMAC.

c. Digital signatures: They offer non-repudiation, which means that only the person who created the message may produce the stamp using their private key; other people can use the public key to validate it, but they are unable to produce the stamp themselves.

2.6 CHECKSUMS

Although checksum techniques are often quite quick, attackers may be able to manipulate them. An attacker can alter a communication packet and reapply the checksum to make it seem authentic if they are aware of the checksum algorithm. It is recommended that enterprises who do not use Transport Layer Security (TLS) use digital signatures or the Message Authentication Code (MAC) technique. It is advised to completely avoid depending on checksums, especially for MQTT.

2.7 MESSAGE AUTHENTICATION CODE ALGORITHMS (MACS)

Compared to digital signatures, Message Authentication Codes (MACs) are generally much faster. They offer robust security as long as a shared key is exchanged before communication begins. One type of MAC, known as HMAC (Hash-based Message Authentication Code), utilizes a cryptographic key and a hash function[40].

HMAC allows senders with the secret key to create a secure stamp. One of its main advantages is its compatibility with TLS, making it secure in various communication settings. Additionally, HMAC is very fast and efficient on low-resource systems. However, a significant drawback is that HMAC uses the same key for both encryption and decryption. This can pose a security risk, as any client with access to the secret key can both sign and validate communications..

3. A COMBINATORIAL DEEP LEARNING METHOD FOR IOT BOTTLENECK IDENTIFICATION

3.1 LITERATURE REVIEW

In several networking paradigms, such as cloud computing, fog computing, IoT, and SCADA systems, security continues to be a primary area of academic interest. At the moment, a lot of researchers are focused on botnet attack detection, an important field where it's critical to discover compromised devices before they can take advantage of the network.

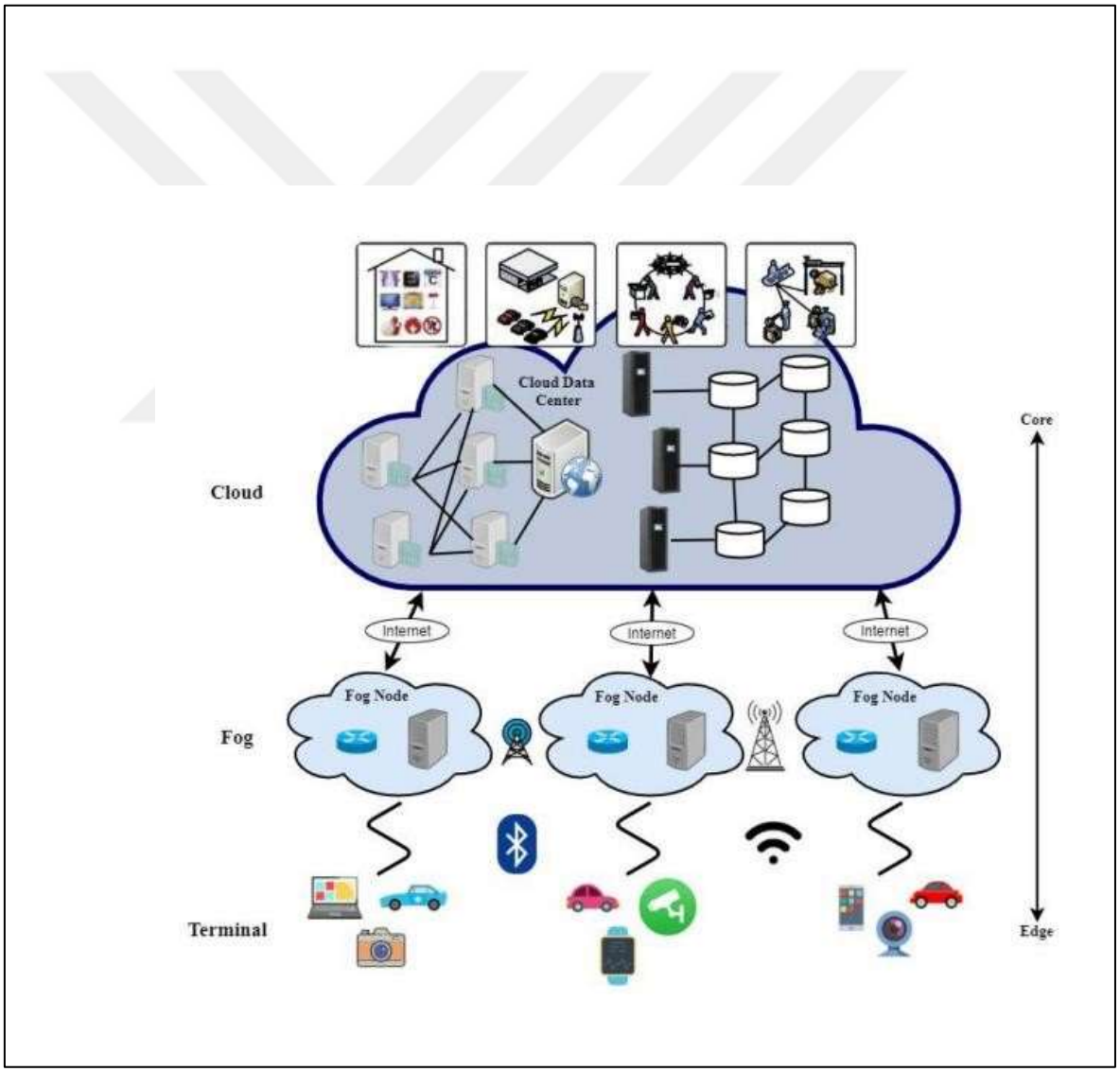

Figure 3.1: Fog Computing and Communications.

Networks can be protected against botnet attacks using artificial intelligence, particularly through machine learning (ML) and deep learning (DL) techniques. Many approaches, primarily centered on anomaly detection strategies, have been developed to counter these threats. Researchers have utilized various ML and hybrid

ML methods for botnet detection, including Bayes Net (BN), Support Vector Machine (SVM), J48, Decision Tree (DT), and Naive Bayes (NB). These machine learning methods can be classified as semi-supervised, unsupervised, or supervised.

For example, Prakash et al. experimented with K-Nearest, SVM-NB for identifying DDoS packets. Their findings showed that KNN performed best, achieving an accuracy rate of 97%, compared to SVM and NB, which had accuracy rates of 82% and 83%, respectively. Other experiments have also demonstrated the effectiveness of SVM, with some achieving average accuracy rates of 95.24%[41].

But ML algorithms have to overcome obstacles including low-value density data handling, learning from large amounts of data, and scalability. ML needs to advance in order to transform big data into insightful information as the amount of data increases exponentially. Large and unstructured datasets are easier for DL algorithms, a type of ML, to manage, which makes them ideal for IoT environments. Several deep learning (DL) and hybrid DL techniques have been used to identify different kinds of malware in Internet of Things devices. For instance,[42] one study achieved 99.87% accuracy with a low false positive rate and quick testing times by defending IoT settings against DDoS, brute force, bot, and infiltration assaults using a DL-based method. Two-way LSTM has been applied in other studies to packet level inspection on Internet of Things networks utilizing both regular IoT traffic and data from Mirai. Furthermore, some research has achieved excellent accuracy rates by deploying detection techniques via Software-Defined Networking (SDN). For example, one method achieved a 95% accuracy rate in detecting DoS, login, and probing attacks using the KDD99 dataset with a Restricted Boltzmann Machine. Using the CTU13 and ISOT datasets, a different study presented a hybrid model that combines CNN and RNN to categorize benign traffic and botnets. Furthermore, in SDN-based systems, DL techniques have shown to be successful in intrusion detection.[43] Research has demonstrated that DL techniques can detect botnet assaults in non-SDN infrastructures; nevertheless, additional

Studies are required to assess the viability and effectiveness of applying DL algorithms to identify and stop botnet assaults on SDN controllers. highlights the new directions in IoT security research by presenting several ML- or DL-based detection algorithms. The investigation of various datasets is still ongoing, with a special emphasis on hybrid deep learning combinations that enhance accuracy and precision while decreasing false positive rates and processing times[44].

Table 3.1: Security Issues in Fog Computing.

Attacks	Description
Spam	Hackers created and disseminated an undesirable message. One of the biggest security risks is spam since it can spread viruses and use up resources.
DoS	Send a ton of fake requests large the Fog nodes in order to prevent genuine users from accessing them.
Man-in-the-middle	a cyberattack wherein a third party is positioned in the way of two connecting parties in order to intercept and alter data that is being sent between them.
Tampering	The attacker modifies, pauses, or removes data packets in an attempt to lower or interfere with fog computing's effectiveness and performance.
Eavesdropping	These types of attacks are also known as spoofing and sniffing attacks. Hackers steal data from personal-computers, smartphones, and any devices without the users' consent and transmit it across the internet.
Jamming	a denial of service attack in which a single node occupies the channel used by other nodes to communicate, preventing them from interacting.
Forgery	The attackers impersonate the victims and use false information to trick them. This attack lowers network performance in terms of energy, storage, and bandwidth because of the fake data packets.
Sybil	An attack where a node assumes the identity of another node in order to take over and compromise fog nodes, exposing users' personal information and producing false sensing data.

3.2 CHALLENGES IN ENSURING SECURITY IN FOG COMPUTING

In recent years, Industry 4.0, IoT, and edge computing have seen significant advancements. At the heart of this system is the fog server, which collects and stores client data along with important information about IoT devices. Figure 7 illustrates the basic architecture of the fog paradigm, showing how edge devices connect to fog servers for communication. These fog servers, in turn, establish connections with cloud servers, allowing millions of users to access services through a network of distributed fog and edge servers[45].

However, fog computing presents unique security challenges because network assets are exposed and accessible through these devices. The highly distributed, heterogeneous, mobile, and resource-constrained nature of fog computing introduces new security risks. Implementing comprehensive security solutions to detect and mitigate threats is challenging due to the limited computational capabilities of fog systems. Fog systems are more susceptible to attacks compared to cloud systems due to their proximity to IoT devices, making them easier targets for exploitation. Moreover, Since fog computing involves the ingestion of personal data from Internet of Things devices and the cloud into itself while at the same time processing large volumes of data at a go, it becomes an attractive target for various cyber threats. Man-in-the-middle attacks, SQL injections, and zero-day exploits, among others, pose significant risks to systems and might lead to sensitive information theft. One perilous threat is a botnet attack. In this case,[46] a hacker hijacks the compromised nodes of a network to execute nefarious actions from a distance. Botnets are commonly applied in denial-of-service attacks, phishing, spam, and other malicious activity, often for monetary gain. The Software Defined Network is the architecture that is new, open, and programmable, accessible to make improvements between the network devices and the control plane. SDN offers a means to deploy network capabilities and is innovative towards the network. Securing architectures based on SDN for fog computing is an area of research that has continued to develop.[47] A robust architecture incorporating controllers based on SDN is needed for both surveillance and the detection of compromised devices within fog computing systems. That order of distributed system challenges billions of connected objects in its surveillance and management. We primarily intend to use SDN to increase the security and monitoring of fog computing environments over the Internet, safeguarding it from

cyberattacks. This involves the detection of numerous potential attacks that may affect system performance and those focusing on fog nodes[48].

3.3 DEEP LEARNING (DL)

3.3.1 Neural Network

A DNN has one input and output layer and more than one hidden layer. These layers in this processing chain perform specific tasks and organize data. Sophisticated neural networks are applied majorly in controlling unstructured data. This state-of-the-art deep neural network method nowadays falls under what is known as "deep learning." Deep learning enables artificial intelligence to categorize and analyze data in ways outside classical techniques and their performances[49]. The clear advantage of deep learning methods, however, lies in the extraction of essential features without a separate feature selection process. Moreover, DNN-based solutions are reliable and robust, with an appropriate design that ensures even zero-day threats can be detected. The real challenge is computational complexity in such deep networks, which must balance performance with resource demands[50].

$$m_i = f\left(\sum_{i=1}^{s} y_i + x_i + v\right) \qquad (3.1)$$

One of the primary usages of these advanced neural networks is in dealing with unlabeled or unstructured data. These deep learning techniques overcome basic rules for the performance of an artificial intelligence grouping and thus can creatively analyze data. A significant advantage of deep learning is that it automatically extracts essential features from the data without any separate step for feature selection. Furthermore, DNN techniques are robust and highly flexible, able to find zero-day threats if appropriately designed[51].

3.3.2 Long-Short-Term-Memory, LSTM

This is achieved by training a specific type of neural network to handle sequence data, referred to as an LSTM. Primarily, LSTM focuses on controlling the information flow between the layers so that output from one layer serves as input to the subsequent layer. Hence, learning from sequential information allows the model to comprehend them better.

In basic terms, LSTM networks have a memory to them where information on past calculations is stored to deal more efficiently with sequences. Performance is greatly increased by how LSTM network neurons are organized to communicate across layers. The three essential gates that make up the LSTM networks are input, forget, and output. Every gate is essential to controlling the information flow[52]:

a. Input Gate: The training data is kept in long-term memory via this gate. It loads the most recent input data into long-term memory and the preceding time step's data into short-term memory. The sigma function receives valuable data from the input gate after it has filtered out unnecessary information. It employs indicator values (0 Forget Gate:

b. Forget Gate: This gate is essential for deciding which data to retain and which to discard. It functions by multiplying the forget vector values with the current input gate to determine what should be kept. The output from the forget gate is used to update the long-term memory before it is passed on to the next cell.

c. Output Gate: This gate generates the final output by using the updated long-term memory, influencing the subsequent computations in the sequence[53].

The functions inside an LSTM cell are represented by the following equations:

$$a_t = \sigma(w_a[n_{t-1}, x_t] + b_a),$$

$$i_t = \sigma(w_i[n_{t-1}, x_t] + b_i),$$

$$\tilde{C}_t = \tanh(w_c[n_{t-1}, x_t] + b_c),$$

$$Ct = at * Ct{-}1 + it * \tilde{C}t,$$

$$O_t = \sigma(w_o[n_{t-1}, x_t] + b_o),$$

$$n_t = o_t * \tanh(C_t) \qquad (3.2)$$

(W) stands for weight values, (b) is utilized as the basis, and (i_t) is the output values of the input layer. (σ) is the activation function and it is used to send the important data to the next cell. Input data is (x_t), output data is n_t, the cellular cell is (c_t), the output of forget gate is

(a_t), and the output gate is (O_t). LSTM features feedback connections, which set it apart from traditional feed-forward neural networks. In addition to analyzing individual data points, it can also assess the whole data flow[54].

3.3.3 Convolutional Neural-Network, CNN

In the case of a convolutional neural network (CNN), there are three types of layers: input, output, and hidden—convolutional layers. At its very core, these layers progressively change data using hierarchical rules in mathematical models to replicate the structure of the human brain. CNNs can harness deep learning to develop advanced artificial intelligence capabilities analogous to human brain activity in extracting valuable insights from data. CNNs are most effective for image and video identification tasks because they can compute input through numerous levels until it attains a pattern and feature. In convolutional layers, filters identify the edges, textures, and other relevant components in the input data. These are then integrated into the deeper layers to give an in-depth understanding of the input. CNNs work very well for complex tasks that involve deep analysis and identification of patterns due to their hierarchical approach[55].

3.4 PROPOSED HYBRID DEEP-LEARNING-APPROACH

To detect IoT botnet attacks, we are putting forth a hybrid deep learning (DL) strategy that combines LSTM networks and DNN. Hybrid models have demonstrated remarkable efficacy, attaining elevated detection precision within a brief temporal span. With the combination of DNN and LSTM, we hope to improve our detection performance. IoT devices provide enormous volumes of data quickly; as a result, LSTM is selected because it can learn from lengthy data sequences efficiently, while DNN improves the model's speed and efficiency to increase its predictive power. Figure 2 depicts the suggested architecture, while Table 3 offers a thorough breakdown of the hybrid model's structure[56].

3.4.1 Dataset

We used traces of both legitimate and malicious IoT traffic from the N_BaIoT dataset to assess and contrast our suggested method. There are 117 properties in this dataset, which

include a tag and 116 network features. Table 2 lists the six IoT devices in the dataset. It also includes two types of botnet malware, GAFGYT and MIRAI, along with benign traffic.

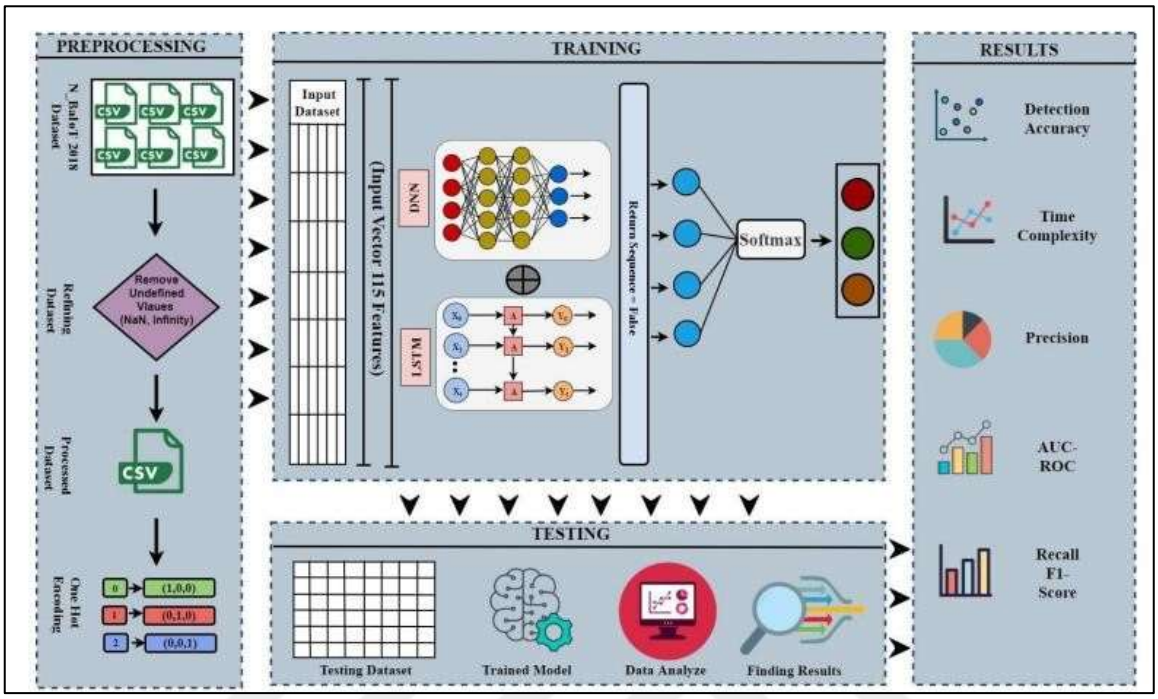

Figure 3.2: Proposed Architecture.

Benign Traffic: This type of traffic is typical for a network.

GAFGYT (sometimes referred to as BASHLITE): A well-known family of Internet of Things botnets with multiple varieties that can download and run malware, execute malicious commands, and launch DDoS assaults in real time. It can initiate TCP flooding, UDP flooding, Junk (sending spam material), Combo (spamming a specified IP address and port), and other attacks[57].

Table 3.2: N_BaloT2018-Dataset.

Devices	Benign	Mirai	Gafgyt
Thermostat	10,00	10,00	10,00
838E Security Camera	10,00	10,00	10,00
737E Security Camera	10,00	10,00	10,00
1011 Web Cam	10,00	-	10,00
1002 Security Camera	10,00	10,00	10,00
1003 Security Camera	10,00	10,00	10,00
Total Records	**60,00**	**50,00**	**60,00**

3.4.2 Detection Methodology

Using deep learning (DL) techniques, our detection methodology leverages the most advanced models in each category to create a highly flexible, reliable, and accurate botnet antimalware strategy. By developing a DL-based system with DNN and LSTM networks, our approach can effectively distinguish between benign and malicious activity.

3.4.3 Technical Details

Table 3 details the specifics of each hybrid method, providing an overview of both the proposed and modern techniques. Through extensive testing, we determined the optimal choices for the optimizer, activation, and loss functions—ReLU, Categorical Cross-Entropy, and Adam, respectively. This iterative process helped us identify the ideal number of layers, neurons, and other key characteristics[58].

Table 3.3: Description of The Algorithm.

Model	Layers	Neurons/ Kernel	$\bar{A}\bar{F}/\bar{L}\bar{F}$	Optimizer	Epochs	Batchsize
DNN - LSTM	DNN Layer (3)	(400,100,50)	RelU / CCE	Adam	5	32
	LSTM Layer(3)	(400,100,50)	-			
	Merge Interface		-			
	Dense Interface	40	-			
	Dense Interface	15	-			
	Output Interface	3	Soft-max			
$CNN2D - LSTM$	CNN2D Layer (3)	(400,100,50)	RelU / CCE	Adam	5	32
	LSTM Interface(3)	(400,100,50)	-			
	Merge Interface Dense Interface	40	- -			
	Dense Interface	15	-			
	Output Interface	3	Soft-max			
$CNN2D - CNN3D$	CNN2D Interface(3)	(400,100,50)	RelU / CCE	Adam	5	32
	CNN3D Interface(3)	(400,100,50)	=			
	Merge Interface		-			
	Dense Interface	40	-			
	Dense Interface	15	-			
	Output Interface	3	Soft-max			

a. Pre-Processing

We carefully pre-process the N_BaIoT dataset in order to optimize our hybrid deep learning approach's efficacy and performance. Our pre-processing procedures guarantee data quality and set up the dataset so that our model may learn from it effectively. This is the method we use:

a. Data Inspection: We load data and inspect it for missing, null, or infinite value information, which could be a problem while training the model.

b. Normalization: We normalize all values between 0 and 1 to standardize data using the Min_Max_Scaler. This will enhance the performance of our deep-learning models and speed up the learning process.

c. Encoding: We encode the target labels through One-Hot Encoding (OHE). This step is a crucial necessity for developing a deep learning system, where category labels are converted to a format understood by the model.

   i. Step 1 (Input): Load N_BaIoT dataset. Contains about 170,000 records of IoT network traffic from six different devices, then load six CSV files containing different device data.

   ii. Step 2 (Refining Dataset): All NaN and infinite values were removed from the dataset to avoid any kind of erroneous occurrences during the training process. We normalized the data using MinMaxScaler to ensure that it is all set for analysis.

   iii. Step 3 (Processed Dataset): After cleaning, all the node information should be combined in one CSV, without NaN and infinite values, for further post-processingp[59]

   iv. Step 4 (One-Hot Encoding): We standardize the target labels into 0, 1, and 2, three categories, by One-Hot Encoding for the performance of the DL system [60].

b. Training

After finishing pre-processing, we go further to train our model using the refined dataset. Here's what we do next:

i. Load Pre-processed Dataset: We start by loading a pre-processed dataset. Out of the 115 features present in the dataset, 90% of them are taken as input to train our algorithm.

ii. Hybrid Deep Learning Phase: In this phase, we use two algorithms—LSTM and DNN—which are run over the data simultaneously and combine the output for better results.

iii. Add Layer: To ensure consistent processing, this layer merges a collection of tensors with similar shapes into a single tensor.

iv. Return Sequence: Here, we specify whether we want the entire output sequence returned as an output or just the final state of the results. We return the final result by default.

SoftMax Function, Step 5: Finally, this function classifies the data into different classes—benign, GAFGYT, MIRAI—by the SoftMax algorithm. Our results section explains how this function helps to minimize prediction errors and increase the detection rate. These are the ways that we will ensure that our hybrid deep learning model may be sufficiently readied for the practical identification and assessment of complex threat patterns in scenarios about the Internet of Things, ensuring high reliability.

```
function Pre-processing(Dâ):
    D = Merge(Dâ)
    D = Convert Datetime to integer
    D = OHE(Labels)
    D = Convert Source IP to integer
    D = Convert Destination IP to integer
    for each Row R in Rows:
        for each Column C in Columns:
            if D[R][C] in ['nan', 'infinity', 'null', ' ', 'NaN']:
                Drop Sample (Row)
    Save D as CSV
end function
```

Figure 3.3: Algorithm 1 Pre-Processing.

c. Testing

It cross-validated 10% of data to validate the proposed technique's effectiveness. With this testing dataset, we tested our hybrid algorithm model for its performance in real-time scenarios after training. The algorithm's output is detailed in the results section following this testing step. We demonstrated the effectiveness of our proposed approach using various state-of-the-art evaluation metrics[61]

3.4.4 Experiment Details

These are:

a. ROC Curve: A graphical view of a binary classification model's diagnostic ability.

b. F1-Score: A test accuracy measurement that combines recall and precision.

c. Accuracy: The number of correct positive observations, divided by the number of all positive observations.

d. Confusion Matrix: This table is frequently used to describe the performance of a classification model on a set of test data for which the actual values are known.

e. Recall: The proportion of true positive observations over the number of positive observations in reality, that have been predicted as such[62].

```
function Pre-processing(D_a)
    D = Merge(D_a)
    D = Convert Datetime to integer
    D = Labels = OHE(Labels)
    D = Convert Source IP to integer
    D = Convert Destination IP to integer
    for R in Rows do
        for C in Columns do
            if D[R][C] in ['nan', 'infinity', 'null', ' ', 'NaN'] then
                Drop Sample (Row)
            end if
        end for
    end for
    Save D as CSV
end function
```

Figure 3.4: Algorithm 2 DNN-LSTM.

Table 3.4: Specification System.

Components	Specification
$\overline{\overline{CPU}}$	Corei7-8750H@2.21GHz
RAM	12 GB
OS	Windows 10

d. ROC Curve

Receiver Operating Characteristic curve shows how well our system can distinguish between true positive and false positive results. Essentially, it helps us see the balance between correctly identifying positives and mistakenly marking negatives as positives. A key measure here is the Area Under the Curve (AUC); the higher the AUC, the better our system performs. In Figure 6, you can see the effectiveness of our method as represented by this ROC curve.

e. Accuracy

The percentage of correctly categorized connections—both normal and intrusive—using the suggested methodology is measured as accuracy. The accuracy formula is:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.3)$$

This metric provides an overall assessment of the model's ability to correctly categorize network connections.

f. Precision

An essential statistic to take into account when assessing our suggested model is precision. Out of all the positive detections produced, it calculates the percentage of accurate positive detections. In essence, accuracy indicates the proportion of identified positives that are real positives. The precision calculation formula is:

$$Precision = \frac{TP}{TP+FP} \quad (3.4)$$

g. Re-Call

Re-call is the proportion of correctly Known positive test to the actual malware samples. It measures how well the model identifies all relevant instances in the dataset. You can measure recall by using the following formula:

TP

$$\text{Recall} = \frac{}{} \tag{3.5}$$

$TP+FN$

F1 _ score

The F1-score,

or F-score, is a metric used to innovate a model's performance on a given dataset. It balances both precision (the accuracy of positive predictions) and recall (the ability to find all positive instances), providing a single value that reflects the model's performance considering both false positives and false negatives. The formula to Measure the F1-score is[63]**:**

TP

$$\text{F} - \text{score} = \frac{}{} \tag{3.6}$$

$2 * TP+FP+FN$

4. EXPERIMENTAL RESULTS

4.1 INTRUSION DETECTION DOMAIN IN NETWORK SECURITY

Monitoring systems for indications of invasions or improper use is known as intrusion detection. To find security policy infractions, intrusion detection systems (IDS) scan network traffic or audit records. A record of user access would be an illustration of an audit trail. Prior to the creation of contemporary intrusion detection systems, anomaly detection was done manually.

In recent years, the rapid advancement of computer systems has profoundly transformed our daily lives, making us more dependent on technology than ever before. According to the Cisco Visual Networking Index 2017, it was projected that by 2021 there would be 3.5 computer devices per person worldwide, with global Internet traffic reaching nearly 106 terabytes per second. This swift technological growth has, unfortunately, also increased the vulnerability of computer systems to various threats.

Despite ongoing research and technological advancements, achieving absolute cybersecurity remains a formidable challenge. Intrusion Detection Systems (IDS) play a vital role in this landscape by monitoring network traffic, analyzing it for anomalies or unauthorized access, and sometimes responding to these intrusions to protect the network. However, current IDS methods have limitations, such as excessive resource usage, delays in response times, and difficulties in reliably safeguarding systems.

In this chapter, we tackle the limitations of traditional IDS methods by introducing an off-policy Q-learning intrusion response model. Conventional IDS techniques continuously monitor network traffic, which can lead to resource exhaustion even when no attacks are occurring. Furthermore, the time required to respond to detected attacks is often considerable, giving intruders the chance to modify or terminate their actions before detection.

Regular updates to protection mechanisms are crucial because attackers can exploit known vulnerabilities to launch subsequent attacks. Traditional machine learning models, such as Naive Bayes and Support Vector Machines, typically focus on categorizing network

characteristics. However, these models may struggle with noisy, heterogeneous data, particularly in large networks where critical information is concentrated in a few features. Our proposed off-policy Q-learning model aims to address these challenges by optimizing resource usage and response times, enhancing the overall effectiveness and reliability of IDS in dynamic network environments.

This work aims to guide readers in discovering efficient defense mechanisms to protect their network systems. With the rise of wireless technology, networks have become more vulnerable, making traditional security measures like firewalls inadequate. Cybercriminals continually develop new techniques, necessitating dynamic and adaptive defense solutions.

we present an automated ML technique for Network Intrusion Detection and Prevention Systems (IDPS:

First Layer: Utilizes machine learning techniques for dimension reduction and classification to monitor the network and identify breaches.

Second Layer: Based on the classification results, employs a Q-learning method to prevent and counteract intrusions, providing an automatic response to attacks.

This IDPS is specifically designed for streaming data and can be integrated with any firewall for online data analysis.

4.2 INTRUSION DETECTION SYSTEMS

By incorporating an intrusion response mechanism alongside IDS, we provide comprehensive protection that not only secures systems but also enables automated decision-making in unknown environments. This approach has significant implications beyond cybersecurity, especially in the healthcare sector, where protecting sensitive patient data is critical. As healthcare systems increasingly rely on interconnected devices and digital records, ensuring robust cybersecurity measures is essential to prevent breaches that could compromise patient confidentiality and safety.

The advanced machine learning techniques for feature reduction and classification in network security are designed to outperform current methods and effectively combat a

variety of attacks. The innovative representation of the computer network as a probabilistic structure allows the decision agent to learn and select optimal policies based on feedback, ensuring continuous network protection.

4.3 FUNCTION OF INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems (IDSs) are automated systems designed to identify and alert users when an intrusion has occurred or is likely to occur. According to the Common Intrusion Detection Framework (CIDF), IDSs typically consist of four components: sensors, analyzers, databases, and reaction units. To enhance their reliability, most modern IDSs use multiple intrusion sensors that collect signals from a wide computational environment. Here are the primary functions performed by intrusion detection systems:

a. Keeping an eye on system activities.
b. Examining the setup of the system.
c. Examining the information files.
d. Identifying known attacks.
e. Recognizing unusual behavior.
f. Taking care of audit data.
g. Emphasizing regular activities.
h. Fixing mistakes in system settings.
i. Holds data on trespassers.

4.4 TYPES OF IDS

Intrusion Detection Systems come in five different varieties. These are the following:

a. IDS for Statistical Anomaly

b. IDS Based on Signatures

c. IDS Based on Hosts

d. IDS based on networks

e. Mixture IDS

4.5 STATISTICAL ANOMALY IDS

IDSs with statistical anomaly bases dynamically contrast usage trends with ingrained usual usage patterns. These patterns include those related to network packet types, memory consumption, and CPU utilization. This kind of intrusion detection system employs a methodology that use statistical approaches to find intrusions and assaults. Setting baseline statistical behavior is the first step in this process. After then, they collect fresh statistical information and calculate the departure from the mean. When a threshold is surpassed, sound an alert.

Pros and cons of Statistical Anomaly IDS:

a. It is independent of particular operating systems

b. It can identify abuse-of-privileges attacks

c. It can dynamically adapt to new vulnerabilities

The following are some drawbacks of statistical anomaly detection systems: high false alarm rates; attacks must drastically alter functioning parameters to be detected; and consistent user behavior is required.

4.6 IDS WITH SIGNATURES

Attack signature databases are used by signature-based intrusion detection systems to compare event data to attack characteristics. Attacks can be identified by their signatures. If information from packet monitoring and audit logs matches database signatures, responses are started. This kind of IDS is quite accurate since it has a low false positive rate.

The following are some drawbacks of Signature Based IDS:

a. It can only identify assaults that are kept in databases;

b. It can fail to identify new, distinct, and original attacks.

4.7 IDS NETWORK BASED

Network intrusion detection systems (IDSs) track activity on individual network segments and check packets for header condition, port, or string signature matches. Typically, network appliances with NICs running in promiscuous mode are used as network IDS. It is distinguished by the sparing of resources used. The components consist of an IDS management station, where warnings are received and the program is managed, and network-based intrusion detection software that runs on a dedicated host and is linked to network traffic via a network interface. Installing it in switched situations is difficult. An additional application for inline NIDS is as a bridge between two switches or between a router and switch.

4.8 IDS HOST BASED

Intelligent agents are used by host-based IDSs to monitor host computer events only for improper behavior. It keeps an eye on modifications to important system files, user privilege changes, and access to them. It is superior than NIDS in detecting trusted insider assaults. It can identify external assaults with a fair degree of effectiveness. It may be set up to keep an eye on connection attempts, network packets, and login attempts. Host-based intrusion detection systems and database servers, are built as host-based programs that operate in the background on deemed sensitive or important hosts.

4.9 IOT-FLOW: USING FLOW DATA FOR MACHINE LEARNING IDS FOR INTERNET OF

Items There are many obstacles to deploying an AI-driven intrusion detection system (IDS) in Internet of Things contexts. These include creating efficient methods for detecting anomalies and intrusions, placing IDS at the best location while guaranteeing scalability, and validating proposals using representative datasets. The implementation of IDS systems is further complicated by the amounts of heterogeneous generated data by smart cities.

4.9.1 IoT-Flow IDS for Smart Cities

Our suggested IoT-Flow IDS has a distributed architecture that is scalable and able to effectively manage a variety of IoT technologies in order to meet the complexity of smart city settings. Rather than examining each packet's payload, the IDS analyzes traffic statistics in order to provide traffic flow information. In Internet of Things contexts, this flow-based method improves scalability and interoperability.

4.9.2 Flow-Based IDS Architecture

A flow-based Intrusion Detection System (IDS) operates through several key steps to ensure network security:

a. Packet Observation: The initial step involves capturing packet from the network and preprocessing it. This stage is critical for gathering the raw data needed for further analysis.
b. Flow Metering & Export: In this step, captured packets are aggregated into flows, and the flow records are subsequently exported to a collector. This transformation of packet data into manageable flow records is crucial for efficient processing, ensuring that the system can handle large volumes of network traffic more effectively.
c. Data Collection: In this step, the collector receives, stores, and pre-processes the flow records from the exporters. Ensuring accurate and efficient data collection is vital for maintaining the integrity and reliability of the Intrusion Detection System (IDS).
d. Data Analysis: The final step involves analyzing the collected flow data. This analysis can be applied in three main areas:
    i. Anomaly Detection: Identifying deviations from normal traffic patterns which may indicate potential security threats.
    ii. Intrusion Detection: Detecting specific patterns that match known intrusion signatures.
    iii. Behavioral Analysis: Understanding and profiling typical network behaviors to detect irregular activities.

4.9.3 Advantages of Flow-Based IDS in IoT

Using a flow-based approach for Intrusion Detection Systems (IDS) in Internet of Things (IoT) environments offers several key benefits:

a. Scalability: Flow-based IDS can handle large volumes of traffic data more efficiently than traditional packet-based IDS, making it suitable for extensive IoT networks.
b. Interoperability: By focusing on flow information rather than packet payloads, flow-based IDS can operate across various IoT technologies and platforms, ensuring seamless integration.
c. Efficiency: Flow aggregation significantly reduces the amount of data that needs to be processed, thereby enhancing the overall efficiency of the Intrusion Detection System (IDS). This improvement allows for faster detection of potential threats, ensuring a more responsive and robust security posture.

4.9.4 Challenges and Considerations

Implementing a flow-based Intrusion Detection System (IDS) requires careful consideration of several factors:

a. Flow Aggregation and Export: Transforming packet data into flows involves complex aggregation and export processes, which are critical for accurate data analysis.
b. Representative Datasets: Validating the effectiveness of an IDS necessitates using datasets that accurately represent the diverse and dynamic nature of IoT traffic. These datasets should mirror real-world conditions to ensure the IDS can handle various scenarios.
c. Scalability: Ensuring the IDS can scale to accommodate the large volumes of data generated in smart city environments is essential. The system must be capable of efficiently processing and analyzing significant amounts of traffic data without compromising performance.

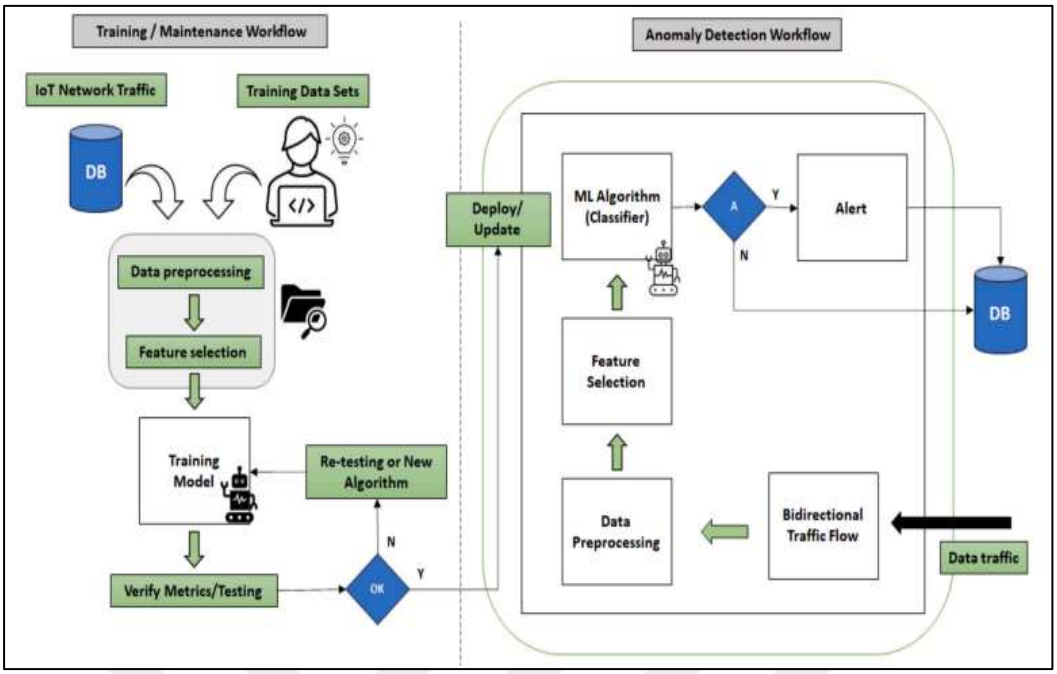

Figure 4.1: Anomaly Detection Workflow Overview.

4.9.5 Workflow overview

The main phases of our solution's workflow in the model training and production environment are depicted in Figure 7. The model selection and training procedures are shown by the left side of the image. Here, datasets that faithfully replicate the data flows characteristic of IoT settings in smart cities are used to train and fine-tune the models. The models that demonstrate the best combined accuracy and recall, as measured by performance criteria, are subsequently implemented in the real-world setting.

Figure 7 illustrates the procedure for anomaly identification in a production environment, shown on the right side. Just like the initial stage of converting traffic data into flows, sensors play a crucial role in gathering information from network packets.

The next stage in transforming traffic data is to aggregate packets into flows, depicted as the Bidirectional Traffic Flow step in Figure 7. Traffic flows in communication networks consist of various specified properties and data. The creation of the IP Flow Information Export (IPFIX) IETF protocol established a universal standard for exporting IP flow data from probes and network devices.

IPFIX provides network managers with a comprehensive view of all incoming and outgoing traffic, enabling services such as measurement, accounting, and invoicing through the transfer of information elements (IE). These elements are categorized into 12 groups based on their semantics and applicability, including identifiers, flow properties, and timestamps. They provide detailed information, such as source and destination IP addresses, ports, packet counts, and protocol types. By utilizing IPFIX IEs, network administrators can gain valuable insights into various aspects of network traffic, from identifying the origin of network flows to understanding their characteristics.

Network sensors equipped with flowmeters capture and export traffic flows into IPFIX records. Flowmeters, such as Yet Another Flowmeter (YAF), are instrumental in this process. Once the traffic data is converted into flows, the data pre-processing stage begins. This stage includes tasks such as resolving missing values and encoding categorical information, ensuring the data is prepared for further analysis.

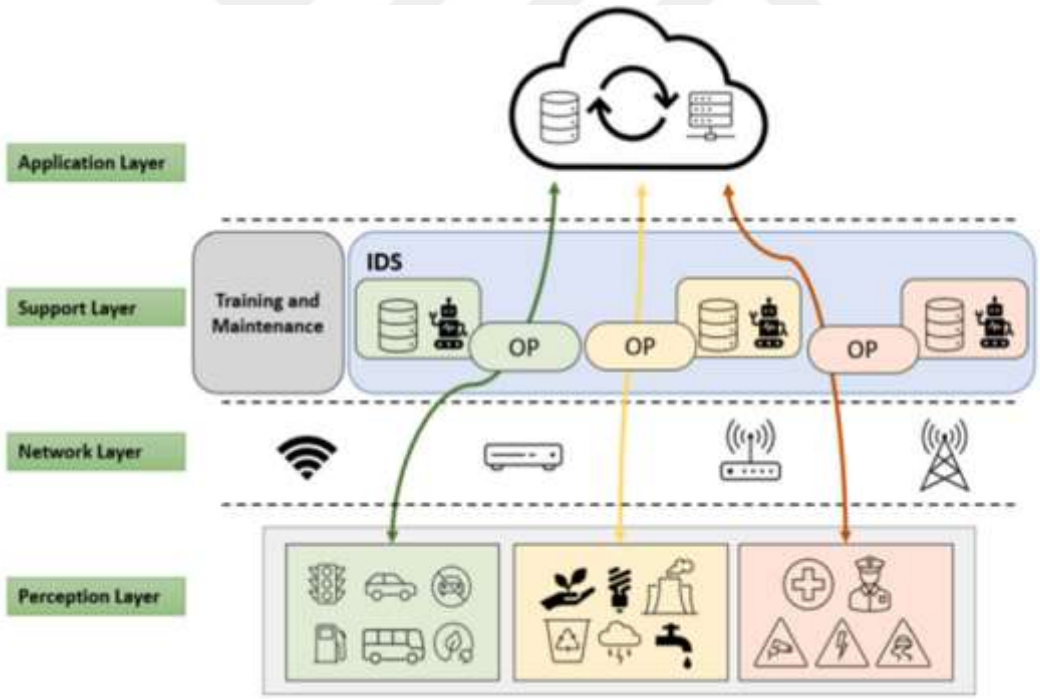

Figure 4.2: IoT-Flow IDS Architecture Proposal.

Subsequently, relevant features for anomaly identification are selected by the machine learning model (classifier) based on their impact on the target attribute. This feature selection process is critical in enhancing the accuracy of anomaly detection within the system.

4.10 IOT-FLOW IDS DISTRIBUTED ARCHITECTURE.

To fully harness the potential of a machine learning-powered IoT intrusion detection system (IDS), it is crucial to seek a computational solution that ensures sufficient processing capacity for data. This solution must not only meet the unique service needs of IoT devices but also address their processing limitations. Our approach leverages fog computing, which offers Fog applications valuable network context information due to its edge placement. This setup provides insights into client status, traffic data, and local network conditions, enhancing the overall effectiveness and responsiveness of the IDS.

The formation of network nodes referred to as Fog Nodes is made possible by fog computing's innate location awareness. These nodes, which represent every attribute of fog computing, function as localized centers for resources and network services. A smart city's architectural layers and the various requirements arising from its assortment of services must be carefully taken into account when integrating a cybersecurity solution into the IoT ecosystem of the city.

Figure 8 illustrates the distributed architecture of our solution, starting with the perception layer that houses the sensors. In this layer, we segment IoT network traffic flows based on service requirements, criticality, or application messaging protocols like MQTT or CoAP. Network segmentation serves the distinct needs of each service and also helps create observation points (OP) for each segment or service. This segmentation facilitates streamlined network analysis and a deeper understanding of network behaviors.

In order to transfer data to the support layer, we use a variety of technologies at the network layer, including Ethernet. Device gateways are pointed at the support layer node, which serves as a service provider for data prior to it moving up to the application layer.

Within the support layer, our infrastructure is built on Fog computing principles, ensuring ample processing and storage capacity resources. This layer serves as the conduit for all

traffic originating from the perception layer or the application layer, effectively managing and processing the flow of data throughout the system.

4.11 SPECIALIZED COMPONENTS

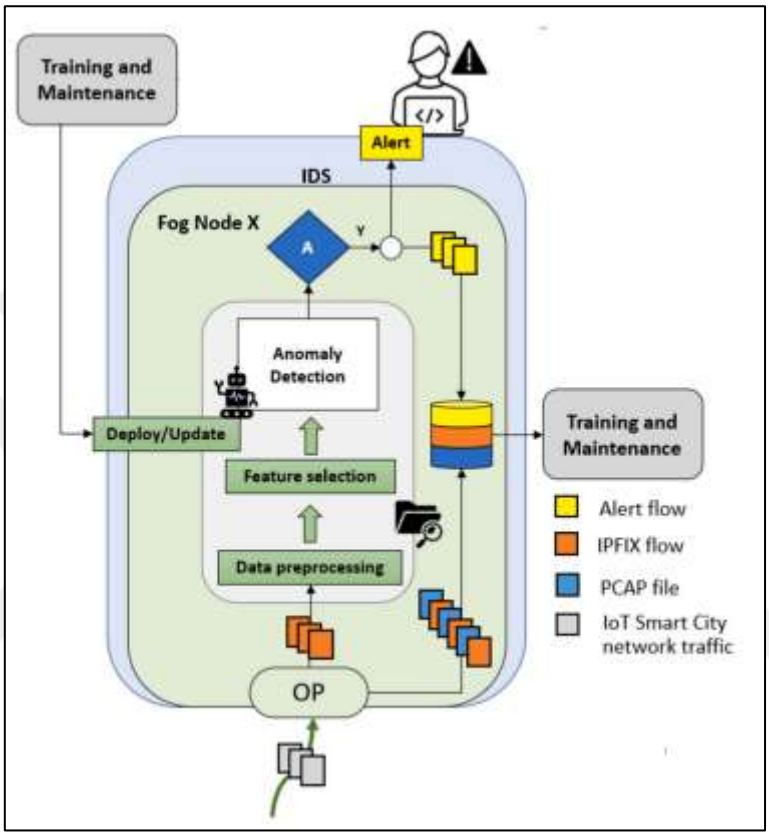

Figure 4.3: Fog Node with Learning-Based IDS.

Fog nodes, exemplified in Fig. 4.4, play a pivotal role as network components responsible for various tasks within the IoT network. These tasks include data observation, storage, transformation, classification, and forwarding. Network traffic undergoes transformation into standardized flows, such as IPFIX, passing through a preprocessing module before reaching the classification model. Regardless of the classification outcome, the data is archived alongside the originating flows and packets.

When employing supervised learning, it's imperative to train the learning module to discern typical network behavior from anomalies. Model training can utilize datasets crafted and

labeled within controlled lab settings or via pre-production networks resembling those in smart cities.

High-performing machine learning models are found in this module and then merged into the production IDS module. A database containing the flow representations (such as IPFIX) and flow classifications of collected network traffic (PCAP files) is kept in the production environment. As a result, the production IDS nodes send tagged, real-world data back to the training module, and the training module feeds updated models into it. This labeled data from the real world is frequently obtained from historical records, and network audits and forensic analysis may validate traffic that was previously reported as harmful.

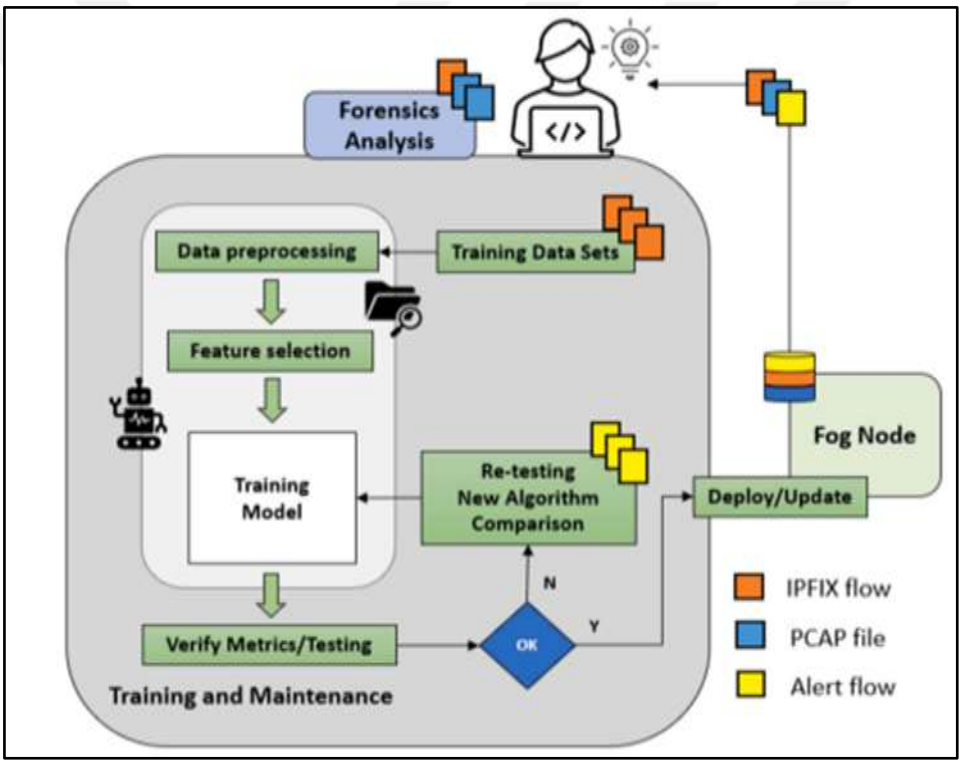

Figure 4.4: Support Layer Training/Maintenance Module.

4.12 CLASSIFICATION ALGORITHMS

The "no free lunch" (NFL) theory was first proposed by Wolpert et al. and highlights the need for experimentation with different machine learning classifiers in order to accomplish classification jobs efficiently. "There is no single learning algorithm that universally performs best across all domains," according to the theory. As such, testing several

classifiers for certain domain issues is crucial. This relates to issues with intrusion detection and categorization in our setting.

Ensembles and single classifiers are the two types of classification methods that we concentrate on. We choose well-researched techniques like Extreme Gradient Boosting (XGBoost), Gradient Boosted Machine (GBM), Random Forest (RF), AdaBoost (AB), and Extremely Randomized Trees (ERT) among ensembles. Numerous factors went into selecting these algorithms. The first drawback of ensemble-based techniques is that a high number of input characteristics may cause overfitting. We also include single classifiers in our study, such as Multi-layer Perceptrons (MLP) and Classification and Regression Trees (CART), to address this.

4.12.1 Ensembles of Classifiers

A basic overview of several classifier ensembles is given in this section. In the literature, ensembles have been shown to be effective algorithms for both classification and regression. Thus, in this analytical work, we have employed five distinct ensembles.

4.12.2 Forest Random (RF)

As an ensemble learning technique, Random Forest (RF) builds many decision trees (predictors) $\{t(x_{in}, \theta_n), n = 1,...\}$, each of which makes a unique prediction based on an input x_{in}. A random collection of variables $\{n\}$, separately sampled with the same distribution, forms the basis for each predictor. The fundamental tenet of RF is that accuracy may be increased and overfitting risk can be decreased by integrating the predictions of many trees.

No tree in RF is ever trimmed; they are all left to grow to their full potential. They vote on the most popular class for a certain input xin after a huge number of trees are formed, which enables them to jointly make predictions on the input data. Relevant research indicates that the maximum depth for tree creation should be limited to 26, and the number of estimators (trees) for performance evaluation should typically be set at 500. Other factors are adjusted using the randomized search.

4.12.3 AdaBoost (AB)

The adaptive meta-estimator known as AdaBoost (AB) begins by using the original dataset to determine the first training weights. These weights are used as input for the classifier's subsequent iterations, which concentrate on cases that were previously misclassified. The categorization of these challenging situations is improved by later classifiers by modifying their weights. By boosting weak learners and eventually convergently learning to a strong learner. A boosted classifier is represented by equation (1), where (x) is an input object and (cp) is a weak learner.

$$Cp(x) = \sum_{p=1}^{P} cp(x) \tag{4.1}$$

Cp(x) in AdaBoost yields a value that represents the expected class. In the training set, each cp produces an output hypothesis (h(xi) for every occurrence. In this case, the weak learner added to the final model is denoted by cp(x) = βph(xi), the error function to be minimized is represented by E(C), and the BC constructed from the last phase is represented by Cp-1(xi). For AB, 50 estimators and a learning rate of 0.1 are the ideal specifications.$Et = \sum_i E[Cp - 1(xi) + \beta\text{ph}(xi)]$

4.12.4 Gradient Boosted Machine (GBM)

An ensemble technique called Gradient Boosting Machine (GBM)[27, 26] is intended to improve decision tree (DT) performance. It works similarly to other boosting approaches, maximizing an arbitrary differentiable loss function as it successively combines weak classifiers (DTs) to produce a strong prediction model. To lower prediction errors, each tree in the series improves on the forecasts made by the trees before it. Formally, we may think of a collection of random variables, x, which are the input variables, and z, which are the output variables.

$$x = \{x_1, x_2, \ldots . x_n\}$$

Using training data, our goal is developing an estimate A (approximation) that maps X to Z where: $\{z, xi\}_{i=1}^{N}$.

$$S = \{(x_i, z_i)\}(|D| = p, x_i \in R^q , zi \in R)$$

$$z\hat{}i = \varphi(xi) = \sum_{m=i}^{M} fm(xi), fk \in A$$

$$A = \{f(x) = w_r(x)(r : R^m \rightarrow K, w \in R^K)\} \quad (4.2)$$

In decision trees (DT), fm is a single tree with structure r and leaf weight w; K is the total number of trees. The instance set is represented by A, and the tree structure that links an instance with its corresponding leaf index is given by $\boldsymbol{U}$. The final prediction is determined by the tree ensemble by summing together the scores (w) of the leaves discovered during the classification of a specific test sample.

4.12.5 Trees With Extreme Randomization (ETC)

Extra Trees, or Extremely Randomized Trees (ETC), are an approach for supervised regression and classification. Regardless of the goal variable, ETC randomly chooses features and cut-points to create an ensemble of unpruned decision trees.

4.12.6 Key Characteristics of ETC:

1. In Extremely Randomized Trees (ETC), A subset of features is randomly selected at each tree node. From this subset, the best cut-point is chosen. In some instances, a single feature and cut-point may be selected at each node. Due to this randomization, the trees are entirely independent of the target attribute values of the training samples.

2. Ensemble Construction: A traditional top-down approach is used by ETC to build the ensemble. To reduce bias and variance, ETC employs the whole training sample instead of bootstrap replicas, as opposed to other tree-based ensemble algorithms.

4.12.7 Dividend Process for Numerical Elements

The Extra-Trees Classifier (ETC) splitting procedure involves three critical parameters:

a. K (Number of Features Selected at Each Node): This parameter affects the robustness of the feature selection process by specifying the number of features to be considered at each split.
b. nmin (Minimum Training Set Size to Split a Node): This parameter helps control the averaging of output noise by setting the minimum number of samples required to split a node.
c. Tcount (Total Number of Trees in the Ensemble): This parameter reduces variance by defining the total number of trees included in the ensemble.
d. ETC typically performs on par with Random Forest (RF) but has the advantage of being computationally faster due to its efficient feature selection process.

4.12.8 Hyperparameters

The hyperparameters for ETC, obtained through randomized search, are:

a. Estimators: 1788

b. Maximum Tree Depth: 10

c. Number of Features Considered for Best Split: $\log_{10}(2)$

d. Criterion: Gini

e. Bootstrapping: Not used

4.12.9 Extreme Gradient Boosting (XGB)

XGB is an advanced and highly optimized version of the Gradient Boosting Machine (GBM) algorithm. Developed by Tianqi Chen and collaborators, XGB is widely recognized for its exceptional performance and versatility in a range of machine learning tasks, particularly in classification and regression problems.

4.12.10 Important Distinctions From GBM

One of the key differences between XGB and GBM lies in their regularization strategies. XGB employs a more advanced and effective regularization technique to control over-fitting

and improve the model's generalization ability. This is crucial for ensuring the model's robustness and preventing it from learning noise in the training data.

Regularization Parameter (ζ): In XGB, the regularization parameter, represented by ζ, plays an essential role in managing the model's complexity. It strikes a balance between avoiding over-complexity, which can lead to poor generalization on unknown data, and effectively fitting the training data. Additionally, the regularization term λ aids in controlling model complexity.

4.12.11 Loss Function and Optimization

Similar to GBM, XGB optimizes the loss function during model training using gradient boosting. For binary classification problems, the LogLoss function, also known as cross-entropy loss, is frequently used. This loss function penalizes incorrect predictions more harshly, particularly when the predicted probability significantly differs from the actual class label.

The LogLoss function L for binary classification is defined as:

$$\zeta = \gamma T_1 + 1\,2\,\lambda \sum_{j=1}^{T} w_j^2$$

$$L = -\frac{1}{N}\sum_{j=1}^{T}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \quad (4.3)$$

where:

N: Represents the total number of observations.

y_i: For observation i, y_i is the binary indicator (0 or 1) indicating whether the predicted class c is accurate.

p_i: The expected probability that observation i belongs to class c is represented by p_i.

4.12.12 Major Implementation Enhancements

Sparse Matrices (DMatrix): XGB introduces sparsity-aware algorithms to efficiently handle sparse matrices (DMatrix). This optimization significantly boosts computational efficiency and memory usage, especially for sparse and high-dimensional datasets.

Improved Data Structures: XGB incorporates enhanced data structures and algorithms, such as histogram-based splitting, to facilitate faster training and prediction times. These improvements contribute to the algorithm's overall performance and scalability.

Parallelization Support: XGB is designed to leverage parallel processing capabilities, allowing it to efficiently utilize multi-core CPUs and distributed computing environments. This parallelization support enhances training speed and scalability for large datasets.

4.12.13 Optimal Hyperparameters

The optimal hyperparameters for XGB are typically determined through hyperparameter tuning techniques such as randomized search or grid search. These hyperparameters include:

a. Number of Estimators (Trees): 100

b. Maximum Tree Depth: 8

c. Minimum Child Weight: 1

d. Booster Type: gbtree

e. Minimum Loss Reduction: 2

f. Sub-Sample Ratio: 0.6

5. EXPERIMENTAL CONFIGURATION

5.1 DESIGN OF EXPERIMENT

The performance evaluation was conducted on a computer running 64-bit Windows 10 Pro, equipped with an Intel® i7-7700 quad-core CPU clocked at 3.60 GHz and 16GB of RAM. Python (version 3.6.1) was used to implement the classifiers.

For parameter hyper-tuning, the evaluation was carried out on the PARAM Shavak system, which runs 64-bit Ubuntu 14.04 and is powered by an Intel® Xeon® Gold 6132 twenty-eight-core CPU with a clock speed of 2.6 GHz and 96GB of RAM.

To assess the classifiers' response times, a Raspberry Pi 3 Model B+ running Raspbian OS with a 64-bit quad-core ARM CPU clocked at 1.4 GHz and 1GB of RAM was used. The classifiers were implemented using the popular machine learning library, scikit-learn.

The STAC online platform application was utilized to perform statistical tests on the performance outcomes.

5.2 DATASETS

Three datasets are utilized in this study: NSL-KDD, UNSW-NB15, and CIDDS-001. The selection of CIDDS-001 and UNSW-NB15 was based on the fact that they are newly created datasets with actual traffic data, which makes them useful for developing precise intrusion detection systems (IDS) that monitor and identify novel forms of denial of service assaults in IoT networks.

Recently released to aid the development of anomaly-based intrusion detection systems, the CIDDS-001 dataset includes over 32 million records of both malicious and legitimate traffic. It contains twelve traits and two labeling attributes. For our study, we selected a random sample of 100,000 instances from the internal server traffic data, comprising 20,000 records of DoS assaults and 80,000 records of normal server traffic. This sample was used for hold-out and cross-fold validation testing of classifiers. In our previous study, we evaluated several machine learning classification methods using the CIDDS-001 dataset.

The recently released UNSW-NB15 dataset, which includes 49 characteristics and 1 class attribute, was also utilized in our experiments. The dataset is divided into two sets: UNSW-NB15 Train (175,341 instances) and UNSW-NB15 Test (82,332 instances). The train set contains 56,000 normal traffic instances and 119,341 attack traffic instances, while the test set includes 37,000 normal traffic instances and 45,332 attack traffic instances. Cross-fold validation was performed using only the train set, whereas hold-out validation used both the train and test sets.

Additionally, classifiers were validated using the NSL-KDD dataset, which features one class attribute and forty-one features. The KDDTrain+ set comprises 25,192 instances, with 13,499 attack traffic instances and 11,743 normal traffic instances. The KDDTest+ set consists of 22,544 instances, with 9,711 attack traffic instances and 12,833 normal traffic instances. To avoid random sampling from the entire NSL-KDD dataset, hold-out and cross-fold validation were applied to each dataset separately.

5.3 VALIDATION METHODS AND EVALUATION METRICS

The overall performance of classifiers is greatly influenced by the selection of input parameter values. To find the optimal input parameters for Random Forest (RF), AdaBoost (AB), XGBoost (XGB), Gradient Boosting Machine (GBM), and Extremely Randomized Trees (ETC) across different datasets, we employ a random search strategy. The RandomizedSearchCV implementation from the Python scikit-learn package is used to fine-tune hyperparameters. RandomizedSearchCV identifies the optimal parameter settings by performing a cross-validated search across all potential parameter values specified by the user.

To evaluate the performance of the classifiers, we used several well-known metrics, including:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{Specificity} = \frac{TN}{FP+TN} \tag{5.1}$$

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{F P R} = \frac{FP}{TN+FP}$$

$$\text{AUC} = \int_0^1 \frac{TP}{TP+FN} \tag{5.2}$$

a. True Positives (TP): The number of accurately categorized attack instances.
b. True Negatives (TN): The number of correctly classified normal instances.
c. False Positives (FP): The number of normal instances incorrectly classified as attacks.
d. False Negatives (FN): The number of attack instances incorrectly classified as normal.

In order to do a thorough evaluation of the various classifiers, we ran experiments with repeated hold-out and repeated k-fold cross-validation (10-fold). The validation approach's variance is reduced and error estimation is stabilized by successive versions.

The dataset was divided 60:40 for hold-out validation, with 60% going toward training and 40% toward testing. We chose k=10 for k-fold cross-validation, doing 100 iterations of hold-out and 10-fold validation to make sure the classification models are stable—that is, consistently generating predictions for the same test data. In order to evaluate classifier performance while reducing the impact of instance sampling that occurs during hold-out validation, 10-fold cross-validation is utilized.

All performance data are presented as mean values from ten rounds of each repeated validation procedure to prevent bias. To assure impartial findings, each experiment was repeated with a distinct set of seeds (inputs to a random number generator).

5.4 STATISTICAL SIGNIFICANCE TESTS

Comparing different algorithms across various datasets is a critical challenge in machine learning research. An algorithm that performs exceptionally well on one dataset may not deliver comparable results on another due to factors such as the presence of outliers, the distribution of features, or inherent algorithmic properties. This unpredictability complicates the algorithm selection process, making it difficult to determine which algorithm is superior.

To address this issue, statistical evaluations are necessary to thoroughly assess performance outcomes.

In this study, we use two statistical significance tests, the Friedman test and the Nemenyi test, to compare classifiers appropriately. These tests help determine whether there are substantial differences in the performances of the classifiers. The null hypothesis (H0) asserts that there is no performance difference between the classifiers, while the alternative hypothesis (H1) claims that at least one classifier performs notably differently from at least one other classifier.

We employed the Friedman test because it is ideal for comparing more than five entities. This test assesses whether at least one classifier significantly outperforms the others across all datasets. If such a classifier is identified, pairwise multiple comparisons are conducted using the Nemenyi post-hoc test. The post-hoc test is crucial for pinpointing specific differences in the classifiers' performances. In summary, the Friedman test looks for any meaningful differences between the classifiers, while the Nemenyi test identifies the specific pairs of classifiers where these differences occur.

Let d be the number of datasets and k be the number of classifiers for the statistical analysis. The classifiers' performance results (X_{ij}) are first rated ($R(X_{ij})$) for every dataset in the Friedman test. Next, for each classifier, the total of these rankings ($R(X_{ij})$) is calculated to give R_j, where j = 1, 2,..., k. Equation (F-" Statistic "), which is used to determine Q, is used to construct the Friedman statistic (F-Statistic).

$$R_j = \sum_{i=1}^{d} R(X_{ij})$$

$$F - \text{ Statistic } = \frac{(d-1)Q}{d(k-1)-Q}$$

$$Q = \frac{12}{dk(k+1)} \sum_{j=1}^{k} \left(R_j - \frac{d(k+1)}{2} \right)^2 \qquad (5.3)$$

With degrees of freedom $f_1 = k - 1$ and $f_2 = (d - 1)(k - 1)$, the F-Statistic is evaluated against the F-quantiles for a certain significance level (α), where α is the selected significance level. The values of d and k in this investigation are 4 and 7, respectively. For every pair of classifiers, the Nemenyi post-hoc test is performed by computing the test statistic γ_{xy}, where $\overline{R_x}$ and $\overline{R_y}$stand for the mean ranks of classifiers x and y, respectively, over all datasets. A substantial difference between classifiers x and y at the specified α significance level is shown by pairs with γxy values surpassing a crucial threshold. Two significance thresholds are taken into account in this study: α = 0.05 and α = 0.1. The 10-fold (10f) validation findings and holdout outcomes are included in the statistical analysis.

$$\gamma_{xy} = \frac{\overline{R_x} - \overline{R_y}}{\sqrt{\frac{k(k+1)\overline{R_J} = \frac{1}{d}\sum_{i=1}^{d} R(X_{ij})}{6d}}} \tag{5.4}$$

5.5 RESULTS AND ANALYSIS

Evaluation of Classifiers' Performance for Intrusion Detection

The performance analysis of single classifiers (CART and MLP) and ensemble approaches (RF, AB, GBM, XGB, and ETC) tailored to the CIDDS-001, UNSW-NB15, and NSL-KDD datasets is covered in detail in this part. To illustrate the usefulness of these classifiers for intrusion detection in Internet of Things applications, the results are contrasted and statistically assessed.

5.5.1 Hold-out Validation Performance

First, we apply hold-out validation to the performance outcomes analysis. The average values of important metrics obtained from the CIDDS-001, UNSW-NB15, KDDTrain+, and KDDTest+ datasets—aside from FPR—are shown in Figure 12.

a. Accuracy: RF achieves the highest accuracy at 94.94%.

b. Specificity: RF also leads in specificity with 91.6%.

c. Sensitivity: GBM excels with a sensitivity of 99.53%.

d. AUC: XGB performs best in terms of AUC, achieving 98.76%.

On the other hand, AB has the lowest specificity (86.72%), sensitivity (97.94%), and AUC value (94.01%) while MLP has the lowest accuracy (82.76%).

5.5.2 False Positive Rate (FPR) Analysis

The average FPR values of the classifiers using hold-out validation are shown in Figure 13 for the four datasets. With an FPR of 8.89%, RF performs better than other classifiers, although AB has the greatest FPR of 13.26%.

5.5.3 Model Building Time (MBT)

The model building time (MBT) for each of the four datasets with hold-out validation is listed for the various classifiers in Table 1. Because it affects resource usage—a critical factor for devices with limited resources—the MBT is essential. As a result, MBT aids in balancing IDS classification performance and resource utilization.

a. RF and CART: Training on all four datasets takes about two seconds.

b. On the KDDTrain+ dataset, GBM and ETC have the longest training durations.

Only during hold-out validation is the MBT computed, which offers information on each classifier's resource efficiency.

5.5.4 Statistical Analysis

The Friedman test and the Nemenyi post-hoc test were utilized to statistically validate the performance comparisons. While the Nemenyi test finds particular classifier pairs with significant performance differences, the Friedman test looks for generally significant differences among the classifiers.

a. Friedman Test: Degrees of freedom are $f_1 = k - 1$ and $f_2 = (d - 1)(k - 1)$, with d = 4 datasets and k = 7 classifiers.

b. Nemenyi Test: Test statistic γ_xy is calculated for all classifier pairs. Pairs exceeding the critical value indicate significant differences at α levels of 0.05 and 0.1.

The outcomes of 10-fold cross-validation (10f) and hold-out are both statistically examined. Error estimate is stabilized and variation is reduced by frequent validation. The datasets used in hold-out validation are divided 60:40 between training and testing. To ensure consistent and objective findings, 100 iterations of validation are performed in 10f validation, with k set to 10. To prevent biased results, different seeds are utilized for every iteration.

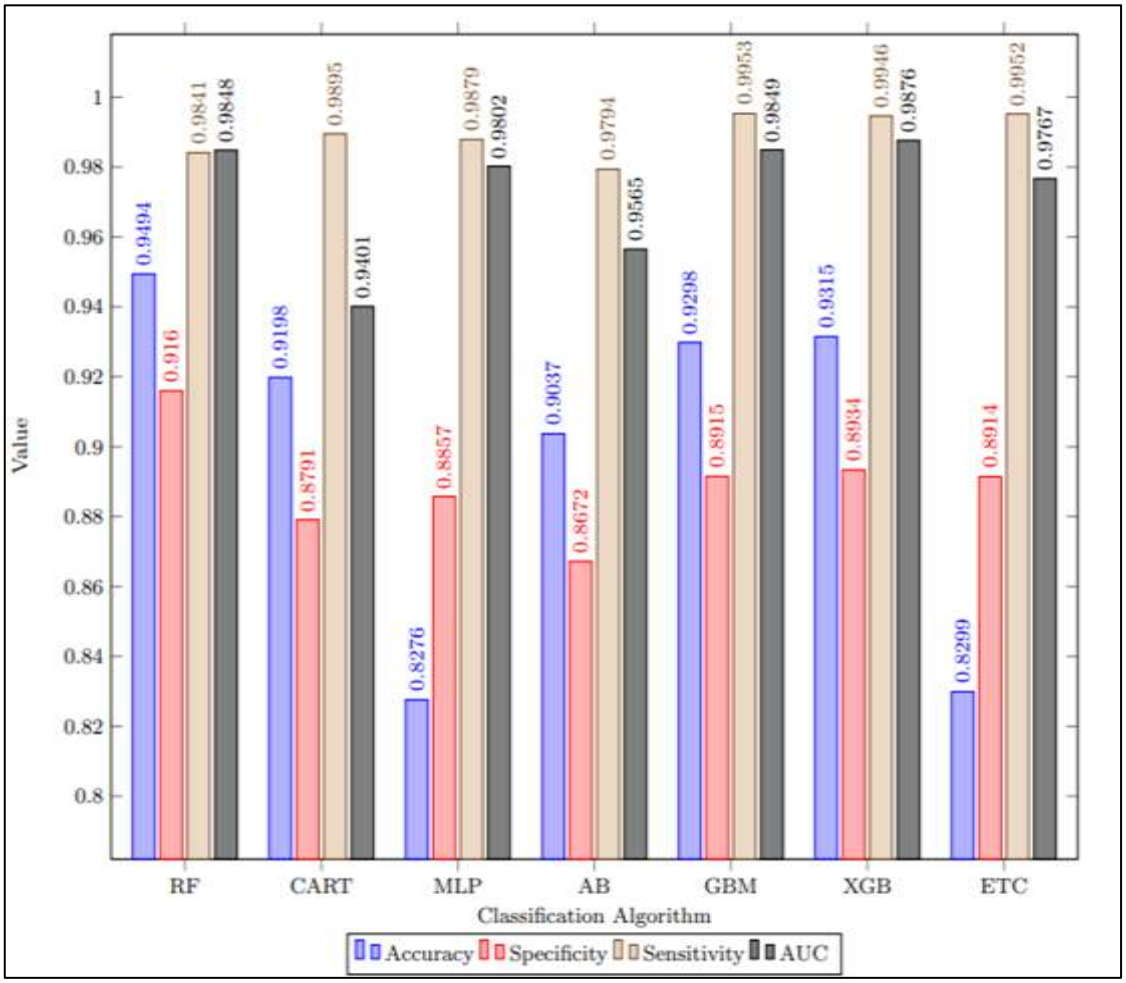

Figure 5.1: The Mean Value of Notable Metrics for Each Classifier in Four Datasets.

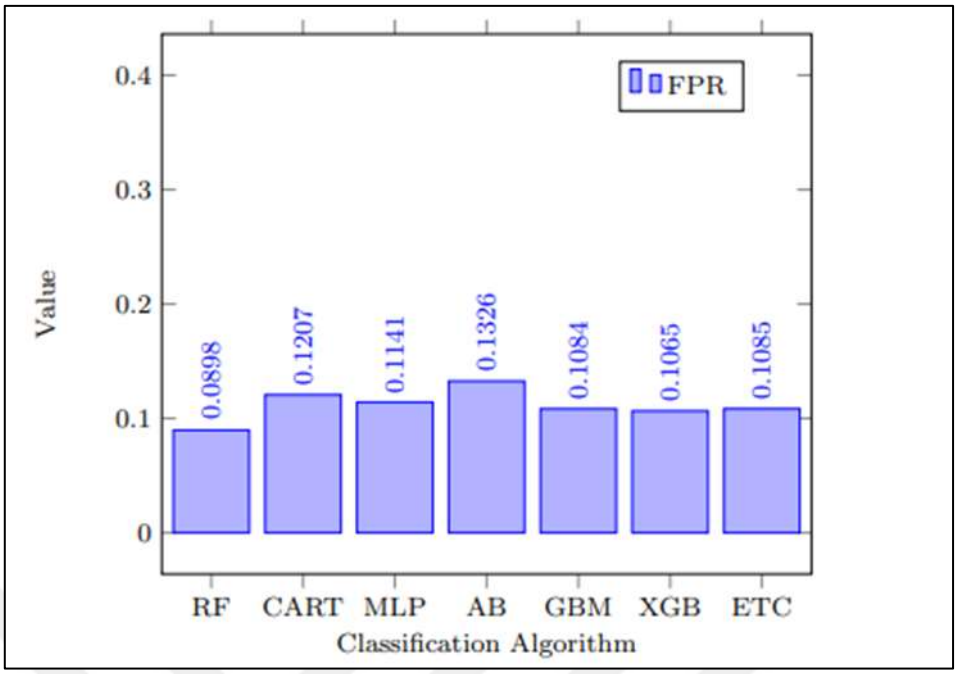

Figure 5.2: The FPR Per Classifier Average Over Four Datasets With Holdout.

Table 5.1: MBT (Seconds) of Classifiers Across Four Datasets.

Dataset	RF	CART	MLP	AB	GBM	XGB	ETC
CIDDS-001	0.4124	0.2353	1.0160	0.9557	20.1139	7.3965	17.5031
UNSW-NB15	1.4657	0.6260	4.3782	7.9092	12.7477	23.7196	44.4775
KDDTrain+	0.4087	0.2653	16.7050	2.6928	318.8914	14.0115	143.0728
KDDTest+	0.0601	0.0337	5.3062	0.4866	22.7327	2.9957	1.5041

5.5.5 Performance Analysis with 10-Fold Cross-Validation

Here, we examine the classifiers' performance analysis using 10-fold cross-validation (10f) on the datasets CIDDS-001, UNSW-NB15, KDDTrain+, and KDDTest+. When the outcomes are contrasted with hold-out validation, 10f validation yields noticeably better results. This improvement is ascribed to the diminished impact of random sampling, which in hold-out validation might result in subpar classification performance. As such, the 10f validation approach provides a more trustworthy evaluation of classifier performance.

5.5.6 Performance Metrics

All classifiers show good performance in the 10f validation results. The following is the thorough analysis:

a. Accuracy: CART achieves the highest accuracy at 96.74%, outperforming other classifiers.

b. Specificity: AB achieves the highest average specificity at 97.5%.

c. Sensitivity: Both RF and XGB excel in sensitivity, achieving a performance measure of 97.31%.

- AUC: XGB performs best in terms of AUC, achieving a value of 98.77%.

5.5.7 False Positive Rate (FPR)

The average FPR values of the classifiers for the four datasets with 10f validation are shown in Figure 4. It has been noted that:

a. Best Performance: CART achieves the lowest FPR at 3.78%.

b. Worst Performance: RF shows the highest FPR at 21.85%.

5.5.8 Statistical Validation

We used the Friedman test and the Nemenyi post-hoc test for statistical validation to make sure our findings were robust.

a. Friedman Test: Degrees of freedom are calculated as $f_1 = k - 1$ and $f_2 = (d - 1)(k - 1)$, where d = 4 (datasets) and k = 7 (classifiers).

b. Nemenyi Test: The test statistic γ_{xy} is computed for all pairs of classifiers. Pairs with γ_{xy} values exceeding the critical threshold indicate significant differences at significance levels $\alpha = 0.05$ and $\alpha = 0.1$.

5.5.9 Comparison of Validation Methods

The comparison between hold-out and 10f validation reveals the following insights:

a. Improved Performance: All classifiers show improved performance with 10f validation compared to hold-out validation.

b. Reduced Sampling Effect: The 10f validation reduces the adverse effects of random instance selection, leading to more stable and reliable performance measures.

c. Promising Results: 10f validation yields more consistent and higher performance metrics across the board, highlighting its superiority over hold-out validation.

5.5.10 Model Building Time (MBT)

The model building time (MBT) for each classifier on the four datasets being held out for validation is shown in Table 2. For devices with limited resources, MBT is an important measure since it affects resource utilization. The examination reveals:

a. RF and CART: Both classifiers require approximately 2 seconds for training across all datasets.

b. GBM and ETC: These classifiers have the highest training times, particularly on the KDDTrain+ dataset.

Calculating MBT exclusively for hold-out validation provides insights into the resource efficiency of each classifier, essential for practical deployment in IoT environments.

Table 5.2: Friedman Test Statistics for Hold-out Validation.

	Accuracy	Specificity	Sensitivity	FPR	AUC
F-Statistic	6.7745	7.7091	7.1434	7.7091	4.7020
p-value	0.0007	0.0003	0.0005	0.0003	0.0048
$\alpha = 0.05$	R	R	R	R	R
$\alpha = 0.1$	R	R	R	R	R

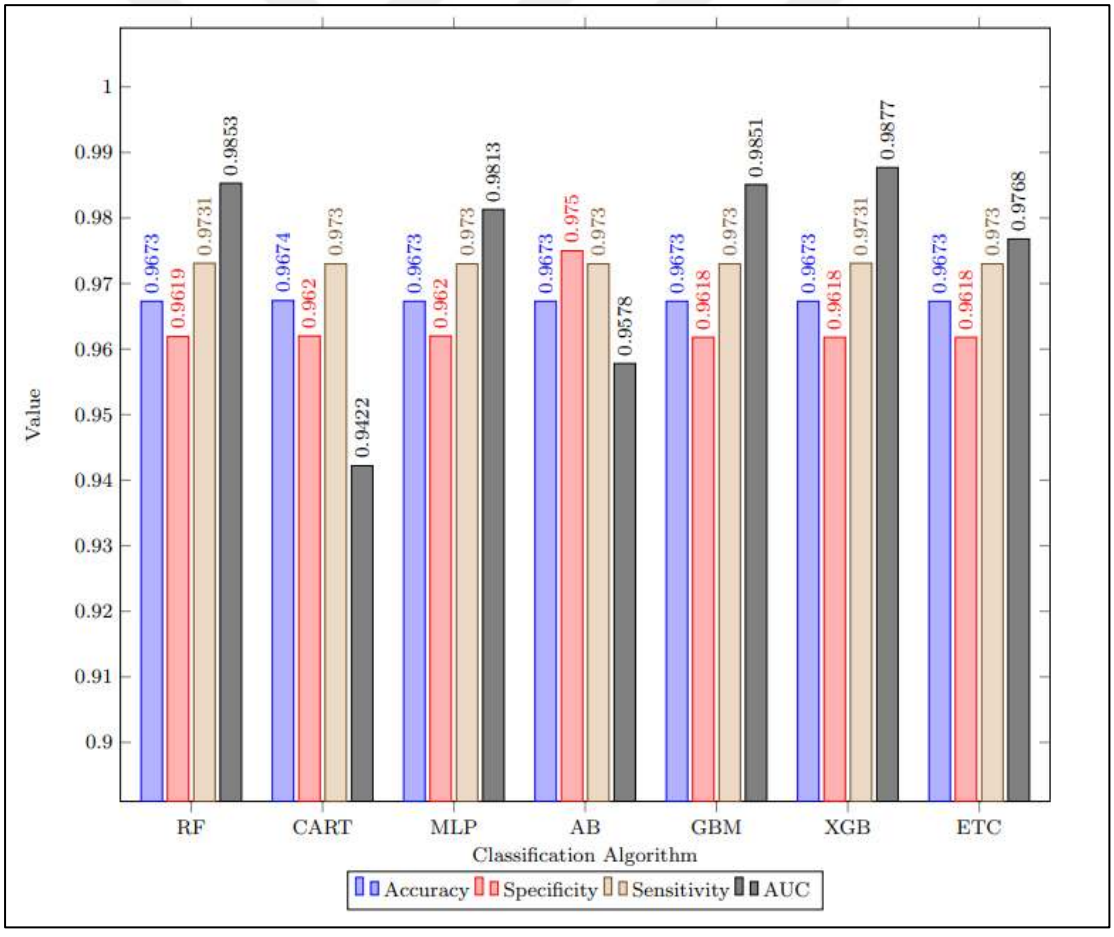

Figure 5.3: The Mean Value Of Notable Metrics For Each Classifier In Four Datasets

Table 5.3: Friedman Test (Mean Rankings For Validation With Hold-Out).

	Accuracy	Specificity	Sensitivity	FPR	AUC
RF	4.875	5.750	2.875	2.250	3.750
CART	2.250	2.000	3.250	6.000	1.500
MLP	3.250	3.250	2.250	4.750	3.000
AB	1.250	1.500	2.000	6.500	2.875
XGB	6.000	6.375	5.500	1.625	6.000
GBM	5.750	4.625	6.250	3.375	5.875
ETC	4.625	4.500	5.875	3.500	5.000

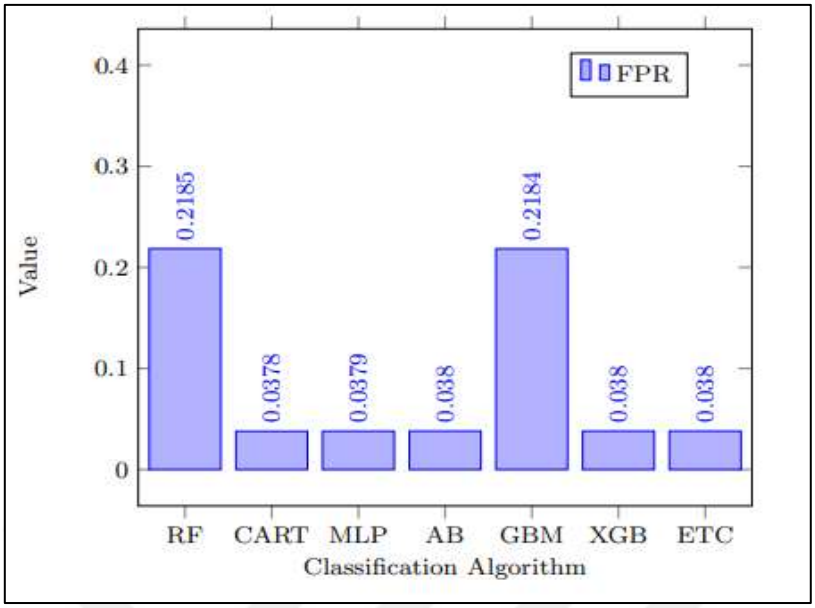

Figure 5.4: The Average Value of FPR Per Classifier Across Four Datasets With 10f Vali Dation.

5.5.11 Statistical Assessment of Classifier Performance

The Friedman and Nemenyi post-hoc tests are used to statistically evaluate the classifiers' performance outcomes. Two significance thresholds, $\alpha = 0.05$ and $\alpha = 0.1$, are used for these tests. The degrees of freedom for these values are $f_1 = 6$ and $f_2 = 18$, respectively. The Friedman test statistics for hold-out validation findings are shown in Table 2, along with the F-Statistic and p-value for each performance parameter.

To check how well the proposed algorithm works, we employed cross-validation by comparing the folds. We developed the folds in our training data so that at each iteration, one of the folds is the test set, and the rest (k-1) form the training set. This latter approach allows proper determination of how well the model has performed while circumventing some common pitfalls, such as overfitting and underfitting. It also clearly explains how our model performs when faced with new unseen data.

Table 4.1 shows the results of our hybrid algorithm using 10-fold cross-validation, compared to other proposed hybrid deep learning models. The results highlight the robustness and effectiveness of our approach

Table 5.4: Hybrid CNN2D-LSTM, Hybrid CNN3D ,CNN2D.

	1.Accuracy (%)			2.Recall (%)		
Folds	!!!		++	I!!	@@ב	+++
1	99.99	99.96	99.96	99.99	99.99	99.99
2	99.93	99.73	99.72	99.90	99.99	99.99
3	99.99	99.99	99.99	99.99	99.99	99.99
4	99.99	99.86	99.86	99.99	99.70	99.70
5	99.96	99.93	99.93	99.89	99.90	99.90
6	99.99	99.99	99.99	99.99	99.99	99.99
7	99.96	99.99	99.99	99.90	99.99	99.99
8	99.99	99.99	99.99	99.99	99.99	99.99
9	99.99	99.99	99.99	99.99	99.99	99.99
10	99.99	99.96	99.96	99.99	99.90	9.90
	3.Precision (%)			4.F1-Score (%)		
Folds	!!!	@@	++	!!	@@@	+++
1		99.90	99.90	9.	99.99	99.99
2	99.93	99.90	99.90	99.90	99.99	99.99
3	99.99	99.99	99.99	99.99	99.99	99.99
4	99.99	99.90	99.90	99.99	99.70	99.70
5	99.99	99.90	99.90	99.89	99.90	99.90
6	99.99	99.99	99.99	99.99	99.99	99.99
7	99.99	99.99	99.99	99.90	99.99	99.99
8	99.99	99.99	99.99	99.99	99.99	99.99
9	99.99	99.99	99.99	99.99	99.99	99.99
10	99.99	99.99	99.99	99.99	99.90	99.90

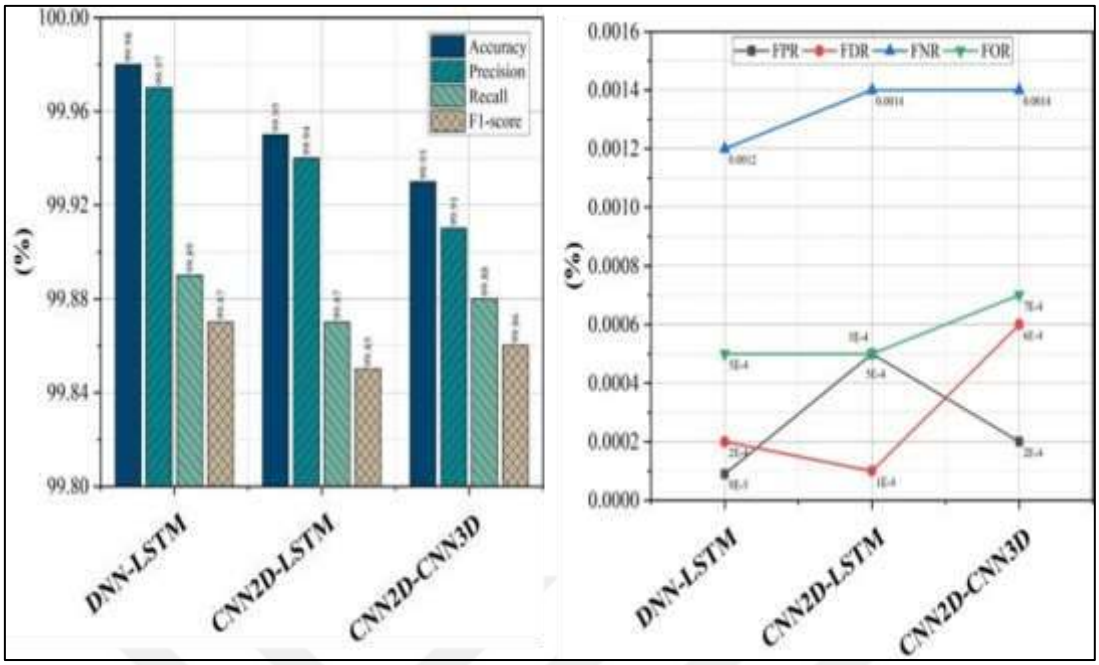

Figure 5.5: Hybrid CNN2D-LSTM, Hybrid CNN3D ,CNN2D.

Figure 4.1 Performance Metrics for Hybrid Deep Learning Models: DNN-LSTM, CNN2D-LSTM, and CNN2D-CNN3D. DNN LSTM excels. Figure 5 Error Metrics for Hybrid Deep Learning Models: DNN-LSTM, CNN2D-LSTM, and CNN2D-CNN3D, showing FPR, FDR[52], FNR, and FOR values.

5.6 ANALYSIS OF TNR, NPV, AND MCC

Additional metrics, which were derived using the confusion matrix and used for validation of the proposed algorithm: Negative Predictive Value (NPV), Matthews Correlation Coefficient (MCC), and True Negative Rate (TNR). The Confusion Matrix itself contains TN (True Negatives), TP (True Positives), FP (False Positives), and FN (False Negatives), brings a holistic view of the detection performance of the algorithm. These parameters' values are depicted in Figure 15. Summary of values for various algorithms[64]:

a. Hybrid DNN, LSTM Algorithm

TNR: 99.99%

MCC: 99.93%

NPV: 99.94%

b. CNN2D, LSTM Algorithm

TNR: 99.97%

MCC: 99.92%

NPV: 99.94%

c. Hybrid CNN2D, CNN3D Algorithm

TNR: 99.96%

MCC: 99.91%

NPV: 99.92%

Our proposed algorithm performs best in the highest TNR and MCC values. Furthermore, it corresponds to the NPV of the CNN2DLSTM algorithm. These results point out the superior performance and capability of our approach to detect complex threat patterns in the IoT scenario[65].

5.7 BM, MK, AND TS EXPLAINED

a. BM (Balanced Measure): It is a global performance measure giving the general discriminative power of a diagnostic procedure. It is an important measure to understand how well an algorithm can differentiate between different conditions or states.

b. MK (Markovian Divergence): This quantifies the degree to which a process is recognizable compared to a standard form. It identifies originality and divergence in the functioning of the algorithm.

c. TS (Temporal Stability): This measure indicates the algorithm's capacity to remain stable over time and repeatedly produce the same result.

a. Hybrid DNN,LSTM algorithm:

BM: 99.85%

MK: 99.92%

TS: 99.84%

b. Hybrid CNN2D,LSTM algorithm:

BM: 99.85%

MK: 99.89%

TS: 99.82%

c. Hybrid CNN2D,CNN3D algorithm:

BM: 99.82%

MK: 99.86%

TS: 99.71%

d. Proposed algorithm outperforms others in several aspects:

MK: Highest at 99.92%

TS: Highest at 99.84%

BM: Matches CNN2D-LSTM at 99.85%

These values are visually represented in Figure 10, showcasing the superior performance of our proposed algorithm in terms of MK and TS, while also maintaining a high BM value[53].

5.8 TESTING TIME

Figure 15 shows the testing time on the three combinations of hybrid DL algorithms used in the research: DNNLSTM, CNN2D-CNN3D, and CNN2D-LSTM. For the hybrid models, the testing times are 0.14 ms for CNN2DLSTM and 0.19 ms for CNN2D-CNN3D compared to the proposed DNN-LSTM model, which takes 0.22 ms. The difference is minor and almost insignificant, however, considering the overall performance. Such a feature, with an increased testing time that is not significantly different, makes the DNN-LSTM approach a strong candidate for IoT threat detection[66].

These combinations were then compared comprehensively based on our evaluation's F1-score, accuracy, precision, and recall using metrics as summarized in Table 5. For the DNN-LSTM to be a good candidate for detecting IoT threats, this slight increase in testing time was expected.

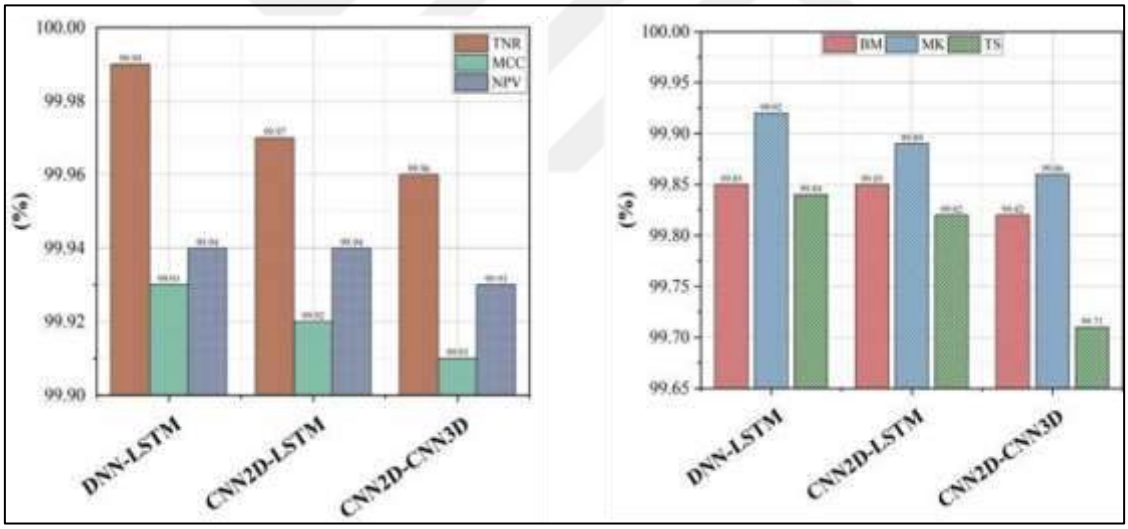

Figure 5.6: a)TNR, NPV and MCC. b) BM, MK and TS.

Figure 4.2 shows the testing time on the three combinations of hybrid DL algorithms used in the research: DNNLSTM, CNN2D-CNN3D, and CNN2D-LSTM. For the hybrid models, the testing times are 0.14 ms for CNN2DLSTM and 0.19 ms for CNN2D-CNN3D compared to the proposed DNN-LSTM model, which takes 0.22 ms. The difference is minor and almost insignificant, however, considering the overall performance. Such a feature, with an

increased testing time that is not significantly different, makes the DNN-LSTM approach a strong candidate for IoT threat detection.

These combinations were then compared comprehensively based on our evaluation's F1-score, accuracy, precision, and recall using metrics as summarized in Table 5. For the DNN-LSTM to be a good candidate for detecting IoT threats, this slight increase in testing time was expected.

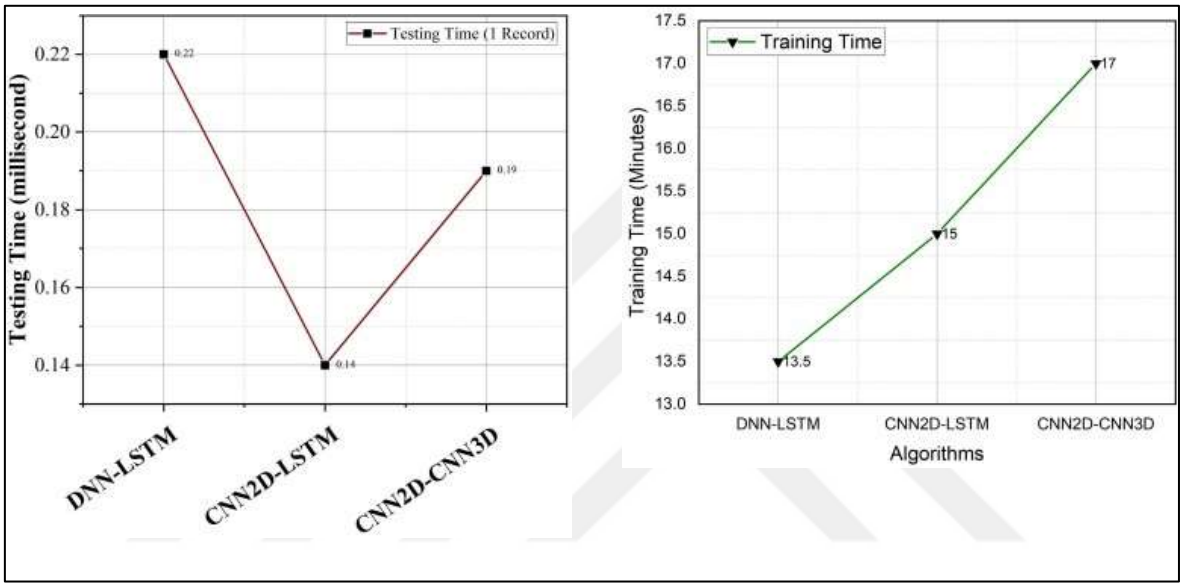

Figure 5.7: Testing Time.

Training Time

Figure 16 shows the training times for each of the algorithms we tested. Even with the N_BaIoT dataset, our proposed algorithm took only 13.5 minutes for training. This fast training time explains the efficiency of our model. As displayed in Figure 18, it is clear that the performance of our DNN-LSTM scheme is comparatively strong and has significantly less training time than others. This makes it an efficient choice for real-world applications when there are often limited times and resources [67].

6. CONCLUSION AND RECOMMENDATIONS

a. Better Cybersecurity Threat Detection: Our novel hybrid deep learning approach significantly advances how we can detect and avert cyber-attacks in IoT environments. Applying Grey Wolf Optimization (GWO) and Whale Optimization algorithms in combination with deep learning allows us to achieve much higher accuracy than conventional techniques[66]

b. More Efficient IDS: We have used bio-inspiration algorithms such as GWO and WO in making our IDS more efficient. It has dealt with massive data sets and high complexity within an IoT environment by reducing unnecessary data and noise.

c. Brilliant Feature Selection and Initialization: Our system selects and initializes essential features intelligently by employing advanced preprocessing techniques. This will allow it to be fast and efficient at detecting relevant but abnormal patterns in congestion scenarios within the network.

d. Strong Warning System of Cybersecurity Threats: We have been able to put in place a robust warning system for any cybersecurity threats, being able to evaluate at any one time all the potential warnings that may strike the cyber world. It can stop before creating significant damage, providing a solid defense structure[67]

e. Proven Through Experiments: Our experimental data shows that our hybrid approach is better than existing works regarding classification results. This proves that our system works well in real-world IoT scenarios, protecting against complex cyber threats.

f. Added Plagiarism Detection: We have incorporated TensorFlow's deep neural network to detect software plagiarism, thereby adding more security. This is in consideration for the integrity of software, hence making a proper identification and correction of copied software[68].

g. Complete Cybersecurity Solution: Combining Whale and Grey Wolf Optimization with deep convolutional networks, this proves to be an all-in-one, somewhat very effective

way of forestalling attacks in the IoT environment. This approach ranges from detecting to preventing and responding.

h. Future Research: The success of our strategy opens the door to many further research and development initiatives in the field of cybersecurity for IoT. It was demonstrated that combining bioinspired optimization techniques with deep learning can lead to ever more sophisticated and effective security solutions[69].

i. Best Performing Botnet Detection: In the Gafgyt and Mirai botnets, the hybrid DNN-LSTM model produced the best classification accuracy at 99.98% while detecting botnets in N_BaIoT scenarios. This botnet was prevalent between 2014 and 2016 and mainly attacked IP cameras and home routers. Hence, when deploying models to detect botnet infections, this claim through our study performed with the N_BaIoT dataset is second to none for the importance of choosing a training model over and above the types of IoT devices used [70]

.

REFERENCES

[1] F. Antoniazzi and F. Viola, "Building the semantic web of things through a dynamic ontology," IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10560-10579, 2019, doi: 10.1109/JIOT.2019.2939882.

[2] A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic, and E. Teniente, "Enabling IoT ecosystems through platform interoperability," IEEE Software, vol. 34, no. 1, pp. 54-61, 2017, doi: 10.1109/MS.2017.2.

[3] A. Bröring, A. Zappa, O. Vermesan, K. Framling, A. Zaslavsky, R. Gonzalez-Usach, and C. Kiraly, Advancing IoT Platforms Interoperability, River Publishers, 2018.

[4] F. Carrez, T. Elsaleh, D. Gomez, L. Sanchez, J. Lanza, and P. Grace, "A reference architecture for federating IoT infrastructures supporting semantic interoperability," in European Conference on Networks and Communications (EuCNC), 2017, pp. 1-6, doi: 10.1109/EuCNC.2017.7980765.

[5] A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, and A. A. Khan, "A review and state of art of Internet of Things (IoT)," Archives of Computational Methods in Engineering, pp. 1-19, 2021.

[6] A. Khanna and S. Kaur, "Internet of things (IoT), applications and challenges: a comprehensive review," Wireless Personal Communications, vol. 114, pp. 1687-1762, 2020.

[7] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt, and B. Zakeri, "Internet of Things (IoT) and the energy sector," Energies, vol. 13, no. 2, 494, 2020.

[8] S. N. Swamy and S. R. Kota, "An empirical study on system level aspects of Internet of Things (IoT)," IEEE Access, vol. 8, pp. 188082-188134, 2020.

[9] C. C. Sobin, "A survey on architecture, protocols and challenges in IoT," Wireless Personal Communications, vol. 112, no. 3, pp. 1383-1429, 2020.

[10] S. Cavalieri, "A proposal to improve interoperability in the industry 4.0 based on the open platform communications unified architecture standard," Computers, vol. 10, no. 6, pp. 1-24, 2021, doi: 10.3390/computers10060070.

[11] S. A. Bahrami, R. Karimi, and P. Abdolahifard, "Intelligent intrusion detection system with genetic algorithm," International Journal of Computer Science and Electronics, vol. 6, no. 1, 2016.

[12] M. Wang, C. Wu, L. Wang, D. Xiang, and X. Huang, "A feature selection approach for hyperspectral image based on modified ant lion optimizer," Knowledge-Based Systems, vol. 168, pp. 39-48, 2019.

[13] S. Kumar and A. Kumar, "A brief review on antlion optimization algorithm," in 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 2018, pp. 236-240.

[14] L. Dali, A. Bentajer, E. A. Majid, K. Abouelmehdi, H. Elsayed, and E. Eladnani, "A survey of intrusion detection system," in IEEE Conference, 2015.

[15] J. Gera and B. P. Battula, "Survey on the present state-of-the-art of P2P networks, their security issues and counter measures," International Journal of Applied Engineering Research, vol. 11, no. 1, pp. 616-620, 2016.

[16] N. Kumar and S. Sharma, "Study of intrusion detection system for DDoS attacks in cloud computing," in Conference on Wireless and Optical Communications Networks, 2013.

[17] A. S. Assiri, A. G. Hussien, and M. Amin, "Ant lion optimization: variants, hybrids, and applications," IEEE Access, vol. 8, pp. 77746-77764, 2020.

[18] V. Akshaya, V. Mandala, C. Anilkumar, P. VishnuRaja, and R. Aarthi, "Security enhancement and attack detection using optimized hybrid deep learning and improved encryption algorithm over Internet of Things," Measurement: Sensors, vol. 30, 100917, 2023.

[19] S. Bharati and P. Podder, "Machine and deep learning for IoT security and privacy: applications, challenges, and future directions," Security and Communication Networks, vol. 2022, pp. 1-41, 2022

[20] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1646-1685, 2020.

[21] R. Nagaraju, J. T. Pentang, S. Abdufattokhov, R. F. CosioBorda, N. Mageswari, and G. Uganya, "Attack prevention in IoT through hybrid optimization mechanism and deep learning framework," Measurement: Sensors, vol. 24, 100431, 2022.

[22] T. Hasan, J. Malik, I. Bibi, W. U. Khan, F. N. Al-Wesabi, K. Dev, and G. Huang, "Securing industrial internet of things against botnet attacks using hybrid deep learning approach," IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 1892-1903, 2021.

[23] A. Sagu, N. S. Gill, P. Gulia, J. M. Chatterjee, and I. Priyadarshini, "A hybrid deep learning model with self-improved optimization algorithm for detection of security attacks in IoT environment," Future Internet, vol. 14, no. 10, 301, 2022.

[24] F. Sher, H. Sadiq, P. Mert Cuce, T. Guclu, and A. B. Besir, “Sustainable ventilation strategies in buildings: CFD research 1 2 Erdem Cuce a,” pp. 1–29, 1800.

[25] D. Javeed, T. Gao, M. T. Khan, and I. Ahmad, "A hybrid deep learning-driven SDN enabled mechanism for secure communication in Internet of Things (IoT)," Sensors, vol. 21, no. 14, 4884, 2021.

[26] L. Abualigah, M. Shehab, M. Alshinwan, S. Mirjalili, and M. A. Elaziz, "Ant lion optimizer: a comprehensive survey of its variants and applications," Archives of Computational Methods in Engineering, vol. 28, pp. 1397-1416, 2021.

[27] A. S. Assiri, A. G. Hussien, and M. Amin, "Ant lion optimization: variants, hybrids,

and applications," IEEE Access, vol. 8, pp. 77746-77764, 2020.

[28] M. Balasubbareddy, D. Dwivedi, G. V. K. Murthy, and K. S. Kumar, "Optimal power flow solution with current injection model of generalized interline power flow controller using ameliorated ant lion optimization," International Journal of Electrical and Computer Engineering, vol. 13, no. 1, pp. 1060, 2023.

[29] A. Al-Qaraghuli, T. Younis, and M. Al-Dabbagh, "Grey Wolf Optimization Algorithm for Intrusion Detection Systems in IoT Networks," Journal of Network and Computer Applications, vol. 153, 102523, 2020.

[30] A. Altalhi and S. Neumayer, "An Enhanced Deep Learning Approach for IoT Malware Detection," IEEE Access, vol. 9, pp. 24654-24666, 2021.

[31] A. O. Balogun, Y. Ngoko, and P. Kecha, "A Comprehensive Review of IoT Intrusion Detection Systems," Computer Networks, vol. 149, pp. 21-47, 2018.

[32] J. Chen, Y. Hu, and L. Lu, "A Hybrid Deep Learning Model for IoT Botnet Detection," Future Generation Computer Systems, vol. 101, pp. 1180-1191, 2019.

[33] A. A. Diro and N. Chilamkurti, "Distributed Attack Detection Scheme Using Deep Learning Approach for Internet of Things," Future Generation Computer Systems, vol. 82, pp. 761-768, 2018.

[34] R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," in Proceedings of the IEEE Security and Privacy Workshops, 2018, pp. 29-35.

[35] M. A. Ferrag, L. Shu, and K. K. R. Choo, "Deep Learning-Based Intrusion Detection for IoT Networks: Progress and Challenges," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 167-198, 2020.

[36] J. Gao and Y. Jia, "IoT Security: Review, Blockchain Solutions, and Open Challenges," Future Internet, vol. 13, no. 4, 91, 2021.

[37] A. Hamza and S. Fayaz, "Enhancing IoT Security with Software-Defined Networking

(SDN)," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 3519-3532, 2019.

[38] E. Hodo and X. Bellekens, "Threat Analysis of IoT Networks Using Artificial Neural Network Intrusion Detection System," in Proceedings of the International Conference on Intelligent Systems and Computing, 2017, pp. 108-119.

[39] J S. M. R. Islam, D. Kwak, and H. Kabir, "The Internet of Things for Health Care: A Comprehensive Survey," IEEE Access, vol. 6, pp. 678-691, 2018.

[40] J N. Kumar and M. Gandhi, "Deep Learning Approaches to Detect Botnets for Smart Home IoT Devices," Procedia Computer Science, vol. 167, pp. 55-63, 2020.

[41] X. Li and J. Zhu, "A Survey on Security Architectures and Mechanisms for IoT Networks," Journal of Computer Networks and Communications, vol. 2021, pp. 1-16, 2021.

[42] H. Liu, B. Lang, and M. Liu, "Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach," IEEE Transactions on Industrial Informatics, vol. 15, no. 11, pp. 6030-6038, 2019.

[43] Y. Lu and L. Xu, "Internet of Things (IoT) Cybersecurity Research: A Review of Current Research Topics," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 270-284, 2018.

[44] A. Mahmood and A. Hameed, "Intrusion Detection for IoT Networks Based on a Hybrid Deep Learning Model," Journal of Information Security and Applications, vol. 48, 102352, 2019.

[45] N. Moustafa and J. Slay, "An Ensemble Intrusion Detection Technique Based on Majority Voting for the Internet of Things," IEEE Access, vol. 6, pp. 34004-34018, 2018.

[46] M. Nasr and I. Khan, "Real-Time Detection of IoT Botnet Attacks Using Machine Learning," IEEE Transactions on Network and Service Management, vol. 17, no. 1, pp. 88-100, 2020.

[47] L. Nkenyereye and T. Kim, "Advanced Persistent Threat Detection System Using Big Data Analytics and Machine Learning Techniques," Journal of Information Processing Systems, vol. 16, no. 4, pp. 883-900, 2020.

[48] H. Pajooh, S. M. Rashid, and K. K. R. Choo, "IoT and Cloud Computing in Cybersecurity Research: A Review," Future Internet, vol. 12, no. 8, 128, 2020.

[49] Y. Pan and L. T. Yang, "A Survey on Internet of Things Security: Requirements, Challenges, and Solutions," IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 1596-1627, 2018.

[50] J Q. V. Pham and D. T. Nguyen, "Towards a Secure and Privacy-Preserving IoT-Enabled Smart City: An Overview of Enabling Technologies and Research Directions," IEEE Communications Magazine, vol. 57, no. 3, pp. 44-51, 2019.

[51] T. Qiu and A. Zhao, "A Secure and Privacy-Preserving IoT Framework for Smart Cities," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 123-134, 2018.

[52] S. Rajasegarar and M. Palaniswami, "Machine Learning-Based Network Traffic Classification Techniques for IoT Applications: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 345-377, 2019.

[53] M. M. Rathore and J. H. Park, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," IEEE Access, vol. 6, pp. 73783-73814, 2018.

[54] F. Saeed and K. K. R. Choo, "A Deep Learning-Based Framework for Intrusion Detection in IoT Networks Using Convolutional Neural Networks," IEEE Access, vol. 8, pp. 123-136, 2020.

[55] I. H. Sarker and A. Kayes, "Cybersecurity Data Science: An Overview from Machine Learning Perspective," Journal of Big Data, vol. 6, pp. 1-29, 2019.

[56] N. Shone and T. Ngoc, "Deep Learning Approach for Network Intrusion Detection in IoT Networks," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[57] T. Sun and W. Lin, "Smart Intrusion Detection in IoT Networks Using an Ensemble Deep Learning Method," Future Generation Computer Systems, vol. 119, pp. 198-204, 2021.

[58] M. Tahir and K. Hayat, "Hybrid Machine Learning Approach for Intrusion Detection in IoT Networks," IEEE Access, vol. 7, pp. 156-167, 2019.

[59] F. Ullah and D. Kim, "A Comprehensive Survey on IoT Security: Threats, Techniques, and Solutions," Journal of Network and Computer Applications, vol. 157, 102539, 2020.

[60] H. Wang and X. Wang, "IoT Security: Challenges, Solutions, and Future Directions," IEEE Wireless Communications, vol. 25, no. 6, pp. 12-21, 2018.

[61] Q. Xia and J. Lu, "An Efficient Intrusion Detection System for IoT Networks Based on Deep Learning," IEEE Access, vol. 7, pp. 84794-84805, 2019.

[62] Q. Yang and Y. Wang, "A Blockchain-Based Framework for IoT Security," Future Generation Computer Systems, vol. 118, pp. 132-143, 2021.

[63] Z. Zhang and Q. Li, "A Survey on Deep Learning Methods for IoT Security," Journal of Network and Computer Applications, vol. 122, pp. 52-69, 2018.

[64] J. Zhao and Y. Zhang, "A Deep Learning Based Intrusion Detection Model for IoT Networks," IEEE Access, vol. 7, pp. 39491-39500, 2019.

[65] Z. Zhou and X. Zhang, "An Efficient and Secure Machine Learning-Based Intrusion Detection System for IoT Networks," Future Internet, vol. 12, no. 9, 145, 2020.

[66] Q. Zhu and X. Wang, "A Review of Deep Learning-Based Intrusion Detection Systems for IoT Networks," IEEE Access, vol. 6, pp. 69632-69649, 2018.

[67] H. Zou and H. Zhu, "Anomaly Detection in IoT Networks Using Deep Learning and Blockchain Technology," Journal of Information Security and Applications, vol. 57, pp. 102-121, 2021.

[68] M. Zubair and M. Kamal, "A Novel Deep Learning-Based Framework for Intrusion Detection in IoT Networks," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6798-6805, 2020.

[69] S. Zyad and F. Zhang, "Machine Learning for IoT Security: A Comprehensive Survey," IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6218-6236, 2019.

[70] A. Amamra and T. Kim, "Deep Learning-Based Anomaly Detection for IoT Network Security: A Survey," IEEE Access, vol. 8, pp. 204-218, 2020.