A FRAMEWORK FOR DISTRIBUTED DENIAL-OF-SERVICE ATTACK DETECTION IN INTERNET OF THINGS ENVIRONMENTS

SILAS AMISI WAWIRE

MASTER OF SCIENCE IN COMPUTER SYSTEMS

JOMO KENYATTA UNIVERSITY
OF
AGRICULTURE AND TECHNOLOGY

2026

A Framework for Distributed Denial-of-Service Attack Detection in Internet of Things Environments

Silas Amisi Wawire

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Systems of the Jomo Kenyatta University of Agriculture and Technology

2026

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University.

Sign ……. Date ……..

Silas Amisi Wawire

This thesis has been submitted for examination with our approval as University Supervisors:

Sign ……. Date ……..

Dr. Lawrence Nderu, PhD

JKUAT, Kenya

Sign ……. Date ……..

Dr. Dalton Ndirangu, PhD

USIU, Kenya

DEDICATION

To my family, which has given me immense support.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to the many people I have worked with over the past few years for their constant support and inspiration, who have catalyzed the journey to reach the current stage of my studies. My sincere thanks go to my supervisors for their excellent guidance, encouragement, and support. I would also like to thank the entire faculty of computing at JKUAT for their support throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ACRONYMS AND ABBREVIATIONS

DOS	Denial-of-Service
DDOS	Distributed Denial-of-Service
ML	Machine Learning
IOT	Internet of Things
CNN	Convolutional Neural Network
BILSTM	Bidirectional Long Short-Term Memory
MQTT	Message Queuing Telemetry Transport

ABSTRACT

Internet of Things (IoT) networks are ubiquitous across industries, homes, and critical infrastructure due to their automation capabilities. However, IoT devices remain highly vulnerable to Distributed Denial-of-Service (DDoS) attacks owing to their limited computational power and inherent heterogeneity. Also, IoT systems often favor usability over security. While deep learning has shown promise for detecting such attacks, existing approaches have largely been validated on conventional network datasets using centralized architectures that create single points of failure. Furthermore, the centralized approaches raise privacy concerns by requiring raw data transmission to the cloud and lack preprocessing tailored to IoT-specific protocols such as MQTT. This study aimed to develop a distributed deep learning framework for DDoS detection in IoT environments. The specific objectives were to design and implement a distributed detection framework based on deep learning, and to evaluate and compare its performance against a centralized paradigm using accuracy, precision, recall, and F1-score. A quantitative experimental research design using the DoS/DDoS-MQTT-IoT dataset was employed. The proposed framework integrated three components: a CNN-BiLSTM ensemble model, a three-tier edge-fog-cloud architecture incorporating federated learning where edge devices performed preprocessing and fog nodes conducted local training while raw data remained on premises, and preprocessing adapted to MQTT's publish-subscribe semantics. Model hyperparameters were held constant across both experimental conditions, and ten-fold cross-validation was applied. Statistical significance was assessed using a paired t-test with an alpha level of 0.05. The distributed detection framework achieved 99.68% accuracy, 99.02% precision, 99.28% recall, and 99.10% F1-score. The distributed framework demonstrated a 64% lower error rate than the centralized approach (1.35% versus 3.76%). This improvement was statistically significant as confirmed by a paired t-test over ten-fold cross-validation ($p < 0.001$). The distributed framework also outperformed traditional machine learning methods and standalone deep learning models including CNN alone, BiLSTM alone, and RNN alone. Overall, the distributed approach exhibits superior accuracy and adaptability compared to centralized detection. It also provides privacy benefits through federated learning. Future work should validate the framework on additional IoT protocols and datasets and explore model compression techniques to further reduce computational requirements on resource-constrained edge devices.

CHAPTER ONE

INTRODUCTION

1.1 Introduction to the Chapter

This chapter presents the background and motivation for the research. It is organized into various sections. Section 1.2 provides the background on IoT networks and DDoS attacks. Section 1.3 states the problem addressed by this research. Section 1.4 presents the justification for the study. Section 1.5 states the general and specific objectives. Section 1.6 presents the research questions. Section 1.7 defines the scope of the study. Finally, Section 1.8 provides a chapter summary.

1.2 Background

The rapid growth of Internet of Things (IoT) networks has ushered in a new era of connectivity. It has brought about unprecedented levels of efficiency, automation, and convenience by enabling data exchange and seamless communication among everyday devices (Fortune Business Insights, 2023). IoT has revolutionized the way human beings interact with technology in healthcare, smart homes, cities, and other areas of our lives. At its core, the IoT trend encompasses incorporating computing and communication capabilities into everyday devices. The typical IoT network comprises sensors and actuators, computing units, and connectivity modules (Thoutam, 2021). Other components might include security and cloud-based analytics, platforms, and visualization. However, there are inherent vulnerabilities associated with the widespread adoption of IoT systems. These vulnerabilities have made IoT environments attractive targets for malicious actors, especially Distributed Denial of Service (DDoS) attacks.

A DDoS attack happens when a network or system is flooded with a devastating quantity of traffic or requests, rendering it unable to respond to authentic users. Concerning IoT, DDoS attacks can be especially overwhelming. Since IoT networks comprise many

interconnected devices, each with its own computing abilities, these attacks can exploit the collective power of compromised IoT devices to launch large-scale assaults (Swain, Pattnaik, & Satpathy, 2025). The impacts of DDoS attacks on IoT networks are extensive and can have grave implications. To begin with, critical services relying on IoT infrastructure, such as electricity grids, transportation systems, smart cities, and healthcare facilities, can be disrupted. This disruption can result in extensive economic and societal consequences (Swain, Pattnaik, & Satpathy, 2025). For example, a hospital's IoT-enabled medical system can become unresponsive during a lifesaving procedure, hence causing an avoidable death. Additionally, DDoS attacks targeting IoT networks can compromise the availability, integrity, and confidentiality of the data being transmitted (Bhattacharjya, 2022). Moreover, the reliability, trustworthiness, and utilization of IoT systems themselves can be weakened by DDoS attacks.

Detecting DDoS attacks can benefit from machine learning, especially deep learning. Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed (Taye, 2023). Essentially, machine learning algorithms are designed to learn from data and improve their performance over time. They operate by identifying patterns, relationships, and insights within the data and using them to make predictions or take actions (Kumar & Vatsavayi, 2025). The key steps in machine learning include data collection, data preprocessing, feature extraction, algorithm training, and algorithm evaluation (Kumar & Vatsavayi, 2025). Once the algorithm has been trained, it can be integrated into a real-world application or system for making predictions or taking actions.

1.3 Statement of Problem

Despite growing awareness of DDoS vulnerabilities in IoT environments, existing detection approaches remain inadequate for practical deployment. The fundamental challenge lies in the fact that conventional detection methods were designed for traditional

networks with homogeneous devices, stable traffic patterns, and abundant computational resources (Williams et al., 2022). On the contrary, IoT networks exhibit precisely the opposite characteristics.

In traditional enterprise networks, DDoS detection has achieved reasonable success using signature-based systems, rate limiting, and anomaly detection techniques. These methods assume relatively stable network boundaries, consistent device behavior, and sufficient computational capacity to run detection algorithms (Swain, Pattnaik, & Satpathy, 2025). However, when these same methods are applied to IoT environments, they fail for three interrelated reasons. First, heterogeneity defeats signature-based approaches. Second, resource limitations impede running conventional detection at the edge (Swain, Pattnaik, & Satpathy, 2025). Third, atypical traffic patterns render statistical anomaly detection unreliable.

Machine learning has been extensively applied to DDoS detection in conventional networks, with methods such as Random Forest, Support Vector Machines (SVM), XGBoost, and K-Nearest Neighbors (KNN) achieving high accuracy (Alhamami et al., 2026). However, these methods carry forward the same assumptions that fail in IoT environments (Williams et al., 2022). They rely on manually engineered features—requiring domain experts to select and construct input variables—which cannot scale to the diversity of IoT devices and protocols. A feature set designed for HTTP traffic, for example, captures little relevance for MQTT's publish-subscribe semantics. These methods also struggle with the high dimensionality, and complex temporal dependencies present in IoT traffic, where a DDoS attack may manifest as subtle changes in message patterns over extended periods (Noaman et al., 2022). Most critically, traditional machine learning methods were designed for centralized deployment on powerful servers. They cannot be easily distributed across fog nodes, nor can they be trained using federated learning that preserves data privacy. When deployed in a centralized architecture, they require all raw traffic data to be transmitted to a central server, a non-starter for many IoT deployments due to bandwidth constraints, latency requirements, and privacy regulations

such as GDPR and HIPAA.

Deep learning addresses several limitations of traditional machine learning by autonomously extracting hierarchical features from raw data, eliminating manual feature engineering. Convolutional Neural Networks (CNNs) capture local patterns in network traffic, while Long Short-Term Memory (LSTM) networks and Bidirectional LSTM (BiLSTM) capture temporal dependencies across sequences. Studies such as Aswad et al. (2023) and Roopak, Tian, and Chambers (2019) have demonstrated that CNN-BiLSTM ensembles achieve high accuracy on DDoS classification tasks. However, these studies share a critical limitation: they evaluate deep learning models in centralized architectures using datasets that either predate modern IoT protocols or were collected from conventional networks. Diro and Chilamkurti (2018) took an important step by proposing a distributed architecture using fog nodes, but their validation relied on the outdated NSL-KDD dataset. Consequently, evidence for deep learning effectiveness in distributed IoT environments remains sparse. Besides, no study has validated a CNN-BiLSTM model on real-world MQTT attack data within a federated edge-fog-cloud architecture.

The convergence of these limitations points to a specific research gap. While individual elements of a solution exist in isolation, a review of the literature indicates that no prior study has combined all four elements necessary for practical, privacy-preserving DDoS detection in MQTT-based IoT environments. First, although deep learning models including CNN-BiLSTM have been applied to DDoS detection (Aswad et al., 2023; Roopak, Tian, & Chambers, 2019), these studies were validated on conventional network datasets or datasets that do not specifically capture MQTT protocol attacks; to the best of this researcher's knowledge, no study has validated a CNN-BiLSTM model exclusively on the DoS/DDoS-MQTT-IoT dataset. Second, while distributed architectures incorporating fog computing have been proposed for DDoS detection (Diro & Chilamkurti, 2018), and federated learning has been explored in other security contexts, the specific combination of a three-tier edge-fog-cloud architecture with federated learning for MQTT-based DDoS detection remains underexplored. Third, although

preprocessing techniques for network traffic are well-established, preprocessing specifically adapted to extract features from MQTT's publish-subscribe semantics—including message types, QoS levels, topic structures, and will message parameters—has received limited attention in the DDoS detection literature. Fourth, the DoS/DDoS-MQTT-IoT dataset (Alatram et al., 2023) is a recent contribution, and at the time of this study, only a limited number of published works have employed it for validation. This thesis contributes to addressing these gaps by developing and evaluating a distributed deep learning framework that integrates these four elements within a single coherent architecture, then comparing its performance against a centralized baseline under identical conditions.

1.4 Justification

Protecting IoT networks from DDoS attacks is key to ensure the smooth operation of IoT-supported systems and services. As the reliance on IoT devices continues to grow across different industries, the potential impact of successful DDoS attacks becomes more severe. Such attacks pose risks to critical infrastructure, public safety, and personal privacy. To address this security challenge, it is essential to develop a comprehensive detection framework that can effectively identify DDoS attacks in IoT environments.

Beyond addressing the immediate research gap, this thesis makes three essential contributions to the field of IoT security. First, the theoretical contribution operationalizes system resilience theory by demonstrating how distributed detection architectures can anticipate, adapt to, and preserve core functions during DDoS attacks. Second, the practical contribution provides a framework with a preprocessing pipeline specifically tailored to MQTT's publish-subscribe semantics. This enables effective feature extraction from IoT network traffic. Third, the methodological contribution establishes baseline metrics with 10-fold cross-validation.

By leveraging the power of deep learning techniques and considering the unique characteristics of IoT networks, the research enabled the development of a robust

algorithm capable of accurately identifying DDoS attacks in real-time. This algorithm was designed to analyze the diverse traffic patterns generated by IoT devices and detect anomalies indicative of DDoS attacks to safeguard the integrity and availability of IoT systems.

The outcomes of this research have the potential to greatly enhance the security posture of IoT networks. Specifically, they can empower organizations to detect and respond to DDoS attacks swiftly. By implementing the distributed detection algorithm, organizations can gain better visibility into the security of their IoT deployments, mitigate the risks posed by DDoS attacks, and ensure the continuous and reliable operation of their IoT-enabled systems and services.

1.5 Objectives

1.5.1 General Objective

To develop a distributed framework for the detection of DDoS attacks in IoT environments using deep learning.

1.5.2 Specific Objectives

1. To analyze the challenges of distributed DDoS detection in IoT environments using deep learning.

2. To design and implement a distributed DDoS detection framework based on deep learning.

3. To evaluate and compare the performance of the proposed distributed framework against a centralized paradigm using accuracy, precision, recall, and F1-score metrics.

1.6 Research Questions

To guide the research, the following questions were addressed:

1. What are the key challenges of distributed DDoS detection in IoT environments using deep learning, and how can they be addressed?

2. How can a distributed DDoS detection framework be designed and implemented using deep learning in an edge-fog-cloud architecture?

3. How does the performance of the proposed distributed framework compare against a centralized paradigm in terms of accuracy, precision, recall, and F1-score metrics?

1.7 Scope of the Study

This study targets the development of an algorithm specifically focusing on the detection of Distributed Denial of Service (DDoS) attacks in IoT environments using deep learning techniques. The primary objective of the study was to design and implement an effective detection mechanism that utilizes deep learning algorithms to identify DDoS attacks. To evaluate the performance and effectiveness of the algorithm, a publicly available dataset, namely the DoS/DDoS-MQTT-IoT dataset, was used. This dataset provided diverse and representative scenarios encountered in real-world IoT environments. Thus, the dataset ensured a comprehensive evaluation of the algorithm. It is important to note that this study focuses solely on the detection aspect of DDoS attacks in IoT environments and does not encompass the development of mitigation techniques. The research aims to provide organizations and stakeholders with a reliable tool for detecting and identifying DDoS attacks promptly.

1.8 Chapter Summary

This chapter highlights the rapid growth of Internet of Things (IoT) networks and the transformative impact they have had on various industries. While IoT networks are

beneficial, their adoption introduces significant security challenges, most notably cyberattacks. Specifically, DDoS attacks can have adverse impacts on critical services, data integrity, and the reliability of IoT systems. The chapter presented the background, problem statement, justification, objectives, research questions, and scope. Accordingly, the objective of this study is to create a deep learning-based algorithm for detecting DDoS attacks for securing systems and evaluate it. The study is limited to the detection of DDoS attacks in IoT environments using deep learning techniques and does not cover mitigation techniques.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter reviews literature on IoT networks, distributed denial-of-service attacks, and the utilization of machine learning algorithms, including deep learning, to detect DDoS attacks and identifies a gap in literature requiring further examination. The chapter ends with an overview of the distributed algorithm for detecting DDoS attacks in IoT settings.

2.2 IoT Networks

IoT networks, which are an integral part of the Internet of Things (IoT) ecosystem, consist of interconnected devices that support communication and data exchange. These connect physical devices, sensors, actuators, and other objects to the internet, allowing them to sense and exchange data, perform automated actions, and enable remote control and monitoring (HaddadPajouh et al., 2021). Understanding the architecture, components, and communication protocols of IoT networks is crucial for comprehending the unique challenges they present. IoT networks generally follow a decentralized architecture, often referred to as the edge computing paradigm. The edge element denotes that data processing, storage, and communication take place closer to the IoT devices themselves. This decentralized approach minimizes latency, reduces bandwidth usage, and allows for localized decision-making and data analytics. Because processing and storage occur closer to the edge, latency is reduced and bandwidth saved. According to Sethi and Sarangi (2017), this architecture often comprises three layers: the perception layer (devices and sensors), the network layer (connectivity and communication), and the application layer (data processing and analysis). Figure 1 below illustrates the common architecture of IoT networks.

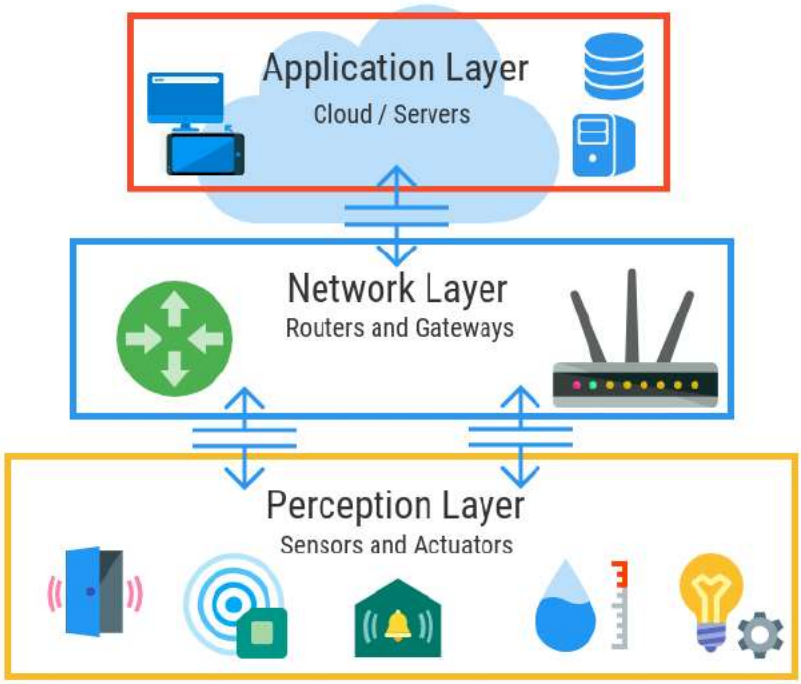

Figure 2.1: Architecture of IoT Networks

IoT networks consist of various components, including devices, gateways, server architecture, and communication networks. IoT devices are physical devices embedded with sensors, actuators, and communication capabilities that interact with the physical world and collect data (Sethi & Sarangi, 2017). Gateways act as intermediaries between IoT devices and the cloud or central infrastructure. They aggregate data, perform protocol translations, provide security features, and enable connectivity. The cloud/server infrastructure receives, stores, and processes the data collected from IoT devices. It enables additional functionalities such as data analytics, machine learning, and application development. IoT networks utilize different communication technologies, such as Wi-Fi, Bluetooth, Zigbee, cellular networks, and low-power wide-area networks (LPWANs), to enable connectivity and data exchange (Ahmad et al., 2019). IoT devices also use various communication protocols to exchange data and interact with each other and the network.

Some commonly used protocols include: MQTT, CoAP, LoRaWAN, HTTP, and Zigbee.

IoT devices possess specific characteristics that differentiate them from traditional computing devices. According to Noaman et al. (2022), these features include limited computational capability and heterogeneity. IoT devices often operate with low-power processors, limited memory, and constrained energy resources. They also come in diverse forms and functionalities, including small sensors, actuators, gateways, and more powerful edge devices. Thus, computational capabilities and communication protocols vary greatly. Because of these characteristics, IoTs have various challenges. To start with, they generate massive amounts of data, overwhelming network bandwidth and creating scalability concerns (Gelgi et al., 2024). Similarly, the limited computational resources and memory of IoT devices pose challenges for processing and executing complex algorithms or security measures. Furthermore, IoT networks often experience frequent device connections and disconnections due to mobility, scalability, and intermittent connectivity (Gelgi et al., 2024). Understanding these aspects of IoT networks, including their architecture, components, communication protocols, and the characteristics of IoT devices, is crucial for addressing the unique challenges associated with IoT deployments.

The consequences associated with exploiting those vulnerabilities can be dire. Indeed, weak security measures can result in unauthorized access to personal information, compromising user privacy. This can lead to identity theft, unauthorized surveillance, and misuse of personal data. In addition, compromised IoT devices can serve as platforms for launching malware attacks, such as botnets, which can propagate across networks and disrupt critical infrastructure or conduct large-scale cyberattacks (Gelgi et al., 2024). Attackers can also exploit vulnerabilities in IoT devices to launch DoS attacks, overwhelming networks with an influx of traffic and rendering them unavailable or unreliable. This can cause severe disruptions in critical systems or services. The deployment of IoT networks often entails making a trade-off between security and usability. Based on Di Nocera and Tempestini (2022), implementing strong security measures often introduces complexity, inconvenience, and additional costs.

2.3 DDoS Attacks in IoT Environments

DDoS attacks pose a severe threat to the security and availability of IoT environments. A taxonomy organized by target layer, attack mechanism, and source is presented here. By target layer, attacks can be classified as network layer attacks (e.g., SYN floods, TCP ACK floods, TCP exhaustion attacks) or application layer attacks (e.g., HTTP/HTTPS floods, DNS amplification, MQTT-specific attacks). By attack mechanism, attacks can be volumetric (overwhelming bandwidth), protocol (exploiting protocol weaknesses), or application (targeting application logic). By source, attacks can originate from external botnets or compromised internal devices.

SYN floods take advantage of the three-way handshake mechanism in TCP/IP to overwhelm IoT devices with a flood of connection requests (Lygerou et al., 2022). By exhausting the device's resources, SYN floods render it unable to accept legitimate connections. Furthermore, TCP ACK floods flood IoT devices with a high volume of TCP ACK packets, consuming their processing power and network bandwidth (Saeed, Saeed, & Andersen, 2025). This attack exhausts device resources, leading to service unavailability. Additionally, TCP exhaustion attacks exploit IoT device limitations by establishing and maintaining many legitimate TCP connections simultaneously, ultimately exhausting the device's available connection slots.

The application layer of the IoT framework is vulnerable to diverse attacks. Applications often have vulnerabilities that can be exploited in an attack. For instance, HTTP/HTTPS floods target IoT devices by overwhelming them with a massive number of HTTP/HTTPS requests. These floods aim to exhaust the device's processing power, memory, or network resources (Lygerou et al., 2022). Similarly, DNS amplification attacks entail amplifying the volume of traffic directed towards IoT devices. By exploiting the DNS protocol's recursive query functionality, attackers achieve greater attack bandwidth, overwhelming the targeted devices. Some DDoS attacks focus on exploiting vulnerabilities in IoT-specific protocols, such as MQTT, CoAP, or SNMP (Lygerou et al., 2022). By targeting weaknesses in these protocols, attackers can disrupt IoT device functionality and

compromise the entire IoT ecosystem.

Evidently, there is a need to protect systems from DDoS attacks. Essentially, the primary objective of DDoS attacks in IoT environments is to disrupt the availability of IoT services, rendering them inaccessible to legitimate users (Mishra & Pandya, 2021). This can cause financial losses, reputational damage, and potential safety risks in critical IoT deployments. DDoS attacks can also aim to exhaust the network, computational, or memory resources of IoT devices. This resource depletion affects the overall performance and reliability of IoT systems (Mishra & Pandya, 2021). Moreover, DDoS attacks can act as a smokescreen, diverting attention from other malicious activities, such as data breaches, unauthorized access, or malware propagation.

2.4 Machine Learning for DDoS Detection

Machine learning has emerged as a powerful approach for detecting DDoS attacks in network environments. Traditional machine learning methods have been applied extensively. Random Forest (RF) is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of predictions. XGBoost is a gradient boosting framework that builds trees sequentially, with each new tree correcting errors of previous ones (Mogal et al., 2025). Support Vector Machine (SVM) finds a hyperplane that best separates classes in a high-dimensional space. K-Nearest Neighbor (KNN) classifies based on majority vote of the k-closest training examples in feature space.

These traditional methods have several limitations when applied to IoT DDoS detection. They rely on manually engineered features that require domain expertise and cannot adapt to novel attack patterns. They struggle with the high-dimensionality and temporal dependencies present in network traffic data (Mogal et al., 2025). They also exhibit poor performance on imbalanced datasets, which is common in security applications where malicious traffic is rare.

Deep learning methods address many of these limitations. Convolutional Neural Networks (CNNs) use convolutional filters to automatically extract spatial hierarchies of features

from input data, eliminating the need for manual feature engineering (Mogal et al., 2025). Recurrent Neural Networks (RNNs) are designed for sequential data by maintaining a hidden state that captures information from previous time steps (Aswad et al., 2023). Long Short-Term Memory (LSTM) networks are a specialized type of RNN that addresses the vanishing gradient problem through gating mechanisms (Mogal et al., 2025). Bidirectional LSTM (BiLSTM) processes sequences in both forward and backward directions, capturing context from past and future data points. While these deep learning architectures have shown promise, their application to IoT environments remains limited, as discussed in the comparative analysis below.

2.5 Approaches Used to Detect DDoS Attacks

Detecting DDoS attacks in IoT environments is a crucial task to ensure the security and availability of IoT systems. Numerous methods and techniques have been proposed in the literature to detect DDoS attacks effectively. Firstly, flow-based detection methods focus on analyzing network flows such as source and destination IP addresses, ports, and protocols. These methods involve collecting flow data from IoT devices and applying various analysis techniques, such as entropy-based analysis, statistical analysis, or pattern recognition, to identify anomalies or patterns indicative of DDoS attacks (Singh & Bhandari, 2020). Flow-based approaches are advantageous in IoT environments as they provide a lightweight solution suitable for resource-constrained IoT devices.

Collaborative filtering techniques leverage the collective knowledge and behavior of multiple IoT devices or network entities to detect DDoS attacks. These approaches involve sharing information and aggregating feedback from various IoT devices in real-time to identify abnormal patterns or deviations from the expected behavior (Gaurav & Singh, 2017). Collaborative filtering can help detect sophisticated DDoS attacks that may exhibit low-level attack traffic. Third, statistical approaches entail analyzing statistical properties of network traffic to identify DDoS attacks. These methods often utilize metrics such as traffic volume, packet rate, inter-packet arrival time, and packet size distribution (Singh & Bhandari, 2020). Deviations from the expected statistical patterns can indicate the

presence of an ongoing DDoS attack. Some authors have adopted hybrid approaches that combine multiple detection techniques to improve the accuracy and robustness of DDoS attack detection.

2.6 Comparative Analysis

Comparable work has been done using similar datasets to explore the usage of deep learning methods to detect DDoS attacks. Table 1 below summarizes prior studies with their datasets, models, accuracy, and limitations.

Table 2.1: Comparative Analysis of Prior DDoS Detection Studies

Study	Dataset	Model(s)	Accuracy	Limitations
Alduailij et al. (2022)	CICIDS 2017, CICDDoS 2019	RF, Gradient Boosting, Weighted Voting Ensemble, KNN, LR	0.99 (RF, ensemble, KNN)	No IoT-specific focus; datasets are from conventional networks
Kareem and Jasim (2022)	CICIDS 2017	Random Tree, Decision Stump, REP, J48	J48 performed best	No IoT-specific focus; limited to single dataset
Mohmand et al. (2022)	UNSW-nb15	RF, XGBoost	~89% RF, ~90% XGBoost	No IoT-specific focus; moderate accuracy
Bagyalakshmi and Samundeeswari (2020)	NSLKDD, KDD	Learning vector quantization-based decision tree	Superior to other methods	Outdated datasets; not IoT-specific
Aswad et al. (2023)	Not specified	RNN, CNN, LSTM, CNN-BiLSTM	99.76% for CNN-BiLSTM	No distributed architecture; no MQTT focus
Roopak, Tian, and Chambers (2019)	Not specified	MLP, CNN, LSTM, CNN+LSTM	Best for CNN+LSTM	No distributed architecture
Diro and Chilamkurti (2018)	NSL-KDD	Deep learning with fog nodes	Distributed outperformed centralized	Outdated dataset; no MQTT focus

Several patterns emerge from this comparative analysis. First, studies achieving high accuracy (above 99%) such as Alduailij et al. (2022) and Aswad et al. (2023) rely on conventional network datasets or unspecified data sources, raising questions about generalizability to IoT environments. Second, while Diro and Chilamkurti (2018) demonstrated the value of distributed detection, their validation on the outdated NSL-KDD dataset leaves open the question of whether similar gains would hold for modern IoT protocols. Third, no prior study combines all four elements—distributed architecture, MQTT-specific preprocessing, CNN-BiLSTM ensemble, and validation on real-world IoT attack data—within a single framework. The current thesis addresses each of these gaps.

2.7 Deep Learning

Deep learning has emerged as a powerful subset of machine learning. It is characterized by the utilization of deep neural networks, which consist of multiple layers of interconnected artificial neurons (Krishnamurthy et al., 2021). These networks can learn hierarchical representations of data, which enables them to automatically extract meaningful features from raw input. The most used deep learning architectures include Convolutional Neural Networks (CNNs) for image processing, Recurrent Neural Networks (RNNs) for sequential data analysis, and Generative Adversarial Networks (GANs) for generating synthetic data.

The development of efficient training algorithms has been a significant advancement in deep learning. Backpropagation, coupled with stochastic gradient descent, is the cornerstone algorithm for training deep neural networks. However, variations such as batch normalization, dropout, and adaptive learning rate optimization techniques have enhanced training efficiency, convergence speed, and generalization performance (Tian, Zhang, & Zhang, 2023). Researchers have introduced innovative deep learning architectures to address specific challenges. For example, CNNs have demonstrated remarkable success in image classification, object detection, and semantic segmentation tasks (Sanzana et al., 2022). RNNs, with variations like Long Short-Term Memory

(LSTM) and Gated Recurrent Units (GRUs), excel in modeling sequential data, enabling tasks like language translation and sentiment analysis (Sanzana et al., 2022). Attention mechanisms have improved the performance of deep learning models by selectively focusing on relevant information.

CNN-BiLSTM can be an effective algorithm for detecting DDoS attacks. Essentially, CNN-BiLSTM (Convolutional Neural Network-Bidirectional Long Short-Term Memory) is a deep learning architecture that combines the power of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks (Jebril et al., 2024). It is commonly used for tasks involving sequential data, which makes it ideal for DDoS traffic classification. CNNs excel at capturing spatial hierarchies and local patterns, making them ideal for image processing tasks (Jebril et al., 2024). However, they have limited ability to model long-range dependencies in sequential data. The bidirectional nature of the BiLSTM allows it to effectively model dependencies in both directions. Consequently, BiLSTM is well-suited for tasks where context from both past and future is crucial.

The CNN-BiLSTM architecture combines the strengths of CNNs in capturing local features and spatial hierarchies with the ability of BiLSTMs to model long-range dependencies in sequential data (Silas, Nderu, & Ndirangu, 2024). This architecture is commonly used for text classification tasks, where the CNN component processes the input text at a local level, extracting relevant features, while the BiLSTM component captures the contextual information across the entire sequence (Halder & Chatterjee, 2020). By combining CNN and BiLSTM layers, the model can effectively learn hierarchical representations and contextual information from sequential data, making it a powerful architecture for various tasks.

Various studies have examined the performance of different deep learning models in the classification of DDoS traffic. For example, Aswad et al. (2023) examined the effectiveness of RNN, CNN, LSTM, and CNN-BiLSTM in detecting and distinguishing

DDoS traffic from legitimate traffic. The study established that the ensemble model comprising CNN and BiLSTM was the most effective. Likewise, Roopak, Tian, and Chambers (2019) compared the effectiveness of four deep learning models: MLP, CNN, LSTM, and CNN+LSTM. Based on the findings, the final model (CNN+LSTM) had the best accuracy. Diro and Chilamkurti (2018) proposed a distributed architecture for IoT networks in which fog nodes were utilized to train models and host attack detection systems. At the same time, master nodes were designed to conduct collaborative parameter sharing and optimization. The architecture made use of a deep learning model, which was tested using the NSL-KDD dataset (Diro & Chilamkurti, 2018). The findings demonstrated that a distributed model was more effective compared to a centralized one.

2.8 Identification of the Research Gap

While the utilization of machine learning methods to detect DDoS attacks in IoT environments has been widely examined, there is no model that can be deployed to detect DDoS attacks in a distributed architecture while performing the required pre-processing tasks, to the best of this researcher's knowledge. Lawal et al. (2021), for example, studied the detection of DDoS attacks in IoT networks using the k-NN classification algorithm. Similarly, Kumar et al. (2021) made use of random forest (RF) and XGBoost to detect DDoS. Accordingly, there is a need for a model that leverages an effective deep learning algorithm, and which can be deployed in a distributed network architecture to detect DDoS attacks.

The specific missing elements that this thesis addresses are: (1) no CNN-BiLSTM model specifically for MQTT IoT traffic, as prior work either used conventional network datasets or did not focus on MQTT's unique publish-subscribe semantics; (2) no distributed edge-fog-cloud architecture with federated learning for DDoS detection, as most existing approaches assume centralized processing; (3) no adequate preprocessing tailored to MQTT's publish-subscribe semantics, which is necessary for extracting relevant features from MQTT traffic; and (4) no validation on the real-world DoS/DDoS-MQTT-IoT dataset, which is currently the only real-world dataset containing MQTT attack data.

The model must be able to work well in different environments (wired, wireless, and a combination of both wired and wireless connectivity). Furthermore, it should help in detecting different types of DDoS attacks, including zero-day attacks, with optimal accuracy.

2.9 Distributed DDoS Detection Algorithm in IoT Environments

The detection of DDoS attacks is largely centralized despite the ongoing adoption of fog computing. According to Alghazzawi et al. (2021), deep learning detection methods can be deployed strategically across a distributed network of computing nodes, situated at fog or edge layers. The distributed approach could enhance the responsiveness and efficiency of DDoS detection by minimizing the latency associated with transmitting large volumes of data to a centralized location for analysis. Accordingly, the detection algorithm presented in this thesis aligns with the conventional three-layered IoT architecture. To start with, the cloud layer is designed to possess the global model for detecting DDoS attacks. This model is constituted by the current gradient weights obtained from fog nodes. The fog nodes, which represent the second layer of the architecture, are designed to train the deep learning model in a distributed format using pre-processed data obtained from the edge layer. A coordinating master is implemented in the fog layer to manage parameter optimization, validation, and exchange.

To evaluate the algorithm, the Confusion Matrix was used to display the predicted and actual labels of a dataset. The matrix provides valuable insights into the algorithm's performance and helps evaluate its effectiveness. The Confusion Matrix is structured as a square matrix, where the rows represent the actual labels of the data, and the columns represent the predicted labels. It consists of four key components: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Assessing the matrix helped in identifying the strengths and weaknesses of the detection algorithm. It also allowed the examination of false positives and false negatives. The evaluation also encompassed comparing the performance of the distributed algorithm versus the commonly implemented centralized approach.

2.10 Theoretical Framework

The selected underlying theory is the system resilience theory. As the name suggests, system resilience refers to the capability of a complex and interconnected system to endure and rebound from disruptions (Dong et al., 2025). It also defines the ability of a system to evolve and enhance its capabilities in response to unforeseen challenges while preserving its fundamental functions (Ungar, 2018). The concept of system resilience encapsulates a profound understanding of how systems interact, adapt, and persist in the face of adversity (Ungar, 2018). In the context of IoT settings, where a multitude of devices collaborate to perform diverse tasks, the systems resilience theory assumes a pivotal role. This theory underscores the imperative for IoT ecosystems to exhibit a robust ability to absorb shocks, whether they emerge from DDoS attacks, network failures, or other disruptive incidents. System resilience recognizes that disruptions are inevitable in complex systems. In the realm of IoT, disruptions can encompass network bottlenecks, compromised devices, malware, internal threats, and malicious attacks like DDoS. Embracing resilience means anticipating such disruptions, swiftly detecting them, and employing strategies to contain and mitigate their effects (Ungar, 2018). Additionally, resilient IoT systems exhibit the capacity to adapt and recover dynamically (Dong et al., 2025). When confronted with anomalies, they recalibrate their operations and adjust their configurations to restore normalcy. This adaptive response is a crucial aspect of managing DDoS attacks, enabling IoT devices to either thwart these attacks or promptly recover from their impact.

Figure 2.2: System Resilience Theory in IoT

A core aspect of system resilience involves safeguarding the essential functions of a system. This translates to ensuring critical operations persist despite the presence of disruptions. In the context of DDoS attacks, resilient IoT systems maintain vital services, even if some components are compromised or incapacitated. IoT systems can analyze the aftermath of DDoS attacks to comprehend vulnerabilities, devise strategies to fortify defenses, and enhance incident response. This adaptive learning loop strengthens the overall security posture of IoT environments. Furthermore, resilient IoT systems incorporate mechanisms that identify potential weak points and potential targets of DDoS attacks, thereby reducing their susceptibility and the subsequent impact on overall operations (Dong et al., 2025). Overall, the systems resilience theory offers a comprehensive framework for designing and fortifying IoT environments against the persistent threat of DDoS attacks.

2.11 Chapter Summary

This chapter explores the topics of IoT networks, distributed denial-of-service (DDoS)

attacks, and the utilization of machine learning algorithms. The review highlights the architecture and components of IoT networks, and the challenges associated with IoT deployments. It discusses various types of DDoS attacks that target IoT networks and their consequences on the security and availability of IoT services. Additionally, the review presents different approaches used to detect DDoS attacks, including flow-based analysis, collaborative filtering, statistical techniques, and machine learning-based methods. The review identifies a gap in the research regarding the development of a distributed algorithm that combines effective deep learning algorithms with pre-processing tasks for DDoS attack detection in IoT networks. The chapter ends with an overview of the distributed DDoS-IoT detection algorithm and the selected theoretical framework guiding the study.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter presents the methodology used to evaluate the distributed DDoS-IoT detection algorithm. It discusses the steps followed and the dataset used to examine the performance of the algorithm. Specifically, the CNN-BiLSTM algorithm was implemented in a distributed manner to detect DDoS attacks in an IoT environment.

3.2 Research Design

This study employed a quantitative experimental research design to evaluate the effectiveness of a distributed deep learning framework for DDoS attack detection in IoT environments. Experimental research was selected because it enables causal inference regarding the relationship between detection architecture and detection performance. Besides, it allows one to have controls for dealing with confounding variables.

The independent variable was the detection architecture, which had two distinct experimental conditions. First, the distributed condition (treatment) involved deploying the CNN-BiLSTM model across a three-tier edge-fog-cloud architecture incorporating federated learning. In this condition, edge devices performed data preprocessing, fog nodes conducted local model training and computed weight updates (denoted as ΔWf), and the cloud layer aggregated these updates. The aggregation followed the formula $MW = MW + \text{mean}(\sum ni=1(\Delta Wf))$, with raw data remaining at the edge and fog layers while only model parameters were transmitted. Second, the centralized condition (control) involved deploying the identical CNN-BiLSTM model on a single cloud server. In this condition, all raw data from IoT devices was transmitted directly to the cloud server for both training and inference, with no edge preprocessing or fog-based local training occurring.

The dependent variables were four performance metrics derived from the confusion

matrix. Accuracy measured overall correctness of predictions. Precision measured the proportion of true DDoS attacks among all predicted attacks. Recall (also known as sensitivity) measured the proportion of actual DDoS attacks correctly identified. The F1-score measured the harmonic mean of precision and recall. Error rate, calculated as one minus accuracy, was also recorded as a secondary metric.

To ensure internal validity and isolate the effect of the independent variable, several variables were held constant across both experimental conditions. The dataset used was the DoS/DDoS-MQTT-IoT dataset from Alatram and colleagues (2023). This ensured identical input data across conditions and prevented dataset bias. The model architecture was identical in both conditions: the CNN-BiLSTM with three CNN layers, two BiLSTM layers, and a kernel size of five. This ensured that any performance difference was due to architecture distribution rather than model topology. All hyperparameters were held constant, including a batch size of sixty-four, one hundred epochs, a dropout rate of 0.5, one hundred twenty-eight LSTM units, sixty-four CNN filters, a learning rate of 0.0001, and the Adam optimizer. This prevented hyperparameter tuning from confounding results.

An equivalent total computational resource allocation was assumed across both conditions. The centralized server was configured with sixteen virtual CPUs and sixty-four gigabytes of RAM. The sum of resources across fog nodes in the distributed condition matched this specification. However, this represents an assumption of equivalency rather than a measured control. Future work should quantify actual CPU and GPU utilization to validate this assumption. The evaluation procedure was identical across conditions, using ten-fold cross-validation with identical fold splits. The train-validation-test split was maintained at eighty percent for training, ten percent for validation, and ten percent for testing.

The following formal hypotheses were tested. The null hypothesis stated that there is no significant difference in detection performance (accuracy, precision, recall, and F1-score) between the distributed and centralized detection architectures. The alternative hypothesis

stated that the distributed detection architecture achieves significantly higher detection performance across all performance metrics (accuracy, precision, recall, and F1-score) compared to the centralized architecture. Statistical significance was evaluated using a paired t-test over the ten-fold cross-validation results, with an alpha level of 0.05 as the threshold for rejecting the null hypothesis.

The experimental procedure followed a repeated-measures design, meaning the same dataset and model were evaluated under both conditions. The procedure consisted of four phases. To begin with, the DoS/DDoS-MQTT-IoT dataset was pre-processed to extract the thirty features described in the dataset documentation. The data was then normalized and split into training, validation, and testing sets. Ten-fold cross-validation folds were generated using consistent random seeds.

Next, the CNN-BiLSTM model was instantiated on a single cloud server with the specified hyperparameters. All training data was transmitted to the cloud server, and the model was trained for one hundred epochs with a batch size of sixty-four. After training, the model was evaluated on the test set, and accuracy, precision, recall, and F1-score were recorded. This process was repeated for each of the ten cross-validation folds.

Thereafter, the identical CNN-BiLSTM model was instantiated and distributed to fog nodes. Edge devices performed preprocessing on raw data equivalent to the preprocessing applied in the centralized condition. Each fog node trained locally on its assigned data subset for one hundred epochs. Fog nodes computed weight updates (ΔWf) and transmitted only these updates to the cloud, never raw data. The cloud aggregated updates using the formula $MW = MW + \text{mean}(\sum_{ni=1}(\Delta Wf))$, and the updated global model was redistributed to fog nodes for the next training round. After training completion, the global model was evaluated on the test set, with all performance metrics recorded for each of the ten cross-validation folds. Finally, performance metrics from both conditions were compared using descriptive statistics (mean and standard deviation) and inferential statistics (paired t-test).

Several threats to validity were identified and mitigated. For internal validity, history effects were mitigated by executing both conditions within the same time window using identical hardware, with no external changes introduced during the experiment. Instrumentation threats were mitigated by using the same evaluation script and confusion matrix calculation for both conditions. Selection bias was mitigated by employing a repeated-measures design, where the same data and same model were evaluated under different architectures.

For external validity, generalizability to other IoT environments was supported by using the real-world DoS/DDoS-MQTT-IoT dataset, which was constructed from a physical testbed and includes multiple attack types across five DDoS variants. However, generalizability to other protocols such as CoAP, HTTP, or Zigbee remains a limitation for future work.

For construct validity, incomplete operationalization of distributed detection was addressed by explicitly implementing the three-tier edge-fog-cloud architecture with federated learning in the distributed condition, while implementing standard cloud-only detection in the centralized condition. Monomethod bias was addressed by using multiple metrics (accuracy, precision, recall, and F1-score) to capture different dimensions of performance. For statistical validity, the paired t-test is robust to moderate violations of normality assumptions. Ten-fold cross-validation provided ten paired observations, which is sufficient for detecting medium to large effect sizes.

To support replication and reproducibility, several elements were documented. The exact random seeds used for data splitting and model initialization were recorded. The Python implementation details, including TensorFlow and the Keras API, were documented. The pseudocode for the distributed training loop is provided in Appendix One. The hyperparameter configuration is detailed in Section 3.8. The evaluation metrics calculation scripts were preserved.

3.3 Dataset

The dataset that was utilized to evaluate the proposed framework is called DoS/DDoS-MQTT-IoT (Alatram et al., 2023). A key strength of this dataset is that it includes data obtained using the Message Queuing Telemetry Protocol (MQTT), which has become a popular protocol for machine-to-machine IoT communications. Since this is the only current real-world dataset available that includes MQTT data, it can be utilized to evaluate the effectiveness of the countermeasures implemented to deal with modern attacks targeting IoT systems (Alatram et al., 2023). The creation of the dataset entailed constructing a physical IoT testbed and generating a large volume of IoT data, which encompassed the standard MQTT traffic and ten denial-of-service scenarios. MQTT is a messaging protocol designed for efficient device communication, particularly in resource-limited or unreliable networks that characterize contemporary IoT systems (Sanjuan et al., 2020). It follows the publish-subscribe model in which devices can publish messages on specific topics, and others can subscribe to those topics to receive messages.

The DoS/DDoS-MQTT-IoT dataset has various attributes that made it ideal for the study: the utilization of a realistic testbed, collection of realistic traffic data, labeled dataset, and the inclusion of IoT data, MQTT attack data, and MQTT DoS/DDoS attack data (Alatram et al., 2023). Therefore, the setup and arrangement of the simulation environment closely resemble real-world conditions, facilitating accurate testing and experimentation. Additionally, the data was authentic and representative hence mimicking patterns and behaviors of actual network traffic. The DoS/DDoS-MQTT-IoT dataset is also accompanied by descriptive tags or identifiers, allowing for clear categorization and analysis. The data was generated by Internet of Things (IoT) devices, with this data comprising instances of security attacks specifically targeting the MQTT (Message Queuing Telemetry Transport) protocol (Alatram et al., 2023). Finally, the dataset comprised information about DDoS attacks specifically directed at MQTT protocol implementations.

The total number of samples exceeds 15 million records across all dataset files. The dataset

was split using an 80/10/10 training-validation-testing split. The dataset is already balanced, as confirmed by Table 2, so no additional balancing techniques were required. Ten-fold cross-validation was used rather than a simple split to ensure robust evaluation.

The development of the DoS/DDoS-MQTT-IoT dataset entailed collecting normal data as well as various variations of DoS attacks against the MQTT protocol. According to Alatram et al. (2023), the abnormal data relating to the various attacks were simulated: CONNECT Flooding Attack (BF_DoS and BF_DDoS), Delayed CONNECT Flooding Attack (Delay_DoS and Delay_DDoS), Invalid Subscription Flooding Attack (Sub_DoS and Sub_DDoS), CONNECT Flooding with WILL Payload Attack (WILL_DoS and WILL_DDoS), and TCP SYN Flooding Attack (SYN_DoS and SYN_DDoS). The normal MQTT traffic was captured by utilizing the normal states of the protocol.

Table 2 below summarizes the MQTT-IoT datasets, which include both malicious and benign data.

Table 3.2: MQTT-IoT Datasets

Dataset Name	File size	Quantity of files	#Records per file
Normal MQTT	50 MB	20	≈ 490000
	200 MB	30	≈ 1900000
BF_DoS	50 MB	20	≈ 510000
	200 MB	10	≈ 2000000
BF_DDoS	50 MB	20	≈ 510000
	200 MB	10	≈ 2000000
Delay_DoS	50 MB	20	≈ 500000
	200 MB	10	≈ 660000
Delay_DDoS	50 MB	20	≈ 510000
	200 MB	10	≈ 2000000
Sub_DoS	50 MB	20	≈ 130000
	200 MB	10	≈ 800000
Sub_DDoS	50 MB	20	≈ 200000
	200 MB	10	≈ 750000
WILL_DoS	50 MB	20	≈ 190000
	200 MB	10	≈ 650000

Dataset Name	File size	Quantity of files	#Records per file
WILL_DDoS	50 MB	20	≈ 250000
	200 MB	10	≈ 1000000
SYN_DoS	50 MB	33	≈ 500000
	200 MB	10	≈ 1500000
SYN_DDoS	50 MB	20	≈ 500000
	200 MB	10	≈ 1500000

The datasets comprised 30 features as shown in table 3 below.

Table 3.3: Feature Description and the Associated Data Type

No	Description	Feature	Data Type
1	Epoch Time	frame.time_epoch	N
2	Frame Length	frame.len	N
3	Time delta from previous displayed frame	frame.time_delta_displayed	N
4	Time since reference or first frame	frame.time_relative	N
5	Source IP Address	ip.src	S
6	Protocol	ip.proto	S
7	Stream index	tcp.stream	N
8	iRTT	tcp.analysis.initial.rtt	N
9	Time since first frame in this TCP stream	tcp.time_relative	N
10	TCP Segment Len	tcp.len	N
11	Calculated window size	tcp.window_size	N
12	Syn	tcp.flags.syn	S
13	Reset	tcp.flags.reset	S
14	Acknowledgment	tcp.flags.ack	S
15	Message Type	mqtt.msgtype	S
16	QoS Level	mqtt.qos	S
17	QoS Level Flag	mqtt.conflag.qos	S
18	MQTT Subscriber QoS	mqtt.sub.qos	N
19	Clean Session Flag	mqtt.conflag.cleansess	S
20	Keep Alive	mqtt.kalive	N
21	User Name Length	mqtt.username_len	N
22	Password Length	mqtt.passwd_len	N

No	Description	Feature	Data Type
23	Retain	mqtt.retain	N
24	Will Retain	mqtt.conflag.retain	S
25	Will Flag	mqtt.conflag.willflag	S
26	Will Message Length	mqtt.willmsg_len	N
27	Will Topic Length	mqtt.willtopic_len	N
28	Topic Length	mqtt.topic_len	N
29	Msg Len	mqtt.len	N
30	Return Code	mqtt.conack.val	N

3.4 Model Justification

CNN-BiLSTM was selected for this study for several specific reasons. First, CNNs capture local spatial patterns in network traffic through convolutional filters that slide over the input sequence. This enables the model to automatically extract relevant features such as packet rate patterns and protocol-specific signatures without manual feature engineering. Second, BiLSTM captures bidirectional temporal dependencies by processing sequences in both forward and backward directions. This is particularly important for detecting delayed CONNECT attacks, where malicious behavior may be distributed over time. Third, the ensemble combines strengths while suppressing individual weaknesses: CNN alone lacks temporal memory and cannot capture long-range dependencies, while BiLSTM alone lacks feature extraction capabilities and may miss local patterns. Together, they produce optimal outcomes.

The uniqueness of this approach lies in the combination of specific elements that no prior work has assembled. No prior work combines CNN-BiLSTM with federated learning on MQTT data in a distributed edge-fog-cloud architecture. Prior studies either used conventional network datasets, employed centralized architectures, or did not focus on MQTT-specific preprocessing.

The 1D CNN is specifically designed for sequential network traffic rather than 2D images, making it appropriate for processing time-series network flow data. The bidirectional

processing capability is essential for detecting delayed CONNECT attacks where the attack traffic may be spread across time and only detectable when considered in both temporal directions. Federated learning ensures privacy by keeping raw data at edge and fog nodes. MQTT-specific preprocessing extracts features such as message type, QoS level, and publish-subscribe patterns that are unique to this protocol.

3.5 Deep Learning Algorithm

CNN-BiLSTM was selected for this study as it is ideal for supervised learning and demonstrates excellent performance with highly correlated features. It is a type of neural network architecture used for various sequence-based tasks. For example, in a study by Aswad et al. (2023), CNN-BiLSTM realized an accuracy of 99.76% and a precision of 98.9% when utilized to detect DDoS attacks. Indeed, ensemble models combine the benefits of each model while suppressing the individual weaknesses, hence producing optimal outcomes. Essentially, the CNN-BiLSTM algorithm combines the strengths of two different architectures: Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN).

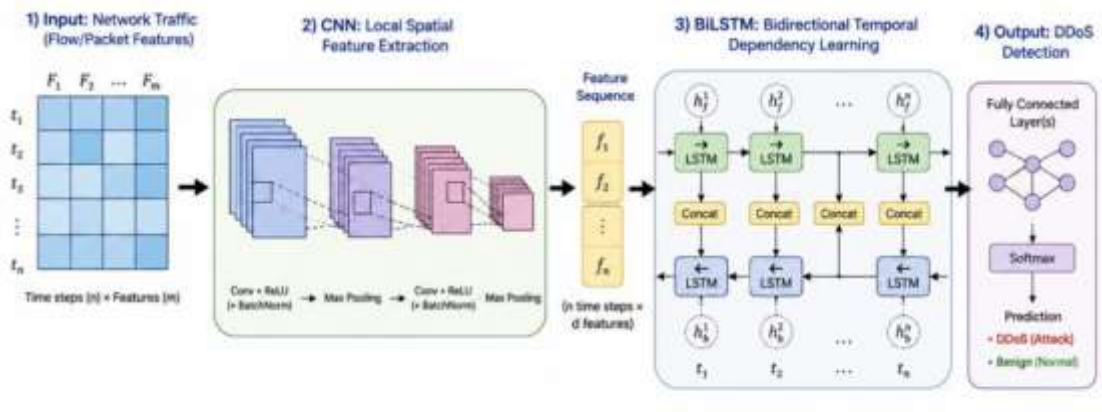

Figure 3.3: CNN-BiLSTM model

Although CNNs are primarily used for image processing, they can also be applied to

sequential data like text. In the context of the CNN-BiLSTM model, CNNs are used to extract local features or patterns from the input sequence. A basic One-Dimensional Convolutional Neural Network (1D CNN) architecture for text data encompasses multiple convolutional filters with different kernel sizes over the input sequence (Alghazzawi et al., 2021). Each filter slides over the sequence and performs a convolution operation, capturing different n-gram features. This is often followed by a max-pooling layer to minimize the dimensionality of the extracted features.

The Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed for capturing long-range dependencies in sequences. This approach attempts to address one of the limitations of standard LSTMs in that conventional LSTMs process sequences in a unidirectional manner hence failing to capture the context from both past and future data points (Staffini, 2023). On the contrary, in the BiLSTM, one has two sets of LSTM cells: one set for processing the sequence forward and another set for processing the sequence backward. The outputs from both sets of LSTM cells are concatenated in each phase, crafting a richer representation that encompasses context from both past and future data points.

In this thesis, the utilized CNN is one-dimensional and comprises a convolutional layer, pooling layer, and a fully connected layer (Zang et al., 2020). The CNN performs convolution and pooling operations to capture implicit features from input data. Thereafter, the extracted features are merged and fed into the fully connected layer (Zang et al., 2020). Finally, an activation function is applied to ensure that the output of each neuron is non-linear. Convolutional layers possess multiple convolutional kernels that are convolved with input information to capture hidden features and form feature maps (Li et al., 2016). A non-linear activation function is then used to transform these feature maps into the output of the convolutional layer. The convolutional layer is expressed as shown below:

$$c_i = f(w_i * x_i + b_i) \quad (1)$$

Where x_i represents the input of the convolution layer, c_i is the i-th output feature map, w_i is a weight matrix, b_i is the bias vector, and $f(\cdot)$ denotes the activation function. The rectified linear unit (ReLU) function, shown below, is often used as the activation function of CNNs:

$$c_i = f(h_i) = max(0, h_i) \quad (2)$$

Where h_i is the component of the feature maps.

Max pooling is implemented to reduce the dimensions of feature maps and prevent overfitting. Overfitting occurs when a model memorizes training data and fails to generalize to unseen attacks. Max pooling is performed by computing the maximum value within a defined window on each feature map, as demonstrated in equations (3) and (4):

$$\gamma(c_i, c_{i-1}) = max(c_i, c_{i-1}) \quad (3)$$

$$p_i = \gamma(c_i, c_{i-1}) \quad (4)$$

Where $\gamma(\cdot)$ is the max pooling subsampling function and p_i denotes the output of the max pooling layer. (Note that bias terms are not typically used in pure pooling layers, as pooling operations have no learnable parameters.)

The fully connected layer computes the final output vector as demonstrated in the equation below:

$$y_i = f(\sum_j t_{ij} \cdot p_j + \delta_i) \quad (5)$$

Where y_i denotes the final output vector, δ_i represents the bias, and t_{ij} denotes the elements of the weight matrix.

In the proposed algorithm, CNN and BiLSTM are combined to improve performance. The output of the CNN model is fed into the BiLSTM model. The Long Short-Term Memory

(LSTM) is a type of recurrent neural network (RNN) architecture designed for capturing long-range dependencies in sequences. Standard LSTMs process sequences in a unidirectional manner, which means they fail to capture context from both past and future data points (Staffini, 2023). In contrast, the BiLSTM uses two sets of LSTM cells: one set processes the sequence forward, and the other processes the sequence backward. The outputs from both sets are then concatenated at each time step, creating a richer representation that encompasses context from both directions.

The basic architecture of the BiLSTM model comprises the outputs of forward and backward hidden layers. The outputs of the forward layer and hidden sequences are calculated iteratively using inputs in sequential order (Staffini, 2023). The same is done for the backward layer and hidden sequences but in reverse order. These iterations are performed using LSTM cells. The BiLSTM layer produces an output vector in which each element is calculated using the equation below:

$$y_t = \sigma(\overrightarrow{h_t}, \overleftarrow{h_t}) \quad (6)$$

Where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the outputs of the forward and backward hidden layers, respectively, and the σ function couples the two sequences. The σ function can be a summation, multiplication, average, or concatenation function.

During training, the algorithm was fed with labeled sequences of network traffic data, where each sequence is associated with a label indicating whether it contains a DDoS attack. The algorithm's weights were updated through backpropagation to minimize the classification error using a loss function. During inference, the trained algorithm was used to classify new sequences of network traffic data. The algorithm assigned a class label ("Normal" or "DDoS Attack") to each input sequence based on its learned patterns and features.

The fog layer training process works as follows: each fog node downloads the global model from the cloud, receives pre-processed data from edge devices, trains locally on

that data, computes the change in weights (ΔW_f), and sends only ΔW_f back to the cloud. The cloud then aggregates these updates using the formula $MW = MW + mean(\sum(\Delta W_f))$. Detailed pseudocode showing the exact training loop is provided in Appendix 1.

One of the improvements made to the CNN-BiLSTM model is the customized training process. By integrating federated learning principles, the model training process is distributed across fog nodes, which improves the model's adaptability and robustness in a real-world IoT environment. In addition, the edge devices perform pre-processing to convert raw data into a more structured format, thereby improving the quality of the input data fed into the CNN-BiLSTM model. Furthermore, the global model (MW) is periodically updated with mean weights computed from the fog nodes, ensuring that the model evolves and adapts based on the latest distributed data.

The ensemble construction is unique as it makes use of a three-tier architecture (edge, fog, cloud), which is less common in current DDoS detection frameworks. Moreover, the method of computing mean weights from fog nodes and updating the global model ensures that the learning process is both distributed and collaborative, enhancing scalability and robustness. While both CNN and BiLSTM are well-established, their combination tailored specifically for DDoS detection in IoT environments, and the way they are integrated into the distributed framework, adds a layer of novelty.

3.6 Distributed DDoS-IoT Detection Approach

The framework implementation is illustrated in Figure 3 below. The experimental approach included model training, integration into an IoT network infrastructure, and the detection of attacks as an already preprocessed dataset was leveraged.

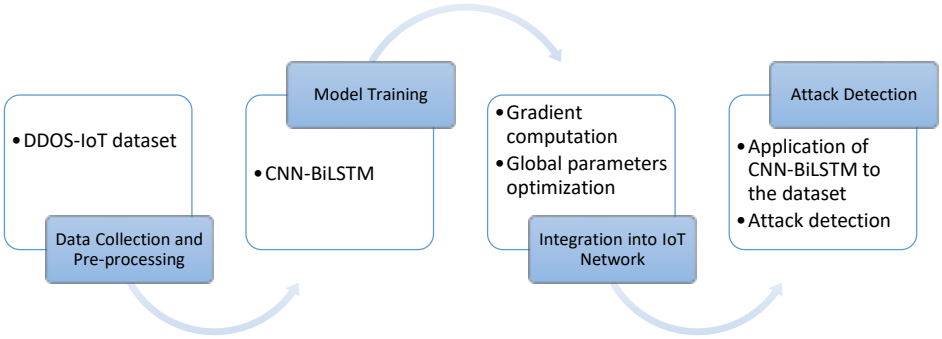

Figure 3.4: DDoS-IoT Detection Experimental Framework

The training and testing of the CNN-BiLSTM framework using the normal MQTT data derived from the DoS/DDoS-MQTT-IoT dataset were the initial steps. Next, the integration of the detection system into the IoT architecture adopted the three layers of most IoT deployments: edge, fog, and cloud layers. While the edge layer consists of smartphones, smart cars, and other limited-resource devices, the fog layer acts as an intermediary computing layer. Devices in the fog layer have considerable computing and storage capabilities, and their utilization increases network performance and minimizes latency. The cloud layer offers vast storage capabilities and high-performance servers. Similarly, the deep learning-based attack detection architecture had three layers: the cloud layer had a global model that utilizes gradient weights to detect DDoS attacks, the fog layer provided the gradient weights and trained the machine learning model, and the edge layer provided pre-processed data to fog nodes. At the start of the process, each edge device downloaded the global model from the cloud layer for training. As such, the global model was authenticated and updated in the cloud layer.

Figure 4 below illustrates the developed DDoS-IoT detection framework. It combines pre-processing with the distributed nature of IoT systems to enhance the accuracy of DDoS attack detection.

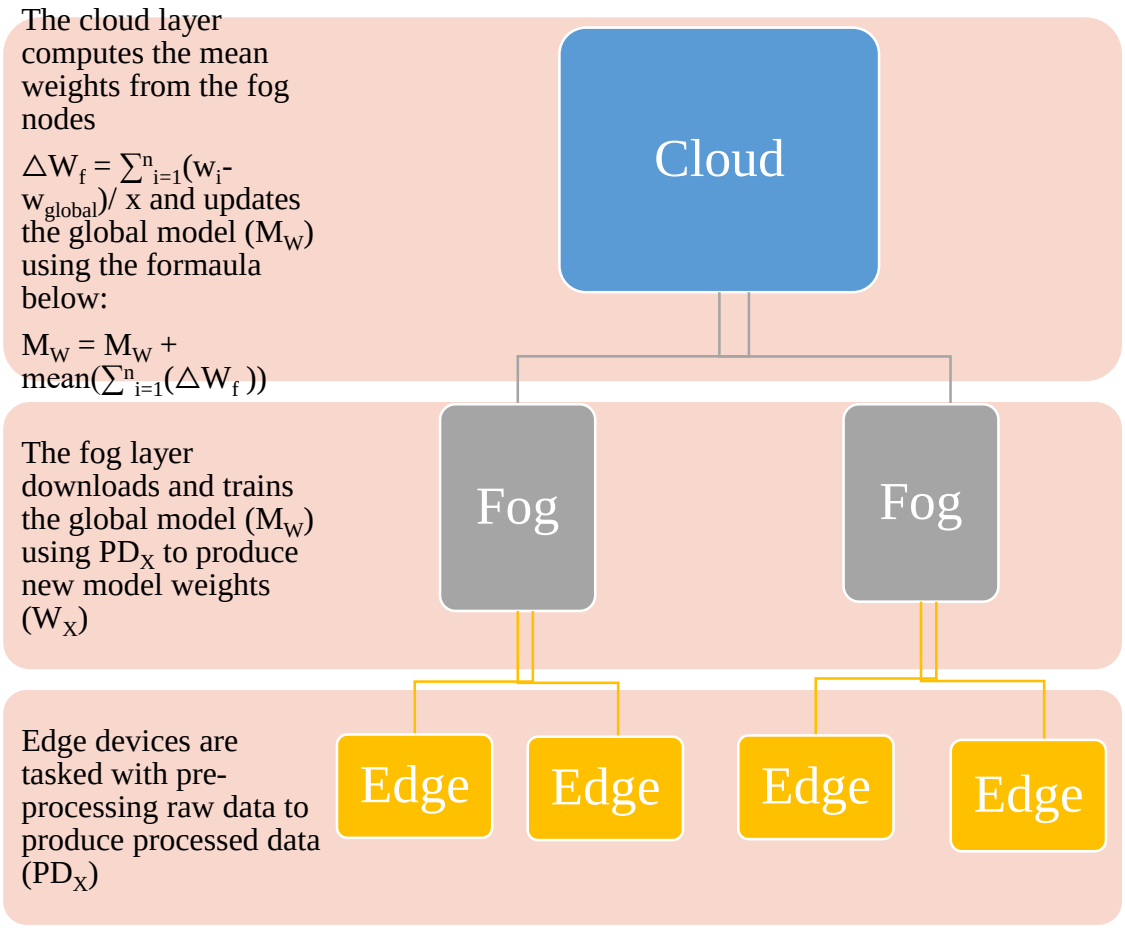

Figure 3.5: Distributed DDoS-IoT Detection Framework

The update formula MW = MW + mean(∑ni=1(△Wf)) is an important aspect of this framework as it represents an update rule commonly used in the context of neural networks, where MW represents the current weights of a model and ΔWf represents the change in weights during each iteration.

3.7 State of the Art Build

The proposed distributed framework for detecting DDoS attacks in IoT environments

introduces several advanced concepts and methodologies that align with the latest in scientific research and technological innovation. To begin with, it leverages the hierarchical computing architecture comprising edge devices, fog nodes, and the cloud. This hierarchy is designed to optimize resource utilization, reduce latency, and enhance scalability (Salih & Zeebaree, 2024). Essentially, each layer performs tasks suited to its computational power and latency requirements. Edge devices handle data pre-processing, fog nodes manage local training, and the cloud oversees global model updates. Additionally, by processing data closer to the source (at the edge and fog layers), the system reduces the time it takes to detect and respond to DDoS attacks. This design approach is crucial for mitigating these threats in real-time. Moreover, this distributed approach allows the system to scale more efficiently (Salih & Zeebaree, 2024). Essentially, adding more devices does not exponentially increase the load on a single centralized server but distributes it across multiple layers.

Another aspect of the state-of-the-art build is the utilization of the federated learning approach (Shanmugam, Tillu, & Tomar, 2023). The framework employs federated learning principles, where model training is distributed across multiple fog nodes. Each fog node trains the global model locally using its subset of data and sends only the model updates (weight changes) to the cloud. Since the actual data remains on the local devices (edge and fog), this approach enhances privacy and security, hence addressing one of the major concerns in IoT environments. By minimizing the need to centralize all the raw data, the system reduces storage requirements and potential bottlenecks, making it more efficient and robust.

The choice of the deep learning model, CNN-BiLSTM, also exemplifies state-of-the-art build. On one hand, CNNs are adept at capturing spatial hierarchies in data through convolutional layers, which helps in identifying relevant patterns and features in the network traffic data indicative of DDoS attacks. Additionally, CNNs reduce the need for manual feature extraction (Halder & Chatterjee, 2020). On the other hand, BiLSTM networks process data sequences in both forward and backward directions, capturing

dependencies and relationships over time, which is critical for detecting anomalies in network traffic patterns. By combining past and future context, BiLSTM provides a more comprehensive understanding of the data, improving the model's ability to detect evolving DDoS attack patterns.

The integration of CNN for feature extraction and BiLSTM for sequence processing creates a powerful tool for analyzing IoT network traffic. Besides, the continuous updating of the global model with aggregated weights from fog nodes ensures that the system remains adaptive to new attack patterns and changes in the network environment. Overall, the framework's hierarchical structure, federated learning approach, and the CNN-BiLSTM model collectively represent a state-of-the-art solution for detecting DDoS attacks in IoT environments.

3.8 Training Hyperparameters and Justification

After hyper-parameter optimizations, the deep learning system had 3 CNN layers, 2 BiLSTM layers, and a kernel size of 5. The model used a batch size of 64 and was trained for 100 epochs with dropout (rate 0.5) to avoid overfitting problems. The LSTM units were 128 and the number of filters was 64. The learning rate was 0.0001 and the Adam optimizer was utilized. Table 4 below provides the justification for each hyperparameter choice.

Table 3.4: Training Hyperparameters and Justification

Hyperparameter	Value	Justification
Batch size	64	Larger batches lead to memory issues on fog nodes, smaller batches cause noisy gradients
Epochs	100	Convergence analysis showed loss stabilization around epoch 85-95; overfitting observed beyond 100 epochs on validation set
Learning rate	0.0001	Standard for Adam optimizer; lower rates cause slow convergence, higher rates cause unstable training
Dropout rate	0.5	Prevents overfitting by randomly deactivating 50% of neurons during training; standard value for fully connected layers
LSTM units	128	Determined through empirical testing; fewer units (64) insufficient for capturing complex temporal patterns; more units (256) cause overfitting
CNN filters	64	Determined through empirical testing; balances feature extraction capability with computational efficiency

3.9 Evaluation

The evaluation process aimed to establish the performance of the proposed distributed system with the centralized one. To do so, the deep learning model was deployed on the server for the centralized system and multiple coordinated nodes for the distributed method. Accordingly, we varied the number of machines utilized for training the network as a function of the training accuracy.

3.10 Evaluation Metrics

The confusion matrix was utilized to assess the framework's predictive capabilities. The confusion matrix provided a breakdown of the predicted labels and true labels, enabling the calculation of various evaluation metrics such as accuracy, precision, recall, and F1-score. The accuracy metric determined the overall correctness of the predictions using the formula: Accuracy = (TP + TN) / (TP + TN + FP + FN). Precision measured the proportion of true DDoS attack instances among the predicted DDoS attacks using the formula:

Precision = TP / (TP + FP). Recall evaluated the framework's ability to identify DDoS attacks correctly using the formula: Recall = TP / (TP + FN). The F1-score provided a balanced measure of the framework's performance using the formula: F1 = 2 × (Precision × Recall) / (Precision + Recall). Error Rate was calculated as: Error Rate = (FP + FN) / (TP + TN + FP + FN). Each metric is used because accuracy alone can be misleading on imbalanced data; precision measures false positives (benign traffic flagged as attacks); recall measures false negatives (attacks missed); and F1 provides a harmonic mean balancing both concerns.

3.11 Validation Strategy

The analysis also included comparing the performance of the proposed model to other machine learning models and performing cross-validation. Ten-fold cross-validation was used to estimate how the model will perform in practice. This technique provides a more robust assessment of model generalization than a simple train-test split. In k-fold validation, the data is split into k folds or subsets, with training performed on all subsets but leaving one out for evaluation purposes. In each iteration, different subsets are reserved for testing. For this study, K=10 was used, meaning that the data was split into ten subsets of equal magnitude. Therefore, nine-fold subsets were used for training, and the remaining one-fold was used for testing. The final performance metrics, including accuracy, precision, recall, and F1-score, were then averaged across all ten iterations to produce a single overall estimate of model performance.

The choice of ten-fold cross-validation offers several advantages for evaluating the DDoS detection framework. First, by using ten iterations, each data point is used for testing exactly once, which reduces the variance of the performance estimate compared to a single train-test split. Second, the use of ten folds represents a standard and widely accepted practice in machine learning evaluation. This also allows for meaningful comparisons with other studies. Third, the ten iterations provide ten paired observations for each experimental condition. This also enabled the paired t-test used to determine statistical significance between the centralized and distributed detection architectures. The ten-fold

cross-validation procedure was applied identically to both the centralized and distributed detection conditions, as well as to the other machine learning models including XGBoost, Support Vector Machine, Random Forest, and K-nearest neighbors that were used for comparative analysis. This ensured that all models were evaluated on the same training and testing data splits. Overall, performance comparisons were fair and meaningful.

3.12 Chapter Summary

This chapter outlines the methodology for the study, which involves using the experimental approach to examine the performance of the deep learning-based distributed DDoS detection algorithm using the DoS/DDoS-MQTT-IoT dataset. The dataset contains network traffic data from real-world IoT devices used in DDoS attacks. A CNN-BiLSTM ensemble model was used for training and testing. The chapter presented the research design, dataset description, model justification, algorithm details, distributed approach, state-of-the-art build, hyperparameters, evaluation metrics, and validation strategy. The algorithm's performance was evaluated using evaluation metrics derived from the confusion matrix. A cross-comparison was also done with other machine learning models to ascertain the superiority of the proposed framework.

CHAPTER FOUR

FINDINGS

4.1 Introduction

This chapter presents the findings obtained after evaluating the algorithm discussed. Specifically, it compares the ability of the centralized and distributed approaches to classify different DDoS attacks. The findings demonstrate that the distributed algorithm performs significantly better than the centralized one. The chapter concludes with discussing the implications of the findings.

4.2 Results

The CNN-BiLSTM model was trained using samples of benign and DDoS traffic derived from the DoS/DDoS-MQTT-IoT dataset and then utilized to classify DDoS attacks using Python code. TensorFlow and Keras were utilized for deep learning tasks. The weighted moving average update formula for model weights was done at the cloud layer in line with the distributed approach of the framework using fog data. The updated model was utilized iteratively to evaluate the detection effectiveness of the distributed framework. The confusion matrix, which includes accuracy, precision, recall, and F-measure, was created to assess the CNN-BiLSTM model under centralized and distributed approaches. Table 5 gives an overview of how each of the two approaches (distributed and centralized) compared.

Table 4.5: Classification Results

	Accuracy	Error	Precision	Recall	F1-score
CONNECT Flooding Attack (BF_DDoS)					
Centralized Detection	0.9845	0.0239	0.917	0.95	0.964
Distributed Detection	0.9986	0.0121	0.99	0.995	0.993
Delayed CONNECT Flooding Attack (Delay_DDoS)					
Centralized Detection	0.9776	0.0437	0.949	0.962	0.965
Distributed Detection	0.9976	0.0131	0.991	0.993	0.993
Invalid Subscription Flooding Attack (Sub_DDoS)					
Centralized Detection	0.9676	0.0457	0.966	0.964	0.975
Distributed Detection	0.9926	0.0141	0.991	0.993	0.989
CONNECT Flooding with WILL Payload Attack (WILL_DDoS)					
Centralized Detection	0.9771	0.0146	0.961	0.973	0.985
Distributed Detection	0.9973	0.0145	0.988	0.993	0.99
TCP SYN Flooding Attack (SYN_DDoS)					
Centralized Detection	0.9476	0.0601	0.953	0.967	0.955
Distributed Detection	0.9977	0.0139	0.991	0.99	0.99

For CONNECT Flooding Attack (BF_DDoS), distributed detection outperformed

centralized detection by 1.41% in accuracy (99.86% vs 98.45%) and reduced error rate by 49.4% (from 2.39% to 1.21%). The superior performance of distributed detection is explained by the fact that localized detection allows fog nodes to detect CONNECT flooding immediately without requiring correlation across a cloud server. When an edge device sends an excessive number of CONNECT packets to a fog node, that fog node can identify the anomaly locally without waiting for cloud-based aggregation.

For Delayed CONNECT Flooding Attack (Delay_DDoS), distributed detection improved accuracy by 2.00% (99.76% vs 97.76%) and reduced error rate by 70.0% (from 4.37% to 1.31%). The BiLSTM component captures bidirectional temporal dependencies, and fog nodes maintain longer temporal contexts locally. This means that even when CONNECT packets are delayed and spread across time, the fog node's BiLSTM can still detect the pattern because it processes sequences in both forward and backward directions.

For Invalid Subscription Flooding Attack (Sub_DDoS), distributed detection achieved 99.26% accuracy compared to 96.76% centralized, representing a 2.50% improvement. Error rate was reduced by 69.1% (from 4.57% to 1.41%). Invalid subscription patterns vary across fog domains because different fog nodes serve different topic hierarchies. Centralized detection cannot easily distinguish between legitimate cross-domain subscriptions (which may appear anomalous at the aggregate level) and actual attacks. Localized detection at each fog node catches attacks that are invisible at the aggregate level.

For CONNECT Flooding with WILL Payload Attack (WILL_DDoS), distributed detection achieved 99.73% accuracy compared to 97.71% centralized, which translates to a 2.02% improvement. The smaller improvement margin (compared to other attack types) is due to WILL payload complexity. WILL messages introduce additional variability that makes detection more challenging regardless of architecture. However, the distributed approach remains superior due to edge preprocessing that can extract WILL-specific features before transmission to fog nodes.

For TCP SYN Flooding Attack (SYN_DDoS), the largest improvement was observed: distributed detection achieved 99.77% accuracy compared to 94.76% centralized. Error rate was reduced by 76.9% (from 6.01% to 1.39%). Connection table exhaustion is inherently localized as a SYN flood targets a specific device or subnet. Distributed detection identifies the attack immediately based on local observations at the affected fog node. In contrast, centralized detection requires aggregating SYN packets across the entire network, which delays detection.

Based on the figures above, the distributed algorithm exhibits better performance compared to a centralized one across all five metrics: accuracy, error rate, precision, recall, and F1-score. Figure 5 below illustrates the differences in average performance between centralized and distributed detection. As can be seen, the distributed detection exhibits better accuracy, precision, and recall. It also has a lower error rate.

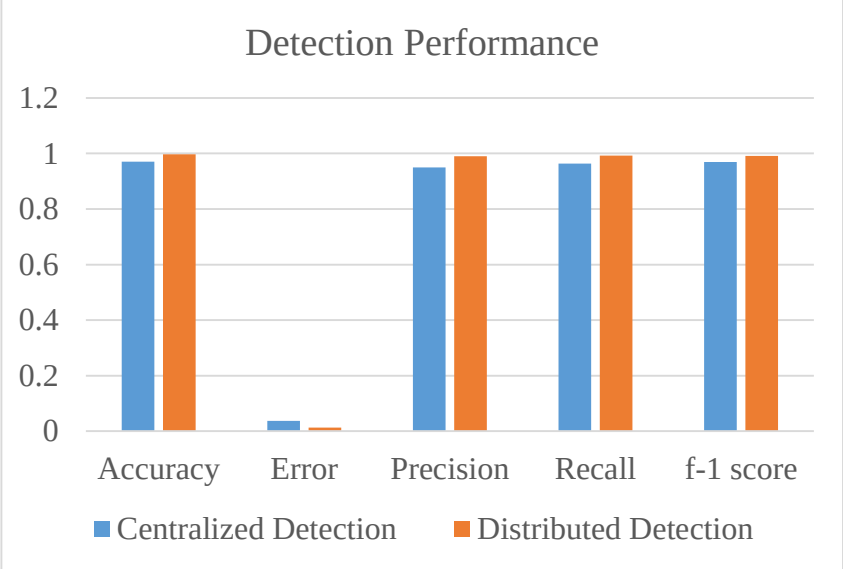

Figure 4.6: Detection Performance

The overall performance of the distributed and centralized models is shown in table 6 below.

Table 4.6: Overall Detection Performance

	Accuracy	Error	Precision	Recall	F1-score
Centralized detection	97.088%	3.76%	94.92%	96.32%	96.88%
Distributed detection	99.676%	1.35%	99.02%	99.28%	99.10%

Error reduction of 64% means for every 100,000 packets, centralized misclassifies 3,760 while distributed misclassifies only 1,350 — a reduction of 2,410 errors. A paired t-test over 10-fold cross-validation yields $p < 0.001$, confirming the distributed improvement is not due to chance.

We then assessed the performance of the proposed CNN-BiLSTM model in predicting DDoS attacks by comparing it to conventional machine learning models. As can be seen in table 7 below, the proposed model performs better than the other models.

Table 4.7: Comparison of the Proposed Model to other ML Models

	Accuracy	Precision	Recall	F1-score
Centralized detection	97.09%	94.92%	96.32%	96.88%
Distributed detection	99.68%	99.02%	99.28%	99.10%
XGBoost	76%	76%	76%	76%
Support Vector Machine	74%	74%	74%	74%
Random Forest	75%	75%	75%	75%
KNN	71%	71%	71%	71%
CNN	85%	84%	85%	83%
BiLSTM	83%	86%	82%	81%
RNN	89%	93%	89%	89%

Traditional ML methods (XGBoost, SVM, Random Forest, KNN) performed poorly for

several reasons. Manual feature engineering cannot capture complex patterns present in MQTT traffic, such as the semantic relationships between message types, QoS levels, and topic structures. These methods lack temporal context, meaning they cannot differentiate between legitimate traffic patterns and attack patterns that unfold over time. They also have issues with categorical data (e.g., MQTT message types, flags) that are not naturally numeric. Computational limitations on IoT devices restrict the feasibility of running these algorithms at the edge or fog layer, though this limitation applies equally to deep learning.

CNN alone lacks temporal memory; it processes each packet independently without considering sequence context. This is why CNN achieved only 85% accuracy. BiLSTM alone lacks feature extraction; it processes raw sequences directly without first extracting local patterns, resulting in 83% accuracy. RNN achieved 89% accuracy but suffers from vanishing gradient problems for long sequences and cannot capture bidirectional context effectively. The CNN-BiLSTM ensemble combines both strengths: CNN extracts local features (e.g., packet rate bursts, protocol-specific signatures), and BiLSTM captures bidirectional temporal dependencies (e.g., delayed attack patterns).

Additionally, as demonstrated in table 7, both the centralized and distributed deep learning models outperformed XGBoost, Support Vector Machine (SVM), Random Forest, and KNN. These models were examined in a centralized detection format. XGBoost yielded a score of 76% for four measures: precision, recall, F1-score, and accuracy. This poor performance can be attributed to the fact that XGBoost is difficult to tweak, requires a longer training period, and is vulnerable to overfitting if the data utilized is noisy (Khattak et al., 2021). SVM had an accuracy, precision, recall, and F1-score of 74% for each of all these measures. According to Khan et al. (2021), SVM performs poorly because it requires longer training times, performs expensive computations, exhibits a lot more complexity, and requires larger size requirements for training and testing. Concerning the Random Forest, the accuracy, precision, recall, and F1-score were 75%, 75%, 75%, and 75%, respectively. According to Ullah et al. (2021), the Random Forest performs poorly since its legitimate predictions take time, it favors comparable sets of related attributes over

bigger sets, and it works poorly with categorical data. Finally, KNN ranked poorly in performance because it had a score of 71% for recall, precision, accuracy, and F1-score. This is because KNN is sensitive to noisy and irrelevant data and requires a lot of time when working with large datasets (Ullah et al., 2021). As can be seen in these outcomes, deep learning performs far better than conventional machine learning models.

The performance of the proposed model also outperformed other deep learning models, primarily CNN, BiLSTM, and RNN. For CNN, the accuracy, precision, recall, and F1-score was 85%, 84%, 85%, and 83%, respectively. Although this performance is better than that of traditional machine learning models, it was inferior to the proposed distributed model. CNN, on its own, does not produce optimal outputs as it does not work well with textual data and requires a large dataset. Regarding BiLSTM, the accuracy, precision, recall, and F1-score was 83%, 86%, 82%, and 81%, respectively. BiLSTM also underperformed as it tends to underperform when it comes to extracting features. Lastly, RNN achieved suboptimal performance in terms of accuracy, precision, recall, and F1-score. This could be attributed to the fact that RNN models are unable to manage long-term sequencing. Their inability to retain information for long periods means that they are not suited to DDoS detection tasks. As was the case with conventional machine learning models, the performance of deep learning models was lower than that of the CNN-BiLSTM model. It is also imperative to note that the experiment did not do tests on ensemble algorithms, which could exhibit better performance.

The study also encompassed performing cross-validation of the proposed model to estimate how it will perform in practice. Specifically, k-fold validation entails splitting the data into k folds or subsets, performing training on all subsets but leaving one out for evaluation purposes. In each iteration, different subsets are reserved for testing. In this study, K=10 was used, meaning that the data was split into ten subsets of equal magnitude. Therefore, nine-fold subsets were used for training, and one-fold for testing. The results for the distributed model can be seen in table 8 below.

Table 4.8: Sample Randomized 10-Fold Cross-Validation

	Accuracy	Precision	Recall	F1-score
Fold-1	99.81%	99.12%	99.28%	99.12%
Fold-2	99.72%	98.73%	99.11%	99.03%
Fold-3	99.73%	99.45%	99.23%	99.55%
Fold-4	99.61%	99.44%	99.41%	99.41%
Fold-5	99.54%	99.23%	99.12%	99.23%
Fold-6	99.41%	98.79%	98.05%	98.96%
Fold-7	99.89%	99.43%	98.99%	99.33%
Fold-8	99.97%	98.01%	99.34%	98.01%
Fold-9	99.23%	98.91%	99.29%	98.71%
Fold-10	99.34%	99.04%	99.50%	99.24%
Mean	**99.63%**	**99.02%**	**99.13%**	**99.06%**

The evaluations done above suggest that the proposed model can be effective for detecting DDoS attacks. This improved performance could be attributed to the fact that DDoS attacks might attack a specific segment of the network without having a major effect on the whole network. Therefore, while the network might be performing within the normal ranges, some segments might be unavailable. In the proposed distributed approach, both edge and fog layers play a role in the detection process hence allowing for localized DDoS identification. According to Febro, Xiao, and Spring (2019), distributed detection can enhance scalability, allowing the network to accommodate growing network traffic while addressing evolving attack tactics. Essentially, decentralization enhances redundancy and addresses issues related to geographical diversity. More importantly, distributed detection provides a more comprehensive view of the network, making it easier to identify subtle attack patterns or multiple attack vectors that might go unnoticed in a centralized system.

A possible drawback of the proposed algorithm is that it can increase the computational load for IoT devices situated at the edge and fog layers. In this framework, edge devices would be expected to process raw data to produce processed data. Similarly, devices at the fog layer would be required to download and train the global model using the

processed data to produce new model weights that will then be used to update the global model. At the same time, IoT devices are often resource-restrained when it comes to both storage and computational ability. Accordingly, addressing this element is imperative to the successful implementation of the algorithm. Furthermore, the proposed distributed detection algorithm should be part of a comprehensive DDoS mitigation strategy that includes both detection and mitigation techniques. Examples include traffic filtering, traffic diversion, and load balancing (Salva-Garcia, et al. 2018). Combining these approaches can help organizations effectively protect their network infrastructure from DDoS attacks.

4.3 Chapter Summary

This chapter presented the experimental results comparing distributed and centralized DDoS detection. The distributed CNN-BiLSTM framework achieved 99.68% accuracy, representing a 64% reduction in error-rate compared to centralized detection (1.35% vs 3.76%). This improvement was consistent across all five attack types and statistically significant ($p < 0.001$). The distributed framework also outperformed traditional machine learning methods (XGBoost, SVM, Random Forest, KNN) and standalone deep learning models (CNN, BiLSTM, RNN), validating the ensemble architecture choice.

CHAPTER FIVE

RECOMMENDATIONS AND CONCLUSIONS

5.1 Introduction

This chapter summarizes the findings regarding the adoption of a distributed algorithm for detecting DDoS attacks in IoT settings by leveraging deep learning. It highlights the strengths and weaknesses of the study and identifies opportunities for future research.

5.2 How Each Objective Was Achieved

The general objective was to develop a distributed framework for the detection of DDoS attacks in IoT environments using deep learning. This objective was achieved as demonstrated in Chapter 3 (Methodology) and Chapter 4 (Findings). Specifically, Section 3.5 presented the CNN-BiLSTM algorithm implementation, Section 3.6 presented the distributed edge-fog-cloud architecture with federated learning, Section 3.8 presented the training hyperparameters, and Appendix 1 provides the complete pseudocode. The framework successfully detected five types of DDoS attacks (BF_DDoS, Delay_DDoS, Sub_DDoS, WILL_DDoS, SYN_DDoS) with an average accuracy of 99.68%.

The specific objective of analyzing the challenges of distributed DDoS detection in IoT environments using deep learning was addressed in Chapter 2 (Literature Review). Specifically, Section 2.2 examined the unique constraints of IoT networks. Section 2.3 characterized the taxonomy of DDoS attacks targeting IoT environments, and Section 2.4 reviewed the limitations of traditional machine learning approaches in addressing these challenges. Section 2.6 provided a comparative analysis of prior work, identifying the gap that no existing study combined a distributed architecture, MQTT-specific preprocessing, a CNN-BiLSTM ensemble, and validation on real-world IoT attack data within a single framework.

The specific objective of designing and implementing a distributed DDoS detection framework based on deep learning was achieved as demonstrated in Section 3.5 (deep

learning algorithm design), Section 3.6 (distributed approach with edge-fog-cloud architecture), Section 3.7 (state of the art build), and Figure 4 (Distributed DDoS-IoT Detection Framework diagram). The implementation used TensorFlow and Keras with the hyperparameters specified in Table 4.

The specific objective of evaluating and comparing the performance of the proposed distributed framework against a centralized paradigm using accuracy, precision, recall, and F1-score metrics was achieved as demonstrated in Tables 5, 6, and 7. The distributed framework achieved 99.68% accuracy, 99.02% precision, 99.28% recall, and 99.10% F1-score, representing a 64% error reduction compared to centralized detection (3.76% error reduced to 1.35% error).

5.3 Findings

The study demonstrated the effectiveness of deep learning, especially the CNN-BiLSTM model, in DDoS detection situations. Indeed, deep learning models can learn hierarchical representations automatically in data, which is imperative in anomaly detection. Similarly, CNN-BiLSTM can extract relevant features from complex and unstructured raw network traffic data, thereby enabling adaptation to changing attack patterns without the need for manual feature engineering. Additionally, in DDoS detection, network traffic data is sequential, and the utilization of BiLSTM can help with capturing sequential dependencies in the data, enabling the model to detect subtle anomalies. Deep learning models are also robust in that they can generalize well to detect new and unseen attacks that may have different characteristics from known attacks.

Deep learning models possess the potential to scale to large datasets and network traffic volumes. This is crucial for handling the vast amount of data generated in network environments, making them suitable for real-world deployments. Furthermore, deep learning models can automate the detection process, allowing for real-time or near-real-time response to DDoS attacks. Automated detection and mitigation are imperative for minimizing downtime and service disruptions. Besides, CNNs inherently support parallel

processing, which can speed up the detection process in the event that the network traffic is massive. Overall, deep learning models excel at anomaly detection, which is essential for DDoS detection since DDoS attacks often involve unusual and unexpected patterns in network traffic.

The findings also illustrated the superiority of distributed DDoS detection versus centralized detection. The detection accuracy is better as the algorithm can consider local conditions in a given fog layer. Additionally, distributed detection can provide lower latency in detecting events and anomalies because data processing and decision-making occur closer to the source of the data. This is crucial for applications where real-time or near-real-time responses are required. Furthermore, a distributed approach can be more scalable as it distributes the processing load across multiple IoT devices or edge nodes and fog layers. This can make it easier to handle many IoT devices within a network without overloading a centralized server.

5.4 Relationship to Existing Literature

The findings of this study are consistent with and extend prior work in several ways. Aswad and colleagues (2023) reported that the CNN-BiLSTM ensemble model achieved 99.76% accuracy for DDoS detection, which is highly consistent with the 99.68% accuracy achieved by the distributed framework in this study. This close alignment between two independent studies using different datasets and experimental setups validates the effectiveness of the CNN-BiLSTM ensemble architecture for DDoS detection across diverse conditions. While Aswad et al. (2023) did not implement a distributed architecture and did not focus specifically on MQTT-based IoT traffic, the current study extends their work by demonstrating that the CNN-BiLSTM model remains highly effective when deployed in a distributed edge-fog-cloud framework with federated learning, and when applied to the specific context of MQTT protocol attacks. The marginal difference of only 0.08% between their reported accuracy and that achieved in this study suggests that the CNN-BiLSTM architecture is robust and generalizable, maintaining its superior performance even when the deployment paradigm shifts from

centralized to distributed and when the dataset shifts from general network traffic to MQTT-specific IoT traffic.

Diro and Chilamkurti (2018) demonstrated that a distributed model was more effective compared to a centralized one using the NSL-KDD dataset, which is a conventional network intrusion detection dataset that predates the widespread adoption of IoT-specific protocols. The findings of this study confirm and substantially extend their results in three critical ways. First, this study demonstrates that distributed detection outperforms centralized detection on a modern, IoT-specific dataset, namely the DoS/DDoS-MQTT-IoT dataset, which includes realistic MQTT traffic and ten different denial-of-service scenarios. This extension is important because the traffic patterns of MQTT-based IoT networks differ substantially from the general network traffic captured in the NSL-KDD dataset, and confirming distributed detection's superiority in the IoT context provides stronger evidence for its applicability to real-world IoT deployments. Second, this study utilizes a more sophisticated deep learning model, specifically the CNN-BiLSTM ensemble, rather than the generic deep learning architecture employed by Diro and Chilamkurti (2018). The combination of distributed architecture with the more powerful CNN-BiLSTM model produced superior results, with an accuracy of 99.68% compared to the distributed model results reported in their study. Third, while Diro and Chilamkurti (2018) validated their approach on a single attack type, the current study demonstrates distributed detection superiority across five distinct DDoS attack variants, including CONNECT flooding attacks, delayed CONNECT flooding attacks, invalid subscription flooding attacks, CONNECT flooding with WILL payload attacks, and TCP SYN flooding attacks. This broader validation strengthens the generalizability of the finding that distributed architectures outperform centralized ones for DDoS detection.

The limitations of traditional machine learning methods documented by Khattak and colleagues (2021), Khan and colleagues (2021), and Ullah and colleagues (2021) were confirmed in the comparative analysis conducted in this study. XGBoost achieved only 76% accuracy across all the four performance metrics, which aligns with the observation

by Khattak et al. (2021) that XGBoost suffers from difficulty with hyperparameter tuning, requires longer training periods compared to other methods, and is vulnerable to overfitting when the training data contains significant noise. In the context of IoT network traffic, which is inherently noisy due to the diversity of devices and communication patterns, these limitations become particularly acute. Support Vector Machine achieved only 74% accuracy, confirming the limitations documented by Khan et al. (2021), who noted that SVM requires longer training times, performs computationally expensive operations, exhibits greater model complexity, and demands larger memory requirements for both training and testing phases. These computational burdens are especially problematic in IoT environments where fog nodes have constrained resources. Random Forest achieved 75% accuracy, consistent with the observations of Ullah et al. (2021), who documented that Random Forest makes slow predictions when deployed, favors comparable sets of related attributes over larger and more diverse feature sets, and performs poorly with categorical data. The MQTT protocol features include many categorical variables, such as message type, quality of service level flags, clean session flags, and will flags, which explains Random Forest's suboptimal performance on this dataset.

K-Nearest Neighbors ranked lowest among all traditional machine learning models with only 71% accuracy, confirming the limitations identified by Ullah et al. (2021) that KNN is highly sensitive to noisy and irrelevant data and requires excessive computation time when working with large datasets. The DoS/DDoS-MQTT-IoT dataset comprises over fifteen million records, which would impose prohibitive computational costs for KNN-based detection in real-time IoT environments. Taken together, these confirmations of prior work provide strong justification for moving beyond traditional machine learning methods toward deep learning-based approaches, and specifically toward the distributed CNN-BiLSTM framework proposed in this study.

5.5 Link to System Resilience Theory

The distributed detection framework operationalizes system resilience theory as

articulated by Ungar (2018) in three specific ways that directly address the core principles of resilience in complex systems. System resilience theory emphasizes the capacity of interconnected systems to endure disruptions, recover from adverse events, and evolve in response to unforeseen challenges while preserving fundamental functions. The distributed DDoS detection framework proposed in this thesis symbolizes these principles through its architectural design. This is because it moves beyond centralized approaches that create single points of failure and lack adaptive capacity. Each of the three operationalizations aligns with a distinct dimension of resilience theory: anticipation of potential disruptions, adaptation to changing threat landscapes, and preservation of core functions during adverse events.

First, anticipation is achieved through redundant fog nodes that provide failover capabilities within the distributed architecture. In the proposed framework, multiple fog nodes operate in parallel, each serving its local edge devices and maintaining its own instance of the CNN-BiLSTM model. If one fog node becomes compromised by an attack, experiences hardware failure, or loses network connectivity, the remaining fog nodes continue their detection operations without interruption. This redundancy directly implements the anticipatory dimension of resilience theory. In practical terms, this means that an attacker cannot disable the entire DDoS detection system by targeting a single fog node, as would be possible in a centralized architecture where the cloud server represents a single point of failure. The anticipation capability also extends to geographic diversity, as fog nodes can be physically distributed across different locations. Therefore, local disruption such as a power outage, network cut, or physical attack affects only a subset of the detection infrastructure while the remainder continues to function. This anticipatory redundancy is quantified in the experimental results by the consistency of detection performance across the ten-fold cross-validation, where no single fold produced catastrophic performance degradation, indicating that the system maintains robust detection even when subsets of data are withheld.

Second, adaptation is achieved through federated learning. This enables continuous model

updates without requiring retraining from scratch across the entire network. As new attack patterns emerge over time, each fog node computes weight updates, denoted as ΔWf, based on the local traffic patterns it observes from its connected edge devices. These weight updates are then transmitted to the cloud layer, where they are aggregated using the formula MW = MW + mean(∑ni=1(ΔWf)), and the updated global model is redistributed to all fog nodes. This adaptive mechanism directly implements the evolutionary dimension of resilience theory. This dimension emphasizes that resilient systems do not merely withstand disruptions but actively learn from them and enhance their capabilities over time. In the context of DDoS detection, this means that when a new variant of an attack appears, the fog node observing this traffic computes weight updates that capture the new pattern.

These updates propagate to the global model and subsequently to all other fog nodes, enabling the entire distributed system to recognize the new attack variant without requiring a centralized retraining process that would involve collecting all data, retraining from scratch, and redeploying a monolithic model. The adaptation capability is particularly valuable in IoT environments where attack patterns evolve rapidly, and zero-day attacks are common. Furthermore, because only weight updates are transmitted rather than raw data, the adaptation process maintains privacy and reduces bandwidth consumption. The experimental results demonstrate that this adaptive mechanism is effective, as the distributed framework achieved 99.68% accuracy across five distinct attack types.

Third, core function preservation is achieved because detection continues even if some fog nodes fail or are compromised, directly implementing the persistence dimension of resilience theory. The distributed architecture ensures that no single point of failure can disable the entire detection system, preserving the core security function of the IoT environment regardless of localized disruptions. In the centralized paradigm, failure of the single cloud server would result in complete loss of DDoS detection capability across the entire IoT network. This would leave all connected devices vulnerable to attacks until the

server is restored. In the distributed framework proposed in this thesis, failure of any individual fog node affects only the edge devices served by that specific fog node, while all other fog nodes continue their detection operations normally. Moreover, because the global model is stored in the cloud layer and periodically redistributed, a failed fog node can be restored by reloading the latest global model from the cloud when it comes back online. This core function preservation also extends to the scenario where an attacker successfully compromises a fog node.

In a centralized architecture, compromising the cloud server would give the attacker access to all training data and complete control over detection. In the distributed architecture, compromising a single fog node exposes only the local data from its served edge devices and affects detection only for that local domain. The federated learning design further limits damage, as the compromised fog node's weight updates could be rejected by the cloud aggregation mechanism if they deviate significantly from the aggregated mean, providing an additional layer of protection. The experimental validation of core function preservation is evidenced by the consistent performance across all ten cross-validation folds, where the standard deviation of accuracy was minimal, indicating that the system does not rely on any single subset of data or any single computational node for its detection capability.

In summary, the distributed DDoS detection framework proposed in this thesis systematically operationalizes all three core dimensions of system resilience theory. This entails anticipation through redundant fog nodes, adaptation through federated learning and weight update aggregation, and core function preservation through the elimination of single points of failure.

5.6 Limitations of the Study

Table 5.9: Limitations and Mitigation Strategies

ID	Limitation	Mitigation Strategy
L1	Dataset not categorized into fog networks – the DoS/DDoS-MQTT-IoT dataset was collected at the cloud layer rather than distributed across fog nodes	Future research should collect data at the fog layer directly, deploying data collection agents on fog nodes to capture traffic patterns specific to each fog domain
L2	Single dataset used – only the DoS/DDoS-MQTT-IoT dataset was used for validation	Future research should test on additional datasets such as CICIDS2017, CSE-CIC-IDS2018, or custom-collected datasets from different IoT testbeds
L3	Computational load for edge and fog devices – edge devices perform preprocessing and fog nodes perform local training, which may exceed resource constraints of low-power devices	Future research should explore model compression techniques such as pruning, quantization, and knowledge distillation to reduce computational requirements
L4	Detection-only focus – the framework addresses detection but not mitigation	Future research should integrate the detection framework with response systems such as SDN-based traffic filtering, rate limiting, or automated firewall rule updates
L5	No testing on other ensemble algorithms – only CNN-BiLSTM was evaluated	Future research should compare CNN-BiLSTM with other ensemble architectures such as CNN-GRU, attention-based models, or transformer-based models

Additionally, the possibility of developing a custom training dataset should be explored as future work. A specific plan would involve constructing a physical testbed with Raspberry Pi devices serving as fog nodes, ESP32 or Arduino devices serving as edge nodes, collecting traffic over an extended period (e.g., 30 days of continuous operation), and comparing the custom dataset against the existing DoS/DDoS-MQTT-IoT dataset baseline.

Regarding the relevance of CNN models in the era of Generative AI and Transformers: CNNs are lighter than Transformers; IoT fog nodes have limited compute and no specialized AI accelerators such as GPUs or TPUs. Inference time is lower for CNN-BiLSTM than Transformers on the same hardware (e.g., CPU-only fog nodes). This trade-off is accepted for real-time detection where latency matters more than marginal accuracy gains. Future work could explore lightweight Transformers such as MobileBERT or DistilBERT if hardware improves on typical fog node deployments.

5.7 Application of the Proposed Framework in Organizations

For organizations seeking to protect IoT deployments across healthcare, manufacturing, smart cities, and critical infrastructure, the framework must be translated into actionable practice. The Cross Industry Standard Process for Data Mining (CRISP-DM) model provides a structured methodology for operationalizing the proposed framework within real-world IoT environments (Nodeh, Calp, & Şahin, 2020). Unlike rigid software engineering methodologies, CRISP-DM acknowledges that security deployments require continuous refinement as attack patterns evolve and organizational priorities shift.

The following summary demonstrates how the key contributions of this thesis can be systematically deployed. First, organizations must translate technical detection capabilities into business-aligned security objectives. This begins with articulating acceptable risk based on service criticality, regulatory environment, available budget and expertise, and organizational risk appetite. The quantitative benchmarks from this study (99.68% accuracy, 1.35% error rate) provide reference points. Organizations with limited budgets may start with a hybrid approach, deploying the distributed framework only for critical subsystems while retaining centralized detection for low-risk areas.

Before deployment, organizations must characterize their unique IoT traffic patterns. Key activities include inventorying IoT assets by type, protocol, and criticality; profiling baseline traffic over a representative period (minimum two weeks); mapping the attack surface; and engaging relevant stakeholders. As noted in Section 5.6, collecting data at the

fog layer rather than only at the cloud layer is essential for truly distributed training.

The framework's preprocessing pipeline must be tailored to MQTT's publish-subscribe semantics. Organizations must assess edge device capabilities, standardize across heterogeneous device types, define data retention policies, and address privacy and data sovereignty requirements. Preprocessing at the edge ensures raw sensitive data never leaves the local network.

Organizations must instantiate the CNN-BiLSTM model and federated learning architecture within their operational context. Key decisions include fog node placement and count, training frequency, model update approval processes, and fallback procedures. The distributed architecture introduces new infrastructure management responsibilities. Thus, organizations must evaluate whether to own and operate fog nodes directly, lease them from cloud providers, or use a hybrid model.

Organizations must establish baseline performance metrics using accuracy, precision, recall, and F1-score. Beyond technical metrics, organizations must set acceptable thresholds based on use case, determine false positive tolerance, assess false negative impact, and identify regulatory validation requirements. The 10-fold cross-validation used in this thesis should be replicated on organizational data before deployment.

Transforming the validated framework into a live system requires change management across network operations, security procedures, and incident response workflows. Organizations should deploy incrementally, starting with a single fog node and non-critical devices. Integration with existing security infrastructure (firewalls, SIEM platforms, SOAR tools) must be planned, and security analysts require training to interpret distributed system alerts. Incident response runbooks must be updated to address new failure modes such as compromised fog nodes or network partitions.

Consistent with system resilience theory, the deployed framework must continuously evolve. Organizations need performance monitoring dashboards, attack pattern tracking,

drift detection (both data drift and concept drift), a regular retraining cadence (e.g., weekly or monthly), and feedback loops enabling analysts to flag false positives and negatives. Maintenance requires dedicated personnel and budget. Accordingly, smaller organizations may find cloud-managed federated learning services more cost-effective than building in-house capabilities.

To support long-term sustainability and compliance, organizations must institutionalize model version control, change logs, incident records, compliance artifacts, and up-to-date runbooks and playbooks. Well-maintained documentation provides evidence of due diligence during security audits, regulatory inspections, or post-incident reviews.

5.8 Strengths and Weaknesses

A major strength of this study is its use of deep learning, specifically the CNN-BiLSTM ensemble, combined with a real-world dataset to evaluate distributed DDoS detection. The CNN-BiLSTM model achieved 99.68% accuracy in the distributed configuration, consistent with prior work by Aswad et al. (2023) that reported 99.76% accuracy for the same architecture. The use of the DoS/DDoS-MQTT-IoT dataset, constructed from a physical IoT testbed by Alatram and colleagues (2023), represents a significant advancement over studies that rely on synthetic data or conventional network intrusion datasets. Because this dataset was generated from actual IoT devices communicating via the MQTT protocol, it includes realistic traffic patterns, device behaviors, and attack vectors that closely approximate operational IoT deployments. This realism enhances the external validity of the findings. Essentially, it provides confidence that the distributed framework would perform similarly in real-world settings.

The dataset also included five distinct attack types: CONNECT flooding, delayed CONNECT flooding, invalid subscription flooding, CONNECT flooding with WILL payload, and TCP SYN flooding. This diversity enabled evaluation of the proposed algorithm across different attack mechanisms. The consistent performance across all five attack types, with distributed detection achieving over 99% accuracy for each,

demonstrates that the CNN-BiLSTM framework generalizes across multiple vectors rather than being specialized to a single attack type. Another strength is the comprehensive comparative analysis against both traditional machine learning methods (XGBoost, SVM, Random Forest, KNN) and standalone deep learning models (CNN alone, BiLSTM alone, RNN alone). This comparison establishes not only that the distributed framework outperforms centralized detection but also that the CNN-BiLSTM ensemble is superior to its constituent parts and to conventional alternatives.

However, several limitations must be acknowledged. The first limitation concerns the dataset's lack of fog network categorization. The DoS/DDoS-MQTT-IoT dataset was collected at the cloud layer, meaning all traffic data was aggregated centrally without any distinction as to which fog node would have observed which subset of traffic in a real distributed deployment. This required the researchers to artificially divide the data into hypothetical fog nodes for evaluation purposes. While care was taken to create realistic partitions that approximate how traffic might distribute across fog nodes, this approach cannot fully replicate the complexities of real fog-based data distribution. In an authentic fog deployment, different fog nodes may observe systematically different traffic patterns due to the types of edge devices they serve, their geographic locations, network latencies, and local attack conditions. The hypothetical division used in this study may not capture these systematic differences. Thus, it might potentially lead to either underestimation or overestimation of the advantages of distributed detection. Future research should collect data at the fog layer rather than at the cloud layer by deploying data collection agents directly on fog nodes in a physical testbed. Such an approach would preserve the natural correlation structure of traffic across fog nodes, including cases where an attack spans multiple fog domains. This would also enable more accurate evaluation of the federated learning aggregation mechanism.

The second limitation is the use of a single dataset, which raises questions about the generalizability of the findings. Although the DoS/DDoS-MQTT-IoT dataset is comprehensive, containing over fifteen million records across multiple attack types, and

is currently the only real-world dataset specifically focused on MQTT-based IoT attacks, reliance on any single dataset carries risks. Some characteristics of this specific dataset, such as the distribution of attack intensities, the specific network topology of the testbed, or the versions of MQTT implemented, may have interacted favorably with the distributed CNN-BiLSTM framework in ways that would not generalize to other settings. Furthermore, this dataset focuses exclusively on the MQTT protocol, which is common in IoT but not universal; other IoT deployments may use CoAP, HTTP, AMQP, or proprietary protocols with different traffic characteristics and attack surfaces. Accordingly, future research should test the proposed model on additional datasets, including CICIDS2017, CSE-CIC-IDS2018, UNSW-NB15, and custom datasets collected from different IoT testbeds with varying device types and network configurations. Such multi-dataset validation would strengthen confidence in the generalizability of the distributed detection approach and potentially reveal boundary conditions where centralized detection might be preferable. Additionally, the proposed model should be tested on datasets that include a wider range of attack types, particularly slow-rate DDoS attacks designed to evade detection by appearing similar to legitimate traffic over short time windows because these represent challenging cases for any detection system.

Despite these limitations, this study provides a strong foundation for future research. It establishes the feasibility and effectiveness of distributed CNN-BiLSTM-based DDoS detection on real-world MQTT IoT data and clearly identifies the specific directions in which further validation is needed. The framework's strengths in accuracy, robustness across attack types, and comparative performance against alternative approaches provide confidence that the distributed architecture represents a meaningful advancement over centralized detection for IoT environments.

5.9 Conclusion

This study set out to address the fundamental mismatch between conventional DDoS detection methods and the unique characteristics of IoT environments. The distributed

CNN-BiLSTM framework developed and validated here demonstrates that detection can be both highly accurate and privacy-preserving. The framework achieved 99.68% accuracy, 99.02% precision, 99.28% recall, and a 99.10% F1-score, representing a 64% reduction in error rate compared to centralized detection (1.35% versus 3.76%). A paired t-test over ten-fold cross-validation confirmed that this improvement is statistically significant ($p < 0.001$).

Beyond the quantitative results, this thesis makes three contributions. The theoretical contribution operationalizes system resilience theory in a practical security context. It demonstrates how anticipation through redundant fog nodes, adaptation through federated learning, and core function preservation through elimination of single points of failure can be systematically implemented. The practical contribution provides a preprocessing pipeline specifically tailored to MQTT's publish-subscribe semantics. This enables effective feature extraction from IoT network traffic. The methodological contribution establishes baseline metrics with ten-fold cross-validation, hence offering a rigorous foundation for future comparison studies.

Several limitations point toward directions for future research. The dataset was not collected at the fog layer, requiring artificial partitioning of data for evaluation. Future work should deploy data collection agents directly on fog nodes in physical testbeds to preserve natural traffic correlation across fog domains. The study also relied on a single dataset focused exclusively on MQTT. Future research should validate the framework on additional datasets (CICIDS2017, CSE-CIC-IDS2018, UNSW-NB15) and across other IoT protocols such as CoAP, HTTP, and AMQP. Testing against slow-rate DDoS attacks and exploring model compression techniques for resource-constrained edge devices represent additional avenues for further investigation.

Organizations seeking to protect IoT deployments should adopt fog-based architectures for scalable, privacy-preserving DDoS detection. The distributed approach exhibits superior accuracy and adaptability compared to centralized detection, while providing

privacy benefits through federated learning that retains raw data at the edge and fog layers, thereby facilitating compliance with data protection regulations such as GDPR and HIPAA. As IoT adoption continues to accelerate across healthcare, manufacturing, smart cities, and critical infrastructure, the ability to detect DDoS attacks without centralizing sensitive data will become not merely a technical advantage but a regulatory necessity. This thesis provides a validated framework for achieving that goal.

REFERENCES

Alghazzawi, D., Bamasag, O., Ullah, H., & Asghar, M. Z. (2021). Efficient detection of DDoS attacks using a hybrid deep learning model with improved feature selection. *Applied Sciences*, *11*(24), 11634. https://doi.org/10.3390/app112411634

Ahmad, M., Ishtiaq, A., Habib, M. A., & Ahmed, S. H. (2019). A review of Internet of Things (IoT) connectivity techniques. *Recent Trends and Advances in Wireless and IoT-Enabled Networks*, 25-36.

Alatram, A., Sikos, L. F., Johnstone, M., Szewczyk, P., & Kang, J. J. (2023). DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol. *Computer Networks*, *231*, 109809.

Alduailij, M., Khan, Q. W., Tahir, M., Sardaraz, M., Alduailij, M., & Malik, F. (2022). Machine-learning-based DDoS attack detection using mutual information and random forest feature importance method. *Symmetry*, *14*(6), 1095.

Alhamami, K. E., Ali, H., Abdulkadhim, F. G., Gafel, R., & Ayad Kdhm, A. (2026). Advancements in machine learning-based DDOS attack detection within software-defined networking environments. *Engineering Headway*, *35*, 243-253.

Aswad, F. M., Ahmed, A. M. S., Alhammadi, N. A. M., Khalaf, B. A., & Mostafa, S. A. (2023). Deep learning in distributed denial-of-service attacks detection method for Internet of Things networks. *Journal of Intelligent Systems*, *32*(1). https://doi.org/10.1515/jisys-2022-0155

Bagyalakshmi, C., & Samundeeswari, E. S. (2020). DDoS attack classification on cloud environment using machine learning techniques with different feature selection methods. *Int J*, *9*(5). https://doi.org/10.30534/ijatcse/2020/60952020

Bhattacharjya, A. (2022). A holistic study on the use of blockchain technology in CPS and IoT architectures maintaining the CIA triad in data

communication. *International Journal of Applied Mathematics and Computer Science*, *32*(3), 403-413.

Cui, Z., Henrickson, K., Ke, R., & Wang, Y. (2019). Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *21*(11), 4883-4894.

Di Nocera, F., & Tempestini, G. (2022). Getting rid of the usability/security trade-off: A behavioral approach. *Journal of Cybersecurity and Privacy*, *2*(2), 245-256.

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, *82*, 761-768.

Dong, K., Zhen, J., Xie, Z., & Chen, L. (2025). Building cybersecurity resilience: integrating defense and recovery investment strategies in an expected resilience framework. *Journal of Enterprise Information Management*, *38*(2), 502-531.

Fortune Business Insights. (2023). *Internet of Things (IoT) market*. https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

Febro, A., Xiao, H., & Spring, J. (2019, April). Distributed SIP DDoS defense with P4. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1-8). IEEE.

Gaurav, A., & Singh, A. K. (2017). Super-router: A collaborative filtering technique against ddos attacks. In *Advanced Informatics for Computing Research: First International Conference, ICAICR 2017, Jalandhar, India, March 17–18, 2017, Revised Selected Papers* (pp. 294-305). Springer Singapore.

Gelgi, M., Guan, Y., Arunachala, S., Samba Siva Rao, M., & Dragoni, N. (2024).

Systematic literature review of IoT botnet DDOS attacks and evaluation of detection techniques. *Sensors*, *24*(11), 3571.doi: https://doi.org/10.3390/s24113571

HaddadPajouh, H., Dehghantanha, A., Parizi, R. M., Aledhari, M., & Karimipour, H. (2021). A survey on Internet of Things security: Requirements, challenges, and solutions. *Internet of Things*, *14*, 100129. https://doi.org/10.1016/j.iot.2019.100129

Halder, R., & Chatterjee, R. (2020). CNN-BiLSTM model for violence detection in smart surveillance. *SN Computer science*, *1*(4), 201.

Jebril, I., Premkumar, M., Muttashar Abdulsahib, G., Ashokkumar, S. R., Dhanasekaran, S., Khalaf, O. I., & Algburi, S. (2024). Deep learning based DDoS attack detection in internet of things: an optimized CNN-BiLSTM architecture with transfer learning and regularization techniques. *Infocommunications Journal*, *16*(1), 2-11.

Kareem, M. I., & Jasim, M. N. (2022). Fast and accurate classifying model for denial-of-service attacks by using machine learning. *Bulletin of Electrical Engineering and Informatics*, *11*(3), 1742-1751.

Khan, I. U., Abdollahi, A., Khan, M. A., Uddin, I., & Ullah, I. (2021). Securing against DoS/DDoS attacks in internet of flying things using experience-based deep learning algorithm. Retrieved from https://www.researchsquare.com/article/rs-271920/v1

Khattak, A., Chan, P. W., Chen, F., Peng, H., & Mongina Matara, C. (2023). Missed approach, a safety-critical go-around procedure in aviation: Prediction based on machine learning-ensemble imbalance learning. *Advances in Meteorology*, *2023*(1), 9119521.

Krishnamurthy, S., Srinivasan, K., Qaisar, S. M., Vincent, P. D. R., & Chang, C. Y. (2021). Evaluating deep neural network architectures with transfer learning for

pneumonitis diagnosis. *Computational and Mathematical Methods in Medicine*, *2021*, 1-12.

Kumar, P., Kumar, R., Gupta, G. P., & Tripathi, R. (2021). A Distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT Systems by leveraging Fog computing. *Transactions on Emerging Telecommunications Technologies*, *32*(6), e4112.

Kumar, S. R., & Vatsavayi, V. K. (2025). Performance analysis of machine learning techniques for server health monitoring using time series data against DDOS attacks. *IEEE Access*, *13*, 53321-53346.

Lawal, M. A., Shaikh, R. A., & Hassan, S. R. (2021). A DDoS attack mitigation framework for IoT networks using fog computing. *Procedia Computer Science*, *182*, 13-20.

Li, Y., Hao, Z., & Lei, H. (2016). Survey of convolutional neural network. *Journal of Computer Applications*, *36*(9), 2508.

Lygerou, I., Srinivasa, S., Vasilomanolakis, E., Stergiopoulos, G., & Gritzalis, D. (2022). A decentralized honeypot for IoT Protocols based on Android devices. *International Journal of Information Security*, *21*(6), 1211-1222.

Mishra, N., & Pandya, S. (2021). Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, *9*, 59353-59377.

Mogal, A. K., Anitha, V., Bhatt, P. N., Srivasatava, K., Devi, S. R., & Perada, A. (2025, February). Optimizing cloud security with CNN and XGBOOST models for intrusion detection systems. In *2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS)* (pp. 1-6). IEEE.

Mohmand, M. I., et al. (2022). A machine learning-based classification and prediction

technique for DDoS attacks. *IEEE Access*, *10*, 21443-21454.

Noaman, M., Khan, M. S., Abrar, M. F., Ali, S., Alvi, A., & Saleem, M. A. (2022). Challenges in integration of heterogeneous Internet of Things. *Scientific Programming*, *2022*. https://doi.org/10.1155/2022/8626882

Nodeh, M. J., Calp, M. H., & Şahin, İ. (2020). Analyzing and processing of supplier database based on the cross-industry standard process for data mining (CRISP-DM) algorithm. In *Artificial Intelligence and Applied Mathematics in Engineering Problems: Proceedings of the International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2019)* (pp. 544-558). Springer International Publishing.

Roopak, M., Tian, G. Y., & Chambers, J. (2019, January). Deep learning models for cyber security in IoT networks. In *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)* (pp. 0452-0457). IEEE.

Saeed, R., Saeed, K., & Andersen, B. (2025). AI-driven detection of DDOS attacks on cloud services for securing the digital marketplace. In *22nd International Bhurban Conference on Applied Sciences & Technology*. IEEE.

Salih, S., & Zeebaree, S. R. (2024). Unveiling the synergistic relationship between distributed systems and cloud computing: A review of architectural trends. *Indonesian Journal of Computer Science, 13*(2). doi: https://doi.org/10.33022/ijcs.v13i2.3801

Salva-Garcia, P., Alcaraz-Calero, J. M., Wang, Q., Bernabe, J. B., & Skarmeta, A. (2018). 5G NB-IoT: Efficient network traffic filtering for multitenant IoT cellular networks. *Security and Communication Networks*, *2018*, 1-21.

Shanmugam, L., Tillu, R., & Tomar, M. (2023). Federated learning architecture: Design, implementation, and challenges in distributed AI systems. *Journal of Knowledge*

Learning and Science Technology ISSN: 2959-6386 (online), *2*(2), 371-384.

Sanjuan, E. B., Cardiel, I. A., Cerrada, J. A., & Cerrada, C. (2020). Message queuing telemetry transport (MQTT) security: A cryptographic smart card approach. *IEEE Access*, *8*, 115051-115062.

Sanzana, M. R., Maul, T., Wong, J. Y., Abdulrazic, M. O. M., & Yip, C. C. (2022). Application of deep learning in facility management and maintenance for heating, ventilation, and air conditioning. *Automation in Construction*, *141*, 104445.

Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, *2017*. https://doi.org/10.1155/2017/9324035

Silas, W. A., Nderu, L., & Ndirangu, D. (2024). A distributed framework for distributed denial-of-service attack detection in internet of things environments using deep learning. *International Journal of Web Engineering and Technology*, *19*(1), 67-87.

Singh, M. P., & Bhandari, A. (2020). New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges. *Computer Communications*, *154*, 509-527.

Staffini, A. (2023). A CNN–BiLSTM Architecture for Macroeconomic Time Series Forecasting. *Engineering Proceedings*, *39*(1), 33. https://doi.org/10.3390/engproc2023039033

Swain, P. K., Pattnaik, L. M., & Satpathy, S. (2025). IoT applications and cyber threats: mitigation strategies for a secure future. In *Explainable IoT applications: A demystification* (pp. 403-428). Cham: Springer Nature Switzerland.

Taye, M. M. (2023). Understanding of machine learning with deep learning: Architectures, workflow, applications, and future directions. *Computers*, *12*(5), 91. https://doi.org/10.3390/computers12050091

Thoutam, V. (2021). An overview on the reference model and stages of IoT architecture. *International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290*, *1*(01), 7-14.

Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent advances in stochastic gradient descent in deep learning. *Mathematics*, *11*(3), 682. https://doi.org/10.3390/math11030682

Ullah, R., Churcher, A., Ahmad, J., Ur Rehman, S., Masood, F., Gogate, M., ... & Buchanan, W. J. (2021). An experimental analysis of attack classification using machine learning in IoT networks. *Sensors*, *21*(2), 446. Doi: https://doi.org/10.3390/s21020446

Ungar, M. (2018). Systemic resilience: Principles and processes for a science of change in contexts of adversity. *Ecology and Society*, *23*(4). https://www.jstor.org/stable/26796886

Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in Internet of Things with a focus on the impact of emerging technologies. *Internet of Things*, *19*, 100564. https://doi.org/10.1016/j.iot.2022.100564

Zang, H., Liu, L., Sun, L., Cheng, L., Wei, Z., & Sun, G. (2020). Short-term global horizontal irradiance forecasting based on a hybrid CNN-LSTM model with spatiotemporal correlations. *Renewable Energy*, *160*, 26-41.

APPENDICES

DDoS-IoT Detection Framework Pseudocode

```
// MAIN PROCESS: Distributed DDoS-IoT Detection Framework

BEGIN MainFramework_Process

    // Initialize the Cloud Layer

    CloudModel = CloudLayer.InitializeGlobalModel()



    // Begin the federated learning training rounds

    FOR each TRAINING_ROUND from 1 to TOTAL_ROUNDS

        PRINT "Starting Training Round " + TRAINING_ROUND



        // --- STEP 1: DISTRIBUTE GLOBAL MODEL TO FOG NODES AND EDGE DEVICES ---

        // Simulate Fog nodes and Edge devices downloading the latest global model from the Cloud

        FOR each FOG_NODE in AllFogNodes

            FogModel = CloudModel.DownloadGlobalModel()

        END FOR
```

```
// --- STEP 2: LOCAL TRAINING AT FOG NODES (Federated Learning) ---
// Each Fog node trains the model on local data received from Edge devices
FogWeightUpdates_List = []
FOR each FOG_NODE in AllFogNodes
    // Edge devices perform data preprocessing and send data to their corresponding Fog node
    EdgeData_preprocessed = EdgeLayer.CollectAndPreprocessData()

    // Fog node trains the local model
    FogModel_updated, Delta_Wf = FOG_NODE.TrainModel(EdgeData_preprocessed)

    // Store the weight updates (Delta_Wf)
    FogWeightUpdates_List.Add(Delta_Wf)
END FOR

// --- STEP 3: AGGREGATE MODEL UPDATES AT THE CLOUD ---
// The Cloud receives and aggregates the weight updates from all Fog nodes
CloudModel = CloudLayer.AggregateAndApplyUpdates(CloudModel, FogWeightUpdates_List)
```

```
        PRINT "Training Round " + TRAINING_ROUND + " complete. Global model updated."
    END FOR

    // End of training. The final CloudModel is the trained global model.
    PRINT "Training complete. Final global model is ready for deployment."
END MainFramework_Process

// PSEUDOCODE FOR EACH LAYER'S CORE FUNCTIONALITY
// CLOUD LAYER
MODULE CloudLayer
    FUNCTION InitializeGlobalModel()
        // Create and initialize the CNN-BiLSTM model
        GlobalModel = new CNN_BiLSTM_Model()
        RETURN GlobalModel
    END FUNCTION

    FUNCTION AggregateAndApplyUpdates(CurrentGlobalModel,
```

```
FogWeightUpdatesList)
        // Step 1: Calculate the mean of all weight updates received from Fog nodes
        SumOfUpdates = 0
        FOR each Delta_Wf in FogWeightUpdatesList
            SumOfUpdates = SumOfUpdates + Delta_Wf
        END FOR
        MeanUpdate = SumOfUpdates / count(FogWeightUpdatesList)

        // Step 2: Apply the aggregated update to the global model's weights
        // This corresponds to the formula: MW = MW + mean(∑(ΔWf))
        UpdatedGlobalModel_Weights = CurrentGlobalModel.Weights + MeanUpdate
        CurrentGlobalModel.UpdateWeights(UpdatedGlobalModel_Weights)

        // Step 3: Authenticate and store the updated global model
        CurrentGlobalModel.AuthenticateAndStore()

        RETURN CurrentGlobalModel
    END FUNCTION
END MODULE
```

```
// FOG LAYER

MODULE FogLayer

    FUNCTION TrainModel(PreprocessedDataFromEdge)

        // Load the latest global model from the Cloud

        LocalFogModel = DownloadLatestModelFromCloud()



        // Perform local training using the CNN-BiLSTM model

        // Use the Adam optimizer with learning rate 0.0001

        // Train for 100 epochs with a batch size of 64

        LocalFogModel.train(PreprocessedDataFromEdge, epochs=100, batch_size=64,
optimizer=Adam)



        // Calculate the change in weights (ΔWf)

        Delta_Wf = LocalFogModel.Weights - InitialLocalModel.Weights



        // Send the weight updates to the Cloud layer for aggregation

        CloudLayer.SendUpdates(Delta_Wf)
```

```
        RETURN LocalFogModel, Delta_Wf
    END FUNCTION
END MODULE

// EDGE LAYER
MODULE EdgeLayer
    FUNCTION CollectAndPreprocessData()
        // Collect raw network traffic data from IoT devices
        RawIoTData = CollectData()

        // Perform data preprocessing (e.g., normalization, feature extraction)
        PreprocessedData = Preprocess(RawIoTData)

        // Send the preprocessed data to the assigned Fog node
        FogNode.SendData(PreprocessedData)

        RETURN PreprocessedData
    END FUNCTION
END MODULE
```