

**A LIGHTWEIGHT INTRUSION DETECTION SYSTEM
USING CONVOLUTIONAL NEURAL NETWORK
(CNN) AND LONG SHORT-TERM MEMORY (LSTM)
IN FOG COMPUTING**

BY

HAWAZEN JAMAAN ALZAHRANI

A Thesis Presented to the
COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

March 2024

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Hawazen Alzahrani** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS**.



Prof. Tarek Rahil Sheltami
(Advisor)



Prof. Aiman Elmaleh
Department Chairman



Dr. Abdulaziz Yagoub Barnawi
(Member)



Prof. Suliman Al-Homidan
Dean of Graduate Studies



Dr. Muhammad Yahya
Mahmoud
(Member)

Date _____



© Hawazen Jamaan Alzahrani

2024

[*To my kids.*]

ACKNOWLEDGMENTS

[

All praise and thank be to Allah.

I would like to express my heartfelt gratitude to my husband for his support throughout my M.Sc study. His encouragement, and belief in my abilities have been instrumental in my success.

To my parents, thank you for your endless prayers and sacrifices. I am also grateful to my siblings, Ahmed and Shatha, for their encouragement and support. Your words of wisdom and unwavering support have helped me stay focused and determined. Seeing your diligent work is a powerful source of inspiration and encouragement for me.

I would like to extend my deepest appreciation to my respected professor Tarek Rahil Sheltami. Your expertise, guidance, and mentorship have played a crucial role in shaping my research and academic growth. I am grateful for the opportunities you provided and the confidence you instilled in me. I would like to thank my committee members: Dr. Abdulaziz Barnawi and Dr. Muhammad Yahya Mahmoud for their advice and the time they spent reviewing this thesis.

I am immensely grateful to Dr. Farouq Aliyu for his invaluable advice and remarkable patience. His guidance has been instrumental in shaping my understanding and providing clarity in complex areas.]

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF ABBREVIATIONS	X
ABSTRACT	XI
ملخص الرسالة.....	XIII
CHAPTER 1 INTRODUCTION	1
1.1 Thesis Motivation	3
1.2 Research Objectives.....	6
1.3 Thesis Breakdown.....	6
CHAPTER 2 BACKGROUND	8
2.1 Fog Computing Architecture	8
2.2 Intrusion Detection System IDS.....	9
2.3 Machine Learning	11
2.4 CICIOT2023 Dataset	12
2.5 Raspberry Pi.....	14
CHAPTER 3 LITERATURE REVIEW	15
3.1 Lightweight IDSs in FC	15
3.2 IDSs in FC	18

3.3	Summary.....	27
CHAPTER 4 METHODOLOGY AND ANALYSIS		30
4.1	Problem Statement	30
4.2	Research Methods.....	32
4.3	Dataset Processing	34
4.4	Anomaly Detection using Machine Learning Models.....	40
4.5	Evaluation Metrics.....	48
4.6	Evaluation Environment	49
4.6.1	Software and Libraries.....	49
4.6.2	Hardware	50
4.7	Experiment Setup	51
CHAPTER 5 RESULTS		56
5.1	Model Performance Evaluation.....	56
5.1.1	comparison with the initial versions of the model	68
5.2	Comparison with Related Work	71
CHAPTER 6 CONCLUSION AND FUTURE WORK		75
6.1	Conclusion	75
6.2	Future Work	76
REFERENCES		77
VITAE		86

LIST OF TABLES

Table 1 Distribution of attack labels on attack classes	37
Table 2 CNN-LSTM parameters	45
Table 3 Evaluation environment specifications	51
Table 4 Detection performance for classes	62
Table 5 Comparison of the proposed system with papers in literature review	72
Table 6 Comparison of proposed model with lightweight IDS	74

LIST OF FIGURES

Figure 1 The structure of the lightweight IDS	5
Figure 2 The proposed methodology of IDS for FC.....	34
Figure 3 The dataset splitting process.....	40
Figure 4 CNN-LSTM architecture.....	41
Figure 5 Experiment setup	53
Figure 6 Circuit diagram.....	55
Figure 7 Accuracy of CICIoT202 binary classification.....	58
Figure 8 ROC curve of CICIoT2023 binary classification.....	59
Figure 9 Accuracy of multi classification.....	60
Figure 10 ROC AUC performance for multi-class CICIoT2023 dataset	63
Figure 11 Performance evaluation of the model for the datasets.....	64
Figure 12 Energy consumption of the model steps.....	66
Figure 13 Latency of the model steps.....	68
Figure 14 comparison of the optimized model performance and initial versions.....	69
Figure 15 Power consumption of optimized and initial version.....	71

LIST OF ABBREVIATIONS

IDS	:	Intrusion Detection System
CNN	:	Convolutional Neural Network
LSTM	:	Long Short-Term Memory
ML	:	Machine Learning
DL	:	Deep Learning
FC	:	Fog Computing
IoT	:	Internet of Things
TP	:	True Positive
TN	:	True Negative
FP	:	False Positive
FN	:	False Negative
AUC	:	The Area Under the ROC (Receiver Operating Characteristic) Curve
ACC	:	Accuracy

ABSTRACT

Full Name : Hawazen Jamaan Alzahrani
Thesis Title : A lightweight Intrusion Detection System using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) in Fog Computing
Major Field : Computer Networks
Date of Degree : February 2024

The Internet of Things (IoT) links various devices to digital services and significantly improves the quality of our lives. IoT connectivity is growing rapidly, which could lead to network vulnerabilities and threats. Many interesting Intrusion Detection Systems (IDSs) have been presented based on machine learning (ML) techniques to overcome this problem, such as support vector machine (SVM), auto-encoder (AE), artificial neural networks (ANN), etc. However, since the fog layer is a resource-constrained environment, a lightweight intrusion detection system is extremely desirable. This thesis proposes a machine learning scheme with convolutional neural networks (CNN) based on LSTM to build an energy-aware, anomalies-based IDS tested on a recent dataset with a focus on reducing and mitigating overhead, achieving high accuracy, and a low false alarm rate. A comparison is made between the CICIoT2023, KDD99, and NSL-KDD datasets to evaluate the performance of the proposed IDS model based on key metrics, including latency, energy consumption, false alarm rate, and detection rate metrics. The obtained findings are above 92% accuracy, indicating that the suggested system can provide a remarkable layer of security. The performance evaluation highlighted the practicality of deploying IDSs with limited resources, as demonstrated by the successful implementation of IDS functionality on the Raspberry Pi as a fog node. The proposed

lightweight model has a maximum power consumption of 6.12 w. The model demonstrates its potential to operate effectively on energy-limited devices, such as low-power fog nodes or edge devices.]

ملخص الرسالة

الاسم الكامل: هوازن جمعان الزهراني

عنوان الرسالة: نظام كشف التسلل خفيف الوزن باستخدام الشبكة العصبية التلافيفية (CNN) والذاكرة الطويلة قصيرة المدى (LSTM) في الشبكة الضبابية

التخصص: شبكات الحاسب

تاريخ الدرجة العلمية: فبراير ٢٠٢٤

تعمل إنترنت الأشياء على ربط الأجهزة المختلفة بالخدمات الرقمية وتحسين نوعية حياتنا بشكل كبير. ينمو قطاع إنترنت الأشياء بسرعة، مما قد يؤدي إلى نقاط ضعف وتهديدات في الشبكة. تم تقديم ودراسة العديد من أنظمة كشف التسلل المثيرة للاهتمام استنادًا إلى تقنيات التعلم الآلي للتغلب على هذه المشكلة. ومع ذلك، فإن طبقة الضباب هي بيئة محدودة الموارد، ومن المرغوب فيه للغاية وجود نظام كشف التسلل خفيف الوزن. تقترح هذه الأطروحة مخططًا للتعلم الآلي باستخدام التعلم الآلي الموفر للطاقة لكشف التسلل الذي تم اختباره على مجموعة بيانات حديثة مع التركيز على تقليل الاستهلاك للموارد، وتحقيق دقة عالية ومعدل إنذار كاذب منخفض. تم إجراء مقارنة بين مجموعات البيانات KDD99 و NSL-KDD لتقييم أداء النموذج المقترح. النتائج التي تم الحصول عليها تزيد عن 90%، مما يشير إلى أن النظام المقترح يمكن أن يوفر طبقة عالية من الأمان. يتمتع النموذج خفيف الوزن المقترح بأقصى استهلاك للطاقة يبلغ 6.12 واط، ويوضح النموذج خفيف الوزن المقترح قدرته على العمل بفعالية على الأجهزة ذات الطاقة المحدودة، مثل عقد الضباب منخفضة الطاقة أو الأجهزة الطرفية

CHAPTER 1

INTRODUCTION

The Internet of Things is expanding daily as new devices are connected to the network. It provides ubiquitous connectivity for devices, services, and systems and is widely used in our daily routines. We experienced an increase in linked gadgets and IoT technologies, ranging from smart homes, drones, and even autonomous vehicles. The Internet of Things continues to grow in popularity, but the security threats are also increasing [1].

Due to their small size and low power, these devices cannot be equipped with advanced defense mechanisms. It is critical to ensure network security and information protection in IoT applications since it could lead to increased network vulnerabilities and threats. It is important to transport the created data to a device with more computational capability in order to carry out the storage, processing, and analysis. The move to cloud computing is impeded by these devices' high traffic volume. Fog computing is a distributed computing paradigm that works nearer to the network's edge, allowing it to pre-process data before sending it to the cloud. It can thus provide faster handling and response for IoT devices [2].

Fog devices are deployed in areas that lack adequate protection, making them susceptible to various cyberattacks. To address these challenges, the Intrusion Detection System (IDS) has been implemented in fog computing to significantly mitigate the security risks posed by attacks on the fog infrastructure. IDS is an effective method for detecting the

presence of attacks or abnormal behaviors in the fog layer by monitoring the software and hardware within a network, with the underlying assumption that the actions of an intruder can be measured and distinguished from those of a normal user. The defense cannot anticipate that there will be a clear separation between an attack and the regular usage of resources by a legitimate user, which presents another difficult situation. Instead, there will likely be some overlap, which could result in an undesirable circumstance.

Machine Learning (ML) algorithms have been implemented to assist in these situations. ML techniques have emerged as a promising approach to the problem of network security in fog computing. Through intelligent algorithms that can learn from different datasets, it can help identify patterns and anomalies in data that may indicate potential security threats. ML-based IDS on fog computing can also leverage the distributed nature of the computing paradigm to implement more efficient and effective intrusion detection mechanisms.

Many interesting IDSs have been presented based on Machine Learning and Deep Learning approaches, such as support vector machine (SVM) [3], auto-encoder (AE) [4], and artificial neural networks (ANN) [5]. Methods have demonstrated encouraging results in identifying significant features within IoT traffic. They have proven to be effective in efficiently detecting network intrusions and attacks. However, implementing intrusion detection systems (IDS) based on IoT technology on fog nodes with limited capacity and low power can be complex and demanding. However, employing a lightweight IDS can potentially offer a beneficial solution for IoT devices. These lightweight IDS utilize machine learning algorithms to effectively detect intrusion attempts without consuming the energy of the fog node [6].

This research aims to build a lightweight intrusion detection system based on machine learning models for IoT systems. This improves security and reduces overhead at fog nodes while providing services to the IoT layer. This approach involves leveraging the capabilities of the fog layer by moving first-line defense closer to the IoT layer and reducing networking and computation costs from the cloud layer. Improving and examining the hybrid machine learning model Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). To the best of our knowledge, most existing FC-IDSs that use the hybrid approach of CNN and LSTM achieved high accuracy [7][8]. However, none of these studies considered energy consumption. The use of old datasets and deploying the model to the cloud layer are the limitations of the work in lightweight IDS [9]. The CICIoT2023 dataset was used, which includes multiple attacks not available in other IoT datasets and helps IoT experts develop new security analytics solutions [10].

1.1 Thesis Motivation

In recent years, the proliferation of interconnected devices and the growing adoption of fog computing have introduced new security challenges in the fog layer. Intrusion Detection Systems (IDS) play a crucial role in detecting and mitigating potential security threats in these environments. However, building energy-aware, anomaly-based IDS in the fog layer poses significant challenges that need to be addressed.

One of the primary limitations of existing IDS models is their reliance on outdated datasets that fail to capture the majority of modern attacks. This research is motivated by the need to overcome this limitation by leveraging a modern dataset, CICIoT2023, which includes a comprehensive range of contemporary attack scenarios. By incorporating this

dataset, the proposed IDS model aims to enhance detection accuracy and ensure its relevance in the face of emerging threats.

Effective attack classification is another critical aspect that needs to be addressed. To achieve this, a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) machine learning model will be employed. This hybrid architecture is capable of handling both binary-class and multi-class attack classification tasks while considering the energy consumption of the model. By optimizing the energy efficiency of the IDS model, this research aims to minimize resource consumption and improve the sustainability of fog computing environments.

Furthermore, the practicality of deploying IDSs using limited resources is an important consideration. Utilizing the Raspberry Pi as a fog node, this study explores the feasibility of implementing IDS functionality in resource-constrained fog computing environments. By demonstrating the viability of IDS solutions in such environments, the applicability and effectiveness of IDS can be expanded.

The proposed IDS model will be evaluated based on key performance metrics, including latency, energy consumption, and detection rate. Through this evaluation, the efficiency and effectiveness of the model will be quantified and compared to existing techniques. Special emphasis will be placed on improving accuracy and reducing the false positive rate, addressing the pressing need for reliable and efficient IDS solutions in the fog layer.

Overall, this research is motivated by the need to address the challenges of energy-aware, anomaly-based IDS in the Fog layer, with a focus on reducing and mitigating overhead. By leveraging a modern dataset, employing a hybrid CNN-LSTM model, and considering

resource constraints, this study aims to contribute to the development of more efficient and effective IDS solutions for fog computing environments. Figure 1 shows the proposed lightweight IDS structure for the fog computing environment.

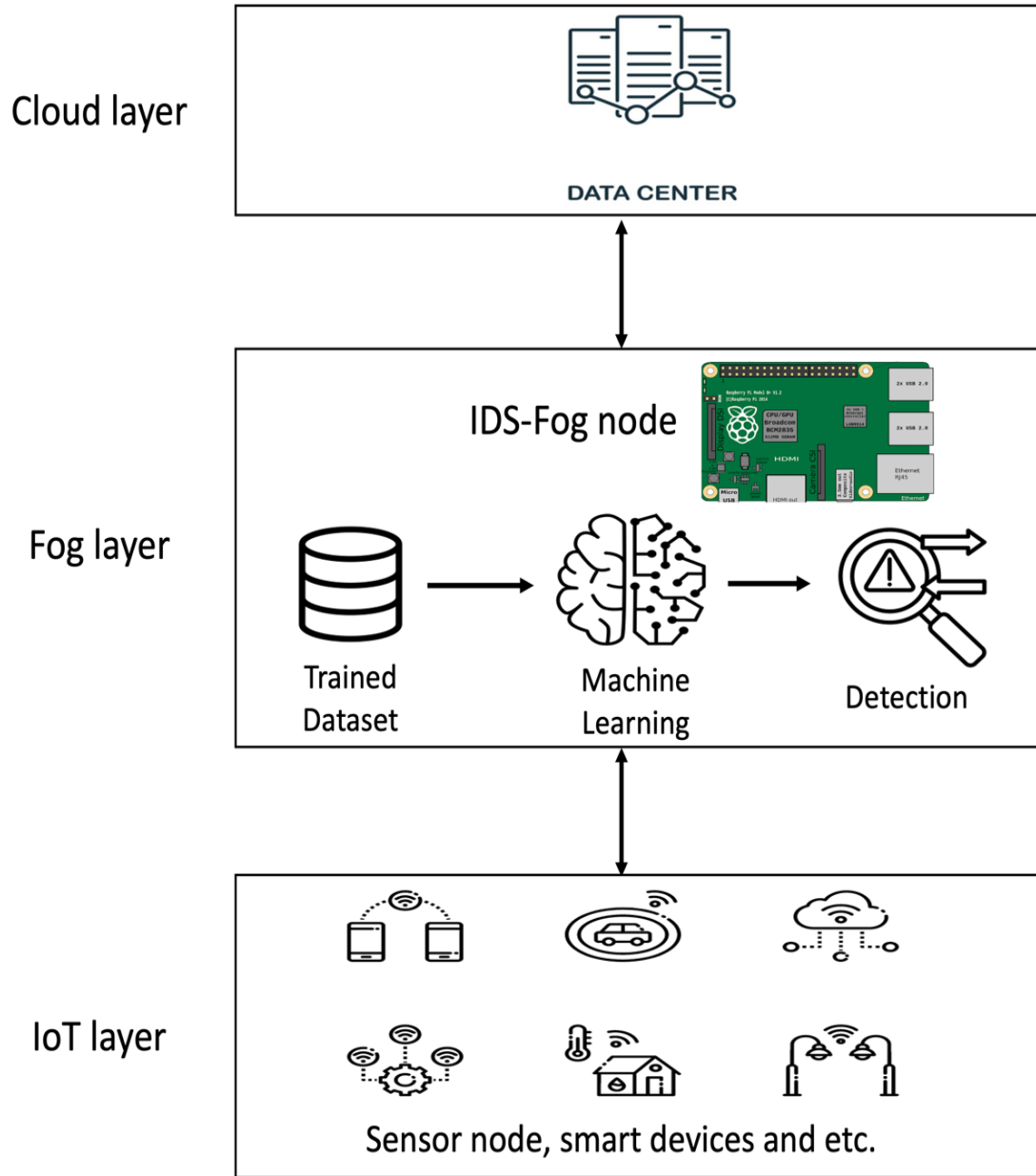


Figure 1 The structure of the lightweight IDS

1.2 Research Objectives

The research objective is to develop an energy-aware, anomaly-based Intrusion Detection System (IDS) in the fog layer while addressing the challenge of reducing and mitigating overhead. The main objectives of this work are:

1. To improve detection accuracy and relevance in the face of emerging threats using the CICIOT2023 dataset, which includes contemporary attack scenarios.
2. To employ a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) machine learning model for effective attack classification while considering energy consumption.
3. To optimize the energy efficiency of the IDS model in order to minimize resource consumption.
4. To explore the practicality of deploying IDSs with limited resources, specifically utilizing Raspberry Pi as a Fog node. This investigation aims to demonstrate the feasibility of implementing IDS functionality in resource-constrained fog computing environments.
5. To evaluate both the binary-class and multi-class attack classification performance based on key metrics such as latency, energy consumption, and detection rate.

1.3 Thesis Breakdown

The rest of the paper is organized as follows: The next chapter discusses the background of the Internet of Things, Intrusion Detection System, Machine Learning, Dataset, and

Raspberry Pi. Chapter 3 is a comprehensive literature review of IDS in fog computing. Chapter 4 is an overview of the methodology of the proposed work. The preprocess, metrics, and experiment have been covered. The evaluation of the results and comparison with previous studies are presented in Chapter 5. The conclusion and future work are presented in Chapter 6.]

CHAPTER 2

BACKGROUND

This chapter is designed to provide introductory information on the different aspects of this research. There are several parts involved in developing an intrusion detection system. First, we described how the architecture of fog computing generally functions. Then we covered the fundamentals of intrusion detection systems. Moreover, the dataset that was used in this research. Then, the hardware that used to run the experiment. Finally, the architecture of the machine learning models that were utilized is as follows.

2.1 Fog Computing Architecture

Fog computing is a distributed computing paradigm that brings the typical cloud computing concept closer to the data sources and users at the edge of a network. By relocating computer operations closer to the devices providing the data, it aims to solve the latency, bandwidth, and network congestion limitations of cloud computing [11].

The architecture is composed of three layers: the edge layer, the fog layer, and the cloud layer. End devices, edge devices, and gateways compose the edge layer, which collects and processes data from numerous sensors and devices at the network's edge [12]. The fog layer contains both intermediary devices like routers and switches and fog nodes that provide computational and storage capabilities closer to the edge. Finally, the cloud layer represents standard cloud data centers, which provide more storage and processing capacity for tasks that require more resources.

To enable the efficient execution of applications at the network's edge, fog computing architecture makes use of essential enabling technologies such as distributed computing, networking, storage, and control functions [11]. Its goal is to serve a wide range of applications that demand real-time data processing, low latency, and optimal network resource use in situations such as the Internet of Things (IoT), smart cities, and autonomous systems [12].

As a result, the fog computing paradigm can be efficiently used in the IoT to host anomaly mitigation systems. This will allow for effective attack detection near IoT devices. Furthermore, by reducing the operational needs of the anomaly mitigation solution on the IoT devices, it would assure efficient operation of the IoT devices [13].

2.2 Intrusion Detection System IDS

An intrusion detection system (IDS) is a security tool or software that analyzes network traffic and system operations in order to detect and respond to potential security breaches or malicious behavior. IDS assists enterprises in detecting and mitigating a wide range of attacks, including intrusions, unauthorized access attempts, and malicious software infections [14]. The intrusion detection system's primary function is to monitor network traffic and detect patterns or signatures that indicate potential threats to security. When it identifies suspicious behaviors, it inspects network packets or log files for known attack patterns or anomalies and generates alarms.

It is deployed in two different methods. The first is at the host level (HIDS) on a node to monitor system operations on its system application files or the OS that is running on the node. A node at this level could be an IoT device or computer system. Second, at the

network level (NIDS) on a border router or gateway, where it analyzes traffic flows on the network [15]. The NIDS is classified as signature-based and anomaly-based. Anomaly NIDS is very effective at identifying previously unknown or novel assaults and zero-day attacks. It can identify changes in network or system activity, such as strange network traffic patterns, irregular user actions, or unexpected system behaviors. This proactive strategy allows for rapid detection and response to security problems, limiting possible harm or illegal access.

Due to its operating technique, the signature-based IDS works effectively in detecting known assaults. However, it does not identify unknown assaults and is insufficient for the limited resources of IoT. The anomaly-based IDS, even if it is considered ideal for IoT, is able to detect unknown assaults. One drawback is that it can generate false alarms by incorrectly classifying normal traffic as abnormal traffic [17]. Machine learning has developed as a significant technology in Network Intrusion Detection Systems (NIDS), aimed at reducing false positives and improving the accuracy and effectiveness of identifying sophisticated and constantly changing attacks. Machine Learning-based Network Intrusion Detection Systems analyze network traffic and detect suspicious activities using machine learning algorithms such as clustering, deep learning, artificial neural networks, and support vector machines [16]. The efficiency of ML-NIDS systems varies depending on the quality and quantity of data ingested, as well as the parameters and machine learning algorithm used. For accuracy, these systems must combine mathematical modeling and hands-on knowledge to detect and overcome potential network-based assaults, guaranteeing that security services are appropriately delivered within the network.

2.3 Machine Learning

AI technologies such as machine learning (ML) and deep learning are frequently used to evaluate and analyze the data and provide classification, decision analysis, and detection of potential behaviors in IoT systems. Deep learning, a subset of machine learning, focuses on training models called artificial neural networks (ANNs) with several layers of interconnected neurons to learn complex patterns and representations. Convolutional Neural Networks (CNNs) can be used in IDS to analyze network packet payloads or images and identify malicious or anomalous patterns [18]. Recurrent Neural Networks (RNNs) can capture sequential dependencies in network traffic or system logs, making them suitable for detecting intrusions that span multiple packets or events. Long Short-Term Memory (LSTM) networks, a type of RNN, are effective in capturing long-range dependencies and temporal patterns, which are advantageous in IDS [19].

The use of CNN-LSTM hybrid deep learning in IDS in fog computing environments has several advantages, including higher accuracy in detecting malicious network activities compared to other IDS models [20]. Suitable for fog computing environments, which involve resource-limited computing nodes, due to their ability to reduce feature extraction overheads and false alarms [8]. Detect network anomalies in real-time, making them suitable for time-sensitive applications [21].

The three basic categories of machine learning approaches are supervised, unsupervised, and semi-supervised learning, which are based on the availability and usage of labeled data for training the models [22]. Supervised learning algorithms learn from labeled examples provided during the training phase. Each example consists of input data

(features) and corresponding output labels. Popular algorithms for supervised learning include decision trees, random forests, support vector machines (SVM), and neural networks [23-25]. Unsupervised learning algorithms learn from unlabeled data, where no predetermined output labels are provided during training. Unsupervised learning algorithms include k-means clustering, hierarchical clustering, principal component analysis (PCA), and generative adversarial networks (GANs) [26]. In semi-supervised learning, a small portion of the data is labeled, and a larger amount of data remains unlabeled. The labeled data is used in a supervised manner to guide the learning process, while the unlabeled data helps in finding additional patterns or improving the model's performance [22].

2.4 CICIOT2023 Dataset

In this study, the most recent dataset is considered. The CICIOT2023 [10] dataset proposes a novel and extensive IoT attack dataset that can facilitate the development of security analytics applications in real IoT operations. The researchers executed 33 attacks in an IoT topology made up of 105 devices, and they were divided into seven categories: DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai. Benign traffic represents legitimate use of the IoT network. In particular, all assaults were carried out by IoT devices that were suspicious and intentionally targeted other IoT devices. This dataset provides different attacks that were not previously available in other IoT datasets.

DDoS (Distributed Denial of Service): DDoS attacks involve overwhelming a target system or network with a flood of traffic from multiple sources, rendering it inaccessible to legitimate users. These attacks include techniques such as ACK Fragmentation,

Slowloris, ICMP/HTTP/UDP/TCP Flood, RST-FIN Flood, PSH-ACK Flood, UDP Fragmentation, ICMP Fragmentation, SYN Flood, and Synonymous IP Flood.

Recon (Reconnaissance): Reconnaissance attacks involve gathering information about a target system or network, often with the intention of identifying vulnerabilities or weaknesses for future exploitation. The most popular variations are Ping Sweep, OS Scan, Vulnerability Scan, Port Scan, and Host Discovery.

Web-based: Web-based attacks target vulnerabilities in web applications or services. These attacks can include techniques such as cross-site scripting (XSS), SQL injection, Browser Hijacking, Uploading Attack, Backdoor Malware, and Command injection.

Brute Force: Brute force attacks involve attempting to gain unauthorized access to a system or network by systematically trying all possible combinations of usernames and passwords until a successful match is found.

Spoofing: Spoofing attacks involve falsifying or manipulating data or network addresses to deceive systems or users. This can include IP spoofing, where the source IP address is forged to appear as if it is from a trusted source.

Mirai: Mirai is a specific type of malware that targets IoT devices, turning them into botnets to launch DDoS attacks. Mirai-infected devices are used to carry out coordinated attacks on other IoT devices or networks within the GRE packet, GREETH, and UDP Plain.

2.5 Raspberry Pi

The Raspberry Pi is a single-board computer (SBC) that is small but powerful, created by the Raspberry Pi Foundation. Due to its low power consumption and flexibility, the Raspberry Pi may be used for a wide range of tasks and is effectively a computer. The Raspberry Pi board is a very adaptable and practical choice for testing various Machine Learning models in a low-power and resource-constrained setting. The Debian-based operating systems that run on the Raspberry Pi enable the simple installation and use of machine learning and deep learning libraries like Scikit-learn. The Raspberry Pi kernel lags behind the main Linux Kernel.

[CHAPTER 3]

[LITERATURE REVIEW

The field of intrusion detection systems (IDS) in the fog computing layer has witnessed significant research efforts aimed at addressing the emerging security challenges in this distributed and resource-constrained environment. In this chapter, we present a comprehensive review of the existing literature on IDS in the fog computing layer, focusing on two distinct sections. The first section examines studies that specifically consider energy efficiency and consumption aspects in IDS design, while the second section explores studies that primarily focus on IDS functionality without explicit consideration of energy constraints.

3.1 Lightweight IDSs in FC

These studies have aimed to develop novel techniques, algorithms, or architectures that optimize IDS performance while minimizing energy and CPU consumption. We explore various approaches, such as energy-aware algorithms and lightweight machine learning models, that contribute to more sustainable and energy-efficient IDS solutions in fog computing.

A lightweight IDS for a fog computing environment utilizing a Multilayer Perceptron (MLP) model with a Raspberry Pi serving as a fog node was proposed by Sudqi Khater et al. [9]. It was used to detect different types of attacks on two datasets: ADFA-LD, which had 94% accuracy, 95% recall, and 92% F1-score, and ADFA-WD, which had 74%

accuracy, 74% recall, and 74% F1-score. Based on their studies, the average testing time is about 750 microseconds, which is a reasonable latency for a lot of Internet of Things applications. By utilizing contemporary learning techniques, their model still has to be enhanced in terms of detection accuracy and computing efficiency. Present IoT network assaults are not detectable by it.

The authors [27] use the ADFA-LD dataset to classify data using a Multilayer Perceptron (MLP) model and a modified Vector Space Representation (MVSR) approach known as N-gram. Additionally, they use the Linear Correlation Coefficient (LCC) to address test data that contains missing N-grams. Principal Component Analysis (PCA) and Mutual Information (MI) are used to choose features. The simulation's outcomes show that the suggested approach produces a 5% False Positive Rate (FPR) and 96% accuracy. It also managed to achieve low energy consumption of 8.809 mj and low CPU time utilization of 4.404 ms. To identify the most recent IoT network threats, their model needs to be enhanced in terms of dataset.

IDS nodes that identify any deviations from typical network activity, classify them as malicious, and isolate the suspected node were suggested by Farouq Aliyu et al. [28]. Exponentially Weighted Moving Average (EWMA) is incorporated into the system to reduce the noise from social media discussions. After adding EWMA, its accuracy increases from 80% to 95%, and it can identify intrusions 0.25–0.5 seconds faster. However, utilizing EWMA results in a minimum 0.75–1.3 second increase in latency. The suggested system's application to other attack types or domains is not examined; instead, the research concentrates exclusively on MITM assaults in fog computing for social media.

The human immune system served as the model for a unique intrusion detection technique that was introduced by Farouq Aliyu et al. [29] in fog computing. Reduced resource usage and increased efficiency can be attained by utilizing a distributed design that makes use of fog nodes and cloud computing. The suggested solution reduced energy usage by 10% as compared to using a neural network on the fog node, and it achieved accuracy of up to 98.8% in KDD99 and 96.7% in NSL-KDD datasets. As the duty cycle increases, the system's accuracy is seen to increase. But as a result, latency and energy consumption increase.

B-Stacking is a unique IoT intrusion detection approach that Souradip Roy et al. [30] devised. Using the CICIDS2017 and NSL-KDD datasets, it applies improved machine learning algorithms to obtain high accuracies of about 99% and 98%, respectively. The model includes optimization techniques like sampling, data scaling, dimensionality reduction, and multicollinearity removal. These processes reduce computing complexity by effectively identifying critical aspects and drastically reducing the quantity of features and data points needed for abnormality identification. According to the experimental results, B-Stacking is a potential intrusion detection technique for Internet of Things environments since it has a high detection rate and a very low false alarm rate. There is no simulation or implementation in the research to evaluate the power consumption.

3.2 IDSs in FC

This section reviews a range of research works that have proposed innovative IDS models, detection algorithms, or data analysis techniques in the fog computing layer. By examining these studies, we gain insights into the advancements made in the field of IDS within the fog computing context, including the utilization of machine learning, anomaly detection techniques, and real-time analysis to enhance the detection capabilities of IDS systems.

The Baseline K-means approach is a semi-supervised novelty identification method that was first presented by Suzan Hajj et al. [31]. It utilizes a small amount of labeled benign data to train a classifier, which is then used to identify more unlabeled data points. The authors describe a cross-layer strategy that combines a cluster-based sampling technique with the federated Baseline K-means algorithm. The memory, power, and compute resources needed to process network traffic are decreased by using the cluster-based sampling technique. The NSL-KDD dataset is used to present experimental results in this research. The findings demonstrate that by fostering cooperation between employees and the coordinator, the suggested federated IDS can raise the true-positive rate by as much as 10%. Nevertheless, there is a rise in the false positive rate, which lowers precision. Also, the degradation in performance metrics over time when utilizing cluster-based sampling.

A hybrid architecture with two modules—a signature-based module and an anomaly-based module—was presented by Lawal et al. [13]. In comparison to the anomaly-based module, the signature-based module demonstrated a quicker attack detection time,

according to the data. With the XGBoost classifier, the anomaly-based module attained a low false-positive rate and good accuracy. Similar to this, Gad et al. [77] examined different machine learning techniques in the context of particular ToN-IoT dataset partitions, assessing their performance in binary and multi-class classification tasks. The authors emphasize how much better the XGBoost method is in the suggested model when compared to other ML algorithms. The suggested IDS by Kumar et al. [78] uses a distributed ensemble design with first-level individual learners such as Gaussian naive Bayes, XGBoost, and k-nearest neighbors. For the final classification, Random Forest employs the prediction outputs from these learners as a second-level classifier. Tu et al. [33] use physical layer security (PLS) to identify and prevent impersonation attacks in fog computing networks. The Q-learning system constantly modifies the test threshold to further improve the impersonation attack detection accuracy. The Extreme Learning Machine (ELM) algorithm and the semi-supervised fuzzy c-means algorithm are combined in the ESFCM approach, which was suggested by Rathore et al. [22] to solve the problem of labeled data and offer quick detection with strong generalization performance. Two stages of classification are involved in the authors' proposal [34] for improved accuracy, which calls for a variety of base learners and ensemble approaches. While attack classification is done at the second level for accurate preventive actions, anomaly detection is the primary focus of the first level in order to minimize delay. The ODM-ADS technique was presented by Moustafa et al. [35]. It allows for self-adaptation to data poisoning attempts, creates a valid profile, and uses an outlier function to identify anomalies. It is suggested as a fog-based framework that can be deployed in fog nodes as Software as a Service, which makes it appropriate for dynamic networks such as cloud,

fog computing, and the Internet of Things. The semi-distributed method selects features using parallel machine learning models on edge devices, and then classifies them using a multi-layer perceptron on fog devices. These concepts are combined in the distributed strategy for decision-making. The suggested techniques by Rahman et al. [37] seek to overcome resource limitations in IoT devices while achieving detection accuracy comparable to centralized IDS. In order to address scalability difficulties and improve response times, Moh et al. [39] investigate the idea of fog computing, which includes relocating processing and storage closer to the network edge. It talks about how to secure fog computing systems and Internet of Things devices using machine learning techniques. It specifically looks at methods for handling fog computing security concerns, controlling IoT data expansion, and identifying anomalies and assaults. In two scenarios—denial of service (DoS) detection and multi-attack detection in F2T computing—experiments are carried out by Diro et al. [40]. They show the efficacy and efficiency of deeper models by contrasting the discriminating attack performance of LSTM networks with shallow methods. The MECNN model, as proposed by Chen et al. [42], integrates a multi-objective evolutionary algorithm based on decomposition (MOEA/D) for CNN model optimization and uses CNN as the classifier for intrusion detection. This approach simplifies the parameter tuning process of deep learning techniques, which can be time-consuming. A novel method for intrusion detection in CPSs using GANs and a fog-based architecture is presented by De et al. [43]. The proposed method effectively addresses the issues of latency and lack of labeled data, as demonstrated by the experimental findings. However, Meng et al. [45] offer a plan of

defense for MFC networks against clever attackers. To arrive at the best defense plans, it integrates the DQL algorithm, prospect theory, and static subjective game modeling.

Several Deep Learning researchers, including Souza et al. [32], offer a binary Extra Tree (ET) classifier that is used to analyze traffic and categorize different traffic flows as intrusive or non-invasive. Intrusive traffic moves on to the second phase, whereas non-intrusive traffic is released right away. It uses a group methodology with classifiers from Extra Tree, Random Forest (RF), and Deep Neural Networks (DNNs). Reputable intrusion datasets including Bot-IoT, IoTID20, NSL-KDD, and CICIDS2018 are used for evaluation. In Rai et al. [38], the Python package H2O is used to perform ensemble techniques such Distributed Random Forest (DRF), Gradient Boosting Machine (GBM), and XGBoost. A Deep Neural Network (DNN) with feature selection based on the genetic algorithm is also utilized. The suggested model performs better than both conventional machine learning models and earlier IDS techniques, according to the results. The Auto-IF method, which was proposed by Sadaf et al. [44], uses deep learning techniques like Autoencoder (AE) and Isolation Forest (IF) to detect intrusions in fog environments. The strategy focuses on binary classification of receiving packets in real-time to distinguish between legitimate and malicious communication. An Incremental Deep Neural Network (DNN) is used to detect the existence of intruders based on the attributed data in a model built by Abdussami et al. [46]. The model takes into account the steps of preprocessing, feature extraction, and intrusion detection. An algorithm called Modified Electric Fish Optimization (M-EFO) is used to optimize the DNN's weights. Souza et al. [47] Deep Neural Networks (DNN) and the k-Nearest Neighbor (kNN) algorithm are combined in the suggested method, which is a hybrid binary

classification technique known as DNN-kNN. The approach of Gain Ratio Attribute Evaluation is employed to determine which attributes are optimal for the detection process. Furthermore, the methodology is engineered to operate with minimal overhead for memory and processing expenses. The main achievements of DL-IDS are emphasized by Sun et al. [58], who point out the high accuracy attained on the CICIDS2017 dataset, the usage of CNN-LSTM hybrid network for feature extraction, and category weight optimization for training. DL-IDS uses a category weight optimization technique to improve the robustness of the model in response to the issue of an uneven amount of samples in the training data representing various attack types. Six distinct DL models are compared by Samy et al. [50] in order to ascertain which, one works best. These models are evaluated using five distinct datasets, each with a unique attack type. The experiments demonstrate that in terms of accuracy and detection rate, the long short-term memory (LSTM) model outperforms the other DL models. A CUDA-deep neural network gated recurrent unit (Cu-DNNGRU) framework trained on the N-BaIoT dataset powers the Cu-DNNGRU system proposed by Attique et al. [51]. The efficiency, accuracy, precision, resource consumption, and computing complexity are all indicated by the simulation results. Tuli et al. [57] goal is to solve the problems that arise from object detection programs used in real-world settings, such as lengthy data transfer times and inadequate system frameworks. With the help of the Aneka framework and deep learning, it offers a lightweight automatic object identification system that can be deployed in cloud and fog computing environments. The system can adjust to the requirements of the user and the application thanks to its two operating modes: high accuracy mode and low latency mode. The suggested model by Çavuşoğlu et al. [75] automatically extracts

characteristics and classifies network traffic data by utilizing deep learning and transfer learning techniques. The deep learning model's transferred and adjusted convolutional layers process the 2D preprocessed feature maps created from the network traffic. Lastly, the traffic data is detected and classified using a thick layer. Fast Fourier transform (FFT) and deep learning (DL) approaches were introduced by Chang et al. [81] and are useful for extracting high-level information that can be used to identify new assaults or tiny alterations. Scalable microgrid monitoring and cyber-attack detection in the communication channels between distributed generation agents (DGAs) are made possible by the distributed approach. In order to minimize the total inferred latency, Jiang et al. [48] suggested model comprises of two primary algorithms: Maximizing Accuracy based on Guaranteed Latency (MAL) and Solution Space Tree Pruning (SSTP). The findings demonstrate that the recommended strategy reduces cloud performance occupation and cost while increasing average latency.

In order to develop defenses against DDoS attack detection and prevention, Gudla et al. [60] assess how well two DLMs—Deep Neural Multilayer Perceptron (DNMLP) and Long Short-Term Memory (LSTM)—perform against traditional machine learning (ML) models like Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Random Forest (RF) in predicting end-user behavior and detecting DDoS attacks. A Nave Bayesian-based intrusion detection system (IDS) that is coupled with a Markov Model and a Virtual Honey Pot Device is given by Singh et al. [61] as a means of identifying DDoS attacks. Roopak et al. [65] also address DDoS attack detection, offering insights into the application of deep learning models, including CNN and LSTM. An approach to DDoS defense for fog and cloud computing settings is

presented by these studies [49] [82]. With the use of SDN and machine learning, it aims to improve service availability by detecting and removing unwanted traffic. Another method for identifying DDoS assaults in IoT networks was presented by Gaurav et al. [52] using a fog layer. This method finds malicious nodes and filters attack traffic using clustering and entropy-based techniques. The CRPS score is utilized for probabilistic forecasting, and Sharma et al. [70] evaluate the detection system using data from the DARPA 99 dataset. They put a lot of emphasis on anomaly spotting DDoS attacks using statistics-based methods. The proposed FOCUS system by Maharaja et al. [76] makes use of a layered protection mechanism that protects the VPN server from potential DDoS attacks by using a virtual private network (VPN), a traffic analysis unit based on machine learning to classify traffic, and challenge-response authentication to validate the source of suspicious traffic. Bishnoi et al. [80] The suggested approach makes use of two deep learning models to handle DDoS attacks: a convolutional neural network (CNN) model and a long short-term memory (LSTM) model. A fog-based attack detection framework employing an ensemble of online sequential extreme learning machines (EOS-ELM) is proposed by Alrashdi et al. [54]. Because of its close proximity to Internet of Things devices at the network edge, the distributed attack detection approach used in fog computing offers lower latency and improved accuracy. The authors draw attention to the shortcomings of centralized-based techniques, highlighting their high latency. The idea of Exact Greedy Boosting (XGBoost), an ensemble learning technique, is used by Reddy et al. [55] to construct the IDS in fog nodes. Mohamed et al. [56] The authors conducted experiments using Raspberry Pi4 as a fog node and evaluated the system's performance based on the UNSW-NB15 and ToN_IoT datasets. It achieved a significant reduction in

execution time compared to other state-of-the-art methods, leading to faster convergence and saving computing power. However, they did not evaluate the energy and power consumption rates.

Diro et al. [59] For Internet of Things applications, the authors show that distributed threat detection at the fog level is more scalable than centralized cloud-based methods. When it comes to cyberattack detection accuracy, deep learning models perform better than shallow machine learning models. The Extreme Learning Machine (ELM) is a fast-learning algorithm that surpasses typical neural networks in terms of generalization power and convergence speed, according to Prabavathy et al. [62]. In the Internet of Things (IoT) context, cyberattacks are classified in real time using OS-ELM, the online version of ELM. Kumar et al. [63] In order to assess anomaly IDS's efficacy, the article suggests using machine learning techniques like Random Forest, K-Nearest Neighbor, and Decision Tree. Almiani et al. [64] A model that adjusts a recurrent neural network trained by an improved backpropagation method was proposed in this study. For more in-depth analysis, the performance was assessed using a range of metrics, including Cohen's Kappa coefficients and Mathew correlation. The DAE-BiLSTM model, as proposed by Selim et al. [67], combines a Bidirectional Long Short-Term Memory (BiLSTM) with a Deep AutoEncoder (DAE) to provide effective feature extraction, noise reduction, and network traffic analysis. For attack and anomaly detection, Rashid et al. [68] investigate a number of machine learning techniques, including as LR, SVM, DT, RF, ANN, and KNN. Additionally, they look into group techniques like stacking, boosting, and bagging to enhance the detection system's functionality. Alzahrani et al. [69] extract features from the data using a correlation-based method to eliminate unnecessary information,

producing a model with no computational overhead. They present the Tab transformer model, which was created especially to handle continuous data. The difficulty of unbalanced data and its effect on the performance of underrepresented classes are acknowledged in the text. A threat intelligence technique was presented by Almaiah et al. [71] to examine and analyze the behavior and propagation of Shamoon assaults in the fog computing edges. However, an anomaly detection system based on the Random Forest machine learning method is proposed by Alrashdi et al. [73]. According to Hameed et al. [72], Weighted Majority Algorithms, or WMAs, are a new class of Ensemble Learning fog-based solutions that are well suited for Internet of Things applications in the medical field. Unstructured medical data must be protected, according to Moqurrah et al. [79]. In their work, Onah et al. [83] provide an improved IDS model called GANBADM (Genetic Algorithm based feature selection and Nave Bayes for Anomaly Detection Model), which combines a Genetic Algorithm Wrapper-Based feature selection approach with a Nave Bayes Classifier. Kumar et al. [84] This study suggests using the interplanetary file system (IPFS) for off-chain storage in blockchain networks and uses artificial intelligence (AI) for real-time analysis and decision-making in blockchain-IoT systems. The 45-variable UNSW-NB15 dataset is used by Zakariah et al. [85] to identify the most pertinent characteristics for IoT system anomaly detection. They build a robust anomaly detection model for IoT devices using a random forest classifier, feature engineering techniques, and active learning methodologies. Impressively, the suggested model's accuracy percentage was 99.7%. A hierarchical strategy in fog and cloud situations is proposed in other articles. A lightweight IDS known as the sample selected extreme learning machine (SS-ELM) is proposed by the authors [36]. The inability of fog nodes

and MEC hosts to hold substantial volumes of training data sets is the driving force behind SS-ELM. Rather, cloud servers store, process, and sample the data sets. The IDS is then trained on fog nodes and MEC hosts using the chosen samples. This strategy improves detection accuracy while cutting down on training time. The fog computing platform suggested by Ramkumar et al. [41] consists of three layers: fog, end point, and cloud. While end point layer handles physical processing and fog layer intrusion detection, cloud layer handles data transformation, feature selection, and classification utilizing log transformation. The RSLO-based ensemble method produces results with excellent accuracy, F-measure, recall, and precision. According to Labiod et al. [53], an intrusion detection system that uses a two-layered fog architecture operates by placing an attack identification module in the cloud and an anomaly detector inside fog nodes. This approach restricts connection between fog nodes and the cloud, allowing real-time data processing while simultaneously lowering latency. To efficiently detect anomalies in network traffic, Xu et al. [66] underline the requirement for a fully functional infrastructure backed by cloud computing and fog computing in addition to a data-driven core engine semi-supervised technique. In order to enhance IoT network security, Roy et al. [74] suggest a two-layer hierarchical detection technique in the Fog-Cloud Infrastructure. The fog layer uses a straightforward feedforward neural network enhanced by a stacking autoencoder for binary classification, while the cloud layer uses a more intricate feedforward neural network for multi-class classification.

3.3 Summary

The literature review chapter provides a comprehensive overview of the existing research on intrusion detection systems (IDS) in the fog computing (FC) layer. Through an extensive examination of previous studies, two main findings emerge:

Firstly, a significant portion of the research in the FC field has neglected the crucial aspect of energy and CPU consumption. Many IDS studies have primarily focused on improving detection accuracy and functionality without explicitly considering the energy constraints of resource-constrained fog devices. This oversight highlights a critical gap in the literature and emphasizes the need for energy-efficient IDS solutions in the FC context.

Secondly, the review reveals that several studies have relied on outdated datasets, which fail to capture the vast majority of modern attacks. This limitation hinders the effectiveness and relevance of IDS models in detecting emerging threats. To address this issue, there is a need for the integration of modern and comprehensive datasets.

Thirdly, the literature review reveals that the deployment of IDS models' hierarchical approach in fog and cloud environments has been explored in previous studies. Some researchers have investigated the benefits of deploying IDS functionality only in cloud layer environments. These approaches allow for distributed detection, improved scalability, and optimized resource utilization. However, it is worth noting that a few studies have focused specifically on deploying IDS solely in the fog layer, emphasizing the advantages of localized detection and reducing the reliance on cloud resources. Localized detection in the fog can minimize latency, reduce network traffic, and provide

faster response times, making it particularly suitable for time-sensitive or critical applications.

CHAPTER 4

METHODOLOGY AND ANALYSIS

This section provides a comprehensive description of the proposed IDS model. The model aims to distribute and decentralize the cloud architecture to fog nodes. First, define the objectives as a problem statement. Followed by the preprocessing of the chosen dataset that was used for building the model on the fog layer. Next, demonstrate the building process of the hybrid machine learning algorithms. Then it covers the evaluation metrics used to assess the model's performance and concludes with a discussion of the evaluation environment and experiment setup.

4.1 Problem Statement

The objective of this research is to address the challenges of building energy-aware, anomaly-based Intrusion Detection Systems (IDS) in the fog layer, with a focus on reducing and mitigating overhead. The problem at hand encompasses several key aspects:

Firstly, existing IDS models often rely on outdated datasets that fail to capture the vast majority of modern attacks, such as [28], [29], and [30]. To overcome this limitation, this study proposes the use of a modern dataset, CICIoT2023, which includes a comprehensive range of contemporary attack scenarios. By leveraging this dataset, the

proposed IDS model aims to enhance detection accuracy and ensure its relevance in the context of emerging threats.

Secondly, to achieve effective attack classification, a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) machine learning model will be employed, which involves dataset preprocessing, constructing the CNN layers, transitioning to LSTM layers, training the model, evaluating the model, and finally optimizing the model based on evaluation results. This hybrid architecture is capable of handling both binary-class and multi-class attack classification tasks while considering the energy consumption of the model. By optimizing the energy efficiency of the IDS model, the research seeks to minimize resource consumption and improve the sustainability of fog computing environments.

Furthermore, the study will explore the practicality of deploying IDSs using limited resources, specifically utilizing the Raspberry Pi as a Fog node. This investigation aims to demonstrate the feasibility of implementing IDS functionality in resource-constrained fog computing environments, thereby expanding the applicability of IDS solutions. The performance evaluation of the proposed IDS model will be conducted based on key metrics, including latency, energy consumption, and detection rate. By assessing these metrics, the research aims to quantify the efficiency and effectiveness of the model in comparison to existing techniques, with a particular emphasis on improving accuracy and reducing the false positive rate.

4.2 Research Methods

Power is one of the most limited resources in the Internet of Things system; the IDS must be designed in a way that is energy-efficient to utilize power moderately. This study aimed to evaluate the IDS model with different datasets in order to identify areas where the model performed well and assess its generalization capabilities. In addition, to know how much the IDS model based on the dataset consumes energy and time.

First, this study proposes the use of a modern dataset, CICIoT2023, which includes a comprehensive range of contemporary attack scenarios. The input data, in this case, the dataset, needs to be preprocessed. This involves tasks such as data labeling, normalization, and splitting the dataset to test, train, and validation. The data should be formatted appropriately to fit the input requirements of the CNN-LSTM model. Then, construct the Convolutional Neural Network (CNN) layers. These layers perform feature extraction by applying filters to the input data, capturing the patterns, and detecting relevant features. CNN layer typically includes convolutional operations, activation functions, and pooling operations to downsample the data.

After that, the output from the last CNN layer is then passed on to the Long Short-Term Memory (LSTM) layers. The LSTM layers are responsible for capturing dependencies and learning sequential patterns in the data. These layers maintain an internal memory mechanism that enables them to retain and utilize information from previous time steps. The constructed CNN-LSTM model needs to be trained using labeled data. This involves an optimization process where the model's parameters are adjusted to minimize a defined loss function. The training data consists of a dataset with corresponding attack labels.

During training, the model learns to recognize patterns indicative of different attack types.

After training, the performance of the CNN-LSTM model needs to be evaluated. This is typically done using a separate evaluation dataset that the model hasn't seen during training. Evaluation metrics such as accuracy, precision, recall, or F1-score are computed to assess the model's performance in classifying attacks. Based on the evaluation results, the model undergoes fine-tuning and optimization to improve its performance. This involved adjusting hyperparameters, modifying the architecture.

Once the CNN-LSTM model has been trained and optimized, it will explore the practicality of deploying IDSs using limited resources, such as the Raspberry Pi as a fog node. Inference involves feeding new, unseen network traffic data into the model to obtain predictions or probabilities of different attack types. Papers [9] and [27] are similar to the methodology used in this study. The authors test the model's performance using the Raspberry Pi. Figure 2 shows the proposed methodology of the IDS.

These steps provide an overview of how a CNN-LSTM model can be employed for attack classification in fog computing environments. The specific implementation details are in the next sections.

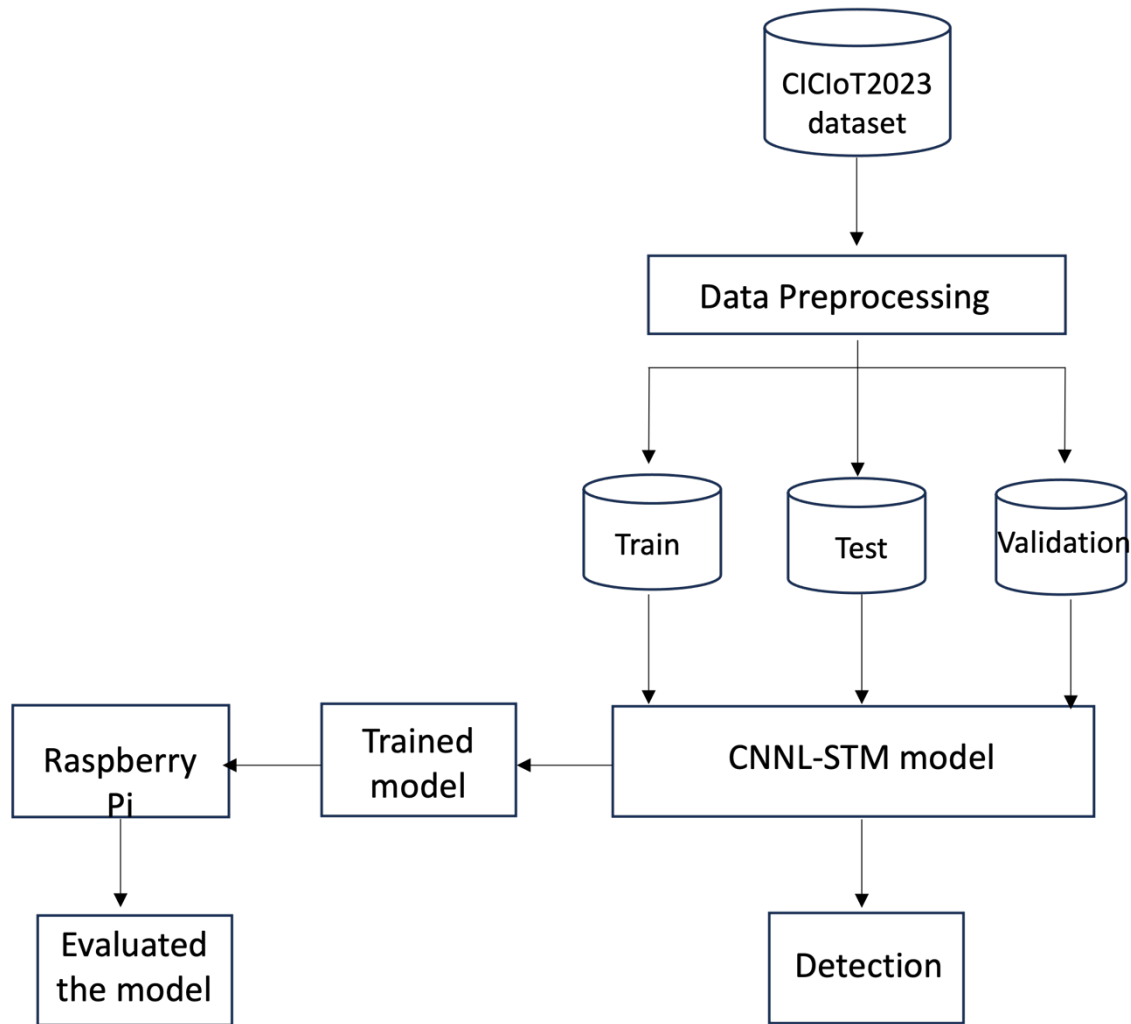


Figure 2 The proposed methodology of IDS for FC

4.3 Dataset Processing

Pre-processing is a crucial step in preparing data for machine learning models. The success of a machine learning algorithm relies heavily on the quality of the input data. The CICIoT2023 dataset introduces an up-to-date, and comprehensive collection of IoT attack data, offering valuable support for the development of security analytics

applications in real-world IoT operations. This dataset provides opportunities to optimize machine learning (ML) models, explore the impact of different features on ML models, and gain insights into classification interpretations.

Pre-processing involves various procedures to ensure the data is in an acceptable format and optimized for the machine learning algorithm. For instance, delete any spaces from feature names and update attack labels to their respective attack classes. Since machine learning models are based on mathematical algorithms and require numerical data, pre-processing involves converting different types of data, such as binary, integers, floats, strings, etc., into numerical values. The data normalization process converts attribute numeric values to a standard scale.

For the classification tasks, the dataset was categorized into binary and multi-class categories. For binary classification, the instances are distributed as normal and abnormal classes. While multi classification identify and categorize the different types of attacks that are present in the dataset. In this case, the attack classes are DDoS, DoS, Mirai, Benign Traffic, Spoofing, and Recon. Table 1 illustrates assigning the appropriate attack class labels to each instance in the dataset. By including the attack labeling step in the preprocessing phase, it ensures that the dataset is properly labeled with the corresponding attack class information. This labeled dataset serves as the foundation for supervised training of the CNN-LSTM model, enabling it to learn the patterns and relationships between the network traffic data and the attack types. If the dataset has a significant difference in the number of instances across attack labels or classes, it can lead to class imbalance. The number of instances of normal traffic is smaller than the number of instances of abnormal traffic in binary classification, however, the model performs well

on both classes. In multi-class classification, the dataset includes multiple attack classes, allowing the model to learn and differentiate between different types of attacks and normal traffic.

In data preprocessing, normalization is the process of translating numerical data to a common scale or range. Its goal is to bring the features or variables to a similar scale, preventing one feature from dominating or biasing the learning algorithm over others. Normalization is very crucial in machine learning applications where various features have varying sizes and ranges. The standardization technique transforms the data to have a mean of 0 and a standard deviation of 1.

To encode the labels, the Label Encoder function was employed for converting categorical labels into numerical representations. It ensures that each unique label is consistently mapped to a unique numerical value, enabling efficient computation during the training and prediction phases of machine learning models. As an illustration, use the binary classification 0,1 to designate abnormal and normal, respectively.

The features of the categorical variables and the dataset will determine which encoding method is used. It is important to note that one-hot encoding can increase the dimensionality of the dataset, especially when there are many unique categories. Attack labeling, on the other hand, addresses this problem by allocating the different labels to the relevant classes. One-hot encoding preserves the distinctiveness of categories, eliminating any unintended ordinality. Without one-hot encoding, the model might incorrectly assume a continuous relationship between categories, resulting in suboptimal performance.

Table 1 Distribution of attack labels on attack classes

Attack class	Attack labels	Numbers of instances
DDoS	UDP_Flood, SynonymousIP_Flood, ICMP_Flood, RSTFINFlood, SYN_Flood, UDP_Fragmentation, TCP_Flood, PSHACK_Flood, ICMP_Fragmentation, ACK_Fragmentation, HTTP_Flood, SlowLoris	328601
DoS	SYN_Flood, UDP_Flood, TCP_Flood, HTTP_Flood	78047
Mirai	greip_flood, greeth_flood, udpplain	25726
BenignTraffic		10507
Spoofing	MITM-ArpSpoofing, DNS_Spoofing	4731
Recon	HostDiscovery, PingSweep, OSScan, PortScan, VulnerabilityScan	3515

One-hot encoding is widely used to construct a binary representation of the labels in binary classification, where there are only two classes. For every distinct label, a new binary feature is created, with a value of 1 indicating the label's presence and 0 indicating its absence. This encoding strategy is appropriate for binary classification problems since it enables the machine learning model to interpret the labels in a binary format. In multi-class classification, the technique known as label binarization is employed to encode category labels. It converts the binary matrix representation of the categorical labels. In

the resulting matrix, each unique label is given its own binary column, with a value of 1 or 0 designating whether the label is present or absent. Machine learning models can effectively handle multi-class classification tasks because of this binary matrix representation, which represents each class as a distinct binary feature and reduces the issue of imbalanced class distributions that can impact performance. It allows the model to learn independently for each class label, potentially improving the representation and prediction of minority classes.

For splitting the dataset, the initial split is to divide the dataset into a training set and a testing set using an 80% to 20% ratio. The training set will be used to train the CNN-LSTM model, while the testing set will be used for the final evaluation of the model's performance. Next, the testing set is further split into a final testing set and a validation set using a 50%: 50% ratio. This secondary split allows for the fine-tuning of the model and hyperparameter adjustment. Figure 3 illustrates the initial and final splitting processes. The final testing set acts as the ultimate benchmark for evaluating the performance of the trained model. It remains untouched until the very end of the model development process. It helps provide an unbiased estimate of the model's performance when applied to completely unseen data. The validation set is used to fine-tune the model and adjust its hyperparameters. It serves as a proxy for evaluating the model's performance during the training process. By monitoring the model's performance on the validation set, adjustments can be made to improve its generalization capabilities and prevent overfitting. The CNN-LSTM model is trained using a training set, where it learns from labeled instances and adjusts its parameters to minimize the defined loss function. It is exposed to input data and attacks class labels, learning patterns, and relationships.

After each training iteration, the model's performance is evaluated using the validation set, which calculates metrics like accuracy, loss, precision, recall, or F1-score. Adjustments to hyperparameters, such as learning rate, batch size, or regularization techniques, are made based on the results. The final evaluation is conducted using the final testing set, assessing the model's ability to generalize to unseen data and accurately classify network traffic instances into attack classes. The evaluation metrics provide an unbiased estimate of the model's performance and effectiveness in detecting and classifying attacks.

By splitting the dataset into training, validation, and final testing sets, you can train and fine-tune the model on the training set, adjust hyperparameters using the validation set, and evaluate the final model's performance on the untouched final testing set. This process helps ensure that the model is robust and capable of generalizing well to new, unseen network traffic data.

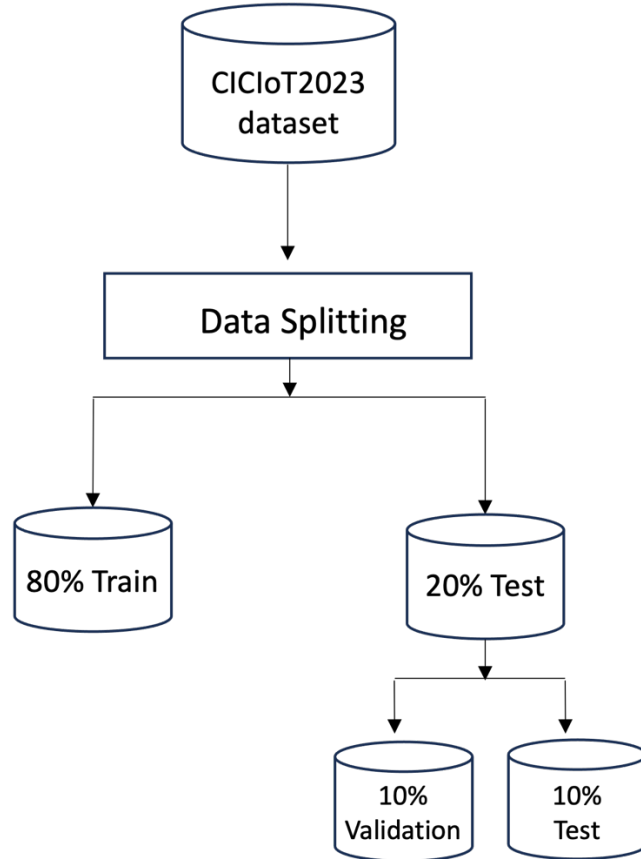
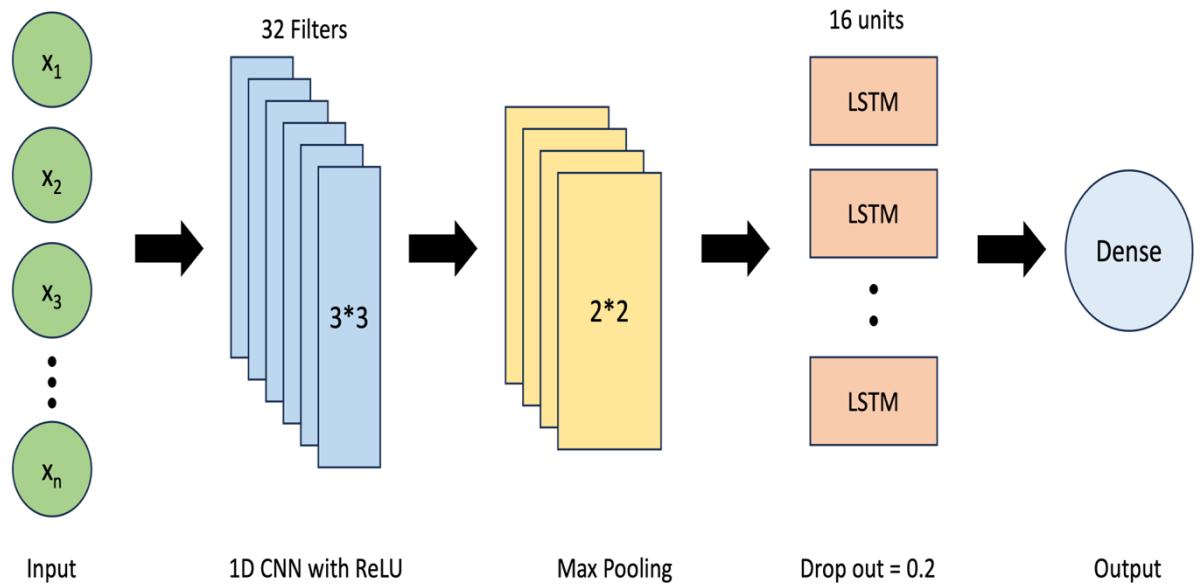


Figure 3 The dataset splitting process.

4.4 Anomaly Detection using Machine Learning Models

In the fields of machine learning and deep learning, the most advanced and high-performing models typically require a significant resource. For this effort, it is essential to conduct research and apply light-machine learning models to effectively implement and deploy them on low-power devices. The suggested IDS model uses long short-term memory and convolutional neural networks as hybrid machine learning methods to identify normal and abnormal traffic. The architecture of the CNN-LSTM model is shown in Figure 4.



[Figure 4 CNN-LSTM architecture]

The structure of the model with the function of each layer is as follows:

- **Conv1D Layer:** This layer performs 1-dimensional convolutions on the input. It has 32 filters, each with a kernel size of 9, which implies that the convolutional layer is designed to extract 32 different patterns, each considering a window of nine adjacent elements in the input sequence. Training the CNN with a large number of filters or a larger kernel size can be computationally expensive, requiring more memory and processing power. Choosing a moderate number of filters and a reasonable kernel size helps strike a balance between model performance and computational efficiency. The activation function used in this layer is ReLU (Rectified Linear Unit), which introduces non-linearity. ReLU is a simple, computationally efficient activation function that is ideal for large-scale IDS models due to its simplicity and ease of implementation. CNN layers benefit

from its ability to handle sparse activations, which help with regularization and model generalization by reducing the risk of overfitting. While LSTM layers can benefit from the non-linearity introduced by ReLU, enabling the model to capture long-term dependencies and temporal patterns in network traffic data.

$$ReLU(x) = \max(0, x)$$

Where x is the output of the first layer Conv1D, then passed into the second layer which is MaxPooling1D layer to reduce the feature map.

- **MaxPooling1D Layer:** This layer performs max pooling on the output of the Conv1D layer. It reduces the spatial dimensions by selecting the maximum value within a window of size 2 while retaining the most important features. The purpose of this layer is to down sample the feature maps, reducing computational complexity and controlling overfitting.
- **LSTM Layer:** This layer utilizes Long Short-Term Memory (LSTM) units for sequential processing. It has 16 units specifies the number of LSTM cells or memory units in the layer, which capture dependencies and patterns in the input sequence. This layer only returns the output at the last time step, rather than the full sequence of outputs. The dropout rate is set to 0.2, which means that 20% of the LSTM units' inputs will be randomly set to zero during training to prevent overfitting.
- **Dense Layer:** This layer is a fully connected layer with 1 unit indicates that the layer has a single neuron. It serves as the output layer of the model. The activation function used in this layer is sigmoid, which ensures that the output of the dense

layer is bounded between 0 and 1, representing the probability of the positive class for binary classification. The compatibility of sigmoid with different frameworks and libraries makes it a convenient choice, as it can be easily integrated into CNN-LSTM models for attack detection. The activation function that used for multi-class classification is SoftMax, which calculates the probabilities of each class. By employing SoftMax activation in the dense layer of a CNN-LSTM model for attack detection, the model can provide discrete class assignments, allowing unambiguous categorization of network traffic into specific attack types. Additionally, the stability of gradients ensures stable and efficient training, leading to improved performance and convergence of the model.

Additionally, the model is compiled with the following settings:

- **Loss Function:** A function for measuring the performance of an algorithm: if the predictions are incorrect, the loss function will return a greater value, whereas if the model makes correct predictions, the loss function will return a lower value. Cross-entropy, which evaluates the difference between two probability distributions for a given random variable or series of events, is commonly employed as a loss function when optimizing classification models. Binary-Crossentropy loss function is commonly used for binary classification problems. Categorical-Crossentropy, which is appropriate for multi-class classification problems.

$$\text{Binary} - \text{loss} = -(y * \log(p) + (1 - y) * \log(1 - p))$$

where y represents the true label of the sample (either 0 or 1). p represents the predicted probability of the positive class (class 1).

$$\text{Categorical-Crossentropy loss} = -\sum(y * \log(p))$$

where $\sum$ represents the summation over all classes, y is a one-hot encoded vector representing the true label of the sample. It has a value of 1 for the true class and 0 for all other classes. The loss is computed by taking the negative sum of the element-wise multiplication of y and $\log(p)$.

- **Optimizer: Adam.** It is an optimization algorithm that adapts the learning rate during training to improve convergence. It adjusts the parameters based on the gradients of the loss function with respect to the parameters computed during backpropagation using the chain rule.

The Conv1D layer performs convolutions to extract features, the MaxPooling1D layer down samples the feature maps, the LSTM layer captures sequential dependencies, and the Dense layer produces the final output. Together, these layers form a CNN-LSTM architecture commonly used for sequence classification tasks. Throughout the fine-tuning and optimization process, model architecture modifications that involve adding or removing layers and adjusting the number of units or filters in each layer have been made in order to enhance its performance. The goal is to adapt the architecture to better capture the underlying patterns in the data and improve the trade-off between energy efficiency and detection rate. In summary, the combination of CNN and LSTM layers in an attack detection model provides the ability to capture both spatial and temporal information, perform hierarchical feature learning, and leverage parameter efficiency for improved generalization. This combination enhances the model's ability to detect attacks by effectively capturing the complex and dynamic nature of network traffic data. By integrating CNN and LSTM layers, the model can achieve a better balance between

feature extraction and capturing long-term dependencies compared to using only CNN layers. This can reduce the need for a very deep network with many convolutional layers, addressing the challenge of modeling long-term dependencies in sequences and potentially alleviating the requirement for an excessively deep architecture. While previous studies have demonstrated high detection performance, understanding the energy consumption implications of CNN-LSTM is crucial for practical deployment and resource management. Table 2 presents the final model parameters that adopted for better results (CNN-LSTM Binary-class and CNN-LSTM Multi-class) compared to the parameters of the previous models.

Table 2 CNN-LSTM parameters

ML-IDS	Layers	Kernel/Neurons	Activation/loss	Optimizer	Epoch	Batch size	Total parameters
CNN-LSTM Binary-class	Conv1D	(9 /32)	ReLU	Adam	20	128	3,473
	MaxPooling	2	-				
	LSTM	16	-				
	Dense	1	Sigmoid/ binary_crossentropy				
Optimized CNN-LSTM Multi-class	Conv1D	(9/32)	ReLU	Adam	20	128	3,558
	MaxPooling	2	-				
	LSTM	16	-				
	Dense	6	Sofmax/ categorical crossentropy				
Initial draft CNN-LSTM model	Conv1D	(3/16)	ReLU	Adam	10	32	1,170
	MaxPooling	4	-				
	LSTM	8	tanh				
	Dense	34	Sofmax/ categorical crossentropy				
Second draft CNN-LSTM model	Conv1D	(5/16)	ReLU	Adam	10	32	23,042
	MaxPooling	4	-				
	LSTM	64	tanh				
	Dense	34	Sofmax/ categorical_crossentropy				

As shown in the table, the initial draft start with low parameters values. Conv1D layer with 16 filters and a kernel size of 3 with ReLU activation function and Adam optimizer. While the LSTM layer has 8 units and tanh activation. Dense layer 34 units which is the classification output of the model before labeling the dataset attack classes. The number of parameters in a model affects its capacity to learn complex patterns. The initial draft model has the smallest number of parameters (1,170), which may result in lower memory consumption. It has a relatively simple architecture with a small number of parameters. This simplicity may help with faster training and lower memory consumption. However, it limits the model's ability to capture complicated patterns in the data, resulting in poor model performance in terms of detecting attacks.

The second draft model builds upon the initial draft and introduces several enhancements. The number of filters is increased from 16 to 32, providing the model with more capacity to extract features from the input data. Additionally, the kernel size is increased from 3 to 5, enabling the model to capture broader patterns. The number of units in the LSTM layer is increased from 8 to 64, which allows the model to capture more long-term dependencies and complex temporal patterns in the data. The number of parameters is significantly increased to 23,042, indicating a higher capacity to learn from the data. By increasing the capacity and complexity of the model, the second draft aims to improve its ability to capture more intricate patterns and relationships in the data. This can potentially lead to better performance in terms of predication. However, it's important to note that increasing the number of parameters can also increase the risk of overfitting, especially with training the model on 34 labels of attack introduces more complexity and increases the model's capacity to handle a larger number of attack classes. The model

needs to learn to differentiate between a broader range of classes, which can be more challenging due to the increased number of parameters and potential overlaps between classes. It's important to consider the class imbalance in the dataset. Some labels have significantly more instances than others, the model can be biased towards the majority classes and struggle to accurately classify the minority classes.

It is worth noting that the tanh function is a non-linear activation function that squashes the input values to the range $[-1, 1]$. In the context of recurrent neural networks, such as LSTMs, the tanh activation function is commonly used because it can capture long-term dependencies and produce outputs that are centered around 0. However, the choice of activation function depends on the specific problem and the characteristics of the data. As shown from the prior and optimized model, there is no noticeable distinction in the model's performance in this instance when the tanh function is not used.

The optimized model further refines the architecture, parameters, and training configuration based on the insights gained from the previous drafts. It aims to strike a balance between model complexity and performance. The increase in the kernel size and filter count in the convolutional layer allows the model to capture more diverse and complex features. Additionally, reducing the number of LSTM units helps mitigate overfitting while still preserving the model's ability to capture relevant temporal information. Classifying the attack labels in 6 classes mitigate the data imbalance and improve training efficiency. In order to identify the combination of epoch and batch size that provides the best balance between training time and model performance and consider the stability and overall performance on the validation set, the optimized version involves iterating through the entire dataset 20 times (epochs) while using a batch size of 128.

4.5 Evaluation Metrics

When evaluating the CNN-LSTM model, several metrics can be used to assess its performance. Here are some commonly used evaluation metrics along with their equations:

- True Positive (TP) indicate the number of attack traffic detected as attack traffic.
- True Negative (TN) indicate the number of normal traffic detected as normal traffic.
- False Positive (FP) indicate the number of attack traffic detected as normal traffic.
- False Negative (FN) indicate the number of normal traffic detected as attack traffic.

Accuracy (ACC): Accuracy measures the overall correctness of the model's predictions.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

Precision: Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive.

$$Precision = TP / (TP + FP)$$

Recall: measures the proportion of correctly predicted positive instances out of all actual positive instances.

$$Recall = TP / (TP + FN)$$

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a balanced evaluation metric.

$$F1\ Score = 2 * (Precision * Recall) / (Precision + Recall)$$

Area Under the Receiver Operating Characteristic curve (AUC-ROC): AUC-ROC measures the model's ability to distinguish between classes by plotting the true positive rate (TPR) against the false positive rate (FPR). AUC-ROC represents the area under the ROC curve, which ranges from 0 to 1. A value closer to 1 indicates better classification performance.

False Positive (FP): False positive is a metric used in binary classification problems, representing the number of negative instances incorrectly classified as positive by the model. It measures the model's propensity to generate false alarms or false positives.

$$False\ Positive = FP / (FP + TN)$$

4.6 Evaluation Environment

4.6.1 Software and Libraries

The ML models are implemented using python 3.9. Numpy, Scikit-learn and Pandas libraries for manipulate and preprocess data before feeding it into machine learning models. Matplotlib and seaborn for data visualization and analysis. TensorFlow's Keras provides classes and functions for defining and training neural network models.

4.6.2 Hardware

The experiment is tested on Raspberry Pi 3 B+, the processor is a 64-bit quad-core ARM Cortex-A53 CPU and 1 GB LPDDR2 SDRAM main memory. Raspberry Pi has been deployed as a fog node in many IoT applications, for example [9][29]. The power consumption and latency at different phases of the model simulation was investigated using Raspberry Pi with the Arduino Uno. Table 3 presents the evaluation environment specification that have been used in this study.

$$P = I * V$$

This equation calculates the power consumption (P) of the Raspberry Pi by multiplying the current (I) passing through the shunt resistor by the voltage across the Raspberry Pi. The current (I) passing through the shunt resistor (R) (1Ω) is the same current passing through the RPi since the two are connected in series. The voltage across the Raspberry Pi (V_{RPi}) measured by subtracting the voltage with the Arduino Uno from the power supply voltage. The power consumption was sampled every 2000 ms.

Table 3 Evaluation environment specifications

CPU	64-bit quad-core ARM Cortex-A53
RAM	1 GB
TensorFlow	2.7
Python	3.9
Arduino	Uno

4.7 Experiment Setup

- **Training and Validation:** The ML models were trained and validated on a PC with a 1.8 GHz Dual-Core Intel Core i5 processor. The classifiers were implemented using Python 3.9. This phase involved feeding the training data to the models, adjusting the model parameters, and iteratively optimizing the models' performance. The validation step ensured that the models were not overfitting and performed well on unseen data.
- **Transfer to Raspberry Pi:** Once the models were trained and saved, they were transferred to the Raspberry Pi 3 Model B+. This involved deploying the model weight as h5 file and architecture as json file to the Raspberry Pi's storage for later use during testing.

- **Dataset and Model on Raspberry Pi:** The datasets used for testing had a sample size of 1000 instances and were also transferred to the Raspberry Pi. These datasets were loaded onto the Raspberry Pi's storage along with the trained ML models.
- **Energy Consumption and Latency Measurement:** The energy consumption and latency were investigated to assess the efficiency of the model simulations during testing. The experimental setup, as illustrated in Figure 5, involved using a Raspberry Pi, an Arduino, and a resistor shunt. The resistor shunt was used to measure the current flowing through the circuit, and the Arduino was used to record the power consumption as a measurement device. The equation $P = I * V$, where P represents power, I represents current, and V represents voltage, was used to calculate the power consumed by the Raspberry Pi.
- **Code Execution and Latency Measurement:** The code for testing the models was executed on the Raspberry Pi through an SSH connection in order to reduce power consumption. This allowed for remote access and control of the Raspberry Pi. The main phases of running the code were recorded, including the import of necessary packages, loading the dataset sample, preprocessing the dataset, and making predictions. Each code execution was performed multiple times manually to ensure accuracy and to avoid errors.
- **Latency Recording:** During code execution, the latency for each step was recorded. This involved measuring the time taken for each phase, such as package import, dataset loading, dataset preprocessing, and prediction. By recording the

latency, the performance of the lightweight IDS could be evaluated in terms of response time.

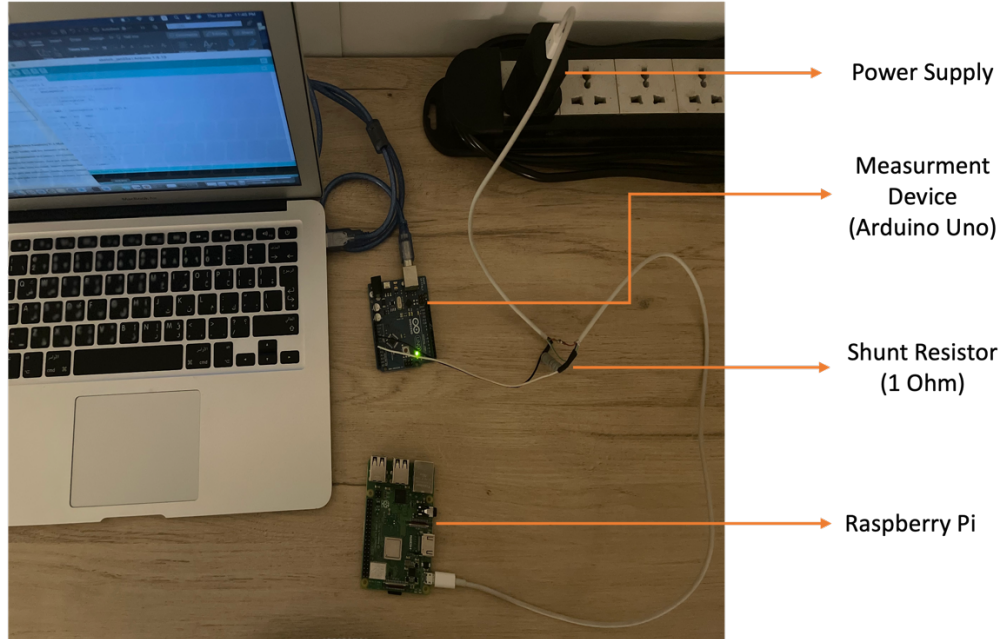


Figure 5 Experiment setup

Figure 6 shows the circuit diagram details of experiment to measure the power pass to RPi.

- The Power Supply: Connect the positive terminal of the power supply to the positive terminal of the RPi.
- Ground Connections: Connect the ground (GND) terminals of the RPi, Arduino Uno, and power supply together. This ensures a common reference point for the circuit.

- **Shunt Resistor:** Insert the 1-ohm shunt resistor between the positive terminal of the power supply and the positive terminal of the RPi. This resistor allows measurement of the current flowing through the circuit. The Arduino's ADC converts the voltage drop across the shunt resistor into a digital value that can be read by the microcontroller. The ADC samples the analog voltage and provides a digital representation of the voltage level.
- **Arduino Uno:** Connect the GND pin of the Arduino Uno to the circuit's ground (GND). This establishes a common ground reference.
- **Measurement Device:** Connect the A0 (analog input) pin of the Arduino Uno to a point between the shunt resistor and the positive terminal of the RPi. The Arduino Uno has several analog input pins, one of which is typically labeled as A0. These pins can read analog voltages ranging from 0 to 5 volts. This connection allows the Arduino Uno to measure the voltage drop across the shunt resistor, which is proportional to the current flowing through the circuit.

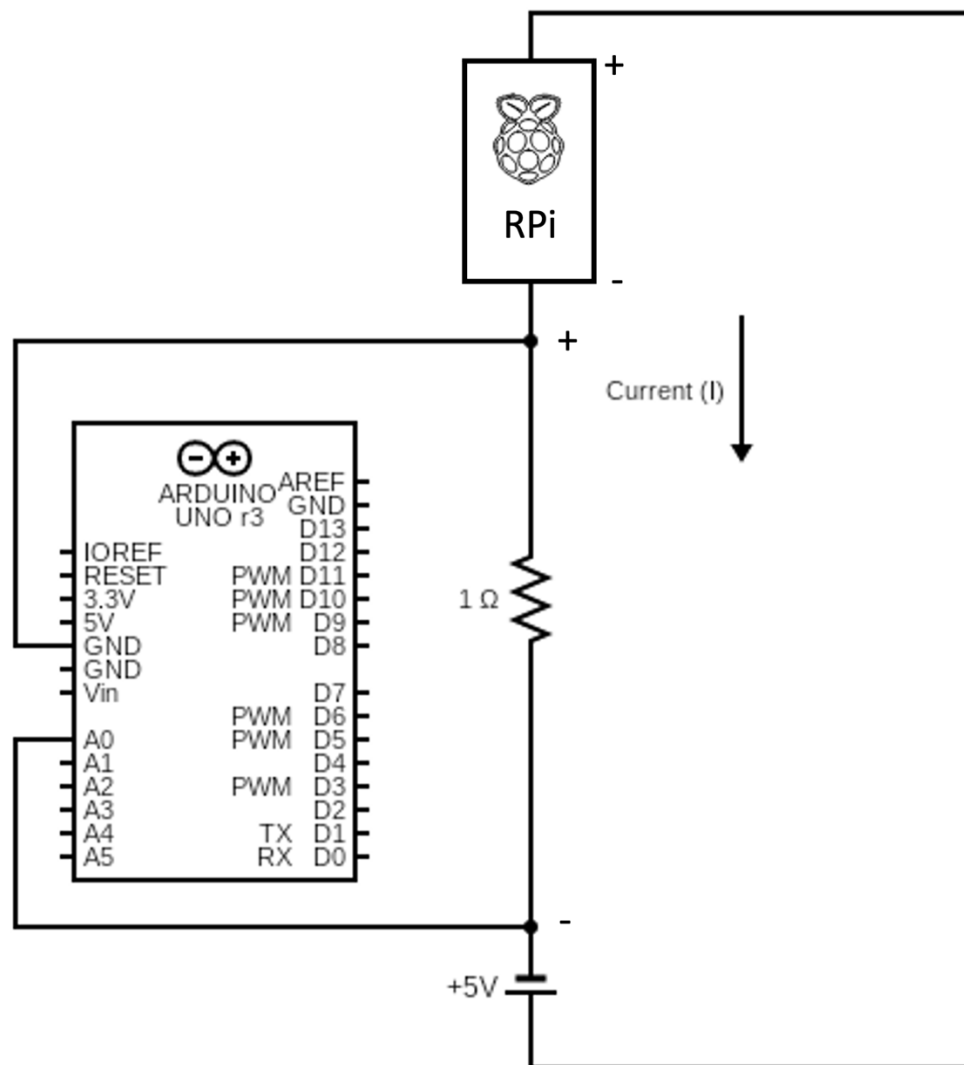


Figure 6 Circuit diagram.

CHAPTER 5

RESULTS

In this section, we conduct a detailed performance analysis and explain the findings of the ML models stated in the preceding paragraph. All models are benchmarked against the CICIoT2023, NSL, and KDD-99 datasets. The collected results are compared and examined with the goal of documenting and assessing model behavior on a recent dataset. Finally, the work was compared to earlier investigations.

5.1 Model Performance Evaluation

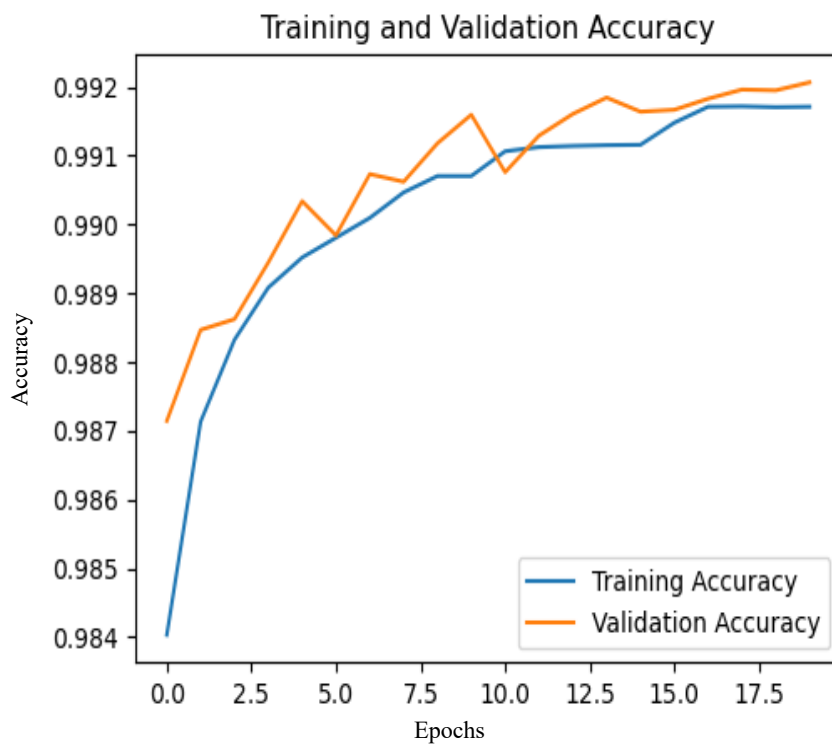
We first evaluate the detection quality of the proposed method for distinguishing between normal and anomalous network traffic. As shown in Figure 7, the training and validation accuracy values are reported at 99.17% and 99.21%, respectively. The training accuracy line shows how well the CNN-LSTM model performs on the training data as the training progresses. It measures the model's ability to correctly classify the training examples from the CICIoT2023 dataset. As the training continues, the model adjusts its internal parameters and learns to better recognize the attack patterns. The validation accuracy line, on the other hand, represents the model's performance on unseen data from the CICIoT2023 dataset. This accuracy serves as an important indicator of how well the model generalizes to new, unseen instances. It reflects the model's ability to effectively detect attacks in real-world fog

computing environments. The validation accuracy (99.21%) is slightly higher than the training accuracy (99.17%) suggests that the model is performing well and generalizing effectively without overfitting to the training data.

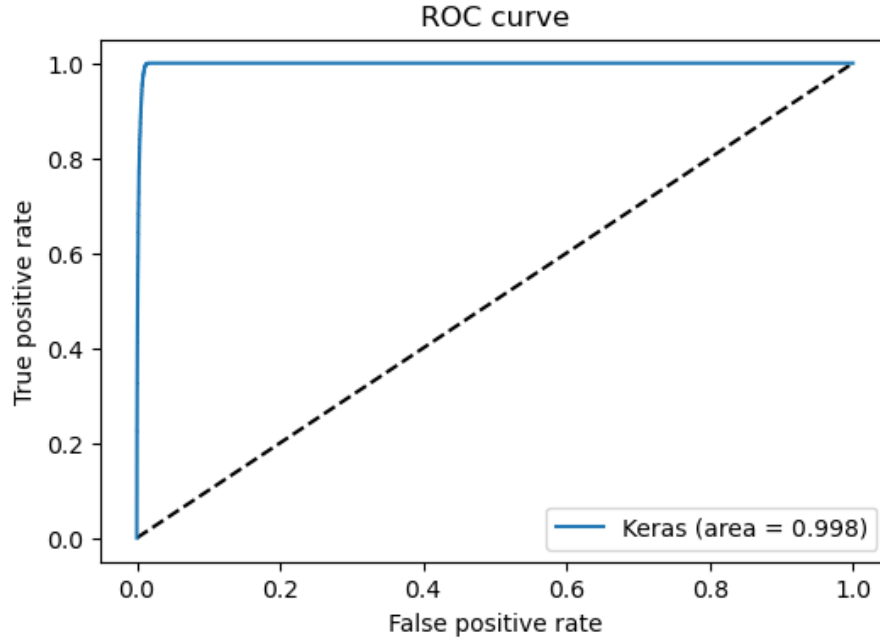
The performance of the binary-class model based on three evaluation metrics has been considered: Recall score, F1 score, and Precision score. Recall measures the proportion of actual positive instances (true positives) that were correctly identified by the model. In this model, a recall score of 94% suggests that the model has a high ability to correctly identify positive instances. The F1 score provides a balance between precision and recall, considering both false positives (precision) and false negatives (recall). In the binary-class model, an F1 score of 92% suggests a reasonable balance between precision and recall. Precision measures the proportion of instances that the model correctly predicted as positive (true positives) out of all positive predictions it made (true positives + false positives). A higher precision score indicates that the model has a low false positive rate. In this case, a precision score of 90% suggests that the model has the ability to limit false positive predictions.

These measures treat each class equally, irrespective of the class distribution or the number of samples in each class. However, in the current dataset, there are significantly more negative cases (attacks) than positive cases (normal). In ROC analysis, the FPR and TPR are used to plot the ROC curve, which visualizes the performance of a binary classification model across different classification thresholds. In Figure 8, the ROC curve illustrates the trade-off between the true positive rate and the false positive rate. A score of 99.79% indicates that the model is avoiding false positive instances, as it achieves a high TPR while keeping the FPR

low. In the top right corner of the graph, the TPR and FPR values are both at their highest. This suggests that the model is effectively identifying positive instances while minimizing false positives, and it is performing well in terms of its discrimination ability.



[Figure 7 Accuracy of CICIOT202 binary classification]



[Figure 8 ROC curve of CICIoT2023 binary classification.]

To evaluate the detection quality of the proposed method for distinguishing between different classes of the CICIoT2023 dataset, we labeled DDoS, DoS, Mirai, Recon, Spoofing, and Benign traffic. Figure 9 presents the training and validation accuracy values for the multi-class model, which are reported about 92.37% and 92.76%, respectively. The graph depicts that the model is performing well, as both lines are relatively close to each other. This suggests that the model is not overfitting or underfitting. As the model undergoes training iterations, or epochs, it gradually achieves higher accuracy values. This indicates that with each training iteration, the model improves its ability to classify the different types of attacks in the CICIoT2023 dataset. The model learns from the training data and adjusts its parameters, enabling it to make more accurate predictions over time.



Figure 9 Accuracy of multi classification.

The dataset is artificially balanced by increasing the number of instances in the minority classes through synthetic data generation using the oversampling technique. The model ensures that each class has a more balanced representation during training, which can help improve the performance and accuracy of the model, especially for the minority classes. The detection performance of the CNN-LSTM model for each class of test dataset is shown in Table 4. The confusion matrix is a crucial tool for evaluating the accuracy and performance of a classification model. It provides a comprehensive view of the model's performance by comparing the true class labels of the data with the predicted class labels. This matrix is especially useful when dealing with multi-class classification problems. It is clear to see that some classes are well classified, particularly those with a

high number of occurrences compared to others in the dataset. For example, Benign Traffic, DoS, and Mirai have very low misclassification rates, followed by DDoS, Recon, and Spoofing. Because of the similarity in data patterns, the models make mistakes when distinguishing between DDoS, Recon, and spoofing. The model demonstrates a high level of accuracy for classifying Benign Traffic with about 99.01%. The model achieves successful classification for DoS and Mirai traffic, with a very low misclassification rate. Mirai refers to a specific type of malware that targets Internet of Things (IoT) devices. This suggests that the model effectively distinguishes normal network traffic from other types of traffic. While the model shows a slightly higher misclassification rate for DDoS with 90.1%, Recon with 92.99%, and 91.29% for Spoofing. This suggests that the model can distinguish between DDoS, Recon, and Spoofing attacks from other types of traffic, although there may be some instances of misclassification. The similarity in data patterns between these classes can arise from certain characteristics they share. For example, both DDoS and Recon traffic can involve high volumes of network requests or scanning activities, which may lead to similarities in traffic patterns. Similarly, Spoofing can attempt to mimic legitimate traffic or disguise its origin, potentially leading to patterns that resemble normal network traffic. However, the classification is generally successful.

Figure 10 represents the ROC AUC with an overall 0.95 score that shows excellent performance. In terms of the ROC curve, random guessing corresponds to a diagonal line from the bottom-left corner to the top-right corner. This line represents the worst possible performance for a binary classification model. For Benign traffic, the model achieves an impressively low FPR of 0.02, indicating high accuracy in correctly classifying non-Benign instances. The TPR of 1.00 signifies that the model accurately detects all

instances of Mirai traffic. The model achieves a relatively low FPR of 0.10, meaning that it misclassifies only 10% of non-DDoS instances as positive. The TPR of 0.90 indicates that the model correctly identifies 90% of DDoS instances. The model's FPR and TPR show moderate misclassification rates for non-Recon and non-Spoofing instances, with 0.07 and 0.08, respectively.

Table 4 Detection performance for classes

Labels	DDoS	Benign Traffic	DoS	Mirai	Recon	Spoofing
DDoS	90.1%	0	0	0	7.45%	9.91%
Benign Traffic	0.01%	99.01%	2.95%	0.07%	0.13%	0
DoS	0	0.44%	98.99%	0.12%	0.17%	0.01%
Mirai	0	0.03%	0.08%	99.47%	0.19%	0.02%
Recon	6.82%	0	0	0	92.99%	4.55%
Spoofing	8.2%	0	0	0	3.51%	91.29%

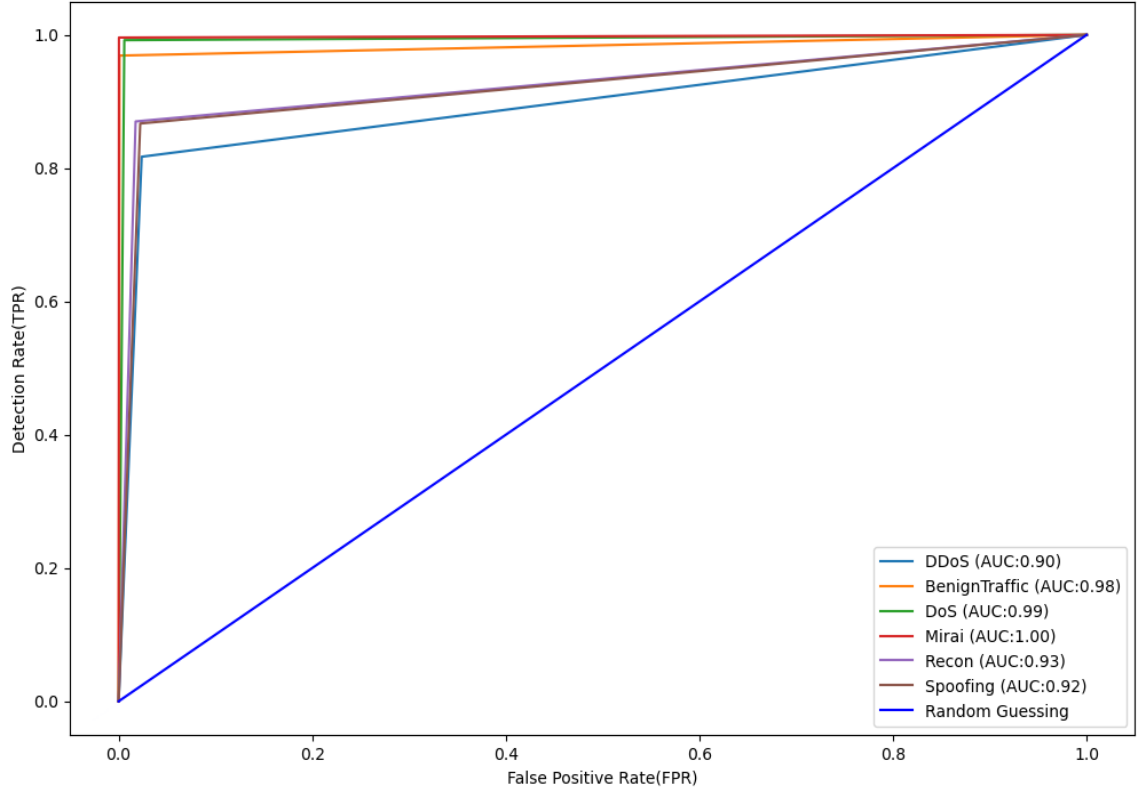


Figure 10 ROC AUC performance for multi-class CICIOT2023 dataset

The results of the evaluation metrics of precision, recall, and f1-score are 94%, 92%, and 93%, respectively. Furthermore, we compared the performance of the proposed model using both NSL-KDD and KDD-99. Figure 11 shows the performance of the CNN-LSTM model on the binary and multi class CICIOT2023, NSL-KDD, and KDD-99 test dataset. It is clear that the model achieved high accuracy rates, with the binary-class model achieving an impressive accuracy of 99.18% and the multi-class model achieving 92.92% accuracy. The precision values of 89% and 94% for the binary-class and multi-class models, respectively, indicate the ability of the models to correctly classify positive instances. The recall values of 95% and 92% for the binary-class and multi-class models, respectively, suggest that the models can effectively identify true positive instances. The F1-score, which combines precision and recall, is 92% for the binary-class model and

93% for the multi-class model, indicating a good balance between precision and recall. These metrics demonstrate the performance of the multi-class model on the NSL and KDD-99 datasets. The model achieved high accuracy rates on both datasets, with a slightly higher accuracy of 99.94% on the KDD-99 dataset. The false alarm rates are relatively low for both datasets NSL and KDD-99, indicating a good ability to avoid false positives. The multi classification of CICIOT2023 achieved a lower false alarm rate compared to other dataset, while KDD-99 had the highest accuracy. In general, the models provide excellent results across multiple datasets and evaluation metrics.



Figure 11 Performance evaluation of the model for the datasets

The variation in performance across the datasets could be attributed to several factors. NSL is a new version of the KDD-99 dataset. The variation in performance of the NSL and KDD-99 datasets is attributed to the fact that the KDD-99 contains many redundant records, which are extremely easy for classifiers to identify and reach high results. While

NSL does not include redundant records in the train set, the classifiers will not be biased towards more frequent records. The CICIoT2023 dataset encompasses both network traffic and IoT data, making it unique compared to the NSL and KDD99 datasets. The inclusion of IoT data introduces additional challenges, as IoT devices have distinct communication patterns and may exhibit different behaviors compared to traditional network traffic. The CICIoT2023, compared to NSL and KDD-99 is a larger and more diverse type of attack dataset, which provides the model with more opportunities to learn and improve its performance in terms of multi-classification.

The Raspberry Pi is a good platform for assessing the performance of the model because of its limited processing resources. The power consumption of the Raspberry Pi, which was utilized to mimic the fog node, peaks at 3.9 W during boot. In idle mode, consumption dropped to 3 W. Figure 12 displays the energy consumption that was recorded for each step of employing CNN-LSTM intrusion detection on the RPi with various datasets. However, the consumption of the cloud layer and the IoT layer are beyond the scope of this study. For the dataset in testing, the maximum power consumption was 6.12 W for the prediction step in the evaluation multi-class CICIoT2023 dataset. The KDD-99 dataset recorded the lowest power consumption in every single run compared with other datasets.

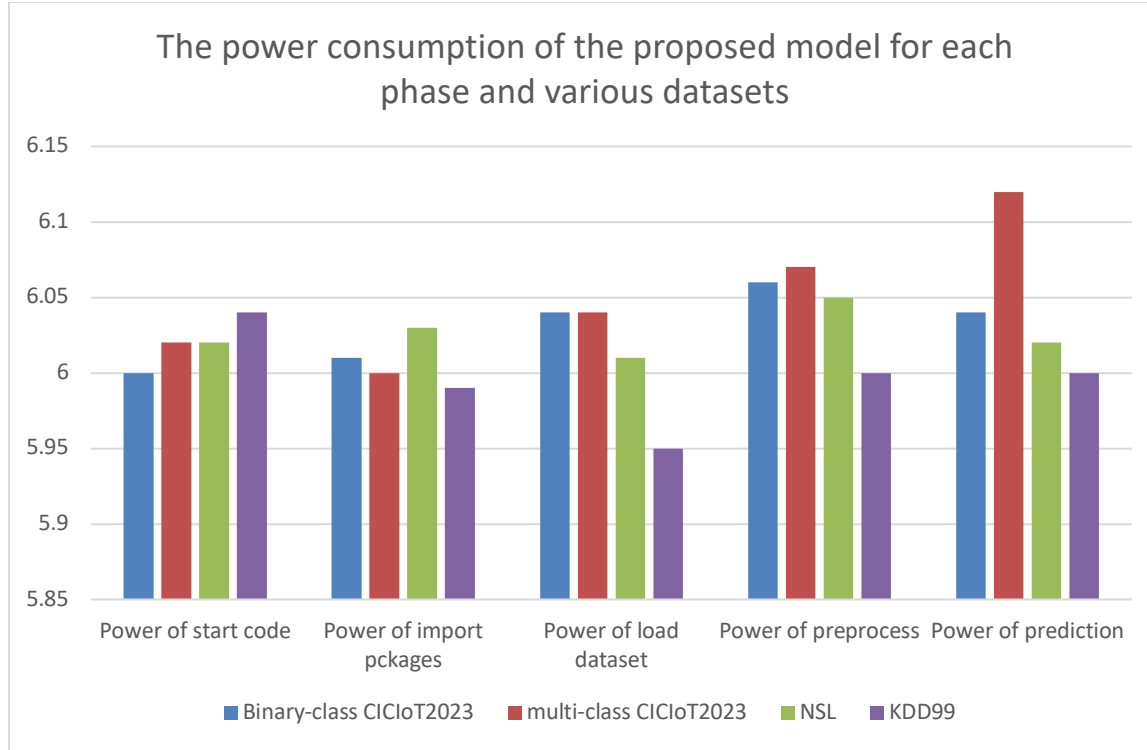


Figure 12 Energy consumption of the model steps.

We also investigated the latency of the proposed model. The latency of the model is the period between when an input is supplied to the model and when an output classifying the packet is detected. The results were obtained using a 1000-instances sample size. The dataset's samples were selected uniformly at random. This helps us acquire statistically accurate latency values. To prevent experimental mistakes, the model code was manually performed 12 times for each dataset. The latency for each step was embedded in the code for testing the model. To get the worst case of CNN-LSTM and different dataset loading and processing, the maximum result was presented. Figure 13 displays the model latency for CICIoT2023, KDD99, and NSL-KDD on the Raspberry Pi. It's worth noting that the presented results of energy and latency for each dataset are from a single run. Each phase of code execution has different latency and time consumption; for example, importing the

packages necessary to run the model has a different cost than uploading a sample dataset for inference, as import packages take more time. The high latency observed during the importing packages step in comparison to the load and preprocess dataset and prediction steps can be attributed to several factors: loading and initializing various libraries and packages required for the execution of the machine learning model. Some packages require configuration steps before they can be used effectively, such as loading model weights. However, with each subsequent run or iteration, the latency associated with importing packages tends to decrease. This reduction in import time can be attributed to the caching mechanisms. As a result, the overhead associated with package import diminishes over time, leading to improved overall execution speed.

$$\textit{Latency model} = ET - ST$$

Where ET represent the end time recorded by time function that embedded to code and ST is the start time of process.

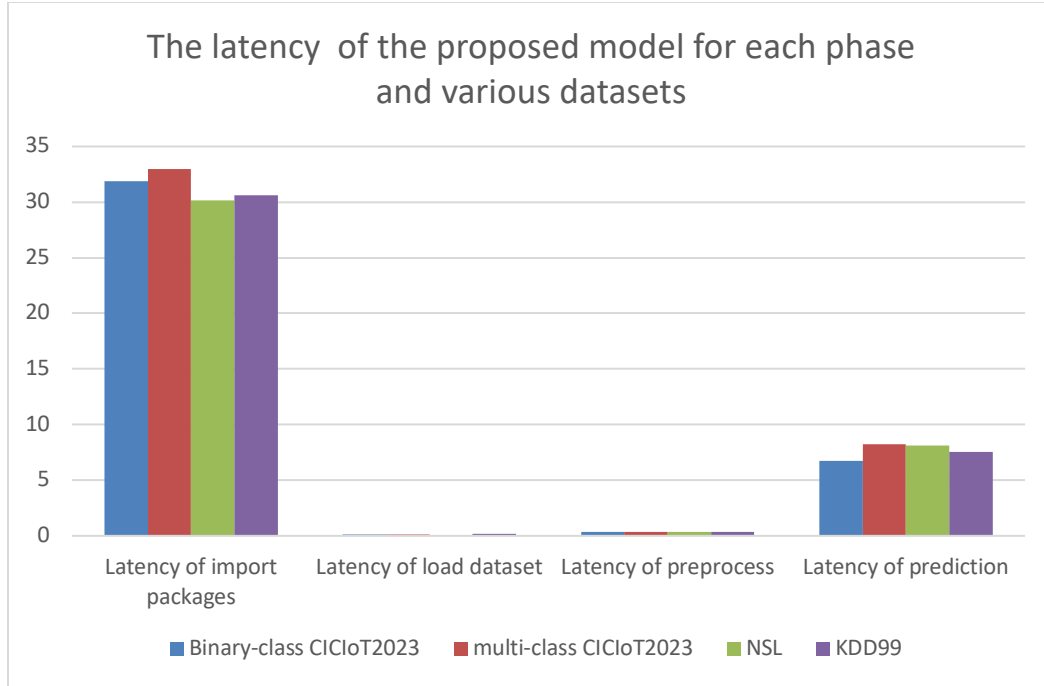


Figure 13 Latency of the model steps.

5.1.1 comparison with the initial versions of the model

This section compares three CNN-LSTM model versions: the initial draft, the second version, and the optimized version, which is the CICIOT2023 multi-classification. The initial draft model, with its simpler architecture, achieves moderate performance. The lower capacity of the model limits its ability to capture complex patterns, resulting in lower accuracy, precision, recall, and F1 score compared to the other models as Figure 14 illustrates. The second version of the model, with increased filter count, kernel size, and LSTM units, shows improved performance compared to the initial draft. The increased capacity allows the model to capture more intricate patterns and relationships in the dataset, resulting in higher accuracy, precision, recall, and F1 score compared to the first draft. The second draft model achieved an accuracy of 97%, precision of 0.91, recall of

0.60, and an F1 score of 0.62. These scores indicate that the model performed well in terms of overall accuracy and precision, but had relatively lower performance in terms of recall and F1 score. In this context, a recall of 0.60 suggests that the model might have missed a significant number of instances belonging to the positive class, resulting in false negatives. F1 score of 0.65 indicates that the model's performance is not well-balanced, with a relatively lower recall dragging down the overall F1 score. To address the low recall and F1 score, it necessary to fine-tune the model's architecture, parameters, and analyzing the class distribution help address the class imbalance and improve the model's ability to capture diverse patterns. These enhancements result indicate that the optimized model is very effective in detecting attacks. It demonstrates that the model is avoiding unnecessary alarms or false accusations, capture and flag as many attack instances as possible.

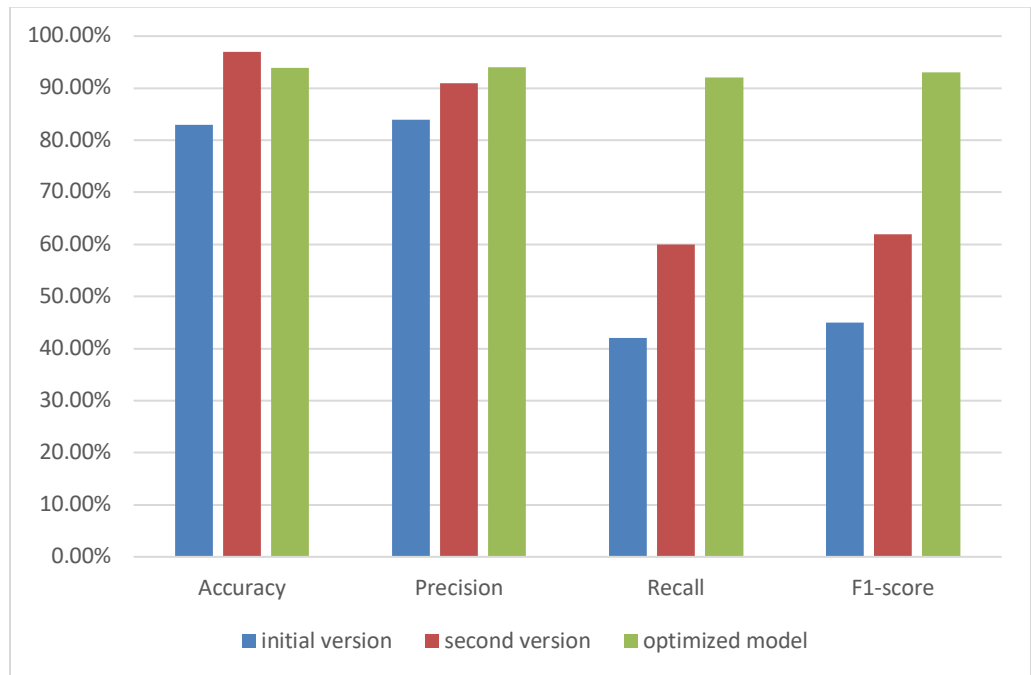


Figure 14 comparison of the optimized model performance and initial versions.

The initial model's performance was subpar, as it failed to effectively detect attacks. To address this, an enhanced second version of the model was developed. However, due to its inadequate accuracy, the second version was excluded from being loaded onto a Raspberry Pi for testing its energy consumption. In other words, since the model did not meet the required level of accuracy, there is no need to assess its energy consumption. In Figure.15 the graph presents the power consumption of the second and optimized model for the start code, import packages, load dataset, preprocess, and prediction phases. Looking at the power consumption values, we can see that the optimized model generally exhibits lower power consumption compared to the second draft model. For instance, in the load dataset phase, the optimized model consumes 6.04 w, while the second draft model consumes 6.05 units. In the preprocess phase, the optimized model consumes 6.07 units of power, whereas the second draft model consumes 6.13 w. Finally, in the prediction phase, the optimized model consumes 6.12 w of power, while the second draft model consumes 6.25 w. The optimized model showcases better power consumption efficiency across all phases, indicating that it can perform the required tasks while utilizing fewer resources.

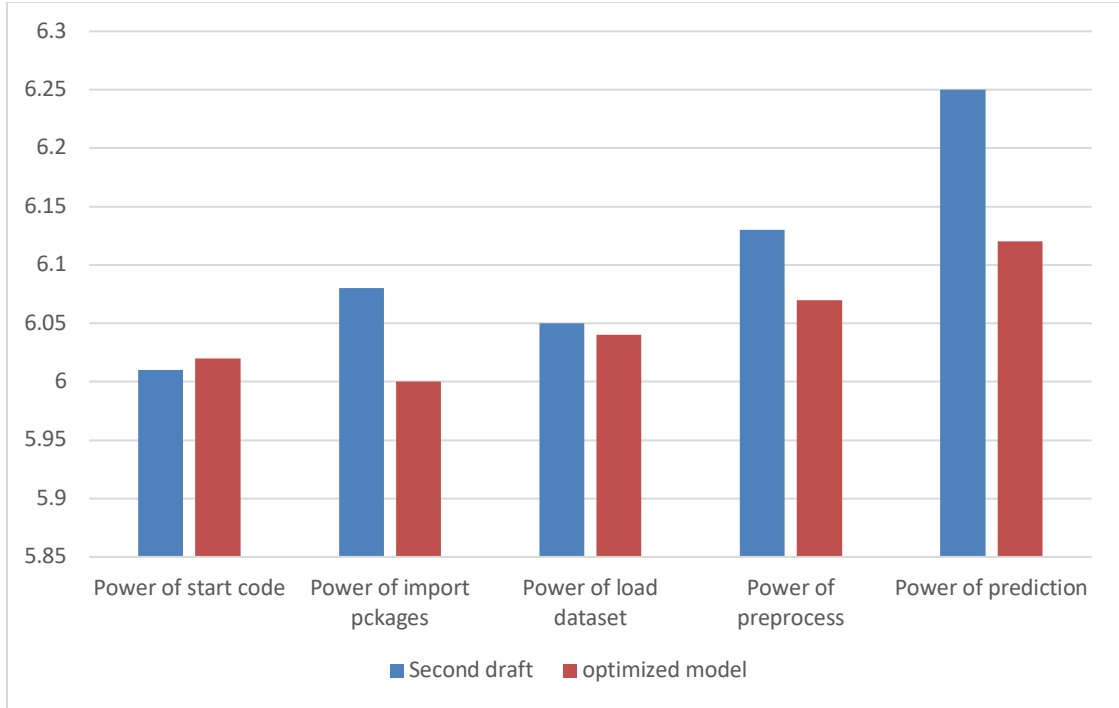


Figure 15 Power consumption of optimized and initial version.

5.2 Comparison with Related Work

This part compares the proposed system with previous studies. These papers were discussed in Chapter 3. This comparison is with the studies that proposed a hybrid classification model combining CNN with LSTM [8, 7, 50]. However, these studies did not take into account energy and CPU consumption. Table 5 presents the comparison in terms of the dataset used for evaluation, the number of epoches, accuracy, false alarm rate, and architecture design. It can be seen that the proposed system outperforms the previous studies in terms of false alarm rates.

Table 5 Comparison of the proposed system with papers in literature review

IDS	Dataset	Epochs	Accuracy	FAR	Architecture
[7]	CIDDS-001	10	99.92% multi-class	0.09	Centralized
[8]	NSL-KDD	50	96.5% multi-class	1.29	Centralized
[50]	NSL-KDD	100	96.07 binary-class	7.03	Distributed
Proposed model	CICIoT2023	20	99.10% binary-class	0.38 binary-class	Centralized
	NSL-KDD		92.92% multi-class	0.10 multi-class	
	KDD-99		99.22% NSL-KDD	0.32 NSL-KDD	
			99.94% KDD-99	0.12 KDD-99	

This part is a comparison with the previous studies that proposed a lightweight IDSs [9, 27, 28, 29, 30]. These studies have proposed different methodologies to develop efficient systems in resource-constrained environments. Due to the lack of a standardized generalization method, it becomes difficult to compare the performance of lightweight systems across different studies. Despite the difficulties in comparing studies directly, each individual study contributes valuable insights and advances our understanding of lightweight system design.

The authors in reference [30] focus on reducing the size of features and data points required for abnormality detection, resulting in lower computational complexity. They were concerned about the system overhead in terms of CPU and memory overhead. Their model used a constant memory of 3.4% CPU with a range of 1.5% to 2.9%.

Papers [28, 29] proposed IDS nodes and simulated them using OMNET++. In terms of lightweight, the studies concern about the energy and latency. The authors [29] investigate the performance of the Raspberry Pi and Orange Pi that are used in different layers of the FC system and record the parameters to simulate the system using OMNET++.

The papers [9, 27] are similar to the methodology used in the proposed work. The authors test the model's performance using the Raspberry Pi. The papers are concerned with the energy consumption and CPU time usage with respect to the number of nodes and gram.

Power is one of the most limited resources in an Internet of Things system, the IDS must be designed in a way that is energy-efficient to utilize power moderately. This study aimed to evaluate the IDS model with different datasets in order to identify areas where the model performed well and assess its generalization capabilities. In addition, to know how much the dataset consumes energy and time from the model, besides the model parameters. Table 6 shows a comparison with previous study in term of energy consumption, latency of detection, and accuracy. As a result, the maximum latency is approximately 8s for detection, and the energy consumption is 6w. Conclusively, the proposed CNN-LSTM model is accurate, energy-efficient, and light.

Table 6 Comparison of proposed model with lightweight IDS

IDS	Algorithm	Dataset	Accuracy	Energy Consumption	Latency of detection
[29]	ANN	KDD99	98.8%	2.45w	6.07s
		NSL-KDD	96.7%		6.27s
Proposed model	CNN-LSTM	CICIoT2023	99.10% binary	6.04w binary	6.76s binary
			92.92% multi	6.12w multi	8.23s multi
		NSL-KDD	99.22% NSL	6.02w NSL	8.11s NSL
		KDD-99	99.94% KDD-99	6.00w KDD	7.52s KDD

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In conclusion, this thesis focused on the development of a lightweight Intrusion Detection System (IDS) for IoT systems in fog computing environments. Fog computing, a distributed computing paradigm closer to the network edge, provides a solution for processing and analyzing data from IoT devices before sending it to the cloud. However, fog devices deployed in vulnerable areas are susceptible to cyberattacks. Therefore, the integration of IDS in fog computing is crucial to detecting and mitigating security risks within the fog infrastructure. Machine Learning (ML) algorithms have been employed in this research to enhance the IDS capabilities in fog computing. ML techniques can learn from datasets and identify patterns and anomalies that indicate potential security threats. By leveraging hybrid machine learning models, specifically Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) with the CICIOT2023 dataset, which includes a comprehensive range of contemporary attack scenarios.

The proposed model performance was evaluated using different dataset, such as NSL-KDD, KDD-99, and CICIOT2023, as binary and multi classification. Furthermore, the research explored the practicality of deploying IDSs with limited resources, specifically utilizing the Raspberry Pi as a fog node. The performance evaluation highlighted the

practicality of deploying IDSs with limited resources, as demonstrated by the successful implementation of IDS functionality on the Raspberry Pi as a fog node. Also, it is found that the model's accuracy and latency improve with every iteration. The highest accuracy result was for KDD-99, and lower false alarm rate for multi classification CICIoT2023. The maximum power consumption was for the data processing step in the evaluation of the multi-class CICIoT2023 dataset. The limitation of this work is that the performance and energy efficiency of the developed light machine learning models are influenced by the specific hardware platform they are deployed on. While these models are optimized for low-power device, such as the Raspberry Pi, it is important to note that the effectiveness of hardware acceleration or optimization techniques may vary across different platforms. Depending on the size and complexity of the dataset, running feature selection algorithms may require a significant amount of time and computational resources. Balancing the need for faster execution with the desire to identify the most informative features can be a challenge, as more exhaustive feature selection methods often require longer processing times.

6.2 Future Work

Future research can focus on further improving the detection techniques employed in the proposed IDS model. This could involve exploring advanced machine learning algorithms, and different feature selection algorithms, and investigating their effectiveness in detecting and classifying emerging attack patterns in fog computing environments yet without consuming resources. The research evaluated the proposed IDS

model using the Raspberry Pi as a Fog node. Future work could extend the evaluation to different fog computing scenarios, considering variations in network architectures, and variations in network traffic volume.

For future work, we recommend performing further experiments focusing on CPU, memory, and power consumption rates.

References

- [1] Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, F. (2017). Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88, 10-28.
- [2] Atlam, H. F., Walters, R. J., & Wills, G. B. (2018). Fog computing and the internet of things: A review. *big data and cognitive computing*, 2(2), 10.
- [3] Hindy, H., Bayne, E., Bures, M., Atkinson, R., Tachtatzis, C., & Bellekens, X. (2020, September). Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset). In *International networking conference* (pp. 73-84). Cham: Springer International Publishing.
- [4] Chen, Y., Ashizawa, N., Yeo, C. K., Yanai, N., & Yean, S. (2021). Multi-scale self-organizing map assisted deep autoencoding Gaussian mixture model for unsupervised intrusion detection. *Knowledge-Based Systems*, 224, 107086.
- [5] Pacheco, J., Benitez, V. H., Felix-Herran, L. C., & Satam, P. (2020). Artificial neural networks-based intrusion detection system for internet of things fog nodes. *IEEE Access*, 8, 73907-73918.
- [6] Jan, S. U., Ahmed, S., Shakhov, V., & Koo, I. (2019). Toward a lightweight intrusion detection system for the internet of things. *IEEE access*, 7, 42450-42471.
- [7] Ullah, I., Raza, B., Ali, S., Abbasi, I. A., Baseer, S., & Irshad, A. (2021). Software defined network enabled fog-to-things hybrid deep learning driven cyber threat detection system. *Security and Communication Networks*, 2021, 1-15.

- [8] Kalaivani, K., & Chinnadurai, M. (2021). A Hybrid Deep Learning Intrusion Detection Model for Fog Computing Environment. *Intelligent Automation & Soft Computing*, 30(1).
- [9] Sudqi Khater, B., Abdul Wahab, A. W. B., Idris, M. Y. I. B., Abdulla Hussain, M., & Ahmed Ibrahim, A. (2019). A lightweight perceptron-based intrusion detection system for fog computing. *applied sciences*, 9(1), 178.
- [10] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CIIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment.
- [11] Sabireen, H., & Neelanarayanan, V. J. I. E. (2021). A review on fog computing: Architecture, fog with IoT, algorithms and research challenges. *Ict Express*, 7(2), 162-176.
- [12] Yang, Y., Luo, X., Chu, X., Zhou, M. T., Yang, Y., Luo, X., ... & Zhou, M. T. (2020). Fog computing architecture and technologies. *Fog-Enabled Intelligent IoT Systems*, 39-60.
- [13] Lawal, M. A., Shaikh, R. A., & Hassan, S. R. (2020). An anomaly mitigation framework for iot using fog computing. *Electronics*, 9(10), 1565.
- [14] Hajiheidari, S., Wakil, K., Badri, M., & Navimipour, N. J. (2019). Intrusion detection systems in the Internet of things: A comprehensive investigation. *Computer Networks*, 160, 165-191.
- [15] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22.
- [16] Mohamed, R. H., Mosa, F. A., & Sadek, R. A. (2022). Efficient intrusion detection system for IoT environment. *International Journal of Advanced Computer Science and Applications*, 13(4).
- [17] Lawal, M. A., Shaikh, R. A., & Hassan, S. R. (2020). Security analysis of network anomalies mitigation schemes in IoT networks. *IEEE Access*, 8, 43355-43374.
- [18] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, 8, 1-74.

- [19] Olah, C. (2015, August 27). Understanding LSTM networks. Understanding LSTM Networks -- colah's blog. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [20] Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and communication networks*, 2020, 1-11.
- [21] Chaganti, R., Suliman, W., Ravi, V., & Dua, A. (2023). Deep learning approach for SDN-enabled intrusion detection system in IoT networks. *Information*, 14(1), 41.
- [22] Rathore, S., & Park, J. H. (2018). Semi-supervised learning based distributed attack detection framework for IoT. *Applied Soft Computing*, 72, 79-89.
- [23] Gunupudi, R. K., Nimmala, M., Gugulothu, N., & Gali, S. R. (2017). CLAPP: A self constructing feature clustering approach for anomaly detection. *Future Generation Computer Systems*, 74, 417-429.
- [24] McDermott, C. D., & Petrovski, A. (2017). Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks. *International journal of computer networks and communications*, 9(4).
- [25] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot, *Sensors* 17 (9) (2017) 1967.
- [26] Hosseinpour, F., Vahdani Amoli, P., Plosila, J., Hämäläinen, T., & Tenhunen, H. (2016). An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach. *International Journal of Digital Content Technology and its Applications*, 10(5).
- [27] Khater, B. S., Abdul Wahab, A. W., Idris, M. Y. I., Hussain, M. A., Ibrahim, A. A., Amin, M. A., & Shehadeh, H. A. (2021). Classifier performance evaluation for lightweight IDS using fog computing in IoT security. *Electronics*, 10(14), 1633.
- [28] Aliyu, F., Sheltami, T., Mahmoud, A., Al-Awami, L., & Yasar, A. (2021). Detecting man-in-the-middle attack in fog computing for social media.
- [29] Aliyu, F., Sheltami, T., Deriche, M., & Nasser, N. (2022). Human immune-based intrusion detection and prevention system for fog computing. *Journal of Network and Systems Management*, 30, 1-27.

- [30] Roy, S., Li, J., Choi, B. J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems*, 127, 276-285.
- [31] Hajj, S., Azar, J., Bou Abdo, J., Demerjian, J., Guyeux, C., Makhoul, A., & Ginhac, D. (2023). Cross-layer federated learning for lightweight iot intrusion detection systems. *Sensors*, 23(16), 7038.
- [32] De Souza, C. A., Westphall, C. B., & Machado, R. B. (2022). Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments. *Computers & Electrical Engineering*, 98, 107694.
- [33] Tu, S., Waqas, M., Rehman, S. U., Aamir, M., Rehman, O. U., Jianbiao, Z., & Chang, C. C. (2018). Security in fog computing: A novel technique to tackle an impersonation attack. *IEEE Access*, 6, 74993-75001.
- [34] Illy, P., Kaddoum, G., Moreira, C. M., Kaur, K., & Garg, S. (2019, April). Securing fog-to-things environment using intrusion detection system based on ensemble learning. In *2019 IEEE wireless communications and networking conference (WCNC)* (pp. 1-7). IEEE.
- [35] Moustafa, N., Choo, K. K. R., Radwan, I., & Camtepe, S. (2019). Outlier dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog. *IEEE Transactions on Information Forensics and Security*, 14(8), 1975-1987.
- [36] An, X., Zhou, X., Lü, X., Lin, F., & Yang, L. (2018). Sample selected extreme learning machine based intrusion detection in fog computing and MEC. *Wireless Communications and Mobile Computing*, 2018, 1-10.
- [37] Rahman, M. A., Asyhari, A. T., Leong, L. S., Satrya, G. B., Tao, M. H., & Zolkipli, M. F. (2020). Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustainable Cities and Society*, 61, 102324.
- [38] Rai, A. (2020, June). Optimizing a new intrusion detection system using ensemble methods and deep neural network. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)*(48184) (pp. 527-532). IEEE.
- [39] Moh, M., & Raju, R. (2018, July). Machine learning techniques for security of Internet of Things (IoT) and fog computing systems. In *2018 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 709-715). IEEE.

- [40] Diro, A., & Chilamkurti, N. (2018). Leveraging LSTM networks for attack detection in fog-to-things communications. *IEEE Communications Magazine*, 56(9), 124-130.
- [41] Ramkumar, M. P., Daniya, T., Paul, P. M., & Rajakumar, S. (2022). Intrusion detection using optimized ensemble classification in fog computing paradigm. *Knowledge-Based Systems*, 252, 109364.
- [42] Chen, Y., Lin, Q., Wei, W., Ji, J., Wong, K. C., & Coello, C. A. C. (2022). Intrusion detection using multi-objective evolutionary convolutional neural network for Internet of Things in Fog computing. *Knowledge-Based Systems*, 244, 108505.
- [43] de Araujo-Filho, P. F., Kaddoum, G., Campelo, D. R., Santos, A. G., Macêdo, D., & Zanchettin, C. (2020). Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment. *IEEE Internet of Things Journal*, 8(8), 6247-6256.
- [44] Sadaf, K., & Sultana, J. (2020). Intrusion detection based on autoencoder and isolation forest in fog computing. *IEEE Access*, 8, 167059-167068.
- [45] Meng, Y., Tu, S., Yu, J., & Huang, F. (2019). Intelligent attack defense scheme based on DQL algorithm in mobile fog computing. *Journal of Visual Communication and Image Representation*, 65, 102656.
- [46] Abdussami, A. A. (2021). Incremental deep neural network intrusion detection in fog based IoT environment: An optimization assisted framework. *Indian Journal of Computer Science and Engineering*, 12(6), 1847-1859.
- [47] De Souza, C. A., Westphall, C. B., Machado, R. B., Sobral, J. B. M., & dos Santos Vieira, G. (2020). Hybrid approach to intrusion detection in fog-based IoT environments. *Computer Networks*, 180, 107417.
- [48] Jiang, W., & Lv, S. (2021). Hierarchical deployment of deep neural networks based on fog computing inferred acceleration model. *Cluster Computing*, 24(4), 2807-2817.
- [49] Priyadarshini, R., Kumar Barik, R., & Dubey, H. (2020). Fog-SDN: A light mitigation scheme for DDoS attack in fog computing framework. *International Journal of Communication Systems*, 33(9), e4389.
- [50] Samy, A., Yu, H., & Zhang, H. (2020). Fog-based attack detection framework for internet of things using deep learning. *IEEE Access*, 8, 74571-74585.

- [51] Attique, D., Wang, H., & Wang, P. (2022). Fog-Assisted Deep-Learning-Empowered Intrusion Detection System for RPL-Based Resource-Constrained Smart Industries. *Sensors*, 22(23), 9416.
- [52] Gaurav, A., Gupta, B. B., Hsu, C. H., Yamaguchi, S., & Chui, K. T. (2021, January). Fog layer-based DDoS attack detection approach for internet-of-things (IoTs) devices. In 2021 IEEE international conference on consumer electronics (ICCE) (pp. 1-5). IEEE.
- [53] Labiod, Y., Amara Korba, A., & Ghoualmi, N. (2022). Fog computing-based intrusion detection architecture to protect iot networks. *Wireless Personal Communications*, 125(1), 231-259.
- [54] Alrashdi, I., Alqazzaz, A., Alharthi, R., Aloufi, E., Zohdy, M. A., & Ming, H. (2019, October). FBAD: Fog-based attack detection for IoT healthcare in smart cities. In 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 0515-0522). IEEE.
- [55] Reddy, D. K. K., Behera, H. S., Nayak, J., Naik, B., Ghosh, U., & Sharma, P. K. (2021). Exact greedy algorithm based split finding approach for intrusion detection in fog-enabled IoT environment. *Journal of Information Security and Applications*, 60, 102866.
- [56] Mohamed, D., & Ismael, O. (2023). Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing. *Journal of Cloud Computing*, 12(1), 1-13.
- [57] Tuli, S., Basumatary, N., & Buyya, R. (2019, November). Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments. In 2019 4th International Conference on Information Systems and Computer Networks (ISCON) (pp. 496-502). IEEE.
- [58] Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and communication networks*, 2020, 1-11.
- [59] 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.
- [60] Gudla, S. P. K., Bhoi, S. K., Nayak, S. R., & Verma, A. (2022). DI-ADS: a deep intelligent distributed denial of service attack detection scheme for fog-based IoT applications. *Mathematical Problems in Engineering*, 2022, 1-17.

- [61] Singh, S., Kumari, K., Gupta, S., Dua, A., & Kumar, N. (2020, June). Detecting different attack instances of DDoS vulnerabilities on edge network of fog computing using gaussian Naive bayesian classifier. In 2020 IEEE international conference on communications workshops (ICC Workshops)(pp. 1-6). IEEE.
- [62] Prabavathy, S., Sundarakantham, K., & Shalinie, S. M. (2018). Design of cognitive fog computing for intrusion detection in Internet of Things. *Journal of Communications and Networks*, 20(3), 291-298.
- [63] Kumar, P., Gupta, G. P., & Tripathi, R. (2021). Design of anomaly-based intrusion detection system using fog computing for IoT network. *Automatic Control and Computer Sciences*, 55(2), 137-147.
- [64] Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., & Razaque, A. (2020). Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory*, 101, 102031.
- [65] 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.
- [66] Xu, S., Qian, Y., & Hu, R. Q. (2019). Data-driven network intelligence for anomaly detection. *IEEE Network*, 33(3), 88-95.
- [67] Selim, I. M., & Sadek, R. A. (2023). DAE-BILSTM: A Fog-Based Intrusion Detection Model Using Deep Learning for IoT. *Journal of Theoretical and Applied Information Technology*, 101(5).
- [68] Rashid, M. M., Kamruzzaman, J., Hassan, M. M., Imam, T., & Gordon, S. (2020). Cyberattacks detection in iot-based smart city applications using machine learning techniques. *International Journal of environmental research and public health*, 17(24), 9347.
- [69] Alzahrani, Abdullah IA, et al. "Anomaly Detection in Fog Computing Architectures Using Custom Tab Transformer for Internet of Things." *Electronics* 11.23 (2022): 4017.
- [70] Sharma, D. K., Dhankhar, T., Agrawal, G., Singh, S. K., Gupta, D., Nebhen, J., & Razzak, I. (2021). Anomaly detection framework to prevent DDoS attack in fog empowered IoT networks. *Ad Hoc Networks*, 121, 102603.
- [71] Almaiah, A., & Almomani, O. (2020). An investigation of digital forensics for shamoon attack behaviour in FOG computing and threat intelligence for incident response. *J. Theor. Appl. Inf. Technol*, 15, 98.

- [72] Hameed, S. S., Hassan, W. H., & Latiff, L. A. (2020, December). An efficient fog-based attack detection using ensemble of MOA-WMA for Internet of Medical Things. In *International Conference of Reliable Information and Communication Technology* (pp. 774-785). Cham: Springer International Publishing.
- [73] Alrashdi, I., Alqazzaz, A., Aloufi, E., Alharthi, R., Zohdy, M., & Ming, H. (2019, January). Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0305-0310). IEEE.
- [74] Roy, S., Li, J., & Bai, Y. (2022). A two-layer fog-cloud intrusion detection model for IoT networks. *Internet of Things*, 19, 100557.
- [75] Çavuşoğlu, Ü., Akgun, D., & Hizal, S. (2023). A Novel Cyber Security Model Using Deep Transfer Learning. *Arabian Journal for Science and Engineering*, 1-10.
- [76] Maharaja, R., Iyer, P., & Ye, Z. (2020). A hybrid fog-cloud approach for securing the Internet of Things. *Cluster Computing*, 23, 451-459.
- [77] Gad, A. R., Haggag, M., Nashat, A. A., & Barakat, T. M. (2022). A Distributed Intrusion Detection System using Machine Learning for IoT based on ToN-IoT Dataset. *International Journal of Advanced Computer Science and Applications*, 13(6).
- [78] Kumar, P., Gupta, G. P., & Tripathi, R. (2021). A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *Journal of ambient intelligence and humanized Computing*, 12, 9555-9572.
- [79] Moqurrab, S. A., Tariq, N., Anjum, A., Asheralieva, A., Malik, S. U., Malik, H., ... & Gill, S. S. (2022). A deep learning-based privacy-preserving model for smart healthcare in Internet of medical things using fog computing. *Wireless Personal Communications*, 126(3), 2379-2401.
- [80] Bishnoi, Shubham, Sagarika Mohanty, and Bibhudatta Sahoo. "A deep learning-based methodology in fog environment for DDOS attack detection." *2021 5th International conference on computing methodologies and communication (ICCMC)*. IEEE, 2021.
- [81] Chang, Q., Ma, X., Chen, M., Gao, X., & Dehghani, M. (2021). A deep learning based secured energy management framework within a smart island. *Sustainable Cities and Society*, 70, 102938.

- [82] Priyadarshini, R., & Barik, R. K. (2022). A deep learning based intelligent framework to mitigate DDoS attack in fog environment. *Journal of King Saud University-Computer and Information Sciences*, 34(3), 825-831.
- [83] Onah, J. O., Abdullahi, M., Hassan, I. H., & Al-Ghusham, A. (2021). Genetic Algorithm based feature selection and Naïve Bayes for anomaly detection in fog computing environment. *Machine Learning with Applications*, 6, 100156.
- [84] 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.
- [85] Zakariah, M., & Almazyad, A. S. (2023). Anomaly Detection for IOT Systems Using Active Learning. *Applied Sciences*, 13(21), 12029.]

Vitae

Name	: [Hawazen Jamaan Alzahrani]
Nationality	: [Saudi Arabian]
Date of Birth	: [11/20/1995]
Email	: [alzahranihawazen@gmail.com]
Address	: [Dammam, Eastern Province, Kingdom of Saudi Arabia
Postal Address	EHDC3205, 3205 D 04, 7805, Al-eskan aljanoubi district, Dammam 32321
LinkedIn	https://www.linkedin.com/in/493866188-هوازن-الزهراني/
Education	BA IN INFORMATION TECHNOLOGY King Abdul- Aziz University 4.47 GPA, Very Good-Second Honors. Courses in the specialized track of NETWORK.
Publication	Attack Simulation on Data Distribution Service-based Infrastructure System - IoTBDS 2023 – 8th international conference on internet of things, Bing data and security.]