Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning

Wudu Bitew Alemayew

A Thesis Submitted to the Department of Computer Science and Engineering
College of Electrical Engineering and Computing
Office of Graduate Studies

Presented in Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science and Engineering

Adama Science and Technology University

June, 2025
Adama, Ethiopia

Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning

Wudu Bitew Alemayew
Advisor: ketema Adere (ph.D)

A Thesis Submitted to the Department of Computer Science and Engineering

College of Electrical Engineering and Computing
Office of Graduate Studies

Presented in Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science and Engineering

Adama Science and Technology University

June, 2025
Adama, Ethiopia

DECLARATION

I hereby declare that this Master's Thesis entitled **"Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning"** is my original work and has not been submitted for the award of any academic degree, diploma, or certificate in any university for a similar purpose. The reference and all sources of materials that are used for this thesis have been duly acknowledged through citation.

_____	_____	_____
Name of student	Signature	Date

ADVISOR RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have closely advised the student while developing this thesis and read the draft thesis entitled "**Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning**" prepared under my guidance by Wudu Bitew Alemayew submitted in partial fulfillment of the requirement for the degree of Masters of Science in Computer Science and Engineering. Therefore, I/we recommend the submission of the revised version of the thesis to the department following the applicable procedures.

Ketema Adere Gemeda (PhD)	_____	_____
Main Advisor	Signature	Date

APPROVAL PAGE OF M.SC. THESIS

I/we, the advisors of the thesis entitled "**Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning**" developed by Wudu Bitew Alemayew, hereby certify that the recommendations and suggestions made by the board examiners are appropriately incorporated into the final version of the thesis.

Ketema Adere Gemeda (PhD) _____ _____

Main Advisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by Wudu Bitew Alemayew, have read and evaluated the thesis entitled "**Detection and Mitigation of RPL Protocol Attacks in 6LoWPAN Using Federated Hybrid Deep Learning**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for the partial fulfillment of the requirements of the degree of Master of Science in Computer Science and Engineering.

_____ _____ _____

Chairperson Signature Date

_____ _____ _____

Internal Examiner Signature Date

_____ _____ _____

External Examiner Signature Date

Final approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and the College Graduate Committee.

_____ _____ _____

Department Head Signature Date

_____ _____ _____

College Dean Signature Date

_____ _____ _____

Office of Postgraduate Studies Dean Signature Date

ACKNOWLEDGMENT

First and foremost, I extend my deepest and heartfelt gratitude to Almighty God. It is by His endless grace, wisdom, and strength that I have been able to reach this milestone. Through the challenges, long nights, and moments of doubt, His presence has been my anchor and my source of perseverance. This work would not have been possible without His divine guidance.

I am profoundly thankful to my advisor, Dr. Ketema Adere, whose insightful guidance and unwavering support have shaped this thesis in more ways than I can express. His thoughtful feedback, patience, and encouragement provided both academic direction and personal motivation. I am especially grateful for his belief in my potential and his dedication to helping me grow as a researcher.

I would also like to express my sincere appreciation to my family and friends. Their constant prayers, moral support, and belief in me have been a great source of strength throughout this journey. To my family, thank you for being my foundation and for encouraging me even when the path was uncertain. Your love and patience have meant everything. This thesis is not just the result of my effort but a reflection of the support, faith, and contributions of many. To all who have been a part of this journey, directly or indirectly, thank you.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
BH	Blackhole
CNN	Convolutional Neural Network
DAO	DODAG Advertises Object
DBN	Deep Belief Network
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DT	Decision Tree
DODAGs	Destination-Oriented Directed Acyclic Graphs
FedAvg	Federated Average
FL	Federated Learning
GRU	Gated Recurrent Unit
HF	Hello Flooding
IDS	Intrusion Detection System
IETF	International Engineering Task Force
IoT	Internet of Things
IRAD	Internet Routing Attacks Dataset
LLNs	Low Power and Lossy Networks
LR	Linear Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-layer Perceptron
PCA	Principal Component Analysis
QoS	Quality of Service
ReLU	Rectified Linear Unit
RF	Random Forest
RPL	Routing Protocol for Low-Power and Lossy Networks
SVM	Support Vector Machine
VN	Version Number
WSN	Wireless Sensor Network
6LoWPAN	IPv6 over Low-Power and Lossy Network

ABSTRACTS

The adoption of 6LoWPAN in IoT networks has made the RPL a standard. Despite its widespread adoption, RPL is highly susceptible to routing attacks that threaten communication reliability, energy efficiency, and data integrity. Existing detection methods to detect RPL attacks often rely on centralized learning which raises privacy concerns, and typically use datasets limited to small or medium-size networks., limiting real-world applicability and generalization. Existing literature did not use more than one model at a time to handle the correlation relationship between RPL network traffic and the change of this traffic over time. To address this gap, this thesis proposed a federated deep learning-based framework for the detection and classification of three impactful RPL attacks: Blackhole (BH), Hello flooding (HF), and Version number (VN) attacks. We have collected a multiclass classification dataset from IRAD, covering four network sizes including 10 nodes, 20 nodes, 100 nodes, and 1000 nodes. This dataset is used to ensure realistic and generalizable evaluation. Preprocessing involved outlier removal, deduplication, and a hybrid feature selection approach combining Random Forest and XGBoost to enhance model performance were used. The round-robin assignment technique was used to assign labels equally to all federated clients. A novel hybrid CNN-GRU model was developed to improve the detection rate. It was evaluated against CNN-LSTM, LSTM, and GRU baselines under a federated learning setup, where data remained decentralized across clients. The proposed CNN-GRU model achieved 99.50% accuracy, demonstrating superior detection performance and computational efficiency. A conceptual mitigation strategy is integrated into the study. Upon detection, specific actions are triggered based on the attack type. For BH attacks, malicious nodes are blacklisted by the root node and excluded from routing by all nodes. For VN attacks, the root node adds the attacker to a watchlist and instructs nodes to ignore its updates. For HF, affected nodes rate-limit or block messages from the attacker.

***Keywords:* Federating hybrid deep learning, attack on RPL, detection, classification, conceptual mitigation, IRAD**

CHAPTER ONE

1. Introduction

1.1. Background of the Study

The Internet of Things (IoT) refers to a network of connected devices, machines, and associated software services. It is highly significant today, as it facilitates energy-efficient automation, improving quality of life(Yavuz et al., 2018). Over the past two centuries, human development has been rapidly accelerating due to the adoption of various technologies. One of the most impactful advancements has been the exponential growth in computing power. Additionally, this connectivity is extending further as more devices are integrated into urban environments to create smarter systems. IoT has opened up new possibilities in smart homes, healthcare, agriculture, and industrial automation. These IoT environments rely heavily on the seamless interconnection of low-power, resource-constrained devices such as sensors and actuators. However, the traditional Internet Protocol (IP) stack was not designed with these limitations in mind. This has led to the development of specialized protocols and architectures tailored for such environments, one of the most notable being 6LoWPAN. These systems can potentially reduce the need for manual labor and make life more efficient and intelligent (Jony & Arnob, 2024).

IoT is expanding rapidly, allowing things or devices to exchange and share information in IPv6 over low-power wireless personal area networks (6LoWPAN) (Alazab et al., 2023). It is among the major technologies to connect devices in the IoT space, and it is an innovative breakthrough, providing an energy-efficient and low-cost networking technology for IoT devices. Its characteristic of allowing equilibrium between performance and resources makes it indispensable for the popularization of IoT systems. IoT devices are generally small, low-power, and resource-limited devices with limited battery life and computational resources. Therefore, it is not possible to directly implement ordinary IPv6 protocols in IoT devices. The improvement to the IPv6 protocol, permitting its usage in IoT, is called 6LoWPAN. 6LoWPAN refers to Low low-power wireless Personal Area Network. 6LoWPAN combines IPv6 with the IEEE 802.15.4 standard, which has been reflected in Wired and Wireless networks, which comes in handy when ANTs need to work over legal frequency bands. In the network of the

6LoWPAN, data transfer depends largely on the energy conservation routing protocol. It is a key enabling technology that allows small, battery-powered devices to connect to the Internet using IPv6. By providing compression, fragmentation, and adaptation mechanisms, 6LoWPAN makes it possible to run the IPv6 protocol over IEEE 802.15.4 networks, which have limited packet sizes and energy resources. Its lightweight nature, low power consumption, and compatibility with IPv6 have made 6LoWPAN widely adopted in IoT systems.

To facilitate communication among resource-limited nodes, numerous protocols have been proposed, and RPL is a popular and promising routing protocol. The task force of IETF designed the protocol in the year 2012 (RFC 6550), and it solves the routing difficulties faced in IoT networks by utilizing destination-oriented directed acyclic graphs (DODAGs), thereby facilitating communication in low-power and lossy networks (LLNs) optimally (Al-Amiedy et al., 2022). The DODAG formation is a critical part of the RPL that enables efficient and effective communication in 6LoWPAN IoT networks. Each node in the DODAG selects its parent based on certain routing metrics such as link quality or energy efficiency, which decides the optimum communication paths. Data can be efficiently propagated from any node towards the root with less delay and energy consumption. Importantly, DODAG avoids looping in communication and there is no loop in the network (Al-Amiedy et al., 2022).

Despite its advantages, RPL suffers from a number of security vulnerabilities due to its lightweight design and the openness of wireless communication. These vulnerabilities have given rise to various routing attacks such as Black Hole attacks, Hello Flooding, and Version Number Attacks, each capable of disrupting network performance, causing data loss, increasing energy consumption, or even bringing down entire IoT systems. These attacks exploit the very mechanisms that make RPL efficient, turning its strengths into weaknesses.

Given the increasing reliance on IoT devices in critical sectors and the vulnerability of their underlying protocols, securing RPL-based 6LoWPAN networks is no longer optional, it is essential. This is the primary motivation behind focusing on this area. While many existing studies have proposed centralized or rule-based security mechanisms, such solutions often fall short in distributed and dynamic IoT environments. They may not scale well, fail to adapt to changing attack patterns, or introduce unacceptable delays and energy costs (Wang et al., 2024). These attacks seriously threaten the security and stability of IoT networks, highlighting the

urgent need to develop more advanced and robust defense mechanisms to protect against such vulnerabilities (Bokka & Sadasivam, 2023). The RPL protocol, which serves as the backbone for routing in 6LoWPAN-based IoT networks, is vulnerable to several routing attacks. Among these, the most frequent and significant threats include version number attacks, rank attacks, black hole attacks, and flooding attacks (Alsubai, S., Alqahtani, A., Garg, H., Sha, M., & Gumaei, 2024). These attacks undermine IoT networks' stability, efficiency, and security by exploiting weaknesses in the routing protocol. In this thesis, our primary goal is to develop robust attack detection and classification on RPL protocol.

Numerous studies (Alazab et al., 2023; Cakir et al., 2020; Shahid et al., 2024; Yavuz et al., 2018) have attempted to address the challenge of securing RPL networks by employing rule-based methods, as well as machine learning (ML) and deep learning (DL) models for attack detection. While these approaches have shown some promise, they suffer from notable limitations. Binary classification, privacy issues during training due to all the data shared with the central server, and lack of diversified dataset which limits the model generalizability are among the key limitations that remain unresolved. These shortcomings prevent existing solutions from being fully effective in real-world, large-scale IoT deployments.

To overcome these limitations, a federated learning framework has been proposed. FL preserves the privacy of individual IoT devices, addressing the privacy concerns often associated with centralized models (Tariq et al., 2024), (Kowsalyadevi & Balaji, 2024). FL has been introduced as a privacy-preserving mechanism, suggesting that the machine learning model parameters, rather than data, are sent over the network to a central point of aggregation. However, when relaxing the privacy concerns, the debate strongly relates to the available network resources. Interestingly, the so far theoretical or even experimental comparison of the two approaches overlooks network conditions and remains of low realism (Drainakis et al., 2020)

In recent years, hybrid deep learning models integrated within federated learning frameworks have gained attention for their ability to preserve data privacy while improving the detection of attacks on RPL in IoT networks.

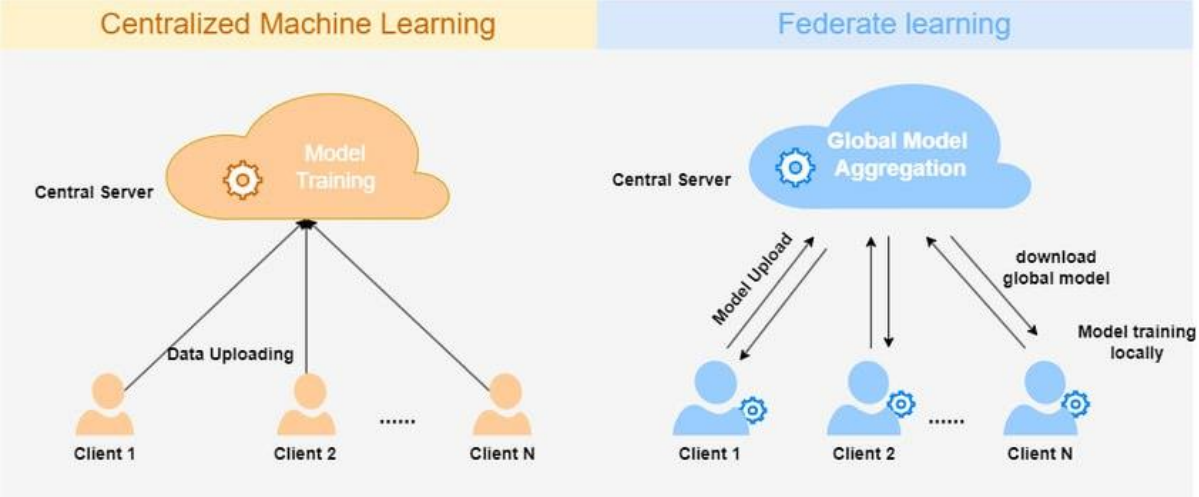

Figure 1- 1:Architecture of centralized machine learning and federated learning (Mawela et al., 2025)

In centralized learning, as illustrated in the above Figure **1-1**, data is sent to the cloud, where the ML model is built. In federated learning, each device trains a model and sends its parameters to the server for aggregation. Data is kept on devices, and knowledge is shared through an aggregated model.

1.2. **Motivation of the Study**

As the world becomes increasingly connected through IoT, ensuring the reliability, efficiency, and security of these networks has become a top priority. Devices such as sensors, smart meters, and actuators are now being deployed in critical areas, including healthcare, agriculture, industry, and smart cities. Most of these devices operate under severe resource constraints, limited power, processing capability, and memory. To enable efficient communication among them, 6LoWPAN has emerged as a practical solution, bridging the gap between constrained IoT networks and the wider Internet using IPv6 (Shahid et al., 2024). At the heart of 6LoWPAN communication lies the RPL. RPL is designed specifically for these kinds of environments, providing a routing framework that is energy-efficient and well-suited to frequent changes in network topology. However, this efficiency comes at a cost. RPL's design choices, such as its reliance on trust among nodes and minimal security overhead, make it vulnerable to a range of attacks, including Black hole, Hello flooding, and Version number attacks. These attacks are not theoretical, they have real-world consequences. A single malicious node can mislead others, disrupt data delivery, consume valuable energy resources, or isolate entire parts of the network.

In sectors where timing and accuracy are critical, like healthcare or industrial automation, such disruptions can lead to significant losses or even life-threatening situations. This growing risk is what sparked the motivation for this research.

Despite the increasing awareness of these threats, most existing security solutions for RPL are centralized, rule-based, or too heavyweight for low-power IoT devices. They either require continuous updates, consume too much energy, or lack adaptability to new or evolving attacks. Furthermore, traditional machine learning approaches that rely on centralizing all the data raise privacy concerns and are not ideal for distributed networks. This thesis is motivated by the need to design a solution that addresses three key challenges:

- Detection accuracy: Recognizing and classifying various types of RPL attacks effectively.
- Resource efficiency: Ensuring the solution works within the limited computational and energy capacities of IoT devices.
- Privacy and scalability: Enabling collaborative learning across devices without exposing raw data.

To meet these needs, this research proposes a federated hybrid deep learning approach, combining the strengths of deep learning for pattern recognition with the federated learning paradigm, which allows model training across distributed nodes without sharing sensitive data. By focusing specifically on RPL attacks in 6LoWPAN networks, the aim is to contribute a scalable, intelligent, and privacy-preserving security framework suitable for the IoT systems of today and tomorrow. FL method keeps the privacy of data, reduces latency and communication overhead (Rey et al., 2023), and leads to RPL becoming tougher against powerful attacks. FL approach preserves data privacy, reduces latency, and minimizes communication overhead (Rey et al., 2023), improving RPL's resilience to sophisticated attacks. Despite the federated learning framework's challenges, it has several advantages in securing the privacy of data, which is one step toward improving RPL security (Shanmugarasa et al., 2023).

1.3. **Statement of the Problem**

RPL is commonly utilized in IoT systems that use 6LoWPAN because of its effectiveness in handling constrained network environments. Nevertheless, RPL is still susceptible to a variety of routing attacks, which present some of the most significant security challenges for IoT

networks based on RPL. These attacks aim at undermining the essential routing functions that ensure the proper and efficient transmission of data packets throughout the network. While research has been carried out on identifying RPL attacks, there are still several substantial gaps. The majority of methods for detecting RPL attacks depend on a centralized model where the training is performed on a dataset gathered from IoT devices stored on a central server (Cakir et al., 2020; Morales-Molina et al., 2021; Shahid et al., 2024). In practical IoT deployments, information is typically distributed across a wide range of devices, rather than being collected in one central repository. Attempting to gather all of this sensitive data for centralized model training raises major privacy concerns and increases the likelihood of data leaks or unauthorized access to data breaches at the center. Additionally, transferring large volumes of data to cloud platforms for machine learning or deep learning purposes introduces significant inefficiencies and costs, especially due to the burden on both network transmission and storage resources. Although centralized learning approaches can be effective under certain controlled conditions, they often encounter serious limitations when applied to real-time IoT environments. Communication delays between devices and a central server can make it difficult to meet the low-latency requirements of such environments. As more devices are added, the burden on network bandwidth increases and the system's ability to scale becomes a major concern. Upgrading infrastructure to handle this growth can be costly and complex. These challenges point to the need for a more suitable alternative, such as federated learning, a decentralized method that enables IoT devices to collaboratively train a shared model without exchanging raw data (Al-Amiedy et al., 2022), keeping users' privacy (Shah et al., 2023).

Existing RPL attack detection techniques did not use hybrid approaches that could significantly optimize detection accuracy (Albishari et al., 2023; Cakir et al., 2020; Choukri et al., 2020; Morales-Molina et al., 2021; Yavuz et al., 2018)(R. Khan et al., 2024)(R. Khan et al., 2024). IoT network traffic consists of both time-series and spatial data, necessitating a model that can effectively handle and learn from both types of information (Ashraf Zargar, 2021; Hadipuspito & Suryani, 2024). Handling spatial features and temporal dependency using different deep-learning models can significantly improve accuracy. To fully exploit these data characteristics, a hybrid deep learning model is required that combines techniques like convolutional neural networks (CNNs) for spatial feature extraction between sample datasets(Abd-El-Atty, B., ElAffendi, M., & El-Latif, 2023) and long short-term memory (LSTM) or gated recurrent units

(GRU) (Cakir et al., 2020) for analyzing temporal patterns, enhancing the detection(N. W. Khan et al., 2023).

The lack of effective mitigating strategies is a critical gap in the detection and classification of RPL attacks, despite the numerous ML and DL-based techniques that have been offered. Most existing studies focus solely on identifying attack behaviors but do not propose concrete solutions to mitigate such attacks once detected. This results in a security gap in which detection is insufficient to defend the network against persistent or recurring attacks. Systems might therefore be susceptible even after an attack has been detected. Thus, even if mitigation techniques are just conceptually proposed, their inclusion alongside detection-based solutions is essential to improve the overall resilience of IoT networks and provide the base foundation for future mitigation work.

Moreover, most existing RPL attack detection systems tend to have a narrow focus, typically addressing only one or two specific routing threats. For instance, (Cakir et al., 2020) introduced a GRU-based model specifically aimed at detecting Hello Flooding attacks, which exploit limited device resources. Similarly, (R. Khan et al., 2024) proposed a federated learning-based method designed solely to identify selective forwarding attacks. Another approach employed an MLP neural network for intrusion detection, targeting only rank attacks to distinguish between normal and malicious traffic. In another effort, a robust AI-driven framework was developed to detect identity impersonation attacks, an area where traditional systems often fall short. This method used unsupervised learning to extract key features from RPL network data, followed by deep neural network (DNN) training to enhance classification accuracy. However, this solution was tailored specifically to identify Clone ID attacks. These narrowly scoped systems leave IoT networks exposed to a wider spectrum of threats. To achieve more reliable protection in RPL-based networks, there is an urgent need for comprehensive detection mechanisms capable of identifying impactful attacks that can lead to severe packet loss, disrupted topology, energy exhaustion, and control message overload, ultimately compromising network stability. Addressing a broader range of threats, including Version Number attacks, Decreased Rank attacks, Black Hole attacks, and various flooding attacks, is essential to maintain the resilience and integrity of the IoT ecosystem.

Another limitation in prior research lies in the datasets used. Many studies rely on data generated from small-scale or simplified networks, which fail to capture the dynamic, heterogeneous characteristics of real-world IoT environments. Small, medium, and large-scale networks are frequently seen in real-world settings, and attack patterns might change greatly between these dimensions. Existing RPL attack detection studies focused on using insufficient and self-generated datasets lacking comprehensive labeling, transparency, and well-organized datasets that often did not provide detailed information about key aspects such as the number of features, selected features, or dataset size. Most of the prior work in detecting RPL attacks has relied on datasets produced from networks of 10 up to 50 nodes, which are not representative of real-world IoT environments where a network can comprise hundreds or even thousands of nodes. This reduced scalability and the generalization capability of previous models make them less effective in a large-scale deployment (N. W. Khan et al., 2023)(Choukri et al., 2020)(Momand et al., 2021). The Internet Routing Attack Dataset (IRAD), which offers attack data across a range of network sizes: Specifically, 10, 20, 100, and 1000 nodes, is used in this study to solve the issue raised above. The study guarantees a more realistic, generalizable, and scalable evaluation of the suggested detection and mitigation architecture by balancing classes across these network sizes and incorporating multiple RPL attack types, such as black hole (BH), hello flooding (HF), and Version number attack (VN) attacks.

1.4. **Research Questions**

Below are questions to answer the above problem mentioned.

RQ1: How do we select important features from IRAD and apply preprocessing techniques to improve federated RPL attack detection and classification?

RQ2: How can a federated hybrid deep learning model be developed to detect and classify RPL attacks?

RQ3: How does the performance of the proposed federated hybrid deep learning architecture compare to baseline models and those presented in the base paper?

RQ4: Using a conceptual framework, how can detected RPL attacks be mitigated within an RPL-based network?

1.5. Objective

1.5.1. General Objective

The general objective is to detect and mitigate RPL protocol attacks in 6LoWPAN using federated hybrid deep learning.

1.5.2. Specific Objectives

- To employ feature selection and preprocessing techniques on the IRAD.
- To propose an architecture for a federated hybrid deep learning.
- To implement the federated hybrid deep learning.
- To evaluate the performance of the proposed federated hybrid deep learning model against baseline models and the methods presented in the base paper.
- To design a conceptual mitigation framework for detected RPL attacks.

1.6. Significance of the Study

In today's ever-growing digital world, IoT has given us wonderful convenience, but it introduces security challenges, especially in low-power and lossy networks 6LoWPAN devices. This thesis addresses those concerns by tackling a critical yet underexplored issue like the vulnerability of RPL protocol to malicious attacks. To improve the security of IoT networks, this paper develops a new federated hybrid deep learning model that can identify and categorize three types of attacks including HF, VN, and BH attacks. The proposed model uses GRU networks for forecasting normal traffic and CNNs for attack pattern recognition in a federated learning environment. Thus, sensitive data does not leave local devices, privacy is protected at the training time. The research progresses beyond detection, proposing conceptual mitigation mechanisms that mitigate once the attack is detected.

1.7. Contribution of the Study

This study makes several significant contributions to RPL attack detection and mitigation in 6LoWPAN-based IoT networks using a federated hybrid deep learning approach. Firstly, a generalized dataset was sampled from four network sizes including 10 nodes, 20 nodes,100 nodes, and 1000 nodes to closely mimic real-world IoT scenarios. The dataset was carefully preprocessed to prepare it for sequential modeling. To reflect the temporal changes of features

pertinent to attack detection, a sequential dataset with a length of 5 was created. Furthermore, a combined feature selection approach was utilized, merging Pearson correlation analysis with Random Forest and XGBoost techniques to pinpoint the most pertinent features, further strengthening the model's robustness and detection capabilities.

Secondly, the round-robin assignment technique was employed to distribute data labels among federated clients, improving detection accuracy. This ensured equal class representation across clients, leading to greater model training stability, faster convergence, and better overall performance in the federated setting. A federated learning framework ensured privacy while keeping accuracy.

Thirdly, we proposed parallel functional API and sequential API model architectures. The proposed models were tested using two Keras structures in a federated learning setup. We demonstrated that the sequential API outperforms the functional API. Four models were implemented using a sequential API structure in a federated learning environment. Among them, the federated CNN-GRU model achieved higher accuracy than the others while maintaining privacy. Upon the detection, the conceptual mitigation strategy was applied to mitigate the identified attacks.

1.8. Scope and Limitations of the Study

This study is centered on the development of a federated learning-based framework for detecting and classifying critical routing attacks that target the RPL protocol within 6LoWPAN-based IoT environments. The focus is specifically placed on three major and impactful types of RPL attacks: Black Hole attacks, which interrupt traffic flow by maliciously dropping packets; Hello Flooding attacks, which overwhelm nodes by faking neighbor relationships; and Version Number attacks, which manipulate routing updates to distort the network topology (Alfriehat et al., 2024). These attacks were selected because they represent distinct and serious threat categories, namely traffic disruption, resource exhaustion, and topological instability, which collectively reflect the range of vulnerabilities that RPL-based systems commonly face.

The proposed framework leverages a hybrid deep learning model, combining GRU for forecasting and CNN for classification, within a federated learning setup. The federated approach allows multiple clients to collaboratively train a shared model without transferring

raw data, thus preserving data privacy and reducing communication overhead. The Federated Averaging (FedAvg) algorithm is used to aggregate the locally trained models, ensuring efficient convergence across clients. The system is trained and tested using the multiclass IRAD dataset, which contains labeled traffic data for four different network scales: 10, 20, 100, and 1000 nodes. Additionally, feature selection techniques and preprocessing steps are applied to enhance model performance and reduce computational complexity.

Despite these contributions, the study is subject to several limitations. First, the scope is restricted to the detection of only three RPL attacks, and does not extend to the full range of known or emerging attacks within low-power IoT networks. Second, although the framework proposes a strategy for mitigation, this part of the work remains conceptual; it has not been implemented or validated in a real-world deployment or full-scale simulation environment. Third, the federated learning setup assumes that data is independently and identically distributed (IID) across all participating clients. This assumption simplifies model training but may not fully reflect the non-IID nature of data in practical IoT systems, where device behavior and data generation can vary significantly. Finally, the entire system is developed and evaluated in an offline, controlled environment, and has not yet been tested under the constraints of real-time operations or limited hardware typically found in physical IoT devices.

By clearly defining what the research addresses and what it does not, this thesis aims to provide a realistic and transparent view of its scope, while laying the foundation for future work that can expand the attack coverage, move toward real-world implementation, and explore adaptive strategies under more complex network conditions.

1.9. **Organization of the Thesis**

This thesis is structured into seven chapters, each contributing to a clear progression from identifying the research problem to presenting the conclusions and future research directions.

- Chapter One introduces the research problem and outlines the motivation behind the study, highlighting the need for secure communication RPL protocol. It presents the key research gaps, defines the objectives, and summarizes the main contributions of the work.
- Chapter Two reviews the existing literature and related works. It explores prior efforts in RPL attack detection and federated learning, identifying limitations in current approaches.

- Chapter Three explains the research methodology, including the rationale behind the chosen approach, the dataset used, and the tools and techniques applied.
- Chapter Four details the proposed federated hybrid deep learning architecture. It describes the model design, combining different models within a federated learning framework
- Chapter Five focuses on the implementation process. It discusses how the architecture was realized using appropriate tools and frameworks and provides insights into training procedures, model integration, and the federated setup.
- Chapter Six presents the experimental results and offers a critical discussion. Performance metrics are analyzed, and the effectiveness of the proposed system in detecting multiple RPL attacks is evaluated and interpreted.
- Finally, Chapter Seven concludes the thesis by summarizing the key findings. It depicts potential directions for future research.

CHAPTER TWO

2. Literature Review and Related Work

2.1. Literature Review

2.1.1. Overview of IoT Network

IoT is the connection of any device that anyone can access from any location, at any moment, via any application over any network(Ray, 2018). IoT has driven numerous innovations, discoveries, and interactions between devices and people, improving the quality of life and optimizing limited resources.

2.1.2. IoT Stack Architecture

The IoT architecture is structured into five layers: physical, data link layer, network layer, transport layer, and application layer. These layers are depicted in Figure 2-1. The physical layer is known as the perception layer in the context of IoT. The main role of this layer is to detect the physical properties of the neighboring material. It is dependent on different sensing technologies like radio frequency identification, wireless sensor networks, and global positioning systems. It also converts data to digital signals, allowing for more efficient transmission across a network. The physical layer heavily relies on nanotechnology and embedded intelligence for its operation (Abdmeziem et al., 2016). Some of the major functions of the data link layer are defining the boundary of a packet, providing frame synchronization, handling the sender and receiver addresses, error detection into the physical communication media, and avoiding or resolving the collision of data (Oliveira et al., 2019). Each protocol is designed for a particular application and has specific characteristics regarding media access control, transfer speed, topology, transmission range, and energy efficiency. The network layer refers to data as packets that are sent across the network routers (U.Farooq et al., 2015). It encompasses all network devices, including switches and routers, required for technologies like 3G, 4G, 5G, Wi-Fi, infrared, 6LoWPAN, and ZigBee, and their communication and routing protocols. The transport layer works with the application layer to ensure error-free data transmission, providing features like packet order delivery, congestion control, multiplexing, byte orientation, data integrity, and reliability(Vashi et al., 2017).

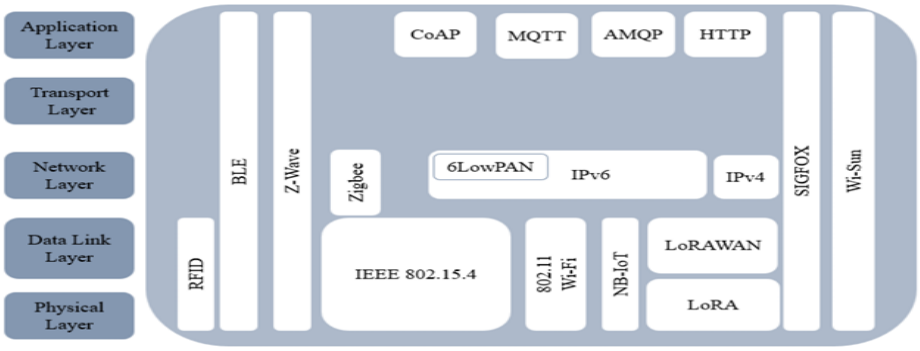

Figure 2- 1. IoT stack architecture (Oliveira et al., 2019)

This chapter briefly gives an overview of the 6LoWPAN network and critical and frequent attacks in routing protocols, such as RPL attacks.

2.1.3. **IoT Communication Technologies**

In certain IoT applications, technological options are limited by hardware capabilities, the necessity for low power consumption, and overall device costs are mandatory. Low power usage is vital for IoT development, but other factors must also be considered, including technology costs, security, ease of use, management, and wireless data rates and ranges. Emerging wireless technologies like ZigBee and Bluetooth offer low-power communication options for IoT, while newer protocols like IEEE 802.11ah, LoRa, and 6LoWPAN are also gaining popularity for their specialized capabilities in IoT connectivity. 6LoWPAN excels in IPv6 compatibility and internet integration introduces and compares IoT communication protocol based on power consumption, data rate, and coverage distance (Mansour et al., 2023).

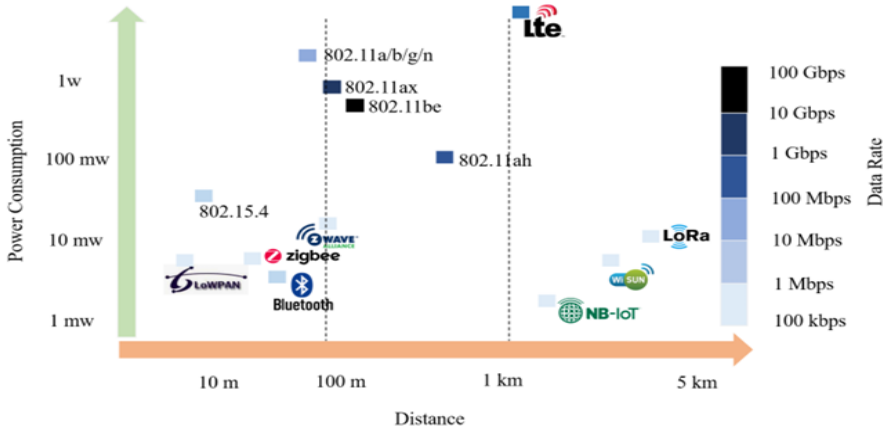

Figure 2- 2:Power consumption, coverage distance, and data rate for the different protocols.

2.1.3.1. **6LoWPAN Communication Technology**

In 2007, the IETF working group developed the 6LoWPAN protocol (Yang & Chang, 2019). The group has developed encapsulation and header compression techniques that shrink IPv6 headers from 40 to 7 bytes for transmission over IEEE 802.15.4 networks, which operate on a mesh architecture with self-organizing, time-synchronized, and self-healing properties. IEEE 802.15.4 is a standard with low-power and low-data-rate wireless communication. It is commonly used in wireless personal area networks and is the foundational protocol for several IoT and mesh networking technologies like ZigBee and 6LoWPAN. 6LoWPAN combines IPv6 with LowPAN networks, enabling low-power devices to communicate wirelessly over short distances (Alkama & Bouallouche-Medjkoune, 2021).

To efficiently transmit IPv6 packets over IEEE 802.15.4 networks, an adaptation layer with header compression is required. Additionally, since traditional IPv6 routing methods are not suitable for the constrained environments in which 6LoWPAN operates, a specialized routing protocol, known as RPL, was specifically developed for 6LoWPAN(Yang & Chang, 2019).

2.1.3.2. **RPL Protocol**

The RPL is especially for the 6LoWPAN protocol (Mansour et al., 2023), the most famous and specific routing protocol for low-power PANs due to the flexibility, energy-efficient routing, and support for the Quality of Service (QoS) (Musaddiq et al., 2020) for wireless networks that are low-power and that are typically prone to packet loss. It is a distance vector-based proactive protocol that runs on IEEE 802.15. It is optimized for multi-hop and many-to-one communication, it however also supports one-to-one messages. The RPL protocol runs on 6LoWPAN, which connects to the IP network through a sink node as depicted in Figure **2- 3** (Gaddour & Koubâa, 2012). RPL has been designed to cater to a range of network environments, including the ones used for home and building automation, industrial systems, urban routing, and smart grids (Miorandi et al., 2012; Xu et al., 2014).

The RPL protocol is a distance-vector proactive routing protocol that forms a hierarchical graph called DODAG through which data can be routed efficiently among the devices, with one or multiple root or sink nodes. Directed acyclic graphs are formed based on a specific objective function (OF) selected by the user, through which data is identified with the most optimal path between sensor devices (Musaddiq et al., 2020). The IETF working group standardized two

default objective functions: objective function zero (OF0) and minimum rank with hysteresis objective function (MRHOF). OF0 selects the shortest path to the sink node as it selects the lowest rank parent node, as per distance in the routing tree. MRHOF decreases link costs by selecting routes that are less expensive than the current path and uses a threshold value for decision-making. RPL uses specific ICMPv6 control messages to communicate DODAG data as well as build the network topology (Cakir et al., 2020). DODAG Information Object (DIO) is a multicast message distributed by the sink node towards all nodes within the network, where nodes are able to select their preferred parent as well as identify the RPL instance. DODAG advertisement object (DAO) message transmits the destination address data to establish a route in the upward direction (Cakir et al., 2020; Choukri et al., 2020). DODAG information solicitation (DIS) message is transmitted by an incoming node to request neighboring nodes for the DIO packets (Cakir et al., 2020; Choukri et al., 2020). Destination advertisement object acknowledgment (DAO-ACK) is sent by nodes to confirm the reception of a DAO message (Yavuz et al., 2018)

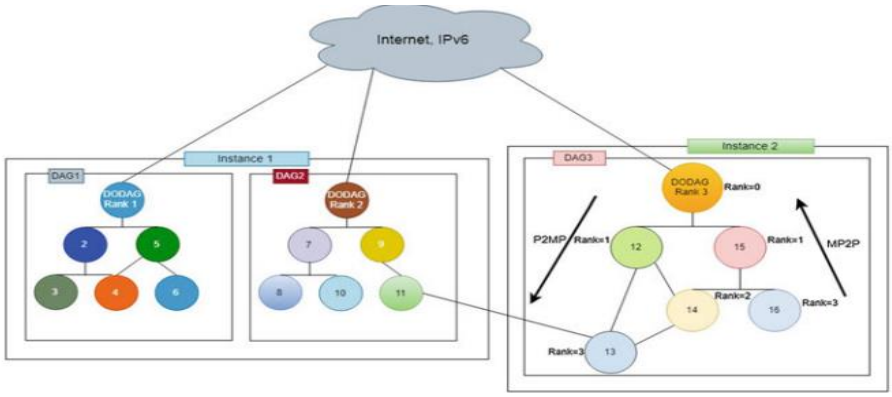

Figure 2- 3 RPL network with three DODAGs in two instances (Cakir et al., 2020).

The 6LoWPAN is chosen for our research area because it efficiently supports IP-based connectivity, enabling seamless integration with existing internet infrastructure while maintaining low power consumption. However, the resource-constrained nature of the 6LoWPAN device's limitation in battery power, processing capacity, and memory can impact the performance and security of the RPL protocol. The RPL control messages introduce vulnerabilities, making them susceptible to rank attacks, black hole attacks, flooding attacks,

and version number attacks. These security weaknesses make 6LoWPAN a critical focus for research on prediction and detection strategies, which is the core of my proposal.

2.1.4. **Routing Attacks Targeting RPL Protocol**

The study (Agiollo et al., 2021) highlighted various routing attacks targeting RPL, such as rank attacks and version attacks, black hole attacks, flooding attacks, etc. Due to the diverse range of attacks that can target the RPL protocol, ensuring secure deployment is challenging. Cyber attackers targeting the RPL protocol exploit ICMPv6 control messages to disrupt. Attacks are classified based on control messages (Bang et al., 2023).

Black Hole Attack: A black hole attack occurs when a malicious node advertises itself as having an optimal route to the destination, causing the neighbor node to forward their packet through it. However, instead of forwarding the data, the malicious node drops all or most of the packets, leading to packet loss, disrupting communication within the network, and affecting network reliability. IoT networks are particularly vulnerable to this attack, and existing IDS often face challenges in effectively detecting and mitigating it due to limitations in scalability and adaptability.

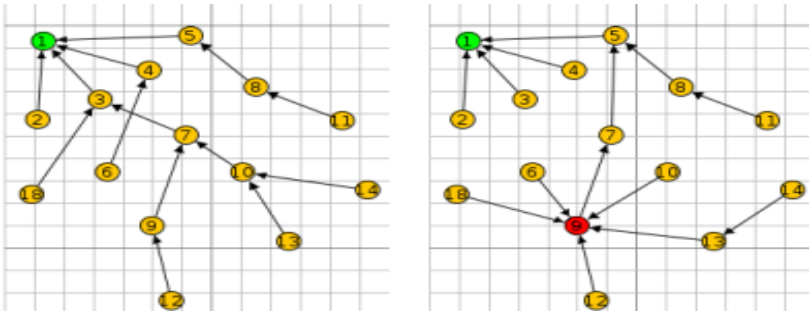

Figure 2- 4: Black hole attack in RPL network (Bang et al., 2023)

Figure 5(a) shows a normal IoT network with 15 nodes forming a functional DODAG. In contrast, Figure 5(b) depicts a black hole attack, where node 9 disrupts RPL routing by positioning itself as a preferred parent, causing neighboring nodes to route their data through it, resulting in data loss. (Reshi et al., 2024) used NS2 and Simulink simulations to create a black hole defense algorithm, achieving a packet delivery ratio of 98.21%, nearly matching unaffected network performance. However, it didn't account for varying network topologies and traffic patterns, limiting its practical application.

Dis Flooding Attack: This attack happens when one or more malicious nodes periodically send DIS messages to nearby nodes within their transmission range, causing nodes in the network to respond with a DIO message. This constant message creates excessive control traffic, increasing congestion and draining the battery life of the nearby nodes. Leading to increased control traffic, which wastes network resources and depletes nodes' energy, especially in energy-constrained 6LoWPAN devices (Al-Amiedy et al., 2022).

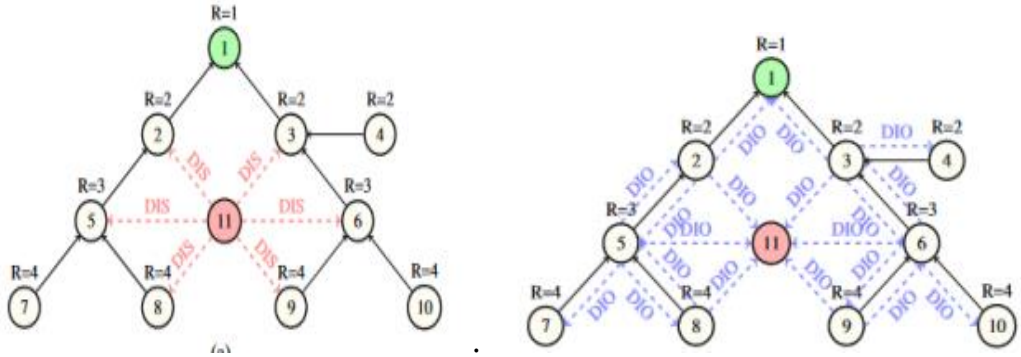

Figure 2- 5 Hello flooding attack manipulation (Al-Amiedy et al., 2022)

In Figure 6(a), the attacker, node 11, broadcasts multicast DIS messages to nearby nodes. Figure 6(b) shows how the attack increases network overhead and power consumption as nodes 2, 3, 5, 6, 8, and 9 receive these messages, prompting them to reset their trickle timers and send out multicast DIO (DODAG Information Object) messages.

DODAG Version Number Attack**:** In RPL, the version number is managed by the root node, and it is only incremented if a global repair of the DODAG is needed. When this happens, the root node broadcasts a DIO message with the updated version number, prompting all nodes in the network to rebuild the DODAG based on the new version. However, in a version number attack, an attacker manipulates the version number unnecessarily, causing frequent and needless reconstructions of the DODAG(Sen, 2021). This disrupts network stability, drains energy, and increases overhead due to repeated reconfiguration processes, impacting both packet delivery and network efficiency.

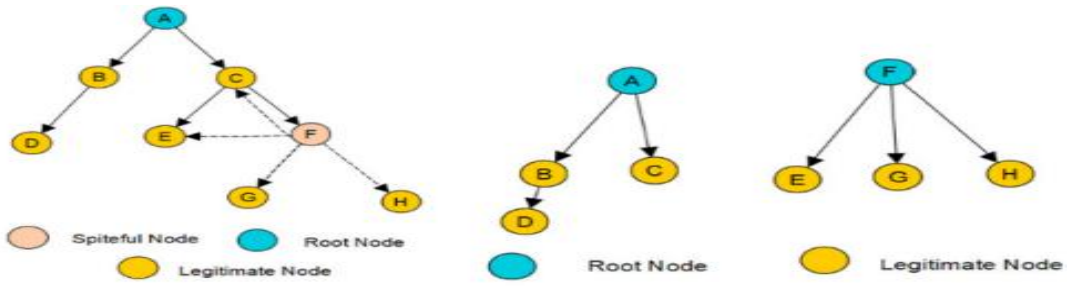

Figure 2- 6: VN attack leads to network instability(Sen, 2021).

Initially, node F joins the DODAG legitimately but later acts maliciously by modifying the version number in the DIO messages (shown by dotted arrows) and sending them to neighboring nodes (C, E, H, and G). While node C rejects the fraudulent message, nodes E, H, and G accept it, updating their version numbers and creating a new DODAG, as depicted in Figure 7(b). In this new configuration, node F becomes the root node, demonstrating the significant consequences of the version number attack.

2.1.5. **RPL Attack Detection Techniques**

Traditional Security Mechanism: Security approaches, including those utilizing cryptography and signature-based detection, struggle to efficiently identify or mitigate RPL attacks because of variations in network topology, increased complexity, and the nature of traffic patterns (Almusaylim et al., 2020).

ML and DL Techniques**:** Big data provides enormous amounts of information that demand sophisticated techniques to derive critical insights, particularly for assessing legitimate and malicious packet behaviors. It involves obtaining essential information from the massive traffic flows within the IoT network. Recent advancements in big data technologies have made ML or DL increasingly feasible to identify distinct behavioral patterns, such as both normal and harmful, from the vast streams of data (Al-Amiedy et al., 2022). ML and DL-based intrusion detection have attracted immense attention towards improving IoT security(da Costa et al., 2019). In IoT, DL is more effective than ML in processing the voluminous, unstructured data created by IoT networks. While ML works fine with smaller data, it struggles to identify complex patterns and interactions in large-scale IoT scenarios. Moreover, ML is more likely to be rooted in hand-crafted feature engineering, which limits its reactivity to new threats. By contrast, DL excels at learning features autonomously from raw input, and it is more adapted

to the heterogeneity and dynamism of IoT systems. In (Al-Amiedy et al., 2022), certain ML techniques are analyzed for predicting and detecting RPL attacks in 6LoWPAN networks. They are k-means clustering, decision trees, and the hybrid model that uses both for improved detection accuracy. Ensemble methods such as boosted trees, bagged trees, subspace discriminant, and boosted trees are used to make the model robust. SVM, ANN, and other ML methods are also used to detect RPL-based attacks and give a robust framework for 6LoWPAN network security. This (Al-Amiedy et al., 2022), examines several deep learning models to detect RPL-based attacks in 6LoWPAN networks. These are MLP, ANN, DNN, DBN, CNN, GRU, and LSTM. Graph CNN is used for the efficient processing of complex network structures and offers solutions for enhancing RPL security.

The federated learning technique addresses the challenges discussed above in (Rey et al., 2023) by enabling decentralized learning on a device in real-time, improving privacy, latency scalability, and bandwidth efficiency. No raw data is shared between a device and a server, but a device learns locally and shares updates with a model, reducing data transmission significantly. This distributed learning technique prevents sensitive information from being sent to the cloud (R. Khan et al., 2024). Federated learning algorithms are as follows:

The Federated Averaging Algorithm (FedAvg) is a popular federated learning algorithm that is easy and efficient. Clients can train local models on their data and then average the model updates among all clients to update the global model. FedAvg is computationally efficient, especially with independent and identically distributed data. The server initiates the global model and sends it to the clients. The clients train the model and send updates to the server, aggregating them to improve the global model every round (Nilsson et al., 2018).

Federated stochastic variance reduced gradient (FSVRG) begins with a computational step in which the central server calculates the full model parameter gradient. After the initial gradient calculation, local gradients are computed by each client and compared to a stored global reference gradient, with the updates corrected in such a way as to reduce variance and improve convergence. This approach reduces the variance by sometimes updating the gradient to update the noise that may be caused by client updates, especially if client data is not homogeneous (Nilsson et al., 2018).

2.2. **Related Works**

This section reviews existing studies on RPL security, providing a summary of the current methods and their strengths and limitations.

(Yavuz et al., 2018) proposed a deep learning-based method to identify routing attacks in IoT networks, following a structured three-stage process: data simulation, feature processing, and model application. They generated a dataset, IRAD, using the Cooja simulator to replicate three RPL attack types, such as HF, VN, and decreased rank attack. During preprocessing, 18 critical features were selected using tools like decision trees, Pearson correlation, and histograms. In the final stage, the MLP model with a Sigmoid function was used to distinguish between normal and attack behaviors, achieving an accuracy of 99.5%. The model showed high detection capability with F1-scores of 94.7% for DRA, 99% for HF, and 95% for VN, and respective AUC scores of 94.2%, 98.1%, and 94.7%.

This study(N. W. Khan et al., 2023) introduces IDS that combines supervised and semi-supervised deep learning techniques to classify network traffic for both known and unknown abnormal behaviors within an IoT environment. The IoTR-DS dataset serves as a case study for detecting and classifying three well-known security attacks: DIS flooding, Rank, and wormhole. The proposed hybrid DL-based IDS is thoroughly evaluated. The evaluation demonstrates a detection accuracy of 95% for predicting attacks.

This study (Albishari et al., 2023) presents a DL-based early detection approach for IoT routing attacks, utilizing the IRAD dataset, which covers hello flood, decreased rank, and version number attacks. To evaluate training efficiency, various classifiers were applied, including logistic regression, KNN, SVM, naïve Bayes, and MLP. The findings indicate that the MLP classifier achieved superior results, with 98.85% accuracy, 97.50% precision, 98.33% recall, and a 97.01% F1 score, surpassing existing models.

In (Choukri et al., 2020), a DL solution for ranking attack detection in RPL networks was presented, utilizing an MLP to classify network traffic as normal or suspicious. The authors developed their dataset using the Cooja simulator, changing parameters such as ICMPv6 and DAG RPL files to simulate RPL attacks. Data preprocessing involved feature extraction, normalization, and sampling to obtain the dataset MLP-trainable. The model used ReLU as the activation function and split the dataset into 80% training and 20% testing. It achieved a 100%

true positive rate (TPR) and a 24% false positive rate (FPR) with precision, recall, F1-score, and accuracy (AC) of 98%, 88%, 92%, and 96%, respectively.

(Morales-Molina et al., 2021) presented a deep neural network (DNN) solution that utilized the detection of coordination intrusion detection (CID) attacks in RPL networks. They used the Cooja simulator to create a dataset with various topologies that contained normal and malicious nodes. The proposed solution involves three primary phases: data preprocessing, unsupervised pre-training, and supervised classification. In the preprocessing phase, they used random over-sampling to balance the dataset, one-hot encoding to transform features, and normalization of the data. They employed an autoencoder for unsupervised pre-training and subsequently a DNN for classifying normal and anomalous network traffic.

(Sahay et al., 2022) Proposed a strong framework for routing attack prediction in IoT low-power and lossy networks (IoT-LLNs). They employed various DL tools, including graph CNN and LSTM models, to effectively learn the spatial and temporal patterns of IoT-LLNs. the FNN model took the outputs of the LSTM to predict attacks. The experimental findings indicated that the proposed framework achieved 94.50% accuracy under normal conditions, 86.13% for topological attacks, 82.46% for resource attacks, and 91.88% for traffic attacks. The authors asserted that their framework can detect and identify various kinds of routing attacks.

A machine learning-based technique to secure the RPL routing protocol, named MLRP, was presented to detect several RPL-based attacks like version number, rank, and DoS attacks (Momand et al., 2021). Researchers have extracted features from packet capture files and utilized various preprocessing techniques. PCA was utilized by them for reducing features and identifying the best features to detect attacks. The SVM classification module was then used to identify and classify RPL attacks based on the training data. As per the experimental results, MLRP was found to achieve an average accuracy of 0.90-0.92. This article suggests a deep hybrid learning model in an asynchronous federated learning framework to enhance cyberattack detection and data privacy in industry IoT systems.

Combining CNN, GRU, and LSTM as one model, the model effectively identifies anomalies in sensor traffic without full node synchronization while maintaining localized data. With an accuracy score of 1.00% in diverse IoT environments, the model demonstrates a strong

adaptability and responsiveness to new threats, moving IoT security and data privacy forward (Muhammad Salman Bukhari et al., 2024).

The paper in the study (Yadollahzadeh-Tabari & Mataji, 2021) introduced an anomaly-based distributed IDS for identifying sinkhole attacks in RPL networks. Their approach utilized a hybrid placement strategy, with monitoring nodes deployed on hosts and an IDS agent operating within the border router. The workflow consists of several stages: first, a lightweight module in each network node collects data and sends it to the border router. The preprocessing step then filters out invalid and faulty data. A machine learning algorithm, genetic programming, selects significant features for high classification accuracy. The classification stage employs three algorithms: DT, SVM, and Bayesian classifiers. The Bayesian classifier achieved the highest detection rate (DR) at all intervals. For false positive rate (FPR), the DT classifier achieved the lowest rates at 2, 4, and 6 minutes, while the SVM classifier had the best FPR at 8 and 10 minutes.

(Pradeepthi et al., 2020) presented an artificial intelligence-based mechanism for identifying selective attacks in RPL networks through an intelligent technique called AI-based packet drop ratio that utilizes neighborhood information. In this protocol, every node is monitoring its packet drop ratio (PDR), whereas the border router verifies the PDR of other nodes. Malicious nodes are detected and subsequently removed based on their PDR values. Simulation results showed that the proposed scheme here enhanced the PDR to 89% for 25 nodes and reduced the end-to-end delay from 14 to 4. The true positive rate was 72%, while the false positive rate was as low as 0.3% for 25 nodes.

The authors (Shahid et al., 2024) emphasized the importance of the ROUT-4 dataset in addressing security challenges related to RPL attacks in IoT settings to enable the design and testing of IDS systems using simulation of different attacks, such as flooding, black hole, DODAG version number, and decreased rank attacks. The research uses exploratory data analysis to improve detection algorithms by examining patterns of attack and resulting network traffic. It utilizes a range of machine learning and deep learning algorithms, such as RF, DT, naive Bayes, transformers, CNN, and LSTM, where RF shows the best performance. Lastly, this paper provides a whole framework for improving IoT security and developing more effective IDS.

The authors of the paper (Cakir et al., 2020) recognized the effectiveness of deep learning methods, particularly GRU, for detecting IoT routing attacks. The paper highlights the advantages of GRU, such as simple architecture and fast training time compared to LSTM networks. GRU was trained on the created dataset from the simulation to detect hello flood attacks. The result reveals a 99.96% accuracy rate, which surpasses other methods like SVM and LR in delay and packet delivery ratio. While the study cites potential scalability problems with additional nodes, it offers GRU-based as a viable solution for routing attack detection and prevention in IoT, with plans for future work on other attack types.

(R. Khan et al., 2024) Introduce a streamlined FL model to detect selective forwarding attacks using the IRAD to enhance detection efficiency. Binary classification methods were employed to evaluate the model's training performance using various machine learning algorithms, including LR, KNN, SVM, and NB. Among these algorithms, SVM and KNN produced the best results in terms of training accuracy and runtime efficiency. The model achieved an accuracy of 95%. These results demonstrate superior performance over current approaches in the field, with notable strengths in classification accuracy, scalability, and privacy protections.

2.2.1. **Summaries of Related Work**

The table below summarizes the related work on detecting RPL attacks in 6LoWPAN networks. Each study is examined in terms of its dataset, methodology, attacks, and limitations to present a comprehensive overview of the research landscape in this field.

Table 2- 1:Summaries of related work

Author	Method	Dataset	Privacy	Attacks detected	Limitation
(Shahid et al., 2024)	LSTM, CNN RF, DT	ROUT-4 dataset	No	BH, HF, DR	• Only for small area network • Raises concerns about data privacy • Focus on only detection • The method do not handle spatial relationship between RPL features and relationship of them over time.
(Sahay et al., 2022)	LSTM, GCNN	IoTR-DS dataset	No	DR, WP	• The privacy of the RPL traffic dataset is not preserved • Only focus on detection and a lack of a comprehensive dataset • Focus on only a small network size

(Choukri et al., 2020)	MLP	Self-generated dataset	No	Rank	• Large number of false positives. • Only focus on rank attack detection • No details about the generated data
(Kamel & Elhamayed, 2020)	CNN	IoT routing dataset	No	HF, SF	• Do not capture temporal features in network traffic effectively. • No details about the used feature and generated data
(Cakir et al., 2020)	GRU	Self-generated dataset	No	HF	• Lack to handle spatial relationships between RPL network features at a specific time window. • Only focus on the HF attack
(R. Khan et al., 2024)	KNN, SVM, LR	IRAD	Yes	SF	• Limited to the SF attack • Unable to handle intricate RPL traffic feature.
(N. W. Khan et al., 2023)	DAE-DANN	IoTR-DS dataset	No	DIS flooding, WH, Rank	• Does not preserve the privacy of IoT nodes
(Albishari et al., 2023)	MLP, SVM, KNN	IRAD	No	DR, SH, SF	• raise concerns about data privacy. • Do not handle extricate interaction b/n network features.
(Muhammad Salman Bukhari et al., 2024)	LSTM, GRU	Edge-IIoTset	Yes	Non-RPL attack	• Typically, did not target RPL attacks • Limited to detection

BH: black hole, SF: selective attack HF: hello flooding VN: version number WH: Wormhole DVN: DODAG version number WP: Worst Parent, SH: sinkhole.

CHAPTER THREE

3. METHODOLOGY AND MATERIAL USED

A federated learning framework was chosen in this research because it offers a step toward the privacy of the data. One big advantage is that it helps keep data private, the data stays on each device at training time, which is important when we're dealing with sensitive information. the model can learn from different data sources without needing to collect everything in one place, which cuts down the bandwidth consumption. It only shares the model parameters with the aggregator, not the data itself. Thus, in addition to keeping privacy, it cuts down on storage and bandwidth use at training time.

3.1. Research Design

we aimed to develop a federated hybrid deep learning model for detecting RPL attack types within 6LoWPAN networks. The proposed model identifies attacks, including VN, BH, and HF within a single system. To achieve this, the study utilizes the IRAD dataset, which contains labeled instances of both normal and attack traffic. The dataset is created from different network size that comprises four classes. Preprocessing techniques have been used to optimize the dataset for federated hybrid deep learning applications, including class balancing, normalization, and outlier detection. Different feature selection methods have been utilized to identify the most relevant attributes, ensuring effective classification of different attack types.

Instead of training separate models for each type of attack, a single unified dataset has been prepared to enable the model to detect multiple attacks simultaneously. More than one attack may simultaneously happen, so this approach mimics real-world network scenarios, allowing the model more generalizable and adaptable (Alrefaei & Ilyas, 2024). The model's performance is evaluated individually for each attack category to ensure a comprehensive assessment. Key performance metrics, including precision, recall, F1-score, and accuracy, are reported to measure the model’s effectiveness in handling different attack scenarios.

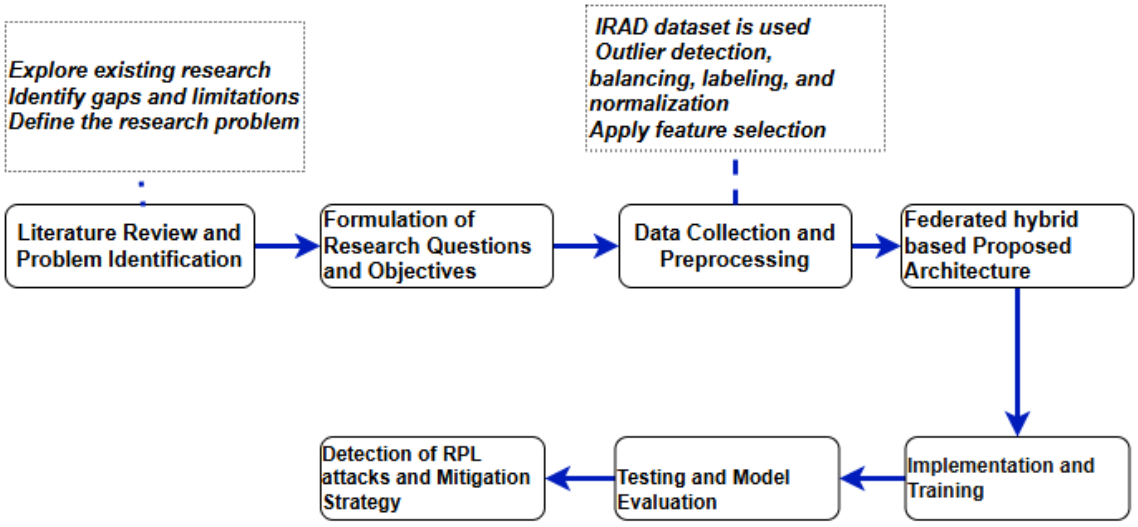

Figure 3- 1: Flow of research methodology.

- Literature review and problem identification: At this stage, the gaps in RPL protocol were extensively derived from related works. In order to have a comprehensive understanding of the subject, relevant works of literature have been prudently collected from authoritative digital libraries and search engines such as IEEE Xplore, Scopus, ACM Digital Library, MDPI, SpringerLink, ScienceDirect, and Google Scholar. The purpose of this was to determine the gaps in knowledge, the challenges of existing solutions, and comprehensive datasets. As a result, the essential problem has been precisely stated in this process, and it becomes the keystone of the present work.
- Formulation of research questions and objectives: With the infusion of ideas from the literature survey work scope well-defined research questions and objectives were set. These were utilized to guide the study in the aspect of detection and classification of attacks on RPL protocol with a federated deep learning approach.
- Data collection and preprocessing: The IRAD Dataset comprises traffic data related to Blackhole, Version Number, and Hello Flooding attacks which were collected as the main source of data. To increase diversity and fidelity, we used samples from four different network sizes (10, 20, 100, 1000 nodes). This enabled the model to generalize to different network scenarios. Data preprocessing such as feature selection, data balancing, normalization, and labeled data cleaning were carried out so that the data is suitable for reliable training and testing.

- Federated hybrid-based proposed architecture: Hybrid deep learning models have been designed under a federated learning environment. The federated hybrid models are adopted in order to exploit both spatial and temporal characteristics and maintain data privacy on distributed nodes.
- Implementation and training: We implemented the proposed models based on TensorFlow/Keras, where training was performed in a federated manner across multiple emulated clients.
- Evaluation on testing data: After training was done, the models were tested on unseen data, to gauge how they performed. Some crucial evaluation scores like Accuracy, Precision, Recall, and F1-score were taken to compare the ability of the models to detect different classes of RPL attacks.
- Detection of RPL attacks and their mitigation schemes: The proposed models can detect several types of RPL attacks such as BH, HF, and VN using the trained hybrid models.

3.2. **Dataset Description**

The IRAD is a publicly comprehensive available and widely recognized dataset for RPL-focused research. They offer valuable resources for developing advanced attack detection systems in IoT networks. The IRAD is used for model training due to its comprehensive scope, and data generated from 10 to 1000 nodes, which allows the model to be used on small, medium, and large-scale networks that mimic real-world scenarios. The dataset's source is freely accessible to the public on the GitHub website(https://github.com/iot-attacks/irad.) For this research, we relied solely on the IRAD dataset to train federated hybrid deep learning models. The dataset contains data related to different attacks, such as version number attacks, black hole attacks, and flooding, as well as normal network behavior.

Although the dataset was simulated, it was generated using a powerful 48 GB RAM virtual environment that incorporates node mobility and realistic RPL routing behavior. This allows the dataset to reflect dynamic IoT scenarios more accurately than static-topology datasets. Thus, it is well-suited for real-world scenarios in the dynamic IoT realm where nodes may move or experience varying communication patterns.

The dataset includes 18 features and 1 label. Features like Length and Info describe the packet structure and type, which help identify abnormal packet content. TR (transmission rate), RR

(reception rate), and the TR/RR ratio capture the balance of sent and received packets, revealing potential packet drops or flooding. Dataset features such as TAT, RAT, TTT, and TRT have transmission and reception behavior over time, used in the detection of attacks like BH or VN manipulation. TPC and RPC indicate how many sources and destinations are active per second, highlighting abnormal node behavior or routing changes. The attackers utilize RPL control messages such as DAO, DIS, and DIO to introduce their malicious activities. All features together provide a comprehensive view of network behavior. As shown in Table **3- 1**, the last feature identifies whether the data point is BH, VN, HF, or benign, while the Label feature classifies it as normal or malicious. These features are used to train the model to accurately detect and classify attack patterns and network behavior.

Table 3- 1: Dataset features with data type and description

No.	Feature	Description
1	No.	Packet sequence number
2	Time	Simulation time
3	Source	Source node IP
4	Destination	Destination node IP
5	Length	Packet Length
6	Info	Packet information
7	TR	Transmission rate
8	RR	Reception rate
9	TR/RR	The ratio of transmission rate to reception rate
10	TAT	Transmission (Trans) average per sec (Transmission average time)
11	RAT	Reception (Rcv)average per sec (Reception average time)
12	TPC	Source count per sec (Transmitted packets)
13	RPC	Destination count per sec (Received packets)
14	TTT	Trans total duration per sec (Total transmission time)
15	TRT	Reception (Rcv) total duration per sec (Total reception time)
16	DAO	DAO packets
17	DIS	DIS packets
18	DIO	DIO packets
19	Label	Label: Benign/Malicious

Table **3- 2** depicts a detailed overview of the raw dataset before sampling. It details the total number of nodes, the ratios of malicious to benign nodes, the total packets per scenario, and the associated benign configurations. The dataset was generated based on four scenarios such as 10

nodes, 20 nodes, 100 nodes, and 1000 nodes. Each attack was examined in each network scenario. This diversity in network size and attack types ensures that the model generalizes well across different IoT deployment scales, mimicking real-world conditions.

Table 3- 2:Detail distribution of raw dataset

Category	Scenario	Nodes	Malicious/Benign Nodes	Malicious Packets	Matching Benign Scenario	Benign Packets
Decreased Rank	DR10	10	2 / 8	130,240	Benign10	121,047
	DR20	20	4 / 16	398,143	Benign20	120,897
	DR100	100	10 / 90	456,816	Benign100	150,078
	DR1000	1000	100 / 900	801,396	Benign1000	739,845
Hello Flood	HF10	10	2 / 8	100,538	Benign10	121,047
	HF20	20	4 / 16	104,302	Benign20	120,897
	HF100	100	10 / 90	300,986	Benign100	150,078
	HF1000	1000	100 / 900	1,739,226	Benign1000	739,845
Version Number	VN10	10	2 / 8	112,044	Benign10	121,047
	VN20	20	4 / 16	203,153	Benign20	120,897
	VN100	100	10 / 90	223,275	Benign100	150,078
	VN1000	1000	100 / 900	1,585,075	Benign1000	739,845

3.3. **Dataset Preprocessing**

Preparing the dataset is crucial to ensure clean and consistent input for deep learning. Appropriate preprocessing improves model performance, reduces training time, and can mitigate learning biases. The following are the preprocessing steps:

Outlier detection: In this phase, the dataset was analyzed for anomalies or extremely high values or lower values that could distort the model's learning process. This step is crucial in ensuring that the model is trained on clean, representative data, security-related IoT datasets where IoT device noise or transmission error can introduce unexpected value

Data Normalization: The dataset features attributes with diverse numerical scales. Using raw values directly may cause some attributes to excessively impact model training. All attributes undergo min-max scaling to address this issue, making certain they lie within a consistent range and enhancing model reliability.

Label Encoding: The dataset has various RPL attack types, necessitating suitable encoding for classification. The label column is converted to integer representations corresponding to different attack categories and benign traffic.

Data Balancing for Federated Clients: Federated learning presents additional challenges concerning data imbalance, as different clients might have distinct distributions of attack types. To facilitate equitable training across all federated nodes, class labels were assigned in a round-robin manner during the balancing process to ensure equal representation of each class.

3.4. Feature Selection Techniques

Feature selection is a dimensionality reduction technique and way of selecting relevant features from a dataset which involves identifying features that have a high relation or score to dependent variables. it can lead to improved accuracy, faster training and detection times, and reduced model complexity and computational cost.

3.4.1. Embedded Methods

Random Forest and XGBoost: These are tree-based feature selection important. These models give importance scores based on how frequently a feature is used to make key decisions within the tree structures. Features with higher importance scores significantly impact attack detection and are prioritized for inclusion in the final feature subset. The combination of Random Forest and XGBoost ensures that both individual and ensemble-based feature interactions were considered, leading to a more robust selection process.

3.4.2. Filter Methods

Correlation-Based Analysis: Features were ranked using this method according to how well they correlate with the target variable and how well they correlate with each other. It is most suitable if it is a continuous categorical variable. It ranks the features according to entropy and determines which are most relevant for each feature (Seba et al., 2024).

3.5. Deep Learning

Based on recent state-of-the-art studies, the following training algorithms are among the most commonly employed approaches for anomaly detection in RPL-based IoT environments (Bokka & Sadasivam, 2023):

RNNs are a type of ANN where units are interconnected in loops, enabling them to process sequential data by maintaining an order-dependent flow of information. RNNs face limitations with short-term memory, struggling with long-term sequence to retain and propagate information across multiple time steps, which can result in the loss of crucial data from earlier in the sequence, affecting prediction accuracy (Cakir et al., 2020).

LSTM is tailored for performing well on extensive data sequences because the ability to remember long-term dependencies and backpropagate optimally solves the shortcomings of RNN short-term memory limitations. Despite their effectiveness, LSTM has a complex architecture and it is computationally intensive which is not directly tailored for constrained IoT devices, prompting the development of GRU(Fu et al., 2017).

The structure of a GRU is closer to that of an LSTM unit but has a less complicated architecture. Rather than having the input, output, and forget gates used in LSTM, GRU has two gates: the reset and update gates. The reset gate determines the degree to which the new input will be blended with historical data, while the update gate determines how much of the previous memory will be kept. These gates govern the information flow and preserve relevant information while eliminating irrelevant information. In this way, the model can learn what elements of the sequence are critical in generating predictions (Cakir et al., 2020)

MLP is effective for various classification tasks, such as detecting anomalous behavior in network traffic (Najar & Manohar Naik, 2022). It captures non-linear relationships and it is effective when dealing with tabular data. MLPs provide a simplified approach for classifying target labels, requiring less computation than more sophisticated architectures. CNN is particularly good at recognizing patterns such as network traffic. It can autonomously extract critical features from input data such as packet size, packet information, and packet length while capturing both short-term fluctuations and long-term trends (Abd-El-Atty, B., ElAffendi, M., & El-Latif, 2023)

The Hybrid deep model improves RPL attack classification in the 6LoWPAN realm by using the advantage of more than two deep learning. It can capture complex network patterns and behavior and leverages the complementary strengths of hybrid architectures, which excel at identifying spatial patterns and handling temporal dependency in sample data analysis (Berguiga et al., 2025). The hybrid framework surpasses individual model performance by

overcoming the challenge that standalone models face, resulting in an outstanding accuracy attack identification system (Fu et al., 2017). CNN-LSTM and CNN-GRU are hybrid deep models, one of these excels at identifying spatial features, while the other architecture is designed to capture temporal relationships. This combination allows for handling a correlation relation between features from the dataset using CNNs and LSTMs or GRUs layers to handle features that change over time. This model is particularly effective when both spatial and temporal aspects are critical, making it valuable for tasks such as RPL attack detection (Benaddi et al., 2023; Elakkiya et al., 2024; Hadipuspito & Suryani, 2024).

3.6. Federated Learning Framework

The federated learning framework employed here includes hybrid deep models to learn efficiently local and temporal dependencies within data. For the nature of the dataset, performance requirements, and computational capacity, FedAvg is selected as the aggregation method. FedAvg is perhaps the most well-known algorithm in federated learning since it is lightweight, efficient, and can handle decentralized data without heavy communication overhead. This option guarantees optimal use of computational resources and the protection of sensitive information while minimizing bandwidth utilization (Sun et al., 2023).

Dataset partitioning: In order to maintain the privacy of the data and simulate real federated learning environments, the data is divided into partitions across multiple nodes. Each client possesses a distinct portion of the dataset, which is trained locally. This method guarantees no sharing of data with one another by being able to maintain the actual data as private while each client trains the model.

Model initialization: Federated learning's algorithm begins with initializing a global model on the central server. The global model is randomly initialized on a similar task to accelerate convergence. Effective initialization enhances the stability of federated learning processes.

Local model training: All the clients train the model locally using their dataset. The training process involves feature extraction using deep learning models that are tailor-made for federated environments. Training is done across several local epochs, where the model gets optimized based on a classification-appropriate loss function.

Model aggregation using Federated Averaging: In each round, clients share the model's parameters with the server instead of raw data following local training. The FedAvg algorithm aggregates these parameters by computing a weighted average of the number of samples at each client. This maintains data privacy and the global model can be refined progressively.

Global model update: In the case of global model parameter updates, the new model is pushed back to all clients to continue training on their local data. The loop of local training, parameter aggregation, and global model updates continues until the model achieves the maximum level of performance.

3.7. Performance Evaluation Metrics

Several measures of performance were utilized to evaluate a classification model, although the measure used is problem-specific.

Accuracy: Estimates the proportion of correct predictions to the overall predictions. It is the most widely used measure in classification analysis. It estimates the proportion of correct predictions by the model to the overall predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \qquad 3\text{- }1$$

Precision: Measures the proportion of correctly identified positive cases out of all cases predicted as positive. It determines the model's ability to avoid false positives. In other words, precision measures how often the model correctly predicts a fault when there is one.

$$Precision = \frac{TP}{TP+FP} \qquad 3\text{- }2$$

Recall: Measures the proportion of actual positives that the model correctly identified. It is critical for attack detection since it assesses the model's ability to capture all true attack instances. In other words, recall measures how many of the actual attack data are correctly predicted as faulty by the model.

$$Recall = \frac{TP}{TP+FN} \qquad 3\text{- }3$$

F1 Score: The harmonic mean of precision and recall, providing a balanced measure when both false positives and false negatives are important. Convenient when the data has an imbalanced dataset, as it balances precision and recall into a single measure. In our case, the F1 score would

provide one measure of the model's overall performance in predicting the occurrence or non-occurrence of attack in the RPL data, considering both false positives and false negatives.

$$Recall = 2 * \frac{Recall*Precision}{Recall+Precision} \qquad 3\text{- }4$$

Confusion matrices: It is a widely used tool in supervised machine learning for evaluating and visualizing the performance of models. It is structured as a square matrix where the rows represent the actual class and the columns represent the predicted classes.

False positive rate (FPR): It is calculated as the ratio of the number of false positive (FP) predictions to the total number of actual negative instances in the dataset. It offers insight into the model's tendency to over-predict the presence of attacks on RPL.

$$FPR = \frac{FP}{FP+TN} \qquad 3\text{- }5$$

False negative rate (FNR): It is calculated as the ratio of the number of false negative (FN) predictions to the total number of actual positive instances in the dataset. It reflects the model's tendency to miss actual attacks, predicting them as benign traffic.

$$FNR = \frac{FN}{FN+TP} \qquad 3\text{- }6$$

3.8. Design and Development Tools

3.8.1. Design Tools

Design tools have been used in this thesis proposal to build and design the research methodology framework and work plan. Draw.io has been used and will be used to design various diagrams, architecture, work plans, and flowcharts.

3.8.2. **Deep Learning Libraries and Tools Utilized**

A functional implementation environment is necessary for any research or system development to guarantee that the suggested system can be adequately trained, assessed, and tested. The suggested systems have been developed, operated, and tested using various hardware, software, and toolsets. These have been separated into two groups and covered below.

- **Python**: Selected due to its flexibility, and broad deep learning support. It contains a massive number of libraries to create and implement training and testing of the model.

- **TensorFlow**: Functions as a core deep learning framework in this work owing to its scalable computation nature for neural networks. It is compatible with both CPU and GPU acceleration and facilitates scalable, high-performance training of models.
- **TensorFlow Federated**: Is an open-source framework by Google to run federated learning and other types of distributed machine learning. It allows federated learning of models on decentralized devices or servers with data protection.
- **Keras**: An API running on top of TensorFlow, it makes it easier to implement and train deep learning models. The high-level API reduces the number of lines of code you need to write, with no loss of flexibility.
- **Pandas**: For data preprocessing, processing, and manipulation: to manage, and analyze data, especially in the form of fast table-based data.
- **NumPy**: Enables numerical calculations to be efficient and large-scale as well as provides core building blocks to perform math operations on data.
- **Scikit-Learn**: This is used for evaluating models with performance metrics, data preprocessing, feature selection, and dataset split.
- **Matplotlib:** This is used for visualizing training and model performance by plotting loss curves, accuracy trends, and other observations.
- **Jupyter Notebook**: Served as the development environment, allowing for interaction execution and testing models.

3.8.3. **Deployment Environment**

The proposed system was deployed on the following hardware and software assets:

Desktop Configuration

- **Operating System**: Windows
- **System Type**: 64-bit operating system, x64-based processor
- **Processor**: Intel® Core™ i5-4590 CPU @ 3.30GHz (4 cores, 4 threads)
- **Memory (RAM):** 8 GB

Laptop Configuration

- **Model**: HP pro
- **Operating System**: Windows
- **System Type**: 64-bit operating system, x64-based processor
- **Memory (RAM**): 4 GB

CHAPTER FOUR

4. PROPOSED SOLUTION ARCHITECTURE

4.1. Introduction

So far, we have discussed the literature review, the research methodology, statement of the problem, and the research questions. In this section, we describe in detail the architecture of detection and mitigation to bridge the gap and answer the research questions raised above.

4.2. Proposed RPL Attack Detection and Classification Architecture

The major gap identified in the literature is that most existing research on RPL attacks relies on centralized learning approaches, which expose sensitive network data to privacy risks and increase communication latency at training time. Additionally, most prior works focus solely on binary classification, detecting attacks without distinguishing between different attack types. This limits real-world applicability, where networks confront multiple RPL attack variations. Our proposed systems address these limitations by utilizing a federated hybrid deep learning model. This study experimented with CNN-GRU, CNN-LSTM, GRU, and LSTM to perform multi-class classification tasks.

The dataset used for training is generated from different network sizes, enhancing the model's robustness and generalizability across diverse IoT environments. These models were trained in a federated learning setting, preserving data privacy while improving detection accuracy. By detecting and classifying multiple attack types, the system better mimics real-world network security challenges, making it a more effective solution for RPL attack mitigation.

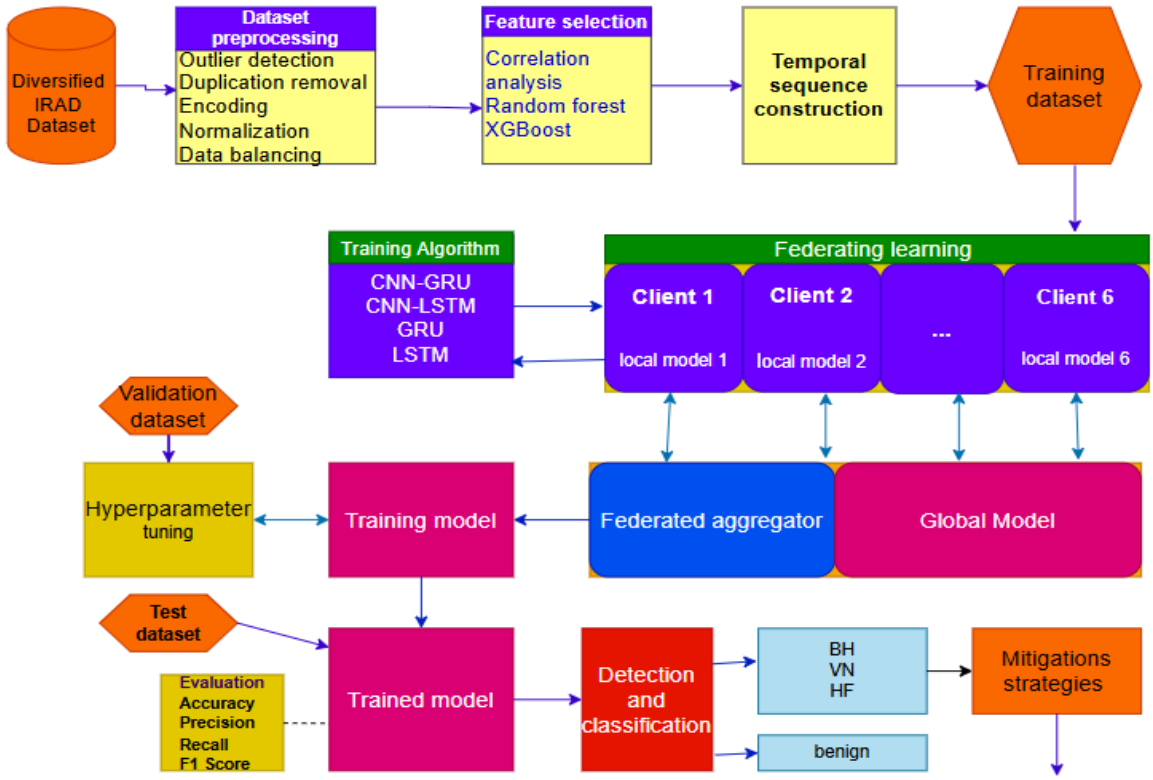

Figure 4- 1 Proposed architecture for RPL attack detection and classification

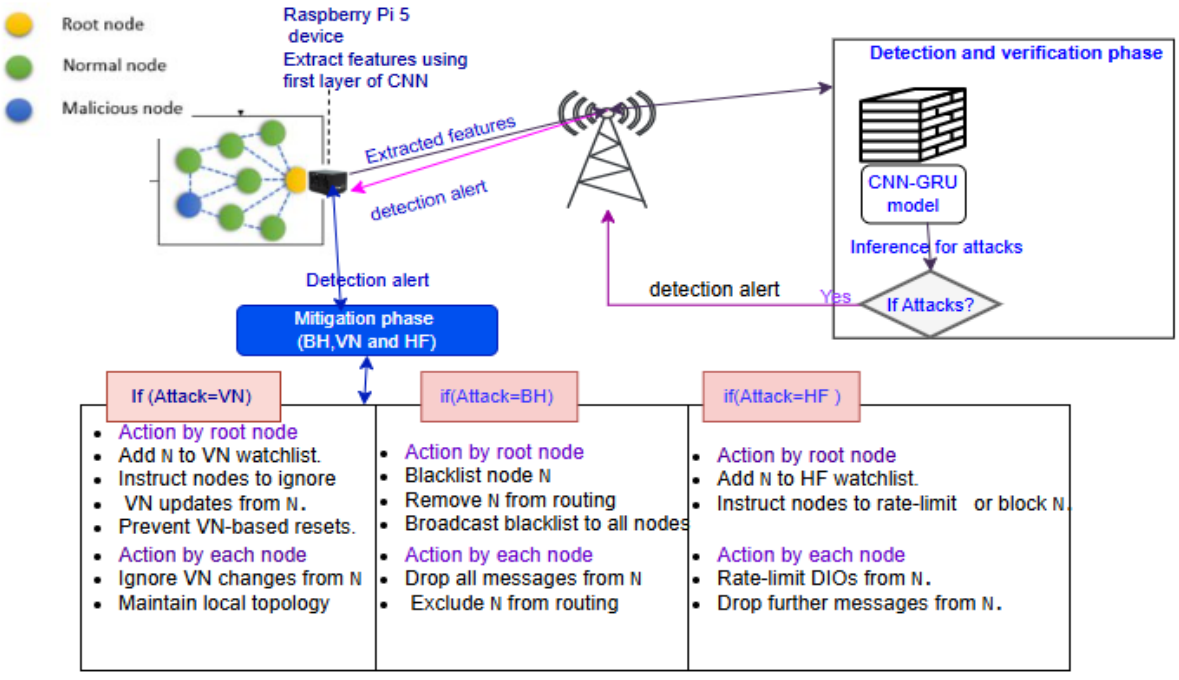

Figure 4- 1: Proposed conceptual architecture for mitigation of attacks on RPL

This diagram depicts a conceptual design for the identification and prevention of three major RPL attacks in 6LoWPAN networks, i.e., Version number, Blackhole, and Hello flooding attack. The system uses a CNN-GRU deep learning model to detect malicious activity based on features extracted from RPL traffic. The model analyzes these features to determine whether or not an attack has occurred. Following attack detection, there is a mitigation phase. Based on the type of detected attack, appropriate countermeasures are initiated. Root node operations are performed centrally in the RPL DODAG, such as ignoring faulty version updates, deleting infected nodes from routing tables, and enforcing rate limits on unbounded control traffic. Local operations are also performed by local network devices, such as packet dropping from attacking nodes or their exclusion from route options. This mitigation framework demonstrates how detection using deep learning can be supplemented with mitigations to enhance the robustness of 6LoWPAN networks against RPL attacks.

4.3. **Hybrid Deep Learning Architecture**

We have proposed two Keras structures for implementing the models. The first architecture, as shown in Figure **4- 3**, is a parallel architecture implemented with the Functional API. Two branches in this architecture accept input shape independently and process. One branch uses convolutional layers to learn spatial correlations between features. The other branch accepts the input directly through GRU layers, with a sole focus on temporal dependencies. This split allows the model to capture both spatial and sequential features simultaneously, rather than using a linear combination. Both branches are subsequently flattened and concatenated after processing, the output is fed to a dense layer. This concatenated representation is passed through the same fully connected structure as the sequential model dense layer with 128 neurons, followed by a dense layer with 64 neurons, both with ReLU activation and dropout for regularization. The final Softmax layer again outputs 4 neurons, according to the four-class classification problem.

The second architecture in Figure **4- 4** represents a sequential model where the input is processed linearly in a sequential manner. The model begins with convolutional layers to capture local spatial patterns and correlations between discovered features over the input vector. These convolutional layers extract features and learn the relationship between features at a specific time in a good way. The output from this block is passed through a stack of GRU layers,

which are added specifically to address the temporal dynamics of the data and capture dependencies emerging over time steps. The sequential flow ensures temporal learning is directly influenced by spatially accurate features. After the GRU layers, the model flattens the representation and passes through two dense layers, the first with 128 neurons for learning higher-level abstractions and the second with 64 neurons for refining the learned representation. Finally, the model makes a prediction using a Softmax layer with 4 neurons corresponding to the four target classes in the dataset benign, BH, HF, and VN. Both were trained in a federating learning fashion and the sequential outperformed the API functional model.

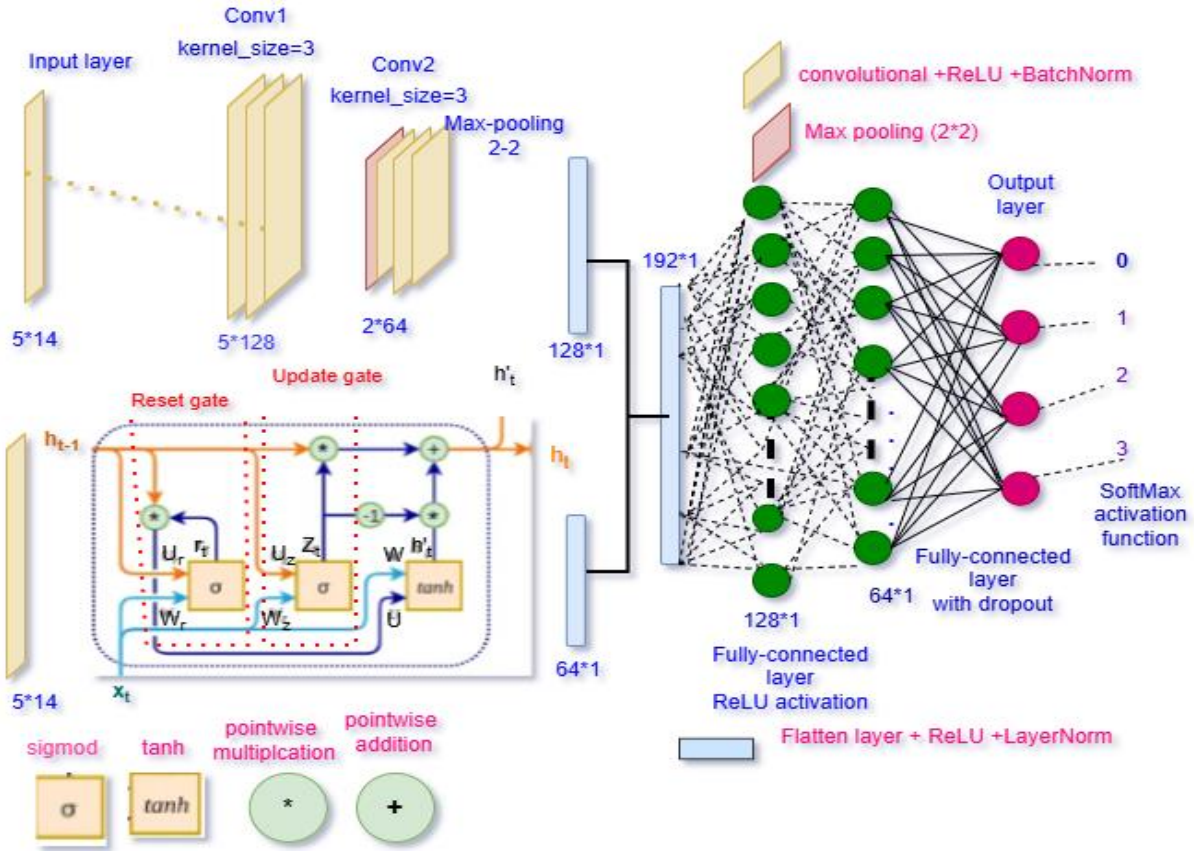

Figure 4- 2: Proposed parallel hybrid CNN-GRU architecture

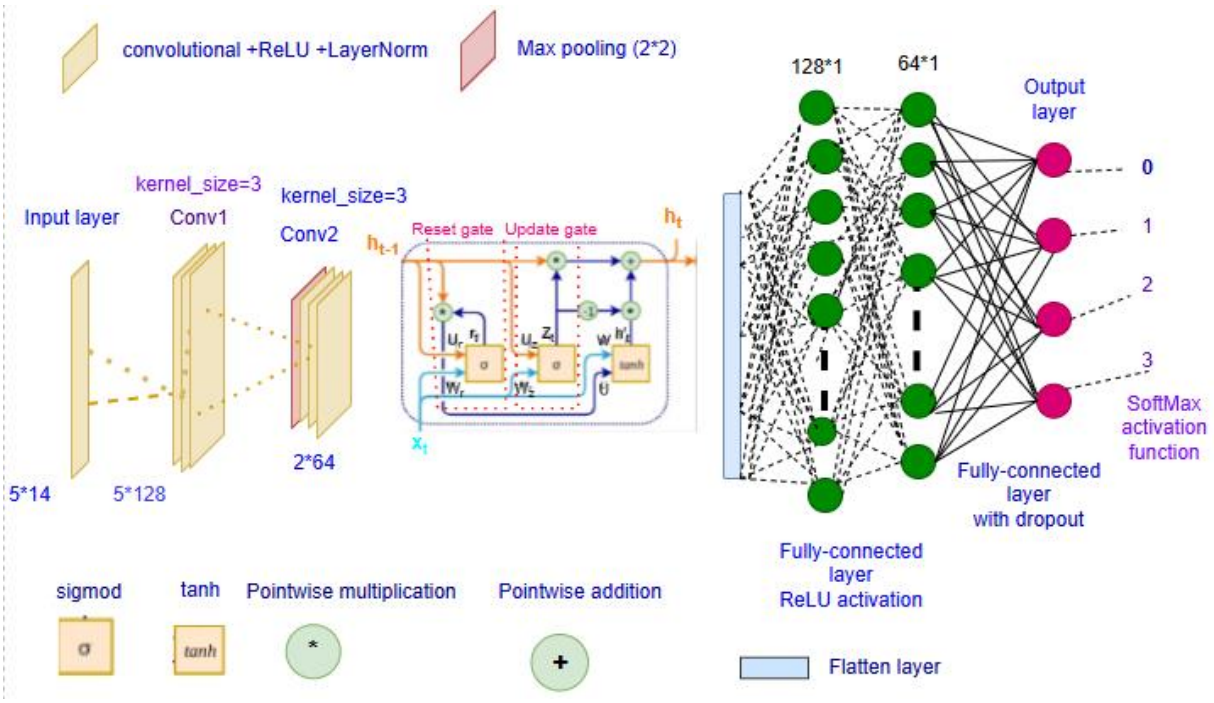

Figure 4- 3:Proposed sequential hybrid CNN-GRU architecture

4.4. Algorithm for Federated CNN-GRU model

Table 4- 1:Algoritm of Federated CNN-GRU Model for RPL Attack Detection.

Algorithm: Federated CNN-GRU Model for RPL Attack Detection.
Input: Federated dataset {D1, D2, ..., Dn} with client IDs Rounds (R), Local Epochs (E), Batch Size (B), Shuffle Buffer (S) Learning rates for clients, server, Adam, and other optimization *Output*: Optimized global CNN-GRU for RPL attack detection *Begin ()* Initialize Federated Learning Environment: Assign unique client IDs. Split the dataset into 70 % for training and 30% for validation and testing. Select a subset of client IDs for training, validating, and testing. Create TensorFlow datasets per client: Filter dataset based on client ID Convert to TensorFlow Dataset batching (B) CNN-GRU Model Definition:

Define Input Layer
CNN Branch:
Conv1D → LayerNorm → Conv1D →LayerNorm → MaxPooling
GRU Branch:
GRU (128) → Dropout → GRU (64)
Concatenate CNN & GRU outputs
Fully Connected (128)→ Dropout (0.2) → Fully Connected (64) → Output Layer(4)
Federated Training Process:
FOR each round (r ∈ {1, ..., R}):
Select a subset of clients' S
Each client C_i trains locally for E epochs:
Retrieve local dataset
Train model using Adam optimizer
Send local model θ_i to the server
Federated Aggregation at Server:
Receive local models {θ1, θ2, ..., θS} from selected clients
Perform Federated Averaging (FedAvg) aggregation:

$$\theta global = \sum_{i=1}^{S} \left(\frac{|Di|}{E(|Di|)} * \theta i\right)$$

Update θ_{global}
Convergence Check
Evaluate θ_{global} on a global test set.
IF validation loss stabilizes or max rounds R reached:
Stop training
ELSE:
Continue training with the next round.
Detection
Apply the model to classify network traffic as VN, BH, HF, and benign.
End Algorithm

4.5. **Algorithm for RPL Attack Detection and Mitigation**

Table 4- 2:Algorithm for conceptual mitigation of RPL attacks.

*Input***:**
Network traffic: Real-time IoT traffic collected
Trained model: CNN-GRU detection model hosted on edge and cloud server
Routing table: The current RPL routing table at the DODAG root
Node list: List of participating RPL node
Output:
Updated routing table: After applying mitigation actions
Updated mitigation instructions were distributed to all nodes
Begin Algorithm
Initialize:

Initialize feature buffer at router border or root device
At DODAG root, Initialize:
blacklisted_node ←0
blacklisted_node ←0
blacklisted_node ←0
For each node sending traffic:
a. At border router:
features = [DIO rate, DAO, reception rate, Rcv Avg/sec, etc]
Normalize features
send features to a cloud-based inference server
b. At main model:
predict label ←CNN-GRU (features)
label: label ∈ {benign, VN, HF, BH}
Attack Detection and Mitigation:
If label == benign:
No mitigation is needed, continue monitoring.
If label == VN:
At DODAG root:
Add N to VN_watchlist
Instruct all nodes to ignore version number changes from N
Prevent topology reset based on fake versions from N
For each node M ∈ Node_List:
Ignore any DIO from N containing abnormal version increments/decrements
Maintain local DODAG consistency
If label == BH:
At DODAG root:
Add N to blacklisted_nodes
Remove N from routing_table
Disseminate blacklist update to all nodes in the DODAG
Local Mitigation (applied at each node):
For each node M ∈ Node_List:
Block routing messages from N
Drop DIO/DAO messages received from N
If label == HF:
At DODAG root:
Add N to HF_watchlist
Send instructions to all nodes to:
(i) Aggressively rate-limit DIO messages from N
(ii) Drop or ignore further communication with N
For each node M ∈ Node_List:
Enforce strict rate limit on DIO messages from N
Block or ignore excessive control messages from N
End Algorithm

4.6. Hyperparameters optimization

A hyperparameter is a configuration value that is set before model training and controls the learning process, ensuring optimal performance on a given task. Identifying the most suitable hyperparameters is a crucial step in model development, as it significantly impacts the model's ability to generalize to unseen data. The following are key parameters that must be considered:

4.6.1. Regularizations Techniques

These are crucial when training models, as they minimize overfitting, ensure stable training and increase performance.

Dropout is a very good method to fight overfitting by deactivating some number of neurons randomly during training. The approach guarantees that the model does not depend too much on some features, thus enhancing its capability to generalize.

Layer normalization normalizes the activations of a layer by calculating the mean and variance over the features for each training example. This makes it invariant to batch size and particularly well-suited for recurrent neural networks and environments where small batch sizes are utilized. We employed layer normalization after each convolutional layer, which enhanced federated learning performance.

4.6.2. Learning Rate

It is the most related hyperparameter to deep learning and machine learning, and it specifies to what extent training parameters are adjusted when training based on the error calculated.

4.6.3. Batch Size

Batch size is a crucial hyperparameter in deep learning that dictates the number of training examples that will be fed to the model before the model's parameters are updated. The appropriate batch size should be chosen carefully to achieve a good balance between computational efficiency and model generalizability. In our experiments on federated models for RPL attack detection, we experimented with various batch sizes to identify the best configuration.

4.6.4. **Activation Function**

Activation functions are employed to bring non-linearity to neural networks so that the model can learn complicated patterns in the data. The selection of a particular function directly affects the network as it affects the accuracy and rate of convergence of the network.

Rectified Linear Unit (**ReLU**) is a very popular activation function in deep learning, thanks to its computational efficiency and its property of preventing the vanishing gradient problem. (Sharma et al., 2020). It is represented by: $f(x) = max(0, x)$ 4- 1

This function ensures only positive values pass through while removing all negative values, introducing non-linearity to activation and enhancing the performance of models. ReLU is very efficient as it allows the model to learn intricate features with quick backpropagation. The **SoftMax** activation function is commonly applied in the output layer for multi-class classification problems as it translates the model's raw outputs to probability distributions across several classes. In the present work, we applied the SoftMax activation function in the final output layer to calculate the probability distribution regarding different types of RPL attacks and normal traffic. This enables the model to assign a probability to each category, thus making it possible to distinguish between various attack patterns, like BH, HF, and VN, and normal traffic.

4.6.5. **Loss Function**

In the context of this thesis, which addresses a multi-class classification problem with integer-encoded true labels, the Sparse Categorical Cross-Entropy loss function is specifically employed to measure the dissimilarity between the predicted probability distribution over the classes and the true class label. It aims to minimize the discrepancy between the expected and actual results. Reducing this loss function improves performance and makes the model's predictions more aligned with the actual values.

4.6.6. **Optimizer Selection in Federating Learning**

Adam (Adaptive Moment Estimation) is also known for its ability to adapt the learning rate for each parameter during training. One of the finest strengths is that it usually works nicely with a very minimal amount of tuning needed. It is a hybrid of the advantage of both RMSProp and momentum and can converge faster in most cases, especially in complex neural networks or

noisy data. This makes Adam a great choice when dealing with deep models or big data where stability and speed are essential (Wang et al., 2024).

SGD (Stochastic Gradient Descent), however, is more straightforward and simpler in the update of its model weights. Its best feature is that it is likely to lead to improved generalization, especially when momentum is incorporated. While it may converge more slowly than Adam, it gives the user finer control and can outperform Adam in some scenarios, particularly where training time is not the critical factor but model correctness is. It's also utilized in research settings where interpretability and high-grained control matter.

CHAPTER FIVE

5. IMPLEMENTATION OF MODEL ARCHITECTURE

The chapter outlines the tools, libraries, and framework used, followed by a detailed explanation of the reprocessing steps, implementation of model architecture, federating learning setup, and training process. Furthermore, it describes the details of the dataset, the feature selection technique, and the evaluation procedures adopted to validate the effectiveness of the developed model.

5.1. Proposed model implementation

5.1.1. Overviews of the Dataset

Developing a robust and generalized deep learning model for RPL attack detection requires a dataset that accurately represents network behavior under both normal and attack conditions. To achieve this, the IRAD, a comprehensive publicly available dataset for RPL-based IoT security research, was utilized. The dataset was produced through actual equivalent simulations utilizing the open-source Contiki/Cooja simulator due to the lack of comprehensive public RPL-based IoT attack datasets before. The Cooja simulation produces raw packet capture files, which are subsequently transformed into comma-separated values (CSV) files for processing in text format. Cooja, a cross-layer network simulator integrated with the Contiki operating system, allows researchers to run actual IoT network stacks and capture network traffic under different conditions. Each simulated IoT node operates using the RPL routing protocol, which is widely used in low-power and lossy networks. The simulation infrastructure consisted of private cloud-based virtual machines with 48 GB RAM and 8 CPUs to accommodate the computational demands of simulating IoT networks ranging from 10 to 1000 nodes.

During the simulation process, both benign and attack scenarios were executed, capturing real-time network activity. The resulting packet-level network data was stored in packet capture format, which was later converted into structured CSV files for further processing. Each dataset contained 18 features, representing critical network metrics such as packet transmission rates, reception rates, and protocol-specific attributes such as DAO, DIS, and DIO messages.

Table 5- 1:Raw dataset samples record

No.	Time	Source	Destinatic	Length	Info
373	2.3868	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
374	2.391132	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
375	2.395466	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
376	2.3998	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
377	2.404134	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
378	2.408466	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
379	2.412799	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
380	2.417134	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
381	2.421472	fe80::c30c:0:0:0	ff02::1a	97	RPL Control (DODAG Information Object)
382	4.572938	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
383	4.576255	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
384	4.579581	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
385	4.582909	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
386	4.586236	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
387	4.589562	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)
388	4.592888	fe80::c30c:0:0:12	fe80::c30c	76	RPL Control (Destination Advertisement Object)

The raw dataset consists of fundamental network communication features, including Time, Source, Destination, Length, and Info, as shown in

Table 5- 1. However, these basic features alone are insufficient for accurately detecting RPL-based IoT attacks. Therefore, additional statistical and behavioral features were derived to capture anomalies associated with malicious activities. During the simulation of the RPL network to generate the dataset, a significant increase in the number of received packets at the malicious node was observed. This anomaly serves as a key indicator for attack detection. Equations 5-1, and 5-2 show reception-based features were calculated within a fixed transmission window of 1000 MS to quantify this behavior**.**

- Reception rate (RR): Measures the rate at which a node receives packets.

$$RR = Received\ Packet\ Count\ of\ the\ Node/1000 \qquad 5\text{- }1$$

- Reception average time (RAT): This represents the average time taken for a node to receive a packet.

$$RAT = Total\ Reception\ Time/(Received\ Packets\ Count\ of\ Node) \qquad 5\text{- }2$$

- Received packets count of node (RPC)**:** Total number of packets received by the node.

Total reception time (TRT): Cumulative time spent receiving packets.

- DIO and DAO packet counts: Since RPL attacks often exploit these control packets their counts are computed to identify irregularities.

HF, BH, and VN attacks lead to an increase in the malicious node's transmitted packets. Therefore, to detect this attack, transmission rate (TR), transmission average time (TAT), transmitted packet count (TPC), total transmission time (TTT), and DIO characteristics were computed.

- TR: Indicates the speed at which a node sends packets.

$$TR = Transmitted\ Packet\ Count\ of\ the\ Node/1000 \qquad 5\text{-}3$$

- TAT: This represents the average time taken for a node to transmit a packet.

$$TAT = (Total\ transmission\ Time)/(Transmitted\ Packet\ Count\ of\ the\ Node) \qquad 5\text{-}4$$

Raw datasets contain data categories, including IP addresses, that the learning algorithm cannot process. The addresses of the source and destination are transformed to node ID from IPv6 format. For instance:

$$fe80::c30c:0:0:12 \Rightarrow 12$$

The following is how the broadcast packets were handled. If the target address in a raw dataset is ff02::1a, then the source node is sending broadcast packets. To prevent any coincidence with another node, its value is changed to 9999:

$$ff02:1a \Rightarrow 9999$$

Additionally, the packets' information in

Table **5-** 1 has been encoded. The RPL protocol uses DAO to provide the chosen parents with unicast destination information. The most significant message type in RPL is DIO. It maintains the node's current rank and uses particular metrics, like distance or hop count, to find the optimal path through the base node. DIS is an additional message type. Nodes obtain the DIO messages via DIS. ACK is a form of acknowledgment message that nodes use to respond. The corresponding codes for these are 1, 2, 3, and 4.

RPL Control (DAO) => 1

RPL Control (DIS) => 2

RPL Control (DIO) => 3

ACK => 4

5.1.2. **Dataset Construction for Multi-Class Classification**

In this research, our objective is to develop a robust attack detection model that effectively scales from small-scale to large-scale IoT networks. Existing literature primarily relies on datasets generated from networks with 10 to 50 nodes. However, this does not accurately reflect real-world IoT environments, where networks often consist of hundreds or even thousands of nodes. This gap limits the generalization capability of previous models. This research aims to develop a generalized attack detection model suitable for a real-world diversified network capable of simultaneously identifying three attacks. In real-world IoT networks, cyber threats do not occur in isolation, but multiple attack types may coexist, leading to complex traffic patterns that require advanced detection mechanisms.

To construct a multi-class dataset, three separate IRAD datasets, each corresponding to a different attack type, were merged into a single dataset. Since all datasets shared identical feature representations, this integration was seamless and allowed for the creation of a more comprehensive and realistic IoT security dataset. The dataset includes four classes:

- Benign traffic: Normal network behavior without malicious activities.
- BH attacks: Malicious nodes drop packets by falsely advertising optimal routes.
- HF attacks: Attackers flood the network with excessive DIO, causing disruption.
- VN attacks: Attackers manipulate RPL version numbers, forcing unnecessary route

To build the dataset, proportional stratified sampling was applied to extract data from RPL-based IoT networks of varying sizes, specifically 10, 20, 100, and 1000 nodes, while preserving the natural distribution of the original data. A larger portion of the samples was drawn from the 1000-node network, which contained the highest volume of traffic data, ranging in the millions, followed by the 100-node, 20-node, and 10-node networks. This sampling approach ensured that the dataset captures a wide range of traffic behaviors across different network densities and scales. Despite the unequal data volume across network sizes, the final dataset was constructed with a balanced class distribution. It contains a total of 2.2 million records, with 550,000 samples for each class: one benign and three attack types, such as BH, HF, and VN. This class-balanced and scale-aware dataset design not only avoids bias but also enables the learning model to generalize effectively across various IoT environments. As a result, the dataset stands

out as one of the most comprehensive and diverse resources available for evaluating RPL-based attack detection systems.

Figure **5- 1a** illustrates the RPL attack classes distribution within in collected dataset**.** To further improve the dataset quality and eliminate noise, an outlier removal process was applied before model training. Given the high variance in IoT RPL traffic data, traditional outlier detection, such as the interquartile range outlier detection technique, was found to be too aggressive. They resulted in the loss of a significant amount of useful data. Instead, an isolation forest-based outlier detection technique was employed. This identified and removed outliers while retaining essential normal and attack patterns. This helped ensure that the dataset remained realistic and representative of actual RPL-based IoT behavior. During preprocessing, a total of 481,252 duplicate records were removed. Before outlier removal, the dataset contained 1,718,748 records. After applying the Isolation Forest method, 743,348 outliers were detected and removed. This resulted in a refined dataset with 975,400 records.

Figure **5- 1b** illustrates the class distribution after the removal of outliers. As a result, the class distribution was altered due to the fact that the outlier detection had more impact on some classes than others. To this end, a class balancing method was applied to address this issue. This method allowed for an equal representation of all four classes, thereby preventing the biasing of the model training while enhancing the generalization capability of the attack detection model. Consequently, the learning model has been able to detect multiple RPL-based IoT attacks at the same time. This method drops a major limitation of existing work that has been limited to detecting one attack at a time.

Table 5- 2:Dataset creation from different network sizes

Class	Network Sizes				
	10 Nodes	20 Nodes	100 Nodes	1000 Nodes	Total Samples
Benign Traffic	70,000	80,000	100000	300,000	550,000
BH Attack	70,000	80,000	100000	300,000	550,000
HF Attack	70,000	80,000	100000	300,000	550,000
VN Attack	70,000	80,000	100000	300,000	550,000
Total Dataset per Network Size	290,000	320,000	390,000	1,000,000	2,200,000

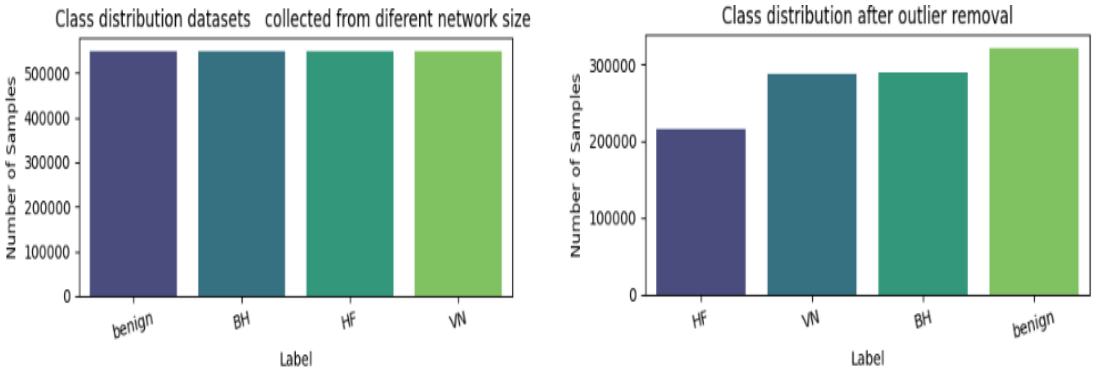

Figure 5- 1:Dataset class distribution before and after outlier removal

5.1.3. **Dataset Preprocessing**

The below code section presents the necessary libraries used for data preprocessing, preparing the federated dataset, and implementing the deep learning models.

```
import pandas as pd
import numpy as np
import collections
import random
import matplotlib.pyplot as plt
import tensorflow_federated as tff
from collections import OrderedDict
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import IsolationForest
import tensorflow as tf
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense, Conv1D, Flatten
from tensorflow.keras.layers import Dropout, MaxPooling1D, LayerNormal
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report, confusion_matrix
import seaborn as sns
import nest_asyncio
```

Figure 5- 2: Necessary libraries for data preprocessing and model training

5.1.3.1. **Data Balancing Using Under-sampling**

In non-synthetic data, the skewness of class labels can cause models to excessively favor the majority class and underrepresent the minority class (Bedi et al., 2020). After outlier and duplicate removal, class imbalance was experienced as shown in Figure **5- 1**. To solve this problem, we applied under-sampling technique, which is used to reduce the number of instances in the majority class, to keep the balance of data better. On the opposite of oversampling techniques such as the synthetic minority over-sampling technique (SMOTE), where synthetic data points are fabricated, under-sampling takes advantage of the actual dataset and does not require the generation of synthetic examples. Under sampling is a big plus in our case because our dataset is large enough so that the model can train on more representative real data and become more generalized. This technique ensures that the model is trained on real data rather than synthetic data so that it is less likely to overfit, and accordingly, its performance is enhanced on unseen data.

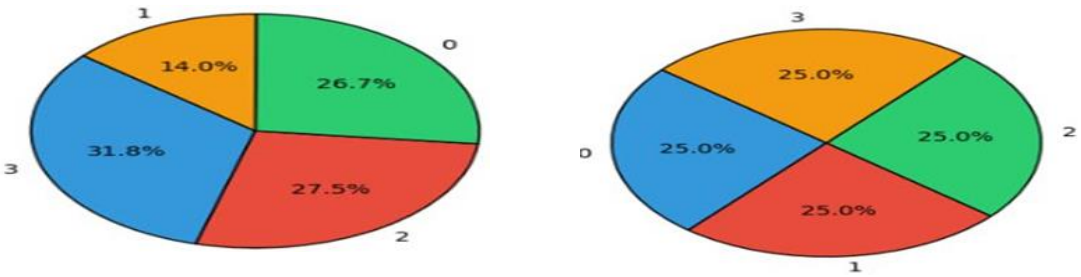

Figure 5- 3:Class distribution before balancing and after balancing

5.1.3.2. **Label Encoding**

The "Label" column, a categorical target variable in our dataset, indicated whether a network instance was an RPL attack or benign. We used Label Encoding to translate these categorical values into numerical representations since machine learning algorithms need numerical input. In particular, HF was encoded as 1, VN as 2, benign as 3, and the BH class as 0. This change preserved the unique features of each class while guaranteeing that the model could efficiently extract patterns from the data.

5.1.3.3. **Dataset Normalization**

Data values within [0, 1] range were linearly rescaled based on the feature normalization, which is the standard preprocessing technique. Reduction of bias, enhancement of detection

precision, and reliable model training and execution are all based on this processing. Precisely, the selected features were min-max scaled which gives the best results. The normalized feature value was computed as follows:

$$X* = \frac{RPL.FEATURE - RPL.FEATUREmin}{RPL.FEATUREmax - RPL.FEATUREmin} \qquad 5\text{-}5$$

*X** is the new feature value from 0 to 1, *RPL.FEATURES* is the original feature value and $RPL.FEATURE_{min}$ and $RPL.FEATURE_{max}$ are the maximum and minimum values of the selected features.

5.1.4. **Feature Selection and Importance Analysis**

We used feature selection methods, such as correlation analysis, Random Forest importance, and XGBoost importance rankings, to increase the model's interpretability and effectiveness. The goal was to find the most pertinent and discriminative characteristics for identifying RPL attacks.

5.1.4.1. **Pearson's Correlation Filtering**

We first used Pearson's correlation analysis to reject features highly associated with each other with a correlation threshold greater than 0.90. Since highly correlated features might introduce noise and redundancy, this stage ensures that only independent and relevant features are retained. We enhance model interpretability and reduce the possibility of overfitting in RPL attack detection by decreasing multicollinearity.

5.1.4.2. **Random Forest Feature Importance**

After removing redundant features, we used Random Forest to evaluate the significance of each remaining feature. RF measures the feature's importance by examining how much each feature helps to reduce classification uncertainty, specifically using Gini impurity. A key advantage of RF is its ability to identify nonlinear interactions.

5.1.4.3. **XGBoost Feature Importance**

In addition to the RF findings, we employed XGBoost, a gradient-boosting technique recognized for its iterative enhancement of weak learners. Unlike RF, which constructs multiple trees concurrently, XGBoost sequentially generates trees, with each new tree aiming to rectify

the errors made by its predecessors. This methodology proves particularly effective in identifying subtle or complex attack patterns.

5.1.4.4. **Hybrid Feature Selection**

To take advantage of both algorithms, we chose a hybrid feature selection technique. Features consistently good in both Random Forest and XGBoost were chosen for the final model. With this selection methodology, only the most dependable, generalizable, and attack-sensitive features have remained. Random Forest leveraged ensemble-based robustness-wise and robustness against overfitting. XGBoost enhanced the feature selection accuracy because it can capture small interactions and eliminate the biases in classification through boosting iterations. The combination of these techniques led us to build a tuned subset of features that is both discriminative whilst being suited for real-time detection in RPL-based IoT scenarios.

```
correlation_matrix = X.corr().abs()
upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape), k=1).astype(bool))
high_corr_features = [column for column in upper_triangle.columns if any(upper_triangle[column] > 0.90)]
X_filtered = X.drop(columns=high_corr_features)
# Step 2: Feature Selection using Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_filtered, y)
rf_importances = pd.Series(rf.feature_importances_, index=X_filtered.columns)
rf_sorted = rf_importances.sort_values(ascending=False)
rf_selected_features = rf_sorted.nlargest(13)
# Step 3: Feature Selection using XGBoost
xgb = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42)
xgb.fit(X_filtered, y)
xgb_importances = pd.Series(xgb.feature_importances_, index=X_filtered.columns)
xgb_sorted = xgb_importances.sort_values(ascending=False)
xgb_selected_features = xgb_sorted.nlargest(13)
```

Figure 5- 4:Feature selection mechanisms

5.1.5. **Dataset Preparation for Federated Learning**

To support a robust and scalable federated learning setup, we utilized TFF version 0.87.0, which offers powerful functions tailored for modern decentralized machine learning workflows. One of the most critical aspects of federated learning is the balanced distribution of data across clients. In our approach, samples were assigned to clients while ensuring class balance by shuffling each label's instances and allocating them evenly. This strategy maintains fairness across the learning environment, avoiding data skew that could bias the model. We employed a total of eight clients, six for training, one for validation, and one for testing. This setup allowed

for strict separation between the training and evaluation phases. Labels were assigned such that client_0 through client_5 handled training, client_6 handled validation, and client_7 was designated for testing. The dataset was split using a 70:15:15 ratio, with stratified sampling applied to maintain the label distribution consistency across subsets.

To prepare the data for efficient training, we addressed potential performance bottlenecks caused by mixed data types. All feature columns were converted to float32, and the labels were cast to int32, ensuring compatibility with TFF operations and improved computational efficiency. TFF's *create_tf_dataset_for_client_fn* function was central to organizing client-specific datasets. This utility:

- Retrieves the dataset associated with a particular client,
- Formats the data into dictionaries containing feature sequences and their labels,
- Converts the data into *tf.data.Dataset* objects for seamless integration with the TFF.

To capture the temporal dynamics of RPL network traffic, we structured the input data into sequences of Five-time steps for each client. This approach is essential for time-series-aware models like CNN-GRU, CNN-LSTM, GRU, and LSTM, which rely on sequential input to learn temporal dependencies and evolving patterns in data(Ashraf Zargar, 2021). The CNN layers extract local features, while the GRU or LSTM layers model how these features change over time, crucial for detecting progressive attack behaviors such as BH, VN, and HF. Without this sequential structure, the models would lack the context needed to differentiate between normal and malicious patterns effectively.

Beyond dataset construction, we implemented a detailed preprocessing pipeline, as shown in Figure **5- 5**. Round-robin assignment manner was used to distribute equal labels across all federated clients, significantly improving the federated training(Learning et al., n.d.). Batching was used with a batch size of 64, allowing for efficient memory utilization and parallel computation. Prefetching with 5 was also used to minimize data loading latency and keep hardware resources consistently engaged.

```
clients#Round-robin labels assignment technique for federated clintes
for label in train_df["Label"].unique():
    label_train_df = train_df[train_df["Label"] == label].sample(frac=1, random_state=42)
    train_client_assignments = ["client_{}".format(i % num_train_clients) for i in range(len(label_train_df))]
    train_df.loc[train_df["Label"] == label, "client"] = train_client_assignments
for label in val_df["Label"].unique():
    label_val_df = val_df[val_df["Label"] == label].sample(frac=1, random_state=42)
    val_client_assignments = ["client_{}".format(i % num_val_clients + num_train_clients) for i in range(len(label_val_df))]
    val_df.loc[val_df["Label"] == label, "client"] = val_client_assignments
for label in test_df["Label"].unique():
    label_test_df = test_df[test_df["Label"] == label].sample(frac=1, random_state=42)
    test_client_assignments = ["client_{}".format(i % num_test_clients + num_train_clients + num_val_clients)
                               for i in range(len(label_test_df))]
    test_df.loc[test_df["Label"] == label, "client"] = test_client_assignments
#  Create Federated Training, Validation, and Testing Data
train_data = tff.simulation.datasets.ClientData.from_clients_and_tf_fn(
    client_ids=train_client_ids,
    serializable_dataset_fn=lambda c_id: create_tf_dataset_for_client_fn(c_id, train_df))
val_data = tff.simulation.datasets.ClientData.from_clients_and_tf_fn(
    client_ids=val_client_ids,
    serializable_dataset_fn=lambda c_id: create_tf_dataset_for_client_fn(c_id, val_df))
test_data = tff.simulation.datasets.ClientData.from_clients_and_tf_fn(
    client_ids=test_client_ids,
    serializable_dataset_fn=lambda c_id: create_tf_dataset_for_client_fn(c_id, test_df))
```

Figure 5- 5:Preparation dataset for federating clients

As shown in Figure **5.6**, the formatting step used a function *batch_format_fn*, which reshaped input sequences into structured tensors while maintaining appropriate label formats. The function *make_federated_data* was then used to apply preprocessing to each client's data individually, employing *tqdm* for progress tracking. This process resulted in a streamlined and modular dataset pipeline tailored for federated learning. The result was an optimized, reproducible, and scalable federated data preprocessing strategy that facilitated effective CNN-GRU model training in the context of RPL attack detection in IoT networks.

```
NUM_EPOCHS = 3,BATCH_SIZE = 64, PREFETCH_BUFFER = 5
def preprocess(dataset):
    def batch_format_fn(element):
        return collections.OrderedDict(
            x=tf.reshape(element['features'], [-1, SEQUENCE_LENGTH, len(usable_features)]),
            y=tf.reshape(element['label'], [-1]))
    return (dataset
            .repeat(NUM_EPOCHS)
            .batch(BATCH_SIZE)
            .map(batch_format_fn)
            .prefetch(PREFETCH_BUFFER))
def make_federated_data(client_data, client_ids):
    return [preprocess(client_data.create_tf_dataset_for_client(client_id))
            for client_id in tqdm(client_ids)]
# Making federated data for all training clients
NUM_CLIENTS = len(np.unique(train_df[client_id_colname]))
sample_clients = train_data.client_ids[:NUM_CLIENTS]
federated_train_data = make_federated_data(train_data, sample_clients)
federated_val_data = make_federated_data(val_data, val_client_ids)
```

Figure 5- 6: Making federated data to the clients

5.1.5.1. **Balanced Dataset Distribution Across** Federated **Clients**

In Figure **5- 7**, we observed an imbalanced distribution of labels across federated clients before we applied any balanced technique, which can negatively impact the model's ability to generalize effectively. This imbalance occurs because some clients receive a disproportionate number of certain attack types while others have a limited representation of specific labels. In federated learning, such an uneven distribution can lead to a biased global model that learns patterns favoring majority classes while struggling to detect minority-class attacks. Consequently, the model's performance may degrade, especially in real-world scenarios where attack patterns are highly dynamic. This imbalance reduces the robustness of the detection system, making it less effective in identifying diverse RPL attack types across different network environments.

To address this issue, we have distributed the class label equally for federated clients. By distributing the dataset in this manner, each client receives a proportional and representative mix of attack and benign traffic data, preventing bias in the training process. This balanced allocation significantly enhances the model's generalization ability across all attack types, improving detection accuracy and robustness (Younis & Fisichella, 2022).

Figure **5.8** presents a balanced distribution of labels across federated clients, achieved through a combination of strategic client assignments.

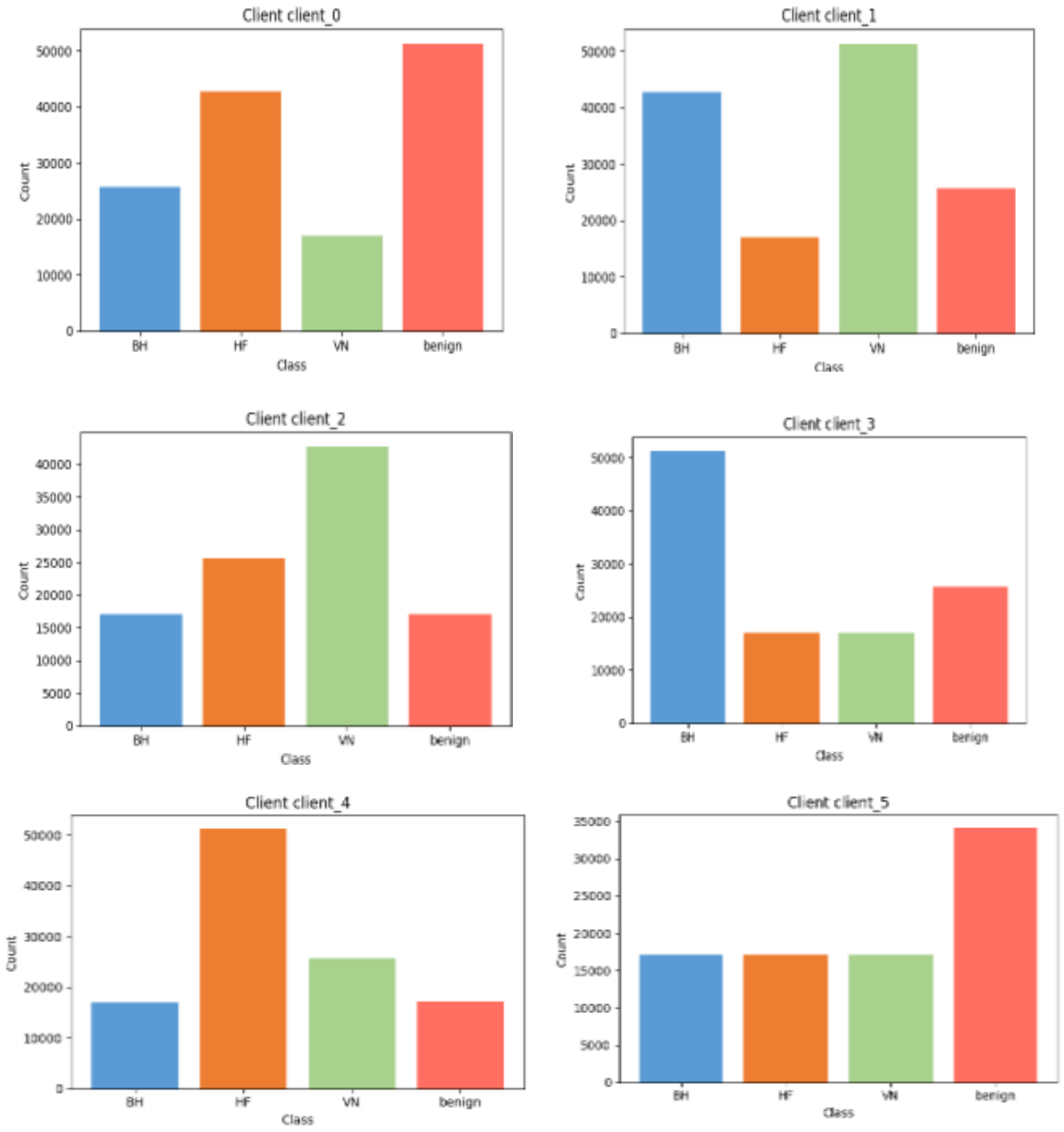

Figure 5- 7:Imbalanced class distribution across federated clients

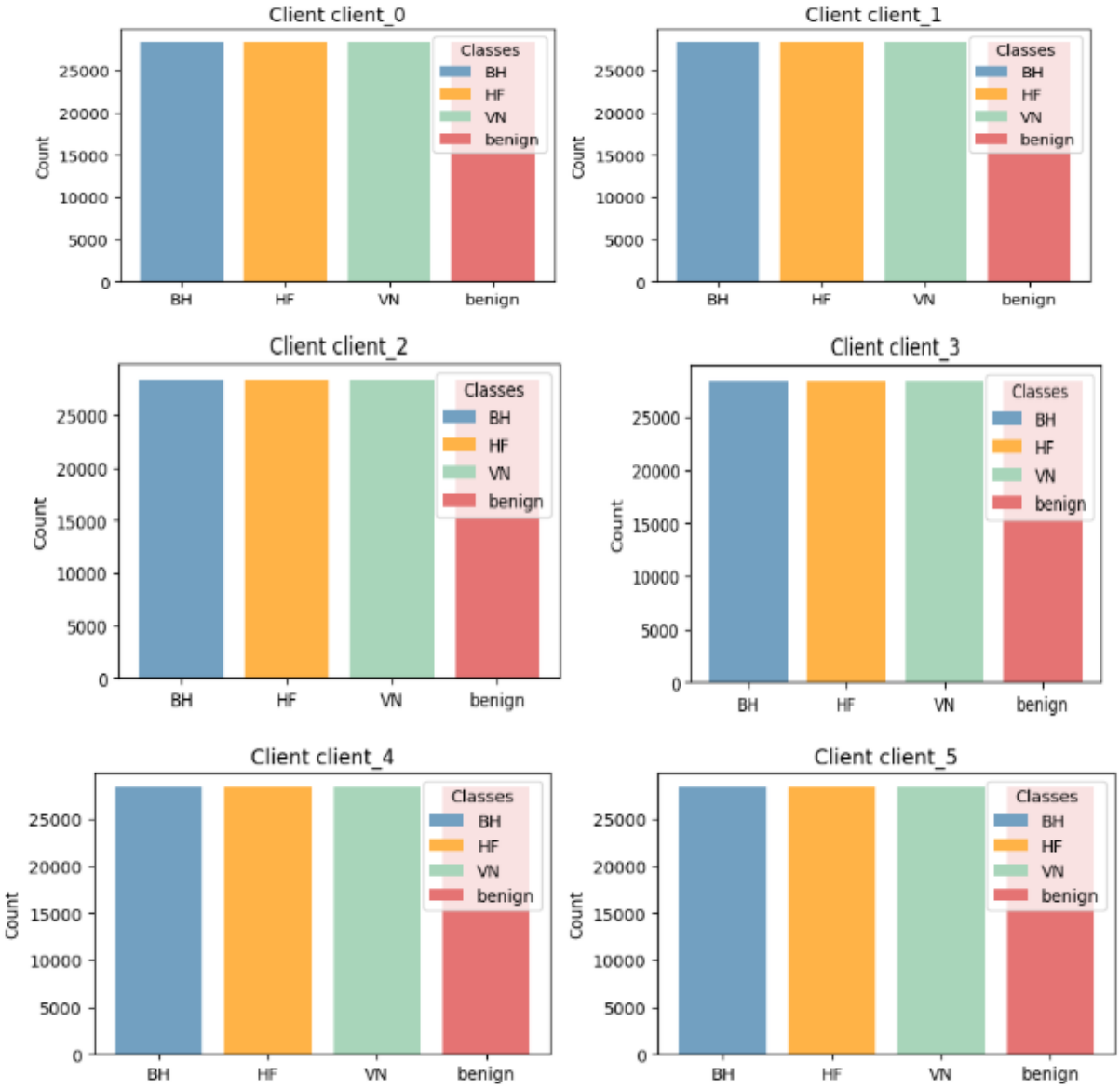

Figure 5- 8:Balanced class distribution across federated clients

5.1.6. **Proposed Model Implementation in Federating Learning Framework**

In this section, we detailed the implementation phase of our research, focusing on developing various deep learning models within a federated learning framework for the detection and classification of RPL attacks. This encompassed the selection and integration of essential software libraries, the meticulous configuration of optimization parameters necessary for effective model training, and the implementation of the designed architecture for the detection and classification framework within the federated learning paradigm. We implemented and

evaluated a range of models, including CNN combined with GRU (CNN-GRU), CNN combined with LSTM (CNN-LSTM), standalone GRU, and standalone LSTM.

5.1.6.1. **Federated CNN-GRU model Implementation for RPL Attack Detection**

The implementation of the federated hybrid CNN-GRU model for the detection and classification of RPL attacks in 6LoWPAN networks is presented in this section. As shown in Figure **4- 2** and Figure **4- 3**, the architectural analysis described in Chapter 4, we explored two primary architectures for our experiment. Considering the benefits of adaptable architectures for handling spatial and temporal data, we created both a Functional API model and a Sequential model each was trained in federating learning, acknowledging that each possesses its distinct advantages and drawbacks. Our evaluation, utilizing the preprocessed IRAD dataset, indicated that the sequential model surpassed the functional API approach, resulting in its choice for final implementation. Consequently, in this thesis report the specifics of the Functional API implementation are not presented further. The rest of this chapter concentrated on the implementation and assessment of the chosen Sequential model.

The input layer received the preprocessed constructed IRAD data consisting of 14 selected features that characterize network behavior. The inputs are first processed by convolutional layers extracting spatial or local patterns from the input features at each time step. The output of the CNN is then passed to the GRU, which handles temporal dependencies, meaning it looks at how these CNN-extracted features change over time. This is important because many RPL attacks exhibit sequential patterns (like a gradual increase in DIS messages or consistent drops in packet reception). The extracted representations are then passed through fully connected layers to perform the final classification, producing predictions across the target attack classes. There are two 1D convolutional layers in the CNN branch. To maintain spatial dimensions, the initial Conv1D layer used padding and made use of 128 filters with a kernel size of 3 and max-pooling with a size of 2 to reduce the size of the dimensions. Batch normalization comes next to speed up convergence and stabilize training. ReLU serves as the activation function, adding non-linearity and guaranteeing effective learning. Using 64 filters with a kernel size of 3, the second convolutional layer improves feature extraction. It is then subjected to additional Layer normalization. The initial GRU layer, which contains 128 units and is set with return_sequences=True, allows the network to maintain sequential dependencies throughout

the time steps. The subsequent GRU layer, comprising 64 units and configured with return_sequences=False, condenses the temporal features into a more compact form. This architectural setup enables the model to effectively capture evolving attack trends over time, thus improving classification accuracy.

At the time of classification, the output of the GRU layers is given to fully connected layers. To prevent overfitting, a dropout rate of 0.2 is used after the first dense layer, which contains 128 neurons with ReLU activation. The learned feature representations are further enhanced with another dense layer with 64 neurons, followed by another dropout layer with a rate of 0.1. The last output layer has a SoftMax activation function to produce class probabilities and consists of neurons for the four attack types, i.e., benign, BH, HF, and VN.

The model is trained in federated learning with the federated algorithm to facilitate privacy-preserving during training. Sensitive network data is preserved when local client models learn separately from their datasets and only share their learned parameters with a global model rather than centralizing the data. Adam optimization was used for client-side optimization, which provides stability and avoids overfitting. Adam optimization algorithm was also implemented on the server side to combine the local updates for effective convergence.

TensorFlow Federated serves as the platform to execute the federated training process, where the local devices separately train their models and transmit the parameters to the aggregator. Federated training is executed for 100 rounds, where the local models progressively update the parameters and refine a global model, thereby creating ongoing improvements in the model's performance. The initial training phase begins with the recording of the baseline performance metrics. Then, the model undergoes a series of 100 federated update steps, where local client models are trained on different data subsets for epochs 3 before their learned parameters are aggregated at the central server. The iterative process enables the model to constantly enhance its detection of RPL attacks while learning to adapt to different network conditions.

The FedAvg algorithm works to move the state of the global model forward in every iteration in a way that makes the model stronger and generalizes better with the collective knowledge from all nodes involved. In every round, the training progress is monitored and performance improvement is assessed by logging metrics such as accuracy and loss. In a federated setting,

the iterative process is continued until the model achieves optimal convergence, effectively balancing detection accuracy.

Table 5- 3:Summary of hyperparameters used in federated CNN-GRU model

Category	Hyperparameter	Optimized	Other Values tried
Model Architecture	Model Type	2 layer each (CNN-GRU)	1 layer each (CNN-GRU)
	Conv1D filters	128 → 64	64 → 32
	GRU units	128 → 64	64 → 32
	Dense layers	128 → 64	64 → 32
	Dropout rate	0.2→0.1	0.1, 0.3, 0.15
	Output layer	4	
Federated Training	Training rounds	100	30,80
	Epochs for local training	3	1,3, 5
	Training clients	6	6,8 10
	Batch size	64	32, 128
	Client optimizer	Adam (LR=0.001)	SGD with momentum
	Server optimizer	Adam (LR=0.001	SGD with momentum
Training	Loss function	Sparse Categoric Cross entropy	
	Evaluation metric	SparseCategoricalAccuracy	

```
def create_keras_model():
    clf = tf.keras.Sequential([
        layers.Conv1D(128, 3, activation='relu', padding='same', input_shape=(SEQUENCE_LENGTH, len(features) - 1)),
        layers.Conv1D(64, 3, activation='relu', padding='same'),
        layers.LayerNormalization(),
        layers.MaxPooling1D(2),
        layers.GRU(128, return_sequences=True),
        layers.GRU(64),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(64, activation='relu'),
        layers.Dropout(0.1),
        layers.Dense(4, activation='softmax')])return clf
keras_model = create_keras_model()
def model_fn():
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
        keras_model, input_spec=preprocessed_example_dataset.element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
def adam_with_l2(learning_rate):
  return tff.learning.optimizers.build_adam(learning_rate=learning_rat)
iterative_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn=model_fn,
    client_optimizer_fn=adam_with_l2(learning_rate=0.001
  server_optimizer_fn=adam_with_l2(learning_rate=0.001)
state = iterative_process.initialize()
NUM_ROUNDS = 100
train_metrics_list = []
val_metrics_list = []
for round_num in range(1, NUM_ROUNDS):
    state, train_metrics = iterative_process.next(state, federated_train_data)
    val_metrics = iterative_process.next(state, federated_val_data)[1]
    print(f'round {round_num:2d}, train_metrics={train_metrics}, val_metrics={val_metrics}')
```

Figure 5- 9::Federating CNN-GRU model sample code

5.1.6.2. **Federated CNN-LSTM Model Implementation for RPL Attack Detection**

To detect and categorize RPL attacks in 6LoWPAN networks, we have implemented a federated hybrid CNN-LSTM model in this study. The model can identify complex patterns in network traffic, which combines the advantages of CNNs for extracting spatial features with LSTMs for simulating temporal correlations. The model was developed using the Keras Sequential API, which allows for a straightforward and organized arrangement of layers. The preprocessed IRAD RPL traffic, which comprises 14 selected features that depict network behavior, is accepted by the input layer at the start of the model. The model sequentially processes input data through convolutional layers to extract spatial characteristics, followed by LSTM to model temporal dependencies. The learned features are then fused and fed to fully connected layers, culminating in the ultimate classification output. To preserve spatial dimensions, the model begins with a 1D convolutional layer comprising 128 filters, kernel 3, and "same" padding,

followed by ReLU activation. Layer normalization is then applied for improved training stability and convergence acceleration. Then, one more 1D convolutional layer is applied with 64 filters and ReLU activation, again followed by batch normalization. Finally, a max-pooling layer with pool size 2 is applied to reduce dimension while preserving the most important features extracted.

After the convolutional layers extract features, the output is fed into a series of LSTM layers to capture temporal relationships. The initial LSTM layer with 128 units and return_sequences=True maintains the temporal structure across the various time steps. This is followed by a second LSTM layer with 64 units and return_sequences=False that compacts the sequential data into a dense representation that works optimally for classification. A dropout of 0.2 is applied after the dense layers to enhance generalization and prevent overfitting. Following the extraction of temporal features, the output is passed on to fully connected layers. The first dense layer consists of 128 neurons with ReLU activation. Then, a second dense layer with 64 neurons is added, along with a dropout layer at a rate of 0.1 to prevent overfitting. Then, a dense output layer with four neurons is applied with SoftMax activation to provide probability distributions across the four target classes of benign, BH, HF, and VN.

TFF was utilized to train this hybrid model in a federated learning environment. By employing this method, sensitive information is kept confidential while local client models are trained on their private data and share their parameters with the global model. The server-side optimizer employs Adam for effective model aggregation, the client models also are trained using the Adam optimizer to improve training stability. The model can learn from decentralized data while preserving privacy. Each of the 100 rounds that comprise the federated training process represents an iteration of model updates from several clients. Local models were trained for 3 epochs per round and only exchanged their learned parameters with the central server after being trained on their respective datasets in each round. By merging these updates and applying them to the global model, the server gradually improves the model's ability to identify RPL threats.

```
def create_keras_model():
    clf = tf.keras.Sequential([
        layers.Conv1D(128, 3, activation='relu', padding='same', input_shape=(SEQUENCE_LENGTH, len(features) - 1)),
        layers.Conv1D(64, 3, activation='relu', padding='same'),
        layers.LayerNormalization(),
        layers.MaxPooling1D(2),
        layers.LSTM(128, return_sequences=True),
        layers.LSTM(64),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(64, activation='relu'),
        layers.Dropout(0.1),
        layers.Dense(4, activation='softmax')])return clf
keras_model = create_keras_model()
def model_fn():
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
        keras_model, input_spec=preprocessed_example_dataset.element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
def adam_with_l2(learning_rate=0.001):
  return tff.learning.optimizers.build_adam(learning_rate=learning_rat)
iterative_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn=model_fn,
    client_optimizer_fn=adam_with_l2(learning_rate=0.001
  server_optimizer_fn=adam_with_l2(learning_rate=0.001)
state = iterative_process.initialize()
NUM_ROUNDS = 100
train_metrics_list = []
val_metrics_list = []
for round_num in range(1, NUM_ROUNDS):
    state, train_metrics = iterative_process.next(state, federated_train_data)
    val_metrics = iterative_process.next(state, federated_val_data)[1]
    print(f'round {round_num:2d}, train_metrics={train_metrics}, val_metrics={val_metrics}')
```

Figure 5- 10: Sample code of federated CNN-LSTM model implementation of

5.1.6.3. **Federated LSTM Model Implementation for RPL Attack Detection**

The preprocessed IRAD, which has 14 selected features, is first accepted by the model through an input layer, after which the data is sent to an LSTM branch. Several fully linked layers are subsequently applied to the output of the LSTM layers. A dropout layer of 0.3 for regularization comes after the first dense layer, which has 128 neurons with ReLU activation. To classify the network instances into one of the four attack classes, such as benign, BH, HF, and VN. A softmax activation function is used before the final output layer, which is followed by a second dense layer with 64 neurons and an extra dropout layer of 0.2.

The LSTM model implementation below was trained within the same federated learning setup as CNN-GRU or CNN-LSTM was trained, using identical hyperparameters, the same number of training clients, and the same number of training rounds

```
def create_keras_model():
    clf = tf.keras.Sequential([
        layers.LSTM(128, return_sequences=True,input_shape=(SEQUENCE_LENGTH, len(features) - 1)),
        layers.LSTM(64),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(64, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(4, activation='softmax')
    ])
    return clf
# Create model instance
keras_model = create_keras_model()
def model_fn():
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
        keras_model,
        input_spec=preprocessed_example_dataset.element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
```

Figure 5- 11: Federated LSTM model implementation sample code

5.1.6.4. **Federated GRU Model Implementation for RPL Attack Detection**

The model begins with an input layer that receives the preprocessed RPL network features. The input sequence is then processed through GRU layers, efficiently capturing sequential dependencies in network traffic data. The first GRU layer consists of 128 units and processes the input sequence while maintaining its temporal structure. This layer returns sequences=True so that the entire sequence is forwarded to the next layer for further feature extraction. The second GRU layer, composed of 64 units and returning sequences=False, condenses the temporal features learned in a compact form for classification.

GRU output is then passed through dense or fully connected layers for ultimate classification. The first dense layer consists of 128 neurons with ReLU activation, followed by a second dense

layer with 64 neurons. Dropout of 0.3 and 0.2 after the dense layer was introduced to prevent overfitting. The output layer contains 4 neurons, which correspond to the four classes such as benign, BH, HF, and VN, and uses a SoftMax activation function to generate probability scores.

The GRU model application presented hereinbelow was developed in conjunction with the federated learning platform used to train either the CNN-GRU or the CNN-LSTM model with the same hyperparameters, the same number of training clients, and the same number of training iterations.

```
def create_keras_model():
    clf = tf.keras.Sequential([
        layers.GRU(128, return_sequences=True,input_shape=(SEQUENCE_LENGTH, len(features) - 1)),
        layers.GRU(64),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(64, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(4, activation='softmax') ])
    return clf
# Create model instance
keras_model = create_keras_model()
def model_fn():
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
        keras_model,
        input_spec=preprocessed_example_dataset.element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])
```

Figure 5- 12:Federated GRU model implementation sample code

5.1.7. **Model Testing and Evaluation**

The performance was evaluated based on Holdout Cross-Validation, a default resampling method that evaluates model performance. The method splits the dataset into various training and testing datasets. Despite being widely applied in machine learning, Stratified K-Fold and K-Fold Cross-Validation methods are ineffective in our scenario because our dataset is excessively large. The two methods are not practical when applied to large datasets because they entail multiple training iterations, which can be time-consuming and computationally costly.

5.1.7.1. **Holdout Cross-Validation Implementation**

In this research, we used Holdout cross-validation, and we split the dataset into three equal parts: 70% for training, 15% for testing, and 15% for validation. This provides enough data to train with and also gives a ratio for testing and tweaking the model in order to adequately test the model's performance. The 15% test set is an unseen set used for final performance measurement, and the 70% training data is utilized in training the model. The 15% validation set is utilized in performance optimization via hyperparameter tuning when tuning the model. 42 was chosen as the fixed random state in order to promote reproducibility as well as reduce variance among runs. This method preserves a trustworthy assessment framework while permitting a single training loop, making it computationally efficient. Fair and insightful comparisons with previous RPL attack detection are made possible by the selected data split ratio.

By using Holdout CV, we evaluated performance and model efficiency without incurring the high computational costs associated with iterative techniques like K-Fold CV. K-Fold CV is not practicable for large datasets like ours because it would take a lot of training time and resources, especially considering the size of our dataset, which is 975,400 sample records. K-Fold CV would slow down the process and place a needless burden on computer resources by requiring numerous rounds of model training. We employ important performance metrics including accuracy, precision, recall, F1-score, and the confusion matrix to evaluate how well the model detects and classifies RPL attacks.

```
test_data_fed = make_federated_data(test_data, test_client_ids)
test_metrics = iterative_process.next(state, test_data_fed)[1]
print('Test metrics:', test_metrics)
model_weights = iterative_process.get_model_weights(state)
keras_model = create_keras_model()
# Assign the trained weights to the Keras model
keras_model.set_weights(model_weights.trainable) test_labels = []
test_predictions = []
test_predictions = []
test_labels = []
for client_id in tqdm(test_client_ids):
    client_dataset = test_data.create_tf_dataset_for_client(client_id).batch(64)
    for batch in client_dataset:
        features = batch['features'].numpy()
        features = features.squeeze(axis=1
        labels = batch['label'].numpy()
        predictions = keras_model(features, training=False)
        predicted_classes = np.argmax(predictions.numpy(), axis=1)
        test_labels.extend(labels)
        test_predictions.extend(predicted_classes)
```

Figure 5- 13:Evaluation of federated models on test data

CHAPTER SIX

6. RESULTS and DISCUSSION

Here, we present the results gained from the proposed models in RPL attack detection and classification. Experimented models include federated CNN-GRU, CNN-LSTM, GRU, and LSTM. Their performance was assessed to determine the most effective model for detecting and classifying RPL attacks. We examined feature importance scores and ranked them based on their statistical contribution to the independent variable.

6.1. Feature Relevance and Selection Outcomes

Feature selection is a critical aspect of deep learning, as not all features contribute equally to the model's performance. Identifying the most relevant features from IRAD is important to improve the accuracy and efficiency of the detection model. Correlation analysis was used first to assess the relationships between features and the target classes. Then, Random Forest and XGBoost techniques were applied to the results of selected features using the correlation analysis to rank and select the most important features. The final important features were obtained from the union of these two techniques.

6.1.1. Correlation Analysis

The results of Pearson's correlation analysis are presented below in Figure **6-1**. The analysis was performed on IRAD containing 18 features, with the remaining features being removed due to high correlations that could lead to multicollinearity. Only the selected features, based on their correlation scores were used for further processing. The x-axis represents the correlation score of each selected feature, while the y-axis represents the features. This emphasizes the relationships between the selected features and their importance in RPL.

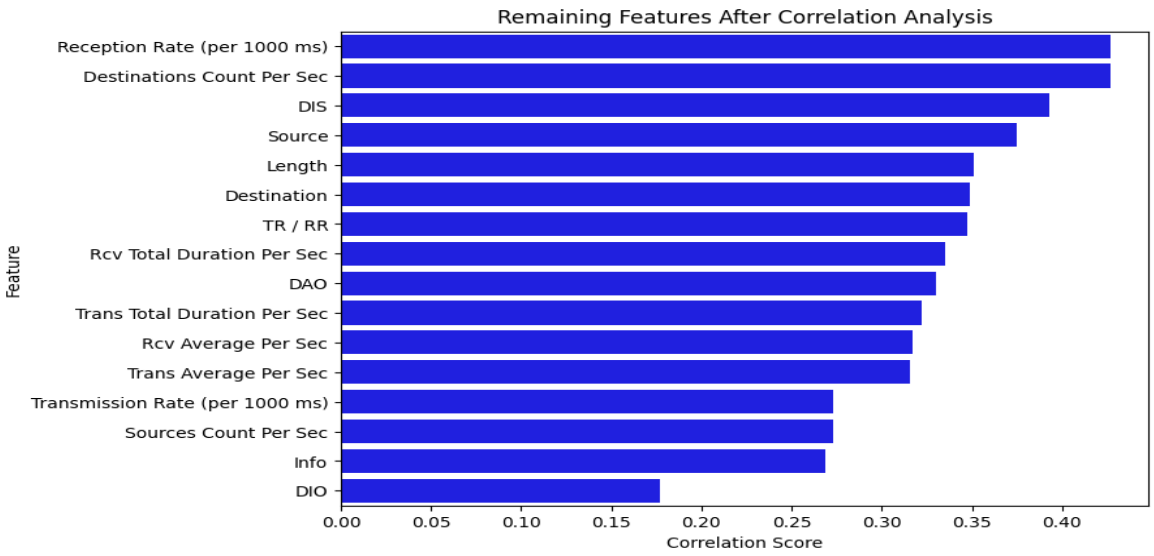

Figure 6- 1:Correlation analysis results in a bar chart

6.1.2. **Random Forest Feature Importance Selection**

Figure 6- 2 presents the features selected based on the results from the correlation analysis. Random Forest, is known for its ability to handle non-linear relationships between features and is robust against overfitting. As a result, Random Forest identified 13 key features that are most relevant for the RPL attack detection system.

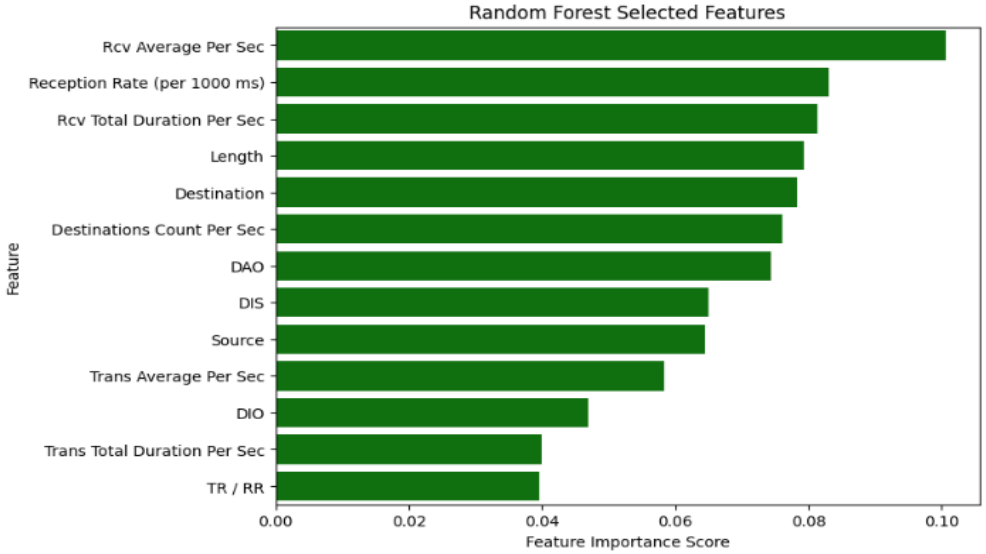

Figure 6- 2:Bar chart representation of random forest feature important

6.1.3. **XGBoost Feature Importance**

XGBoost was applied to select the most important features for model generalization, based on the results of the correlation analysis on the IRAD dataset. This technique is chosen due to its ability to handle complex relationships and provide high predictive accuracy. As a result, XGBoost identified important features that are critical for detecting RPL attacks.

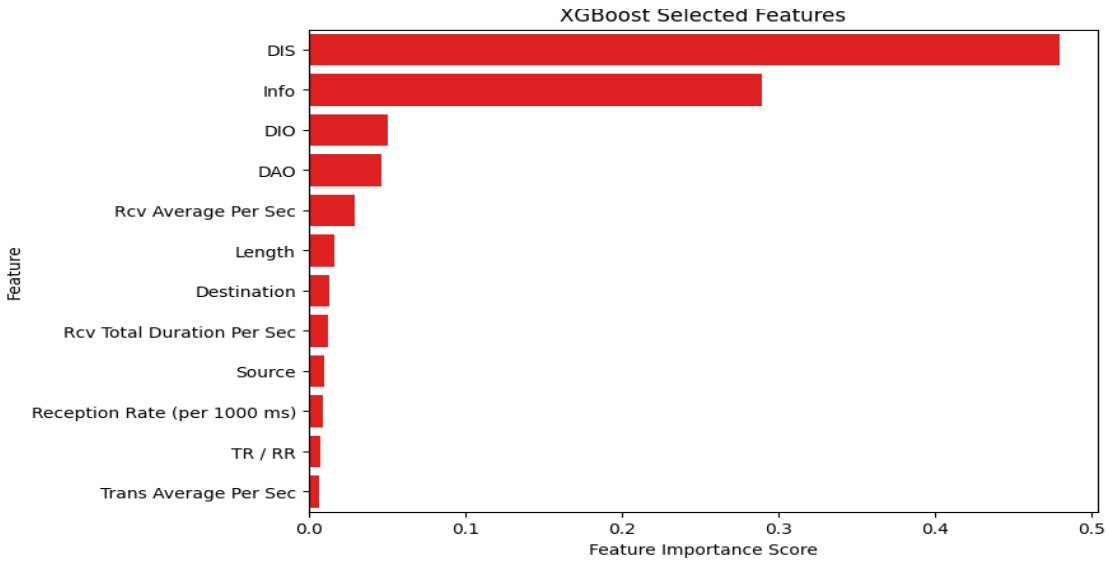

Figure 6- 3: Bar chart presentation of XGBoost feature selection score

6.1.4. **Hybrid Important Features Selection**

A hybrid feature selection approach was used that combines Random Forest and XGBoost to leverage the strength of each method when used individually. Random Forest is effective at capturing feature importance based on impurity reduction, but it struggles with features that have complex interactions. On the other hand, XGBoost is highly efficient in handling feature interactions but can sometimes overlook less dominant yet relevant features. The results of this hybrid feature selection method are presented in Figure **6- 4** with 14 important score features.

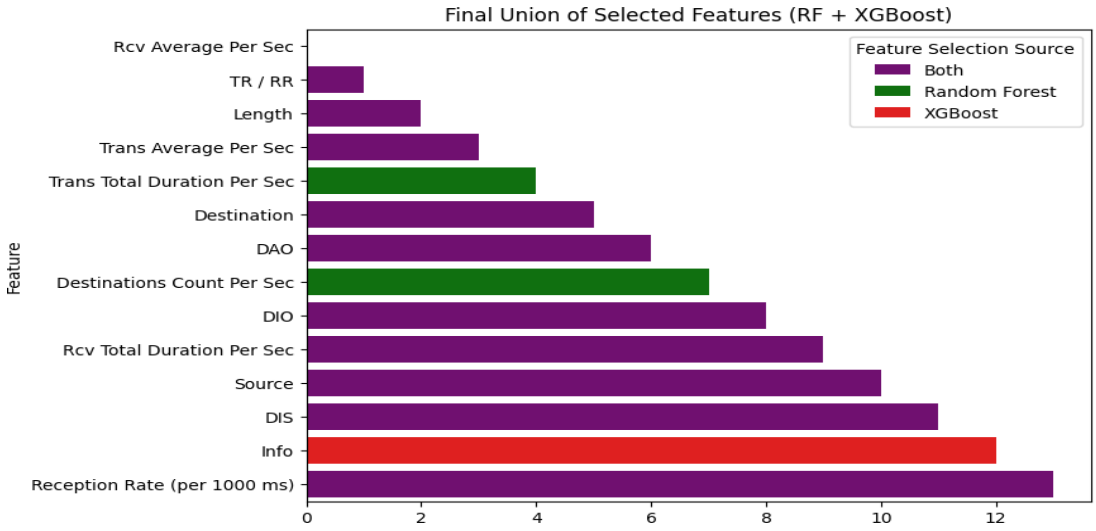

Figure 6- 4: Union of selected features score using XGBoost and Random Forest

6.2. **Results of the Proposed Federated Hybrid Deep Learning Models**

In this section, we presented and discussed the experimental results obtained from four different deep-learning models: LSTM, CNN-GRU, CNN-LSTM, and GRU. These models were implemented for the detection and classification of RPL attacks using the IRAD dataset. Each model was evaluated individually by feeding it with statistically selected features to recognize malicious activities within the network traffic data. The results of these experiments are analyzed in the following subsections, where we assess their effectiveness based on standard evaluation metrics. Finally, we determined the model that delivers optimal performance for the specified task based on these metrics.

6.2.1. **Experiment One: Federated CNN-GRU Model**

The training was conducted throughout 100 rounds, utilizing a hold-out data split of 70% for training, 15% for validation, and 15% for testing. An adaptive learning rate reduction technique was employed, which monitors the validation loss and decreases the learning rate whenever there is no noticeable improvement, promoting smoother convergence and assisting in the prevention of overfitting. The model's final assessment was conducted using a test dataset that included 146,310 samples. Figure **6- 5** demonstrates that the training process reveals a clear and consistent improvement in performance throughout the rounds. In the initial round, the model achieved a training accuracy of 71.51% and a loss of 0.5633, while the validation accuracy was noted at 73.15%, along with a validation loss of 0.5231. This early phase

illustrates the model's initial comprehension of the learning patterns within the dataset. As training continued, the model steadily progressed towards greater accuracy levels. Upon completing 100 communication rounds, the model attained an optimal training accuracy of 99.61% and a loss of 0.0101, while the validation accuracy stabilized at 99.49% with a minimal validation loss of 0.0197.

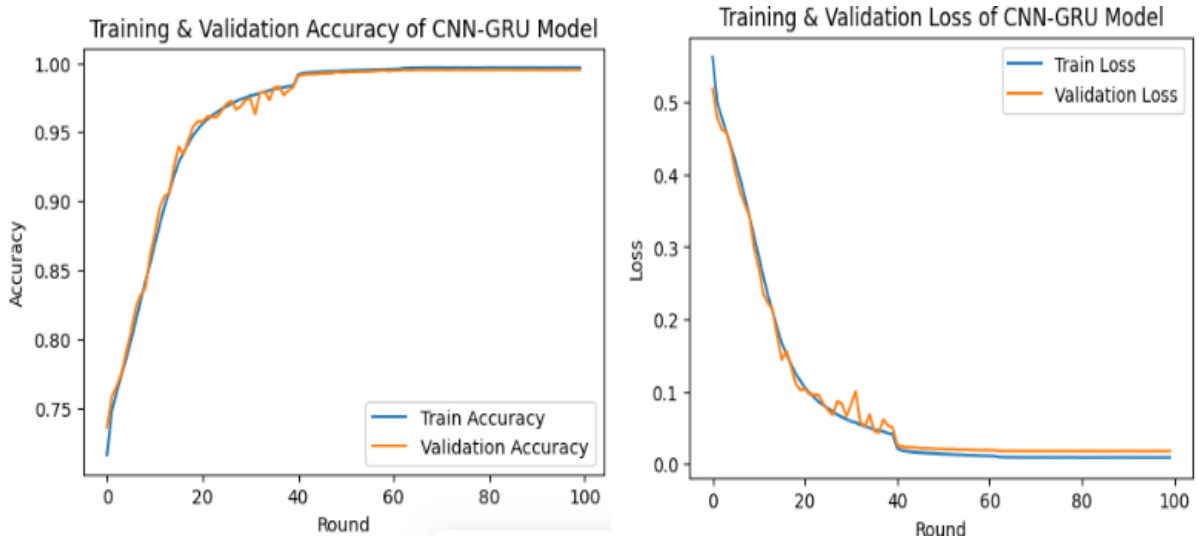

Figure 6- 5: Training and validation accuracy and loss of CNN-GRU model

Result and Discussion of the CNN-GRU Model

The federated CNN-GRU model showcased outstanding performance upon final evaluations, achieving an overall test accuracy of 99.50%. Every performance metric, including precision, recall, and F1-score, surpassed 99.50%, highlighting the model's robust capacity to identify anomalies in RPL-based IoT networks. The detailed performance metrics for each class are shown below in Table 6-1.

Table 6- 1: Class-wise performance evaluation of federated CNN-GRU model

Class	Precision (%)	Recall (%)	F1-Score (%)
BH Attack	99.40	98.90	99.15
HF Attack	99.93	99.97	99.95
VN Attack	99.50	99.62	99.56
Benign	99.16	99.50	99.33
Overall	99.50	99.50	99.50

Per-Class Performance Analysis

The CNN-GRU model demonstrated outstanding effectiveness in identifying HF attacks, achieving a precision of 99.93% and a recall of 99.97%, resulting in an almost perfect F1-score of 99.95%. HF attacks usually produce a notable spike in DIO messages, establishing a distinctive and easily recognizable signature. The model's sequential learning architecture is particularly skilled at detecting these temporal patterns, which enhances its accuracy. VN attacks were reliably identified with a precision of 99.50% and a recall of 99.62%, leading to an F1-score of 99.56%. This suggests that VN attacks possess a misleading nature. VN attacks subtly manipulate control message versioning in ways that blend with regular network behavior, making them more difficult to identify. As a result, the model may experience some challenges in detecting VN attacks compared to the more apparent behaviors shown by HF.

The detection of BH attacks achieved a precision of 99.40%, though the recall dropped to 98.90%, producing an F1-score of 99.15%, marginally lower than that of the other attacks. This is due to the BH attack's highly disruptive nature, as it aggressively drops all incoming packets, resulting in sudden and erratic traffic patterns that complicate the model's ability to learn distinguishing features reliably. Implementing advanced feature selection or adaptive classification thresholds could help address this problem. Instances of benign traffic achieved a precision of 99.16% and a recall of 99.50%, leading to an F1-score of 99.33%. The model's high recall about both HF and VN classes indicates a low rate of misclassifying normal traffic as malicious, which is essential for minimizing unnecessary alerts and ensuring operational stability. Nonetheless, further reducing the misclassification of attacks as benign could be achieved by focusing on deeper learning for borderline cases.

6.2.2. **Experiment Two: Federated CNN-LSTM Model**

Figure **6- 6** depicts that federated training was carried out over 100 rounds, demonstrating steady enhancements in both accuracy and loss reduction. The training process showed stable convergence, improving validation accuracy over time. In round 1, the model recorded a training accuracy of 70.64% and a validation accuracy of 73.73%, while the training and validation losses were noted at 0.5880 and 0.5149, respectively. These figures indicate the model's initial capacity to learn and extract spatial-temporal features from the data provided. As the training continued, the performance metrics saw significant advancement. In the final

round100, the model sustained its strong performance, achieving a training accuracy of 99.05%, validation accuracy of 99.20%, and very low loss values.

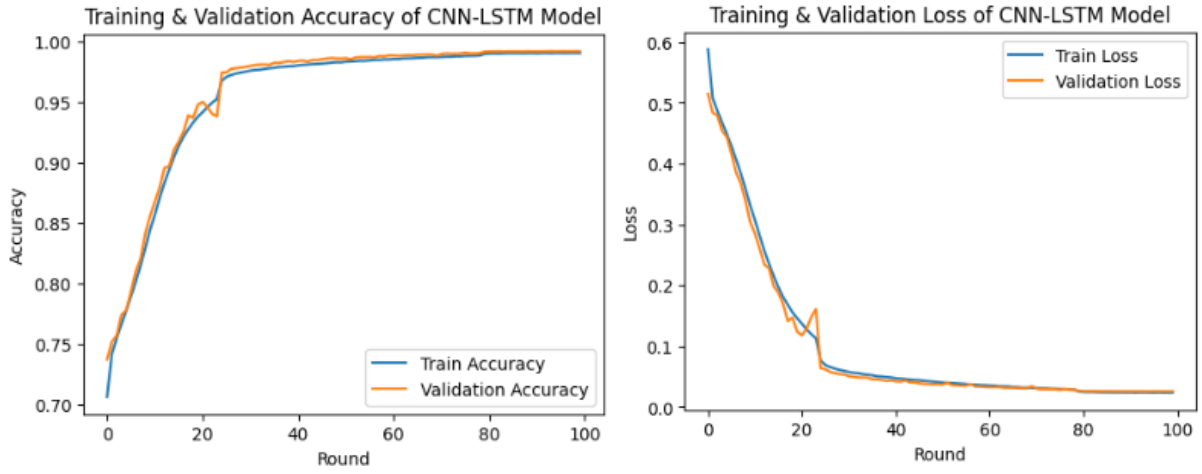

Figure 6- 6: Training and validation accuracy and loss of CNN-LSTM model

The CNN-LSTM model demonstrated outstanding classification performance, achieving a test accuracy of 99.17%. This high accuracy highlights the strength of combining CNN's spatial feature extraction with LSTM's temporal pattern recognition, effectively capturing complex behaviors in RPL-based IoT network traffic. Table 6-2 summarizes the overall precision, recall, and F1-score stood at 99.17%, 99.17%, and 99.17%, respectively. These results indicate that the model is highly capable of identifying various classes with minimal misclassifications, making it an effective solution for anomaly detection in distributed IoT environments.

Table 6- 2: Class-wise performance evaluation of federated CNN-LSTM model

Class	Precision (%)	Recall (%)	F1-Score (%)
BH Attack	99.15	98.33	98.74
HF Attack	99.76	99.91	99.84
VN Attack	99.09	99.22	99.16
Benign	98.68	99.22	98.95
Overall	99.17	99.17	99.17

Per-Class Performance Analysis

The model exhibited almost flawless classification next to the CNN-GRU model for HF attacks, achieving a precision of 99.76% and a recall of 99.91%, which resulted in an F1-score of 99.84%. This indicates that nearly all HF instances were accurately detected, highlighting the model's proficient temporal learning from the burst patterns of DIO messages that define HF attacks. The detection of VN attacks was also promising, with a precision of 99.09% and a recall of 99.22%, leading to an F1-score of 99.16%. This validates the model's capability to recognize the subtle behaviors linked to modifications in version numbers, illustrating the efficiency of the CNN-LSTM architecture in identifying subtle anomalies. In the case of BH attacks, the detection demonstrated a precision of 99.15% and a recall of 98.33%, finishing in an F1-score of 98.74%. Although the precision reflects a minimal number of false positives, the slightly lower recall suggests that some BH attack instances may have been overlooked or incorrectly classified, most likely due to their deceptive resemblance to benign traffic. Nonetheless, the model still achieved commendable performance in this category. The model sustained a high precision of 98.68% and a recall of 99.22%, resulting in an F1-score of 98.95% for benign traffic instances. The high recall next to HF and VN classes demonstrated the model's efficiency in identifying normal behavior, even though the precision indicates that a few attack instances were incorrectly classified as benign. This balance is often acceptable in intrusion detection contexts, where false negatives tend to be more critical than false positives.

6.2.3. **Experiment Three: Federated GRU Model**

This study assessed the effectiveness of the Federated GRU model in detecting RPL attacks. Given the dataset's size of 975,400 records, utilizing k-fold cross-validation was considered too computationally intensive. Therefore, a hold-out validation method was selected to facilitate efficient training and evaluation while ensuring the model's robustness.

Figure **6- 7** illustrates, that the federated GRU model was trained for 100 federated rounds, exhibiting consistent improvement in accuracy and a significant loss reduction, which indicates efficient convergence. In the initial round, the model attained a training accuracy of 64.99% and a validation accuracy of 70.58%, with corresponding loss values of 0.7058 and 0.5844. As training progressed, the model’s performance steadily advanced, reaching a training accuracy of 98.81% and a validation accuracy of 99.06% by round 100. The training and validation losses also decreased considerably to 0.0291 and 0.0345, respectively. On the test dataset, the finalized

GRU model achieved a test accuracy of 99.06%. Figure 6-7 summarizes that the model demonstrated excellent classification capabilities with an overall precision of 99.06%, a recall of 99.05%, and an F1-score of 99.06%.

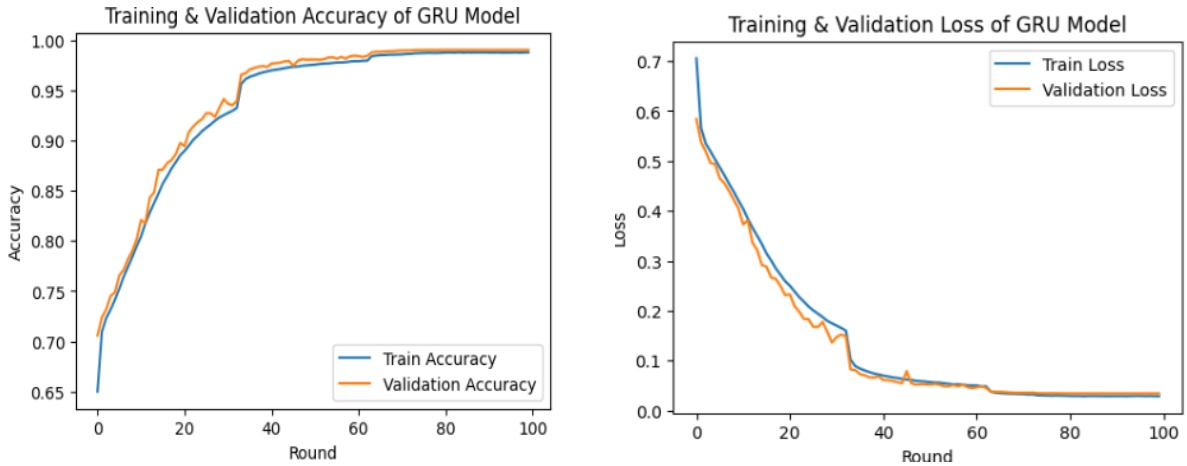

Figure 6- 7: Training and validation accuracy and loss of the GRU model

Table 6- 3: Class-wise performance evaluation of the GRU model

Class	Precision (%)	Recall (%)	F1-score (%)
BH Attack	99.11	98.17	98.64
HF Attack	99.83	99.86	99.84
VN Attack	98.53	99.16	98.84
Benign	98.77	99.05	98.91
Overall	99.06	99.05	99.06

A detailed per-class analysis showed consistently strong results. While achieving a precision of 99.83%, a recall of 99.86%, and an F1-score of 99.84%, the GRU model detected HF attacks with slightly lower accuracy compared to the CNN-GRU and CNN-LSTM models. Similarly, the model effectively detected VN attacks, achieving a precision of 98.53% and a recall of 99.16%, leading to an F1-score of 98.84%. These results reflect the GRU model's sensitivity to temporal irregularities associated with version number attacks. For BH attacks, the model attained a precision of 99.11% and a recall of 98.17%, which produced a high F1-score of 98.64%, demonstrating its strong performance even in the presence of more challenging behaviors like selective forwarding. In classifying benign traffic, the model maintained high

reliability, recording a precision of 98.77% and a recall of 99.05%, with an F1-score of 98.91%, confirming that normal traffic was seldom mistaken for malicious activity.

6.2.4. **Experiment Three: Federated LSTM Model**

This experiment evaluates the performance of the federated LSTM model for detecting RPL attacks in 6LoWPAN networks. Given the dataset size of 975,400 sample records, performing k-fold cross-validation was computationally prohibitive. Therefore, we employed a hold-out validation approach to ensure efficient training and evaluation.

As shown in Figure **6- 8**, the training process involved 100 federated rounds, during which the model demonstrated a consistent and steady improvement in training and validation accuracy. In the first round, the model achieved a training accuracy of 64.78% and a validation accuracy of 70.25%, indicating a solid starting point for learning from distributed IoT traffic data. By round 99, the training accuracy had reached 98.56% and the validation accuracy stood at 98.63%, accompanied by a very low validation loss of 0.0405. In the final round 100, the training accuracy slightly increased to 98.57%, with validation accuracy holding steady at 98.63%, confirming strong convergence and the absence of overfitting. These results reflect the model's capability to effectively capture temporal patterns in network traffic across distributed nodes in a federated learning environment.

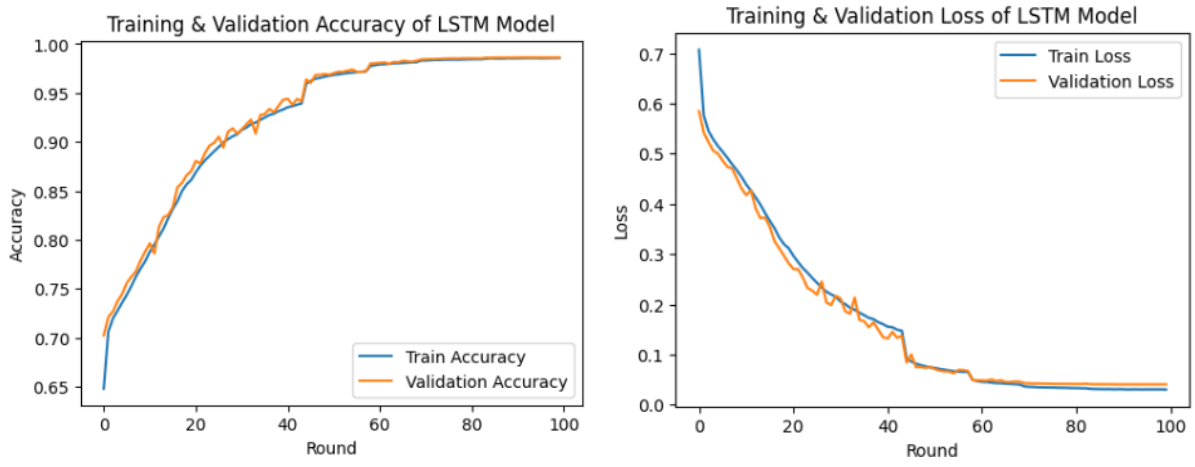

Figure 6- 8:Training and validation accuracy and loss of the LSTM model

The federated LSTM model achieved a test accuracy of 98.63%, demonstrating its robust ability to generalize to unseen data. The overall performance metrics further reinforced this result, with

precision, recall, and F1-score all recorded at approximately 98.63%, indicating balanced and stable classification ability across all categories, as illustrated in the below Table 6-4.

Table 6- 4: Class-wise performance evaluation of the federated LSTM model

Class	Precision (%)	Recall (%)	F1-score (%)
BH Attack	98.05	97.77	97.91
HF Attack	99.83	99.60	99.72
VN Attack	97.86	98.98	98.42
Benign	98.79	98.17	98.48
Overall	98.63	98.63	98.63

Analysis and Key Observations

A closer look at class-wise performance revealed detection capabilities for each traffic type. Although BH attacks are slightly less identifiable compared to other classes, the model achieved a precision of 98.05%, a recall of 97.77%, and an F1-score of 97.91%, demonstrating its effectiveness in detecting disruptive forwarding behaviors. The HF attack was detected with a higher metrics value than others, with a precision of 99.83%, a recall of 99.60%, and an F1-score of 99.72%, highlighting the model's strength in detecting flooding-based attacks that exploit control messages. The VN attack class also demonstrated good detection accuracy, achieving a precision of 97.86%, a recall of 98.98%, and an F1-score of 98.42%, indicating the model's capability to distinguish version inconsistencies in routing messages. For the benign class, the model yielded a precision of 98.79%, a recall of 98.17%, and an F1-score of 98.48%, affirming its reliability in correctly identifying normal traffic patterns while minimizing false positives.

6.2.5. **Evaluation of Experimental Models using Confusion Matrix**

Federated CNN-GRU, CNN-LSTM, GRU, and LSTM models were evaluated on 146,310 samples,36,578 per class: BH, HF, VN, and benign. The confusion matrix provided insights into the model's classification performance, including true positives, false positives, and false negatives, highlighting its strengths and areas for improvement.

6.2.5.1. **Confusion Matrix for Federated CNN-GRU Model**

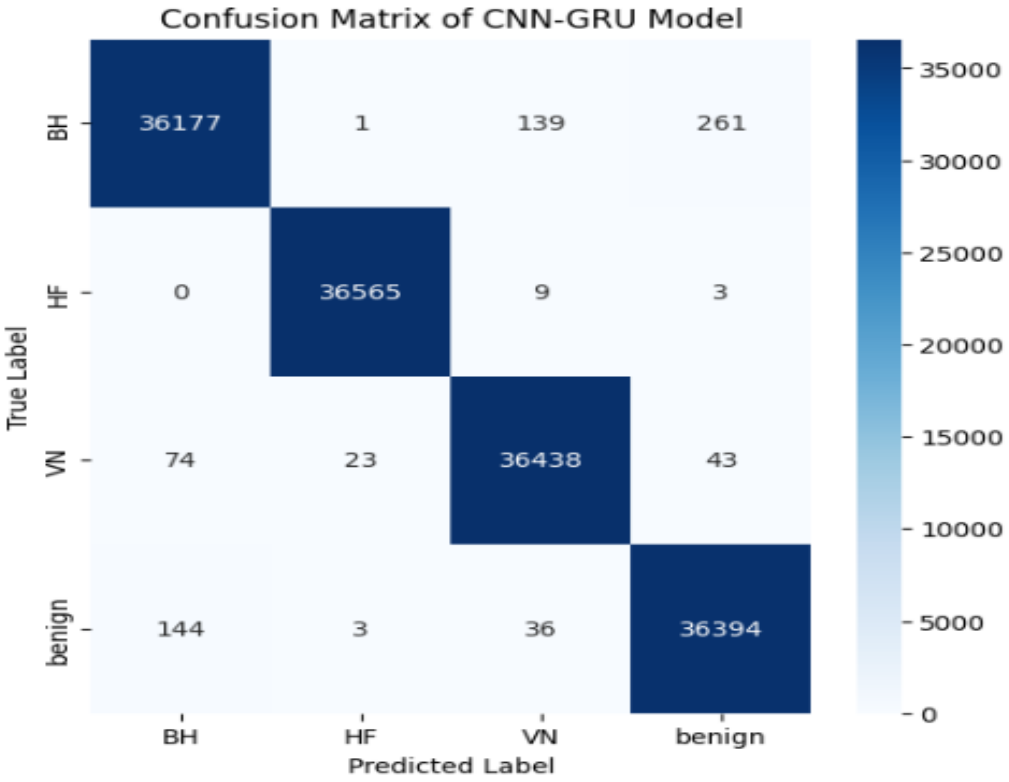

Figure 6- 9: Evaluation of federated CNN-GRU model using confusion matrices

The model accurately identified 36,177 out of 36,578 BH instances, demonstrating a high level of precision in detecting this type of attack. However, 261 BH instances were incorrectly classified as benign, 139 BH instances were misidentified as VN, and 1 BH instance was wrongly classified as HF. The false positives involving BH are minimal, with only 0 HF, 74 VN, and 144 benign instances misclassified as BH. This indicates that the model possesses a robust capability to differentiate BH attacks from other categories. The model successfully classified 36,565 out of 36,577 HF instances. Only 9 HF instances were misclassified as VN, while 3 HF instances were labeled as benign. The false positives for HF were low, with just 1 BH, 23 VN, and 3 benign instances misidentified as HF. This demonstrates that the model is proficient at distinguishing HF attacks from other categories.

The model effectively classified 36,438 VN instances from a total of 36,578. There were 74 instances misclassified as BH, 23 as HF, and 43 as benign. The false positives for VN were relatively low, with 139 BH, 9 HF, and 36 benign instances erroneously classified as VN. This

reflects some overlap between VN, BH, and HF attack types, highlighting the need for further refinement to improve classification boundaries. The model exhibited outstanding performance in recognizing benign traffic, accurately identifying 36,394 out of 36,577 instances. Only 3 benign instances were incorrectly classified as HF, while 36 were misclassified as VN and 144 as BH. The false positives related to benign traffic were minimal, with 261 BH, 3 HF, and 43 VN instances wrongly labeled as benign.

6.2.5.2. **Confusion Matrix for Federated CNN-LSTM Model**

The confusion matrix provides an in-depth evaluation of the model's ability to classify BH, HF, VN, and benign traffic. Each class's performance is analyzed in terms of true positives, false positives, and false negatives, offering insights into the model's strengths and weaknesses. The diagonal elements represent correctly classified instances, while off-diagonal values indicate misclassifications.

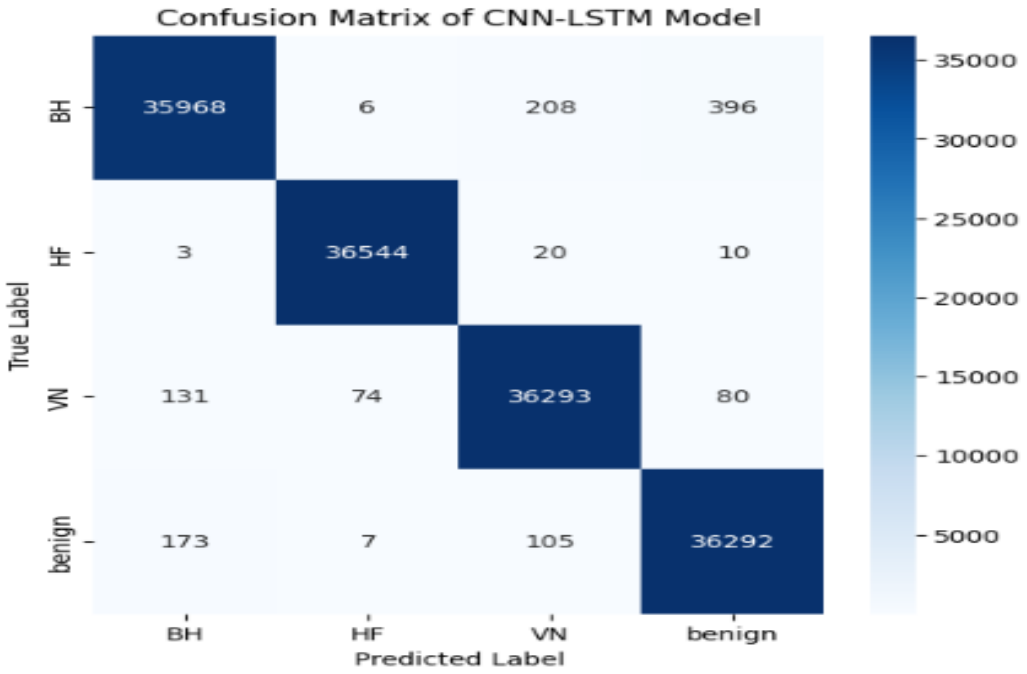

Figure 6- 10: Evaluation of federated CNN-LSTM model using confusion matrices

Out of 36,578 BH instances, the model correctly classified 35,968 samples, demonstrating high accuracy. A total of 208 BH instances were incorrectly categorized as VN, 396 BH as benign, and 6 BH as HF. In terms of false positives, 3 HF, 131 VN, and 173 benign instances were misidentified as BH. While the model effectively detects BH attacks, the misclassifications into

VN and benign categories suggest there is potential for improvement. The model accurately recognized 36,544 out of 36,577 HF samples. The misclassifications consisted of 20 as VN, 10 as benign, and 3 as BH. False positives were low, with just 6 BH, 74 VN, and 7 benign instances mistakenly classified as HF. This reflects the model's ability to accurately distinguish HF traffic, although a few confusions with VN and benign remain.

From the 36,578 VN class, 36,293 instances were correctly classified. However, 131 were misclassified as BH, 74 as HF, and 80 as benign. The false positives for VN consisted of 208 BH, 20 HF, and 105 benign samples incorrectly classified as VN. This indicates a moderate overlap between VN and other attack types, highlighting the necessity for enhanced feature differentiation. The model exhibited high precision in identifying benign traffic, accurately classifying 36,292 out of 36,577 benign cases. Nonetheless, 173 benign cases were incorrectly identified as BH, 7 as HF, and 105 as VN. Regarding false positives, the model erroneously categorized 396 BH, 10 HF, and 80 VN cases as benign. Although the accuracy for the benign class is high, misclassifying attack traffic as benign could present a security threat by allowing certain risks to remain unnoticed. Tackling these false negatives is essential for enhancing the system's reliability in real-world cases.

6.2.5.3. **Confusion Matrix for Federated GRU Model**

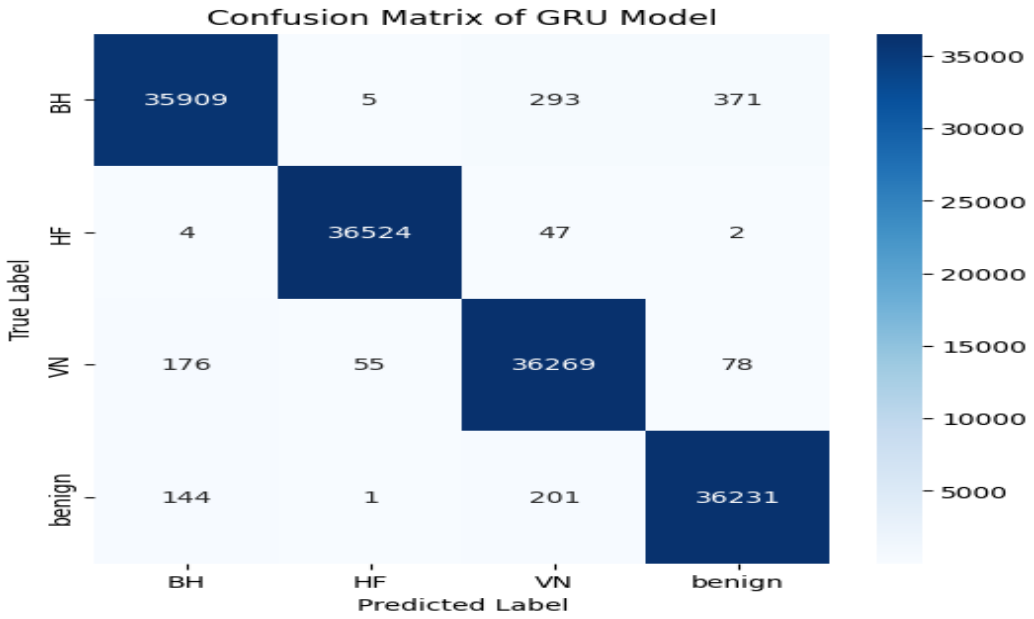

Figure 6- 11:Evaluation of federated GRU model using confusion matrices

The model successfully identified 35,909 out of 36,578 instances for the BH class, demonstrating robust detection capabilities. Nevertheless, 293 BH instances were incorrectly classified as VN, 371 as benign, and 5 as HF. The misclassification of BH traffic as benign is especially troubling from a security perspective, as these attacks might go unnoticed, which poses a threat to the system. While the number of BH instances identified as HF is small, the confusion with VN and benign classes indicates some level of feature overlap. Furthermore, false positives for BH consisted of 4 HF, 176 VN, and 144 benign instances, suggesting that certain normal and attack traffic patterns may be similar to BH behavior.

For the HF class, the model demonstrated excellent performance, accurately classifying 36,524 out of 36,577 instances. Only 47 HF instances were misclassified as VN, 4 as BH, and 2 as benign. These results indicate a very low rate of confusion between HF and other traffic types, highlighting the model's ability to effectively distinguish HF attacks. The false positives for HF were minimal, with 5 BH, 55 VN, and 1 benign instance mistakenly identified as HF, suggesting clear differentiation in the feature space for this attack class. In the VN class, 36,269 out of 36,578 instances were correctly classified, indicating a solid detection capability. However, 176 VN instances were misclassified as BH, 55 as HF, and 78 as benign. Although the model performs well overall, the misclassification of VN traffic as benign, though limited, could affect detection reliability. The false positives for VN included 293 BH, 47 HF, and 201 benign instances, revealing a moderate level of overlap between VN features and those of other classes, particularly benign traffic. The benign class exhibited high accuracy, correctly identifying 36,231 out of 36,577 instances. Misclassifications saw 144 benign instances labeled as BH, 201 as VN, and 1 as HF. While the number of false alarms is relatively low, benign traffic incorrectly classified as BH could result in unnecessary alerts in a live security monitoring environment. Additionally, only 371 BH, 2 HF, and 78 VN instances were incorrectly classified as benign, reinforcing the model's effectiveness in minimizing the risk of attacks being mistaken for normal traffic.

6.2.5.4. **Confusion Matrix for the Federated LSTM Model**

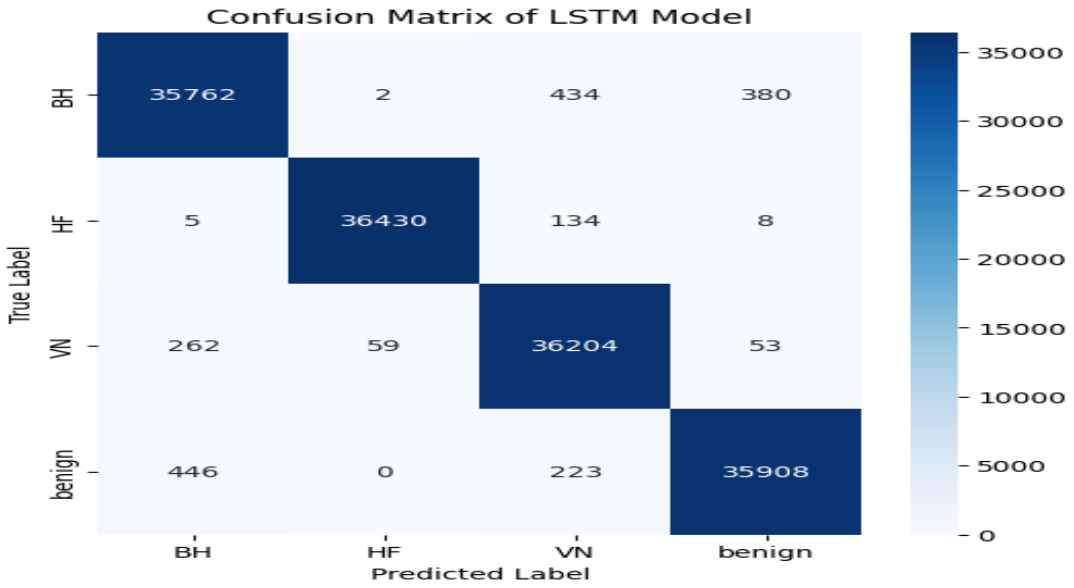

Figure 6- 12: Confusion matrices of federated LSTM model

The model successfully identified 35,762 out of 36,578 BH instances, demonstrating robust detection abilities for the BH category. Nevertheless, 434 BH instances were incorrectly categorized as VN, and 380 BH instances were inaccurately classified as benign, which poses a risk of missing potential attacks. Additionally, only 2 BH instances were misclassified as HF, which represents a negligible error. The false positives for BH, where instances from other categories were wrongly identified as BH, consist of 5 HF, 262 VN, and 446 benign instances, suggesting some overlap in features, especially with the benign class. In the HF category, the model attained very high accuracy, correctly identifying 36,430 out of 36,577 HF instances. However, 134 HF instances were misclassified as VN, and 8 were designated as benign, indicating a minimal misclassification rate amid normal traffic. Only 5 HF instances were misidentified as BH, and the false positives for HF were limited, including 2 BH and 59 VN instances incorrectly classified as HF, which confirms that the model effectively differentiates HF traffic from other classifications.

For the VN class, the model accurately classified 36,204 out of 36,578 VN instances, highlighting strong classification efficacy. However, 262 VN instances were mislabeled as BH, and 53 were classified as benign, which could lead to occasional misidentifications.

Additionally, 59 VN instances were incorrectly categorized as HF. The false positives for VN encompass 434 BH and 134 HF instances, indicating some confusion between VN and other attack types, although this remains within an acceptable range for detection systems. The benign class exhibited outstanding classification performance, with 35,908 out of 36,577 instances being accurately classified. However, 446 benign instances were misidentified as BH, and 223 as VN. No benign instances were mislabeled as HF, highlighting the model's strong capability to differentiate normal traffic from attack types. False positives for benign were minimal, with 380 BH, 8 HF, and 53 VN instances misclassified as benign, indicating that the model seldom confuses attacks with benign activities.

6.2.6. **Comparison of Experimented Models**

This section illustrates the performance of the four federated deep learning models developed for RPL attack detection: CNN-GRU, CNN-LSTM, GRU, and LSTM. Each model was evaluated using accuracy, precision, recall, and F1-score metrics, with special emphasis on their effectiveness across the four traffic classes: BH attack, HF attack, VN attack, and benign. The federated CNN-GRU model achieved the highest performance, with overall precision, recall, and F1-score of 99. 50%. It demonstrated exceptional classification abilities across all attack types. The BH attack was identified with 99.41% precision, 98.98% recall, and 99.19% F1 score. HF attack was classified with a precision of 99.94%, a recall of 99.97%, and an F1-score of 99.95%. The VN attack was detected with high reliability, showing 99.56% precision, 99.66% recall, and 99.61% F1 score. Benign traffic was also correctly identified with 99.23% precision, 99.53% recall, and 99.38% F1-score.

The CNN-LSTM model also performed effectively, achieving an overall precision, recall, and F1 score of 99. 17%. The model demonstrated solid detection of BH attacks with 99.15% precision, 98.33% recall, and 98.74% F1 score. HF attacks were classified with high accuracy, reaching 99.76% precision, 99.91% recall, and 99.84% F1 score. The VN class received 99.09% precision, 99.22% recall, and 99. 16% F1-score. Benign traffic was also identified with 98. 68% precision, 99.22% recall, and 98.95% F1-score.

The Federated GRU model scored overall precision of 99.06%, a recall of 99. 05%, and F 1-F1-score of 99. 06%. The BH class was classified as 99. 11% precision, 98. 17% recall, and 98. 64% F1-score. HF attacks achieved high scores with 99.83% precision, 99. 86% recall, and 99.

84% F1-score. For VN traffic, the model reached 98.53% precision, 99.16% recall, and 98.84% F1-score. The benign class showed 98.77% precision, 99.05% recall, and 98. 91% F1-score.

The federated LSTM model recorded an overall precision, recall, and F1-score of 98.63%. The BH attack class was detected with 98.05% precision, 97.77% recall, and 97.91% F1-score. HF traffic was identified with 99.83% precision, 99.60% recall, and 99.72% F1-score. The VN class scored 97.86% precision, 98.98% recall, and 98.42% F1-score. For the benign class, the model achieved 98.79% precision, 98.17% recall, and 98.48% F1-score.

All four models demonstrated reliable performance in identifying both normal and malicious traffic in RPL networks. The federated CNN-GRU model stands out with its consistently high metrics across all classes, indicating its strong potential for real-world deployment in RPL-based IoT security systems.

Table 6-5 compares the false positive (FPR) and false negative rate (FNR) of four models across four classes. CNN-GRU achieves the lowest error rate overall, indicating superior classification performance, while LSTM shows higher FPR and FNR, especially for BH classes

Table 6- 5: Comparison of experimental models using FPR and FNR

Model	Class	FPR (%)	FNR (%)
CNN-GRU	BH	0.20	1.10
	HF	0.03	0.03
	VN	0.17	0.38
	Benign	0.28	0.50
CNN-LSTM	BH	0.28	1.67
	HF	0.08	0.09
	VN	0.30	0.78
	Benign	0.44	0.78
GRU	BH	0.30	1.83
	HF	0.06	0.15
	VN	0.49	0.85
	Benign	0.41	0.95
LSTM	BH	0.65	2.23
	HF	0.06	0.40
	VN	0.72	1.02
	Benign	0.40	1.83

Table 6- 6: Baseline models comparison based on detected samples

Models	Total test sample	Correctly Classified	Misclassified
CNN-GRU	146,310	145,628	682
CNN-LSTM	146,310	145,097	1,213
GRU	146,310	144,933	1,377
LSTM	146,310	144,304	2,006

Table 6- 7: Comparison of experimental models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Best class	Weakest class
CNN-GRU	99.50	99.50	99.50	99.50	HF	BH
CNN-LSTM	99.17	99.17	99.17	99.17	HF	BH
GRU	99.06	99.06	99.05	99.06	HF	BH
LSTM	99.68	98.63	98.63	98.63	HF	BH

Table 6- 8: Class-wise performance comparison of the experimental models

Class	Model	Precision (%)	Recall (%)	F1-score (%)
BH	CNN-GRU	99.40	98.98	99.19
	CNN-LSTM	99.15	98.33	98.74
	GRU	99.11	98.17	98.64
	LSTM	98.05	97.77	97.91
HF	CNN-GRU	99.93	99.97	99.95
	CNN-LSTM	99.76	99.91	99.84
	GRU	99.83	99.86	99.84
	LSTM	99.83	99.60	99.72
VN	CNN-GRU	99.50	99.62	99.56
	CNN-LSTM	99.09	99.22	99.16
	GRU	98.53	99.16	98.84
	LSTM	97.86	98.98	98.42
benign	CNN-GRU	99.16	99.50	99.33
	CNN-LSTM	98.68	99.22	98.95
	GRU	98.77	99.05	98.91
	LSTM	98.79	98.17	98.48

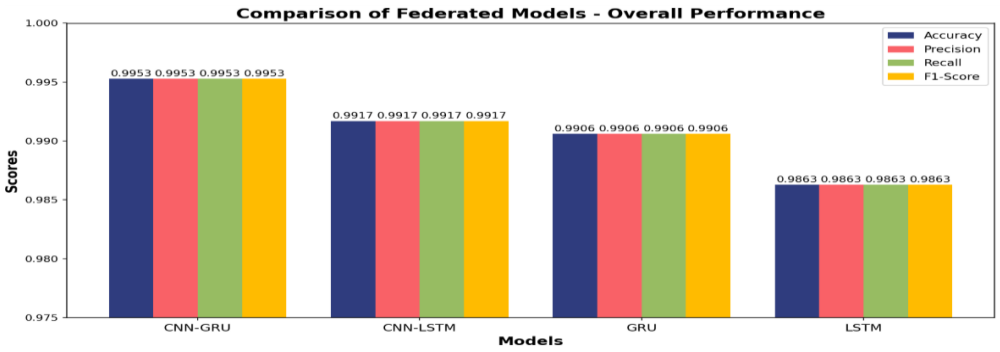

Figure 6- 13: Overall Performance comparison of experimented federated models

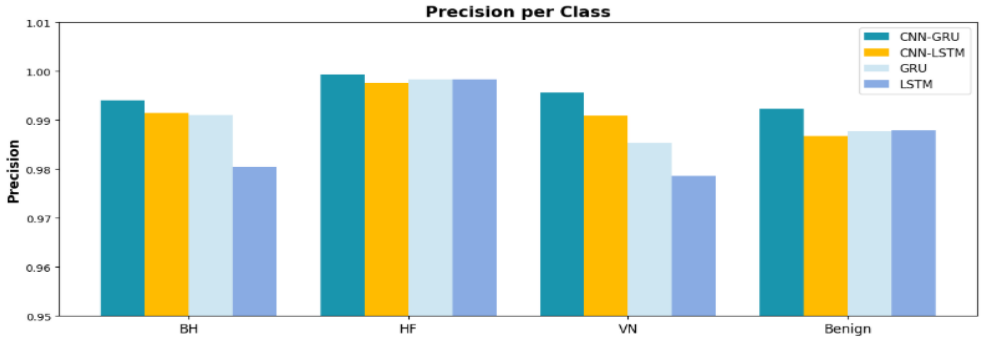

Figure 6- 14: Precision performance analysis per class for experimental models

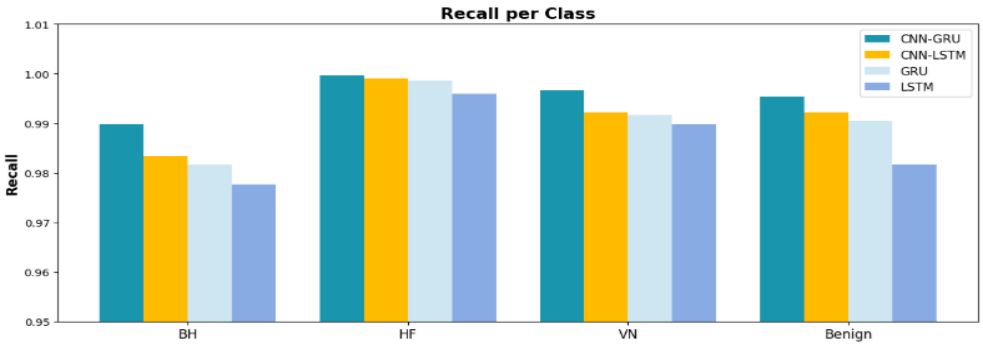

Figure 6- 14: Recall performance analysis per class for the experimental models

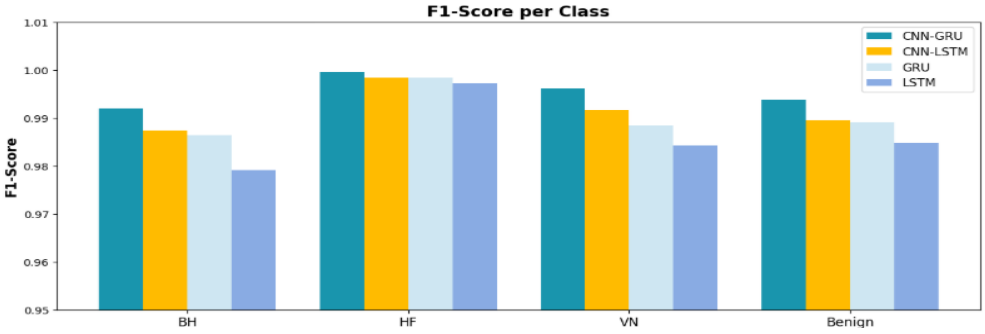

Figure 6- 15: F1-Score performance analysis per class for experimental models

6.3. **Comparison of Our Proposed Models with Base Paper Models**

The performance of the proposed federated hybrid deep Learning model was rigorously evaluated against five centralized models adopted from the base paper, all trained on the same dataset for a fair and consistent comparison. These baseline models include Centralized LSTM, CNN, FCNN, DT, and RF. Among them, the Centralized LSTM performed best with an overall accuracy of 97.61%, followed by RF with 96.02%. However, the proposed federated model significantly outperformed them all, achieving an outstanding 99.50% accuracy, along with 99.50% precision, recall, and F1-score.

On a class-wise level, the improvements are even more compelling. For the BH attack, our model achieved an F1-score of 99.15%, notably higher than LSTM with 96% and RF with 94.72%, indicating a superior ability to detect more subtle or evasive forms of the attack. For the HF attack, although most models performed well, the proposed model still led with an exceptional F1-score of 99.95%, reflecting near-perfect classification. When it comes to the VN attack, our model again outshined the rest with a 99.56% F1-score, surpassing LSTM's 98% and RF's 95.64%. Furthermore, our model demonstrated a strong capability to identify benign traffic, scoring 99.33% in F1, which is critical to avoid false positives and unnecessary mitigations.

These results are particularly impressive considering that the baseline models were trained using a centralized learning approach, which assumes all network traffic data is collected in one place. In contrast, our model operates under a federated learning paradigm, where data remains decentralized across multiple clients. Despite this challenge, the federated model not only preserves data privacy and security but also delivers significantly better performance, proving its practical value in real-world IoT environments like 6LoWPAN, where privacy-preserving intelligence at the edge is essential.

Table 6- 9: Proposed model vs. base paper models on same dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Proposed Federated CNN-GRU Model	99.50	99.50	99.50	99.50
LSTM	97.61	97.60	97.61	97.60
RF	96.02	96.10	96.02	96.02
DT	93.15	93.20	93.15	93.12
CNN	89.93	89.96	89.93	89.89
FCNN	83.67	85	84	83

6.4. **Discussion of Conceptual Framework for Mitigation**

As illustrated in Figure **4- 1**, a conceptual mitigation architecture is presented for the security of RPL-based IoT networks. Upon a lightweight CNN-GRU model to classify RPL attacks, the system sends an alert to the DODAG root or network management system dynamically to take action for mitigating those detected attacks. During the inference process, a device (such as a Raspberry Pi 5) intercepts raw RPL traffic and extracts initial features using the first convolutional layer of the CNN. These features are then sent to a centralized server where a unified, pre-trained CNN-GRU model performs a comprehensive analysis. The system records the ID of the node and pertinent statistics, classifying the traffic as either benign or one of the three attack types. When an attack is detected, the framework implements a response strategy tailored to the type of attack:

- For a BH attack, there are two actions, centralized and local actions. In the centralized, the detection alert should be sent to the DODAG root and this root modifies the routing table and disseminates this blacklisted node to each node in the network. In local action, each node receives the blacklisted node, and it does not perform routing with this node, dropping any advertisements from this node going forward.
- For VN attack: In the centralized action the root directs all nodes to ignore the attacker's fake version number increase or decrease. In local action, each node ignores suspicious VN from the attack to avoid topology corruption. The root node applies blocking policies to prevent VN-induced topology resets, thus preserving network stability.
- During an HF attack: Root, upon receiving a flooding report, directs all nodes to strictly rate-limit DIO messages from the suspect. the root instructs the nodes to ignore all

communication with the flooding source to resolve resource exhaustion. and each node aggressively applies a rate limit on DIO message from the flooding neighbor

Routing table evolution adaptation for BH attack: The routing tables below are constructed accordingly, as shown in the RPL nodes topology figure below.

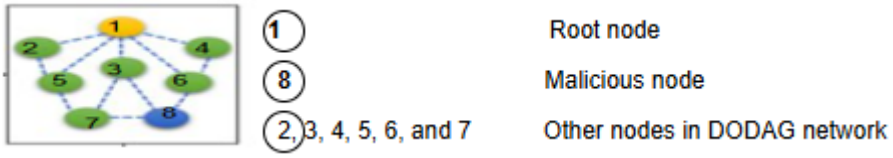

Figure 6- 16: RPL nodes topology

Before attack detection, the network assumes all RPL nodes to be trustworthy, allowing unrestricted participation in routing activities. The routing table is as follows:

Table 6- 10: Conceptual Routing table before attack detection

Destination IP	Next Hop	Interface	Rank	Status
2001:db8::1 (DODAG Root)	::1 (Self)	Lo0	0	Root node
2001:db8::2	2001:db8::1	eth0	1	Active
2001:db8::3	2001:db8::1	eth1	1	Active
2001:db8::4	2001:db8::1	eth2	1	Active
2001:db8::5	2001:db8::1	eth3	1	Active
2001:db8::6	2001:db8::1	eth4	1	Active
2001:db8::7	2001:db8::3	eth5	2	Active
2001:db8::8	2001:db8::6	eth6	2	Active

After a suspicious node is detected, the CNN-GRU model flags node 2001:db8::8 as suspicious, indicating abnormal behavior. The root node modifies the routing table, marks it as blacklisted, and disseminates this information to all nodes in the network. Each node reroutes to another parent and blocks any advertisement messages from the malicious node thereafter.

Table 6- 11: Routing table when mitigation applied

Destination IP	Next Hop	Interface	Rank	Status
2001:db8::1 (DODAG Root)	::1 (Self)	Lo0	1	Root Node
2001:db8::2	2001:db8::1	eth0	1	Active
2001:db8::3	2001:db8::1	eth1	1	Active
2001:db8::4	2001:db8::1	eth2	1	Active
2001:db8::5	2001:db8::1	eth3	1	Active
2001:db8::6	2001:db8::1	eth4	1	Active
2001:db8::7	2001:db8::3	eth5	2	Active
2001:db8::8	2001:db8::6	eth6	2	**BH (black listed)**

While this framework is conceptual and not implemented in a live testbed, it offers a significant step toward self-adaptive IoT security. It proposes an intelligent, layered defense strategy that combines anomaly detection, automated routing adjustments, and firewall enforcement.

6.5. **Research Questions Discussion**

This research study has 4 research questions, which lie in the first chapter of section **1.4**. Hence, our study has answered this question according to their order as follows.

- **RQ1**: How do we select important features from IRAD and apply effective preprocessing techniques for a federating framework to improve RPL attack detection and classification?

To address this research question, we designed a comprehensive and strategic pipeline focused on generalization, performance, and real-world relevance. We began by collecting datasets from different network sizes, such as small, medium, and large, to ensure our model could generalize effectively across various IoT scenarios. Despite differences in network scale, all datasets shared a consistent set of features, which allowed us to seamlessly merge them into a unified dataset representing four classes. This approach mimics realistic deployment environments, making the resulting model robust across heterogeneous conditions. To improve data quality, we applied Isolation Forest for outlier detection, which efficiently removed anomalous records based on patterns in the feature space without requiring labeled data. Duplicate records were also identified and eliminated to avoid redundancy and prevent biased learning. Normalization

was performed using Min-Max scaling to ensure that all features were transformed into the same range. This step is particularly important for deep learning models, which are sensitive to input magnitude. To address the class imbalance, random under-sampling was implemented, reducing the dominance of class samples and encouraging the model to learn attack patterns.

We further examined feature relevance through Pearson correlation analysis to identify and discard features with high correlation, thus reducing redundancy and potential multicollinearity. To refine feature selection, we employed Random Forest and XGBoost classifiers to independently rank the importance of features based on their contribution to model performance. What sets our approach apart is the integration of both techniques: we created a union of the top-ranked features from each model. This hybrid feature selection strategy ensured that no critical features were overlooked by a single feature selection technique, capturing complementary insights from both methods and strengthening the model's learning capacity. An essential enhancement was introduced through temporal feature transformation. Sequential data has been created to present features that change over time. This creation significantly benefits time-aware models like LSTM and GRU, enabling them to effectively capture sequence dependencies crucial for identifying stealthy or evolving attacks.

To enhance the training efficiency of our federated learning framework, we optimized the input pipeline using key TensorFlow dataset transformations. Specifically, we employed batching, shuffling to prevent model overfitting due to ordered data, and prefetching to overlap data loading with model execution. These optimizations helped minimize training latency and improved model convergence across distributed clients, particularly in resource-constrained settings. To align with the federated learning paradigm, we employed a round-robin distribution method to allocate data to each federated client. This ensured balanced class representation across all clients and avoided skewed training, ultimately improving both convergence and detection accuracy. This combination of intelligent preprocessing, multi-method feature selection, and strategic data distribution forms the foundation of a highly adaptable and accurate RPL attack detection framework, bridging the gap between theoretical development and real-world IoT conditions.

- **RQ2**: How can a federated hybrid deep learning model be developed to detect and classify RPL attacks?

To address this research question, a federated learning framework was designed and implemented using TensorFlow and Keras, where multiple deep learning architectures were explored and evaluated in a distributed setting. The IRAD dataset was carefully preprocessed, and a sequential dataset was constructed to reflect the temporal characteristics of RPL network traffic. The entire dataset was then partitioned into training, validation, and testing subsets, with eight clients assigned to the federated setup: six for training, one for validation, and one reserved for final testing.

The training data was evenly divided among the six federated training clients, ensuring that each client received a fair and balanced portion. Two architectural approaches were proposed and implemented: The Sequential API and the Functional (Parallel) API. Within each of these architectures, four well-established models including GRU, LSTM, CNN-GRU, and CNN-LSTM, were experimented with to determine the most effective structure for the task. These models were trained in a federated manner across the six clients, and hyperparameter tuning was carried out using the dedicated validation client, which helped refine model updates before aggregation.

Throughout the training process, the Adam optimizer was used both at the client and server levels, with a learning rate of 0.001 over 100 communication rounds. After comprehensive experimentation, the Sequential API implementation consistently outperformed the Functional API. Once the final model was established, it was evaluated on the separate test client to assess its effectiveness in detecting and classifying RPL attacks.

- **RQ3**: How does the performance of the proposed federated hybrid deep learning architecture compare to baseline models and those presented in the base paper?

To answer this research question, a thorough evaluation of the proposed federated hybrid deep learning model was carried out against both the baseline model and other models from base paper. The base paper comparison included classical machine learning models including RF, and DT and deep learning like CNN, LSTM, and FCNN. Our architecture integrated federated learning with a CNN-GRU hybrid to address both privacy and performance was compared to baseline models and models from base paper. The performance was assessed using standardized metrics including accuracy, precision, recall, and F1-score across RPL attack types such as BH, HF, and VN. The proposed model achieved superior results, especially excelling in precision

and recall, demonstrating its robustness and generalizability. Notably, it achieved **99.50%** accuracy, outperforming most existing works while ensuring privacy-preserving distributed training. This comparative evaluation validates the strength of our federated CNN-GRU model as an effective solution for RPL attack detection and classification in 6LoWPAN networks.

- **RQ4**: Using a conceptual framework, how can detected RPL attacks be mitigated within a 6LoWPAN network?

To address RQ4, a conceptual mitigation framework is designed to respond to detected RPL attacks specifically VN, BH, and HF in 6LoWPAN networks. This framework defines the actions to be taken by the DODAG root and the network nodes once an attack is identified.

For VN attacks, the root node maintains a watchlist of suspicious nodes and instructs the rest of the network to ignore abnormal version number changes. This prevents topology instability caused by malicious version advertisements. Each node reinforces this by ignoring version updates from the identified attacker. In the case of BH attacks, the root node removes the attacker from its routing table and broadcasts the blacklist to the network. All nodes then stop routing through or accepting messages from the blacklisted node. During HF attacks, the root node alerts other nodes to apply strict rate-limiting on DIO messages from the flooding source.

CHAPTER SEVEN

7. CONCLUSION and FUTURE WORKS

7.1. Conclusion

The increasing adoption of 6LoWPAN in IoT networks has highlighted the critical need for robust security mechanisms against RPL attacks, which pose risks to the reliability, confidentiality, and availability of network operations. In this work, we proposed a novel framework for the detection, classification, and conceptual mitigation of RPL attacks by employing a federated hybrid deep learning approach. A carefully generalized dataset was created from the IRAD source, covering small, medium, and large network sizes to comprehensively represent real-world IoT scenarios. This inclusion ensures that the trained models are highly adaptable across varying scales of IoT networks. An extensive dataset was assembled from IRAD sources, initially comprising 2,200,000 samples drawn from diverse network sizes, including 10 nodes,20 nodes,100 nodes, and 1000 nodes, to simulate realistic IoT environments.

Through rigorous preprocessing, including duplicate removal, outlier detection, and normalization, the dataset was refined to 975,400 samples. A hybrid feature selection technique was implemented, where Pearson correlation analysis was conducted, followed by an independent evaluation of the results using the Random Forest Classifier and XGBoost methods. The highest-ranked features were identified by each model and merged, leading to a total of 14 unique features including the time feature which was not used for training, but used to structure the dataset to the sequence. These selected features were subsequently used for training, enhancing the model's resilience and reducing feature bias. The dataset was partitioned with 70% allocated for training, 15% for validation, and 15% for testing. A federated learning framework was adopted to ensure data privacy, leveraging IID data distribution and a round-robin assignment strategy that guaranteed balanced class label distribution across clients, significantly improving convergence and performance in decentralized training scenarios.

Four deep learning models, namely CNN-GRU, CNN-LSTM, GRU, and LSTM, were experimented with different architectures and thoroughly trained within the federated learning

framework, achieving accuracies of 99.50%, 99.17%, 99.06%, and 98.68%, respectively. Among these models, CNN-GRU demonstrated the highest performance, reaching an outstanding 99.50% accuracy in detecting and classifying RPL attacks. This exceptional result led to the selection of CNN-GRU as the most effective model for ensuring secure and reliable IoT network operations. This study highlights that combining federated learning with hybrid deep learning architectures and advanced feature selection significantly enhances detection capabilities while preserving the privacy of sensitive IoT data. It further demonstrates that using generalized datasets encompassing various network scales ensures strong adaptability in real-world IoT environments.

7.2. **Future Works**

While this study has made significant advancements in the detection, classification, and conceptual mitigation of RPL attacks in 6LoWPAN networks using federated hybrid deep learning, several directions remain open for future exploration.

Firstly, personalizing the federated learning framework is an important next step. Developing a personalized federated learning approach that adapts to each client's local data distribution would allow the system to become more robust, particularly when handling heterogeneous datasets where client data varies significantly. Training federated clients under non-independent and identically distributed conditions and across diverse network environments can further enhance the adaptability and real-world applicability of the proposed model. Secondly, while the current system is specifically designed to detect BH, VN, and HF RPL attacks, extending the detection framework to include a broader range of RPL attacks would make the system even more comprehensive and resilient. Thirdly, although federated learning already provides a baseline of privacy preservation, integrating advanced privacy-enhancing techniques such as differential privacy, secure multi-party computation, or homomorphic encryption could further strengthen the privacy guarantees of the system during federated training and communication, particularly for highly sensitive IoT deployments.

Special Acknowledgments: -This research was funded by Adama Science and Technology University with grant number: **ASTU/ SM-R/1168/25**

Adama, Ethiopia

Reference

Abd-El-Atty, B., ElAffendi, M., & El-Latif, A. A. A. (2023). A novel image cryptosystem using Gray code, quantum walks, and Henon map for cloud applications. *Complex and Intelligent Systems*, *12*(11), 604–624. https://doi.org/https://doi.org/10.1007/s40747-022-00829-z

Abdmeziem, M. R., Tandjaoui, D., & Romdhani, I. (2016). Architecting the internet of things: State of the art. *Studies in Systems, Decision and Control*, *36*(June), 55–75. https://doi.org/10.1007/978-3-319-22168-7_3

Agiollo, A., Conti, M., Kaliyar, P., Lin, T. N., & Pajola, L. (2021). DETONAR: Detection of Routing Attacks in RPL-Based IoT. *IEEE Transactions on Network and Service Management*, *18*(2), 1178–1190. https://doi.org/10.1109/TNSM.2021.3075496

Al-Amiedy, T. A., Anbar, M., Belaton, B., Kabla, A. H. H., Hasbullah, I. H., & Alashhab, Z. R. (2022). A Systematic Literature Review on Machine and Deep Learning Approaches for Detecting Attacks in RPL-Based 6LoWPAN of Internet of Things. *Sensors*, *22*(9). https://doi.org/10.3390/s22093400

Alazab, A., Khraisat, A., Singh, S., Bevinakoppa, S., & Mahdi, O. A. (2023). Routing Attacks Detection in 6LoWPAN-Based Internet of Things. *Electronics (Switzerland)*, *12*(6). https://doi.org/10.3390/electronics12061320

Albishari, M., Li, M., Zhang, R., & Almosharea, E. (2023). Deep learning-based early stage detection (DL-ESD) for routing attacks in Internet of Things networks. *Journal of Supercomputing*, *79*(3), 2626–2653. https://doi.org/10.1007/s11227-022-04753-4

Alfriehat, N., Anbar, M., Aladaileh, M., Hasbullah, I., Shurbaji, T. A., Karuppayah, S., & Almomani, A. (2024). RPL-based attack detection approaches in IoT networks: review and taxonomy. In *Artificial Intelligence Review* (Vol. 57, Issue 9). Springer Netherlands. https://doi.org/10.1007/s10462-024-10907-y

Alkama, L., & Bouallouche-Medjkoune, L. (2021). IEEE 802.15.4 historical revolution versions: A survey. *Computing*, *103*(1), 99–131. https://doi.org/10.1007/s00607-020-00844-3

Almusaylim, Z. A., Jhanjhi, N. Z., & Alhumam, A. (2020). Detection and mitigation of RPL rank and version number attacks in the internet of things: SRPL-RP. *Sensors*

(Switzerland), *20*(21), 1–25. https://doi.org/10.3390/s20215997

Alrefaei, A., & Ilyas, M. (2024). Using Machine Learning Multiclass Classification Technique to Detect IoT Attacks in Real Time. *Sensors*, *24*(14). https://doi.org/10.3390/s24144516

Alsubai, S., Alqahtani, A., Garg, H., Sha, M., & Gumaei, A. (2024). A blockchain-based hybrid encryption technique with anti-quantum signature for securing electronic health records. https://doi.org/10.2991/ijcis.2018.25905181

Ashraf Zargar, S. (2021). *Introduction to Sequence Learning Models: RNN, LSTM, GRU*. *April*. https://doi.org/10.13140/RG.2.2.36370.99522

Bang, A. O., Rao, U. P., Kaliyar, P., & Conti, M. (2023). Assessment of Routing Attacks and Mitigation Techniques with RPL Control Messages: A Survey. *ACM Computing Surveys*, *55*(2), 1–35. https://doi.org/10.1145/3494524

Bedi, P., Gupta, N., & Jindal, V. (2020). Siam-IDS: Handling class imbalance problem in Intrusion Detection Systems using Siamese Neural Network. *Procedia Computer Science*, *171*(2019), 780–789. https://doi.org/10.1016/j.procs.2020.04.085

Benaddi, H., Jouhari, M., Ibrahimi, K., Benslimane, A., & Amhoud, E. M. (2023). Improvement of Anomaly Detection System in the IoT Networks using CNN-LSTM Approach. *Proceedings - IEEE Global Communications Conference, GLOBECOM*, 3771–3776. https://doi.org/10.1109/GLOBECOM54140.2023.10437475

Berguiga, A., Harchay, A., & Massaoudi, A. (2025). HIDS-RPL: A Hybrid Deep Learning-Based Intrusion Detection System for RPL in Internet of Medical Thing Networks. *IEEE Access*, *13*(March), 38404–38429. https://doi.org/10.1109/ACCESS.2025.3545918

Bokka, R., & Sadasivam, T. (2023). Securing IoT Networks: RPL Attack Detection with Deep Learning GRU Networks. *International Journal of Recent Engineering Science*, *10*(2), 13–21. https://doi.org/10.14445/23497157/ijres-v10i2p103

Cakir, S., Toklu, S., & Yalcin, N. (2020). Rpl attack detection and prevention in the internet of things networks using a gru based deep learning. *IEEE Access*, *8*, 183678–183689. https://doi.org/10.1109/ACCESS.2020.3029191

Choukri, W., Lamaazi, H., & Benamar, N. (2020). RPL rank attack detection using Deep Learning. *2020 International Conference on Innovation and Intelligence for Informatics* https://doi.org/10.1109/3ICT51146.2020.9311983

da Costa, K. A. P., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, *151*, 147–157. https://doi.org/10.1016/j.comnet.2019.01.023

Drainakis, G., Katsaros, K. V., Pantazopoulos, P., Sourlas, V., & Amditis, A. (2020). Federated vs. Centralized Machine Learning under Privacy-elastic Users: A Comparative Analysis. 2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020. https://doi.org/10.1109/NCA51143.2020.9306745

Elakkiya, E., Bista, R. B., Shah, C., Rajput, A., Gupta, A. K., & Chaudhary, R. (2024). RBFN-Augmented DDoS Detection with CNN-GRU Fusion. *2024 15th International Conference on Computing Communication and Networking Technologies, ICCCNT 2024*, 1–7. https://doi.org/10.1109/ICCCNT61001.2024.10724299

Fu, R., Zhang, Z., & Li, L. (2017). Using LSTM and GRU neural network methods for traffic flow prediction. *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, 324–328. https://doi.org/10.1109/YAC.2016.7804912

Gaddour, O., & Koubâa, A. (2012). RPL in a nutshell: A survey. *Computer Networks*, *56*(14), 3163–3178. https://doi.org/10.1016/j.comnet.2012.06.016

Hadipuspito, E. B., & Suryani, V. (2024). Hybrid Deep Learning for Botnet Attack Detection on IoT Networks using CNN-GRU. *2024 International Conference on Data Science and Its Applications (ICoDSA)*, 133–139. https://doi.org/10.1109/icodsa62899.2024.10652152

Jony, A. I., & Arnob, A. K. B. (2024). A long short-term memory based approach for detecting cyber attacks in IoT using CIC-IoT2023 dataset. *Journal of Edge Computing*, *3*(1), 28–42. https://doi.org/10.55056/jec.648

Kamel, S. O. M., & Elhamayed, S. A. (2020). Mitigating the impact of iot routing attacks on power consumption in iot healthcare environment using convolutional neural network. *International Journal of Computer Network and Information Security*, *12*(4), 11–29. https://doi.org/10.5815/ijcnis.2020.04.02

Khan, N. W., Alshehri, M. S., Khan, M. A., Almakdi, S., Moradpoor, N., Alazeb, A., Ullah, S., Naz, N., & Ahmad, J. (2023). A hybrid deep learning-based intrusion detection system for IoT networks. *Mathematical Biosciences and Engineering*, *20*(8), 13491–

13520. https://doi.org/10.3934/mbe.2023602

Khan, R., Tariq, N., Ashraf, M., Khan, F. A., Shafi, S., & Ali, A. (2024). FL-DSFA: Securing RPL-Based IoT Networks against Selective Forwarding Attacks Using Federated Learning. *Sensors*, *24*(17), 1–25. https://doi.org/10.3390/s24175834

Kowsalyadevi, K., & Balaji, N. V. (2024). Federated Learning-based Routing Vulnerability Analysis and Attack Detection for Healthcare 4.0. *International Journal of Intelligent Engineering and Systems*, *17*(2), 412–428. https://doi.org/10.22266/ijies2024.0430.34

Learning, S., Hamood, M., Albaseer, A., Abdallah, M., & Member, S. (n.d.). *Efficient Data Labeling and Optimal Device Scheduling in HWNs Using Clustered Federated*. 1–17.

Mansour, M., Gamal, A., Ahmed, A. I., Said, L. A., Elbaz, A., Herencsar, N., & Soltan, A. (2023). Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions. *Energies*, *16*(8), 1–39. https://doi.org/10.3390/en16083465

Mawela, C., Issaid, C. Ben, & Bennis, M. (2025). A Web-Based Solution for Federated Learning With LLM-Based Automation. *IEEE Internet of Things Journal*, 1–14. https://doi.org/10.1109/JIOT.2025.3542897

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516. https://doi.org/10.1016/j.adhoc.2012.02.016

Momand, M. D., Khan Mohsin, M., & Ihsanulhaq. (2021). Machine Learning-based Multiple Attack Detection in RPL over IoT. *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*. https://doi.org/10.1109/ICCCI50826.2021.9402388

Morales-Molina, C. D., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L. K., Perez-Meana, H., Olivares-Mercado, J., Portillo-Portillo, J., Sanchez, V., & Garcia-Villalba, L. J. (2021). A dense neural network approach for detecting clone id attacks on the rpl protocol of the iot. *Sensors*, *21*(9), 1–24. https://doi.org/10.3390/s21093173

Muhammad Salman Bukhari, S., Zafar, M. H., Houran, M. A., Qadir, Z., Kumayl Raza Moosavi, S., & Sanfilippo, F. (2024). Enhancing cybersecurity in Edge IIoT networks: An asynchronous federated learning approach with a deep hybrid detection model. *Internet of Things (Netherlands)*, *27*(June). https://doi.org/10.1016/j.iot.2024.101252

Musaddiq, A., Zikria, Y. Bin, & Kim, S. W. (2020). *Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network*.

Najar, A. A., & Manohar Naik, S. (2022). DDoS attack detection using MLP and Random Forest Algorithms. *International Journal of Information Technology (Singapore)*, *14*(5), 2317–2327. https://doi.org/10.1007/s41870-022-01003-x

Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., & Jirstrand, M. (2018). A performance evaluation of federated learning algorithms. *DIDL 2018 - Proceedings of the 2nd Workshop on Distributed Infrastructures for Deep Learning, Part of Middleware 2018*, 1–8. https://doi.org/10.1145/3286490.3286559

Oliveira, L., Rodrigues, J. J. P. C., Kozlov, S. A., Rabêlo, R. A. L., & de Albuquerque, V. H. C. (2019). MAC layer protocols for internet of things: A survey. *Future Internet*, *11*(1), 1–42. https://doi.org/10.3390/fi11010016

Pradeepthi, C., Vijaya Geetha, V. J., Ramasubbareddy, S., & Govinda, K. (2020). Prediction of real estate price using clustering techniques. In *Advances in Intelligent Systems and Computing* (Vol. 1054). https://doi.org/10.1007/978-981-15-0135-7_27

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Reshi, I. A., Sholla, S., & Najar, Z. A. (2024). Safeguarding IoT networks: Mitigating black hole attacks with an innovative defense algorithm. *Journal of Engineering Research (Kuwait)*, *12*(1), 133–139. https://doi.org/10.1016/j.jer.2024.01.014

Rey, V., Miguel, P., Sánchez, S., Huertas, A., & Bovet, G. (2023). *Federated Learning for Malware Detection in IoT Devices*.

Sahay, R., Geethakumari, G., & Mitra, B. (2022). A holistic framework for prediction of routing attacks in IoT-LLNs. *Journal of Supercomputing*, *78*(1), 1409–1433. https://doi.org/10.1007/s11227-021-03922-1

Seba, A. M., Gemeda, K. A., & Ramulu, P. J. (2024). Prediction and classification of IoT sensor faults using hybrid deep learning model. *Discover Applied Sciences*, *6*(1). https://doi.org/10.1007/s42452-024-05633-7

Sen, S. (2021). *A Cross-Layer Intrusion Detection System for RPL-Based Internet of Things A Cross-Layer Intrusion Detection System for RPL-Based Internet of Things*. *October*

2020. https://doi.org/10.1007/978-3-030-61746-2

Shah, H., Patel, R., & Tawde, P. (2023). Federated Learning to Preserve the Privacy of User Data. *2023 Somaiya International Conference on Technology and Information Management, SICTIM 2023*, 23–27. https://doi.org/10.1109/SICTIM56495.2023.10104860

Shahid, U., Zunnurain Hussain, M., Zulkifl Hasan, M., Haider, A., Ali, J., & Altaf, J. (2024). Hybrid Intrusion Detection System for RPL IoT Networks Using Machine Learning and Deep Learning. *IEEE Access*, *12*, 113099–113112. https://doi.org/10.1109/ACCESS.2024.3442529

Shanmugarasa, Y., Paik, H. young, Kanhere, S. S., & Zhu, L. (2023). A systematic review of federated learning from clients' perspective: challenges and solutions. In *Artificial Intelligence Review* (Vol. 56, Issue s2). Springer Netherlands. https://doi.org/10.1007/s10462-023-10563-8

Sun, T., Li, D., & Wang, B. (2023). Decentralized Federated Averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(4), 4289–4301. https://doi.org/10.1109/TPAMI.2022.3196503

Tariq, N., Khan, R., Almufareh, M. F., Humayun, M., & Shaheen, M. (2024). A Federated Learning Mechanism for Mitigating Selective Forwarding Attacks in RPL-Based Internet of Things. *2024 IEEE International Conference on Communications Workshops, ICC Workshops 2024*, 871–877. https://doi.org/10.1109/ICCWorkshops59551.2024.10615385

U.Farooq, M., Waseem, M., Mazhar, S., Khairi, A., & Kamal, T. (2015). A Review on Internet of Things (IoT). *International Journal of Computer Applications*, *113*(1), 1–7. https://doi.org/10.5120/19787-1571

Vashi, S., Ram, J., Modi, J., Verma, S., & Prakash, C. (2017). Internet of Things (IoT): A vision, architectural elements, and security issues. *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, *December 2018*, 492–496. https://doi.org/10.1109/I-SMAC.2017.8058399

Wang, X., Yang, W., Hou, C., & Luo, H. (2024). Routing Attack and Detection Methods in the RPL- based Internet of Things. *2024 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 127–130.

https://doi.org/10.1109/CyberC62439.2024.00031

Xu, L. Da, He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, *10*(4), 2233–2243.

https://doi.org/10.1109/TII.2014.2300753

Yadollahzadeh-Tabari, M., & Mataji, Z. (2021). Detecting Sinkhole Attack in RPL-based Internet of Things Routing Protocol. *Technology Journal of Artificial Intelligence and Data Mining*, *9*(1), 73–85. https://doi.org/10.22044/JADM.2020.9253.2060

Yang, Z., & Chang, C. H. (2019). 6Lowpan Overview and Implementations. *International Conference on Embedded Wireless Systems and Networks*, 357–361.

Yavuz, F. Y., Ünal, D., & Gül, E. (2018). Deep learning for detection of routing attacks in the internet of things. *International Journal of Computational Intelligence Systems*, *12*(1), 39–58. https://doi.org/10.2991/ijcis.2018.25905181

Younis, R., & Fisichella, M. (2022). FLY-SMOTE: Re-Balancing the Non-IID IoT Edge Devices Data in Federated Learning System. IEEE Access, 10, 65092–65102. https://doi.org/10.1109/ACCESS.2022.3184309