Privacy-Preserving and Resource-Aware Intrusion Detection Using Federated Few-Shot Learning

Ahsan Saleem

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

October 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ahsan Saleem**

Entitled: **Privacy-Preserving and Resource-Aware Intrusion Detection Using Federated Few-Shot Learning**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Dongyu Qiu

_____ External Examiner
Dr. Amr Youssef

_____ Examiner
Dr. Dongyu Qiu

_____ Supervisor
Dr. Walaa Hamouda

Approved by _____
Dr. Jun Cai, Graduate Program Director
Department of Electrical and Computer Engineering

07-10-2025 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Privacy-Preserving and Resource-Aware Intrusion Detection Using Federated Few-Shot Learning

Ahsan Saleem

With the rapid expansion of Internet of Things (IoT) networks, ensuring secure and privacy-preserving communication has become increasingly important. Anomaly-based Network Intrusion Detection Systems (NIDS), which rely on transmitting large volumes of network data to central servers to train NIDS, are becoming less ideal in IoT contexts due to growing concerns about data privacy and the limitations of centralized processing. Federated Learning (FL) offers a promising alternative by enabling decentralized model training across distributed devices without the need to share data. However, FL alone often struggles due to the limited availability of labeled intrusion data in real-world settings. To address these limitations, this thesis proposes a Federated Few-Shot Learning (FFSL) framework that integrates FL with Few-Shot Learning (FSL) to enhance generalization from a small number of labeled examples while preserving user privacy. The approach leverages a metric-based prototypical network integrated with a Long Short-Term Memory (LSTM) architecture to model temporal patterns in network traffic. Evaluations on the ToN_IoT dataset show that the FFSL framework achieves strong performance in terms of precision, recall, and F1-score.

Despite demonstrating effectiveness in overcoming issues of data privacy and limited labeled data, deploying such models on real-world IoT devices presents several challenges. Most IoT devices operate under strict resource constraints, including limited processing power, memory, and battery capacity, making it difficult to run large deep learning models efficiently. Furthermore, selecting IoT devices for FL training without considering their available resources can lead to suboptimal performance and premature device failure. To address this, the thesis introduces resource-aware lightweight extension of the FFSL framework designed specifically for resource-constrained IoT devices. Light-FFSL incorporates dynamic scheduling to adaptively select participating devices based

on their available resources, and uses convergence-based updates to reduce communication overhead. Moreover, a prune-and-quantize strategy is applied to the LSTM-based prototypical model, significantly reducing model size and inference latency for resource-constrained IoT devices. Experiments conducted on the ToN_IoT and Bot_IoT datasets and the results demonstrate that the proposed model maintains competitive detection performance while improving inference time and reducing model size.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Walaa Hamouda, for her exceptional mentorship, unwavering support, and insightful guidance throughout the course of this research. Her profound knowledge, patience, and encouragement have been crucial in overcoming challenges and refining the direction of this work. Her critical feedback helped elevate the technical depth and academic rigor of the study, and I am truly fortunate to have worked under her supervision.

I would also like to thank the faculty of the Department of Electrical and Computer Engineering at Concordia University for fostering a collaborative and intellectually stimulating research environment. The resources, technical support, and opportunities provided by the department were essential to the successful execution of this work.

I extend my sincere appreciation to my colleagues and lab mates for their constructive discussions, collaboration, and moral support. The exchange of ideas and shared enthusiasm has greatly enriched my learning experience.

On a personal note, I am profoundly grateful to my family for their constant love, patience, and unwavering belief in me. Their emotional and moral support has been the bedrock of my academic journey. I would particularly like to thank my parents and spouse, whose sacrifices and encouragement laid the foundation for my education.

Lastly, I acknowledge all those, named and unnamed, who directly or indirectly contributed to the completion of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The proliferation of Internet of Things (IoT) technologies has transformed the modern digital landscape, enabling real-time sensing, automation, and intelligent control across critical domains such as healthcare, agriculture, transportation, and industrial systems Baccour et al. (2022). Industrial forecasts predict that the number of IoT devices globally will surpass 40 billion by 2030 Shafin, Karmakar, and Mareels (2023). These devices form highly distributed and interconnected networks capable of autonomous operation and data-driven optimization. However, their widespread deployment and often minimal security configurations expose them to a variety of cyber threats Shah and Sengupta (2020). Attacks such as Distributed Denial of Service (DDoS), spoofing, injection, and malware propagation can compromise not only data confidentiality and integrity but also the safety and availability of critical infrastructure.

To mitigate these risks, Network Intrusion Detection Systems (NIDS) have emerged as a vital component of IoT cybersecurity architectures Abdulkareem, Foh, Shojafar, Carrez, and Moessner (2024). By continuously analyzing network traffic for signs of abnormal or malicious behavior, NIDS help identify and respond to threats before they escalate. NIDS models, especially those based on Machine Learning (ML) models, are effective in data-rich, high-resource environments. However, their reliance on centralized data collection for model training, extensive labeled datasets, and high computational power makes them poorly suited for deployment in privacy-sensitive IoT

environments facing evolving cyber threats and operating under resource constraints.

1.2 Motivation

The deployment of NIDS in IoT environments presents several unresolved challenges that limit the effectiveness of existing solutions. Foremost among these is the issue of data privacy K. He, Kim, and Asghar (2023). IoT devices frequently generate sensitive and application-specific data, such as health records, industrial telemetry, or location information, that must be protected against unauthorized access. ML-based NIDS often rely on centralized training paradigms, requiring data to be transferred to a central location for training the model. This not only increases the risk of data leakage but also violates emerging data protection regulations. As a result, there is a growing demand for decentralized learning approaches that preserve privacy by retaining data on local systems.

A second challenge arises from the scarcity of labeled attack data in real-world IoT deployments Al-Shurbaji et al. (2025). In IoT environments, the continuous emergence of novel and evolving attack patterns, combined with heterogeneous traffic behaviors, poses significant challenges to effective data collection and labeling . Supervised learning models, which dominate the current literature, typically require extensive labeled datasets to achieve reliable detection performance. However, the dynamic and distributed nature of IoT systems often results in limited representation of certain attack types, especially novel or zero-day threats, thereby reducing the generalizability of trained models.

Finally, IoT devices are inherently constrained in terms of processing power, memory, and energy availability Chaabouni, Mosbah, Zemmari, Sauvignac, and Faruki (2019), making the deployment of computationally intensive DL models impractical. These limitations underscore the need for lightweight, resource-efficient intrusion detection solutions capable of operating under such constraints. Effectively addressing this challenge is critical for enabling reliable threat detection in resource-constrained IoT environments.

These challenges, data privacy, limited labeled data, and constrained computational resources, collectively motivate the need for a novel intrusion detection framework. Such a framework should

support decentralized training, exhibit robust performance under limited supervision, and remain resource-aware enough for deployment on resource-constrained IoT devices. Addressing these requirements is essential for achieving effective intrusion detection in modern IoT networks.

1.3 Contribution

The main contributions of this thesis are as follow:

- **Privacy-Preserving Collaborative Learning:** We develop a FL-based NIDS that enables IoT devices to collaboratively train an intrusion detection model without sharing traffic data. This design preserves data confidentiality and eliminates the need for centralized model training, making it suitable for privacy-sensitive enviroment.
- **Few-Shot Learning for Efficient Detection with Limited Labels:** To overcome the scarcity of labeled attack data, the framework integrates LSTM-based prototypical FSL model. This allows the system to generalize effectively from a limited number of labelled samples, facilitating the detection of both frequent and previously unseen attack types.
- **Dynamic Scheduling and Convergence-Based Updates:** A dynamic scheduling mechanism is introduced to optimize device participation for FL training based on their resource availability, including energy levels, latency, and computational capacity. Furthermore, a convergence-based update strategy reduces redundant communication and lowering FL training overhead.
- **Lightweight Model via Pruning and Quantization:** To enable deployment on resource-constrained IoT devices, the model incorporates structured pruning and post-training quantization techniques. These methods reduce memory footprint and improves inference time while preserving classification accuracy.
- **Comprehensive Experimental Evaluation:** The proposed framework is evaluated on IoT datasets e.g. ToN_IoT and Bot_IoT. Results demonstrate that the proposed frameworks achieves competitive detection performance while significantly reducing communication cost, model size, and improves inference time.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents the necessary background and related work, covering foundational concepts in NIDS, FL, FSL, and model compression techniques such as pruning and quantization. It also surveys existing approaches and identifies their limitations in the context of privacy, data scarcity, and resource constraints in IoT environments. Chapter 3 introduces the Federated Few-Shot Learning (FFSL) framework, describing its architecture, learning process, and the use of LSTM-based prototypical networks to address the challenges of decentralized learning and limited labeled data. Chapter 4 extends this work by proposing the resource-aware Lightweight Federated Few-Shot Learning (Light-FFSL) framework, which incorporates dynamic device scheduling, convergence-based updates, and lightweight model optimization to enable efficient deployment on resource-constrained IoT devices. Chapter 5 concludes the thesis by summarizing the main contributions, and outlining potential directions for future research in secure and adaptive intrusion detection for IoT networks.

Chapter 2

Background and Literature Review

This chapter presents the foundational background and relevant literature that contextualize the proposed research. It begins by discussing network intrusion detection and their inherent limitations in IoT environments. The discussion then introduces FL as a privacy-preserving alternative to centralized training, followed by an overview of the FSL technique that addresses the challenge of limited labeled data. Compressed model design strategies, such as pruning and quantization, are also discussed in the context of deployment on resource-constrained IoT devices. The chapter concludes with a critical review of existing work and highlights the key research gap addressed by this thesis: the absence of a unified framework that effectively combines privacy preservation, data efficiency, and computational scalability for intrusion detection in IoT networks.

2.1 Network Intrusion Detection in IoT

Network intrusion detection is a critical aspect of securing networked systems, including IoT environments Rahman, Al Shakil, and Mustakim (2025). A NIDS monitors network traffic to detect signs of malicious activity, unauthorized access, or violations of security policies. In IoT networks, where devices are geographically spread, heterogeneous in design, and often lack built-in security features, NIDS are typically deployed at gateways, edge nodes, or cloud aggregation points to provide continuous traffic analysis and real-time threat detection.

NIDS techniques are generally classified into two primary categories Abdulganiyu, Tchakoucht,

and Saheed (2024):

- **Signature-based detection:** This method relies on a database of known attack signatures. Traffic is analyzed and compared to these predefined patterns, allowing rapid and accurate identification of previously encountered threats. While efficient and effective for known attacks, this approach cannot detect novel, evolving, or zero-day threats, and requires regular updates to maintain its efficacy.

- **Anomaly-based detection:** This methods learn the statistical or behavioral profile of normal or attack network traffic and identify deviations from normal pattern as potential intrusions. These techniques are more flexible in detecting unknown or sophisticated attacks. They have been widely applied in various network environments, ranging from enterprise systems to large-scale critical infrastructures.

Although these NIDS approaches have proven successful in many applications, their direct application to IoT networks is fraught with challenges. Signature-based detection lacks adaptability, while anomaly-based systems suffer from high computational demands and a reliance on data that is often unavailable in IoT contexts Rajapaksha et al. (2023) Chou and Jiang (2021). More critically, IoT devices operate under tight resource constraints which preclude the deployment of large DL models directly on-device Chaabouni et al. (2019). Additionally, NIDS typically assume centralized architectures where traffic data is collected and analyzed in a central server or cloud. Transmitting sensitive traffic data off-device not only exposes it to potential interception but also consumes precious bandwidth and energy K. He et al. (2023).

2.2 Federate Learning

FL is a distributed ML approach designed to enable multiple decentralized clients, such as IoT devices, mobile phones, or edge nodes, to collaboratively train a shared global model without exchanging their data Nguyen et al. (2021). Instead of centralizing training data, each device trains a model locally using its private dataset and transmits only model updates (e.g., weights or gradients) to a coordinating server. The server then aggregates these updates, typically using algorithms like

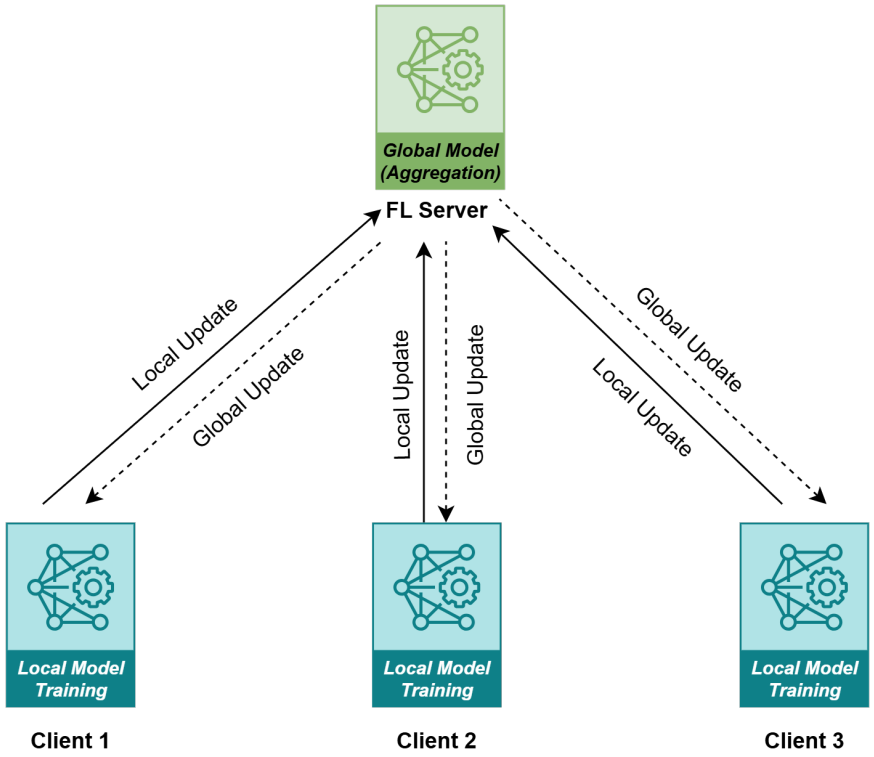


Figure 2.1: Federated Learning.

Federated Averaging (FedAvg) Lim et al. (2020) and redistributes the updated global model to all clients for the next training round, as shown in Figure 2.1.

The core idea behind FL is to support collaborative learning while preserving data locality, reducing privacy risks, and improving communication efficiency. This is particularly beneficial in IoT environments, where devices often handle sensitive information, and are bandwidth-constrained. FL enables intelligent modeling across distributed devices without violating data privacy or incurring the communication cost of centralizing massive, heterogeneous datasets.

The key advantages of FL in IoT scenarios include: **Privacy preservation:** Sensitive data remains on-device, significantly reducing the risk of data leakage or regulatory violations. **Communication efficiency:** Only compact model updates are transmitted, making FL suitable for devices operating under limited bandwidth conditions. **Decentralized intelligence:** FL enables collaborative learning while allowing devices to retain knowledge unique to their operational context.

2.3 Few-Shot Learning

FSL is an ML paradigm designed to enable models to learn and generalize from a very limited number of labeled examples. Unlike supervised learning approaches, which require large volumes of labelled data to achieve acceptable performance, FSL aims to recognize new classes or tasks with only a few of samples per category Song, Wang, Cai, Mondal, and Sahoo (2023). This learning paradigm is particularly relevant in network intrusion detection for IoT environments, where labelled examples of rare or emerging cyberattacks are scarce and labeling new data is time-consuming, expensive, and often impractical.

FSL methods can be broadly categorized into three main classes Parnami and Lee (2022): Metric-based method, learn an embedding function that maps input samples into a feature space where classification can be performed by comparing distances between representations. Optimization-based method, aim to learn model parameters or initialization strategies that allow for fast adaptation to new tasks using just a few gradient descent steps. Model-based method, introduce architectural enhancements, such as memory modules, dynamic weights, or external controllers, to adapt to new tasks with few examples.

2.4 Prototypical Network

In this thesis, we adopt a prototypical network, a widely used metric-based FSL approach that is both computationally efficient and well suited for deployment on resource-constrained IoT devices. Prototypical networks operate on the principle of learning a shared embedding space in which each class is represented by a prototype vector, typically computed as the centroid (mean vector) of its support examples in the embedding space Snell, Swersky, and Zemel (2017). During inference, query samples are classified by measuring their distance, often using Euclidean distance, to each class prototype. This simple yet effective strategy allows the model to make reliable predictions even when trained on very limited labeled data.

A key advantage of this architecture is its support for episodic training, where each episode mirrors an FSL scenario by sampling a small set of support and query examples per class. This approach

aligns training with inference, improving generalization to unseen classes, where rare or novel attacks often arise. To capture temporal dependencies in network traffic, the prototypical network integrates an LSTM encoder, which learns sequence-aware embeddings that help identify subtle, time-based patterns indicative of malicious activity.

2.5 Model Compression Techniques

DL models have achieved performance in a variety of tasks, including image recognition, speech processing, and network intrusion detection. However, their application in IoT environments is significantly constrained by device limitations such as low memory capacity, limited computational power, and strict energy budgets. Deploying large, complex models on such devices without optimization often results in infeasible inference times, excessive power consumption, or outright failure due to insufficient resources.

To address these limitations, two model compression techniques, structured pruning and post-training quantization Liang, Glossner, Wang, Shi, and Zhang (2021), are widely adopted in resource-constrained environments to reduce model size, enabling efficient deployment of DL models on constrained edge and IoT devices.

2.5.1 Structured Pruning

Structured pruning is a model compression technique that removes entire structures within a neural network, such as neurons, filters, attention heads, or recurrent units, based on their relative importance Cheng, Zhang, and Shi (2024). This is in contrast to unstructured pruning, which removes individual weights and typically results in sparse but irregular weight matrices that are less efficient on standard hardware.

Structured pruning maintains the original architecture's high-level structure, making the pruned model more compatible with standard computational libraries. The reduction in model complexity leads to improvements in inference speed, memory footprint, and energy efficiency, which are especially valuable in resource-constrained environments. The criteria for pruning can be based on: the magnitude of weights or activation values, sensitivity analysis (e.g., how much removing a

component affects accuracy), or learning-based approaches. Once the least important structures are identified and pruned, the model is typically fine-tuned on the training data to recover any performance lost during pruning.

2.5.2 Post-Training Quantization

Post-training quantization is a technique used to reduce the computational and memory requirements of a DL model by lowering the numerical precision of its parameters and activations. Standard models use 32-bit Floating-Point (FP32) values, which consume substantial memory and require relatively expensive arithmetic operations. Quantization replaces these with lower-precision representations, most commonly 8-bit Integers (INT8) Kulkarni et al. (2022). This conversion significantly reduces the model's memory footprint and can improve inference speed, particularly on hardware optimized for integer arithmetic such as mobile CPUs, or microcontrollers.
Post-training quantization does not require retraining, making it suitable for compressing pre-trained models quickly and efficiently. While quantization may introduce slight accuracy degradation, modern calibration and quantization-aware training techniques help to minimize this loss, making quantization a practical choice for deploying DL models in resource-constrained settings.

2.6 Related Work

Despite notable advancements, deploying NIDS in resource-constrained IoT environments remains a significant challenge due to computational overhead, data privacy concerns, and limited availability of labeled data. Gyamfi et al. Gyamfi and Jurcut (2022) proposed a lightweight NIDS combining One-class Support Vector Data Description (OI-SVDD) and Extreme Learning Machine (AS-ELM) with Multi-access Edge Computing (MEC) support to offload computations, while Zhao et al. Zhao et al. (2021) introduced a neural network-based NIDS with Principal Component Analysis (PCA)-based feature reduction. Yang et al. proposed NIDS-CNNRF Yang et al. (2025), which integrates Convolution Neural Network (CNN) for feature extraction and Random Forest (RF) for classification, with PCA for redundancy removal and ADASYN for data imbalance, improving detection efficiency in IoT networks. Similarly, the Gaussian Mixture Cramér-Wold auto-encoder

(GMCWAE) Bian and Liu (2025) framework combines Gaussian mixture priors with Wasserstein autoencoders to extract compact yet expressive features, thereby reducing the computational burden in constrained IoT environments. However, while these approaches improve efficiency, they do not incorporate privacy-preserving mechanisms nor address the scarcity of labeled data, making them insufficient for data-sensitive and evolving network attacks in IoT deployments.

Tong and Zhang Tong and Zhang (2024) addressed label scarcity via self-supervised learning; however, their method lacks privacy-preserving mechanisms. Similarly, Few-Shot (FS)-IDS Ouyang, Li, Kong, Song, and Li (2021) introduced a CNN-based few-shot learning model that achieved high detection accuracy in SCADA networks, yet did not support privacy-preserving, limiting its suitability for privacy-sensitive IoT environments. Zha et al. introduced A-NIDS Zha et al. (2025), an adaptive IDS that uses clustering with stacked CTGANs to generate synthetic traffic and mitigate catastrophic forgetting. Wu et al. proposed Temporal Contrastive Graph (TCG)-IDS C. Wu et al. (2025), a temporal contrastive graph-learning model that learns robust temporal embeddings in a self-supervised manner, reducing reliance on large labeled datasets. While both methods effectively address the challenge of limited labeled data, they remain computationally heavy and lack integration with privacy-preserving mechanisms, which limits their applicability to resource-constrained IoT environments.

FL-based NIDS offers improved privacy. Friha et al. Friha et al. (2022) presented Federated Learning-based Intrusion Detection System (FELIDS) for agricultural IoT, and Bhale et al. Bhale, Chowdhury, Biswas, and Nandi (2023) developed Optimum Placement Strategy to Mitigate mixed-rate DDoS attack in IoT System (OPTIMIST) for mixed-rate DDoS mitigation using LSTM and Generative Adversarial Network (GAN)-generated data. While effective, both approaches still depend on extensive labeled datasets. FL enhancements via knowledge distillation Zhao et al. (2022) and transfer learning Zhang, Luo, Carpenter, and Min (2022) improve adaptability but increase computational demands, limiting suitability for low-power devices. Benameur et al. Benameur, Dahane, Souihi, and Mellouk (2024) and Aljuhani et al. Aljuhani, Alamri, Kumar, and Jolfaei (2023) offer compressed or explainable FL-based NIDS, but label efficiency remains a concern.

Semi- and self-supervised FL-based models such as FL-SSAL (Federated Learning-Empowered

Semi-Supervised Active Learning) Naeem, Ali, and Kaddoum (2023), and Semi-supervised Spatiotemporal DL model (SS-Deep-ID Abdel-Basset, Hawash, Chakrabortty, and Ryan (2021) address labeled data scarcity but remain resource-intensive. Yao et al. Yao, Peng, Li, and Shen (2025) proposed a Packet-level Adversarial Traffic Generation (PATG) framework to evaluate the robustness of deep learning-based NIDS under realistic adversarial traffic in Industrial IoT. In addition, an interaction-aware trust management scheme combines ML-based intrusion detection with device reputation scoring to mitigate on–off and bad-mouthing attacks in IoT systems Farhat, Eldosouky, Ibnkahla, and Matrawy (2025). Although these works enhance robustness and privacy in adversarial and trust-aware contexts, they still overlook resource-efficiency and the data scarcity dimension, which are critical in IoT deployments.

In summary, while existing approaches address parts of the problem space, few simultaneously ensure lightweight design, robust privacy preservation, and effective learning from limited labeled data, highlighting the need for integrated solutions.

Chapter 3

Federated Few-Shot Learning for Privacy-Preserving and Robust Network Intrusion Detection

3.1 Introduction

In the previous chapter, we presented the background and foundational concepts relevant to intrusion detection in IoT environments, including FL, FSL, and model optimization techniques. We also reviewed the limitations of existing NIDS approaches in addressing key challenges: preserving data privacy, mitigating labeled data scarcity, and operating under computational constraints in IoT networks. In this chapter, we introduce the proposed Federated Few-Shot Learning (FFSL) framework, which combines the decentralized, privacy-preserving capabilities of FL with the data-efficient generalization of FSL to enable robust intrusion detection across distributed IoT devices. With the growing adoption of intelligent IoT systems, FL has emerged as a promising approach for collaborative model training without sharing data, thereby preserving sensitive information. At the same time, FSL addresses the challenge posed by evolving attacks with few labeled examples by enabling the model to learn effectively from a limited number of labeled attack samples. We detail the integration of FL with an LSTM-based prototypical FSL model tailored for NIDS, illustrating

how this integration enables efficient detection of both frequent and previously unseen attacks while maintaining data privacy. The rest of this chapter covers the system and threat models, followed by the FFSL-NIDS framework, the performance evaluation, and finally the conclusion.

3.2 System Model

As illustrated in Fig. 3.1, we consider an IoT network where multiple IoT nodes communicate with edge gateways. Each edge gateway monitors network traffic and performs intrusion detection using a NIDS. These edge gateways are connected to a cloud infrastructure for further data aggregation and management.

The IoT network comprises of N IoT nodes, denoted as $D_1, D_2, \cdots, D_N$. At any given time t, each IoT node generates network traffic represented by $x_i(t)$, where i indicates the specific IoT node. This traffic includes various network parameters such as packet size, communication protocol, timestamp, etc. The traffic $x_i(t)$ generated by each node can be formalized as: $x_i(t) = \{p_1, p_2, \cdots, p_k\}, \quad p_k \in \mathbb{R}^d$ where p_k represents the network parameters of the k-th packet in the traffic, and d is the number of network parameters. The IoT nodes transmit their network traffic to an edge gateway G_j, where $j = 1, 2, \ldots, M$ denotes the specific gateway. Over a time period T, the total communication flow from an IoT node D_i to a gateway G_j can be described as:

$$\text{Flow}(D_i, G_j) = \sum_{t=1}^{T} x_i(t) \tag{1}$$

where T is the total duration of the communication, and $x_i(t)$ represents the network traffic sent during that time.

Each edge gateway G_j aggregates the network traffic from multiple IoT nodes in its domain $\mathcal{D}_j$, which is the set of IoT nodes communicating with gateway G_j. The total traffic $X_j(t)$ received by the edge gateway G_j from all IoT nodes in $\mathcal{D}_j$ at time t is given by:

$$X_j(t) = \sum_{D_i \in \mathcal{D}_j} x_i(t). \tag{2}$$

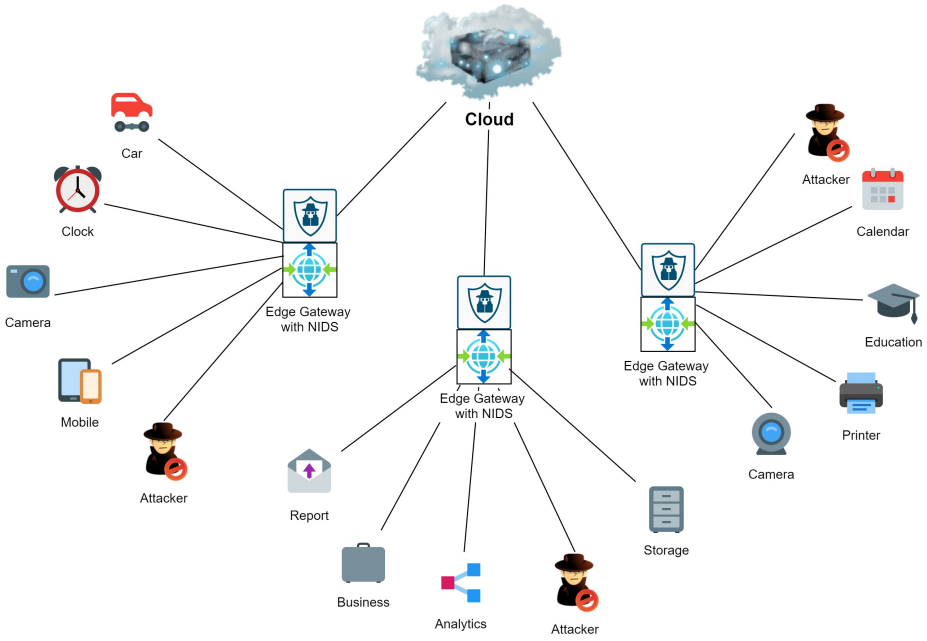


Figure 3.1: System Model.

Once the edge gateway receives the network flow from each node $D_i \in \mathcal{D}_j$, the NIDS at the gateway analyzes the traffic. It directly processes the network flow $x_i(t)$, examining each flow for potential signs of malicious activity. The intrusion detection process at gateway G_j can be modeled as the task of classifying the traffic $x_i(t)$ from node D_i, where the NIDS computes a classification score $s_i(t)$ based on the network flow:

$$s_i(t) = f_{\theta_j}(x_i(t)) \tag{3}$$

where f_{θ_j} is the detection model employed by the NIDS at gateway G_j, parameterized by θ_j. The score $s_i(t)$ indicates whether the traffic $x_i(t)$ is classified as benign or malicious.

3.2.1 Threat Model

In this threat model, we distinguish between common threats, frequent attacks that the NIDS has been trained to detect, and evolving or new threats, which are uncommon or novel attacks that the NIDS has not encountered before.

(1) **Common threats:** are those threats that are represented in the local training dataset at edge gateway G_i denoted by X_{train}. This dataset consists of samples representing frequent, known

attacks and is defined as $X_{train} = \{(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \cdots, (x_{i_{n_i}}, y_{i_{n_i}})\}$. Here $x_{i,j} \epsilon \mathbb{R}^d$ is the feature vector for the j-th sample, representing parameters of network traffic such as packet size, protocol type, and source/destination IP address. The corresponding label, $y_{i,j} \epsilon \{0, 1, ..., k_f\}$, indicates whether the traffic is benign or malicious. Here, $y_{i,j} = 0$ denotes benign traffic, while $y_{i,j} > 0$ indicates an attack, with k_f being the number of frequent attack classes.

(2) **New threats:** refer to uncommon or novel attack types that the system has never seen during training. For each new threat type, the network flow is represented by $x_q(t)$, where $x_q \epsilon \mathbb{R}^d$ is the feature vector of an unseen attack at time t and y_q provides the labels. This setup highlights the challenge of generalization. The NIDS, trained primarily on familiar, frequent attacks, must now detect and classify previously unseen threats using only a few labeled examples that were not part of the model's training data.

3.3 The Proposed FFSL-NIDS

In this chapter, we propose an FFSL framework for NIDS in IoT networks. The FFSL framework combines FL for collaborative model training without sharing data. Meanwhile, FSL detects new attacks using only a few labeled examples. By leveraging an LSTM network, the framework captures temporal dependencies in network traffic. This approach enhances both detection accuracy and generalization to novel attacks. In what follows, we detail the proposed framework, as illustrated in Fig. 3.2.

3.3.1 Data Preprocessing

A comprehensive preprocessing approach is applied to IoT network traffic data to prepare it for training and testing in the proposed framework. In what follows, we provide details on feature scaling and encoding methods used in data preprocessing, as well as the approach to data division.

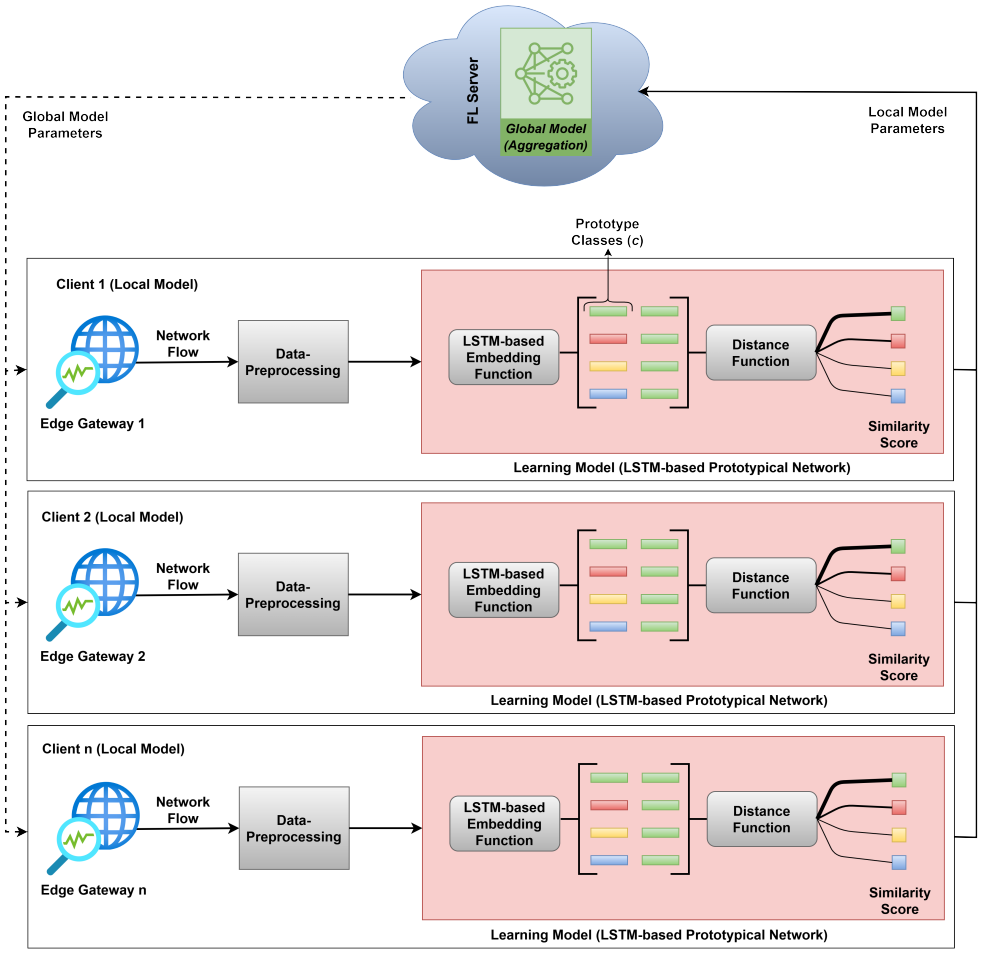


Figure 3.2: FFSL-NIDS Framework.

Feature Scaling and Encoding

IoT network traffic data includes features that vary in scale, such as packet sizes, time intervals, and byte counts, as well as categorical features like protocol types (e.g., TCP, UDP) and flags. To ensure all features contribute equally to model training, standardization is applied using the Standard Scaler technique Raju, Lakshmi, Jain, Kalidindi, and Padma (2020). This method scales numerical features to a mean of zero and a standard deviation of one, which is essential for gradient descent-based techniques when feature scales differ significantly. Additionally, categorical feature encoding is performed using factorization. This method assigns a unique integer ID to each category, resulting in a compact numeric representation instead of a high-dimensional sparse vector. For example, a protocol feature with values like TCP, UDP, and ICMP is transformed into integers such as 0, 1, and

2, significantly reducing input dimensionality and computational overhead. These preprocessing steps ensure that both numerical and categorical data are effectively processed, enabling the model to learn accurately from diverse data types.

Data Division Based on Attack Types

To evaluate the model's ability to generalize to both known and new attack types, the data is split into two sets: (i) The Training Set, which contains frequent, known attack types along with benign traffic. This set enables the model to learn representation patterns. (ii) The Testing Set, which includes previously unseen attack types that were not present during training. This setup allows us to assess the model's capability to detect new threats. It mirrors real-world scenarios in which the model must classify emerging attacks using its FSL capability.

3.3.2 Federated Learning

The FL paradigm ensures that network traffic data remains securely at each edge gateway, thereby addressing privacy concerns by avoiding direct data sharing. In this setup, edge gateways collaborate to train a shared global model without transferring local data. The system operates in communication rounds. During each round, each gateway G_i performs local training on its dataset X_{train} and sends only the computed model updates to a central server. The central server then aggregates these updates to refine the global model, which is subsequently distributed back to each gateway.

Local Model Training

The local model at each gateway G_i is parameterized by θ_i, and the objective is to find the optimal parameters θ_i^* that minimize a general loss function $\mathcal{L}_i(\theta_i)$ over the local dataset X_i:

$$\theta_i^* = \arg\min_{\theta_i} \mathcal{L}_i(\theta_i) \tag{4}$$

where $\mathcal{L}_i(\theta_i)$ represents the loss function. The local model θ_i is updated by minimizing the loss function via gradient descent:

$$\theta_i^{t+1} = \theta_i^t - \eta \nabla \mathcal{L}_i(\theta_i^{(t)}) \tag{5}$$

where η is the learning rate and $\nabla \mathcal{L}_i(\theta_i^{(t)})$ is gradient loss for current parameters. Each gateway updates its local model parameters by computing the gradient $\nabla \mathcal{L}_i(\theta_i)$ and adjusting the model weights iteratively until it converges to an optimal solution θ_i^*.

Federated Average

After local training, each edge gateway sends its updated model parameters θ_i to the central server. The central server aggregates these local updates using the Federated Averaging (FedAvg) algorithm, defined by:

$$\theta_{\text{global}}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{\sum_{i=1}^{N} n_i} \theta_i^{(t)} \tag{6}$$

where n_i is the number of training samples at gateway G_i, and $\theta_{\text{global}}^{(t+1)}$ represents the updated global model at iteration $t + 1$. The central server then sends these global model parameters back to the gateways for further local training or real-time deployment.

3.3.3 The Learning Model

To identify new, unseen attacks with only a few examples, we integrated an FSL approach, specifically a prototypical network, with an LSTM-based model. This integration enables the model to classify novel attacks based on a limited number of labeled examples. In this learning model, each edge gateway G_i is presented with a few-shot task consisting of a support set and a query set. The support set S contains a few labeled examples for each class, while the query set Q contains unlabeled examples. For a given class k, the support set S for class k be defined as $S_k = \{(x_s^1, y_s^1), (x_s^2, y_s^2), ..., (x_s^K, y_s^K)\}$, where x_s^i is the feature vector for the i-th support example, and $y_s^i = k$ is the corresponding class label. The query set Q is defined as $Q = \{x_q^1, x_q^2, ..., x_q^M\}$,

where x_q^M represents the feature vector for the M-th example in the query set. The LSTM network processes the sequential data (i.e., network traffic flows) to generate embeddings for each support and query example. The embedding $f_\theta(x_t)$ for each traffic flow x_t is computed according to:

$$h_t = \text{LSTM}(x_t, h_{t-1}, \theta) \tag{7}$$

where h_t is the hidden state at time step t, θ represents the LSTM parameters. The final hidden state h_T serves as the embedding for the entire sequence. The prototypical network then uses these embeddings to generate a prototype for each class k. The class prototype c_k is computed by averaging the embeddings of the support examples:

$$\mathbf{c}_k = \frac{1}{K} \sum_{i=1}^{K} f_\theta(x_s^i) \tag{8}$$

where K is the number of labeled examples in the support set.

These prototypes are then used to classify new, unlabeled attacks by comparing each query sample to the corresponding class prototypes. Given a prototype c_k and samples x_q from the query set Q, the LSTM processes each query traffic sample x_q to generate an embedding $f_\theta(x_q)$. Each query sample x_q is classified by calculating the Euclidean distance between its embedding $f_\theta(x_q)$ and the class prototypes c_k as follows:

$$y_q = \arg\min_k \| f_\theta(x_q) - \mathbf{c}_k \|_2 \tag{9}$$

where $f_\theta(x_q)$ is the embedding for the query sample, c_k is the prototype for class k and y_q is the predicted class label for the query sample.

In FSL, the model is trained on a support set that contains only a few labeled examples for each class of attack, whether frequent or new. The model's primary objective is to classify query examples based on learned class prototypes. To achieve this, an LSTM network generates embeddings that capture temporal dependencies in network traffic data. A cross-entropy loss function is then used to train the model to learn these embeddings and make accurate predictions. For the FSL model, a

cross-entropy loss function is defined as:

$$\mathcal{L}_{\text{few-shot}} = -\sum_{(x_q, y_q) \in Q} \Big(y_q \log p(y_q \mid x_q, \theta) + (1 - y_q) \log(1 - p(y_q \mid x_q, \theta)) \Big) \tag{10}$$

where $p(y_q|x_q, \theta)$ is the predicted probability that the query sample x_q belongs to class y_q, given the model's parameters θ.

The learning model utilizes FSL, which not only enables effective training with limited data but also helps reduce computational overhead requiring fewer training samples, minimizing the computational resources needed for model training.

3.3.4 Real-time Attack Detection

The Real-Time Attack Detection Module analyzes IoT network traffic in real-time through a distributed architecture. Federated clients, or edge gateways, monitor and analyze local traffic independently. Each client segments incoming traffic into flows, preprocesses the data, and performs local detection using a trained model. Although clients operate independently, the global model is periodically updated through FL. Each client sends its model parameters to a central server, where they are aggregated to continuously improve the global model.

3.4 Performance Evaluation

In this section, we discuss the dataset, FL approach, and experimental setup, followed by a comprehensive evaluation of the FFSL-NIDS framework. We first analyze training loss and accuracy across federated clients using frequent attack data. Then, we compare FFSL-NIDS with FS-IDS Ouyang et al. (2021) to assess its performance.

3.4.1 Dataset Description

The ToN_IoT Booij, Chiscop, Meeuwissen, Moustafa, and Den Hartog (2021), developed by the Cyber Range Lab at the Australian Centre for Cyber Security (ACCS), simulates complex IoT environments for evaluating NIDS in distributed networks. It includes data from various IoT devices

operating on Windows, Linux, and MacOS. In this study, we focused on the network traffic subset to detect network-based attacks. The dataset comprises eight attack types: scanning, Denial of Service (DoS), injection, Distributed Denial of Service (DDoS), password, Cross-Site Scripting (XSS), ransomware, and backdoors Booij et al. (2021).

3.4.2 Federated Learning and Experiment Setup

For the FL setup, we implemented a framework with five clients, each representing an independent IoT edge gateway. The dataset was divided into five subsets, allowing each client to train its model locally and maintain data privacy. After local training, only the model parameters were sent to a central server, where the FedAvg algorithm was used to aggregate and update the global model. This process was repeated over multiple communication rounds, preserving privacy and reducing bandwidth usage in distributed IoT environments.

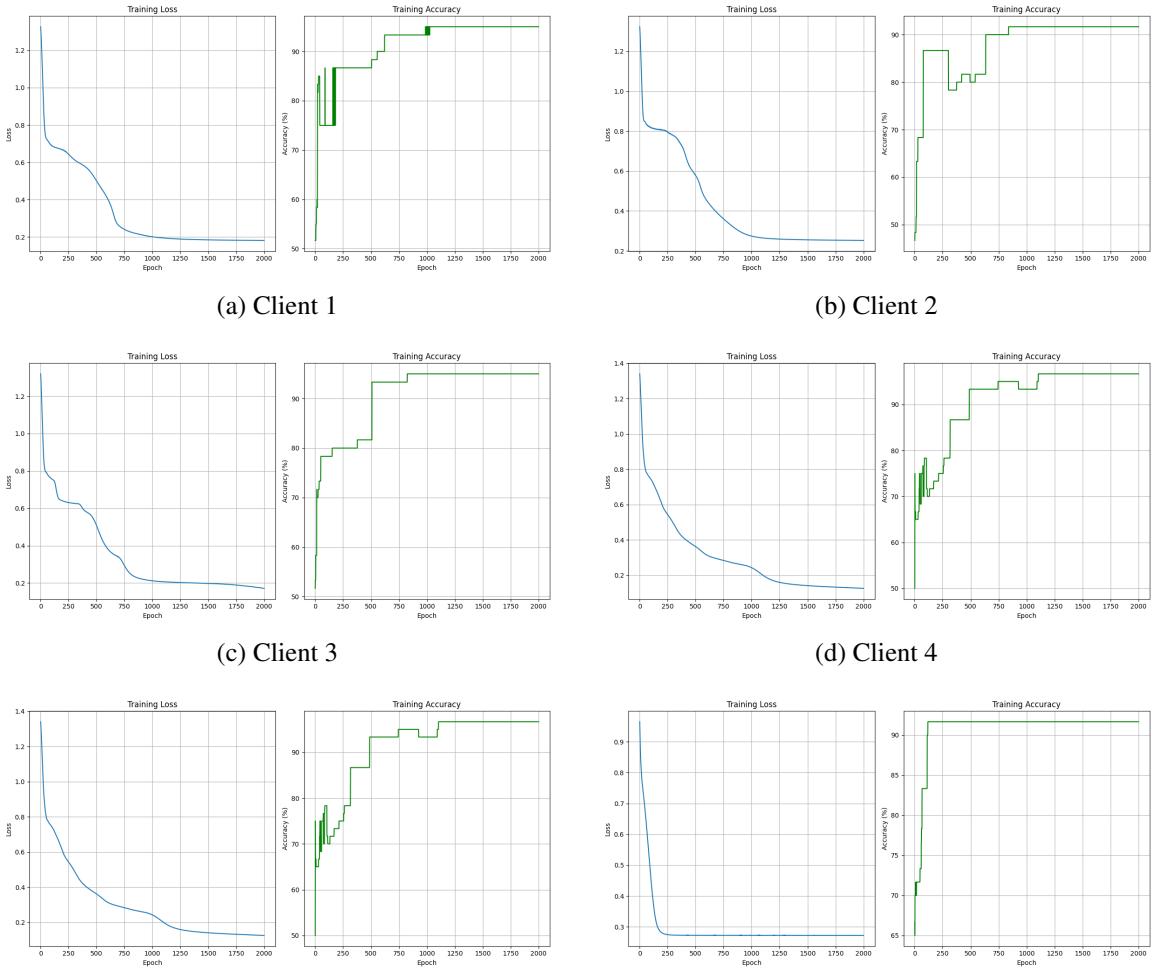
Before applying FL, we split the dataset by attack types. The four most frequent attack types were used to train the FFSL-NIDS model, while the remaining four were reserved for testing. This setup simulated the detection of new, unseen attacks. To capture the sequential nature of network traffic data, we used a bidirectional LSTM architecture with an input dimension of 128, 64 hidden units, and two LSTM layers.

3.4.3 Numerical Results

During training, we evaluated the model's performance using training loss and accuracy metrics. Each client model was trained individually for 2,000 epochs to ensure thorough learning, given the limited samples available per epoch. After completing local training, the client models were aggregated at the FL server. The global model was then tested on one client to assess overall performance.

Figure 3.3 illustrates the training loss and accuracy of the FFSL-NIDS model across all clients. The results show a significant decrease in training loss, starting from approximately 1.2 to 1.4 and converging to between 0.2 and 0.3 within the first 500 epochs. This trend reflects effective learning and error minimization. The global model converges even faster, stabilizing at a loss of around 0.2 and consistently achieving an accuracy above 90%, demonstrating the advantages of

FL. Additionally, the global model's training loss shows a smoother and more stable reduction, benefiting from the combined knowledge of all clients. The consistent training curves across clients suggest that the models learn at similar rates, even with variations in local datasets.



(a) Client 1

(b) Client 2

(c) Client 3

(d) Client 4

(e) Client 5

(f) Global Model

Figure 3.3: FFSL-NIDS model training loss and accuracy across five clients and the global model.

Figure 3.4 presents a comprehensive analysis of training time across various support-query configurations in the FFSL-NIDS framework. For each (k, q) pair, where $k = 1, 3, 5, 10, 15$ and $q = 5, 10, 15, 20, 30$, the model was trained independently on five clients to assess the training time of the FFSL model over different few-shot configurations. For example, in the $(k = 1, q = 5)$ configuration, individual client training times were 20.89, 19.36, 21.75, 19.80, and 19.29 seconds, resulting in an average of approximately 20.22 seconds. As expected, configurations with higher

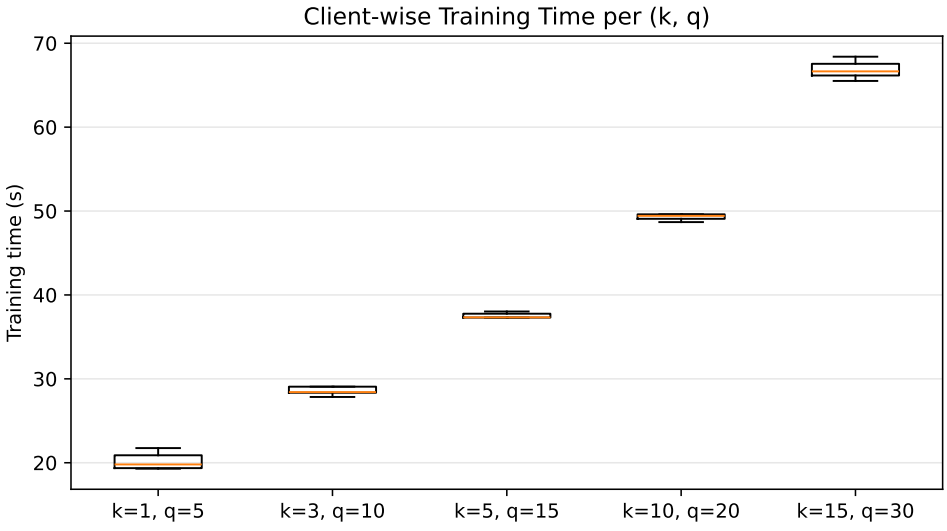


Figure 3.4: Training time (in seconds) for each client and the global model.

support and query values led to increased training times, with the $(k = 15, q = 30)$ setting averaging around 66.45 seconds across clients. Despite this growth, the training time for each client remained low, straying under 70 seconds even as the number of few-shot examples increased substantially, highlighting the robustness of the FFSL approach. Furthermore, the parallelism enabled by the FL setup significantly reduces overall training time while ensuring privacy preservation and scalability.

Figure 3.5 and 3.6 presents a comparative evaluation of the proposed FFSL-NIDS and the FS-IDS using Precision–Recall (PR) and Receiver Operating Characteristic (ROC) curves. In the PR analysis, FFSL-NIDS achieves a higher area under the curve (AUC = 0.903) than FS-IDS (AUC = 0.880), indicating improved precision–recall trade-offs across a wide range of recall values. This advantage is most evident in the mid-recall region, where FFSL-NIDS sustains higher precision, demonstrating its capability to reduce false positives while preserving strong recall performance. Similarly, the ROC results in Figure 3.6 demonstrate an AUC of 0.902 for FFSL-NIDS compared to 0.876 for FS-IDS, highlighting its enhanced ability to distinguish between benign traffic and various attack types. The performance gains are due to the use of an LSTM-based architecture in FFSL-NIDS, which captures temporal dependencies in sequential network traffic more effectively than the CNN-based FS-IDS. By leveraging these temporal patterns, FFSL-NIDS achieves consistently higher true positive rates for equivalent false positive rates, thereby demonstrating its robustness

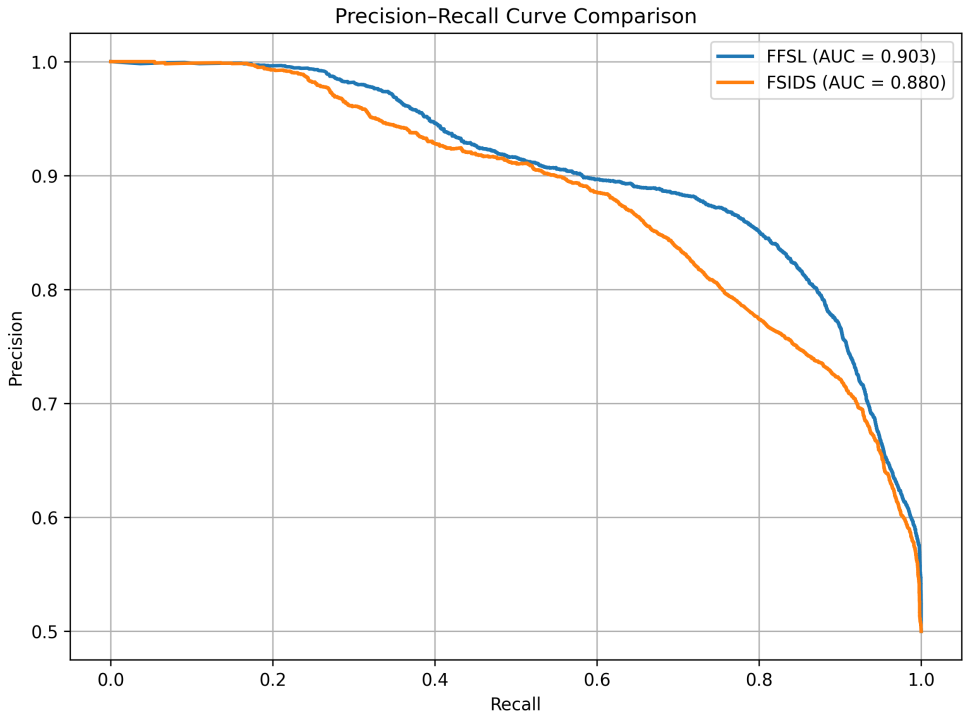


Figure 3.5: PR curves comparing FFSL-NIDS and FS-IDS.

and effectiveness in network intrusion detection.

To further evaluate FFSL-NIDS's effectiveness in detecting new, unseen attacks, we varied the number of queries Q (5, 10, 15, 20, and 30) and shots S (1, 3, 5, 10, and 15). This variation allowed us to assess the model's ability to generalize to previously unseen attack patterns.We performed multiple iterations for each configuration and calculated the average score to ensure a robust and reliable evaluation. Performance was measured using macro-precision, macro-recall, and macro-F1-score Grandini, Bagli, and Visani (2020), which captured the model's performance in detecting and classifying network intrusions.

As illustrated in Figure 3.7, FFSL-NIDS consistently outperforms FS-IDS across all shot and query configurations in Precision, Recall, and F1-Score. For example, in the 3-shot, 10-query setting, FFSL-NIDS achieves a Precision of 0.86 compared to FS-IDS's 0.80, with the performance gap widening in higher shot and query configurations. FFSL-NIDS also shows a 4-7% improvement in Recall and a 5-6% improvement in F1-Score compared to FS-IDS, especially at higher shot settings. These results indicate that FFSL-NIDS not only achieves higher detection accuracy and robustness

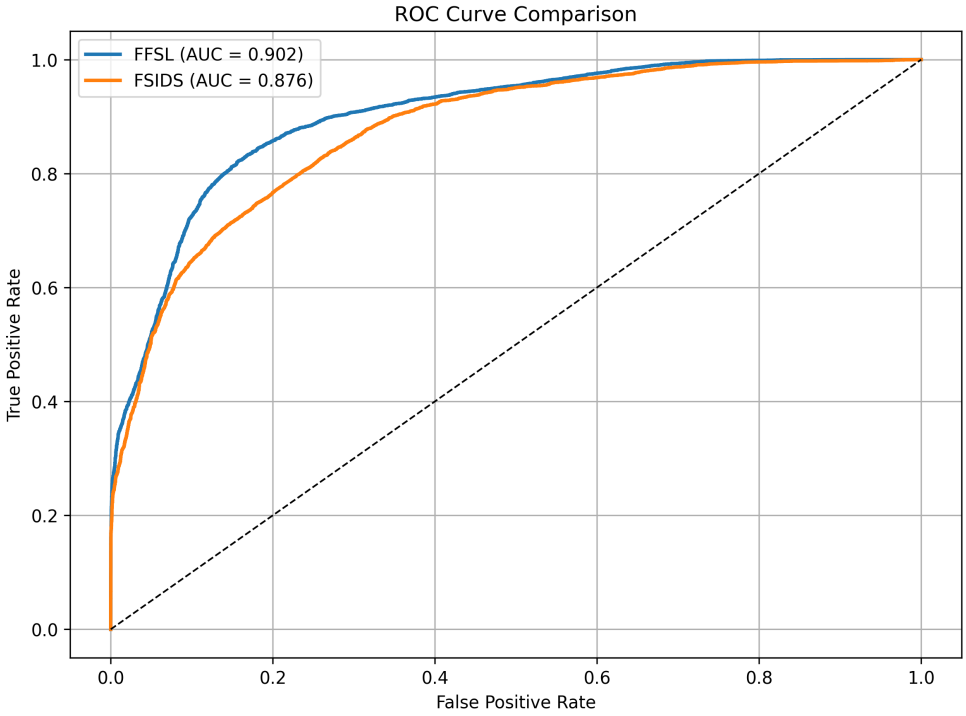


Figure 3.6: ROC curves comparing FFSL-NIDS and FS-IDS.

than FS-IDS but also generalizes effectively to unseen attacks, even in low-shot scenarios. Its stable performance at higher query counts highlights its reliability and scalability, making it ideal for real-world applications where labeled data is limited. Furthermore, by FL learning, FFSL-NIDS learns from distributed attack data while preserving data privacy, a key advantage over FS-IDS.

3.5 Conclusion

In this chapter, we proposed an FFSL framework to enhance NIDS in IoT networks. Our solution addresses key challenges such as data privacy, limited labeled data, and scalability in large-scale IoT networks. Using FL, edge gateways collaboratively train a global model without sharing traffic data. This approach ensures privacy and reduces communication overhead. The integration of FSL further enables the detection of new, unseen attacks with minimal labeled examples. Additionally, an LSTM network is used to capture temporal dependencies in network traffic, enhancing the model's ability to identify suspicious patterns. We evaluate the framework's performance through extensive experiments on the benchmark TON_IoT dataset. Finally, compares its detection of novel attacks with the FS-IDS technique.

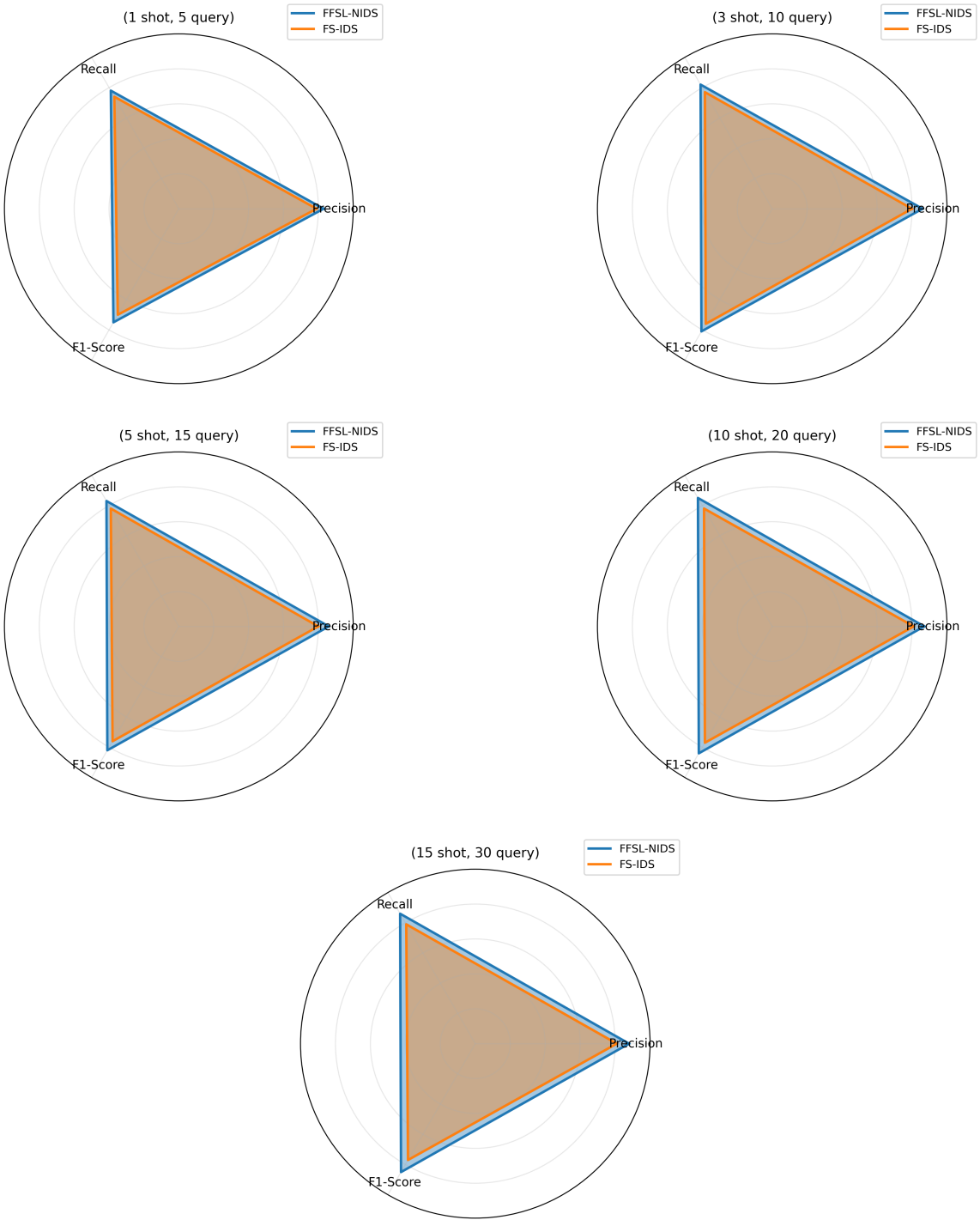


Figure 3.7: Comparison of FFSL-NIDS vs FS-IDS on unseen attack with different k-shot and query values.

Chapter 4

A Resource-Aware FFSL Framework for IoT Network Intrusion Detection

4.1 Introduction

In the previous chapter, we introduced the FFSL framework for network intrusion detection in IoT networks. While FFSL provides privacy-preserving and data-efficient learning capabilities, its deployment on real-world IoT devices remains challenging due to limitations in processing power, memory, and communication bandwidth. In this chapter, we present resource-aware Lightweight Federated Few-Shot Learning (Light-FFSL), an enhanced framework designed to meet the operational constraints of resource-constrained IoT environments. The increasing scale and heterogeneity of IoT deployments necessitates learning approaches that are both sustainable and computationally efficient. The proposed Light-FFSL framework addresses these concerns by incorporating dynamic scheduling to manage device participation for FL training based on their resource availability and convergence-based update strategies to reduce unnecessary communication overhead. Furthermore, to minimize the computational footprint and improves inference time, model compression techniques, e.g., structured pruning and post-training quantization, are applied to the LSTM-based FSL model. This chapter presents the design, implementation, and evaluation of Light-FFSL, demonstrating its effectiveness in detecting network attacks while maintaining computational and communication efficiency.

4.2 System Model and Problem Formulation

4.2.1 System Model

We consider an IoT network environment, as shown in Fig. 4.1, which comprises a set of devices $D_i = \{D_1, D_2, \ldots, D_{N_i}\}, \quad i \in \{1, 2, \ldots, M\}$. Each device D_{ij} generates network traffic over time t, represented as:$x_{ij}(t) = \{p_1, p_2, \ldots, p_k\}, \quad p_k \in \mathbb{R}^d$, where p_k represents the k-th network parameter (e.g., packet size, protocol type, timestamps), and d is the feature space dimensionality. The traffic data at a given time step t follows a probabilistic distribution:

$$X_{ij}(t) \sim P_{x_{ij}}, \quad \forall t \in T. \tag{11}$$

where $P_{x_{ij}}$ represents the underlying distribution of network traffic features. The real-time data stream $X_{ij}(t)$ is used for intrusion detection. Each device is equipped with anomaly-based NIDS that processes network traffic $x_{ij}(t)$ and assigns a classification score $s_{ij}(t) = f_{\theta_{ij}}(x_{ij}(t))$, where $f_{\theta_{ij}}$ is the detection function parameterized by θ_{ij}. The classification decision follows a general decision rule:

$$y_{ij}(t) = g(s_{ij}(t), \tau_{ij}). \tag{12}$$

where $g(\cdot)$ is a general function mapping the detection score $s_{ij}(t)$ to a decision label based on a threshold τ_{ij}.We consider that each IoT device operates under constrained computational resources, meaning that each device has limited resources available for running NIDS. To model these limitations, we define a resource vector for each device as follows:

$$R_{ij} = \{CPU_{ij}, MEM_{ij}, EN_{ij}\}, \quad CPU_{ij}, MEM_{ij}, EN_{ij} \in \mathbb{R}^+. \tag{13}$$

where CPU_{ij} denotes the available processing power, MEM_{ij} represents the available memory, and EN_{ij} is the energy. Each device executes local computations based on available resources. The computational complexity of intrusion detection on each device is modeled as:

$$C_{ij} = \mathcal{O}(T \cdot h^2). \tag{14}$$

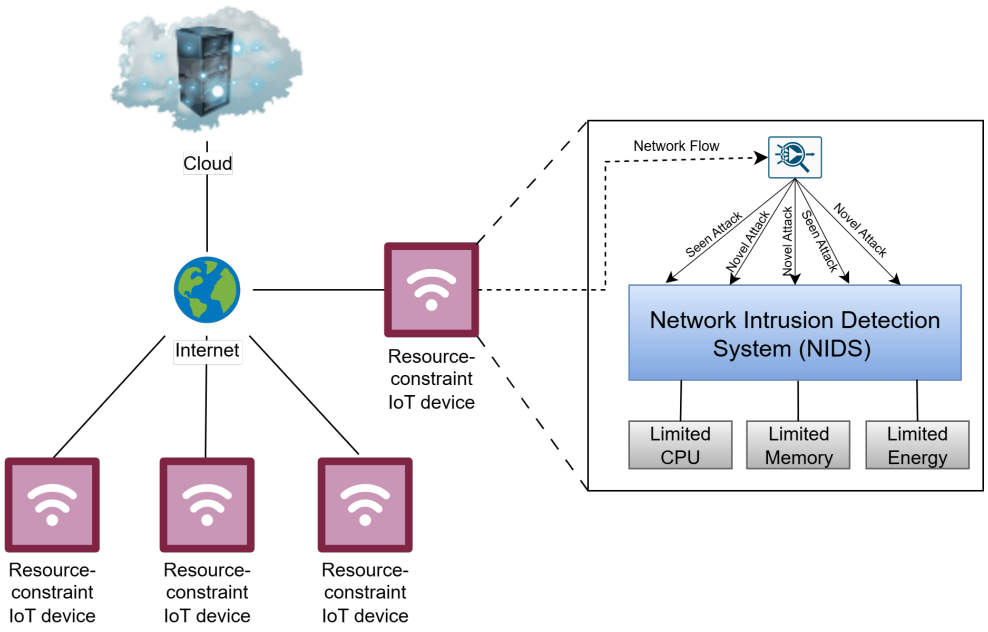


Figure 4.1: System Model

where T denotes the sequence length (i.e., number of time steps per traffic flow), and h is the hidden dimension. The training of NIDS must take data privacy into consideration, ensuring that sensitive network traffic information is not exposed during the learning process. To preserve the confidentiality of data, all training must be conducted locally on the device without exposing sensitive inputs.This is quantified as:

$$\mathbb{E}[(\mathcal{I}_{ij}(t))] \leq \epsilon, \quad \forall t. \tag{15}$$

where $(\mathcal{I}_{ij}(t))$ quantifies the amount of data that could be exposed for training the model, and ϵ represents an upper bound on allowable data sharing.

Given the dynamic and evolving nature of cyberattacks in IoT environments, the system must detect not only previously seen attacks but also emerging threats. Let K_{seen} represent the set of known attack classes for which extensive labeled data is available, and let K_{novel} denote the set of evolving or previously unseen attack classes, for which only a few labeled examples are accessible. Such capability is essential for maintaining robust intrusion detection in the face of continuously evolving cyber threats.

4.2.2 Problem Formulation

Given the system model, our objective is to maximize NIDS effectiveness across IoT devices while minimizing computational overhead, reliance on labeled data, and privacy leakage during training. We therefore formulate a cost-based objective that penalizes classification error, computational cost, labeled-data usage, and information leakage:

$$\min_{\theta} \sum_{i=1}^{M} \sum_{j=1}^{n_i} \Big[\alpha_1 \left(1 - Acc_{ij}(\theta)\right) + \alpha_2 C_{ij} + \alpha_3 L_{ij} + \alpha_4 \Pi_{ij} \Big], \qquad \alpha_k \geq 0. \tag{16}$$

Here, $Acc_{ij}(\theta)$ is the detection accuracy of the model f_θ evaluated on device D_{ij}; C_{ij} denotes the computational cost incurred on D_{ij}; L_{ij} measures the amount of labeled data required on D_{ij}; and Π_{ij} quantifies privacy loss as the information leakage from transmitted information for model training. The nonnegative weights α_k trade off accuracy (via error), computation, label efficiency, and privacy.

To ensure practical feasibility in IoT environments, the optimization problem is subject to the following constraints:

Resource Constraints

To ensure computational feasibility and preserve resources for co-running tasks, we bound the per-device normalized resource usage by predefined thresholds δ. Let $U_{ij}^{\mathrm{CPU}}(t)$, $U_{ij}^{\mathrm{MEM}}(t)$, and $U_{ij}^{\mathrm{EN}}(t)$ denote the CPU, memory, and energy usage of device D_{ij} during training, and let $(\cdot)_{ij}^{\max}$ be the corresponding device-specific budgets. We enforce

$$\frac{U_{ij}^{\mathrm{CPU}}(t)}{\mathrm{CPU}_{ij}^{\max}} \leq \delta_{\mathrm{CPU}}, \quad \frac{U_{ij}^{\mathrm{MEM}}(t)}{\mathrm{MEM}_{ij}^{\max}} \leq \delta_{\mathrm{MEM}}, \quad \frac{U_{ij}^{\mathrm{EN}}(t)}{\mathrm{EN}_{ij}^{\max}} \leq \delta_{\mathrm{EN}}, \qquad \forall\, i, j, t, \tag{17}$$

with $0 < \delta_{\mathrm{CPU}}, \delta_{\mathrm{MEM}}, \delta_{\mathrm{EN}} \leq 1$. Setting these thresholds strictly below 1 preserves a safety margin on each device, preventing overload.

Privacy Constraint

To ensure data privacy during model training, the framework requires that sensitive training data on each device remains local and is not exposed in any form to external entities. Let x_{ij} denote the training data available on device D_{ij}, and u_{ij} denote the information communicated externally for the training. The privacy leakage is quantified as the mutual information between these two quantities:

$$\Pi_{ij} = I\big(u_{ij}; x_{ij}\big). \tag{18}$$

To guarantee privacy preservation, the leakage must remain below a predefined privacy threshold ϵ_{priv}:

$$\Pi_{ij} \;\leq\; \epsilon_{\text{priv}}, \quad \forall\, i, j. \tag{19}$$

This constraint ensures that the training process does not reveal raw data, labeled samples, or any directly reconstructable transformations of the original network traffic, thereby preserving the confidentiality of device-resident data.

Labeled Data Constraint

In practical IoT environments, labeled data is scarce, particularly for evolving attack patterns. To capture this limitation, we impose a constraint on the availability of labeled samples relative to the requirement of conventional supervised training. Let $|\mathcal{T}_L|$ denote the number of labeled training samples available, and $|\mathcal{T}_{\text{ideal}}|$ denote the labeled dataset size typically required for fully supervised NIDS. We define

$$\frac{|\mathcal{T}_L|}{|\mathcal{T}_{\text{ideal}}|} = \frac{k}{N_{\text{large}}}, \quad \text{with} \quad \frac{k}{N_{\text{large}}} \ll 1, \tag{20}$$

where K is the number of target classes, k is the number of labeled samples available per class under constrained conditions, and N_{large} is the number of labeled samples per class typically required for conventional supervised training.

This constraint reflects the reality that only a small fraction of the labeled data needed for supervised learning is available in IoT settings. It emphasizes the need for detection models that remain effective and generalize well under severe label scarcity.

These formulations ensures that the NIDS optimally balances intrusion detection accuracy and computational efficiency, while ensuring privacy preservation and effective learning under few labeled examples. The constraints reflect the real-world challenges of deploying NIDS in resource-constrained IoT environments, where labeled data is scarce, privacy concerns are critical, and computational power is limited.

4.2.3 Threat Model

We categorize the threat model into two categories: frequent threats that the NIDS has been trained to recognize, and new/evolving threats that are uncommon, novel, and unseen in training.

Frequent Attacks

Let $\mathcal{Y}_{\text{freq}} = \{1, \ldots, k_f\}$ denote the set of frequent attack classes, and use 0 to represent benign traffic. The training dataset is then given by $\mathcal{D}_{\text{train}} = \{(x_j, y_j)\}_{j=1}^{n}$, where each feature vector $x_j \in \mathbb{R}^d$ represents a traffic sample and its corresponding label y_j belongs to $\{0\} \cup \mathcal{Y}_{\text{freq}}$.

New and Evolving Attacks

Let $\mathcal{Y}_{\text{new}}$ denote the set of attack classes that are absent during training, with $\mathcal{Y}_{\text{new}} \cap \mathcal{Y}_{\text{freq}} = \varnothing$. The NIDS model is required to detect such novel attacks and, given only a small number of labeled samples, efficiently extend the operational label set. This reflects practical conditions in which the threat landscape continuously evolves, and rare or emerging attacks may not be represented in the training data, with only a few labeled samples available for adaptation.

4.3 Light-FFSL-based NIDS

4.3.1 Framework Overview

In this section, we present a resource-aware Light-FFSL framework tailored for network intrusion detection in privacy-aware and resource-constrained IoT environments. The proposed framework is designed to address three critical challenges: (1) preserving data privacy during model training, (2) enabling learning from limited supervision, and (3) ensuring computational efficiency

on low-power devices.

To ensure data privacy, the framework adopts the FL paradigm, which enables decentralized model training by keeping sensitive data on local devices. This approach significantly reduces the privacy risks associated with centralized data aggregation. To address the limited availability of labeled data, FSL is employed, allowing the model to generalize effectively from a minimal number of labeled examples. To enhance computational efficiency, the framework incorporates three key strategies: dynamic scheduling, convergence-based federated updates, and model pruning and quantization. Dynamic scheduling mechanisms select participating IoT devices for federated training based on their available computational capacity, ensuring balanced workload distribution. Convergence-based update strategies further reduce communication overhead by initiating global model aggregation only when meaningful performance improvements are observed. Additionally, a pruned and quantized LSTM-based prototypical FSL model is implemented to reduce model complexity and memory usage, enabling efficient deployment on resource-constrained IoT devices.

An illustrative Fig. 4.2 provides an overview of the proposed Light-FFSL-based NIDS framework. The subsequent sections detail the core components of the proposed approach.

4.3.2 Federated Learning

FL is a decentralized ML paradigm that enables multiple IoT devices to collaboratively train a shared model without exchanging data Nguyen et al. (2021). Unlike centralized learning, where sensitive data is transmitted to a central server for processing, FL ensures data privacy by keeping training data localized on individual devices. This approach is particularly beneficial in IoT environments, where privacy regulations and limited communication bandwidth make centralized learning impractical.

In the proposed framework, FL is employed to train the intrusion detection model across distributed IoT devices while mitigating privacy risks and reducing communication overhead. The training process follows the standard Federated Averaging (FedAvg) approach Lim et al. (2020), which consists of four main steps: local model training, model aggregation, global model distribution, and secure aggregation to ensure that only the aggregated update is revealed to the server without exposing individual contributions.

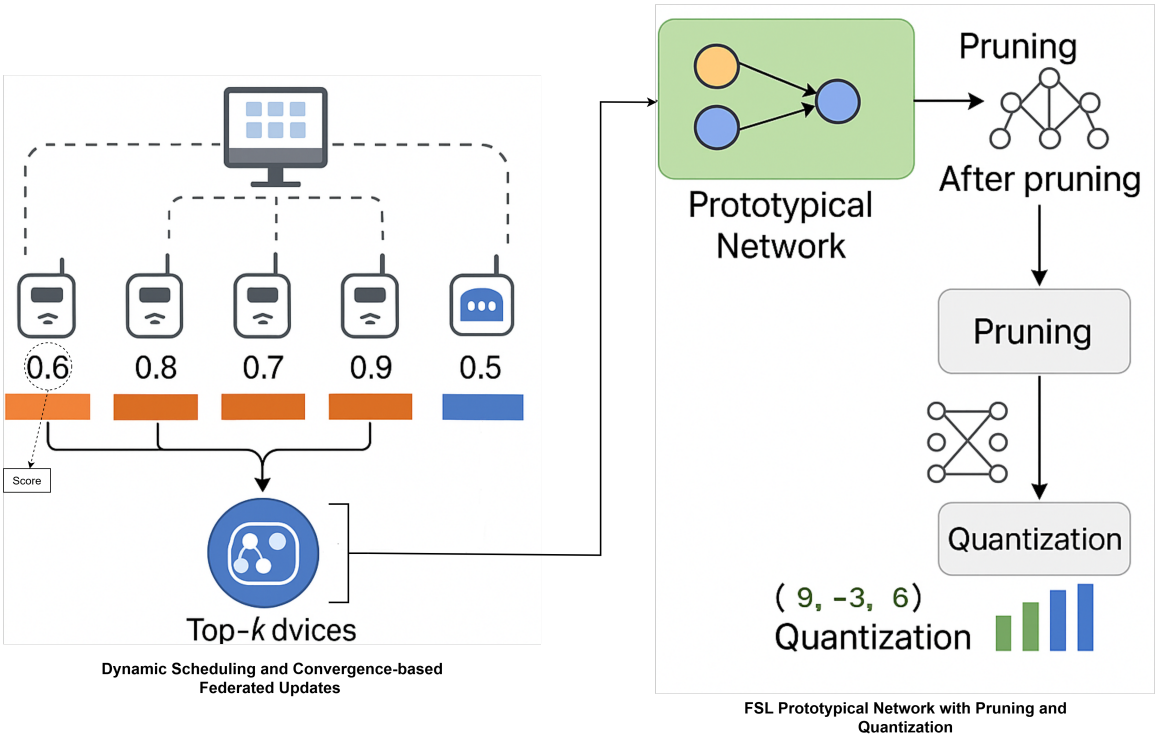


Figure 4.2: Light-FFSL: dynamic scheduling and convergence-based federated updates and few-shot learning with model pruning and quantization.

- *Local Model Training*: Each participating IoT device i trains a local model w_i using its private dataset T_i. The local objective function is defined as:

$$F_i(w) = \frac{1}{|T_i|} \sum_{j \epsilon T_i} \mathcal{L}(w; x_j, y_j) \tag{21}$$

where $\mathcal{L}(w; x_j, y_j)$ represents the loss function for input-output pairs (x_j, y_j).

- *Model Aggregation*: The locally trained models are sent to a central aggregator, which computes the global model update using a weighted average:

$$w_{t+1} = \sum_{i=1}^{N} \frac{|D_i|}{\sum_j |D_j|} w_i^t \tag{22}$$

where N is the number of participating devices, and w_i^t is the locally trained model at round t.

- *Global Model Distribution*: The updated global model w_{t+1} is then redistributed to the IoT devices for the next training iteration, allowing continuous improvement of the model without requiring raw data exchange.

- *Secure Aggregation:* While FL prevents data from leaving IoT devices, individual model updates may still expose sensitive information through gradient inversion or membership inference attacks. To mitigate this risk, FedAvg can be extended with Secure Aggregation (SA). In this mechanism, each client i masks its local update before transmission as

$$\hat{\mathbf{w}}_i^t = \mathbf{w}_i^t + \mathbf{r}_i - \mathbf{r}_{i+1}, \quad \mathbf{r}_{N+1} \triangleq \mathbf{r}_1, \tag{23}$$

  where $\mathbf{w}_i^t$ is the local model update and $\hat{\mathbf{w}}_i^t$ is the masked model update of client i at round t, and $\mathbf{r}_i$ is a random mask generated by client i. Due to the circular cancellation of masks, the server only recovers

$$\sum_{i=1}^{N} \hat{\mathbf{w}}_i^t \;=\; \sum_{i=1}^{N} \mathbf{w}_i^t, \tag{24}$$

  thus concealing individual contributions while still enabling correct global model aggregation.

As the model is trained across distributed devices, maintaining efficiency in resource-constrained environments becomes critical. Allowing all devices to participate equally in every training round can lead to rapid energy depletion, communication overhead, and imbalance in computation. Moreover, device heterogeneity in terms of processing power, availability, and energy levels can significantly impact training effectiveness. To address these challenges, the proposed Light-FFSL incorporates a dynamic scheduling and convergence-based FL Update for FL training.

4.3.3 Resource-Aware Device Scheduling

Standard FL approaches typically assume uniform device participation, which is impractical in heterogeneous and resource-constrained IoT networks where devices differ in energy capacity, computational power, and network conditions. Frequent selection of high-performing devices accelerates convergence but quickly depletes their limited resources, while underutilizing weaker devices introduces bias and unfair workload distribution. To overcome these limitations, the proposed

Light-FFSL framework integrates a dynamic scheduling mechanism that optimizes both the selection and frequency of device participation across FL rounds. Unlike the traditional top-k selection strategy, which randomly chooses devices, dynamic scheduling adaptively modulates participation based on real-time metrics including available energy, latency, computational power, convergence contribution, and availability. This approach ensures fairness, sustains device longevity, and balances workload distribution. Moreover, as the model progresses toward convergence, the mechanism gradually reduces redundant updates by lowering the participation frequency of less impactful devices, thereby minimizing communication overhead and energy consumption while maintaining effective training.

Scoring Metrics for Device Participation

Each device is evaluated using a scoring system that reflects its suitability for participation in the current FL round. The score for each device i is computed as:

$$S_i = \Theta_1 S_E + \Theta_2 S_L + \Theta_3 S_C + \Theta_4 S_{CR} + \Theta_5 S_A. \tag{25}$$

where S_E is the available energy that ensures energy-constrained devices participate less frequently. S_L is a latency, which prioritizes low-latency devices to enhance real-time model updates. S_C is a computational power that accounts for the device's processing capabilities. S_{CR} is a convergence rate, which rewards devices contributing to rapid model convergence. S_A is an availability that reflects device reliability based on historical uptime and participation.

The required metrics are obtained from device-local training statistics, such as validation loss trends and runtime system indicators, which are inherently available during model updates. As a result, the scheduling mechanism leverages existing information without introducing additional measurement or communication burden, preserving the lightweight nature of the framework.

The weights $\theta_1, \cdots, \theta_5$ can dynamically be adjusted based on system priorities and satisfy:

$$\sum_{j=1}^{5} \Theta_j = 1. \tag{26}$$

Probability and Frequency of Selection

Once scores are computed, the top-k highest-scoring devices are selected for full participation, while non-top devices ($i \notin K$) are probabilistically included. The probability of inclusion for top-k devices is given by:

$$P_i = \frac{S_i}{\sum_{j \in k} S_j}, \tag{27}$$

ensuring that higher-scoring devices in the top-k contribute more frequently. For non-top devices, the probability of participation is:

$$P_i = \frac{S_i}{\sum_{j \notin k} S_j}. \tag{28}$$

This ensures that lower-ranked devices are not completely excluded but rather contribute at a reduced rate based on their computed score.

Beyond selection, participation frequency is also modulated dynamically. High-scoring devices participate frequently but not at a constant rate. Their frequency is adjusted based on energy constraints and their marginal contribution to global model convergence. If a top-performing device exhibits low remaining energy, its participation frequency is automatically reduced:

$$f_{base} = f_{init} \times \left(1 - \frac{t}{T}\right) \times \left(\frac{E_i}{E_{max}}\right) \tag{29}$$

$$f_i = f_{base} \times S_E, \tag{30}$$

where f_{base} is the base participation frequency. Similarly, if a device's updates show diminishing improvement to the global model, its participation rate is decreased to reduce redundant updates. This reduction follows:

$$f_i \leftarrow f_i \times \left(1 - \frac{\Delta_{threshold}}{\Delta L_i}\right), \tag{31}$$

where ΔL_i represents the local model's convergence progress, and $\Delta_{threshold}$ is a predefined threshold indicating when training updates become less significant.

Convergence-Based FL Updates

As the FL-based NIDS model progresses through multiple training rounds, the impact of individual device updates on global model performance diminishes over time. In early training stages, frequent participation from high-performing devices accelerates convergence by quickly reducing the loss function. However, as the model approaches a stable state, continuing to collect updates from all devices at the same frequency becomes inefficient. This results in unnecessary communication overhead, increased energy consumption, and redundant updates that offer little improvement to model accuracy. Furthermore, in late-stage convergence, allowing all devices to participate equally leads to diminishing returns, as many updates contribute minimal gradient adjustments.

To address this inefficiency, we introduce a convergence-based update modulation mechanism that dynamically reduces the participation frequency of devices as the global model nears convergence. By adjusting device participation based on their local model convergence trends and the global loss reduction rate, this mechanism ensures that only the most impactful updates are prioritized, while redundant updates are minimized.

The participation frequency for each device is adjusted based on its local convergence rate S_{CR} and the global convergence trend:

$$f_i^{convergence} = f_i \times \left(1 - \frac{\Delta_{threshold}}{\Delta L_{global}} \times S_{CR}\right). \tag{32}$$

As the global model stabilizes, dynamic participation reduces update frequency across all devices, conserving resources, preventing redundant updates, and minimizing network congestion.

The final participation probability for each device is computed as:

$$P_i^{final} = P_i \times f_i^{convergence}, \tag{33}$$

ensuring that participation dynamically adjusts to both device conditions and model convergence trends. The **algorithm 1** illustrates dynamic participation of devices and convergence-driven updates in FL.

Now we discuss the dynamic participation and convergence-based FL update with an example.

Algorithm 1 Dyanmic Scheduling and Convergence-Based FL Updates

Require: N (devices), k (selection size), $\Delta_{\text{threshold}}$, f_{base}, $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$
Ensure: Participating devices in the FL round

▷ Calculate global convergence measure
$\Delta_{\text{global}} \leftarrow |L_{\text{prev}} - L_{\text{current}}|$
▷ Compute and rank scores for all devices
for $i \in N$ **do**
    $S_i \leftarrow \sum_{j=1}^{5} \theta_j S_j$
end for
Sort devices based on S_i; select top-k devices set $\mathcal{K}$
for $i \in N$ **do**
▷ Assign initial participation probabilities
    if $i \in \mathcal{K}$ **then**
        $P_i \leftarrow \frac{S_i}{\sum_{j \in \mathcal{K}} S_j}$
    else
        $P_i \leftarrow \frac{S_i}{\sum_{j \notin \mathcal{K}} S_j}$
    end if
▷ Adjust frequency based on device energy
    $f_i \leftarrow f_{\text{base}} \times S_E$
▷ Update frequency considering convergence rate
    $f_i^{\text{conv}} \leftarrow f_i \times \left(1 - \frac{\Delta_{\text{threshold}}}{\Delta_{\text{global}}} S_{CR}\right)$
▷ Calculate final probability of participation
    $P_i^{\text{final}} \leftarrow P_i \times f_i^{\text{conv}}$
▷ Decide participation based on final probability
    if $P_i^{\text{final}} > \text{Rand}(0, 1)$ **then**
        Mark device i as participating
    else
        Mark device i as not participating
    end if
end for
return participating devices

Consider an FL-based NIDS deployed across an IoT network with five devices (D1–D5), each assigned a participation score S_i based on available energy, latency, computational power, convergence rate, and availability. In an early training round, the top-3 devices (D1, D2, D3) are selected for full participation, while D4 and D5 are probabilistically included based on their normalized scores. Initially, each device's participation frequency is modulated by its energy level, ensuring that energy-constrained devices contribute less frequently. As training progresses, the global model loss reduction is monitored, and once it drops below a predefined threshold $\Delta_{threshold}$, the participation frequency of all devices is dynamically reduced using a convergence-based update $f_i^{convergence}$. For instance, if D1 has a high convergence impact, it maintains frequent updates, whereas D3, with minimal impact, sees its updates reduced. The final participation probability is adjusted as P_i^{final} , ensuring that only the most effective updates are transmitted, thereby minimizing redundant communication and energy consumption while preserving model accuracy.

4.3.4 Few-Shot Learning Model

As we optimize FL learning updates through dynamic scheduling and convergence-based updates, the framework improves client participation efficiency, accelerates model convergence, and reduces communication overhead while maintaining decentralized training and data privacy. However, to further optimize anomaly-based NIDS, a more robust DL model is required to address the challenges of resource-constrained IoT devices and limited labeled data. Traditional supervised learning models struggle with the evolving nature of network threats due to their dependency on extensive labeled datasets, which are often unavailable in real-world scenarios. Additionally, IoT devices, due to their limited computational resources, cannot efficiently process high-dimensional network traffic data using standard DL models.

To overcome these limitations, we integrate FSL with FL, leveraging prototypical networks with LSTM to enhance classification performance with minimal labeled data. This approach enables efficient generalization from a small number of labeled attack patterns, ensuring adaptability to emerging threats.

Given an input sequence $X = x_1, x_2, \cdots, x_T$, where T represents the sequence length, the LSTM

processes each time step iteratively:

$$h_t = \sigma\left(W_h h_{t-1} + W_x x_t + b\right). \tag{34}$$

where h_t is the hidden state at time step t, W_h and W_x are weight matrices, b is the bias term, and σ represents an activation function, often a tanh or ReLU function. The final feature representation $f_\theta(x)$ is obtained from the last hidden state:

$$f_\theta(x) = h_T. \tag{35}$$

This embedding $f_\theta(x)$ serves as the input for prototype computation in the few-shot classification stage.

Prototype Computation and Query Classification

In an FSL setup, the dataset is divided into a support set and a query set. The support set consists of a few labeled examples per class, providing reference points for classification, while the query set contains unseen samples that need to be classified based on their similarity to the support set.
To compute class prototypes in an FSL setup, we calculate the mean embedding of support set samples for each class:

$$p_c = \frac{1}{|S_c|} \sum_{x \epsilon S_c} f_\theta(x), \tag{36}$$

where p_c is the prototype representation of class, S_c is the support set for class c and $f_\theta(x)$ is the embedding of a sample x obtained from the LSTM encoder. Query samples are classified based on their Euclidean distance from

$$d\left(q, p_c\right) = \left\| f_\theta\left(q\right) - p_c \right\|^2, \tag{37}$$

where q represents the query sample. The model assigns q to the nearest prototype, allowing classification even with minimal labeled samples.

4.3.5 Model Compression

The few-shot learning model improves adaptability under limited labeled data, but its direct deployment on IoT devices can still be hindered by computational and memory constraints. To overcome these limitations, Light-FFSL incorporates model compression techniques that preserve detection accuracy while significantly reducing resource consumption. In particular, structured pruning and post-training quantization are employed to remove redundant parameters, minimize model size, and accelerate inference.

Model Pruning

To reduce the computational complexity of the LSTM-based prototypical network prior to deployment, Light-FFSL applies a two-stage pruning strategy. First, we apply pre-pruning techniques that optimize the architecture before the training process begins. Instead of training a full-scale model and pruning afterward, we construct a smaller model using structured sparsity principles Y. He and Xiao (2023). The process involves reducing the number of LSTM units, removing redundant neurons, and applying low-rank approximations to weight matrices:

$$W \approx U_k S_k V_k^T, \tag{38}$$

where U_k and V_k are reduced-dimensional matrices, and S_k contains the top k singular values. This transformation significantly reduces the model's complexity before training, allowing efficient deployment on resource-constrained IoT devices. Following initial training, structured pruning is applied to further refine the model by removing weights that contribute the least to performance. Given an LSTM weight matrix W, we remove entire groups of parameters (e.g., neurons or gates) whose collective importance falls below a threshold τ :

$$\mathcal{G}'_k = \begin{cases} \mathcal{G}_k, & \text{if } \|\mathcal{G}_k\|_2 \geq \tau, \\ 0, & \text{otherwise,} \end{cases} \tag{39}$$

where $\mathcal{G}'_k$ represents the pruned group of LSTM parameters.The pruning ratio r is computed as:

$$r = \frac{\left\|W'\right\|_0}{\|W\|_0}, \tag{40}$$

where $\|W\|_0$ and $\left\|W'\right\|_0$ denote the number of nonzero parameters before and after pruning, respectively. Once pruning is completed, fine-tuning is performed to recover any accuracy loss. By combining pre-pruning and structured pruning, we achieve a highly efficient model that retains predictive performance while requiring fewer computational resources.

Model Quantization

To further reduce the memory footprints, we apply Full Integer Quantization H. Wu, Judd, Zhang, Isaev, and Micikevicius (2020), which is particularly suitable for resource-constrained IoT devices. This approach converts both weights and activations to integer representations, reducing the memory footprint and accelerating inference speed. Instead of performing computations in floating-point precision, the model operates entirely in integer arithmetic, making it significantly more efficient for edge devices with limited computational capacity.

Formally, full integer quantization is performed by mapping weights and activations to integer representations using scaling factors:

$$A_{quant} = round(\frac{A}{s_A}), \quad W_{quant} = round(\frac{W}{s_W}), \tag{41}$$

where A_{quant} and W_{quant} are the quantized activation and weight values, respectively. s_A and s_W are learned scaling factors for activations and weights, respectively. This ensures that numerical precision is preserved while reducing the model's overall complexity.

Algorithm 2 illustrates the optimized FSL with pruning and quantization. The algorithm begins by taking labeled training data $D = \{(x_i, y_i)\}$, a support set S, and a query set Q, aiming to produce an optimized Prototypical Network model M. It starts with pre-pruning, where the LSTM units are reduced using low-rank decomposition via SVD, approximating the weight matrix as $W \leftarrow U_k S_k V_k^T$ (Line 2) to minimize unnecessary computations before training. The model then

proceeds to training, where for each batch in the dataset, it trains on support set samples. For each class c in S, a class prototype is computed as $p_c \leftarrow \frac{1}{|S_c|} \sum_{x \in S_c} f_\theta(x)$ (Line 3-7), ensuring that each class is represented efficiently. After training, the algorithm applies structured pruning, iterating through all weights $W_{i,j}$ and setting those below a predefined threshold τ to zero (Line 9-13), reducing complexity while preserving accuracy. Since pruning removes parameters, the model undergoes fine-tuning to recover any performance loss. Next, post-quantization is applied, where weights and activations are converted into integer representations using $A_{quant} = round(A/s_A)$ and $W_{quant} = round(W/s_W)$ (Line 15-18), significantly improving memory efficiency and inference speed. During inference, for each query sample q, the Euclidean distance to each class prototype is computed as $d(q, p_c) = ||f_\theta(q) - p_c||^2$ (Line 22), with the query being assigned to the class with the minimum distance, computed as $y_q \leftarrow \arg\min_c d(q, p_c)$ (Line 25), ensuring few-shot classification.

4.3.6 Complexity Analysis

The computational complexity of the proposed algorithm consists of multiple stages. First, the LSTM network used for feature extraction has a complexity of $\mathcal{O}(T \cdot n^2)$, where T is the sequence length of the input data and n denotes the number of LSTM units. After applying structured pruning, this complexity is reduced to $\mathcal{O}\big(T \cdot (r \cdot n^2)\big)$, where $0 < r < 1$ is the pruning ratio, thereby decreasing the computational load by removing redundant parameters. The computation of class prototypes for each class incurs a cost of $\mathcal{O}(|S_c| \cdot d)$, where $|S_c|$ is the number of support samples in class c and d is the embedding dimension of the feature space. During inference, classifying each query sample requires computing the Euclidean distance to all C class prototypes, resulting in a complexity of $\mathcal{O}(C \cdot d)$. The post-quantization step converts weights and activations to integer values, improving runtime efficiency in practice, although it does not affect the asymptotic complexity. Thus, the overall computational complexity of the proposed algorithm is:

$$\mathcal{O}\big(T \cdot (r \cdot n^2) + |S_c| \cdot d + C \cdot d\big).$$

Algorithm 2 Optimized Few-Shot Learning with Pruning and Quantization

Require: Labeled training data $D = \{(x_i, y_i)\}$, support set S, query set Q
Ensure: Optimized Prototypical Network model M

▷ Pre-Pruning: Reduce model complexity via low-rank decomposition
$W \leftarrow U_k S_k V_k^T$
for each batch in training data **do**
  for each class c in S **do**
    $p_c \leftarrow \frac{1}{|S_c|} \sum_{x \in S_c} f_\theta(x)$
  end for
end for
▷ Structured Pruning: Remove neurons below threshold τ
for each parameter group $\mathcal{G}_k$ in model **do**
  if $\|\mathcal{G}_k\|_2 < \tau$ **then**
    Set $\mathcal{G}_k = 0$
  end if
end for
▷ Post-Quantization: Convert weights and activations to integer representations
for each weight W and activation A **do**
  $A_{\text{quant}} \leftarrow \text{round}(A/s_A)$
  $W_{\text{quant}} \leftarrow \text{round}(W/s_W)$
end for
▷ Inference and Classification: Predict class for each query sample
for each query sample $q \in Q$ **do**
  for each class prototype p_c **do**
    $d(q, p_c) = ||f_\theta(q) - p_c||^2$
  end for
  ▷ Assign class with closest prototype
  $y_q \leftarrow \arg\min_c d(q, p_c)$
end for

4.4 Performance Evaluation

This section presents the performance evaluation of the proposed Light-FFSL framework in a simulated IoT environment, where multiple resource-constrained devices are simulated in Python to reflect realistic IoT hardware characteristics. The evaluation uses widely adopted IoT security datasets: TON_IoT Booij et al. (2021), and Bot_IoT Koroniotis, Moustafa, Sitnikova, and Turnbull (2019), focusing on convergence analysis, communication efficiency, computational overhead, and model performance.

4.4.1 Experimental Setup

IoT Devices

To simulate a real-world IoT network, 50 IoT devices were simulated using Python. Each device operated as an independent process with constrained computational resources, mirroring hardware limitations such as low CPU frequency, limited memory (512 MB to 1 GB), and restricted network bandwidth (1–5 Mbps). Lightweight TCP sockets facilitated communication between devices and the central server, with network conditions including random latency (10–100 milliseconds) and packet loss (0–5%) to model realistic IoT environments.

Model Architecture

Each device trained a lightweight LSTM-based prototypical network, chosen for its effectiveness in FSL scenarios. The model consisted of a single LSTM layer with 64 hidden units followed by an embedding layer outputting 32-dimensional feature vectors. Classification was based on the Euclidean distance between embeddings and class prototypes, a standard method in FSL literature. To make the model deployable on resource-constrained devices, model compression techniques were applied. During local training, 30% of the model's weights were pruned using unstructured pruning methods. After training, 8-bit post-training quantization was performed using TensorFlow Lite, significantly reducing the model size and inference time.

Federated Learning Configuration

A modified federated averaging approach was employed to coordinate training across devices. In each communication round, device participation was determined dynamically based on simulated resource availability. Devices were evaluated across five metrics. Thresholds were established for each factor to determine eligibility for full participation: an energy level exceeding 20%, CPU availability greater than 40%, communication latency below 100 milliseconds, and device uptime above 80%. Devices satisfying all thresholds were selected for full participation, while those failing to meet one or more criteria were probabilistically included based on their normalized scores.
To optimize communication efficiency, a convergence-based update mechanism was integrated at both the local and global levels. At the local level, devices transmitted model updates only when their validation loss improved by at least 0.5% compared to the previous round, or when the model exhibited stability, defined as a relative change in validation loss of less than 0.1% over five consecutive local training epochs. At the global level, the server monitored the convergence of the aggregated global model. If the global validation loss reduction was less than 0.1% over five consecutive aggregation rounds, the system dynamically reduced the device participation rate by decreasing the number of devices selected for each subsequent round.

Data Preprocessing

The TON_IoT and Bot_IoT datasets were used for evaluation. Preprocessing included normalization of numerical features, one-hot encoding of categorical features, and removal of irrelevant attributes.

4.4.2 Datasets

The evaluation of the proposed Light-FFSL framework was conducted using two publicly available IoT security datasets: TON_IoT and Bot_IoT.

Table 4.1: Experimental Settings

Parameter	**Setting**
Device Emulation	Python 3.10 processes simulating resource-constrained IoT devices
Number of Devices	50 virtual IoT nodes
Device Configuration	CPU frequency: 1.5 GHz (simulated), RAM: 512 MB–1 GB
Network Conditions	Latency: 10–100 ms, Packet Loss: 0%–5%
Communication Protocol	Lightweight TCP sockets
Base Model	LSTM-based Prototypical Network
LSTM Hidden Units	64
Embedding Dimension	32
Model Compression	30% unstructured pruning, 8-bit quantization (TensorFlow Lite)
Optimizer	Adam
Learning Rate	0.001
Batch Size	32 episodes per communication round
Federated Learning Strategy	Modified FedAvg with dynamic scheduling and convergence-based updates
Device Selection Thresholds	Energy $> 20\%$, CPU availability $> 40\%$, Latency < 100 ms, Uptime $> 80\%$
Local Update Condition	Upload if loss improves $\geq 0.5\%$ or loss change $< 0.1\%$ over 5 epochs
Global Stability Condition	Reduce device participation if global loss change $< 0.1\%$ over 5 rounds
Dynamic Participation	Top 50% devices initially; top 30% after global model stabilizes
Maximum Rounds	500
Software Stack	PyTorch 2.0, TensorFlow Lite, Scikit-learn
Datasets	TON_IoT, Bot_IoT

Frequent/Evolving Split Details

We treat the top-m classes ranked by training frequency as $\mathcal{C}_{\text{freq}}$, while the remaining classes $\mathcal{C}_{\text{new}}$ are excluded from training and only appear during evaluation. This split reflects the generalized few-shot setting, where the model must adapt to previously unseen attack classes from only a few labeled examples. For comparison with conventional supervised baselines, we also consider a closed-set evaluation in which all classes are included during training and testing. In this setting, supervised models are trained with the full labeled dataset, whereas our few-shot model remains constrained to using only a few labeled samples per class. This dual evaluation protocol ensures fairness by assessing supervised approaches in their standard regime, while also highlighting the label efficiency and adaptability of few-shot learning methods.

TON_IoT

The TON_IoT dataset, developed by the Cyber Range Lab at UNSW Canberra Booij et al. (2021), offers a comprehensive representation of modern IoT security threats. It includes telemetry data from IoT sensors, operating system logs, network traffic captures, and system processes. For this work, only the network traffic subset was utilized. The dataset contains various types of attacks such as reconnaissance, backdoor access, ransomware, denial of service (DoS), distributed denial of service (DDoS), injection attacks, and data exfiltration. The dataset consists of 44 features and presents a moderate level of class imbalance, where attacks slightly outnumber normal traffic. Preprocessing involved feature normalization, categorical encoding, and removal of irrelevant attributes.

BoT_IoT

The BoT-IoT dataset, also produced by UNSW Canberra Koroniotis et al. (2019), captures extensive network traffic generated in a controlled IoT environment. It includes both benign traffic and a wide range of attack scenarios. The attacks represented in the dataset include denial of service (DoS), distributed denial of service (DDoS), probing activities such as port scanning, and information theft attacks, including data leakage and password theft. Each record is characterized by 45 features. Preprocessing involved feature normalization, categorical encoding, and removal of irrelevant attributes.

4.4.3 Numerical Results

In this section, we evaluate the proposed Light-FFSL framework in terms of device participation and convergence behavior. We also evaluate its performance in comparison with existing NIDS techniques.

Device Participation and Convergence Analysis

This section analyzes the impact of dynamic participation and convergence-based updates on IoT devices and federated training performance.

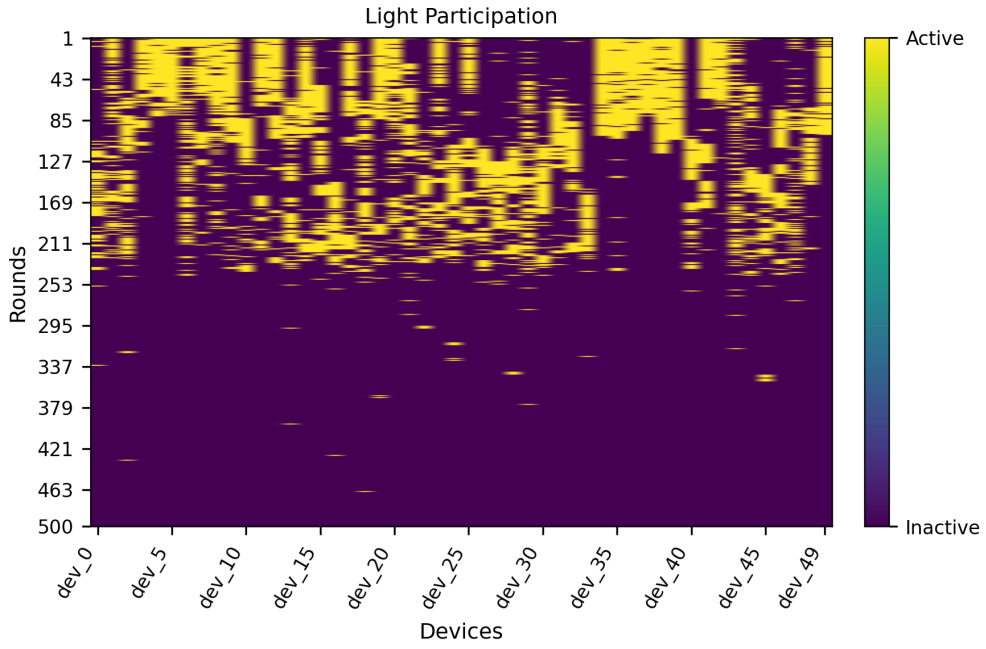


Figure 4.3: Heatmap of participation frequency for 50 IoT devices over 500 communication rounds.

To investigate device engagement during training, Figures 4.3 and 4.4 illustrates heatmaps of participation patterns under Standard FL and Light-FFSL across 500 communication rounds. Each row corresponds to a device (indexed 0–49) and each column to a training round, with shading indicating participation status. Standard FL shows relatively uniform participation in the early rounds, followed by a sharp decline after approximately 120 rounds, leaving most devices inactive. By contrast, Light-FFSL maintains adaptive and heterogeneous participation, where subsets of devices are selectively engaged based on their normalized suitability scores derived from system metrics. This scheduling strategy sustains intermittent activity beyond 250 rounds, distributing training contributions more evenly and mitigating premature device drop-off. These results demonstrate that Light-FFSL not only balances energy consumption and computational cost but also enhances long-term device engagement compared to Standard FL.

Figure 4.5 compares device sustainability between Standard FL and the proposed Light-FFSL framework over 500 communication rounds. In Standard FL, the number of active devices declines sharply, with nearly all devices exhausted before round 150 due to uniform participation that disproportionately burdens resource-constrained nodes. In contrast, Light-FFSL sustains device availability for a much longer duration, with a gradual decline that maintains a significant

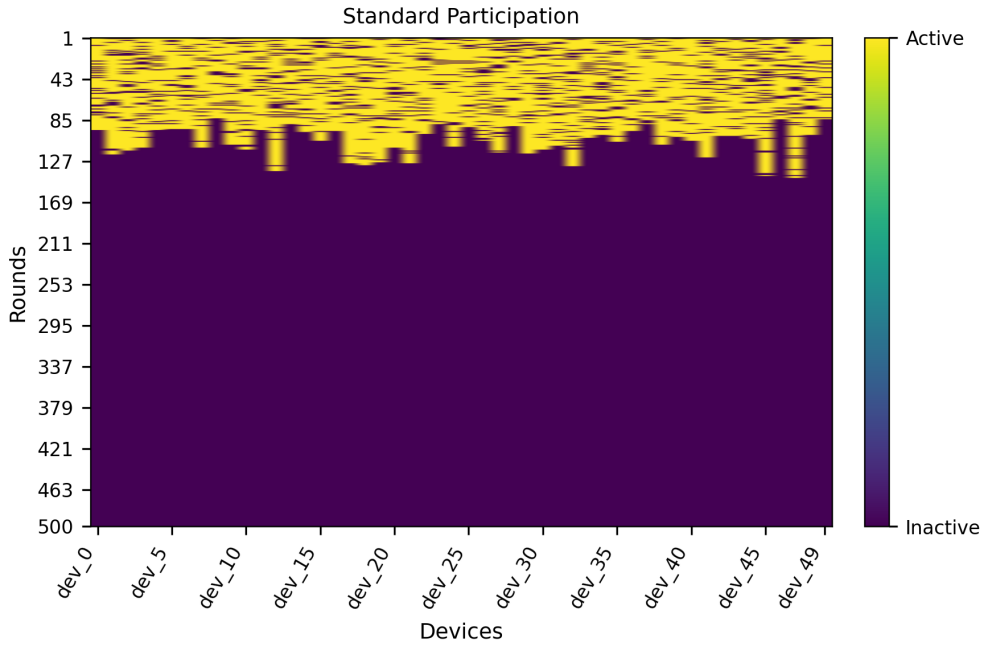


Figure 4.4: Heatmap of participation frequency for 50 IoT devices over 500 communication rounds.

proportion of devices active even beyond 400 rounds. This improvement stems from its adaptive scheduling mechanism, which leverages normalized suitability scores to balance device workload and energy usage. By preventing premature depletion of weaker devices, Light-FFSL distributes training responsibility more evenly, thereby extending network longevity and ensuring more consistent participation across rounds.

Figure 4.6 compares the global loss trajectories of Standard FL and the proposed Light-FFSL framework over 500 communication rounds. Both approaches exhibit a rapid decline in loss during the initial phase, indicating efficient convergence in the early rounds. Standard FL achieves slightly faster reduction within the first 100 rounds; however, Light-FFSL demonstrates superior stability and consistently lower loss beyond approximately 150 rounds. This improvement is attributed to the adaptive scheduling mechanism of Light-FFSL, which selectively engages devices based on normalized suitability scores, thereby balancing energy consumption and computational contribution without overburdening resource-constrained nodes. As a result, Light-FFSL sustains convergence performance in the later stages of training, outperforming Standard FL in terms of long-term optimization stability and efficiency.

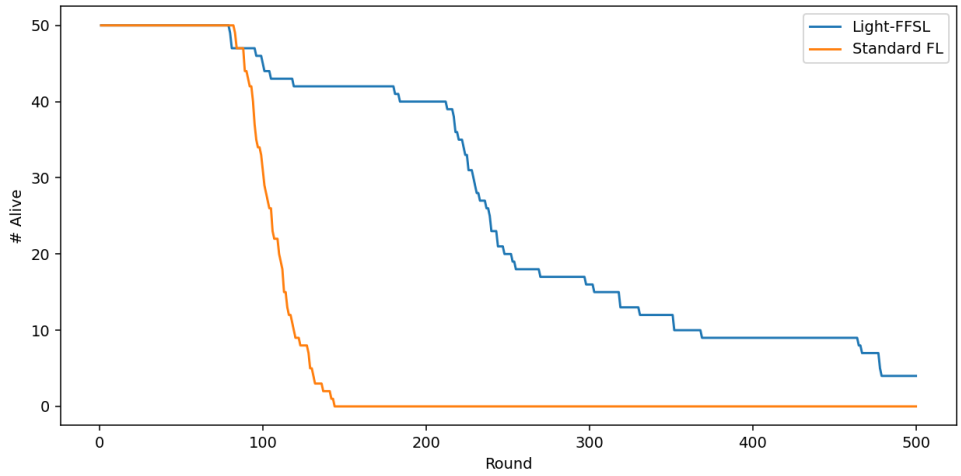


Figure 4.5: Number of dead devices across federated rounds.

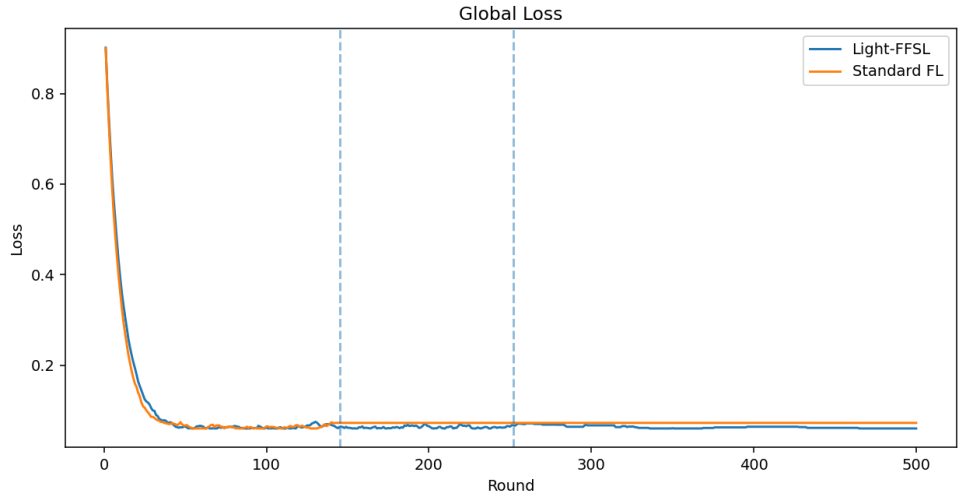


Figure 4.6: Global loss over federated rounds.

Model Efficiency

To evaluate the efficiency of the proposed Light-FFSL framework for deployment in resource-constrained IoT environments, we first examine the impact of pruning through the analysis of weight distribution and non-zero parameter counts across model layers. We then assess performance in terms of model size and inference time. Specifically, Light-FFSL is compared with the baseline Light-FFSL model (uncompressed) and the FS-IDS. In addition, two optimized configurations, a pruned variant and a pruned and quantized variant, are evaluated to determine their effectiveness in reducing computational load and memory requirements.

Figure 4.7 illustrates the number of non-zero parameters across key layers before and after pruning. The reduction is most prominent in the larger layers (e.g., 30×256 and 64×256), indicating effective removal of redundant weights, while smaller layers (e.g., 64×32) remain largely unaffected to preserve critical model capacity. Figure 4.8 presents the corresponding weight distributions for the original and pruned models. The pruned model exhibits a sharp increase in the frequency of near-zero weights, reflecting the induced sparsity that contributes to lower computational and memory demands. These results confirm that pruning successfully reduces parameter count without severely impacting the core structure of the network, supporting its suitability for deployment in resource-constrained IoT environments

As illustrated in Figure 4.9a, the pruned-and-quantized Light-FFSL configuration achieves the most compact architecture, requiring only 0.043 MB of storage. This corresponds to a 62.7% size reduction compared to the standard FFSL model and a 66.9% reduction relative to the FS-IDS system. The pruned only variant of Light-FFSL attains a 29.6% reduction in model size relative to the standard FFSL model and a 37.7% reduction compared to FS-IDS. Figure 4.9b compares inference times across configurations. The pruned-and-quantized Light-FFSL achieves the lowest latency, completing predictions in just 0.65 seconds, a 79.7% reduction compared to the standard FFSL model and an improvement of over 80% relative to FS-IDS. The pruned-only variant also reduces latency to 2.66 seconds, reflecting moderate efficiency gains from structured sparsity. These improvements result from the combined effects of pruning and quantization. Pruning eliminates low-contribution parameters, introducing sparsity and reducing model complexity. Quantization further enhances

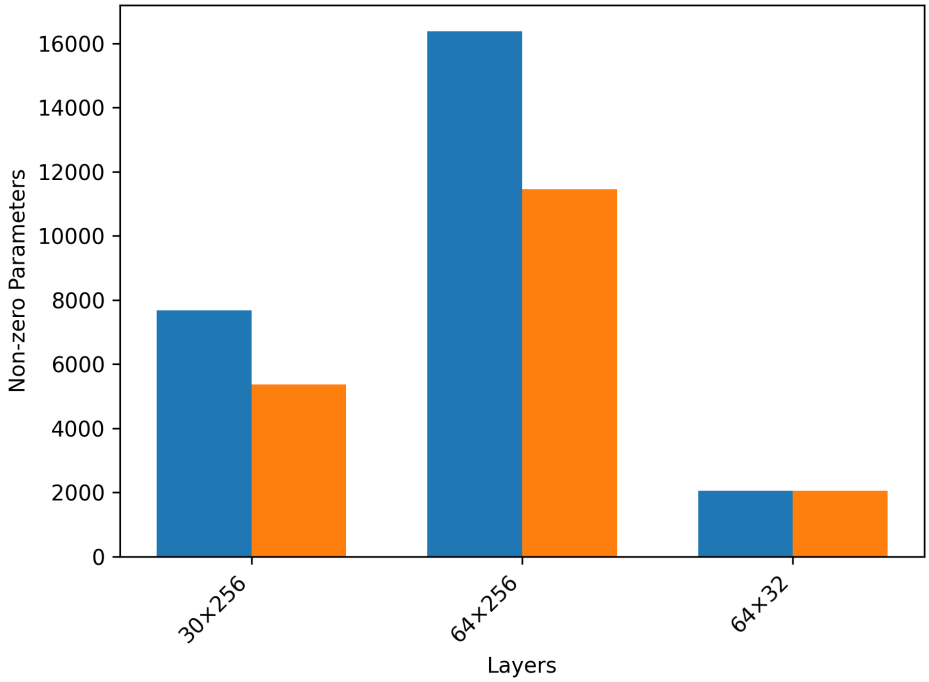


Figure 4.7: Layer-wise Non-zero Count: Baseline Light-FFSL vs. Pruned Light-FFSL

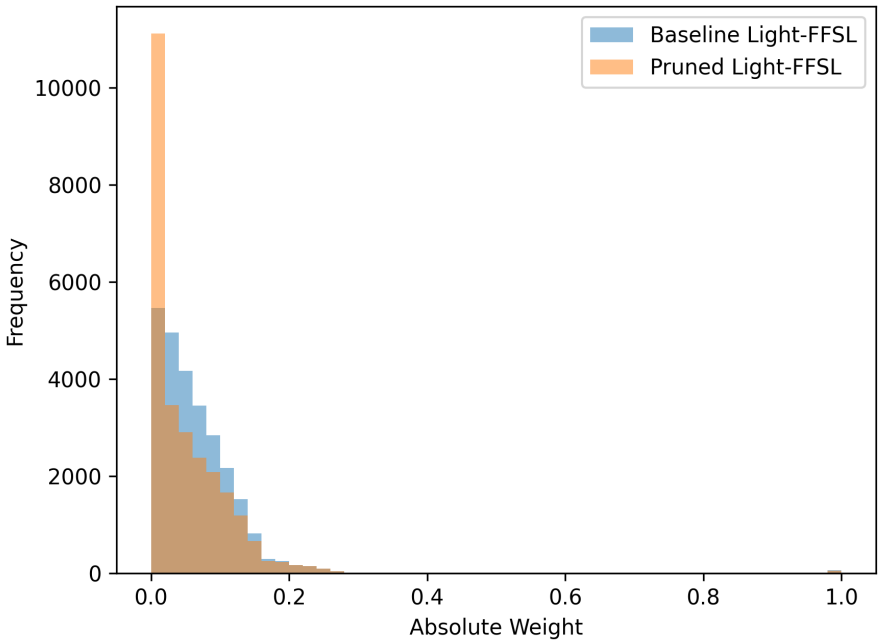
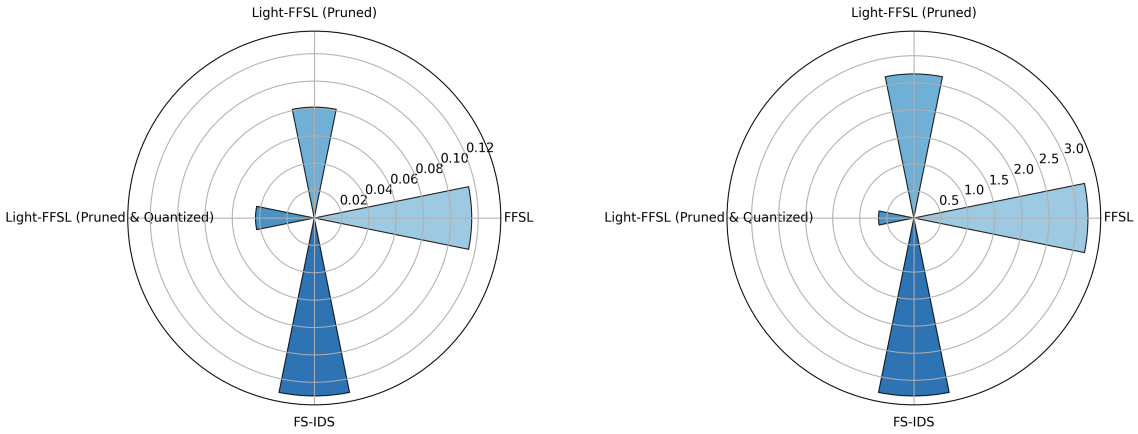


Figure 4.8: Weight Distribution: Baseline Light-FFSL vs. Pruned Light-FFSL

efficiency by converting floating-point weights and activations into low-precision integer formats, thereby lowering memory usage and computational overhead. Together, these optimizations enable substantial compression and faster inference, reinforcing the suitability of the proposed framework for real-time deployment in resource-constrained IoT environments.



(a) Model Size (MB)

(b) Inference Time (seconds)

Figure 4.9: Comparison of model size and inference time for Light-FFSL variants, FFSL, and FS-IDS.

Effect of Sequence Length on Model Performance

Table 4.2 reports the weighted accuracy, precision, recall, and F1-score of the model across different sequence lengths T. The results show that model performance remains relatively consistent across $T = 15$ to $T = 30$, with only minor fluctuations in all metrics. While $T = 25$ yields the highest overall scores, the differences compared between shorter or longer sequences is marginal, suggesting that the choice of sequence length has a limited impact on detection capability. However, the computational complexity, measured in Multiply–Accumulate Operations (MACs), increases linearly with T, even though performance remains almost flat. This highlights the importance of considering both accuracy and complexity when selecting T, as longer sequences introduce higher computational cost without providing substantial performance improvements in resource-constrained IoT environments.

Table 4.2: Model Performance and Computational Complexity Across Varying Sequence Lengths (T)

T	**Accuracy**	**Precision**	**Recall**	**F1-score**	**MACs**
10	0.8141	0.8312	0.8141	0.8180	753,664
15	0.8681	0.8862	0.8681	0.8704	1,122,304
20	0.8647	0.8744	0.8647	0.8641	1,490,944
25	0.8860	0.8948	0.8860	0.8874	1,859,584
30	0.8746	0.8869	0.8746	0.8750	2,228,224

Performance Analysis on Evolving Attacks

To evaluate the performance of the proposed Light-FFSL framework on evolving attacks, we conduct a comparative evaluation against two methods: the standard FFSL model and the FS-IDS. As shown in Figure 4.10, the experiments are conducted over multiple (k, q) configurations, specifically k = 3, 5, 10 and q = 5, 15, 30, using two benchmark datasets: ToN_IoT and Bot_IoT. Model performance is evaluated using four standard classification metrics: accuracy, precision, recall, and F1-score, providing a comprehensive assessment of detection capability under varied FSL conditions. In the ToN IoT dataset, the proposed Light-FFSL framework consistently achieves performance comparable to the performance of the standard FFSL model while outperforming the FS-IDS across all few-shot configurations. Under the highest support-query setting $(k = 10, q = 30)$, Light-FFSL achieves an accuracy of 0.922, only 0.95% below FFSL and approximately 2.2% higher than FS-IDS. A similar pattern is observed in F1-score, where Light-FFSL attains 0.920, trailing FFSL by just 0.9% and exceeding FS-IDS by 4.6%, demonstrating its ability to maintain balanced precision and recall despite compression. This performance persists across intermediate settings. For $(k = 5, q = 15)$, Light-FFSL reaches an accuracy of 0.841, marginally lower than FFSL but 2.0% higher than FS-IDS. The corresponding F1-score is 0.834, closely aligned with FFSL and notably surpassing FS-IDS. Even under the most constrained scenario $(k = 3, q = 5)$, Light-FFSL maintains competitive accuracy (0.773), slightly below FFSL yet outperforming FS-IDS by 3.0%. The F1-score remains robust at 0.7403, compared to 0.753 for FFSL, underscoring the model's stability in low-data environments.

In the Bot_IoT dataset, the proposed Light-FFSL model demonstrates performance that consistently tracks within approximately 2% of the standard FFSL, while still outperforming the FS-IDS. Under the highest support-query configuration ($k = 10, q = 30$), Light-FFSL achieves an accuracy of 0.915, only 2.0% lower than FFSL, yet 1.0% higher than FS-IDS. The corresponding F1-score of 0.906 similarly trails FFSL by approximately 2.0%, indicating that compression has minimal effect on classification precision and recall. At the intermediate setting ($k = 5, q = 15$), Light-FFSL attains an accuracy of 0.8754, nearly identical to FFSL within 0.05% and 1.8% higher than FS-IDS. Its F1-score of 0.8737 also remains close to that of FFSL, while surpassing FS-IDS by 1.0%. Even under the most constrained configuration ($k = 3, q = 5$), Light-FFSL sustains competitive accuracy at 0.840, trailing FFSL by just 0.5% and outperforming FS-IDS by 1.5%. The corresponding F1-score of 0.809 reflects minor degradation relative to FFSL and a 0.7% improvement over FS-IDS. The modest reduction in performance observed in Light-FFSL is primarily attributed to the quantization process, which compresses the model by converting floating-point weights and activations into 8-bit integer representations. While this technique is critical for substantially reducing both memory footprint and inference latency, it can introduce a slight performance compromise. Nevertheless, the integration of structured pruning and quantization in Light-FFSL limits the performance decline to less than 2% compared to the uncompressed FFSL model, while consistently delivering a 3–5% improvement over the FS-IDS. These findings indicate that the efficiency–accuracy trade-off introduced by compression is both minimal and effectively managed. Consequently, Light-FFSL emerges as a practical and scalable solution for real-time intrusion detection in resource-constrained IoT environments, where stringent limitations on memory, computation, and latency coexist with the need for reliable threat detection.

Comparison with Existing NIDS Approaches

To ensure a fair evaluation, all models are compared using only seen attack classes, thereby eliminating biases that could arise from differing treatment of unseen threats across methods. Table 4.3 summarizes the performance of Light-FFSL against several NIDS approaches. The results show that Light-FFSL achieves a competitive accuracy of 97.5% with balanced precision, recall, and F1-score, while maintaining an exceptionally lightweight design (0.043 MB) and low inference latency

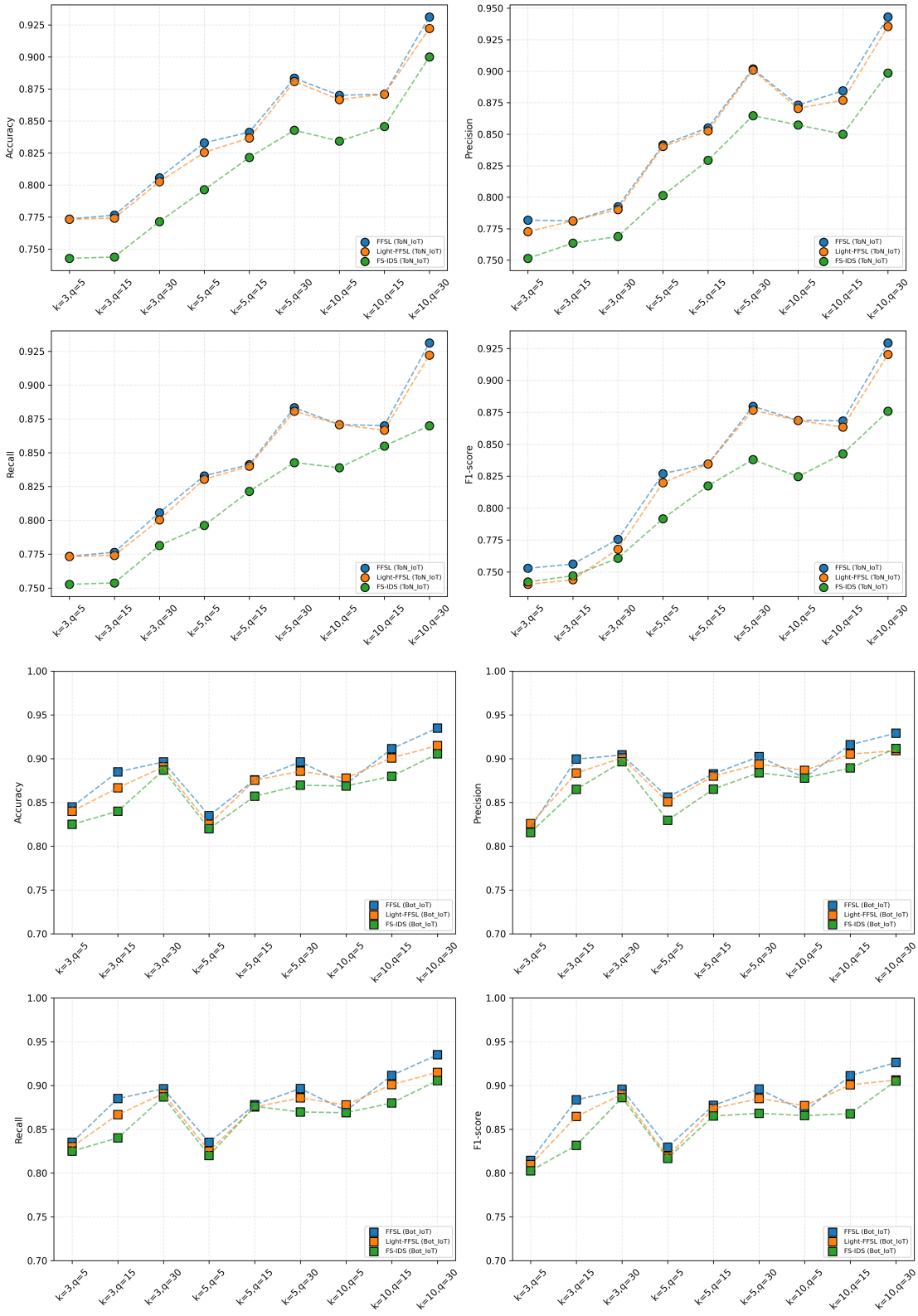


Figure 4.10: Comparison of Light-FFSL, FFSL, and FS-IDS on ToN_IoT and bot_IoT datasets.

(0.654 s). In contrast, methods such as GMCWAE and NIDS-CNNRF achieve strong detection performance but at the cost of significantly larger model sizes and higher inference times, making them less suitable for resource-constrained IoT devices. While A-NIDS exhibit a smaller memory footprint, they suffer from higher inference time and reduced performance compared to Light-FFSL. In contrast, ANN achieves lower inference time but at the cost of higher memory consumption and reduced performance. Overall, this consistent and fair evaluation highlights that Light-FFSL achieves the best balance between detection accuracy, computational efficiency, and deployment feasibility, demonstrating its suitability for securing resource-constrained IoT environments.

Table 4.3: Performance Comparison of Light-FFSL with Existing NIDS Approaches

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Model Size (MB)**	**Inference Time (s)**
GMCWAE Bian and Liu (2025)	0.9423	0.9444	0.9826	0.9635	1.202	2.299
A-NIDS Zha et al. (2025)	0.9414	0.9963	0.9261	0.9602	0.345	0.993
NIDS-CNNRF Yang et al. (2025)	0.9825	0.9974	0.9802	0.9883	193.622	4.702
ANN Farhat et al. (2025)	0.9056	0.9077	0.9056	0.9064	9.013	0.392
Light-FFSL (Proposed)	0.9753	0.9772	0.9752	0.9758	0.043	0.654

4.5 Conclusion

This chapter proposed resource-aware Light-FFSL, a lightweight federated few-shot learning framework designed to address the critical challenges of intrusion detection in resource-constrained IoT environments. By integrating dynamic scheduling, convergence-based updates, and a prune-and-quantize LSTM-based prototypical network, the proposed system effectively balances computational efficiency, data privacy, and learning with minimal labeled data. Extensive experimental evaluation across two benchmark datasets (ToN_IoT and Bot_IoT) demonstrated that Light-FFSL not only achieves competitive detection performance compared to full-capacity models but also significantly reduces model size and inference latency, making it highly suitable for low-power IoT deployment. Moreover, the framework maintains device longevity and ensures sustainable participation across FL rounds through adaptive device selection. These results show that Light-FFSL is a practical and scalable solution for securing IoT networks under stringent resource and privacy constraints.

Chapter 5

Conclusion and Future Works

This thesis has addressed the challenges of intrusion detection in resource-constrained and privacy-sensitive IoT environments by proposing a framework that integrates FL, FSL, and model compression techniques. The research was motivated by the limitations of traditional NIDS, which rely heavily on centralized data collection, large labeled datasets, and high-capacity computing resources assumptions that do not hold in practical IoT deployments. In response, the proposed solutions were designed to operate effectively under constraints of data privacy, label scarcity, and limited computational capabilities.

The thesis introduced a FFSL framework, which enables distributed IoT devices to collaboratively train a network intrusion detection model without sharing data. By incorporating a prototypical FSL architecture with LSTM-based feature encoders, the framework demonstrated improved generalization to evolving or previously unseen attack types, even with limited labeled data. Building upon this foundation, the Lightweight FFSL (Light-FFSL) framework was developed to specifically address the challenges posed by resource-constrained IoT environments. Light-FFSL introduced dynamic scheduling to optimize device participation based on resource availability, and implemented convergence-based update mechanisms to reduce unnecessary communication overhead. Additionally, structured pruning and post-training quantization were applied to produce a compact, deployable model suitable for real-time intrusion detection on low-power IoT devices.

The effectiveness of the proposed methods was validated through extensive experimentation using publicly available benchmark datasets (ToN_IoT and Bot_IoT) and a simulated FL environment

comprising 50 heterogeneous IoT nodes. The results demonstrate that FFSL effectively trains models using only a few labeled examples across distributed clients, outperforming FS-IDS in terms of detection performance. Furthermore, Light-FFSL maintains competitive detection accuracy while significantly reducing model size and inference latency. Compared to standard FFSL, FS-IDS and existing NIDS techniques, Light-FFSL achieves superior trade-offs between accuracy and efficiency, making it well-suited for deployment in resource-constrained IoT environments.

Despite the promising outcomes achieved in this thesis, several limitations remain. First, the simulation-based evaluation, although realistic, cannot fully capture the complexities of real-world deployments, such as adversarial behavior, large-scale network failures, or device dropouts. Future research could explore concept drift adaptation techniques to enhance responsiveness to evolving and previously unseen attack patterns, as well as self-supervised and continual learning methods to improve adaptability without requiring full retraining.

Additional directions include addressing the heterogeneity of IoT devices and their inherently non-IID data distributions through personalized or clustered federated learning approaches. Realistic adversarial evaluation under domain-specific constraints, adaptive architecture selection based on device capabilities, and multimodal feature fusion to leverage diverse network features all represent promising extensions. Moreover, richer evaluation metrics, such as communication overhead, energy consumption, detection latency, and false alarm rates, could further evaluate the scalability and practicality of Light-FFSL in large-scale IoT deployments.

Appendix A

Appendix List of Publications

A.1 Lists of Publication

The following publication(s) have resulted from the research presented in this thesis:

- Ahsan Saleem, and Walaa Hamouda. ”Federated Few-Shot Learning for Robust and Privacy-Driven Network Intrusion Detection in IoT.” In ICC 2025-IEEE International Conference on Communications. IEEE, 2025.
- Ahsan Saleem, and Walaa Hamouda. ”Lightweight Federated Few-Shot Learning-based Network Intrusion Detection for resource-constrained IoT devices.” IEEE Internet of Things Journal (2025) (Minor Revisions) .

References

Abdel-Basset, M., Hawash, H., Chakrabortty, R. K., & Ryan, M. J. (2021). Semi-supervised spatiotemporal deep learning for intrusions detection in iot networks. *IEEE Internet of Things Journal*, *8*(15), 12251–12265.

Abdulganiyu, O. H., Tchakoucht, T. A., & Saheed, Y. K. (2024). Towards an efficient model for network intrusion detection system (ids): systematic literature review. *Wireless Networks*, *30*(1), 453–482.

Abdulkareem, S. A., Foh, C. H., Shojafar, M., Carrez, F., & Moessner, K. (2024). Network intrusion detection: An iot and non iot-related survey. *IEEE Access*.

Aljuhani, A., Alamri, A., Kumar, P., & Jolfaei, A. (2023). An intelligent and explainable saas-based intrusion detection system for resource-constrained iomt. *IEEE Internet of Things Journal*.

Al-Shurbaji, T., Anbar, M., Manickam, S., Hasbullah, I. H., ALfriehate, N., Alabsi, B. A., ... Hashim, H. (2025). Deep learning-based intrusion detection system for detecting iot botnet attacks: a review. *IEEE Access*.

Baccour, E., Mhaisen, N., Abdellatif, A. A., Erbad, A., Mohamed, A., Hamdi, M., & Guizani, M. (2022). Pervasive ai for iot applications: A survey on resource-efficient distributed artificial intelligence. *IEEE Communications Surveys & Tutorials*, *24*(4), 2366–2418.

Benameur, R., Dahane, A., Souihi, S., & Mellouk, A. (2024). A novel federated learning based intrusion detection system for iot networks. In *Icc 2024-ieee international conference on communications* (pp. 2402–2407).

Bhale, P., Chowdhury, D. R., Biswas, S., & Nandi, S. (2023). Optimist: lightweight and transparent ids with optimum placement strategy to mitigate mixed-rate ddos attacks in iot networks.

IEEE Internet of Things Journal, *10*(10), 8357–8370.

Bian, D., & Liu, J. (2025). Gmcwae: A representation learning technique for network intrusion detection in iot. *IEEE Internet of Things Journal*.

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., & Den Hartog, F. T. (2021). Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal*, *9*(1), 485–496.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys & Tutorials*, *21*(3), 2671–2701.

Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chou, D., & Jiang, M. (2021). A survey on data-driven network intrusion detection. *ACM Computing Surveys (CSUR)*, *54*(9), 1–36.

Farhat, A., Eldosouky, A., Ibnkahla, M., & Matrawy, A. (2025). Interaction-aware trust management scheme for iot systems with machine learning-based attack detection. *IEEE Internet of Things Journal*.

Friha, O., Ferrag, M. A., Shu, L., Maglaras, L., Choo, K.-K. R., & Nafaa, M. (2022). Felids: Federated learning-based intrusion detection system for agricultural internet of things. *Journal of Parallel and Distributed Computing*, *165*, 17–31.

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Gyamfi, E., & Jurcut, A. D. (2022). Novel online network intrusion detection system for industrial iot based on oi-svdd and as-elm. *IEEE Internet of Things Journal*, *10*(5), 3827–3839.

He, K., Kim, D. D., & Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *25*(1), 538–566.

He, Y., & Xiao, L. (2023). Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *46*(5), 2900–2919.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, *100*, 779–796.

Kulkarni, U., Hosamani, A. S., Masur, A. S., Hegde, S., Vernekar, G. R., & Chandana, K. S. (2022). A survey on quantization methods for optimization of deep neural networks. In *2022 international conference on automation, computing and renewable systems (icacrs)* (pp. 827–834).

Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, *461*, 370–403.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., ... Miao, C. (2020). Federated learning in mobile edge networks: A comprehensive survey. *IEEE communications surveys & tutorials*, *22*(3), 2031–2063.

Naeem, F., Ali, M., & Kaddoum, G. (2023). Federated-learning-empowered semi-supervised active learning framework for intrusion detection in zsm. *IEEE Communications Magazine*, *61*(2), 88–94.

Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Poor, H. V. (2021). Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *23*(3), 1622–1658.

Ouyang, Y., Li, B., Kong, Q., Song, H., & Li, T. (2021). Fs-ids: a novel few-shot learning based intrusion detection system for scada networks. In *Icc 2021-ieee international conference on communications* (pp. 1–6).

Parnami, A., & Lee, M. (2022). Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*.

Rahman, M. M., Al Shakil, S., & Mustakim, M. R. (2025). A survey on intrusion detection system in iot networks. *Cyber Security and Applications*, *3*, 100082.

Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., Madzudzo, G., & Cheah, M. (2023). Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Computing Surveys*, *55*(11), 1–40.

Raju, V. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., & Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *2020*

third international conference on smart systems and inventive technology (icssit) (pp. 729–735).

Shafin, S. S., Karmakar, G., & Mareels, I. (2023). Obfuscated memory malware detection in resource-constrained iot devices for smart city applications. *Sensors*, *23*(11), 5348.

Shah, Y., & Sengupta, S. (2020). A survey on classification of cyber-attacks on iot and iiot devices. In *2020 11th ieee annual ubiquitous computing, electronics & mobile communication conference (uemcon)* (pp. 0406–0413).

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, *30*.

Song, Y., Wang, T., Cai, P., Mondal, S. K., & Sahoo, J. P. (2023). A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*, *55*(13s), 1–40.

Tong, J., & Zhang, Y. (2024). A real-time label-free self-supervised deep learning intrusion detection for handling new type and few-shot attacks in iot networks. *IEEE Internet of Things Journal*.

Wu, C., Sun, J., Chen, J., Alazab, M., Liu, Y., & Xiang, Y. (2025). Tcg-ids: Robust network intrusion detection via temporal contrastive graph learning. *IEEE Transactions on Information Forensics and Security*.

Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*.

Yang, K., Wang, J., Zhao, G., Wang, X., Cong, W., Yuan, M., … Tao, J. (2025). Nids-cnnrf integrating cnn and random forest for efficient network intrusion detection model. *Internet of Things*, 101607.

Yao, W., Peng, H., Li, Q., & Shen, X. (2025). Modeling realistic adversarial traffic against deep learning-based intrusion detection system in industrial iot. *IEEE Internet of Things Journal*.

Zha, C., Wang, Z., Fan, Y., Bai, B., Zhang, Y., Shi, S., & Zhang, R. (2025). A-nids: adaptive network intrusion detection system based on clustering and stacked ctgan. *IEEE Transactions on Information Forensics and Security*.

Zhang, J., Luo, C., Carpenter, M., & Min, G. (2022). Federated learning for distributed iiot intrusion

detection using transfer approaches. *IEEE Transactions on Industrial Informatics*, *19*(7), 8159–8169.

Zhao, R., Gui, G., Xue, Z., Yin, J., Ohtsuki, T., Adebisi, B., & Gacanin, H. (2021). A novel intrusion detection method based on lightweight neural network for internet of things. *IEEE Internet of Things Journal*, *9*(12), 9960–9972.

Zhao, R., Wang, Y., Xue, Z., Ohtsuki, T., Adebisi, B., & Gui, G. (2022). Semisupervised federated-learning-based intrusion detection method for internet of things. *IEEE Internet of Things Journal*, *10*(10), 8645–8657.