IMT School for Advanced Studies, Lucca

and

Università degli Studi di Milano, Milan
Italy

Robust AI-based DDoS Attack Detection in Complex Scenarios

PhD Program in Cybersicurezza

Track in Software, System, and Infrastructure Security

XXXVIII Cycle

By

Mulualem Bitew Anley

2026

The dissertation of Mulualem Bitew Anley

PhD Program Coordinator: Mirco Tribastone, IMT School for Advanced Studies Lucca

Advisor: Prof. Vincenzo Piuri, Università degli Studi di Milano

Co-Advisor: Prof. Angelo Genovese, Università degli Studi di Milano

Co-Advisor: Prof. Pierangela Samarati, Università degli Studi di Milano

The dissertation of Mulualem Bitew Anley has been reviewed by:

Cristina Alcaraz, Universidad de Malaga, Spain

Radu-Emil Precup, Polytechnic University of Timișoara, Romania

IMT School for Advanced Studies Lucca
2026

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I thank God and Saint Mary for sustaining me throughout this PhD journey.

I am deeply grateful to my supervisor, Prof. Vincenzo Piuri, for his unwavering support from day one, his generosity, and the breadth of his scientific vision, which steadied me in difficult times, and for his trust throughout this journey. I also thank my co-supervisor, Prof. Angelo Genovese, for his guidance at every step from drafting to publication and for the close, collaborative spirit that shaped this work. My sincere thanks also go to Prof. Pierangela Samarati for her constant kindness, timely advice, and for broadening my perspective on data security and privacy.

I would like to thank everyone who assisted with administrative procedures, especially Prof. Pasquale Coscia and Prof. Ruggero Donida Labati, for their continued support. I am also grateful to my colleagues at the University of Milan, members of the SPDP and IEBIL laboratories, for their help and camaraderie. I further thank the IMT School for Advanced Studies Lucca, the PhD Program Coordinator, and the administrative staff for their support.

I am grateful to my hosts during my research stay at NTNU (NORCICS), Norway, Prof. Sokratis Katsikas, Prof. Georgios Spathoulas, and Prof. Jia-Chun Lin, for their warm hospitality and support, collaborative work, and to the colleagues of the Department of Information Security and Communication Technology (IIK) for their constructive feedback. I also thank Dr. Tibebe Beshah for his unwavering support and for hosting my seminar in Addis Ababa, and Dr. Worku Abebe at the University of Gondar for hosting and collaborating.

My deepest gratitude goes to my father and my family for their endless love and encouragement. With much love, I thank my wife, Rediet Bereket, and our sons, Fiyazan Mulualem and Amesiyas Mulualem, who have shared every up and down of this journey. Rediet has been my anchor, shouldering daily responsibilities, offering steadfast emotional support, and caring for our children during the most demanding periods. To my little ones, watching you grow has been my greatest motivation; you are the reason I kept going.

I also acknowledge the use of an AI tool (ChatGPT, OpenAI) solely to support language editing for clarity and readability. All scientific contributions, technical content, analyses, interpretations, and conclusions are entirely my own, and I reviewed and verified all changes.

I also acknowledge the use of an AI tool (ChatGPT, OpenAI) solely to support language editing for clarity and readability. All scientific contributions, technical content, analyses, interpretations, and conclusions are entirely my own, and I reviewed and verified all changes.

Publications

1. **Journal Articles**
   - M. B. Anley, P. Coscia, A. Genovese, and V. Piuri, "FELACS: Federated learning with adaptive client selection for IoT DDoS attack detection", *Computers & Security*, vol. 158, art. 104642, Nov. 2025, pp. 1–13. ISSN: 0167-4048. DOI: 10.1016/j.cose.2025.104642
   - M. B. Anley, A. Genovese, D. Agostinello, and V. Piuri, "Robust DDoS attack detection with adaptive transfer learning", *Computers & Security*, vol. 144, art. 103962, Sep. 2024, pp. 1–10. ISSN: 0167-4048. DOI: 10.1016/j.cose.2024.103962
2. **International Conferences**
   - M. B. Anley, A. Genovese, T. B. Tesema, and V. Piuri, "FLIFRA: Hybrid data poisoning attack detection in federated learning for IoT security", in *Proc. of the 2025 IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC 2025)*, Vienna, Austria, October 5-8, 2025, pp. 1-8.
   - M. B. Anley, A. Genovese, and V. Piuri, "TransferEdge: Transfer learning approach to detect evolving DDoS threats in Edge-IIoT", in Proc. of the 9th Int. Forum on Research and Technologies for Society and Industry (RTSI 2025), Gammarth, Tunisia, August 24-26, 2025, pp. 11-16.
   - M. B. Anley, O. Ekpo, TM.H. Gedara, "Cybersecurity Assessment of Digital Twin in Smart Grids", in *Proc. of the 8th Italian Conference on Cyber Security (ITASEC 2024)*, Salerno, Italy, April 8-12, 2024.
3. **Book Chapter**
   - M. B. Anley, A. Genovese, and V. Piuri, "Deep Learning for DDoS attack detection in IoT: A survey", in Security and Cryptography. SECRYPT 2023/SECRYPT 2024, ser. Communications in Computer and Information Science, P. Samarati, and S. De Capitani di Vimercati (eds.), vol. 2588, Springer, Cham, 2026, pp. 99-121. ISBN: 978-3-032-09598-5.

Abstract

The world is now pervasively connected, and billions of Internet of Things (IoT) devices are integrated across healthcare, energy, transportation, industry, and homes, so local faults can quickly escalate into systemic, cross-border risks. In this context, Distributed Denial of Service (DDoS) attacks can disrupt safety-critical services, degrade industrial control systems, and propagate through heterogeneous cloud edge infrastructures. Modern botnets execute multi-vector DDoS attacks, combining volumetric floods with application-layer assaults that overwhelm bandwidth and server resources while outpacing existing detection approaches. Artificial intelligence (AI)-based detection is further constrained by scarce and imbalanced labels, distribution shifts across time and domains, strict latency and resource budgets at gateways and edge nodes, and privacy compliance limits on central aggregation. Meanwhile, the lowered barrier to data poisoning and model manipulation raises the stakes for robust, privacy-preserving AI.

Existing approaches span signature and rule-based systems, statistical anomaly detection, and classical ML. Yet signatures and hand-tuned thresholds break under unseen variants and adaptive adversaries, while traditional models depend on brittle manual features. Deep Learning (DL) improves detection, but many studies rely on dated datasets, lack principled model tuning, and struggle with minority attack classes. Centralized training also concentrates sensitive traffic, raising privacy and compliance concerns. Federated learning reduces data movement, but real deployments face non-IID data, device variability, inefficient random client selection, and exposure to poisoning, backdoor, and evasion attacks.

The purpose of the thesis is to design a robust methodology for AI-based DDoS detection in complex scenarios by designing an adaptive, scalable, privacy-preserving, and resilient framework. This thesis asks how to build detectors that remain accurate, efficient, and trustworthy under these complex, large-scale conditions. We address this with a three-part methodology. First, we build a centralized DL baseline that standardizes preprocessing, mitigates class imbalance, reduces dimensionality, and systematically tunes models to achieve accurate detection at low false positive rates. Second, we introduce *TransferEdge*, which reuses pretrained models and applies dataset-aware adaptation so edge systems can maintain accuracy with fewer labels and lower compute, even when environments shift. Third, we develop a federated learning track that couples *FELACS* with an adaptive client-selection strategy that prioritizes high-impact participants to speed convergence and improve accuracy with *FLIFRA*, a dual-layer defense that screens suspicious updates at the client and downweights them at the server to resist poisoning. We evaluate across multiple public datasets and stress conditions, showing faster learning, higher accuracy, and stronger robustness under realistic constraints.

In summary, our analysis reveals that a carefully tuned centralized baseline provides strong detection capability under realistic constraints, *TransferEdge* enables label and compute efficient adaptation as environments shift, and, in distributed settings, *FELACS* speeds learning and improves accuracy by prioritizing the most informative clients while *FLIFRA* strengthens resilience against adversarial manipulation without exposing raw data. Collectively, these components deliver an end-to-end, privacy-preserving, and attack-resilient AI framework for scalable DDoS detection in complex IoT ecosystems.

Abbreviations

Acc — Accuracy
AE — Autoencoder
AI — Artificial Intelligence
AUROC — Area Under the Receiver Operating Characteristic
CNN — Convolutional Neural Network
DDoS — Distributed Denial of Service
DT — Decision Trees
DL — Deep Learning
DNN — Deep Neural Network
DNS — Domain Name System
DRRA — Dynamic Reputation-based Robust Aggregation
F1 — F-measure
FedAvg — Federated Averaging
FELACS — Federated Learning with an Adaptive Client Selection
FedProx — Federated Proximal
FL — Federated Learning
FLIFRA — Federated Learning Isolation Forest with Robust Aggregation
FPR — False Positive Rate
FTP — File Transfer Protocol
FTL — Federated Transfer Learning
GD — Gradient Descent
GRU — Gated Recurrent Unit
HPO — Hyperparameter Optimization
HTTPS — Hypertext Transfer Protocol Secure
IDS — Intrusion Detection System
ICMP — Internet Control Message Protocol
iForest — Isolation Forests

IoT/IIoT — (Industrial) Internet of Things
IID — Independent & Identically Distributed
ISP — Internet Service Provider
KNN — K-Nearest Neighbors
LAN — Local Area Network
LSTM — Long Short-Term Memory
LR — Learning Rate
NB — Naive Bayes
NIDS — Network Intrusion Detection System
NSGA-II — Non-dominated Sorting Genetic Algorithm II
ML — Machine Learning
MLP — Multi-Layer Perceptron
OSI — Open Systems Interconnection
PCA — Principal Component Analysis
Prec — Precision
Rec — Recall
ResNet — Residual Network
RL — Reinforcement Learning
RNN — Recurrent Neural Network
SDN — Software Defined Network
SGD — Stochastic Gradient Descent
SQL — Structured Query Language
SNMP — Simple Network Management Protocol
SSL — Secure Sockets Layer
SVM — Support Vector Machine
TCP — Transmission Control Protocol
TL — Transfer Learning
UDP — User Datagram Protocol
VGG — Visual Geometry Group

Chapter 1

Introduction

1.1 Motivation

IoT devices are now integral to daily life, from personalized health monitors and smart home appliances to city-scale infrastructures, but this ubiquity both centralizes sensitive data and expands the attack surface. Among the most pernicious threats are intrusion attempts and DDoS attacks, which saturate network resources with malicious traffic, degrading availability and reliability, and, in safety-critical settings such as healthcare monitoring, putting human lives at risk.

As networks and IoT ecosystems scale, adversaries exploit the enlarged surface to launch multi-vector, high-volume DDoS attacks whose scale and complexity outpace traditional detection and delay effective mitigation. Detection approaches generally fall into signature-based and anomaly-based. Signature-based methods match predefined attack "signatures" to identify known threats, while anomaly-based methods use statistical or ML models to detect deviations from normal behavior and thereby surface novel or previously unseen attacks. Within ML, DL models are increasingly favored for their strong classification performance and ability to learn feature representations automatically, enabling highly accurate detectors that can better adapt to the evolving characteristics of DDoS traffic over time.

Despite their promise, DL-based DDoS detectors for IoT are constrained by the scarcity of high-quality labels, especially for emerging variants, and a severe class imbalance that biases training and degrades detection. Centralized training further heightens privacy risk by concentrating sensitive data and is often impractical in real deployments. To address these challenges, this thesis proposes a privacy-preserving, robust, and scalable framework for IoT DDoS defense. The proposed approach unifies centralized, transfer, and federated learning with targeted algorithmic optimizations to balance detection accuracy, computational efficiency, and data confidentiality. Acknowledging the susceptibility of federated systems to poisoning, this thesis introduces a dual-layer defense to detect and mitigate malicious contributions. Through rigorous theoretical analysis and comprehensive empirical evaluation, this thesis delivers both foundational contributions and practical mechanisms for securing next-generation connected systems. Detailed methodologies, experiments, and results appear in the chapters that follow.

1.2 Research Questions & Problem Statement

IoT environments are increasingly targeted by sophisticated DDoS attacks, yet existing DL solutions struggle with limited labeled data, severe class imbalance, and the privacy risks of centralized training, while resource constraints on edge devices and the threat of poisoning attacks in federated systems further complicate reliable deployment. In addition, the interaction between detection accuracy, computational overhead, communication efficiency, and fairness among heterogeneous IoT nodes remains underexplored, particularly when comparing centralized architectures, transfer-learning fine-tuning strategies, and federated learning paradigms. To bridge these gaps, this thesis designs a robust, unified, privacy-preserving framework that optimizes model design, adaptation, and collaboration under real-world constraints, motivating the following research questions:

- **RQ1.** What detection accuracy do centralized DL architectures

(DNN, CNN, RNN, LSTM), tuned via HPO, achieve on binary (attack vs. benign) and multi-class (specific DDoS types) classification?

- **RQ2.** How can transfer learning via targeted fine-tuning strategies be optimized to detect evolving DDoS patterns with limited labels and minimal retraining, while maximizing accuracy and computational efficiency on resource-constrained devices?
- **RQ3.** How can FL (client selection and aggregation) preserve privacy, sustain accuracy under IID and non-IID data, minimize communication, and maintain fairness across heterogeneous IoT nodes?
- **RQ4.** How effective is a dual-layer defense combining local anomaly filtering with global trust-based aggregation in identifying and mitigating poisoning attacks in federated IoT DDoS detection, and what is its impact on overall robustness and privacy?

1.3 Research Objective

This thesis aims to build a robust, adaptive, scalable, privacy-preserving, and resilient framework for DDoS detection across complex scenarios. The proposed approach employs advanced DL architectures, optimized through layer-wise architecture search and HPO, to extract rich traffic representations. Because limited and imbalanced DDoS data can induce overfitting in high-capacity networks, it leverages transfer learning by fine-tuning models pretrained on large public intrusion-detection datasets, thereby accelerating convergence, improving generalization on small IoT datasets, and preserving low-level temporal patterns. To respect privacy, bandwidth, and system-heterogeneity constraints, this thesis adopts federated learning with a multi-objective client-selection scheme that balances computation, communication, and data diversity, enabling collaborative training without sharing raw logs. To ensure the integrity of distributed training under poisoning, this thesis proposes a dual-layer defense that includes a lightweight iForest on clients to filter anomalous local updates, paired with a DRRA algorithm on the server that weights contributions by trust.

1.4 Contributions of the Thesis

This thesis proposes a robust AI-based pipeline for DDoS attack detection across complex scenarios. The pipeline advances through four stages: (i) establishing strong centralized baselines via exhaustive HPO for both binary and multi-class DDoS detection; (ii) leveraging transfer learning to adapt pretrained models with minimal labeled data; (iii) enabling privacy-preserving scalability through FL with a novel, multi-objective client-selection scheme, and (iv) robust poisoning attack detection in federated learning with the dual-layer approaches. Together, these stages balance accuracy, communication efficiency, and device constraints, while improving resilience to data poisoning, as summarized in Figure 1.

- **Deeply Adaptive Centralized Baseline for Learning-Based DDoS Detection.**
  This thesis conducts a fair, dataset-agnostic comparison of centralized deep-learning model families DNNs, CNNs, RNNs, and LSTMs under a common HPO protocol and a standardized preprocessing pipeline. It reports accuracy, precision, recall, macro-averaged F1 (macro-F1), and AUROC, establishing the methodological basis for subsequent stages

- **Transfer Learning based Evolving Attack Detection.**
  This thesis designs transfer-learning strategies that fine-tune pretrained backbones (VGG8/16/19, ResNet50, and custom Conv4/8/18) to accelerate convergence and improve generalization under limited and imbalanced labels. We study head re-initialization, layer freezing, and lightweight adapters, demonstrating data-efficiency gains over training from scratch.

- **Federated Learning With Adaptive Client Selection.**
  This thesis develops an FL scheme that jointly optimizes model quality, communication cost, and client resource constraints. Raw traffic remains local; only model updates are shared. A custom client-selection strategy balances data diversity, fairness, and effi-

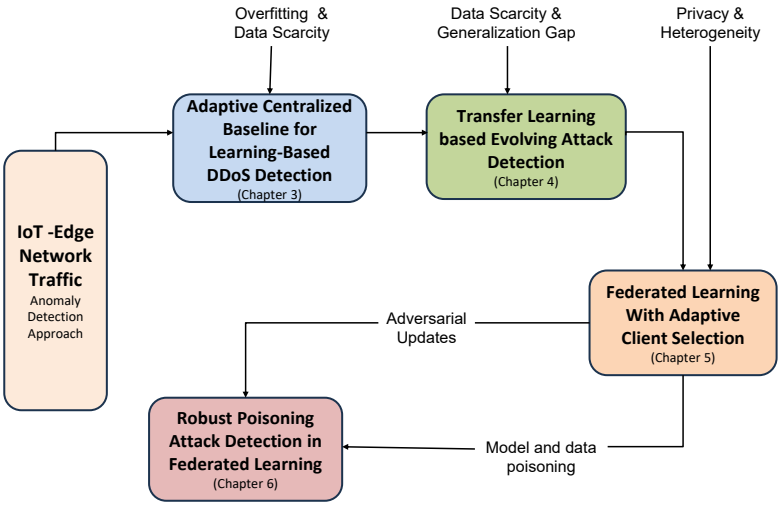


Figure 1: Workflow of the thesis contributions paradigm, illustrating each methodological module, the gaps it addresses, and the corresponding solutions in a step-by-step structure.

ciency, improving rounds-to-target-accuracy and bytes transmitted under IID and non-IID participation.

- **Robust Poisoning Attack Detection in Federated Learning.** This thesis introduces a poisoning attack defense-in-depth mechanism for FL, combining client-side iForest to flag anomalous local updates with server-side DRRA that weights contributions by trust over time. This mitigates label model-poisoning while preserving benign-case utility.

1.5 Organization of the Thesis

The structure of this thesis is as follows:

- *Chapter 2: Literature Review*. This chapter surveys existing work on DDoS detection, beginning with a comprehensive taxonomy that contrasts signature-based, anomaly-based, and hybrid detection methods. Then, review state-of-the-art DL approaches covering key CNN and RNN architectures, feature-learning techniques, and hyperparameter-search methods. Next, examine TL applications in cybersecurity, detailing how pretrained intrusion models have been adapted via various fine-tuning paradigms. The chapter continues with federated learning research in IoT security, highlighting client selection strategies and aggregation defenses. Finally, this thesis discusses poisoning-attack models and the latest robust aggregation techniques, thereby setting the stage for the proposed dual-layer defense.

- *Chapter 3: Deeply Adaptive Centralized Baseline for Learning-Based DDoS Detection*. This chapter presents the proposed custom DL framework for DDoS detection. After defining the datasets and preprocessing pipeline, it introduces DNN, CNN, LSTM, and RNN architectures and describes the proposed layer-wise architecture search and hyperparameter optimization process. Baselines are compared in binary (attack vs. benign) and multi-class (multiple attack types) classification performance, reporting precision, recall, and F1 across folds. Through ablation studies that isolate the impact of depth, filter sizes, and learning rates, demonstrating how each design choice closes the overfitting & data scarcity gap identified in Chapter 2.

- *Chapter 4: Transfer Learning based Evolving Attack Detection*. Building on the centralized DL baselines, this chapter introduces fine-tuning strategies to leverage large, publicly available intrusion-detection datasets. Three adaptation schemes are evaluated: (i) freezing early convolutional layers, (ii) reinitializing higher layers, and (iii) selective layer adaptation; decision criteria for choosing among them are formalized. A suite of experiments compares convergence speed and generalization on the benchmark datasets, quantifying improve-

ments in detection accuracy and training efficiency. The results validate how transfer learning overcomes data scarcity and accelerates model detections in resource-constrained settings.

- *Chapter 5: Federated Learning With Adaptive Client Selection*. This chapter transitions to a decentralized training setup that respects IoT device privacy and connectivity constraints. It formalizes a multi-objective client-selection algorithm that jointly optimizes device compute capability, data representativeness, and communication budget. The federated training protocol and aggregation rules are specified, and simulation results on heterogeneous client populations are reported. The analysis covers bandwidth utilization, convergence behavior, and per-client detection performance, showing that the approach bridges the privacy and heterogeneity gap while preserving model quality.

- *Chapter 6: Robust Poisoning Attack Detection in Federated Learning*. This chapter addresses adversarial threats to the proposed federated detector. It first defines the poisoning threat model clients submitting malicious updates to degrade the global model, and then introduces a dual-layer defense that combines on-device anomaly filtering via lightweight iForest with server-side DRRA. Controlled poisoning experiments quantify robustness gains, false-positive trade-offs, and performance recovery under attack. Results indicate that the defense framework secures collaborative training without compromising detection accuracy.

- *Chapter 7: Conclusions and Future Work*. The final chapter summarizes the thesis contributions: from deep learning optimization through transfer and federated learning innovations to poisoning attack defenses. It reflects on the limitations encountered, such as the representativeness of the dataset and real-world deployment challenges, and distills key lessons. Finally, the chapter outlines promising future research directions, including adaptive client-selection heuristics, continual learning under non-stationary attacks, and

hardware-aware model compression for on-device inference. The structure of this thesis is summarized in Figure 1.

Chapter 2

Literature Review

IoT technologies continue to expand across areas such as personal healthcare, home automation, industrial control systems, and smart-city infrastructures, with the number of interconnected devices steadily increasing. This growing ecosystem enlarges the potential attack surface for DDoS campaigns, whose rate, magnitude, and sophistication are now surpassing the capabilities of traditional detection approaches. ML, especially DL, has emerged as a strong candidate for detecting and mitigating IoT-driven DDoS attacks; however, several practical challenges persist, including scarce and imbalanced labeled data, shifts in data distributions, concept drift, and the privacy risks associated with centralized training. The increasing adoption of large language models and automated tooling further lowers the barrier for adversaries to mount poisoning attacks, intensifying concerns around data privacy. This chapter (i) introduces a taxonomy of DDoS attacks in IoT systems, (ii) describes their core characteristics and evolution, (iii) surveys detection strategies across centralized, transfer learning, and federated learning frameworks, and (iv) analyzes poisoning threats in federated learning together with the corresponding

The content of this chapter is based on the published work: M.B. Anley, A. Genovese, and V. Piuri, "Deep Learning for DDoS Attack Detection in IoT: A Survey," in *Security and Cryptography. SECRYPT 2023 / SECRYPT 2024*, P. Samarati and S. De Capitani di Vimercati (eds.), Communications in Computer and Information Science, vol. 2588, Springer, Cham, 2026, pp. 99–121. Reproduced with permission from Springer Nature.

defense mechanisms, including the available datasets [1].

2.1 Taxonomy of DDoS Attacks

This section reviews the definition and evolution of DDoS attacks, their motivations and targets, and the main taxonomy used to classify them. The taxonomy adopted here is in the published work by the structure introduced in [1] and has been further extended and adapted for this thesis.

2.1.1 DDoS Attack: A Definition

DDoS attacks typically occur when an adversary compromises numerous devices commonly referred to as bots or zombies by infecting them with malware that enables remote manipulation. The attacker, known as the botmaster, coordinates these compromised devices to form a botnet, which can include hundreds to millions of hosts, with IoT devices often being preferred targets due to their limited security protections [6,7]. Botnets serve as the main infrastructure for launching DDoS attacks, in which large volumes of malicious traffic are directed at a victim server with the intent of overwhelming resources and interrupting normal service operations [8,9] [1].

A recent development in this ecosystem is the emergence of DDoS-as-a-Service (DDoSaaS), which enables individuals to rent botnets for executing DDoS campaigns. These services are commonly found on dark-web marketplaces and underground forums, offering customizable attack characteristics, including duration, traffic intensity, and selected targets, often with payments conducted via cryptocurrencies to protect user anonymity [10]. This service-based model has drastically reduced the technical expertise required to mount impactful attacks, making it possible for non-experts to launch disruptive operations against businesses and online platforms. The combination of large-scale botnets and easily accessible DDoSaaS platforms continues to expand the magnitude, complexity, and availability of DDoS attack capabilities [7], [1].

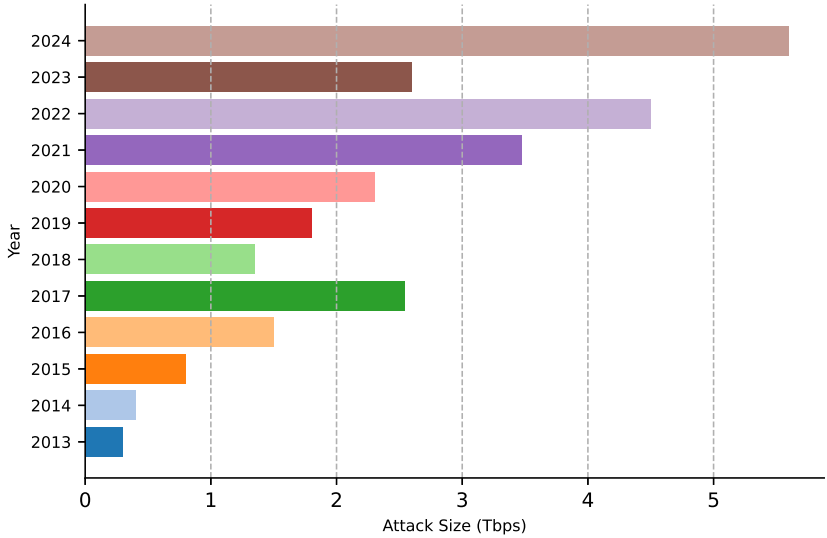


Figure 2: Annual trend of the largest DDoS attacks, measured in Tbps, over the past years [1].

2.1.2 The Evolution of DDoS Attacks

DDoS attacks have undergone substantial transformation since their early days, increasing in sophistication, magnitude, and overall effect. This progression mirrors advancements in digital technologies and the growing dependence on the Internet as a critical component of global infrastructure [11]. Figure 2 presents the largest DDoS incidents recorded in Tbps over the past 12 years. The values shown are compiled from [1, 12–17].

The origins: 1974-2007.

- One of the earliest documented cases resembling a denial-of-service event occurred in 1974, when a 13-year-old student accidentally caused a system failure at the University of Illinois. The incident, involving David Dennis and the PLATO system, exposed how inherent system constraints could be exploited intentionally or not to halt operations. This version is frequently referenced as an early illustration of network disruption and underscores the vulnerabilities

present in shared computing platforms [14].

- In 1988, the Morris Worm rapidly spread across thousands of ARPANET hosts, severely disrupting network operations and becoming one of the earliest large-scale incidents of its kind. The worm exploited vulnerabilities in UNIX systems and is often cited as a landmark example of early network-borne attacks, originally intended to approximate the size of the ARPANET [18].

- In 1996, Panix, one of the oldest internet service providers, experienced what is widely recognized as the first recorded DDoS incident, remaining offline for several days after a large-scale SYN Flood attack. Nonetheless, [16] notes another milestone event: on 22 July 1999, the University of Minnesota's network was overwhelmed within minutes by a coordinated assault from 114 compromised machines running the Trin00 attack script, marking an early example of botnet-driven DDoS activity.

- In 2000, a high-profile attack by a Canadian teenager known as Mafiaboy targeted major websites such as CNN, Yahoo, and eBay, demonstrating the potential for significant economic and reputational damage [19].

- In July 2001, the Code Red worm spread rapidly, and by the time it was identified on July 19, it had already compromised nearly 359,000 Microsoft IIS servers, resulting in billions of dollars in losses. The malicious payload operated solely in system memory, leaving no artifacts on persistent storage [1].

- In January 2003, the SQL Slammer worm propagated with unprecedented rapidity by exploiting a vulnerability in Microsoft SQL servers. Its outbreak led to widespread service disruptions worldwide and, according to estimates, compromised roughly 75,000 machines across various regions in under half an hour [1].

- In 2007, Estonia experienced a wave of severe DDoS attacks during a period of heightened political tension with Russia, an event

widely regarded as one of the earliest examples of cyber warfare. The coordinated assaults disrupted multiple sectors simultaneously, including government services, media outlets, and financial institutions [12, 14].

Botnets and IoT vulnerabilities: The 2010s until now.

- In 2013, Spamhaus, a prominent non-profit organization specializing in anti-spam and DDoS protection, was targeted by a major attack. The incident, reaching an estimated peak of 300 Gbps, quickly overwhelmed its defenses, leading to service outages and notable reputational impact [12, 14].
- In 2014, CloudFlare was targeted by a DDoS attack that reached an estimated peak of 400 Gbps, leveraging a flaw in the widely deployed NTP protocol [13].
- In 2016, Dyn was targeted by a major DDoS attack powered by the Mirai botnet, which had compromised more than 600,000 IoT devices. Estimates suggest that the attack peaked at around 1.5 Tbps, overwhelming Dyn's DNS infrastructure and causing widespread service outages. As a result, several high-profile platforms, including GitHub, Netflix, PayPal, and Airbnb became temporarily inaccessible [12] [1].
- In February 2018, GitHub was hit by a massive DDoS attack that leveraged a Memcached amplification vulnerability. The attacks peaked at 1.35 Tbps and lasted for about 20 minutes, temporarily overwhelming the platform's DDoS protections and disrupting access to its services.
- In February 2020, AWS was targeted by a sustained DDoS attack lasting three days, reaching peak traffic volumes of 2.3 Tbps. The incident leveraged an LDAP-related vulnerability and, while it resulted in minimal service interruption, it did inflict notable reputational consequences for the cloud provider [12, 14].

- In June 2022, Google announced that it had successfully mitigated a 4.5 Tbps HTTP DDoS attack, marking the largest application-layer attack recorded to date [15].

- In February 2023, a widespread ransomware attack group targeted VMware ESXi servers worldwide. On February 4, the CSIRT [16] issued an advisory reporting that the attack leveraged a previously known vulnerability (CVE-2021-21974) affecting VMware ESXi installations. Although VMware had released a patch for this flaw in early 2021, many systems remained unpatched and therefore exposed. Estimates from the U.S. Cybersecurity and Infrastructure Security Agency (CISA) and the National Security Agency (NSA) indicated that more than 3,800 servers were compromised during the incident.

- In 2024, DDoS attacks reached record-breaking volumes of up to 5.6 Tbps, marking a new peak in large-scale volumetric attacks. These attacks were mainly driven by IoT-based botnets and reflection/amplification techniques (HTTP/2, DNS, NTP), often targeting cloud services, government platforms, and critical online infrastructure. The scale and automation of these attacks highlight the growing threat posed by insecure IoT devices and increasingly efficient attack orchestration [20].

2.1.3 DDoS: Motivation and Target

The motivations driving DDoS attacks are multifaceted and depend heavily on the intentions of the adversaries. Despite this variability, the underlying objectives can typically be grouped into several broad categories [21]. Table 1 illustrates some examples of damages to attacked systems, including the attackers' motivations, goals, impacts, methods, and levels of disruption.

Ransom. DDoS attacks, which overwhelm a target with excessive traffic, are frequently incorporated into extortion campaigns. In such scenarios,

Table 1: Summary of notable DDoS attacks, attackers' motivation, goals, methods, techniques, targets, and attack size [1].

Attacked system	Motivation	Goal	Method	Techniques	Target	Resources	Attack Size
Microsoft Azure [22]	Activist	Financial gain	DDoS	Botnet	Web	Network bandwidth	2.4 Tbps
Poland's railway [23]	Political	Disruption	DoS	Signal spoofing	Railway	Radio–frequency	Unknown
Akamai [24]	Unknown	Unknown	DDoS	Botnet	Web	Network bandwidth	633.7 Gbps
Killnet (US Hospital) [25]	Political	Revenge	DDoS	Botnet	Web	Network bandwidth	Unknown
KA-SAT NETWORK (VIASAT)	Warfare	Disruption	DoS	Exploitation	Satellite	Firmware	Weeks

attackers demand payment in exchange for stopping the disruptive activity. This tactic may also be used against business competitors to hinder their operations, ultimately providing the attacker with potential financial or market advantages [26] [1].

Ideological or political motivations. Adversaries driven by political or ideological agendas typically demonstrate strong technical expertise, firm beliefs, and often operate within organized groups. According to ENISA's most recent analysis of the DDoS/DoS threat landscape, approximately 66% of DoS incidents stem from political motives [10]. Although the goals of these operations differ, they are consistently rooted in ideological, political, or religious intent. A well-known example is the coordinated cyberattacks against Estonia in 2007 [27].

Cyber warfare and espionage. This category primarily includes operations driven by political or military objectives, typically associated with state or state-aligned actors. Such adversaries are highly coordinated and

well-resourced, with sufficient financial capacity and long-term planning capabilities to carry out sophisticated campaigns. Their attacks commonly focus on the critical infrastructure, communication networks, and financial systems of adversarial states [17].

Individual disappointment. In some situations, DDoS attacks stem from personal grievances rather than monetary incentives or ideological aims. Individuals motivated by frustration or conflict may single out particular organizations or people with the intention of causing disruption or damage [21].

Threat or testing. These attacks are often initiated by relatively inexperienced individuals seeking to enhance their technical skills, typically by leveraging existing botnets or assembling new ones. In many instances, such activities are exploratory in nature and carried out without a clear malicious objective, sometimes merely for experimentation or amusement [21]. In contrast, organized criminal groups may launch attacks as demonstrations of capability, using them to advertise the effectiveness of their botnets to potential buyers in underground markets. Additionally, human errors remain a significant contributor to cyber incidents, occasionally triggering denial-of-service conditions, as historically illustrated by the Morris Worm [18].

2.1.4 DDoS Attack Taxonomy

A variety of taxonomies have been introduced to classify DDoS attacks, taking into account elements such as attack layers, techniques, vectors, impacts, and newly emerging forms [28], [21]. Some works also differentiate attacks based on their intensity, separating low-rate from high-rate threats [8]. In this survey, we adopt the primary categorization criteria and group DDoS attacks according to the targeted network layer, the attack methodology, the vector used, the resulting impact, and the degree of sophistication involved (see Figure 3). The following sections elaborate on these categories and highlight several notable DDoS incidents.

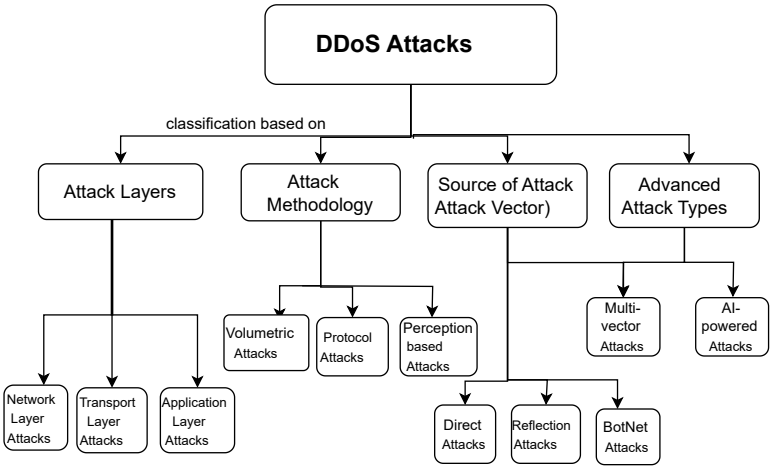


Figure 3: DDoS attack taxonomy by layer, methodology, source or attack vector, and emerging types [1].

Classification based on Attack Layers

This approach categorizes DDoS attacks based on which layer of the OSI model they target. Each layer has specific protocols and vulnerabilities that attackers exploit [21].

Network-layer attacks. This layer handles the routing and forwarding of data packets across interconnected networks. DDoS attacks targeting this layer attempt to exhaust network bandwidth and disrupt normal traffic flow. Below are several of the most prominent network-layer DDoS attack types [1].

1. **ICMP Flood.** In this attack, the adversary inundates the target with ICMP Echo Request (ping) packets. The victim attempts to answer each request with an Echo Reply, consuming both inbound and outbound bandwidth and ultimately causing severe network congestion [29].

2. **SYN Flood.** In this type of attack, an adversary sends a large volume

of TCP/SYN packets, often using spoofed source addresses, to a specific port on the target system. The server replies with SYN/ACK messages and waits for the final step of the TCP handshake, which never occurs. As these half-open connections accumulate, the system's resources are gradually depleted [30].

3. **Smurf Attack.** This attack leverages ICMP messages and IP broadcast addresses to inundate a victim with excessive traffic. The attacker sends ICMP Echo Requests to a network's broadcast address while spoofing the victim's IP, causing all reachable hosts to reply and collectively overwhelm the target [31].

Transport-layer attacks. The transport layer is responsible for end-to-end data transmission across networks, providing mechanisms for error detection and flow control. DDoS attacks targeting this layer seek to overwhelm servers or network devices by disrupting normal data flow. Common transport-layer DDoS attack subtypes include:

1. **SYN Flood.** This attack takes advantage of the TCP three-way handshake. The attacker sends a large number of TCP/SYN packets, often using spoofed IP addresses, to the target server. The server replies with SYN-ACK messages and waits for the final ACK, which never arrives. As these half-open connections accumulate, the server's resources become depleted [30].

2. **UDP Flood.** Because UDP is a connectionless protocol, an attacker can overwhelm a victim by sending several UDP packets to randomly selected ports. The targeted system must check whether any service is listening on each port and, if not, generate an ICMP "Destination Unreachable" response. This repeated processing consumes significant system and network resources [30].

3. **Amplification Attacks.** These attacks commonly occur over both UDP and TCP, where the adversary sends small requests to a third-party server while spoofing the victim's IP address. The server then sends its much larger response to the unsuspecting target. By

exploiting protocols with high amplification potential, including DNS or NTP, the attacker can generate massive volumes of traffic, overwhelming the target's resources [31].

Application-layer attacks. The application layer plays a central role in network communication, enabling services such as email, web browsing, and file transfer that support numerous user applications. DDoS attacks at this layer focus on overwhelming the resources of web servers, application servers, or specific web applications, ultimately slowing them down or rendering them inaccessible to legitimate users [32]. Some of the most common application-layer DDoS attack types are listed below:-

1. **HTTP Flood.** In this attack, a web server is overwhelmed by a large volume of HTTP requests. Unlike volumetric attacks that rely on malformed packets or massive traffic bursts, HTTP floods use legitimate-looking requests. The intensity and volume alone, however, are sufficient to exhaust the server's processing capacity [30].

2. **DNS Flood.** Although often viewed as an infrastructure-level threat, DNS floods can fall under application-layer attacks when the objective is to disrupt DNS services supporting a web application. The attacker bombards the DNS server with large numbers of queries, pushing it beyond its handling capacity and causing the associated application to become slow or unreachable [32].

3. **SSL Attacks.** SSL-related application-layer attacks take advantage of weaknesses in SSL/TLS mechanisms to undermine secure communication. Examples include SSL renegotiation and SSL stripping, which can inject unauthorized content or downgrade encrypted sessions, compromising security. Vulnerabilities such as the Heartbleed bug in TLS libraries may also be exploited for DDoS purposes, enabling attackers to leak sensitive data or disrupt critical services [32].

Classification by Attack Method

This classification looks at the technique or method employed by the attack.

1. **Volumetric DDoS attacks.** These attacks seek to saturate the victim's available bandwidth by generating massive amounts of traffic. Common examples include UDP Floods, ICMP Floods, amplification and DNS reflection attacks, SYN Floods, and IP/ICMP fragmentation [31].

2. **Protocol.** This category classifies DDoS attacks based on the specific communication protocols they aim to disrupt, such as HTTP, DNS, SMTP, NTP, VoIP, and ICMP. While these protocols are generally designed to manage resources efficiently [31], they remain susceptible to DDoS exploitation, particularly those that see the highest usage.

3. **Application-layer attacks.** These attacks overload a web service by generating an excessive number of application-level requests. Common examples include HTTP Flood, DNS Flood, and Slowloris [32].

Classification by Attack Vector

Another perspective for classifying DDoS attacks is based on their *attack vectors*, which describe the specific delivery technique or exploitation method used by the adversary. Under this classification, three main categories of DDoS attacks can be distinguished. In a *direct attack*, the attacker overwhelms the target system by sending a large volume of packets or service requests straight to the victim, exhausting its ability to process legitimate traffic. In contrast, *reflection attacks* rely on sending spoofed requests to intermediary servers while impersonating the victim's IP address; these servers then return their responses to the spoofed address, inadvertently flooding the victim with amplified traffic. *Amplification attacks* operate similarly to reflection attacks but intentionally exploit third-party services capable of generating responses significantly larger

than the attacker's initial request. Examples such as DNS and NTP amplification take advantage of these protocols' legitimate request–response mechanisms to produce disproportionately large replies that substantially increase the scale and impact of the attack [31].

Classification by Impact

Several works in the literature classify DDoS attacks based on the type of impact they are designed to achieve. These taxonomies typically emphasize three key objectives: disruption, degradation, and depletion. Disruption-focused attacks aim to completely halt access to a service or resource. In contrast, degradation attacks reduce service quality by slowing operations to the point where legitimate users experience significant delays. Depletion-style attacks concentrate on exhausting critical resources, including bandwidth, connection tables, or CPU cycles, which ultimately cause service failures or severe performance reduction [28], [1].

Emerging and Advanced Attack Type

Emerging DDoS attacks increasingly manifest as multi-vector campaigns that combine volumetric flooding with Layer-7 HTTP and API request barrages within a single operation. Recent threat-intelligence reports reflect this shift, highlighting a rise in the frequency, duration, and sophistication of such multi-vector attacks. In parallel, industry analyses indicate that AI-enabled automation is reshaping the threat landscape by allowing attackers to orchestrate and adapt attacks more rapidly, underscoring the need for layered defense mechanisms capable of spanning multiple layers and dynamically responding to adversarial behavior.

2.2 Centralized Deep Learning Approaches for DDoS Detection

The rapid proliferation of IoT devices, from smart home appliances to industrial control systems, has substantially expanded the attack surface for DDoS campaigns, with billions of often underprotected endpoints

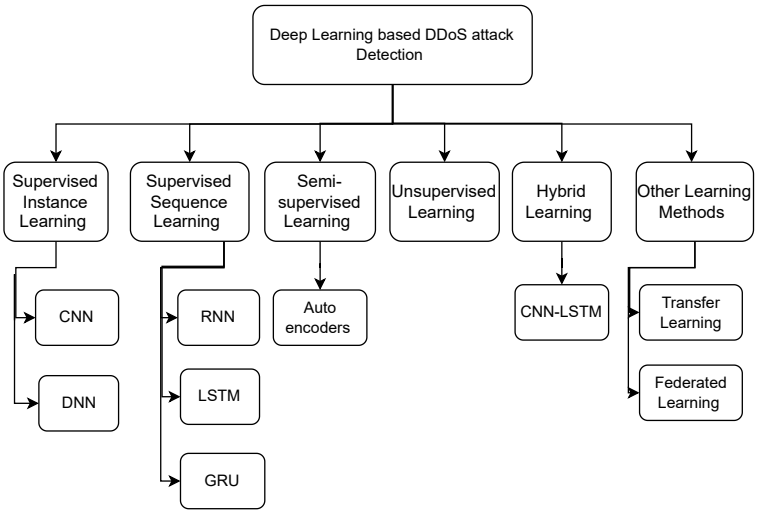


Figure 4: A comprehensive taxonomy of deep learning-based DDoS attack detection strategies

now susceptible to volumetric attacks that can cripple networks and services [10]. Traditional DDoS countermeasures are typically classified into signature-based approaches, which rely on known attack fingerprints but falter against zero-day variants; anomaly-based methods, which detect deviations from learned traffic norms via statistical or classical ML models [33, 34], and hybrid systems that seek to combine both paradigms to improve detection rates. While effective against many volumetric floods, these techniques often struggle with the scale, heterogeneity, and evolving sophistication of modern IoT-driven threats [11].

DL has emerged as a powerful alternative for DDoS detection in IoT environments, automatically extracting rich temporal and spatial features from high-dimensional traffic data and uncovering subtle or novel attack patterns without manual feature engineering [35]. However, deploying DL-based detectors at scale faces key challenges, namely, the

scarcity of labeled examples for emerging or stealthy attacks, pronounced class imbalances, and the privacy risks inherent in centralizing sensitive telemetry for model training. To overcome these obstacles, recent work has advanced various DL paradigms: transfer learning enables rapid adaptation with limited new data [36]; Few shot and adaptive learning facilitate efficient generalization to unseen attack types [37]; and federated learning preserves data confidentiality by training collaboratively across edge nodes without raw data exchange [38–40]. Moreover, hybrid architectures that combine convolutional, recurrent, and attention mechanisms have been proposed to better capture complex attack signatures [41, 42]. In general, DL-based DDoS detection methods can be classified by their architectural designs, data representations, and learning paradigms, including supervised instance learning, supervised sequence learning, semi-supervised approaches, and hybrid techniques [43], detailed in the following subsection. In particular, DL-based DDoS detection methods can be categorized based on their architectural, data representation, and learning approaches [43] into several distinct types (see Figure 4). These include *i)* supervised instance learning, *ii)* supervised sequence learning, *iii* semi-supervised learning, *iv)* hybrid learning, *v)* transfer learning, *vi)* adaptive learning, *vii)* few-shot learning, and *viii)* federated learning.

2.2.1 Supervised Instance Learning

Supervised instance-based learning with DL techniques aims to assign each data sample to a predefined class using labeled datasets. For DDoS detection, these approaches train DL models to differentiate normal network traffic from various attack patterns. The models learn to categorize incoming flows into designated classes, whether benign or representing specific DDoS attack types. Common DL architectures applied in supervised instance learning include DNNs and CNNs [1].

DNN-based IDS

Several works have demonstrated the practicality of using DNNs and CNNs for intrusion and DDoS detection. In [44], the authors introduced

DNN and LSTM models trained on the CIC-IDS2017 dataset and assessed their performance on the synthesized ANTS2019 dataset. To examine the ability to detect previously unseen attacks, both datasets were combined, and the models were retrained, resulting in improved detection of newly generated attack types. Despite these gains, the models showed limitations in capturing more complex traffic patterns and handling multi-class classification tasks, and they were not evaluated in a real-time operational environment.

In [45], a feed-forward DNN architecture was proposed for identifying DDoS attacks. This model incorporates feature-learning capabilities to automatically derive meaningful representations from raw network traffic. Additionally, attention mechanisms are integrated into the DNN, enabling the system to dynamically emphasize the most relevant features within the traffic data.

In [46], a proposed DL model for DDoS attacks in private cloud environments, specifically targeting bandwidth and connection flooding attacks. Employing algorithms such as Decision Trees (DT), K-Nearest Neighbors (KNN), Naive Bayes (NB), and DNNs, the authors compare various classifiers to identify the most effective for DDoS detection in an OpenStack-based cloud. The DNN model, chosen for its high precision and accuracy with dynamically generated datasets, outperforms other models when applied to cloud datasets. However, the study's reliance on the KDDCup99 dataset is a limitation, and there is a lack of detailed analysis of LAN and cloud datasets.

CNN-based DL Model

The work in [47] introduced a CNN ensemble model for the SDN environment, incorporating RNN, LSTM, and RL components. The CNN model architecture employs multiple convolutional layers to effectively filter and classify network traffic.

Further progress in applying CNNs to DDoS detection has resulted in models capable of adapting to the evolving landscape of cyber threats. In [48], the authors proposed a CNN-based approach and examined how variations in hyperparameters, the image format (grayscale or RGB), the

depth of convolutional layers, and kernel dimensions affect detection performance. Their evaluation was conducted using both the CIC-IDS2018 and KDDCup99 datasets. In another contribution, [49] introduced a near real-time CNN model for mitigating DDoS attacks targeting SDN controllers in ISP environments. To accelerate detection, the model was implemented to operate autonomously. The authors considered two experimental configurations: one using a custom-built dataset for both training and testing, and another using the CIC-DDoS2019 dataset for model training and evaluation.

The study in [50] introduced a multistage approach for accurately detecting DDoS attacks while keeping computational costs low. In the initial stage, a controller performs entropy-based analysis on potentially suspicious traffic directly at the router or switch. The second stage employs a CNN model to further differentiate normal traffic from malicious activity with finer granularity.

The rapid expansion of IoT deployments has increased the need for effective and accurate systems capable of detecting malicious traffic, especially DDoS attacks. In response, numerous DL-based NIDS solutions have been proposed. For instance, [51] introduced detection models that combine CNN and LSTM architectures to provide a comprehensive analysis of DDoS threats in industrial IoT environments. In [52], the authors developed a CNN-driven model for classifying botnet attacks and additionally evaluated traditional ML classifiers such as NB and KNN using the N-BaIoT and Bot-IoT datasets. Furthermore, [53] presented LUCID, a lightweight framework that leverages a CNN along with a preprocessing technique that transforms traffic flows into array-like structures and segments them into time-windowed substreams. This spatial encoding reduces computational overhead and execution time, making it well-suited for deployment on resource-constrained devices [1].

2.2.2 Supervised Sequence Learning

Supervised sequence learning focuses on training models using ordered data, allowing them to preserve information from earlier inputs. Tech-

niques such as RNNs, LSTMs, and Gated Recurrent Units (GRUs) have been extensively applied to DDoS detection because of their strong capability to process and model sequential traffic patterns [1].

Long short-term memory (LSTM)

Among these, models designed to handle sequential and time-series data have shown great promise due to their ability to handle the complex and evolving nature of network traffic. In [54] proposed LSTM models in capturing the temporal dynamics of network traffic, effectively distinguishing between normal and anomalous patterns. These studies have demonstrated better performance over traditional ML approaches in managing complex time series data.

In [55], the authors propose a framework for detecting DDoS attacks within a complex Agriculture 4.0 IoT architecture, which is structured into three layers: sensor, fog, and cloud, with IDS components deployed at each fog node. The study evaluates three DL-based IDS models that use LSTM, DNN, and CNN. The LSTM architecture includes an input layer followed by four LSTM layers. All models are trained and tested using three variants of the CIC-DDoS2019 dataset, comprising 2-class (binary), 7-class, and 13-class attack scenarios. On the binary classification task, the LSTM and CNN models outperform the DNN, while in the multi-class experiments, all three models are able to correctly identify benign traffic [1].

In [56] a solution composed of 4 levels is presented, two LSTM layers, a dropout layer, and a dense layer trained and evaluated on CIC-IDS2017, to which no feature extraction is applied. The authors propose 3 scenarios: in the first, they make a comparative analysis between the proposed model and the models that do not use all the features of the dataset, in the second, they evaluate the generation capacity of the model, and in the third, they analyze only some traffic packets.

In [57], the authors explored the use of LSTM architectures for detecting DDoS attacks, showing that LSTMs can effectively learn temporal dependencies in traffic data and often outperform traditional ML techniques in accuracy. More recent work by [58] proposed a two-stage deep

learning model, LSTM-AE, which integrates an autoencoder with an LSTM to further boost detection performance. The addition of the autoencoder improves feature extraction and representation, enabling the system to better characterize network traffic and achieve higher detection rates. Moreover, the method described in [59] presented a heuristic IDs approach using LSTM models. Their work also included fine-tuning hyperparameters and optimizing model architectures to achieve better detection accuracy and performance. In another study, [60] introduced a collaborative LSTM-based DDoS attack detection framework that utilizes adaptive thresholds and global network views to address the challenges of irregular traffic patterns and aggregating feedback from multiple detection sites to achieve higher accuracy and lower false positive rates in the context of evolving and irregular network traffic patterns.

DDoS attacks significantly impact public cloud services, and to counteract these threats, LSTM-based models have been effectively applied for detection. The work in [61], proposed an LSTM-based system, referred to as LSTM-CLOUD, designed specifically for the detection and prevention of DDoS attacks within public cloud network environments. The system employs a signature-based approach to detect attacks. It consists of two primary modules: detection and defense. The detection module uses an LSTM DL model, while the second module activates a defense mechanism to protect cloud infrastructures from identified threats.

Gated recurrent unit

GRU-based models are effective for DDoS detection because they process sequential data well and can learn long-term dependencies within traffic patterns. In practical settings, GRUs and LSTMs offer comparable performance, but GRUs typically train faster since they use fewer tensor operations than LSTMs [1].

In [62], an SDN defense system against DDoS attacks is proposed, positioned before the SDN controller. The system is composed of a GRU-based detection module, which generates an alarm by invoking the mitigation module. Like other RNNs, GRUs also possess memory and operate through the use of two gates: reset, responsible for determining which

information must be forgotten, and update, which selects the information to remember. The proposed model consists of an input layer, a GRU layer, a dropout layer, a fully connected layer, and a dense layer, with an output layer. It has been trained and evaluated on CIC-DDoS2019 and CIC-IDS2018 datasets and compared against other DL-based (DNN, CNN, LSTM) and ML-based models.

The work in [63] presented an efficient detection approach to protect against real-world new types of DDoS attacks using the GRU model for detecting DDoS attacks. The experimental outcomes demonstrate successful DDoS classification for both reflection and exploitation attacks utilizing the CIC-DDoS2019 dataset. Furthermore, [64] proposes integrating the GRU model with fuzzy logic for DDoS port scanning attack detection in SDN. The model was tested using two datasets: one with emulated traffic and the CIC-DDOS2019 dataset. The results demonstrate that combining GRU networks with fuzzy logic presents a viable option for detecting anomalies in SDN environments.

2.2.3 Semi-supervised Learning

Semi-supervised learning techniques, particularly AE, offer a powerful solution for detecting DDoS attacks by utilizing both labeled and unlabeled data to enhance detection accuracy and adaptability. In [65], the model is trained and evaluated using synthetic datasets containing DDoS traffic, generated within the Kali Linux environment, alongside the CIC-IDS2017 and NSL-KDD datasets. The model acts as a middleware system between the router and the firewall, performing analysis, feature extraction, normalization, and dimensionality reduction using PCA. The processed data is then fed into the AE model, with its output serving as input for the SVM classifier. Testing is conducted on a subset of the CIC-IDS2017 dataset, with comparisons made against pure SVM and SVM with PCA applied.

The work [66] proposes an architecture formed by a stacked sparse AE for feature learning, with 8 encode-layers and 8 decode-layers, and a DNN, with two hidden layers, for the binary classification of the traffic, all evaluated on CIC-IDS2017 and NSL-KDD datasets.

2.2.4 Hybrid Learning Approach

A hybrid deep learning approach strengthens DDoS detection by directly addressing the complexity and rapidly evolving characteristics of modern attacks. By integrating different DL architectures, including CNNs, RNNs, and autoencoders into a unified framework, researchers can take advantage of the distinct capabilities each model provides [67]. This combination supports more effective feature extraction, representation, and classification of network traffic, resulting in higher detection accuracy and improved identification of both known and previously unseen DDoS attacks.

For example, the work presented in [68] introduced a hybrid DL approach combining AE with MLP for DDoS detection and classification. Their work demonstrated the effectiveness of this combined approach in achieving accurate detection and classification results. In [69] proposed an efficient hybrid DL-based DCNNBILSTM, IDS. Their work combines CNN and Bidirectional-LSTM (BILSTM) architectures. CNNs excel in extracting features and identifying patterns and anomalies in data, whereas BILSTMs effectively comprehend sequence and temporal dependencies in data streams. shows the effectiveness of this hybrid approach for IDS tasks.

In [67], DDoSNet is proposed, a framework composed of an RNN and an AE operating in an SDN network. It is trained and evaluated on the CIC-DDoS2019 dataset. The autoencoder consists of three main components. First, the input layer takes the raw data and transforms it into a compact representation through a sequence of encoder layers. This compressed feature set has a lower dimensionality than the original input. Next, the decoder layers reconstruct the data by reversing the encoding process, producing an output that closely matches the original feature space. The reconstructed features are then fed into an RNN module, which processes the temporal patterns and forwards them to a final classification layer. Furthermore, in the work [70], a hybrid DL model was developed, and optimization techniques were employed to detect DDoS attacks more effectively. By combining CNN and LSTM models

with a unique data simplification method called Jumping Gene, adapted NSGA-II, they conducted experiments on the CIC-IDS2017 dataset in identifying DDoS attacks in IoT.

2.2.5 Unsupervised Learning

Unsupervised learning techniques have emerged as powerful tools for anomaly detection, particularly in cybersecurity, where labeled data are scarce or unavailable. Algorithms such as iForest, Local Outlier Factor (LOF), One-Class SVM, and autoencoders have shown effectiveness in learning the distribution of benign traffic and identifying deviations evaluated multiple unsupervised models, including iForest, LOF, One-Class SVM, and VAEs on industrial IoT datasets, demonstrating rapid, near-real-time anomaly detection capabilities and highlighting the value of leveraging reconstruction-based models for cybersecurity applications. Furthermore, a convolutional autoencoder is regularized adversarially to improve detection accuracy and efficiency in raw traffic flows with limited computational overhead.

In the DDoS detection domain, unsupervised learning has been applied using clustering and reconstruction models to flag abnormal patterns, clustering with autoencoders in an unsupervised pipeline, achieving high F1-scores (90%) and low false positive rates (0.5%) when tested on KDDCUP99, UNSW-NB15, and CAIDA DDoS datasets leveraged a multi-scale CNN-LSTM autoencoder, with error correction via Isolation Forest components, significantly outperforming conventional unsupervised techniques on CICDDoS2019, NSL-KDD, and UNSW-NB15 datasets. Additionally, an LSTM-Autoencoder model tailored for time series analysis of traffic features was developed, achieving over 99% accuracy on reflection-based DDoS variants (DNS, LDAP, SNMP) by relying on reconstruction error as the anomaly indicator. Unsupervised methodologies are especially valuable for detecting zero-day and evolving DDoS threats without predefined signatures. An entirely unsupervised system for identifying zero-day DDoS intrusions in IoT environments using random projections and anomaly scoring. In parallel, SINF (Sliced Iterative Nor-

malizing Flows) approaches combined with Open-Set Recognition have demonstrated exceptional generalization in detecting both known and unfamiliar DDoS attacks, achieving near-perfect detection performance on CICIDS2017 and CICDDoS2019 datasets (99.8% accuracy). These findings underscore the strong potential of unsupervised techniques in operational security systems, particularly for adaptive, signature-less DDoS detection.

2.3 Transfer Learning based Intrusion Detection

This subsection is based on our previously published work[1]. Over the past decade, progress in ML and DL has largely been driven by two factors: increasingly advanced model architectures and the availability of substantial labeled datasets. However, this data-centric success introduces a practical limitation that acquiring and annotating enough high-quality, domain-specific samples is often costly, slow, and, in several security-critical scenarios, not feasible at all. TL provides a remedy to this challenge by leveraging knowledge obtained from a source task to accelerate learning on a related target task where data is limited [71, 72].

Depending on the strategy employed, TL may focus on learning representations that align the distributions of source and target domains (feature-based techniques), reusing pretrained model parameters that are subsequently fine-tuned to a new setting (parameter-based methods), or adjusting source instances through weighting or selection to better reflect the target distribution (instance-based approaches).

Within cybersecurity and DDoS detection in particular, the potential of TL is especially significant. Malicious traffic patterns evolve quickly as adversaries introduce new amplification methods and evasion strategies, and each operational environment (such as ISP backbones, enterprise networks, or IoT edge deployments) carries its own distinct noise characteristics. By transferring models trained in one context to another, practitioners

[1]This section incorporates adapted material from: M.B. Anley, A. Genovese, and V. Piuri, "TransferEdge: Transfer Learning Approach to Detect Evolving DDoS Threats in Edge-IIoT," in *Proceedings of the 9th International Forum on Research and Technologies for Society and Industry (RTSI 2025)*, Gammarth, Tunisia, August 24–26, 2025, pp. 11–16. Reused material has been adapted for inclusion in this thesis.

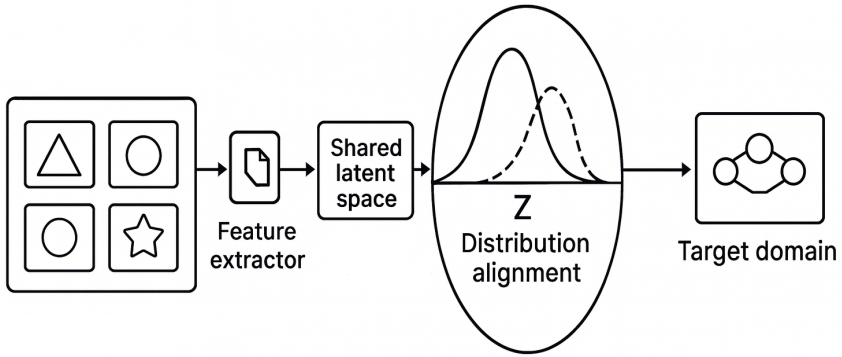


Figure 5: Feature-based transfer learning.

can bypass the heavy burden of collecting and labeling extensive attack datasets. Recent surveys highlight that TL not only enhances detection performance and reduces training overhead but also supports privacy-preserving workflows and improves resilience to concept drift [73] , [3].

2.3.1 Transfer Learning Paradigms

TL relaxes the i.i.d. assumption by transferring knowledge from a source to a target domain/task, and is commonly categorized into feature-based, parameter-based, and instance-based paradigms [71,74]. These paradigms are particularly relevant to network security, where labeled data are scarce, domains shift over time, and deployment conditions vary across networks and organizations. Comprehensive surveys chart the evolution from classical TL to deep transfer, highlighting robustness and domain shift as core challenges that TL seeks to mitigate.

Feature-Based Transfer

Learn a shared latent space where source and target distributions are aligned (via MMD/adversarial alignment), then train a classifier on domain-invariant features. This reduces domain shift while keeping task heads simple (see Figure 5). Feature-based TL aligns source–target

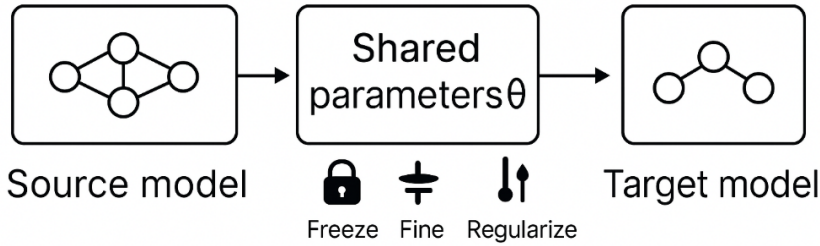


Figure 6: Parameter-based transfer learning.

representations to obtain domain-invariant embeddings before (or during) model training. Classical alignment minimizes statistical discrepancies between domains: CORAL aligns second-order statistics (covariances) either as a preprocessing transform or as a differentiable loss (Deep CORAL) [75,76]. Adversarial representation learning enforces domain confusion via a gradient reversal layer, as in Domain-Adversarial Neural Networks (DANN), yielding features that are discriminative for the task yet indiscriminative for the domain [77]. These methods are the backbone of modern unsupervised domain adaptation pipelines, later specialized to security/IDS settings.

Parameter-Based Transfer

Start from a pre-trained source model and transfer weights to the target, freezing or regularizing some layers and fine-tuning others. The prior parameters act as a strong inductive bias, speeding convergence and improving generalization with limited target data (see Figure 7). The approach includes freezing and sharing subsets of layers, selective finetuning with regularization toward source parameters (i.e L_2-style penalties), and multi-task schemes that share backbones while specializing heads. Surveys of deep TL characterize these strategies and discuss their stability–plasticity trade-offs, regularization, and optimization details when domains differ substantially [71,74]. In security contexts, parameter-based TL is attractive due to rapid adaptation and reduced labeling cost, but can overfit domain-specific artifacts without additional alignment or drift

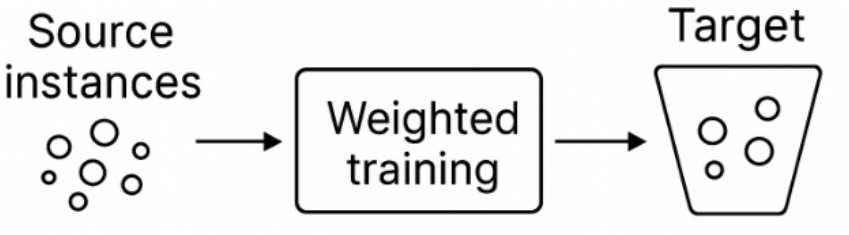


Figure 7: Instance-based transfer learning.

handling.

Instance-Based Transfer

Instance-based TL reweights or selects source samples to better match the target distribution under covariate shift. Reweight or select source examples according to their relevance to the target, then train using the weighted data:

$$w(x) = \frac{p_t(x)}{p_s(x)} \tag{2.1}$$

This directly corrects covariate shift without changing model architecture or features. Kernel Mean Matching (KMM) estimates importance weights by matching source and target means in an RKHS, correcting sample selection bias [76]. Importance-Weighted Cross Validation (IWCV) provides (nearly) unbiased risk estimates for model selection under shift [78]. TrAdaBoost reduces misaligned source instances while increasing target-consistent ones throughout boost rounds [79]. These techniques remain influential baselines and components for modern pipelines facing traffic or domain shifts.

2.3.2 Transfer Learning for Evolving Attacks

Evolving (non-stationary) attack landscapes induce concept and feature drift in traffic, degrading static models. Recent surveys on drift-aware IDS emphasize continual adaptation, domain shift handling, and robust

evaluation protocols [80, 81] [2] In network intrusion detection, domain-invariant representation learning has been applied to generalize across networks and time (DL-NIDS with adversarial domain adaptation) [82]. Continual/semi-supervised approaches such as INSOMNIA update models as traffic characteristics drift, improving long-horizon robustness [83]. Emerging security-focused TL surveys also highlight open issues: scarce labels, label noise, and poisoning risks during adaptation [84]. Overall, combining TL with drift detection and continual learning is key to sustaining DDoS or IDS performance under evolving threats.

A variety of studies have investigated the use of transfer learning and deep learning for anomaly-based DDoS detection. For example, [85] tackles the issue of limited training data in intrusion detection by adopting a CNN-driven TL methodology, allowing models to benefit from knowledge gained on large labeled datasets and thereby improving accuracy in settings with constrained resources. Extending the idea to critical infrastructure protection, [86] proposes a TL-enabled IDS tailored for Communication-Based Train Control (CBTC) systems. Their approach combines one-dimensional CNNs with LSTM layers to jointly learn spatial patterns and temporal behavior associated with cyberattacks [2].

In response to the scarcity of labeled data, a growing body of work has begun leveraging pre-trained models as effective tools for cybersecurity applications. For example, [87] utilized pre-trained transformer architectures to mitigate distribution shifts and address data limitations in communication networks. Likewise, [88] proposed a dual-autoencoder deep transfer learning framework that facilitates knowledge transfer from labeled to unlabeled IoT datasets. Their method improves the detection of previously unseen attacks, illustrating the strength of transfer learning for emerging threat scenarios.

Transfer learning has also shown promise in next-generation network environments, such as 5G, especially for mitigating DDoS attacks. The

[2]The following subsection is based on the published work: M. B. Anley, A. Genovese, and V. Piuri, "TransferEdge: Transfer learning approach to detect evolving DDoS threats in Edge-IIoT," in *Proceedings of the 9th International Forum on Research and Technologies for Society and Industry (RTSI 2025)*, Gammarth, Tunisia, August 24–26, 2025. The material has been adapted and expanded for inclusion in this thesis.

studies in [89] explored fine-tuning pre-trained CNN and BiLSTM models on limited real-world 5G traffic samples, achieving strong results in identifying complex DDoS behaviors. These findings highlight transfer learning's ability to generalize across diverse and dynamic network settings, reinforcing its importance for zero-day intrusion detection in highly heterogeneous systems [2].

The combination of ensemble techniques with transfer learning has also contributed to improved detection reliability. In [90], the authors presented an IDS that merges several CNN architectures, VGG16, Inception, and Xception, and enhances them through hyperparameter tuning and ensemble strategies, achieving superior performance compared to standalone deep models. Cross-network transfer learning, examined in [91], further demonstrated that aligning knowledge from labeled source networks with unlabeled target networks can yield strong intrusion detection results, making it particularly applicable to diverse industrial settings. Additionally, [3] introduced a resilient DDoS detection approach that employs adaptive transfer learning to mitigate data scarcity and heterogeneity challenges by transforming both tabular and image-based traffic data into DL ready representations, supported by systematic hyperparameter optimization and fine-tuning [2].

Prior research offers a strong basis for tackling key challenges in DDoS detection, especially when dealing with evolving attack behaviors in data-limited industrial IoT settings. Building on these findings, this paper introduces a new transfer learning approach designed to identify emerging DDoS threats within Edge-based IIoT environments [2].

2.3.3 Edge-Aware TL for DDoS Detection

Edge-aware Transfer Learning (EATL) adapts transfer strategies to the realities of IoT and Mobile Edge Computing (MEC), including limited CPU, memory, and energy budgets, strict latency targets, and persistent domain shifts across networks (such as home gateways, enterprise subnets, and micro–datacenters). Recent IDS and DDoS studies show that deep models trained on public datasets often degrade when deployed at the edge, mo-

tivating pipelines that keep inference close to data sources while enabling rapid, privacy-preserving adaptation with minimal backhaul [92], [93].

Methods. Parameter-based TL (warm-start from a pre-trained NIDS, then fine-tune a small head or the last block) paired with compression (quantization and pruning) meets device budgets and improves convergence with few local labels. Feature-based and domain adaptation TL learn domain-invariant traffic representations, allowing models trained in one network to generalize to others. Adversarial and transformer-based approaches have reported strong cross-domain NIDS. When source-target covariates differ substantially, instance-based TL reweights source flows by importance to reduce negative transfer during fine-tuning. Together, these edge-conscious choices lower compute or communication costs while improving robustness [84], [94].

Edge + privacy. Many deployments combine TL with FL, so sites adapt locally and share only updates (*federated transfer learning*). Empirical results on IoT and 5G IDS show that local fine-tuning plus periodic aggregation (like FedAvg and FedProx) improves accuracy without exporting raw traffic, while allowing each site to tune model size to device limits. For evaluation, recommended metrics include macro-F1 and minority-class recall under cross-site splits (train on A, test on B or C), latency and energy per flow, and adaptation speed, all aligned with edge constraints [95].

Edge deployments introduce tight constraints on compute, memory, energy, and communication, motivating federated and on-device transfer. FTL blends TL with FL to share knowledge without centralizing data, but faces new challenges (statistical heterogeneity, communication budgets, privacy) [96]. Surveys of FL for edge computing review communication-efficient training, client heterogeneity, and system co-design issues that directly impact how transfer is carried out at the edge. To meet resource limits during adaptation and finetuning, model compression (pruning, quantization) and knowledge distillation are widely recommended for on-device and TinyML settings [85, 94]. Designing edge-aware TL thus requires jointly optimizing transfer strategy (feature and parameter or instance), training protocol (centralized, FL, or FTL), and efficiency techniques to maintain accuracy–latency–energy trade-offs in realistic IoT

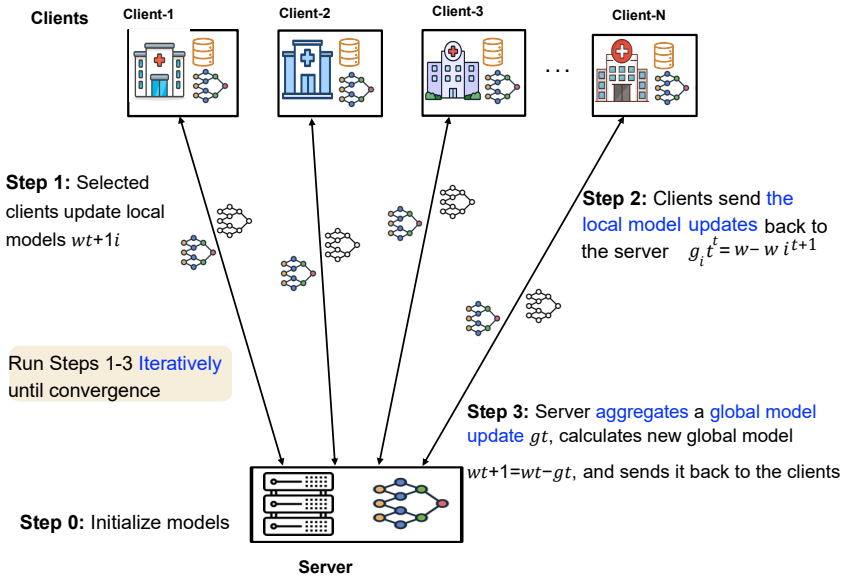


Figure 8: Federated learning architecture.

deployments.

2.4 Federated Learning for IoT Security

2.4.1 Federated Learning Fundamentals

The following subsection is based on the work presented in our published article [4][3]. The material has been adapted and integrated into the broader methodological framework of this thesis.

[3]This subsection is based on the published article: M.B. Anley, P. Coscia, A. Genovese, and V. Piuri, "FELACS: Federated Learning with Adaptive Client Selection for IoT DDoS Attack Detection," *Computers & Security*, 2025, Article 104642. Licensed under CC BY. See: `https://creativecommons.org/licenses/by/4.0/`. Reused material complies with all applicable license terms.

FL in IoT Security and Intrusion Detection Scenarios In contrast to centralized ML-based DDoS detection approaches [3, 42], which face issues such as privacy leakage, increased latency, and possible network congestion [97], FL offers a decentralized alternative. In an FL-based IDS, model training is distributed across IoT devices, significantly reducing data transmission requirements and mitigating privacy risks [98–100]. FL-driven IDSs can match the accuracy of centralized systems while preserving data privacy [101] and accommodating the inherently distributed nature of IoT networks, where devices often differ in computational capacity and data availability [102].

Several studies have specifically examined FL settings with non-IID data, a common challenge in IoT environments where devices may collect abundant samples of certain attack types but very limited data for others [103, 104]. In FL, the statistical relationship between client datasets may follow an *i.i.d.* regime where clients' local samples are drawn from the same underlying distribution or a *non-i.i.d.* regime where client data distributions differ due to variations in devices, environments, and user behaviors. In practice, non-i.i.d. data are common and introduce substantial heterogeneity across clients. Due to this non-i.i.d. behavior, standard aggregation methods such as FedAvg, which weigh client updates uniformly [105], frequently struggle to cope with the data heterogeneity encountered in real deployments; consequently, heterogeneity can slow global model convergence and degrade overall performance [4].

2.4.2 Client Selection in FL

This subsection builds on the formulation presented in [4]. In FL, the training process is distributed across multiple clients, which collaboratively learn a global model by sharing locally computed updates rather than raw data. Efficient client selection mechanisms ensure an effective model training procedure while optimizing the incurred communication costs and preserving the diversity of the given data [105, 106]. Studies on the selection of clients in FL tasks have explored various strategies that consider client availability, data quality, computational capacity, and

model performance contributions [107, 108].

Beyond random client sampling [105, 108–110], a straightforward strategy for client selection is to choose participants based on how much they are expected to decrease the global loss. Methods such as those in [107, 111] favor clients whose local updates are predicted to yield greater improvements in model accuracy, often resulting in faster convergence than random selection, especially under non-IID data distributions where certain clients hold more informative or diverse samples. However, these schemes do not account for the computational limitations of individual devices or the communication overhead involved, which can introduce delays and reduce efficiency in IoT environments with constrained resources. More recent work has sought to refine client selection and model performance by incorporating device resources and capability constraints [112]. FedProx, introduced to better accommodate environments with heterogeneous computational power by allowing clients to perform only part of the local training. This reduces instability in updates that arises when datasets differ across devices [113]. Nonetheless, FedProx does not explicitly consider the value or diversity of client data when determining which devices should participate [4].

Further developments in resource-aware selection include methods such as FedCS [108] and FedMCCS [114]. These strategies evaluate clients based on their availability, processing capability, and communication bandwidth; dynamically regulate the number of participants, and allocate additional workload to more capable devices. However, both approaches rely on clients truthfully revealing their resource conditions—an assumption that may not hold in practice due to privacy constraints or the possibility of inaccurate self-reporting in real-world IoT deployments [4].

Recent work has also examined client selection strategies that leverage data diversity to improve the generalization ability of FL models. In the approach of [104], data heterogeneity is mitigated through a limited data-sharing mechanism in which a small, globally accessible dataset is used to better align client objectives. Although this technique accelerates convergence, it compromises a core FL principle, privacy, since sharing even a small portion of raw data may reveal sensitive information. To avoid this

drawback, the method introduced in [38] promotes diversity-aware client participation, selecting clients with heterogeneous data distributions to strengthen model robustness without exchanging data. However, its effectiveness depends on reliably estimating data diversity, which can be challenging in practice, particularly within IoT environments where data characteristics shift rapidly over time [4].

To incorporate both data-related properties and the resource capabilities of participating devices, several multicriteria client selection strategies have been introduced to strengthen the reliability and security of FL systems. For instance, [115] presents SecureFedPROM, a framework that combines security, performance, and system-efficiency indicators within a weighted multi-armed bandit (MAB) formulation to prioritize clients dynamically. In addition, the approach employs an MAB-driven scheduler that continuously learns each client's computational and communication behavior, enabling the system to reduce per-round training delays [4].

2.4.3 Poisoning Attacks in FL

The exponential growth of IoT ecosystems has significantly transformed the digital landscape, creating interconnected devices that facilitate seamless data exchange and automation. These devices generate a large amount of data, essential for improving services and enabling predictive capabilities. However, the sensitivity and distributed nature of the data have also made IoT environments attractive targets for sophisticated cyberattacks. Ensuring the security of these decentralized networks is paramount, particularly against malicious activities that threaten their reliability and functionality. Thus, FL has emerged as a promising paradigm for securing IoT ecosystems, enabling decentralized model training across IoT devices while ensuring data privacy and reducing the communication burden by avoiding the transfer of raw data [105] [5].

With its decentralized nature and reliance on clients, FL introduces

The following subsections are based on the work presented in the FLIFRA framework: M.B. Anley, A. Genovese, and V. Piuri, "FLIFRA: A Dual-Layer Poisoning Detection Framework for Federated Learning in IoT," presented at the IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2025. Adapted for inclusion in this thesis.

critical vulnerabilities for poisoning attacks. These attacks can take various forms, each aiming to compromise the performance of the global model. These attacks are particularly dangerous in IoT settings, where diverse, non-IID data distributions exacerbate their impact. For instance, backdoor attacks embed triggers in input data to induce targeted misclassifications, allowing adversaries to manipulate predictions under specific conditions [116], thus inducing IoT-based intrusion detection systems to misclassify malicious traffic as benign. Moreover, label-flipping attacks can corrupt the training process by altering class labels, confusing the model, and degrading overall accuracy [117]. Another attack example is represented by gradient manipulation, which directly alters the shared gradients during training and corrupts the global model by injecting subtle but harmful updates during aggregation [118].

2.4.4 Poisoning Attacks Detection in FL

As illustrated in Figure 9, the architecture of FL in IoT-Edge environments spans multiple domains of IoT applications, including smart transport, healthcare, IIoT, and smart cities. In this FL setting, IoT-Edge can include smart sensors, wearable devices, and industrial machinery that collect and process local data to train ML models. Each device operates independently within its domain, maintaining data privacy by keeping raw data locally and only sharing model update gradients or parameters with a central server. The server collects the model updates produced by each client device and combines them to generate an improved global model. This updated model is then sent back to the participating IoT-Edge nodes for the next round of local training. While this distributed learning loop enables scalability, it also introduces vulnerabilities, making the system susceptible to data-poisoning adversaries [119].

In data poisoning attacks, one or more IoT-Edge devices are compromised, allowing an adversary to inject poisoned data into local training. For instance, in a smart home scenario, a compromised device trains its local model on malicious data and sends tainted updates to the central server. When these updates are aggregated with benign ones, they

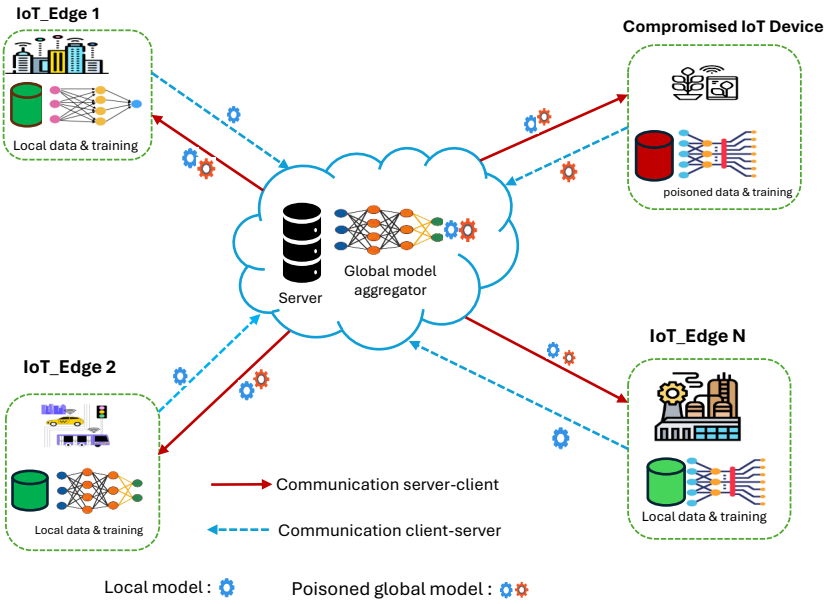


Figure 9: Architecture of an FL-based data poisoning attack in IoT-Edge environments across domains.

degrade the global model's performance and introduce bias. This corrupted global model is then redistributed to all devices, propagating the attack [120].

2.4.5 Threat Models in FL

Data poisoning threats typically arise from adversarial clients who manipulate their local datasets or training processes to corrupt the global model. Threat models vary by scope and intent: label-flipping attacks inject mislabeled samples to bias decision boundaries; backdoor attacks embed hidden triggers so the global model behaves maliciously under specific inputs while appearing benign otherwise; feature-space poisoning perturbs input distributions to degrade accuracy or fairness; and multi-round consistent poisoning gradually introduces correlated malicious updates

across rounds, evading simple anomaly detection. Attackers may operate under white-box assumptions (full knowledge of model and aggregation) or black-box assumptions (limited visibility, but exploiting aggregation weaknesses), and may be targeted (forcing misclassifications of specific classes) or untargeted (degrading overall model performance). These threat models highlight the unique vulnerabilities of federated learning, where decentralized and non-IID client data amplify the difficulty of detecting poisoned contributions.

2.4.6 Defense Mechanisms

FL is increasingly vulnerable to data poisoning attacks, where malicious clients inject corrupted data to degrade the global model's performance, posing significant threats to model integrity, particularly in IoT ecosystems. The resource constraints and dynamic conditions inherent to IoT environments exacerbate the challenges of detecting and mitigating such attacks [121]. Existing detection strategies can be broadly categorized into client-side and server-side approaches [122]. Client-side techniques include iForests [123], which focus on identifying outliers by analyzing local training data and algorithms. However, these methods struggle to detect coordinated attacks that mimic benign behavior, limiting their effectiveness in real-world scenarios. To address this, the approach described in [124] proposes FL-WBC, which identifies the attack parameter space and perturbs it during local training to mitigate the long-lasting effects of attacks. Although promising, such client-side methods often lack scalability and adaptability to dynamic environments.

On the server side, various techniques were proposed to reduce the influence of malicious updates by filtering or aggregating model updates from clients. Provenance-based methods [121], for instance, trace the origin of model updates to identify malicious contributions, but rely heavily on accurate provenance data, which is often impractical in resource-constrained IoT environments. Other server-side approaches, such as client filtering [125] and clustering-based methods [126], aggregate updates only from trusted clients. Robust aggregation methods, including

Krum [127], Trimmed Mean, and median aggregation [128], have been widely adopted to mitigate poisoning attacks by reducing the influence of outliers. These methods average the least likely tainted updates, providing resilience against a small number of Byzantine (adversarial) clients. However, their effectiveness reduces as the number of compromised clients increases.

Moreover, Fed-LSA [129] employs autoencoders to detect poisoned updates in the latent space, offering a novel approach to identifying malicious contributions. However, its performance degrades in non-IID data scenarios, limiting its applicability. Similarly, FreqFed [130] leverages frequency-domain analysis to detect malicious updates but assumes stationary data distributions, which restricts its effectiveness in dynamic IoT environments. Despite significant progress, challenges persist in detecting and mitigating data poisoning attacks in FL. Existing methods often focus exclusively on either client-side detection or server-side detection, resulting in incomplete defenses. Client-side approaches may fail to detect sophisticated attacks [124], while server-side methods lack granular insights into local data distributions [119].

2.5 Available DDoS Benchmark Datasets

In IDs and DDoS attack detection, Table 2 provides a summary of commonly used datasets, including the number of features and types of attack classes for DDoS attack detection. Details regarding these datasets are provided below.

KDDCup99 dataset [131]. The KDDCup 1999 dataset was developed for a competition focused on identifying intrusions and malicious behavior in networked systems. It contains around five million connection records, including both normal traffic and 22 distinct attack types, which are grouped into four primary classes: Denial of Service (DoS), Remote-to-Local (R2L), User-to-Root (U2R), and Probe [132]. Each entry is represented by 42 features organized into three categories that basic features, traffic-related

features, and content-based features [1].

NSL-KDD dataset [132]. The ISCX NSL-KDD dataset, released in 2009 by the Canadian Institute for Cybersecurity, was introduced as an enhanced successor to KDDCup99. It eliminates redundant entries and provides a more balanced distribution of attack categories. The dataset retains a structured set of 42 features and offers a manageable number of samples by removing a large portion of duplicates, 78.05% from the training set and 75.15% from the test set, which helps reduce bias and improves the reliability of intrusion detection research. In total, it contains roughly 150,000 labeled instances, comprising 125,973 training samples and 22,544 testing samples [133]. Today, it remains a widely used benchmark for assessing intrusion detection approaches and supporting fair comparisons across methods [1].

UNSW-NB15 dataset [134]. These datasets include analysis, fuzzers, backdoors, DoS, exploits, reconnaissance, generic, shellcode, and worms - 9 unique attack types and 49 characteristics. These attack categories encompass various cyber threats, facilitating a comprehensive evaluation of the IDS. It contains 2,540,044 records, with 2,218,761 benign records and 321,283 malicious records.

CIC-IDS2018 dataset [135]. The dataset was generated by the Communications Security Establishment (CSE) together with the Canadian Institute for Cybersecurity (CIC) using a controlled laboratory setup that recorded both network traffic and system logs. It includes normal activity as well as seven distinct categories of cyberattacks, carried out by 50 attacker machines targeting an emulated organization consisting of 420 hosts and 50 servers distributed across five departments [135]. A total of 80 features describe each flow, covering benign traffic and several well-known DDoS attack types, including GoldenEye, Slowloris, Hulk, Slow-HTTPTest, LOIC-HTTP, HOIC, and LOIC-UDP. Attack traffic constitutes approximately 82.7% of all records, whereas benign flows represent the remaining 17.3%. For this chapter, we utilize the CICIDS2018 dataset[4] , chosen for its realistic representation of IoT and non-IoT network be-

[4] https://www.unb.ca/cic/datasets/ids-2018.html

haviors and its comprehensive coverage of both normal and malicious scenarios [1].

CIC-DDoS2019 dataset [136]. This dataset provides fine-grained information on multiple DDoS attack vectors, such as UDP floods, TCP SYN floods, and HTTP floods, enabling detailed analysis of their individual characteristics [136]. It comprises more than 80 flow-based features and covers a broad spectrum of DDoS scenarios, including both reflection-based attacks (e.g., MSSQL, NTP, SSDP) and exploitation-based attacks (e.g., UDP and SYN floods). Data collection was conducted over two separate days: the training set was recorded on January 12, 2019, and includes 12 attack types, whereas the test set was captured on March 11, 2019, and contains 7 attack types. The dataset is highly imbalanced, with benign traffic representing only 0.16% of the samples and attack traffic 99.84% [42].

CIC-IoT2023 [137]. The CIC-IoT2023 dataset, developed by the Canadian Institute for Cybersecurity, offers an extensive resource for advancing IoT security research. It replicates practical deployment conditions using 105 IoT devices and includes 33 attack types grouped into seven major categories: DDoS, DoS, Reconnaissance, Web-based attacks, Brute Force, Spoofing, and Mirai. By incorporating a diverse set of malicious behaviors and addressing gaps found in earlier datasets, CIC-IoT2023 supports the creation and evaluation of more effective security analytics for IoT environments [1].

BoT-IoT [138], The dataset, created by the Cyber Range Lab at the University of New South Wales (UNSW) Canberra, contains a mixture of benign traffic and malicious botnet activity generated within a realistic network environment. Its extensive collection of 69.3 GB of PCAP files and more than 72 million flow records captures a wide spectrum of cyberattacks, including DDoS, DoS, operating system and service scanning, keylogging, and data exfiltration [139]. The BoT-IoT dataset is openly accessible[5] and is widely regarded as one of the most representative resources for IoT network behavior, offering detailed coverage of both normal opera-

[5] `https://research.unsw.edu.au/projects/bot-iot-dataset`

Table 2: Summary of DDoS attack detection cybersecurity datasets that are available in public.

Dataset	Year	Format	Cardinality #	Features #	Attack Type
KDDCup99	1998	Packet	5.2m	42	DoS, Probe, R2L, U2R
NSL-KDD	2009	Packet	0.15m	42	DoS, Probe, R2L, U2R
CAIDA-DDoS 2007	2007	Packet	56m	14	DDoS
ISCX2012	2012	Packet	4.8m	41	Infiltration, DDoS, HTTP DoS, Brute Force SSH
UNSW-NB15	2015	Packet	2.5m	49	Fuzzers, Reconnaissance, Exploits, Backdoors, Generic, Shellcode, DoS Attacks, Worms
CIC-IDS2017	2017	Packet	2.8m	80	DDoS, DoS, Botnet, Heartbleed, Brute Force SSH, Web Attack, Infiltration, Brute Force FTP
CIC-IDS2018	2018	Packet	16m	80	Heartbleed, Botnet, Bruteforce, DoS, Web Attacks, DDoS, Infiltration
CIC-DDoS2019	2019	Packet	72m	80	DDoS, MSSQL, NTP, SSDP, NETBIOS, PortMap
BoT-IoT	2018	Packet	72m	115	DDoS, DoS, OS and Service Scan, Keylogging, Data Exfiltration
ToN-IoT	2019	Packet	22.3m	42	DoS, DDoS, Ransomware
CIC-IoT2023	2023	Packet	46.6m	46	DDoS, DoS, Reconnaissance, Web-based, Brute Force, Spoofing, Mirai

tions and adversarial activities that support cutting-edge research in IoT cybersecurity [1].

TON_IoT dataset [140]. The TON_IoT datasets, developed by the School of Engineering and Information Technology at UNSW Canberra, serve as a comprehensive resource for assessing AI-based cybersecurity solutions in Industry 4.0 and IoT/IIoT contexts. They encompass multiple data modalities, including telemetry from industrial IoT sensors, system logs from both Windows and Ubuntu machines, and captured network traffic. All data is collected from an advanced testbed designed to emulate real-world IoT ecosystems. The datasets reflect numerous cyberattack scenarios, including DoS, DDoS, and ransomware targeting different system layers and are provided in raw, processed, and train/test, ready formats, accompanied by detailed feature documentation [1].

2.6 Summary

This chapter begins with a comprehensive taxonomy of DDoS attacks, outlining their fundamental characteristics, tracing their historical evolution, and examining both attacker motivations and typical targets. A structured classification is provided, covering major attack vectors including volumetric, protocol, and application layer floods and highlighting the increasing sophistication and adaptability of these threats in modern networked environments.

The discussion then transitions to centralized DL approaches for DDoS detection, reviewing supervised instance and sequence-based methods as well as semi-supervised, unsupervised, and hybrid techniques. The strengths and limitations of each approach are critically evaluated, with particular emphasis on their applicability to dynamic and large-scale network settings. The chapter also examines transfer learning paradigms, emphasizing their potential to detect evolving attack patterns, address cross-domain scenarios, and adapt to edge-aware constraints in resource-constrained IoT environments. Finally, the chapter explores the role of FL in IoT security, highlighting its fundamental principles and advantages for privacy-preserving distributed detection. It analyzes the challenges posed by poisoning attacks in FL, presenting common adversary models and reviewing existing defense mechanisms. These include anomaly detection strategies, robust aggregation methods, and hybrid defenses designed to preserve model integrity against both targeted and untargeted poisoning attempts. The discussion concludes with a survey of publicly available cybersecurity datasets that enable empirical evaluation and benchmarking of these approaches.

Chapter 3

Deeply Adaptive Centralized Baseline for Learning-Based DDoS Detection

3.1 Introduction

The rapid proliferation of the IoT has dramatically increased connectivity, embedding billions of devices into everyday life. From smart homes to industrial systems, these devices deliver efficiency and convenience, but their ubiquity also enlarges the attack surface. Among the most significant risks are DDoS attacks, which overwhelm targets with excessive and malicious traffic. IoT devices often deployed with minimal hardening and irregular update hygiene are attractive for botnet recruitment and abuse. Recent reports by the ENISA document the growing sophistication and scale of DDoS threats, including the role of dark-web services, ransom-driven campaigns, and increasingly complex, multi-vector attack strategies [10]. In turn, modern DDoS campaigns leverage botnets, amplification, and cross-vector orchestration to stress even well-provisioned infrastructures, leaving classical signature or threshold-based

IDS ill-equipped to detect novel or stealthy volumetric attacks [141].

DDoS defenses are commonly categorized into signature-based, anomaly-based, and hybrid approaches [33]. Signature-based methods match traffic to known attack patterns and thus struggle with zero-day variants. Anomaly-based systems flag deviations from learned baselines using statistical techniques and ML, enabling the discovery of previously unseen attacks but risking false alarms if baselines drift [34]. While traditional ML has shown utility for DDoS detection, its effectiveness is often constrained by the volume, heterogeneity, and nonstationarity of network traffic. DL offers a way to learn discriminative representations directly from raw or lightly processed flows, improving generalization to emergent attack behaviors [35]. Prior work has explored DNNs, CNNs, and RNNs/LSTMs for modeling spatial and temporal traffic patterns, reporting promising results on benchmark datasets and unseen attack signatures [48, 142].

A persistent challenge, however, is the sensitivity of DL performance to hyperparameter choices. Manual tuning and exhaustive grid searches are computationally expensive and may still overlook high-performing regions of the search space, especially on large and heterogeneous datasets. Moreover, although DNNs, CNNs, RNNs, and LSTMs can excel on individual benchmarks, their cross-dataset generalizability, given divergent feature spaces, attack mixes, and class imbalances, remains underexplored, raising questions about robustness and real-world applicability.

To address these gaps, this thesis presents an automated HPO framework for DL-based DDoS detection. The proposed pipeline integrates PCA-based dimensionality reduction to compress high-dimensional flow, thereby accelerating convergence with hyperband [143] via Keras Tuner and efficiently exploring architectures and training configurations. On this compact feature space, the thesis systematically designs and tunes three model families: fully connected DNNs, 1D-CNNs (for spatial patterns), RNNs, and LSTMs (for sequential dependencies), and evaluates them on four real-world benchmark datasets: CIC-IDS2018, CIC-DDoS2019, UNSW-NB15, and KDDCup99. Empirically, the proposed approach improves detection accuracy, reduces false-alarm rates, and significantly lowers the computational cost of model selection.

Contributions (i) Introduces a unified DL framework that integrates PCA-based feature compression with Hyperband-driven hyperparameter optimization (via Keras Tuner) to systematically tune DNN, CNN, RNN, and LSTM architectures for DDoS detection. (ii) Provides a comprehensive, dataset-agnostic evaluation on CIC-IDS2018, CIC-DDoS2019, UNSW-NB15, and KDDCup99, benchmarking the optimized deep models against strong classical ML baselines under a shared preprocessing and metric protocol (accuracy, macro-F1, AUROC, precision, recall). (iii) Validate practical applicability by achieving state-of-the-art detection performance while substantially reducing HPO compute overhead, supporting near-real-time deployment in resource-constrained IoT environments. By bridging DL with efficient hyperparameter search, our work advances IoT security and lays a foundation for deployable, cross-domain DDoS detection systems.

3.2 Methodology

3.2.1 Threat Model and Problem Statement

This chapter adopts a unified threat model and defender profile to guide the security analysis across the proposed detection approaches. Specifically, it considers an external adversary controlling a heterogeneous botnet that mounts multi-layer DDoS attacks, and a defender deployed at cloud/edge gateways that detects malicious activity from flow/window-level telemetry. The empirical validation relies on real benchmark datasets obtained from external sources that are (CIC-IDS2018, CIC-DDoS2019, UNSW-NB15, KDDCup99, and BoT-IoT), rather than simulated environments, to ensure realism and comparability across studies.

Threat model. The adversary operates externally and controls a botnet composed of heterogeneous IoT and cloud hosts. It can launch multiple categories of DDoS attacks, including volumetric L3/L4 floods (e.g., UDP amplification and TCP SYN floods) and application-layer request floods (e.g., HTTP GET/POST floods). These attacks manifest in ag-

gregated flow/window telemetry as abnormal packet and byte rates, increased connection-attempt rates, TCP flag imbalances (e.g., SYN/ACK asymmetry), and broader destination fan-out within fixed observation windows. The analysis assumes that the adversary does not compromise the monitoring infrastructure at the gateway; instead, it attempts to induce detectable deviations in traffic behavior through coordinated attack traffic.

Defender. This work proposes a DL-based detection mechanism deployed centrally at cloud or edge gateways, where NetFlow/IPFIX-like records are exported. Each dataset provides flow-level records that include attributes such as flow duration, packet and byte counts, TCP flags, exporter timestamps, and other traffic features, with variations across datasets. Let $\mathbf{x}_i \in \mathbb{R}^D$ denote the standardized feature vector of the i-th flow (or window), with label $y_i \in \{0, 1\}$, where 1 denotes DDoS and 0 denotes benign traffic. To reduce dimensionality and accelerate training, PCA projects $\mathbf{x}_i$ into $\mathbf{z}_i = P\,\mathbf{x}_i \in \mathbb{R}^d$, with $d \ll D$. Using labeled data $\mathcal{D} = \{(\mathbf{z}_i, y_i)\}_{i=1}^N$ from widely used benchmarks (CIC-IDS2018, CIC-DDoS2019, UNSW-NB15, and KDDCup99), the model trains a centralized detector that outputs a binary decision per flow (time window). For comparability across datasets, the analysis maps dataset-specific attack categories to a unified binary taxonomy {benign, DDoS}, while also reporting multiclass results when distinguishing specific DDoS types is relevant.

Model Selection and Hyperparameter Optimization. This work employs HPO to search over three deep model families tailored for flow-based traffic data: fully connected DNNs, 1D-CNNs (to capture spatial patterns), RNNs, and LSTMs (to capture temporal dependencies). Each model $f_{\theta,h}$ is characterized by trainable parameters θ and hyperparameters $h \in \mathcal{H}$, where the search space includes depth, width, kernel size, learning rate, batch size, and other model-specific factors (see Table 3). Conduct the search using Hyperband, implemented through Keras Tuner, which adaptively allocates training resources among candidate configurations throughout the experiments, assuming trustworthy exporters, synchronized clocks, and routable and stable label quality in the bench-

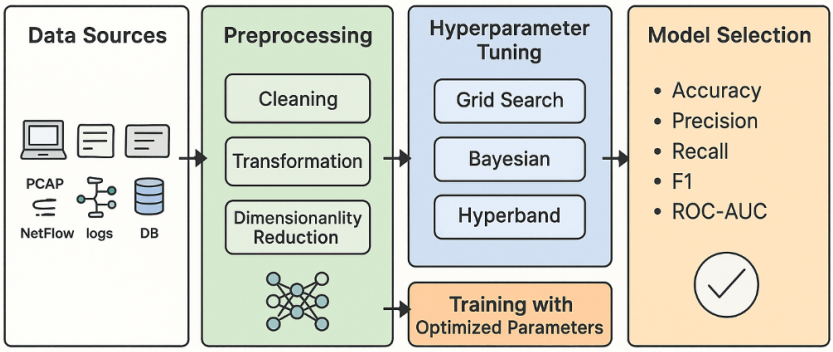


Figure 10: Centralized-based DDoS attack detection.

mark datasets.

3.2.2 Model Architectures

DL-based DDoS detectors fall into supervised instance models (classify individual flows), supervised sequence models (exploit temporal context), semi-supervised approaches, and hybrids that combine these paradigms [43]. Consistent with the datasets and real-time constraints, this work focuses on supervised instance and sequence learning over flow-level (payload-agnostic) features.

Deep Neural Networks (DNNs)

Researchers have widely applied feed-forward DNNs to flow-level features for distinguishing benign from malicious traffic and, in some cases, for identifying specific attack subtypes. Empirical studies show that DNNs, as well as simple CNN baselines, can learn effective decision boundaries on benchmark datasets such as CIC-IDS [44]. More recent work incorporates attention mechanisms into DNNs to emphasize salient features and improve detection performance [45]. Comparative evaluations against classical ML methods in cloud-based settings further indicate that DNNs generally achieve higher precision and accuracy. However,

the heavy reliance on legacy datasets (e.g., KDDCup99) and the limited consideration of deployment contexts such as LANs or cloud stratification reduce the external validity of these findings [46].

Convolutional Neural Networks (CNNs)

CNNs effectively capture local spatial structures in flow or window-based features while enabling fast inference. Prior work in SDN and ISP contexts demonstrates that CNN ensembles and near–real-time detectors can be deployed within controllers and edge devices [47, 49]. Sensitivity analyses further examine the impact of kernel depth, kernel size, and input encoding (e.g., grayscale versus RGB heatmaps) on datasets such as CIC-IDS2018 and KDDCup99 [48]. To enhance efficiency, multi-level systems combine lightweight entropy-based filters with downstream CNNs for fine-grained classification [50], while other approaches transform packet and flow traces into array-like tensors to simplify processing on resource-constrained devices [53]. CNNs have also been successfully applied to botnet and DDoS detection in IoT-specific datasets, including N-BaIoT and BoT-IoT [52].

Recurrent and Hybrid Models (RNN/LSTM, CNN–RNN)

Sequence models leverage temporal dynamics in network traffic, with LSTM and GRU variants mitigating vanishing gradients and improving the detection of long-range dependencies [144, 145]. Hybrid architectures often combine CNN layers to capture spatial features with RNN and LSTM layers for temporal context, thereby reducing false positives and enhancing recall across diverse datasets [51, 146]. Reported designs range from plain LSTMs to more advanced variants, including LSTM autoencoders for representation learning [58], heuristic threshold-based LSTM detectors [59], collaborative LSTMs with adaptive thresholds and multi-site feedback mechanisms [60], and cloud-oriented LSTM systems with integrated defense triggers [61]. These models are most frequently evaluated on CIC-IDS2017/2018/2019 datasets using binary or multi-class labels, sometimes directly on raw traffic without feature extraction [54–57].

Despite these advances, most studies still tune hyperparameters manually or through coarse grid search, with limited attention to latency, throughput, or memory footprint under real-time constraints. Cross-dataset and multiclass generalization also remain underexplored, hindered by heterogeneous feature spaces and class imbalance. Moreover, the reliance on legacy or synthetic datasets continues to complicate claims of robustness and practical applicability.

3.2.3 Hyperparameter Optimization

Search space & Algorithms

Hyperparameters play a critical role in shaping a model's performance, efficiency, and robustness. By systematically tuning these parameters, we seek to balance accuracy, computational cost, and resilience to diverse conditions. In the context of DDoS detection, hyperparameter optimization must also account for the variability of attack scenarios and the heterogeneous characteristics of network traffic, which influence the optimal parameter configurations.

Several techniques are commonly used for hyperparameter optimization, including random search, grid search, Bayesian optimization, and Hyperband. Among these, Hyperband improves upon random search by applying explore–exploit principles to efficiently prioritize candidate configurations and allocate resources more effectively. This work employs the Hyperband implementation available in the Keras Tuner library, as it offers a practical balance between runtime, resource utilization, and detection performance.

The proposed work is structured as a four-step optimization process. *(i) Model definition:* We select and define the DL architecture tailored to each dataset, establishing the foundation for optimization. *(ii) Hyperparameter selection:* We identify the model-specific parameters to be tuned (including depth, width, learning rate, batch size, kernel size). *(iii) Search space definition:* We specify the range of values or distributions for each hyperparameter. *(iv) Search algorithm specification:* We apply Hyperband to efficiently explore the defined space, as summarized in Table 3.

Table 3: Global hyperparameters and search space.

Hyperparameter	Type	Search space
Learning rate	Continuous	log-uniform $[10^{-5}, 10^{-1}]$
Batch size	Categorical	$\{32, 64, 128, 256\}$
Optimizer	Categorical	{Adam, SGD, RMSprop}
Units per layer	Integer	$[64, 512]$
Dropout rate	Continuous	uniform $[0.0, 0.5]$
Activation function	Categorical	{ReLU, Tanh, Sigmoid}

Table 4: Model-specific hyperparameters and search space.

Hyperparameter	Type	CNN	RNN/LSTM
# convolutional layers	Integer	$[1, 4]$	–
Filters per conv. layer	Integer	$[32, 256]$	–
Kernel size	Categorical	$\{3, 5, 7\}$	–
Pool size	Categorical	$\{2, 3\}$	–
Fully connected units (post-CNN)	Integer	$[64, 256]$	–
# recurrent layers	Integer	–	$[1, 3]$
Hidden units per layer	Integer	–	$[64, 256]$
Recurrent dropout rate	Continuous	–	uniform $[0.0, 0.5]$
Bidirectional	Categorical	–	{True, False}
Sequence length (timesteps)	Integer	–	$[20, 100]$

3.2.4 Databases used and Preprocessing

The description of the dataset and the preprocessing steps is based on our previously published work [3][1]. and the dataset description can be out in the [1].

To assess the effectiveness of the proposed adaptive transfer learning models, we utilized four widely studied cybersecurity datasets, KDDCup'99 [131], UNSW-NB15 [134], CSE-CIC-IDS2018 [135], and CIC-DDoS2019 [136]. These datasets are commonly regarded as standard benchmarks in the cybersecurity field [147, 148]. They cover a broad

[1]This subsection incorporates adapted material from our published article: M.B. Anley, A. Genovese, and V. Piuri, "Robust DDoS Attack Detection with Adaptive Transfer Learning," *Computers & Security*, 2024. Reused material has been adapted for inclusion in this thesis.

range of attack behaviors, enabling the training of DL models capable of recognizing diverse threat types. In the sections that follow, we provide detailed descriptions of each dataset along with the preprocessing steps applied.

KDDCup'99 dataset [131]. The KDDCup'99 dataset was developed for the KDDCup 1999 competition, which focused on creating effective techniques for identifying unauthorized access and malicious behavior in computer networks. It contains a large collection of network connection records—approximately 5 million entries covering both normal traffic and 22 different attack types, grouped into four primary categories. These include DoS attacks (back, LAND, ping of death, teardrop, Neptune, and Smurf), U2R attacks (buffer overflow, loadmodule, perl, and rootkit), R2L attacks (ftp_write, guess_password, imap, multihop, PHF, spy, warezclient, and warezmaster), and probe attacks (portsweep, ipsweep, NMAP, and Satan). Each record in the dataset is described by 42 features [149]. The dataset was preprocessed following the procedures in the work [3]. Categorical variables (protocol type, flag, service) were transformed via one-hot encoding. All features were then scaled to the 0,1 range using min–max normalization. Duplicate records, entries with missing (NaN) values, and all-zero columns were removed. After preprocessing, the dataset contained 494,020 instances and 42 features.

UNSW-NB15 dataset [134]. The UNSW-NB15 dataset includes 9 distinct attack families and a total of 49 features. The attack categories comprise Analysis, Fuzzers, Backdoors, DoS, Exploits, Reconnaissance, Generic, Shellcode, and Worms. These classes represent a broad spectrum of modern cyber threats, making the dataset suitable for comprehensive IDS evaluation. After preprocessing, 642,566 records and 45 selected features were retained for subsequent analysis [3].

CSE-CICIDS2018 dataset [135]. This dataset captures network activity within a controlled laboratory setup, including both benign traffic and seven different categories of cyberattacks. The testbed consists of 50 attacker machines and an emulated victim organization composed of five departments, incorporating 420 hosts and 50 servers. Network

traffic and system logs were collected from every device involved [135]. The dataset covers a wide range of malicious behaviors, such as DoS, DDoS, port-scanning activities, and various forms of malware. To facilitate machine-learning-based analysis, the creators released a preprocessed version in CSV format, containing 80 flow-based features extracted using CICFlowMeter-V3. In this work, we focus on the portions of the dataset pertaining specifically to benign traffic and DDoS scenarios. The DDoS subset includes seven attack types: GoldenEye, Slowloris, Hulk, SlowHTTPTest, LOIC-HTTP, HOIC, and LOIC-UDP, together with normal network flows [3]. The proposed work performed the preprocessing and discovered and removed duplicate rows in the dataset, eliminating 3,708,162 redundant entries. Additionally, that removed 17 columns, which comprised socket-related features and only zero values. After conducting normalization and data quality enhancement procedures, the dataset consists of 7,384,563 rows and 66 features [3].

CIC-DDoS2019 dataset The CIC-DDoS2019 dataset, presented in [136], provides an extensive collection of traffic traces covering multiple DDoS attack vectors, including UDP floods, TCP SYN floods, and HTTP floods. This level of detail enables a fine-grained examination of the behavioral characteristics associated with each attack type [3]. During preprocessing, 59,936,580 rows and 20 columns, primarily composed of zero-valued or socket-related features exhibiting little variation, were removed. This reduction yielded a cleaned dataset of roughly 10 million records with 66 retained attributes [2].

The dataset originally contains 18 attack categories, spanning both reflection and exploitation-based techniques: DrDoS-LDAP, DrDoS-MSSQL, DrDoS-NetBIOS, DrDoS-NMP, DrDoS-SSDP, DrDoS-UDP, UDP-lag, Web-DDoS, SYN, TFTP, DrDoS-DNS, DrDoS-NTP, Portmap, NetBIOS, LDAP, MSSQL, UDP, and UDPLag. For multi-class classification, similar attacks were consolidated based on their underlying mechanisms, traffic patterns, and naming conventions. For example, UDP-based attacks include DrDoS-UDP, UDP, UDPLag, and UDPLag, which are grouped because they share the common objective of overwhelming a target through excessive packet transmission. This grouping strategy, consistent with prior

Table 5: Traffic types and their cardinality from the preprocessed CIC-DDoS2019, CSE-CIC-IDS2018, UNSW-NB15, and KDDCup'99 datasets.

CIC-DDoS2019		CSE-CIC-IDS2018		UNSW-NB15		KDDCup'99	
Traffic	**Cardinality**	**Traffic**	**Cardinality**	**Traffic**	**Cardinality**	**Traffic**	**Cardinality**
TFTP	4,396,770	Benign	6,412,040	Benign	321,283	Benign	97,280
UDP	2,510,574	GoldenEye	41,406	Generic	215,481	DoS	391,457
NTP	1,112,902	Slowloris	9,908	Exploits	44,525	U2R	52
SSDP	891,220	LOIC-HTTP	575,364	DoS	16,353	R2L	1,124
SYN	687,524	LOIC-UDP	1,730	Recon.	13,987	Probe	4,107
MSSQL	484,070	HOIC	198,861	Fuzzers	24,246	–	–
SNMP	114,179	lowHTTPTest	55	Analysis	2,677	–	–
DNS	113,252	HULK	145,199	Backdoor	2,329	–	–
BENIGN	99,154	–	–	Shellcode	1,511	–	–
LDAP	48,469	–	–	Worms	174	–	–
NetBIOS	31,052	–	–	–	–	–	–
Portmap	1,638	–	–	–	–	–	–
WebDDoS	414	–	–	–	–	–	–
Total	*10,491,218*	*Total*	*7,384,563*	*Total*	*642,566*	*Total*	*494,020*

literature [150], streamlines the dataset without altering the core attack semantics, ultimately improving dataset manageability and model training efficiency. After merging, the dataset comprises 12 attack categories that include TFTP, UDP, NTP, SSDP, SYN, MSSQL, SNMP, DNS, BENIGN, LDAP, NetBIOS, Portmap, and WebDDoS. Binary classification was also performed to differentiate benign traffic from DDoS-related flows. Table 5 summarizes the cardinality of the processed dataset [3].

3.2.5 Data Preprocessing

We preprocess the data by performing *(i)* cleaning, *(ii)* transformation, and *(iii)* dimensionality reduction.

Data Cleaning. Preprocessing validates and corrects inconsistencies to ensure training integrity. Non-informative columns (e.g., socket-related fields, all-zero features) are dropped. Duplicate records are often due to redundant network captures, and rows with NaN values are removed. Any remaining infinite values are assigned 1 (sentinel) before normalization, preserving the integrity of statistical analyses.

Data Transformation. This stage focuses on converting the datasets into a consistent numerical format suitable for model training. Numerical

features are first normalized to the $[0,\ 1]$ interval using min–max scaling. Categorical attributes are then transformed into numerical representations through label encoding techniques. Two forms of encoding are applied: (i) label encoding, which assigns a unique integer to each category, and (ii) one-hot encoding (OHE), which maps each category to an n-dimensional binary vector, where n denotes the number of distinct categories. During preprocessing, continuous features were normalized using min–max normalization to preserve consistent value ranges across datasets, whereas categorical variables that include protocol or service indicators were encoded using OHE. Additionally, temporal features were extracted from timestamp fields to better capture evolving network traffic behaviors [3].

Data Dimensionality Reduction. This phase focuses on reducing the feature space to minimize noise, speed up training, and ensure a unified feature representation across the different datasets. To accomplish this, principal component analysis (PCA) is applied, with the optimal number of principal components determined through the maximum likelihood estimation (MLE) approach, a statistical method used to identify the parameter values that best describe the underlying data distribution [151]. Because network traffic data typically exhibit high dimensionality, we combined correlation analysis with PCA to eliminate redundant features while preserving more than 95% of the dataset's original variance. This reduction strategy lowers computational overhead and enhances the effectiveness of downstream deep learning model training [2].

3.2.6 Evaluation Metrics

Detection performance is evaluated using classification metrics standard in cybersecurity research, accuracy, precision, recall, macro-F1, and AUROC.

- *Accuracy.* measures the proportion of correctly classified instances over the total, calculated as $(TP + TN)/(TP + TN + FP + FN)$.

- *Precision.* quantifies the fraction of correctly identified attacks among all predicted attacks, given by $TP/(TP + FP)$.

- *Recall.* (or detection rate) captures the fraction of actual attacks correctly identified, expressed as $TP/(TP + FN)$.

- *F1-score.* is the harmonic mean of precision and recall, balancing false positives and false negatives, defined as $2 \cdot (\text{Precision} \cdot \text{Recall})$ $/(\text{Precision} + \text{Recall})$.

- *ROC-AUC.* evaluates the trade-off between true positive and false positive rates across thresholds, with the area under the Receiver Operating Characteristic curve providing a threshold-independent measure of separability.

- *Confusion Matrix.* summarizes classification outcomes in terms of true positives, true negatives, false positives, and false negatives, providing a complete view of model errors and correct predictions.

3.3 Experimental Results and Discussion

This section reports the empirical results of the method, covering binary classification (DDoS vs. benign) and multi-class classification by attack type. Training dynamics are examined via learning curves (loss and accuracy), followed by a discussion of limitations and implications.

3.3.1 Binary Class Detection

Binary tasks (Table 6), Across all four datasets, all models achieve strong performance (Accuracy $\geq$ 94.8% and ROC–AUC $\geq$ 95.0%), with **LSTM** consistently ranking first. The gains are small but systematic over CNN and DNN, typically $\approx$0.2–0.4 percentage points (pp) on F1 and $\approx$0.1–0.2 pp on ROC-AUC (e.g., on CIC-IDS2019 LSTM reaches 99.0% Accuracy / 98.6% F1 / 99.5% ROC–AUC vs. CNN's 98.8 / 98.2 / 99.3). On KDD-Cup99, an older, highly imbalanced benchmark, the absolute numbers are lower (LSTM F1 of 95.0%) but the ranking persists. **RNN** trails the other

Table 6: Binary classification performance of deep learning models across four benchmark datasets. Best results per dataset are shown in **bold**.

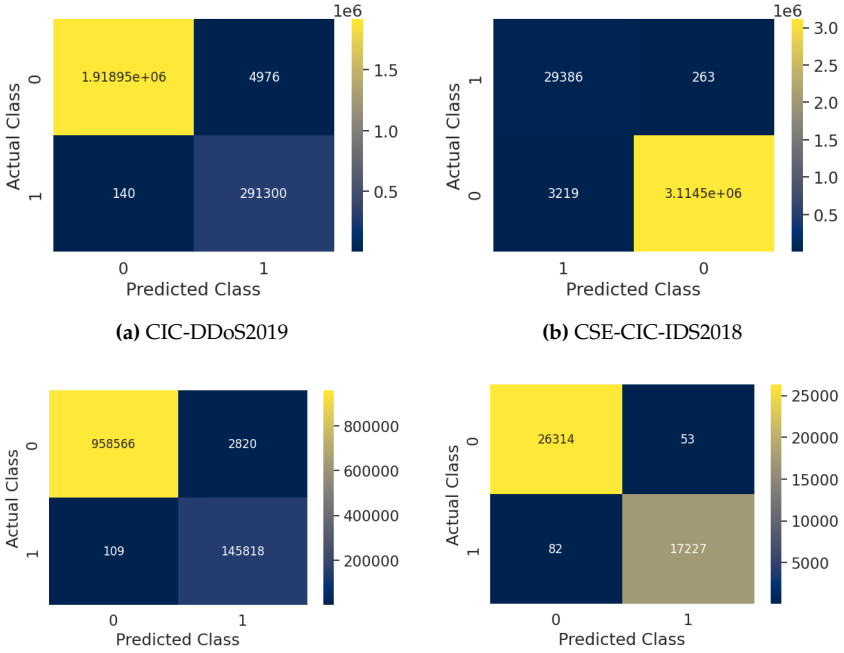
Dataset	Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	ROC-AUC (%)
CIC-IDS2018	DNN	98.5	98.0	97.8	97.9	99.0
	CNN	98.7	98.3	97.9	98.1	99.2
	RNN	98.2	97.8	97.4	97.6	98.8
	LSTM	**98.9**	**98.6**	**98.1**	**98.3**	**99.3**
CIC-IDS2019	DNN	98.6	98.2	98.0	98.1	99.1
	CNN	98.8	98.4	98.1	98.2	99.3
	RNN	98.4	98.0	97.8	97.9	98.9
	LSTM	**99.0**	**98.7**	**98.5**	**98.6**	**99.5**
KDDCup99	DNN	95.0	94.5	94.0	94.2	95.5
	CNN	95.3	95.0	94.8	94.9	95.8
	RNN	94.8	94.2	94.0	94.1	95.0
	LSTM	**95.5**	**95.1**	**95.0**	**95.0**	**96.0**
UNSW-NB15	DNN	97.0	96.5	96.0	96.3	97.5
	CNN	97.3	97.0	96.8	96.9	97.8
	RNN	96.8	96.2	96.0	96.1	97.0
	LSTM	**97.5**	**97.2**	**97.0**	**97.1**	**98.0**

architectures by a small margin ($\approx$0.3–0.7 pp), suggesting that deeper gating and longer effective memory (LSTM) or localized feature interaction CNN better fit these feature spaces than a basic recurrent stack.

The consistent yet modest LSTM advantage indicates some temporal/ordering signal or feature co-occurrence that benefits gated recurrence even when inputs are primarily tabular. At the same time, the narrow margins imply that rich temporal structure is not the dominant factor in these binary settings; **CNN1D** and **DNN** remain highly competitive, often within a few tenths of a percentage point. High ROC-AUC ($\geq$97.8% on CIC/UNSW) further suggests stable separability across thresholds–useful when reporting operating points (e.g., fixed FPR at 1% and 0.1%) (see Figure 11).

3.3.2 Multi-class Detection

In the multi-class experiments, the same general trends are observed; however, performance gaps widen on minority classes. Macro-averaged F1 tends to drop relative to micro-averaged metrics due to class imbalance, and per-class analysis reveals frequent confusions between semantically



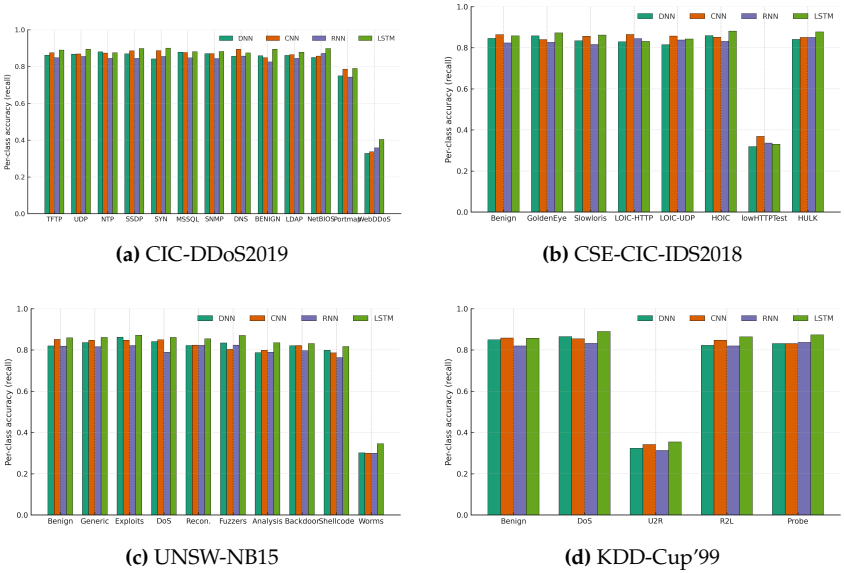
(a) CIC-DDoS2019 **(b)** CSE-CIC-IDS2018

(c) UNSW-NB15 **(d)** KDD-Cup'99

Figure 11: Confusion matrix results for the binary classification of the DNN models for the above four datasets

similar attacks (e.g., DoS vs. DDoS). In such cases, **CNN1D** occasionally matches the performance of **LSTM**, suggesting that localized feature interactions can substitute for temporal modeling when sequential dependencies are weak. To provide a fair assessment, we therefore emphasize macro-F1 and balanced accuracy in addition to ROC-AUC.

As shown in Figure 13, detection accuracy improves with larger sample sizes and higher traffic cardinality, particularly for underrepresented classes. Furthermore, across all four datasets, the models consistently achieve high accuracy in detecting benign traffic, which serves as a stable reference point for evaluating attack-class performance.



(a) CIC-DDoS2019

(b) CSE-CIC-IDS2018

(c) UNSW-NB15

(d) KDD-Cup'99

Figure 12: Per-class detection accuracy for DNN, CNN, RNN, and LSTM. Results are computed on the test split at the selected operating threshold (validation-tuned)

3.3.3 Class-Level Behavior and Error Patterns

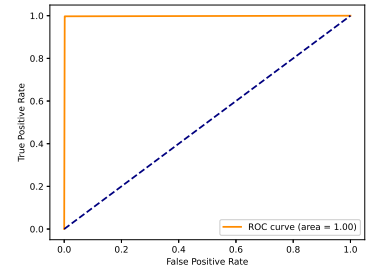
Across the four datasets, the per-class bar charts 13 show a consistent pattern: LSTM and CNN deliver the best recalls, followed closely by DNN, while plain RNN trails slightly. High-volume or easier classes (e.g., TFTP, UDP, NTP in CIC-DDoS2019, Benign in all datasets, DoS, Probe in KDDCup99, and Generic, Exploits in UNSW-NB15) achieve high recall in the mid-80s to high-80s range. In contrast, rare or inherently harder classes display marked degradation, e.g., WebDDoS in CIC-DDoS2019, lowHTTPTest in CSE-CIC-IDS2018, Worms, Shellcode in UNSW-NB15, and U2R (and to a lesser extent R2L) in KDDCup99, where recalls drop into the 0.3–0.6 band. The relative ordering (LSTM $\gtrsim$ CNN $\gtrsim$ DNN $\gtrsim$ RNN) remains stable across attack types, but the gaps widen on the lowest-cardinality classes.

3.3.4 Training Dynamics (Loss/Accuracy Curves)

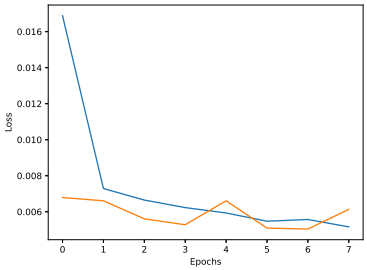
The ROC curves for both RNN and CNN on CIC-IDS-2018 (Figs. 11a and 14a) demonstrate strong separability in the binary setting, with the CNN trace rising more steeply in the low-FPR region, critical when false alarms must be tightly controlled. The training dynamics further show that the CNN converges faster and more smoothly, whereas the RNN plateaus later and exhibits mild train-validation divergence after several epochs, suggesting slight overfitting when temporal inductive bias is not strongly rewarded by the features. Accuracy curves mirror these trends, stabilizing around the early-stopping epoch. The confusion matrices remain largely diagonal, consistent with high overall accuracy; however, residual errors are asymmetric, with benign $\rightarrow$ attack misclassifications being more consequential. These observations motivate selecting operating thresholds from ROC (and PR) curves at fixed FPR budgets and, where useful, applying simple post hoc calibration.

On CIC-IDS-2018, the CNN achieves a favorable accuracy–latency trade-off and stronger low-FPR performance, making it a practical default for inline detection. The RNN remains competitive, but its temporal modeling provides limited gains in this binary, tabular setting. To further reduce residual errors, we apply imbalance-aware training strategies (e.g., class weighting or focal loss) and tune thresholds on a validation split to meet operational FPR targets (e.g., 1% or 0.1%). For robustness, we also report macro-averaged metrics alongside micro accuracy and confirm stability across random seeds.

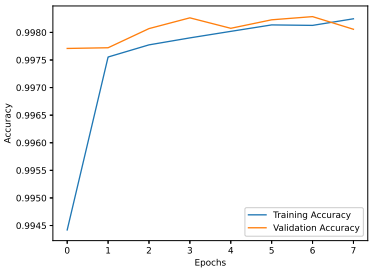
Overall, the centralized evaluation of DL models for DDoS detection demonstrates that binary classification of benign versus attack traffic is highly accurate across benchmarks, with CNNs providing a favorable balance between detection accuracy and inference latency. In the more complex multi-class setting, performance degrades on minority attack types due to class imbalance and semantic overlap (e.g., DoS vs. DDoS), underscoring the need for imbalance-aware training and careful threshold tuning. Training dynamics further reveal that CNNs converge faster and generalize more smoothly than sequence models, while RNNs and



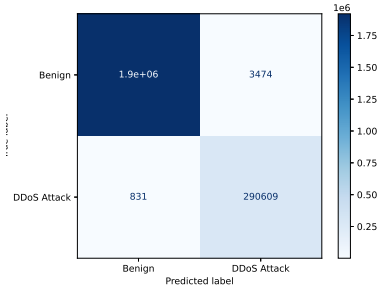
(a) ROC Curve



(b) Loss Curve



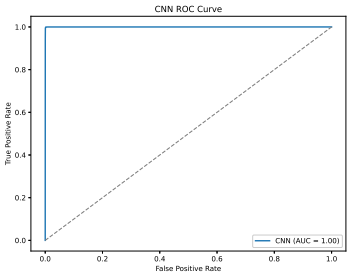
(c) Accuracy Curve



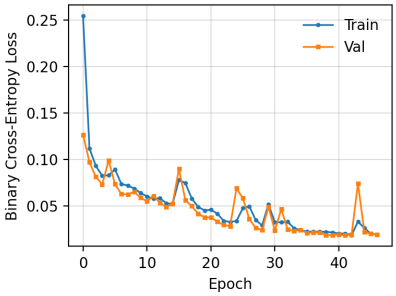
(d) Confusion Matrix

Figure 13: Comparison of performance metrics for the RNN model on the CIC-IDS-2018 dataset.

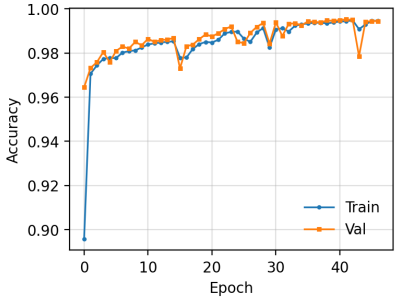
LSTMs offer limited advantages in tabular, flow-based data where temporal dependencies are weak. Taken together, these results suggest that CNN-based detectors, complemented by class-balancing techniques and robust threshold calibration, represent a practical and scalable baseline for centralized DDoS detection in complex, large-scale network scenarios. Furthermore, systematic hyperparameter optimization proves essential for achieving robust trade-offs between accuracy, efficiency, and generalization across diverse datasets.



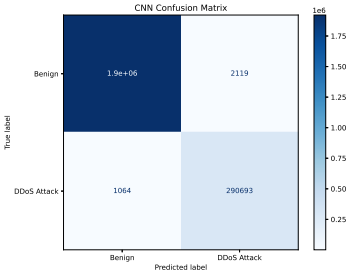
(a) ROC Curve



(b) Loss Curve



(c) Accuracy Curve



(d) Confusion Matrix

Figure 14: Comparison of performance metrics for the CNN model on the CIC-IDS-2018 dataset.

3.3.5 Limitations and Future Work

Data imbalance and drift. One of the major challenges observed in the proposed experiments is the presence of severe class imbalance in benchmark datasets such as CIC-DDoS2019, where benign traffic is significantly underrepresented compared to attack traffic. This imbalance tends to bias performance metrics, particularly accuracy, and can lead to inflated evaluations of model effectiveness. However, real-world deployments are further complicated by data drift, where traffic distributions evolve due to new attack variants or changes in benign behavior. Addressing drift requires adaptive mechanisms, online learning, or continual training to

ensure long-term robustness.

Cross-domain generalization. Although the proposed models achieved strong performance within individual datasets, transferring these models across domains remains a considerable challenge. Differences in feature schemas, traffic distributions, and attack characteristics b/n datasets (CIC-IDS2018, CIC-DDoS2019, UNSW-NB15, and BoT-IoT) introduce heterogeneity that limits direct generalization. This underscores the difficulty of deploying models trained on one environment into another without significant performance degradation. Promising extensions include domain adaptation techniques, such as feature alignment and adversarial transfer, as well as TL strategies that fine-tune models on small amounts of target-domain data. These directions could significantly enhance cross-domain robustness and ensure better applicability of detection systems in diverse IoT and enterprise environments.

Security considerations. While this work primarily focused on centralized deep learning-based detection, adversarial threats such as evasion and poisoning remain critical concerns. Attackers may attempt to generate adversarial traffic patterns that mimic benign flows or inject poisoned samples into training pipelines to degrade model performance. These considerations suggest that future DDoS detection systems must move beyond accuracy to emphasize adversarial robustness and resilience against sophisticated attack vectors.

Compute constraints. Another limitation lies in the computational cost associated with DL models at the edge, where IoT devices and gateways often have limited resources. Models with large architectures may be impractical due to constraints on memory, power, and latency. Lightweight optimization techniques such as model quantization, pruning, and knowledge distillation provide potential pathways to reduce complexity while preserving detection accuracy. Furthermore, edge-cloud collaborative frameworks could enable the offloading of heavy computations to more capable servers, while leaving lightweight inference tasks at the edge. Balancing detection accuracy with efficiency is essential for practical adoption in resource-constrained IoT environments.

3.4 Summary

This chapter presented a centralized DL pipeline for DDoS detection that integrates preprocessing, dimensionality reduction, automated hyperparameter optimization, and fine-tuning. Standardized flow-level features addressed class imbalance and applied PCA to compress inputs while retaining essential traffic characteristics. Using Hyperband, we efficiently tuned DNN, 1D-CNN, RNN, and LSTM models across CIC-IDS2018/2019, UNSW-NB15, and KDDCup99. Evaluation employed F1, ROC/PR-AUC, and accuracy on both binary and multi-class tasks. Results showed that centralized DL consistently improved detection accuracy, while PCA reduced overfitting and computational cost, and Hyperband identified competitive configurations with limited resources. CNNs and LSTMs performed strongly on harder classes, whereas compact DNNs offered the best accuracy–latency trade-off for inline deployment.

The section highlights practical limitations and implications. Minority, low-cardinality attacks remain the dominant source of errors, underscoring the need for imbalance-aware training and calibrated thresholds. Performance is further challenged by distribution shifts across time, domains, and traffic mixes, motivating continual monitoring and periodic re-tuning. Efficiency can be enhanced through model compression techniques such as quantization and distillation. These findings naturally lead into the next chapter on transfer learning, where we leverage pretrained representations and apply lightweight adaptation to improve generalization under limited labels and evolving attack landscapes.

Chapter 4

Transfer Learning based Evolving Attack Detection

4.1 Introduction

The rapid expansion of edge-based IoT deployments across critical infrastructures, from smart cities to industrial IoT systems, has transformed how data is processed and decisions are made at the network edge. Yet this increasing interconnection also creates substantial security exposure, particularly to DDoS attacks that can disrupt service availability, degrade performance, and threaten the integrity of essential operations [28].

Although anomaly-based detection methods supported by deep learning show strong potential due to their ability to automatically extract features, their performance is often constrained by the requirement for large, high-quality labeled datasets. This limitation reduces their adapt-

This chapter is based on the following published works by the author:

1. M.B. Anley, A. Genovese, and V. Piuri, "Robust DDoS Attack Detection with Adaptive Transfer Learning," *Computers & Security*, 2024. Licensed under CC BY. See: `https://creativecommons.org/licenses/by/4.0/`.

2. M.B. Anley, A. Genovese, and V. Piuri, "TransferEdge: Transfer learning approach to detect evolving DDoS threats in Edge-IIoT," Proceedings of the IEEE RTSI 2025 Conference. This material is reused here in accordance with the publisher's permitted rights for thesis inclusion.

Reused content complies with all applicable license terms and publication policies.

ability in dynamic, data-limited IoT environments [152]. IoT ecosystems further complicate this challenge due to resource limitations, privacy constraints, and the significant effort required to label data for DL model training [88, 153].

Transfer learning has emerged as a viable strategy for intrusion and DDoS detection in IoT, particularly for edge IIoT devices commonly found in critical infrastructure [89]. Several TL-based approaches have been proposed: a CNN-based method in [153], the use of pre-trained transformer models to address distribution shifts in [87], a dual autoencoder framework in [88], and fine-tuning of pre-trained CNN and BiLSTM architectures for 5G DDoS detection in [154]. Despite these advancements, important gaps remain, especially in developing fine-tuning strategies that enable efficient and practical deployment of pre-trained models on resource-limited edge IIoT devices [2].

To address these challenges, this thesis proposes *TransferEdge*, a TL framework for Edge–IIoT DDoS detection. TransferEdge leverages CNNs and pretrained backbones and introduces computationally efficient fine-tuning strategies that align source and target distributions while reducing compute and label requirements. Cross-domain evaluations were performed to assess the robustness of transfer-learning variants under distribution shifts with few labels.

Contributions. *(i)* Computationally efficient transfer learning that optimizes fine-tuning for specific datasets, including domain alignment, differential fine-tuning with variable learning rates, and selective layer updates of attack-sensitive components to cut computational overhead while preserving accuracy. *(ii)* Validation on real-world cybersecurity datasets (CIC-IDS208/19, UNSW-NB15, BoT-IoT), demonstrating effectiveness in resource-constrained Edge and IIoT settings in terms of detection accuracy, latency, and training cost. *(iii)* Exploration of image-based encodings by converting packet features into array-RGB heatmaps to test model robustness and cross-domain transfer, and to compare against tabular CNN baselines. Moreover, this chapter is organized around our published works [2, 3] for comprehensive methodological and experimental details; see those papers.

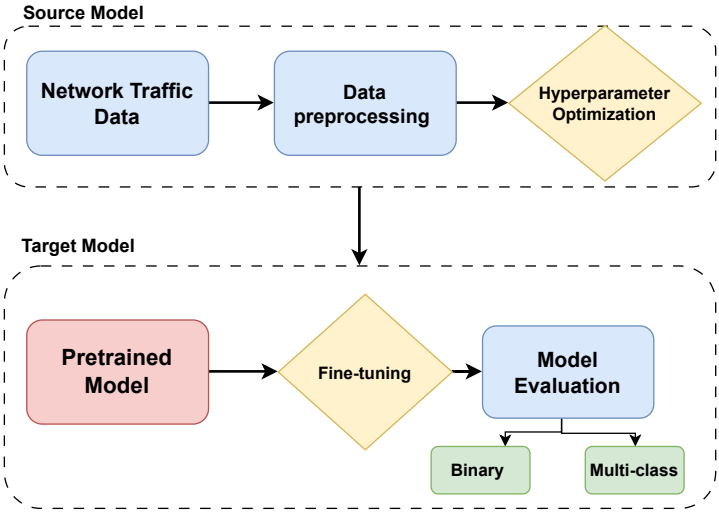


Figure 15: Transfer learning for network traffic classification.

4.2 Methodology

This subsection details the transfer learning methodology for edge-based IIoT DDoS detection. The domain-adaptation framework, baseline architectures, TransferEdge fine-tuning strategies, and the evaluation methodology are described.

4.2.1 Overview of the Transfer Learning Framework

Transfer learning is an approach that reuses knowledge obtained from one domain (the source domain) to enhance model performance in a related but distinct domain (the target domain). In this paradigm, a model is first trained on a large dataset from the source domain D_S, and the acquired representations are then transferred to a target domain D_T that typically contains far fewer training samples. This process can be represented as

follows: Given a source dataset:

$$D_S = \{(x_i^S, y_i^S)\}_{i=1}^{N_S} \tag{4.1}$$

with N_S samples, a model f_S is trained to learn representations by optimizing parameters θ_S:

$$\theta_S^* = \arg\min_{\theta_S} \frac{1}{N_S} \sum_{i=1}^{N_S} \ell(f_S(x_i^S; \theta_S), y_i^S) \tag{4.2}$$

where $\ell(\cdot)$ is the loss function. The learned source weights initialize the target CNN for the edge-based IIoT domain. Because the target data are limited, fine-tuning minimizes the average loss on the target samples. The learned parameters θ_S^* are transferred to the target model f_T in the target domain:

$$D_T = \{(x_j^T, y_j^T)\}_{j=1}^{N_T} \tag{4.3}$$

where $N_T \ll N_S$. The model is then fine-tuned or adapted:

$$\theta_T^* = \arg\min_{\theta_T} \frac{1}{N_T} \sum_{j=1}^{N_T} \ell(f_T(x_j^T; \theta_T), y_j^T) \tag{4.4}$$

This enables the model to preserve the informative representations learned from D_S while adapting to the distinctive patterns present in D_T. Such adaptation allows the models to capture the specific characteristics of the edge IIoT environment without losing the essential DDoS-related features obtained during source-domain training.

4.2.2 Baseline Model Architecture

In this work, a CNN model trained on UNSW-NB15 serves as the source model; knowledge is transferred to BoT-IoT using a suite of fine-tuning strategies. In addition, several state-of-the-art DL architectures are implemented and fine-tuned to benchmark performance on the target dataset. Architectural details appear below.

CNN model

CNNs serve as the foundational architecture for automatic hierarchical feature extraction from network traffic. CNNs are chosen for their simplicity and computational efficiency properties that align with the resource constraints of edge-based IIoT devices, while providing a strong baseline for DDoS detection reported in the literature [155].

VGG8/VGG16

VGG8 is used as a lightweight member of the VGG family, with fewer layers to balance accuracy and resource efficiency. Its simple, sequential design enables efficient inference on constrained hardware [156]. VGG16 is also included for its deeper architecture, which learns more detailed feature representations [157].

ResNet18

ResNet18 incorporates residual skip connections, originally introduced by Microsoft [158], which allow the network to be trained at greater depth without suffering from vanishing gradients. This design makes it effective in capturing complex patterns within diverse edge-generated data.

EfficientNet

EfficientNet employs a compound scaling approach that simultaneously adjusts depth, width, and resolution [159]. This strategy delivers strong accuracy while maintaining a relatively small parameter count and reduced computational overhead, making it highly suitable for deployment in resource-limited Edge-IIoT environments.

Xception

The framework employs Xception pre-trained on ImageNet [160] for its transferable feature extraction, which generalizes well across tasks even under limited-label regimes typical of resource-constrained IIoT settings.

4.2.3 Fine-Tuning Strategies

The *edgeTransfer* framework applies four fine-tuning strategies to enhance the transfer of learned representations from the source domain to the target IIoT environment. These strategies, illustrated in Figure 16, are described below.

TL0 - Baseline Transfer Learning (No Freezing)

In this baseline configuration, TransferEdge updates the entire pre-trained network on the target dataset without freezing any layers. By allowing all parameters to adjust, the model can fully internalize the characteristics of new or evolving attack behaviors and learn features specific to the target domain.

TL1 - Last-Layer Retraining

In this setting, TransferEdge assumes that most of the pre-trained feature extractor captures domain-general patterns that remain relevant to the target task. Consequently, only the final classification layer is retrained, while the preceding layers remain fixed. This approach limits the number of trainable parameters and focuses solely on adapting the decision boundaries to the target attack distribution [2].

TL2 - Differential Fine-Tuning

Differential fine-tuning applies layer-specific learning rates depending on each layer's role in representation learning. Early convolutional layers responsible for low-level, generic feature extraction are updated with a small learning rate (1×10^{-5}), whereas deeper layers associated with higher-level, domain-specific patterns are trained with a larger rate (1×10^{-3}). TransferEdge updates intermediate layers at moderate rates, balancing stability with adaptability [2].

TL3 -Selective Layer Fine-Tuning

Selective fine-tuning adopts a targeted strategy by updating only those layers that contribute most to performance on the target domain. TransferEdge uses gradient-based sensitivity analysis to identify these layers [161]. Specifically, the gradients of the loss with respect to each layer's parameters are evaluated to measure their influence on the final prediction. Layers exhibiting large gradient magnitudes are deemed critical for capturing attack-specific features and are fine-tuned, while the remaining layers remain frozen. The subsequent equations formalize this selection mechanism [2].

In particular, which consider the edge IIoT traffic data X, a pre-trained model $f(X;\theta)$ which computes predictions $\hat{y}$ and the cross-entropy loss $\mathcal{L}$:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i, \tag{4.5}$$

where $\theta = \{\theta_1, \ldots, \theta_L\}$ are model parameters across L layers. During backpropagation, TransferEdge quantifies the sensitivity of layer l as the expectation of gradient magnitudes over training batches:

$$S_l = \mathbb{E}_{\text{batches}}\left[\left\|\frac{\partial \mathcal{L}}{\partial \theta_l}\right\|_2\right]. \tag{4.6}$$

Layers with higher S_l are prioritized for adaptation, as they most influence threat detection accuracy. TransferEdge classifies layers by S_l and selects the top-k for adaptation. Parameters in these layers are updated via:

$$\theta_l^{t+1} = \theta_l^t - \eta \cdot \mathbb{I}(l \in \text{Top-}k) \cdot \frac{\partial \mathcal{L}_{\text{new}}}{\partial \theta_l}, \tag{4.7}$$

where $\mathbb{I}(\cdot)$ is an indicator function (1 if l is selected, 0 otherwise), and $\mathcal{L}_{\text{new}}$ uses target datasets. Non-selected layers remain frozen, reducing computational costs by $1 - \frac{k}{L}$ for the edge deployment.

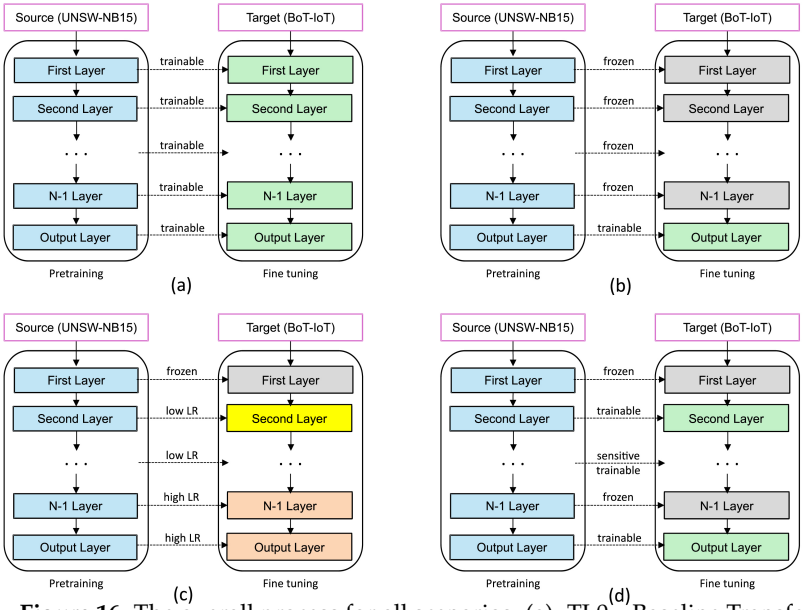


Figure 16: The overall process for all scenarios. (a): TL0 – Baseline Transfer Learning (No Freezing); (b): TL1–Last Layer Retraining; (c) TL2–Differential Fine-Tuning; (d): TL3 – Selective Layer Fine-Tuning [2].

4.2.4 Evaluation Metrics

Accuracy, precision, recall, F1 score, and robustness metrics are used to evaluate the results of the proposed TransferEdge approaches.

4.3 Experiments and Evaluation

4.3.1 Dataset

The pipeline trains a robust source model on UNSW-NB15 and adapts it to the smaller BoT-IoT dataset to emulate edge-based IIoT scenarios with limited data. To assess adaptability to evolving threats, the target dataset includes previously unseen attack types. The following sections detail preprocessing and dataset characteristics.

UNSW-NB15 dataset

The UNSW-NB15 dataset[1] is a widely used benchmark for evaluating network intrusion detection techniques, as it captures diverse traffic behaviors and multiple categories of cyberattacks. In our preprocessing pipeline, duplicate entries were removed, missing values were addressed, irrelevant attributes were discarded, numerical features were normalized using min–max scaling, and categorical attributes were encoded through one-hot encoding. Table 7 summarizes the attack types and their corresponding distribution used in our experiments.

BoT-IoT dataset

The BoT-IoT dataset[2] is another widely adopted benchmark for intrusion detection, reflecting typical network traffic alongside a range of attack vectors. In TransferEdge, the smaller-scale samples from this dataset are employed to emulate realistic Edge-IIoT environments, where data availability is limited. Serving as the target domain, BoT-IoT enables the evaluation of how effectively the pretrained model adapts under resource-constrained conditions. To incorporate evolving IIoT attack scenarios, TransferEdge also considers attack categories that were not encountered during source domain training. Preprocessing involved cleaning missing and duplicate values, normalizing all numerical attributes via min–max scaling, and applying one-hot encoding to categorical fields. The attack categories and their proportions used in our experiments are listed in Table 7 [2].

4.3.2 Data conversion

The pre-trained CNN architectures used for cross-domain adaptation, VGG16, VGG19, and ResNet50, were originally trained on large-scale image datasets. In contrast, network traffic data are typically collected in non-image formats, such as `.csv` or `.pcap` files. To effectively apply transfer learning for DDoS detection, these non-visual traffic records must

[1] `https://research.unsw.edu.au/projects/unsw-nb15-dataset`
[2] `https://research.unsw.edu.au/projects/bot-iot-dataset`

Table 7: Class Distribution in UNSW-NB15 and BoT-IoT Datasets after pre-processing.

UNSW-NB15	# Records	BoT-IoT	# Records
Benign	2,218,761	Benign	92,543
Generic	215,481	DoS	32,480
Exploits	44,525	DDoS	5,194
DoS	16,353	Theft	1,587
Recon.	13,987	Recon.	21,639

be converted into an image-based representation that is compatible with CNN models [3].

In this work, the numerical features of each dataset are first scaled to the range $[0,\ 1]$ to achieve initial normalization. After this step, a quantile transformation is applied to each feature. This procedure discretizes the normalized values into quantile bins and maps them onto a new interval of $[0,\ 255]$, corresponding to the typical intensity values used in image processing. This transformation enables the data to be represented as pixel values. Using the quantile-transformed features, images are then generated for each class in the datasets, representing both benign traffic and various attack types. The initial images are constructed as 9×9 pixel grids with three color channels (RGB), allowing different feature variations to be encoded through color intensities. When a class contained fewer instances, padding was applied to preserve uniform image dimensions. To ensure compatibility with widely used pre-trained CNN architectures, including VGG16, VGG19, and ResNet50, the 9×9 RGB images are subsequently resized to 224×224 pixels while maintaining the three-channel structure [3].

Following the conversion of flow and packet records into image representations, the experiments employed the following dataset sizes. For CSE-CIC-IDS2018, the binary classification task (benign vs. attack) consisted of 41,883 images. The multiclass extension, which distinguishes individual DDoS subtypes in addition to benign traffic, expanded the dataset to 269,616 images to provide sufficient coverage for each class. For CIC-DDoS2019, a total of 78,368 images were generated across 12 attack categories and benign traffic, distributed among training, validation, and

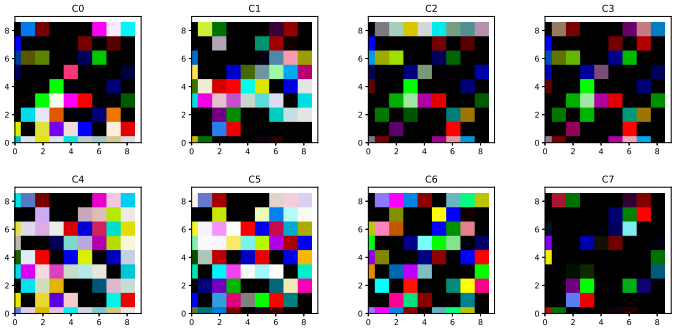


Figure 17: Representative samples showcasing converted images from each category, ranging from Class 0 to Class 7, derived from the CSE-CIC-IDS2018 dataset. The classes include: Benign Traffic (C0), DDoS Attacks - LOIC-HTTP (C1), DDoS Attack - HOIC (C2), DoS Attacks - Hulk (C3), DoS Attacks - GoldenEye (C4), DoS Attacks - Slowloris (C5), DDoS Attack - LOIC-UDP (C6), and DoS Attacks - SlowHTTPTest (C7) [3].

test sets. The experimental setup also incorporated 12,154 images derived from KDDCup'99 and 5,629 images from UNSW-NB15. Figure 17 displays representative examples of the encoded images for classes C0–C7 (C0–C6: attack classes; C7: benign), while Figure 18 illustrates sample images corresponding to the CIC-DDoS2019 categories [3].

4.3.3 Hyper parameters

To identify the optimal hyperparameter configuration, a uniform search space was defined for the experiment. The learning rate was sampled within the interval 1×10^{-5} to 1×10^{-4}; batch sizes were selected from $\{64, 128, 256\}$; weight decay values ranged from 0 to 1×10^{-4}; and the number of fine-tuning epochs varied between 20 and 100. Four transfer learning configurations were tested across the CNN, VGG8/16, ResNet18, EfficientNet, and Xception architectures. Hyperparameter optimization was performed using random search, with 50 trials allocated to each

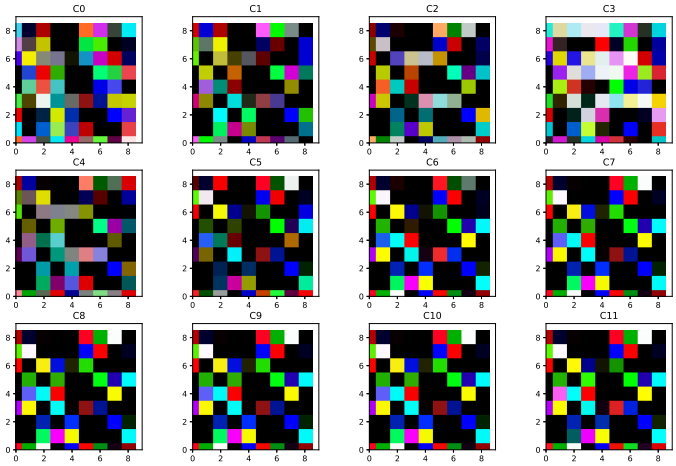


Figure 18: Representative samples showcasing converted images from each category, ranging from Class 0 to Class 11, including benign traffic from the CIC-DDoS2019 dataset. The classifications are as follows: Benign (C0), NTP (C1), TFTP (C2), Syn (C3), UDP (C4), MSSQL (C5), DNS (C6), LDAP (C7), DrDoS_SNMP (C8), NetBIOS (C9), SSDP (C10), and WebDDoS (C11) [3].

model scenario. All experiments were executed on Google Colab using a T4 GPU equipped with 16 vCPUs and 64 GB of RAM [3].

4.4 Experimental Results and Discussion

4.4.1 Comparative Analysis of Fine-tuning Strategies

TransferEdge fine-tuning strategies were evaluated across CNN baselines and pretrained architectures; results are summarized in Table 8. In the TL0 setting (no freezing) on the BoT-IoT dataset, VGG16 achieved near-perfect performance (99.99% precision, recall, and F1-score), outperforming CNN (99.90%), ResNet18 (99.83%), and the other models. However, VGG16 required 68.7 minutes for training, nearly three times longer than CNN

Table 8: Performance Comparison of Models on BoT-IoT Using TL0 – Baseline Transfer Learning (No Freezing).

Model	Accuracy	Precision	Recall	F1-Score	Train Time [m]
CNN	99.90	99.90	99.90	99.87	22.4
VGG8	99.87	99.87	99.87	99.87	41.8
VGG16	99.99	99.99	99.99	99.99	68.2
ResNet18	99.83	99.83	99.83	99.83	72.4
EfficientNet	99.87	99.87	99.87	99.87	47.1
Xception	99.91	99.91	99.91	99.91	53.51

Table 9: Performance Comparison of Models on BoT-IoT Using TL1 – Last Layer Retraining.

Model	Accuracy	Precision	Recall	F1-Score	Train Time [m]
CNN	99.89	99.89	99.89	99.86	38.7
VGG8	96.28	96.28	99.42	99.35	65.2
VGG16	99.64	99.64	99.64	99.64	108.4
ResNet18	99.84	99.84	99.84	99.84	59.7
EfficientNet	99.97	99.97	99.97	99.96	95.3
Xception	99.64	99.64	99.64	99.64	82.69

Table 10: Performance Comparison of Models on BoT-IoT Using TL2 – Differential Fine-Tuning.

Model	Accuracy	Precision	Recall	F1-Score	Train Time [m]
CNN	99.91	99.91	99.91	99.88	35.2
VGG8	99.84	99.84	99.84	99.85	62.4
VGG16	99.88	99.89	99.88	99.88	95.6
ResNet18	99.83	99.83	99.83	99.83	89.9
EfficientNet	99.87	99.87	99.87	99.87	71.9
Xception	99.75	99.75	99.75	99.75	77.83

(22.4 minutes), highlighting the need for a trade-off between detection accuracy and efficiency. In this sense, EfficientNet offered a balance, achieving 99.87% accuracy with a reduced training time of 47.1 minutes.

Table 9 presents the results for TL1 Last Layer Retraining, where EfficientNet achieved the highest performance (99.97% across all metrics), followed closely by CNN (99.89%) and ResNet18 (99.84%). VGG16 demonstrated strong but slightly lower results (99.64%), while VGG8 lagged with 96.28%, despite maintaining high recall.

In scenario TL2 - Differential Fine-Tuning, as shown in Table 10, CNN achieved the highest performance (99.91%), followed closely by VGG16 (99.88%), VGG8 (99.84%), and ResNet18 (99.83%). EfficientNet also per-

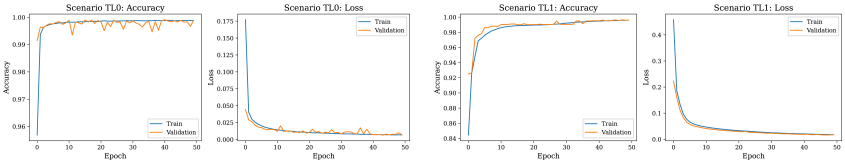
Table 11: Performance Comparison of Models on BoT-IoT Using TL3 – Selective Layer Fine-Tuning.

Model	Accuracy	Precision	Recall	F1-Score	Train Time [m]
CNN	99.89	99.89	99.89	99.86	8.4
VGG8	99.85	99.68	99.84	99.76	12.1
VGG16	99.92	99.92	99.92	99.92	24.7
ResNet18	99.88	99.88	99.88	99.88	27.5
EfficientNet	99.84	99.67	99.84	99.76	14.9
Xception	99.91	99.91	99.91	99.91	16.59

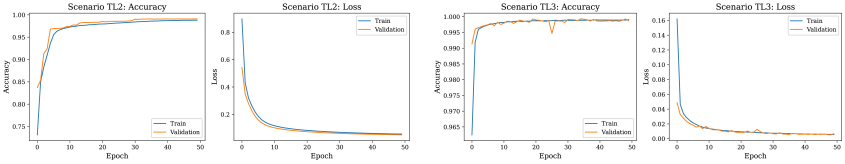
formed well (99.87%), maintaining a strong balance across all metrics.

Table 11 presents the performance metrics of the evaluated models on the BoT-IoT dataset using the TL3 selective layer fine-tuning strategy. Among the architectures tested, VGG16 attains the highest accuracy at 99.92%, followed closely by Xception with 99.91%. Both models also exhibit strong precision, recall, and F1-score values, reflecting their reliability in identifying different DDoS attack types while keeping false positives to a minimum.

These findings highlight the critical role of transfer learning in enhancing DDoS detection within resource-limited Edge IIoT systems. The full fine-tuning approach (TL0), which updates all layers, leads to notable improvements in detection accuracy, particularly for VGG16, though at the cost of increased computational demands. Conversely, TL3 provides substantial reductions in training time relative to TL0, TL1, and TL2, demonstrating its suitability for Edge-IIoT deployments where efficiency and low computational overhead are essential [2].



(a) TL0 **(b)** TL1



(c) TL2 **(d)** TL3

Figure 19: Training and validation performance of TransferEdge strategies (TL0–TL3) for DDoS attack detection: transfer from UNSW-NB15 (source) to BoT-IoT (target) [2].

4.4.2 Analysis of Training Dynamics

Figure 19 presents the training behavior of the CNN-based transfer learning pipeline as knowledge is transferred from the source to the target domain. Under the TL0 (Baseline Transfer Learning, no frozen layers) configuration, the model quickly fits the source-domain distribution, showing signs of overfitting, as evidenced by unstable validation loss (variations of approximately ± 0.02 around epoch 15), despite achieving rapid early convergence (roughly 99% accuracy by epoch 5).

Conversely, the TL2 (Differential Fine-Tuning) strategy provides a more favorable balance between learning stability and generalization. It sustains nearly 99% validation accuracy with limited deviation between training and validation curves (gap $< 2\%$, $\Delta_{\text{loss}} < 0.1$ beyond epoch 20), suggesting more reliable adaptation across domains. Although TL0 reaches almost perfect training accuracy (99.9%), its validation improvement follows a slower trajectory, indicating the progressive alignment of deeper hierarchical features.

Overall, the comparison underscores a key trade-off: TL2 achieves

a stable and computationally efficient adaptation process, while TL0 prioritizes maximal accuracy at the expense of higher computational demand. These findings highlight the importance of selective, layer-aware fine-tuning for maintaining domain-invariant features essential for cross-domain DDoS attack detection [2].

4.4.3 Adaptability Analysis

The transfer learning experiments highlight the capability of the proposed approach to enable CNN-based models to adjust to evolving operating conditions. The results show that a network trained on one dataset can be effectively adapted to another, often with relatively short fine-tuning times. In this study, we assessed the adaptability of six backbone architectures (CNN, VGG8, VGG16, ResNet18, EfficientNet, and Xception) across both transfer directions (UNSW-NB15 $\rightarrow$ BoT-IoT and BoT-IoT $\rightarrow$ UNSW-NB15). To further examine how well the methodology handles dynamic network environments, TransferEdge also performs a two-stage transfer learning procedure. In this setup, models are first trained on one dataset, fine-tuned on the second, and then fine-tuned once again on the original dataset (e.g., UNSW-NB15 $\rightarrow$ BoT-IoT $\rightarrow$ UNSW-NB15). This process is evaluated in both forward and reverse directions, and the results show that the accuracy remains consistent with that of CNNs trained directly on a single dataset. Overall, the findings indicate that EdgeTransfer's staged transfer mechanism effectively reduces domain shift, making it a robust approach for real-time DDoS detection in IIoT edge systems operating under rapidly changing network conditions [2].

4.4.4 Cross Domain Adaptation

Customized CNN model. In this study, the custom CNN architectures were trained using the Adam optimizer across several datasets selected to reflect a broad range of intrusion scenarios. The datasets include UNSW-NB15, KDDCup99, CSE-CIC-IDS2018, and CIC-DDoS2019. Each dataset contributes distinct characteristics: UNSW-NB15 provides re-

Table 12: Accuracy results of transfer learning source model for binary and multiclass classification.

Variants	Dataset	Accuracy		
		Conv4	Conv8	Conv18
Binary	CSE-CIC-IDS2018	**99.82**	99.81	99.73
	CIC-DDoS2019	99.90	**99.94**	99.88
	KDDcup99	99.42	**99.67**	99.59
	UNSW-NB15	**98.78**	98.71	98.35
Multi	CSE-CIC-IDS2018	**99.84**	99.82	96.98
	CIC-DDoS2019	**91.99**	91.84	91.76
	KDDcup99	95.12	95.95	**95.95**
	UNSW-NB15	97.51	97.55	**97.61**

alistic traffic behaviors, KDDCup99 offers a wide variety of intrusion patterns, CSE-CIC-IDS2018 contains diverse modern attack scenarios, and CIC-DDoS2019 captures detailed representations of DDoS activity. This variety enables a comprehensive evaluation of model robustness across heterogeneous network conditions and attack types [3].

For training, categorical cross-entropy was used for multiclass classification and binary cross-entropy for binary tasks. As summarized in Table 12, the Conv4 model achieved 99.90% accuracy, Conv8 reached 99.94%, and Conv18 attained 99.88% when distinguishing benign traffic from DDoS attacks in the CIC-DDoS2019 dataset. In multiclass settings, Conv4 and Conv8 obtained accuracies of 99.84% and 99.82%, respectively, on the CSE-CIC-IDS2018 dataset, while Conv18 achieved 97.61% on UNSW-NB15 [3].

Transferability and adaptability are assessed by training on one dataset and evaluating on multiple target datasets that represent distinct network environments. The target datasets used in this evaluation, CSE-CIC-IDS2018, CIC-DDoS2019, KDDCup'99, and UNSW-NB15, enable a comprehensive assessment of the models' adaptability across diverse network environments [3].

The source model, initially trained on the CIC-DDoS2019 dataset, exhibited strong generalization capability when transferred to various target datasets. In the binary classification setting, the Conv18 model

Table 13: Accuracy evaluation of models in detecting benign to attack traffic detection tasks transferred to various target datasets.

		Target dataset			
Dataset	**Model**	**CIC-DDoS2019**	**CSE-CIC-IDS2018**	**UNSW-NB15**	**KDDcup99**
CICDDoS 2019	Conv4		99.81	99.78	99.73
	Conv8		99.92	99.67	99.88
	Conv18		**99.99**	**99.92**	**99.94**
CSE-CIC-IDS2018	Conv4	99.67		99.32	99.48
	Conv8	99.81		99.46	99.74
	Conv18	**99.88**		**99.46**	**99.78**
UNSW-NB15	Conv4	98.82	99.21		98.24
	Conv8	**98.88**	99.23		**98.46**
	Conv18	98.73	**99.42**		98.33
KDD cup99	Conv4	96.25	**99.04**	95.24	
	Conv8	**97.02**	98.33	96.34	
	Conv18	96.66	98.13	**98.46**	

Table 14: Accuracy evaluation of models in multiclass attack detection tasks transferred to various target datasets.

		Target dataset			
Dataset	**Model**	**CIC-DDoS2019**	**CSE-CIC-IDS2018**	**UNSW-NB15**	**KDDcup99**
CIC-DDoS 2019	Conv4		99.83	98.04	96.98
	Conv8		99.91	98.23	98.76
	Conv18		**99.92**	**99.42**	**98.88**
CSE-CIC-IDS2018	Conv4	89.92		99.34	99.84
	Conv8	89.52		99.42	99.91
	Conv18	**93.62**		**99.84**	**99.96**
UNSW-NB15	Conv4	83.42	92.34		94.24
	Conv8	86.78	93.64		**94.46**
	Conv18	**88.92**	**93.86**		94.33
KDD cup99	Conv4	85.25	95.04	**96.84**	
	Conv8	85.54	**95.33**	96.04	
	Conv18	**85.66**	95.13	96.81	

fine-tuned on CSE-CIC-IDS2018 achieved an exceptionally high accuracy of 99.99% in distinguishing benign from DDoS traffic. Detailed results are reported in Table 13.

Overall, the results indicate that the proposed transfer learning strat-

egy maintains highly consistent behaviour across different source-to-target combinations. The small variations observed in binary classification highlight the flexibility and robustness of the approach. Importantly, the transferred models outperform models trained solely on individual datasets, demonstrating the advantages of leveraging source-domain knowledge. These findings confirm the effectiveness of the methodology in surpassing traditional single-dataset training baselines [3].

For multiclass attack-type detection, transferring Conv18 from CIC-DoS2019 to CSE-CIC-IDS2018 produced an accuracy of 99.92%, while the opposite transfer direction resulted in 93.62% performance, as shown in Table 14. When compared with prior experiments, the results consistently show that models transferred from CIC-DDoS2019 maintain high accuracy across both binary and multiclass tasks, suggesting that the richer feature space of the source dataset enhances adaptability to new target domains.

When transferring models from smaller and less feature-rich datasets to larger and more complex ones, declines in performance were observed for certain attack categories. For example, Conv4 achieved 83.42% accuracy when transferred from UNSW-NB15 to CIC-DDoS2019. This reduction is likely due to disparities in dataset scale, diversity, and feature distribution, which make adaptation to the more complex dataset more challenging. In contrast, transferring models from complex datasets to simpler ones typically improved accuracy. As an illustration, Conv18 trained on CIC-DDoS2019 achieved notably strong performance when transferred to KDDCup99.

These observations highlight that models trained on large, diverse datasets develop richer feature representations, enabling more effective knowledge transfer to smaller or less complex domains. This capability reduces the need for extensive labeling in the target domain while still achieving high predictive accuracy, making the approach particularly valuable in scenarios with limited annotated data [3].

Pre-trained models. In this set of experiments, a transfer learning strategy was adopted to exploit the feature extraction capabilities of ImageNet-pretrained CNN architectures, namely VGG16, VGG19, and ResNet50. The methodology converts network traffic records into image-based rep-

Table 15: Performance metrics on various datasets and VGG16, VGG19, and ResNet50 pre-trained models for binary classification.

Dataset	Model	Accuracy %	Recall	Precision	F1-score
CSE-CIC-IDS2018	VGG16	99.99	99.99	99.98	99.98
	VGG19	**100.00**	**100.00**	**100.00**	**100.00**
	ResNet50	99.98	99.98	99.78	99.82
CIC-DDoS 2019	VGG16	**99.99**	**99.98**	**99.99**	**99.98**
	VGG19	99.99	99.96	99.97	99.98
	ResNet50	99.94	99.96	99.98	99.97
KDD cup99	VGG16	99.69	**99.98**	**99.99**	99.98
	VGG19	**99.90**	99.96	99.97	**99.98**
	ResNet50	99.56	99.96	99.98	99.97
UNSW-NB15	VGG16	98.36	97.68	**98.70**	**98.86**
	VGG19	**98.64**	**98.68**	98.68	98.67
	ResNet50	98.60	98.62	98.67	98.65

resentations, as depicted in Figure 17. For the CSE-CIC-IDS2018 dataset, 41,883 images were utilized to perform binary classification between benign and malicious traffic. The study was then extended to multiclass detection, distinguishing several DDoS attack types in addition to benign flows, which required a substantially larger set of 269,616 images to adequately represent the increased number of classes and capture finer-grained variations across attacks. For the CIC-DDoS2019 dataset, 78,368 images were generated to cover 12 attack categories and benign traffic across training, validation, and test splits. Representative samples from this dataset are shown in Figure 18. Furthermore, the approach incorporated 12,154 images derived from the KDDCup'99 dataset and 5,629 images from UNSW-NB15.

The pretrained CNNs were fine-tuned for both binary (DDoS vs. benign) and multiclass (attack-type) classification tasks. A comprehensive hyperparameter tuning process was applied across all architectures, including the selection of the number of frozen layers during fine-tuning [3].

The results summarized in Table 15 highlight the strong binary classification performance of the VGG16, VGG19, and ResNet50 models in distinguishing benign traffic from DDoS attacks. For the CSE-CIC-IDS2018 dataset, VGG19 attains scores 100% accuracy, recall, precision, and F1-

Table 16: Performance metrics on various datasets and VGG16, VGG19, and ResNet50 pre-trained models for multi-class classification.

Dataset	Model	Accuracy %	Recall	Precision	F1-score
CSE-CIC-IDS2018	VGG16	99.21	99.21	99.21	99.21
	VGG19	**99.97**	**99.98**	**99.98**	**99.98**
	ResNet50	99.81	99.82	99.79	99.80
CIC-DDoS 2019	VGG16	92.19	92.56	92.30	92.28
	VGG19	**92.65**	**92.29**	**92.24**	**92.91**
	ResNet50	91.71	91.21	91.82	91.03
KDD cup99	VGG16	97.46	97.23	97.49	97.98
	VGG19	**98.99**	**98.96**	98.97	**98.98**
	ResNet50	98.94	98.96	**98.98**	98.97
UNSW-NB15	VGG16	97.58	97.92	**98.68**	**98.42**
	VGG19	**97.59**	97.63	97.63	97.63
	ResNet50	97.36	97.23	97.04	97.64

score. On the CIC-DDoS2019 dataset, all three architectures achieve similarly high performance, with VGG16 and VGG19 reaching 99.99% accuracy and ResNet50 achieving 99.94%. Across the KDDCup99 and UNSW-NB15 datasets, VGG19 again delivers the best results, recording accuracies of 99.90% and 98.64%, respectively. These outcomes demonstrate the superior effectiveness of VGG19 in binary DDoS detection, underscoring its reliability in accurately separating benign and malicious traffic across diverse network environments [3].

The performance metrics reported in Table 16 offer a detailed assessment of the adaptive pre-trained models VGG16, VGG19, and ResNet50 applied to multiclass classification across several benchmark datasets. Among these architectures, VGG19 consistently achieves the highest accuracy. On the CSE-CIC-IDS2018 dataset, VGG19 attains 99.97% accuracy. For CIC-DDoS2019, it again leads with an accuracy of 92.65%. In the KDD-Cup99 dataset, VGG19 reaches 98.99%, marginally surpassing ResNet50 at 98.94%. Likewise, on UNSW-NB15, VGG19 maintains strong performance with an accuracy of 97.59%. These results highlight VGG19's superior adaptability and effectiveness in handling complex multiclass intrusion detection tasks [3].

Overall, VGG19 demonstrates the most consistent performance across

Table 17: Comparison of proposed approach metrics with state-of-the-art methods by dataset and class variants.

Dataset	variants	References	Model	Accuracy (%)
CIC-DDoS2019	Binary	[53]	CNN	99.87
		Our model	VGG19	99.99
	Multi-class	[162]	DNN	81.77
		[42]	CNN	77.29
		Our model	Conv18	93.62
CSE-CIC-IDS2018	Binary	[163]	VGG16	98.8
		Our model	VGG19	100.00
	Multi-class	[42]	CNN	99.88
		Our model	Conv18	99.92
KDDcup99	Binary	[164]	CNN-LSTM	97.4
		Our model	ResNet50	99.32
	Multi-class	[153]	TL-ConvNet	93.86
		Our model	VGG19	98.99
UNSW-NB15	Binary	[164]	CNN-LSTM	94.43
		Our model	Conv18	99.92
	Multi-class	[165]	DBN	96.34
		Our model	Conv18	99.84

all datasets, indicating its strong capacity to capture and generalize intricate feature patterns. While ResNet50 performs competitively, particularly on KDDCup99 and UNSW-NB15, these variations emphasize the importance of selecting model architectures that align with dataset characteristics. The consistently strong outcomes of VGG19 underline its robustness for multiclass classification in cybersecurity applications [3].

In the binary classification setting, the transferred VGG19 model on the CSE-CIC-IDS2018 dataset surpasses traditional deep learning approaches. However, for multiclass tasks, customized transferred CNNs, Conv18, show clear advantages. These observations further demonstrate the significant impact of transfer learning on enhancing model performance, particularly for IDS and DDoS attack detection.

To assess the effectiveness of the proposed model, we conducted a comprehensive comparison against state-of-the-art DL and transfer learning approaches evaluated on similar datasets. The Conv18 architecture achieved 99.92% accuracy in identifying network attacks, outperforming the VGG16-based IDS reported in [163], which attained 98.8% on the CSE-CIC-IDS2018 dataset. Moreover, the fine-tuned VGG19 model

reached 100% accuracy in separating benign from DDoS traffic on the same dataset.

For the CIC-DDoS2019 dataset, prior works [42,162] reported accuracy levels of 77.29% and 81.77%, respectively, for attack-type classification. In contrast, our model achieved a substantially higher accuracy of 93.62%. Similarly, [153] presented the TL-ConvNet model with a 93.86% accuracy on KDDCup99, whereas the adaptive VGG19 variant developed in this study attained 98.99%. For UNSW-NB15, the Deep Belief Network (DBN) approach proposed by [165] reached 96.34% accuracy, while our Conv18 model achieved 99.84% in attack-type detection.

As summarized in Table 17, these results demonstrate that the proposed method consistently surpasses existing solutions, particularly in DDoS detection and fine-grained attack classification, confirming the strength of the adaptive transfer learning strategies employed.

4.5 Summary

This chapter presents **TransferEdge**, a transfer learning and fine-tuning framework specifically designed for resource-constrained Edge-IIoT devices. By leveraging pre-trained models with dataset-specific fine-tuning strategies, TransferEdge mitigates domain shift and label scarcity while improving detection quality. Cross-domain experiments-*UNSW-NB15* $\rightarrow$ *BoT-IoT* show that TransferEdge-enhanced architectures (*CNN*, *VGG8/16/19*, *ResNet18*, *EfficientNet*, *Xception*, and a custom *Conv18*) more reliably identify novel attack patterns than vanilla baselines; in binary classification, *VGG19* cleanly separates benign from malicious traffic, while for multiclass or the attack-type recognition the custom *Conv18* achieves higher accuracy, precision, recall, and F1. Additional transfers (from *CIC-DDoS2019* to other datasets) consistently maintain high accuracy in both binary and multiclass settings, with gains increasing as feature dimensionality grows, underscoring the effectiveness and scalability of transfer learning in strengthening industrial IoT cybersecurity under limited labels and changing traffic distributions. Next, move to federated learning, modeling stronger adversaries (poisoning, backdoor, and evasion) and hardening

training with anomaly screening, robust aggregation, and cryptographic safeguards for privacy-preserving, large-scale deployment.

Chapter 5

Federated Learning With Adaptive Client Selection

5.1 Introduction

IoT technologies today encompass a wide range of interconnected devices, from personal healthcare systems and domestic automation platforms to industrial infrastructures and smart-city deployments. As these systems increasingly mediate sensitive personal and operational data, their exposure to cybersecurity risks becomes a significant concern. Among the various threats targeting IoT environments, DDoS attacks remain particularly disruptive, as they degrade service availability by overwhelming constrained networked resources [166].

Traditional IDs based on ML and DL have been explored extensively for mitigating such attacks [167, 168]. However, these centralized approaches struggle to scale in distributed IoT ecosystems and often conflict with privacy constraints that prevent organizations from sharing raw traffic data. FL has therefore emerged as an appealing paradigm

This chapter is based on the published article: M.B. Anley, P. Coscia, A. Genovese, and V. Piuri, "FELACS: Federated Learning with Adaptive Client Selection for IoT DDoS Attack Detection," *Computers & Security*, 2025, Article 104642. Licensed under CC BY. See: `https://creativecommons.org/licenses/by/4.0/`. Reused content complies with the license terms.

for collaborative threat detection, enabling joint model training without transferring sensitive information among participating devices or organizations [105, 169–171].

In a standard FL workflow, the server dispatches a global model to a set of selected clients, which then train locally and return their updates for aggregation. A crucial determinant of model performance and convergence is the mechanism used to select clients for each training round. In IoT environments, this process becomes particularly challenging due to the pronounced heterogeneity among devices. Clients may generate datasets of varying sizes and feature distributions, often capturing different attack types with highly imbalanced representations. Device-specific operational conditions further contribute to statistical non-uniformity, complicating both learning and aggregation [172].

Most existing FL approaches rely on random or semi-random client selection [105, 108, 109]. While simple and lightweight, such strategies fail to consider the distinct value or relevance of the data available at different clients, potentially resulting in uninformative updates and slower global convergence [173]. More recent approaches incorporate device-level constraints that are available computing power, energy budget, or expected response time to improve the reliability of client participation [38, 104, 107, 111, 112, 114, 174]. Although effective in addressing hardware heterogeneity, these methods often treat data importance and system conditions as static, even though IoT environments are characterized by dynamic workloads, changing attack distributions, and intermittent connectivity [175].

Collectively, these limitations highlight the need for a client-selection strategy that can jointly account for data relevance, device capabilities, and communication efficiency dynamically and adaptively. Addressing this gap, this chapter develops the *FELACS* framework introduced in our earlier work [4]. FELACS formulates client selection as a multiobjective decision process that integrates statistical indicators of data quality (entropy, variance), device-side resource metrics (latency, computation time, energy consumption), and participation stability. By synthesizing these criteria, FELACS identifies the subset of clients expected to yield the most

beneficial contributions to the global model at each communication round. The implementation used for this thesis is publicly available.[1]

Beyond resource and efficiency considerations, FELACS is tailored to the operational characteristics of IoT-based DDoS detection. In such settings, attack patterns often evolve rapidly, and the ability to prioritize clients holding informative or attack-relevant data is essential for timely detection. FELACS therefore adopts a data-centric view of client selection, promoting participants whose local observations are most likely to enhance model robustness. This leads to faster convergence, improved accuracy, and enhanced responsiveness under real-time threat conditions. By shifting away from static or resource-only selection heuristics, FELACS offers a comprehensive and adaptive strategy aligned with the heterogeneous and dynamic nature of IoT environments.

5.2 Methodology

5.2.1 System Model and Problem Formulation

This section presents FELACS, an adaptive client-selection algorithm developed to improve federated learning–based DDoS detection in IoT environments. We first describe the overall system model and subsequently formalize the client-selection problem. For clarity, Table 18 provides a summary of the notation and symbols used throughout the mathematical formulation.

5.2.2 System Model

The FELACS framework operates within an FL setting to support DDoS detection across IoT networks. The system is composed of a central coordinating server and a set of distributed clients $\mathcal{N}$, where each client $k \in \mathcal{N}$ holds its own local dataset $\mathcal{D}_k$ derived from the traffic observed within its private IoT domain.

The server maintains the global model parameters $\mathbf{w}_t$, which represent the classifier for legitimate and malicious (DDoS) traffic during round t.

[1]Source code: `https://github.com/mulerkal/FELACS`

Table 18: Notations and descriptions used throughout the paper [4].

Symbol	Description
$\mathcal{N}$	Set of all clients participating in FL.
k	Client index, where $k \in \mathcal{N}$.
$\mathcal{D}_k$	Local dataset held by client k.
t	Communication round index.
$S_t \subseteq \mathcal{N}$	Subset of clients selected in round t.
$\mathbf{w_t}$	Global model in round t.
$\mathbf{w_k^t}$	Updated local model parameters after client k trains on $\mathcal{D}_k$ in round t.
$\nabla F_k(\mathbf{w_t})$	Gradient of the loss function F_k of client k evaluated at $\mathbf{w_t}$.
η	Learning rate for local stochastic gradient descent (SGD) updates.
K	Upper bound imposed on the number of clients selected per round.
$R_{\max}$	Total resource budget available for client selection.
$L_{\max}$	Maximum allowable communication latency per client.
$H_{\min}$	Minimum Shannon entropy required for the data of a client to qualify.
$\alpha, \beta, \gamma, \delta$	Hyperparameters for weighting the components of the importance score.
I_i	Importance score of client i, defined as $I_i = \alpha\, H(\mathcal{D}_i) + \beta\, V(\mathcal{D}_i) - \gamma\, P_i - \delta\, L_i$.
E	Number of local training epochs executed on each selected client per round.
y_i^c	One-hot encoded true label of sample i for class c.
$f(x_i; \mathbf{w_t})_c$	Predicted probability of class c for input x_i under model $\mathbf{w_t}$.
C	Total number of classes involved in the classification task.
n_k	Number of data points contained in $\mathcal{D}_k$.
$n \sum_{k \in S_t} n_k$	Total number of samples across all selected clients in round t.
R_i	Resource consumption (e.g., CPU cycles) of client i.
P_i	Power consumption of client i.
L_i	Network latency of client i.
u_i	Utility score of client i.
$H(\mathcal{D}_i)$	Shannon entropy of the data distribution at client i.
$V(\mathcal{D}_i)$	Variance of the data distribution of client i.
T_{dl}	$\frac{\text{model size}}{\text{downlink bandwidth}} + \text{latency}$.
$T_{\text{comp}}^{(i)}$	$E \cdot \frac{N_i}{\text{batch size}} \cdot t_{\text{batch}}^{(i)}$.
$T_{\text{ul}}^{(i)}$	$\frac{\text{update size}}{\text{uplink bandwidth}} + \text{latency}$.
T_{agg}	Aggregation time per round.

At every communication round, the server distributes the current global model to the participating clients, who then conduct local training on their datasets and submit their parameter updates back to the server.

After an initial setup phase, the FL procedure evolves through a sequence of communication rounds. Each round consists of three main stages:

1. selection of the clients that will participate in the round,

2. local computation and model updating performed by the selected clients, and

3. aggregation of the received updates to refine the global model.

These iterations continue until reaching the maximum number of rounds, and achieving convergence of the global model is satisfied.

Step 1: Client Selection. In each round t, the server assesses the capabilities of each device (i.e., its network latency, power consumption, and computational resources) and uses the proposed FELACS algorithm to select a subset of clients $S_t \subseteq \mathcal{N}$ from the total pool of clients $\mathcal{N}$, with $|\mathcal{N}| = K$ total clients. This step is explained in detail in Section 5.2.3.

Step 2: Client Updating. The selected clients download the current global model $\mathbf{w}_t$. Each client $k \in S_t$ trains the model on its local dataset $\mathcal{D}_k$ for E epochs and computes the updated local model parameters $\mathbf{w}_t^k$. The update process is as follows (Equation 5.1) :

$$\mathbf{w}_t^k = \mathbf{w}_t - \eta \nabla F_k(\mathbf{w}_t) \quad , \tag{5.1}$$

where η is the learning rate and $\nabla F_k(\mathbf{w}_t)$ represents the gradient of the local loss function $F_k(\mathbf{w}_t)$, which quantifies the error induced by the model on the local dataset of the client $\mathcal{D}_k$. We define the local loss function $F_k(\mathbf{w}_t)$ as the cross-entropy loss (Equation 5.2):

$$F_k(\mathbf{w}_t) = -\frac{1}{|\mathcal{D}_k|} \sum_{i \in \mathcal{D}_k} \sum_{c=1}^{C} y_i^c \log f(x_i; \mathbf{w}_t)^c \quad , \tag{5.2}$$

where y_i^c is the one-hot encoded label produced for class c, $f(x_i; \mathbf{w}_t)^c$ is the predicted probability for class c, and C is the total number of classes.

Step 3: Model Aggregation. The server aggregates the local updates obtained from the selected clients to update the global model. The server-side steps involve collecting updates $\mathbf{w}_t^k$ from each selected client $k \in S_t$ and aggregating the updates to form a new global model, thereby obtaining $\mathbf{w}_{t+1}$:

$$\mathbf{w}_{t+1} = \sum_{k \in S_t} \frac{n_k}{n} \mathbf{w}_t^k \quad , \tag{5.3}$$

where n_k is the number of data points for client k and $n = \sum_{k \in S_t} n_k$. By aggregating these updates from the selected clients, the global model optimizes the overall objective:

$$F(\mathbf{w}_t) = \frac{1}{K} \sum_{k=1}^{K} F_k(\mathbf{w}_t) \quad , \tag{5.4}$$

where K is the total number of clients. The system model is detailed in Algorithm 1.

5.2.3 FELACS Problem Formulation

The overarching aim of FELACS is to accelerate model convergence and enhance the accuracy of DDoS attack detection by identifying, at each communication round t, an optimal subset of participating clients S_t. The selection mechanism is formulated as an optimization problem that seeks to maximize the resulting detection performance while satisfying the computational and communication constraints inherent to federated IoT environments. The decision process is governed by the aggregate importance score I_i associated with each candidate client, which determines its suitability for contributing to the global update.

The **importance score** for client i is defined as follows in Equation 5.5:

$$I_i = \alpha H(\mathcal{D}_i) + \beta V(\mathcal{D}_i) - \gamma P_i - \delta L_i \quad , \tag{5.5}$$

where $H(\mathcal{D}_i)$ is the Shannon entropy of the data of client i, representing

Algorithm 1: FELACS System Model

Input: Client pool $\mathcal{N}$, total number of rounds T, maximum number of clients per round K, number of local epochs E, learning rate η.
Output: Global model $\mathbf{w}_t$.

Initialization.
$\mathbf{w}_0 \leftarrow$ initial model parameters

for $t \leftarrow 0$ **to** $T-1$ **do**

Step 1: Client Selection.
$S_t \leftarrow SelectClients(\mathcal{N}, K)$(see Algorithm 2)

Step 2: Client Updating.
foreach $k \in S_t$ ***in parallel*** **do**
Client k downloads $\mathbf{w}_t$ and trains for E epochs on $\mathcal{D}_k$
$\mathbf{w}_t^k \leftarrow \mathbf{w}_t - \eta \nabla F_k(\mathbf{w}_t)$

Step 3: Model Aggregation.
$n_k =$ number of data points for client k
$$\mathbf{w}_{t+1} = \sum_{k \in S_t} \frac{n_k}{\sum_{j \in S_t} n_j} \mathbf{w}_t^k$$

the diversity of the data; $V(\mathcal{D}_i)$ is the variance of the local data of client i, which reflects the statistical heterogeneity of the data distribution of this client; P_i represents the power consumption of client i; L_i represents the network latency of client i; and $\alpha, \beta, \gamma, \delta$ are hyperparameters for controlling the relative influence of each factor (see also Table 18).

The proposed framework interprets the task of determining an optimal client subset S_t at each communication round t as a multi-objective optimization problem, instantiated as a multidimensional knapsack problem (MDKP). Within this formulation, every participating device i is associated with a cost vector capturing its resource consumption, energy usage, and communication latency, as well as a utility value quantifying its data relevance and anticipated contribution to the global model update. The goal is to maximize the aggregate utility of the selected clients while ensuring that the cumulative cost remains within the prescribed system budget.

FELACS formulate the optimization problem to maximize the total

importance score of the selected clients while respecting the imposed resource, power, latency, data diversity, and participation constraints (Equation 5.6):

$$\underset{S_t}{\text{argmax:}} \sum_{i \in S_t} (I_i) \tag{5.6}$$

$$\text{subject to:} \sum_{i \in S_t} R_i \leq R_{\text{max}}, \tag{5.7}$$

$$\sum_{i \in S_t} P_i \leq P_{\text{max}}, \tag{5.8}$$

$$L_i \leq L_{\text{max}}, \ \forall i \in S_t, \tag{5.9}$$

$$H(\mathcal{D}_i) \geq H_{\text{min}}, \ \forall i \in S_t, \tag{5.10}$$

$$|S_t| \leq K. \tag{5.11}$$

The optimization problem must satisfy the following constraints.

1. *Client resource constraints*. The total resource usage $\sum_{i \in S_t} R_i$ of the selected clients must not exceed the available resource budget R_{max}. This constraint helps manage the computational and communication limitations of IoT devices, ensuring an efficient client selection process without overburdening the system (Equation 5.7).

2. *Power constraint.* The total power consumption level $\sum_{i \in S_t} P_i$ for the selected clients should be bounded by P_{max} to ensure that battery-powered devices retain sufficient energy for completing their training procedures (Equation 5.8).

3. *Latency constraints*. The latency L_i of each selected client should be bounded by L_{max} to ensure timely model updates (Equation 5.9).

4. *Data diversity constraints*. To ensure diversity, clients with high data redundancy levels are excluded. The entropy of the data of the selected clients $H(\mathcal{D}_i)$ must exceed a threshold H_{min} (Equation 5.10).

5. *Client selection constraints*. The number of selected clients $|S_t|$ should not exceed a predefined limit K (Equation 5.11).

The defined multidimensional client selection problem is a nondeterministic polynomial-time (NP)-hard problem, so FELACS adopts a fast three-step greedy heuristic approach. *i)* First, discard any client i for which $L_i > L_{\max}$ or $H(\mathcal{D}_i) < H_{\min}$, thereby enforcing the latency and diversity constraints up front. For each remaining client, then compute the importance-per-resource ratio ρ_i as shown below (5.12):

$$\rho_i = \frac{I_i}{R_i} \quad , \tag{5.12}$$

which quantifies the marginal benefit of selecting i relative to its cost. The importance-per-resource ratio balances its potential benefit against its resource consumption level. *ii)* Then, FELACS sort the clients in descending order of ρ_i (an $\mathcal{O}(n \log n)$ operation). *iii)* Finally, FELACS scans the sorted list of clients in descending order of ρ_i, incrementally adding clients to S_t until the inclusion of an additional client would breach at least one constraint (this pass costs $\mathcal{O}(n)$). Whenever a violation occurs, a repair procedure is invoked: the offending client is removed, and the algorithm proceeds down the ρ_i ranking to identify the next most suitable candidate whose inclusion satisfies all constraints.

Algorithm 2 details the proposed client selection algorithm. FELACS employs a three-stage greedy heuristic with a computational complexity of $\mathcal{O}(n \log n)$, linear memory usage, and minimal computation and communication overhead—properties that make it particularly well-suited for IoT environments where devices are resource-limited, and connectivity is intermittent. Alternative methods for addressing the MDKP problem include integer linear programming (ILP) formulations, LP-relaxation techniques followed by rounding [176,177], population-based metaheuristics [178], and multiobjective optimization strategies designed for FL settings [179]. Although such methods may provide stronger optimality guarantees or improved approximation performance, they often suffer from exponential increases in runtime, require extensive parameter tuning, or incur high communication and memory costs. These drawbacks translate into considerable latency and energy usage, making them impractical for deployment in constrained and intermittently connected IoT

Algorithm 2: FELACS Algorithm for Client Selection

Input: Client pool $\mathcal{N}$, maximum number of clients K, resource budget $R_{\max}$, latency $L_{\max}$, entropy threshold $H_{\min}$, weights $(\alpha, \beta, \gamma, \delta)$.
Output: Selected client subset S_t.

Step 1.
Compute importance score I_i.
foreach $i \in \mathcal{N}$ **do**
$I_i \leftarrow \alpha\, H(\mathcal{D}_i) + \beta\, V(\mathcal{D}_i) - \gamma\, P_i - \delta\, L_i$
Compute importance-per-resource ratio ρ_i.
foreach $i \in \mathcal{N}$ **do**
$\rho_i \leftarrow \frac{I_i}{R_i}$

Step 2: Sort clients.
Sort $\mathcal{N}$ in descending order of ρ_i

Step 3.
Client selection under constraints.
$S_t \leftarrow \emptyset$
foreach i *in sorted* $\mathcal{N}$ **do**
if $|S_t| < K$ ***and*** $\sum_{j \in S_t} R_j + R_i \leq R_{\max}$ ***and*** $L_i \leq L_{\max}$ ***and*** $H(\mathcal{D}_i) \geq H_{\min}$ **then**
$S_t \leftarrow S_t \cup \{i\}$
Repair constraint violations.
foreach $j \in S_t$ ***if*** $(L_j > L_{\max}$ ***or*** $H(\mathcal{D}_j) < H_{\min})$ **do**
Find next-best $k \notin S_t$ with the highest I_k satisfying all constraints
Replace j with k in S_t
return S_t

systems [4].

In this chapter, each client k retains its local dataset $\mathcal{S}_k$ and performs training locally, with raw samples (and any underlying packet traces) never leaving the client. The server acts as the coordinator/defender and observes only the information exchanged through the FL protocol, i.e., the received client updates and the resulting aggregated global model, without direct access to client-local data. The training proceeds for 100 communication rounds using the hyperparameters specified in the subsequent subsection.

5.2.4 Experimental Design

The Flower[2] FL framework is employed for simulations due to its versatility and flexible customization suited to IoT and client–server set-

[2] `https://flower.ai`

Table 19: Traffic types and their cardinality across four datasets.

CIC-IDS2018		CIC-DDoS2019		BoT-IoT		CIC-IoT2023	
Type	**Card.**	**Type**	**Card.**	**Type**	**Card.**	**Type**	**Card.**
Benign	1 603 315	TFTP	4 396 770	Benign	590 298	DDoS	5 338 243
LoIC-HTTP	143 746	UDP	2 510 574	DDoS	379 302	DoS	1 269 264
HOIC	49 677	NTP	1 112 902	Reconn	108 795	Mirai	413 754
Hulk	36 227	SSDP	891 220	Theft	31 746	Benign	172 642
SlowHTTPTest	10 308	SYN	687 524	Scan	9 164	Spoofing	76 807
GoldenEye	2 419	MSSQL	484 070	–	-	Recon	55 531
Slowloris	433	SNMP	114 179	–	–	Web	3 836
LOIC-UDP	154	DNS	113 252	–	–	BruteForce	1 988
–	–	BENIGN	99 154	–	–	–	–
–	–	LDAP	48 469	–	–	–	–
–	–	NetBIOS	31 052	–	–	–	–
–	–	Portmap	1 638	–	–	–	–
Total: 1 846 279		**Total:** 10 491 218		**Total:** 1 119 305		**Total:** 7 332 065	

tings [180].

5.2.5 Datasets

Experiments were conducted on widely used benchmark datasets for intrusion and DDoS detection: CIC-IDS2018 [135], CIC-DDoS2019 [136], BoT-IoT [138], and CICIoT2023 [137]. These datasets enabled both binary and multiclass classification and supported FL model training. Moreover, BoT-IoT and CIC-IoT2023 describe realistic IoT traffic in modern networks. Table 19 lists the different datasets in terms of their classes and numbers of samples.

To emulate an FL setting for datasets that lack an inherent client-wise partitioning, this chapter adopts a simulated FL environment in which FELACS distributes preprocessed data across five clients via random sampling. This procedure introduces non-IID characteristics into the data. Table 20 presents an example using the CIC-IDS2018 dataset, where each client receives a different number of samples per class, thereby reflecting a realistic heterogeneous distribution commonly observed in IoT deployments. This configuration enables us to assess the model's ability to operate under non-IID conditions. The subsequent subsection offers a concise overview of the datasets and their key properties, as

Table 20: Data distribution of the CIC-IDS2018 dataset among five clients after performing preprocessing to simulate non-IID data.

Traffic Type	IoT1	IoT2	IoT3	IoT4	IoT5
Benign	269 431	351 565	503 600	248 003	230 716
LOIC-HTTP	24 155	31 519	45 150	22 234	20 688
HOIC	8 347	10 891	15 602	7 683	7 154
Hulk	6 087	7 943	11 378	5 602	5 217
SlowHTTPTest	1 731	2 260	3 237	1 593	1 487
GoldenEye	406	529	759	373	352
Slowloris	71	94	135	66	67
LOIC-UDP	29	41	17	31	36
Total	310 230	404 804	579 865	285 555	265 687

previously discussed in our FELACS paper [4].

CIC-IDS2018. This dataset was created to emulate realistic network activity[3], capturing multiple attack scenarios across five days, including DoS, DDoS, brute-force, and infiltration attempts. Each entry consists of 80 features generated by CICFlowMeter, providing information such as timestamps, source and destination IP addresses, protocol types, and packet length statistics [135].

CIC-DDoS2019. Developed by the Canadian Institute for Cybersecurity, this dataset provides a comprehensive benchmark for DDoS detection. It combines benign background traffic generated using the B-Profile system to simulate 25 users across services such as HTTP, HTTPS, FTP, SSH, and email—with a wide variety of reflection and amplification attack scenarios[4]. The training portion contains 12 different attack types, while the test set includes 7 variants. All network traffic was collected as raw PCAP files for each host and subsequently processed using CICFlowMeter to extract more than 80 flow-based features, resulting in tens of millions of labeled flows suitable for both binary and multiclass DDoS detection research [136], [4].

BoT-IoT. The BoT-IoT dataset was generated at the Cyber Range Lab of the University of New South Wales (UNSW) Canberra and captures both

[3] `https://www.unb.ca/cic/datasets/ids-2018.html`
[4] `https://www.unb.ca/cic/datasets/ddos-2019.html`

Table 21: Data distribution of the BoT-IoT dataset.

Traffic type	IoT1	IoT2	IoT3	IoT4	IoT5
Benign	295 649	148 000	73 000	36 500	37 149
DDoS	189 651	94 825	47 413	23 706	23 707
Reconn	54 395	27 197	13 599	6 800	6 804
Theft	15 873	7 937	3 968	1 984	1 984
Scan	4 582	2 291	1 145	572	574
Total	560 150	280 250	139 125	69 562	70 218

benign and malicious DDoS traffic within a realistic network testbed[5]. The traffic originates from several IoT devices, including smart refrigerators, audio systems, thermostats, lighting units, and door locks labeled as IoT1 through IoT5, as shown in Table 21. Each device emulates distinct network conditions and attack behaviors, enabling a thorough assessment of the proposed method in a federated learning context. The dataset includes a range of cyberattacks that include DDoS, DoS, reconnaissance, service scanning, and data theft [138].

CIC-IoT2023. The CIC-IoT2023 dataset provides large-scale IoT attack traffic collected from a realistic testbed comprising 105 heterogeneous devices. It includes 33 attack types grouped into seven categories DDoS, DoS, reconnaissance, web-based exploits, brute-force attempts, spoofing activities, and Mirai botnet command attacks alongside benign traffic[6] [137]. For this paper, each IoT device is treated as a client in FL experiments to simulate a realistic IoT environment, leveraging this dataset to evaluate the effectiveness of the proposed adaptive client selection approach.is widely recognized for its realistic portrayal of IoT network traffic

5.2.6 Data Distributions

In FL, the data distribution across clients can be either independent and identically distributed (i.i.d.) or non-i.i.d., depending on whether clients' local datasets follow the same underlying distribution.

[5] `https://research.unsw.edu.au/projects/bot-iot-dataset`
[6] `https://www.unb.ca/cic/datasets/iotdataset-2023.html`

i.i.d. vs. non-i.i.d. Let client $k \in \{1, \ldots, K\}$ hold a local dataset $\mathcal{S}_k$ sampled from an unknown distribution $\mathcal{D}_k$. The setting is *i.i.d.* when $\mathcal{D}_1 = \cdots = \mathcal{D}_K$ (approximately equal in practice), whereas it is *non-i.i.d.* when at least one client distribution differs, i.e., $\exists\, k \neq j$ such that $\mathcal{D}_k \neq \mathcal{D}_j$. In realistic IoT deployments, non-i.i.d. data commonly arises due to device heterogeneity, user behavior, and temporal or contextual variations in traffic.

For each dataset, we randomly split samples into 80% training and 20% centralized testing. We partition the training set across K federated clients so that each client retains a distinct local subset representing its own traffic characteristics. We sample the 20% test set once, keep it fixed across all experiments, and ensure it remains disjoint from all client-local partitions to support consistent and fair evaluation of the learned global models [4].

In this emulation, each client stores its partition locally and performs training on flow/window-level feature vectors derived from the underlying benchmark traffic; raw PCAP traces are not exchanged. The server coordinates training by selecting a subset of clients per round, broadcasting the current global model, and collecting only model updates (or weights/gradients) from participating clients. After aggregation, the server evaluates the updated global model on the fixed centralized test set at the end of each round and records accuracy, precision, recall, and F1-score to quantify per-round improvements and convergence behavior.

The server aggregates client contributions at the round level using sample-size weighting, i.e., each client update is weighted by its number of local training samples to avoid over-representing small clients. To model non-i.i.d. label heterogeneity, we generate client-wise class proportions using Dirichlet sampling $\mathrm{Dir}(\alpha)$, a standard approach in federated learning [181]. For each class c, we draw proportions $(p_{1,c}, \ldots, p_{K,c}) \sim \mathrm{Dir}(\alpha)$ and allocate approximately $p_{k,c}\, N_{\mathrm{train},c}$ samples of class c to client k, where $N_{\mathrm{train},c}$ denotes the number of training samples in class c. The concentration parameter α controls the degree of skew: we consider *mild* heterogeneity ($\alpha = 10$, close to i.i.d.), *moderate* heterogeneity ($\alpha = 1$), and *severe* heterogeneity ($\alpha = 0.1$), where some clients may observe almost

Table 22: Proposed 1D-CNN architecture of the model.

Layer	Filters/Units	Output Shape	Activation	Parameters
Input window	–	(|input|)	–	0
Conv1D	64	(254, 64)	Sigmoid	256
MaxPool1D	–	(127, 64)	–	0
Dropout	–	(127, 64)	–	0
Conv1D	128	(125, 128)	ReLU	24 704
MaxPool1D	–	(62, 128)	–	0
Dropout	–	(62, 128)	–	0
Conv1D	256	(60, 256)	ReLU	98 560
MaxPool1D	–	(30, 256)	–	0
Dropout	–	(30, 256)	–	0
Flatten	–	(7 680)	–	0
Dense	256	(256)	ReLU	983 168
Dropout	–	(256)	–	0
Dense (output)		(|output|)	Softmax	256

exclusively benign or attack traffic [4]. Since non-i.i.d. partitions can bias local updates toward client-specific distributions, the client selection strategy in this chapter aims to improve representativeness and stabilize aggregation under heterogeneous IoT conditions.

5.2.7 Model Architecture

The CNN model employed in the FELACS experiments is tailored for DDoS detection in IoT environments. It consists of two consecutive Conv1D layers, each followed by a ReLU activation and batch normalization to improve feature stability. Max-pooling layers are inserted between convolutional blocks to progressively reduce spatial dimensionality, while dropout layers are incorporated to limit overfitting. The resulting feature maps are flattened and fed into a fully connected layer with 256 units, followed by a final Softmax layer for multiclass output (see Table 22). This architecture is selected for its effectiveness in learning hierarchical representations from network traffic, enabling accurate discrimination between benign and malicious flows in IoT systems [3].

For comparative evaluation, we also incorporate the VGG16 architecture 16-layer convolutional network consisting of five convolutional blocks, each formed by sequential 3×3 convolutions followed by max-

pooling, and three fully connected layers at the top. Owing to its depth, VGG16 performs more expressive hierarchical feature extraction than the custom CNN model, and has demonstrated strong performance in DDoS detection tasks [3, 42]. In our experiments, FELACS is combined with both the shallow CNN and the VGG16 architecture, referred to as FELACS+CNN and FELACS+VGG16, respectively, to examine how architectural depth influences convergence behavior and classification accuracy under heterogeneous IoT traffic conditions.

During model evaluation, early stopping is applied at each local client to terminate training once performance ceases to improve. Within the federated learning framework, which varies the number of communication rounds to study update dynamics, the set of participating clients in each round is determined by the FELACS selection mechanism. The learning rate is optimized to ensure stable and efficient convergence, and a dropout rate of 20% is introduced to enhance robustness against noisy or incomplete IoT data. Binary cross-entropy is employed for distinguishing benign from DDoS traffic, while categorical cross-entropy is used for multiclass prediction of specific DDoS attack types.

5.2.8 Evaluation Metrics

To evaluate the performance of the FL models, we employ the following metrics that *i)* accuracy, representing the proportion of correctly classified samples relative to the total number of samples. *ii)* precision, defined as the ratio between true positives and all samples predicted as positive; *iii)* recall, given by the ratio of true positives to the total number of actual positive instances; *iv)* the F1 score, computed as the harmonic mean of precision and recall; and *v)* training time, measured as the overall duration required to complete the training process, including both communication and computation phases (Equation 5.13).

We define the total training time T_{total} as the sum of the per-round computation and communication latencies over all R federated rounds:

$$T_{\text{total}} = \sum_{r=1}^{R} \left[T_{\text{dl}} + \max_{i \in \mathcal{S}} \left(T_{\text{comp}}^{(i)} + T_{\text{ul}}^{(i)} \right) + T_{\text{agg}} \right] \quad . \tag{5.13}$$

In each federated round, client i performs local computations on its own data in time $T_{\text{comp}}^{(i)} \left(E \times \frac{N_i}{\text{batch size}} \times t_{\text{batch}}^{(i)} \right)$. The server first incurs a download delay of T_{dl} when the global model is sent to clients (server(client). After the local training process, client i uploads its update in time $T_{\text{ul}}^{(i)}$ (client(server).
Since the server waits for the slowest client, the combined communication and computation cost in that round is $\max_{i \in \mathcal{S}} \left(T_{\text{comp}}^{(i)} + T_{\text{ul}}^{(i)} \right) + T_{\text{dl}}$. Once all the updates arrive, the server aggregates them in time T_{agg}.

5.3 Experimental Results and Discussion

This section includes an evaluation of the proposed adaptive client selection approach for conducting FL to detect DDoS attacks within IoT environments. Evaluation proceeds in four stages. First, performance is assessed in client-side (standalone) settings. Next, FL experiments are conducted and compared with a centralized baseline. The proposed client-selection algorithm is then benchmarked against state-of-the-art methods under both IID and non-IID partitions, reporting accuracy and related metrics. Finally, communication overhead, latency, sensitivity (TPR), and convergence time are measured across federated communication rounds [4].

5.3.1 Classification Accuracy Achieved on the Client Side

In this experiment, we first establish a local training baseline by evaluating the detection performance of the CNN architecture when trained independently on each dataset partition, with each partition emulating the traffic collected by a distinct, heterogeneous IoT client.

For the CIC-IDS2018 dataset, the analysis involves five distributed clients, and the corresponding results are summarized in Table 23. The model exhibits strong client-side performance, achieving perfect accuracy, precision, recall, and F1 scores (100.00%) for benign traffic across all clients. For DDoS and other DoS-related attack classes, the model continues to perform reliably, attaining an average precision of 99.28% across attack

Table 23: Client-side evaluation results obtained on the CIC-IDS2018 dataset.

Class	Metric	Client accuracy [%]				
		IoT1	**IoT2**	**IoT3**	**IoT4**	**IoT5**
Benign	Precision	100.00	100.00	100.00	100.00	100.00
	Recall	100.00	100.00	100.00	100.00	100.00
	F1 score	100.00	100.00	100.00	100.00	100.00
DDOS attack–HOIC	Precision	97.12	97.23	97.14	97.18	97.34
	Recall	97.59	97.36	98.23	99.49	98.16
	F1 score	97.86	97.66	97.09	98.12	98.54
DDOS attack–LOIC-UDP	Precision	78.03	76.38	77.98	74.86	74.22
	Recall	87.20	89.10	84.18	85.00	87.00
	F1 score	83.34	84.48	84.08	84.80	84.00
DDOS attack–LOIC-HTTP	Precision	92.26	92.22	94.86	94.59	94.93
	Recall	98.21	99.87	98.49	98.99	98.62
	F1 score	95.12	95.10	96.91	96.87	96.93
DoS attack–GoldenEye	Precision	95.66	95.09	98.80	98.81	98.02
	Recall	98.21	99.02	99.87	99.49	99.99
	F1 score	96.04	97.64	99.13	99.81	99.64
DoS attack–Hulk	Precision	97.07	98.42	98.08	98.76	98.74
	Recall	99.27	98.82	99.41	98.51	99.36
	F1 score	98.19	99.85	99.30	99.87	99.15
DoS attack–SlowHTTPTest	Precision	77.11	67.48	68.82	78.37	79.02
	Recall	44.57	44.46	44.41	44.87	44.05
	F1 score	61.13	61.81	61.64	61.51	61.64
DoS attack–Slowloris	Precision	82.11	89.48	79.62	84.37	82.02
	Recall	77.40	75.62	76.49	76.98	76.50
	F1 score	88.07	85.42	86.08	86.64	86.32
Avg. accuracy				99.28		

categories. The only notable reduction occurs in detecting SlowHTTPTest attacks, where the recall drops to 44.57% due to the low representation of this class.

Overall, the model achieves an average accuracy of 99.28% across clients and attack types, demonstrating its robustness for IoT-focused DDoS detection. Across all client-side evaluations, the average precision, recall, and F1 score are 98.05%, 97.86%, and 96.54%, respectively [4].

For the CIC-DDoS2019 dataset, Table 24 summarizes the per-class performance obtained over 12 DDoS attack categories in addition to benign traffic. The client-side assessment indicates that the proposed method attains an overall accuracy of 95.85%. Benign flows are identified with an accuracy of 98.10%, while the DDoS attack classes collectively reach

an average accuracy of 95.60%. Across these attack categories, the model achieves a mean precision of 95.47%, an average recall of 95.40%, and a mean F1 score of 95.50%, reflecting consistent and well-balanced detection performance.

For the BoT-IoT dataset, the classification outcomes summarized in Table 25 highlight the model's effectiveness in detecting IoT-related DDoS attacks. Most performance metrics approach or reach 100%, demonstrating highly reliable behavior across clients. In particular, the *Benign* and *Reconn* classes consistently achieve perfect or near-perfect accuracy, precision, recall, and F1 scores, reflecting the robustness of the model in identifying these traffic types. Similarly, the *DDoS* and *Theft* classes maintain strong performance, with metrics exceeding 99.00% for the vast majority of evaluations, reinforcing the model's capability to detect these attacks effectively. Even for the *Scan* class, typically more challenging due to its subtle traffic patterns, the model performs exceptionally well, achieving approximately 98% accuracy. Overall, the client-side assessment yields an average accuracy of 99.20%, a precision of 99.70%, a recall of 99.85%, and an F1 score of 99.58%, indicating consistently high performance across all evaluated attack categories.

For the CIC-IoT2023 dataset, which includes eight DDoS attack types and benign traffic, the client-side evaluation yields an overall accuracy of 96.95% (Table 26). Benign samples are detected with 97.50% accuracy on average, whereas the average accuracy attained for the IoT attack types is 96.80%. Across these attack categories, the mean precision is 96.70%, the mean recall is 96.60%, and the mean F1 score is 96.65%.

5.3.2 Comparison with Centralized and FL Models

This experiment compares the accuracy of FELACS with centralized standalone models and with federated aggregation strategies, random client selection [110, 182], FedAvg [105], FedProx [113], and FedMCCS [114].

For the CIC-IDS2018 dataset, the proposed FELACS method attains an accuracy of approximately 95.0%, as illustrated in Figure 20a, surpassing all other FL strategies as well as the centralized baseline. For the

Table 24: Client-side evaluation results obtained on the CIC-DDoS2019 dataset.

Class	Metric	Client accuracy [%]				
		IoT1	**IoT2**	**IoT3**	**IoT4**	**IoT5**
Benign	Precision	98.98	99.08	98.85	98.75	98.72
	Recall	99.60	99.55	99.34	99.73	99.86
	F1 score	99.29	99.31	99.09	99.24	99.29
DNS	Precision	71.75	80.00	75.95	73.41	74.09
	Recall	67.65	76.33	75.16	78.87	79.67
	F1 score	69.86	78.93	73.85	74.81	76.98
SNMP	Precision	83.64	69.20	65.72	78.94	73.88
	Recall	86.28	74.71	63.55	75.14	74.61
	F1 score	85.70	77.85	62.59	76.05	74.15
LDAP	Precision	51.75	65.37	73.15	69.98	67.27
	Recall	52.28	71.28	86.41	90.25	65.53
	F1 score	56.83	68.03	79.34	81.16	66.26
MSSQL	Precision	89.22	85.16	93.15	92.69	88.67
	Recall	94.38	93.19	93.89	95.57	93.89
	F1 score	95.69	89.00	93.48	93.82	90.86
NTP	Precision	99.51	99.63	99.63	99.54	99.32
	Recall	99.05	99.44	99.35	99.06	99.34
	F1 score	99.28	99.54	99.49	99.37	99.33
NetBIOS	Precision	87.14	81.48	78.57	76.84	82.75
	Recall	70.94	82.30	73.18	72.72	80.67
	F1 score	81.36	85.69	75.78	74.78	81.54
SSDP	Precision	76.45	71.39	83.15	76.37	73.41
	Recall	82.28	73.23	93.89	73.72	71.29
	F1 score	79.52	72.27	88.20	74.82	72.37
Syn	Precision	98.50	97.25	97.63	98.62	95.24
	Recall	96.12	95.83	98.73	98.72	99.17
	F1 score	97.73	96.79	98.18	98.44	97.17
TFTP	Precision	100.00	99.69	99.98	100.00	99.93
	Recall	99.32	99.34	99.37	99.20	99.22
	F1 score	99.66	99.65	99.67	99.60	99.57
UDP	Precision	91.21	95.79	97.57	97.43	97.82
	Recall	97.34	93.87	95.23	95.83	95.82
	F1 score	94.05	94.65	96.40	96.62	96.89
WebDDoS	Precision	77.14	73.66	76.85	82.96	76.68
	Recall	69.43	71.28	74.18	95.57	74.34
	F1 score	73.63	72.28	75.27	88.82	75.57
Avg. accuracy				95.85		

Table 25: Client-side evaluation results obtained on the BoT-IoT dataset.

Class	Metric	Client accuracy [%]				
		IoT1	IoT2	IoT3	IoT4	IoT5
Benign	Accuracy	100.00	100.00	100.00	100.00	100.00
	Precision	100.00	100.00	100.00	100.00	100.00
	Recall	100.00	100.00	100.00	100.00	100.00
	F-measure	100.00	100.00	100.00	100.00	100.00
DDoS	Accuracy	99.01	99.28	99.28	99.23	99.25
	Precision	99.31	99.49	99.63	98.77	99.10
	Recall	99.77	99.60	99.88	99.87	99.86
	F-measure	99.73	99.72	99.82	99.84	99.86
Reconn	Accuracy	100.00	100.00	100.00	100.00	100.00
	Precision	99.73	98.59	99.70	99.60	99.29
	Recall	100.00	100.00	100.00	100.00	100.00
	F-measure	100.00	100.00	100.00	100.00	100.00
Theft	Accuracy	99.00	99.00	99.00	99.00	99.00
	Precision	99.03	99.10	99.11	99.11	99.37
	Recall	100.00	100.00	100.00	100.00	100.00
	F-measure	99.15	99.22	99.44	99.55	99.75
Scan	Accuracy	97.00	98.00	98.00	98.00	98.00
	Precision	98.00	98.08	98.15	98.08	98.15
	Recall	98.83	99.93	99.93	99.93	99.93
	F-measure	97.38	99.00	99.03	99.00	99.03
Avg. accuracy				99.20		

CIC-DDoS2019 dataset (Figure 20b), both FedMCCS and FELACS achieve the highest performance among the federated approaches, with accuracies near 92.0%, again exceeding the centralized model. On the BoT-IoT dataset (Figure 20c), FELACS maintains its lead, yielding an accuracy of roughly 92.0% and consistently outperforming the competing FL methods. For the CIC-IoT2023 dataset, Figure 20d shows that the centralized model (CL-sin) reaches about 99.0% accuracy on this more complex partition. Among FL-based strategies, FELACS achieves the strongest results, with an average accuracy close to 98.0%. Although the centralized approach retains a slight advantage in this case, FELACS matches or exceeds all other federated baselines.

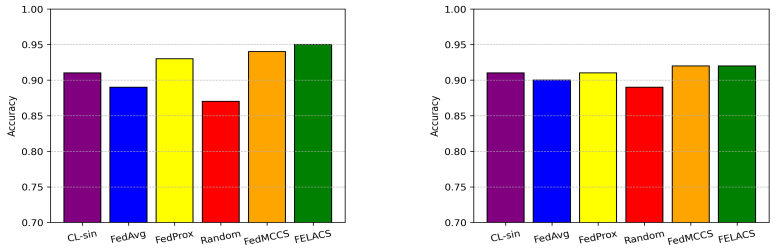
The results across all datasets demonstrate that the proposed method effectively adapts to evolving traffic distributions, maintaining high accuracy under diverse conditions and consistently outperforming other FL methods strategies occasionally surpassing even the centralized model.

Table 26: Client-side evaluation results obtained on the CIC-IoT2023 dataset.

Class	Metric	Client accuracy [%]				
		IoT1	**IoT2**	**IoT3**	**IoT4**	**IoT5**
Benign	Precision	92.43	95.48	93.32	94.57	94.06
	Recall	95.75	97.24	96.22	97.87	98.23
	F1 score	93.08	93.20	93.20	93.20	93.20
BruteForce	Precision	97.57	95.89	96.94	94.87	94.87
	Recall	91.83	95.68	93.72	95.68	95.68
	F1 score	95.32	95.40	95.34	94.64	94.96
DDoS	Precision	99.67	97.64	98.39	98.74	99.82
	Recall	99.94	98.30	98.92	97.34	97.07
	F1 score	99.55	99.54	98.02	98.90	97.48
DoS	Precision	97.56	95.47	89.43	94.23	95.92
	Recall	87.57	89.66	87.76	93.64	96.04
	F1 score	94.70	93.68	89.86	92.72	95.99
Mirai	Precision	100.00	100.00	100.00	100.00	100.00
	Recall	99.45	99.78	99.00	99.92	99.32
	F1 score	99.78	99.89	99.58	99.98	99.74
Recon	Precision	98.95	98.00	97.62	96.89	97.70
	Recall	100.00	100.00	100.00	100.00	99.74
	F1 score	99.08	99.01	98.76	98.46	98.68
Spoofing	Precision	83.79	83.80	83.00	83.00	83.00
	Recall	97.88	97.00	97.00	97.00	96.00
	F1 score	89.90	89.34	89.20	89.32	89.67
Web	Precision	74.64	89.00	78.27	76.66	74.64
	Recall	79.69	72.62	78.68	80.70	81.71
	F1 score	76.66	79.69	78.68	68.68	77.34
Avg. accuracy				98.86		

These outcomes highlight two important insights. First, federated learning can match or exceed centralized performance when client heterogeneity is properly addressed. Second, by dynamically selecting clients that contribute the most informative and resource-feasible updates, FELACS enhances each aggregation step, improving global model accuracy without compromising privacy or introducing overhead associated with centralized data collection.

The results indicate that the proposed methodology is adaptable to modern IoT networks under a range of network conditions.



(a) CIC-IDS2018 **(b)** CIC-DDoS2019

(c) BoT-IoT **(d)** CIC-IoT2023

Figure 20: Accuracies achieved by various aggregation strategies and centralized standalone models on the considered datasets. It is possible to observe how the proposed FELACS method achieves the highest accuracy among the tested FL-based techniques, in some cases surpassing the centralized version *(CL-sin)* [4].

5.3.3 Impact of the Number of Clients on Global Accuracy

In this section, we examine how the performance of the proposed approach evolves as the number of participating clients varies. Table 27 reports the mean classification accuracy of the global model over 50 communication rounds for client pools of 10, 25, and 50 devices across four benchmark datasets. In every configuration, both FELACS+CNN and FELACS+VGG16 surpass the accuracy achieved by FedAvg, FedProx, FedMCCS, and random selection, with the VGG16-based variant consistently yielding the best results.

For the CIC-IDS2018 dataset, FELACS+CNN attains accuracies of 98.20%, 99.80%, and 99.40% for 10, 25, and 50 clients, respectively, while FELACS+VGG16 further enhances performance to 98.50%, 99.90%, and 99.60%, offering improvements of up to 1.40% over the strongest base-

Table 27: Impact of the number of clients on the accuracy of the global model.

Algorithm	CIC-DDoS2019			BoT-IoT			CIC-IoT2023		
	10	25	50	10	25	50	10	25	50
FedAvg	91.80	92.40	93.00	96.80	97.60	98.00	94.20	94.80	95.10
FedProx	92.30	93.00	93.60	96.90	97.70	98.10	94.50	95.10	95.40
Random	91.20	91.80	92.40	96.60	97.30	97.80	93.80	94.30	94.60
FedMCCS	90.10	90.70	91.20	93.12	93.86	94.88	92.50	93.00	93.40
FELACS+CNN	93.00	94.00	94.50	97.10	97.90	98.30	95.20	95.80	96.00
FELACS+VGG16	93.50	94.50	95.00	97.40	98.10	98.50	95.50	96.10	96.30

line (FedProx). Similarly, on CIC-DDoS2019, FELACS+CNN records accuracies of 93.00%, 94.00%, and 94.50%, whereas FELACS+VGG16 increases these values to 93.50%, 94.50%, and 95.00%, outperforming FedProx by margins of up to 1.40%. For the BoT-IoT dataset, FELACS+CNN reaches 97.10%, 97.90%, and 98.30%, while FELACS+VGG16 achieves 97.40%, 98.10%, and 98.50%, again exceeding FedProx by as much as 1.40%. Finally, on the CIC-IoT2023 dataset, FELACS+CNN yields accuracies of 95.20%, 95.80%, and 96.00%, and FELACS+VGG16 attains 95.50%, 96.10%, and 96.30%, surpassing all federated baselines by approximately 0.60–1.00%.

The experiments conducted with varying numbers of participating clients highlight two main observations. First, increasing the client population enhances global model performance by introducing greater data diversity. Second, combining FELACS with the deeper VGG16 backbone results in further accuracy improvements compared with the CNN baseline, indicating that FELACS can sustain high-quality aggregation even under strongly non-IID data distributions.

However, the benefits diminish as the client pool grows larger: scaling to 100 clients produces only marginal accuracy gains relative to configurations with fewer participants. Although a broader client set captures more heterogeneous data, this advantage saturates beyond a certain threshold. At the same time, the use of VGG16 with its deeper architecture and significantly larger parameter count raises the computational and communication cost for resource-limited edge and IoT devices.

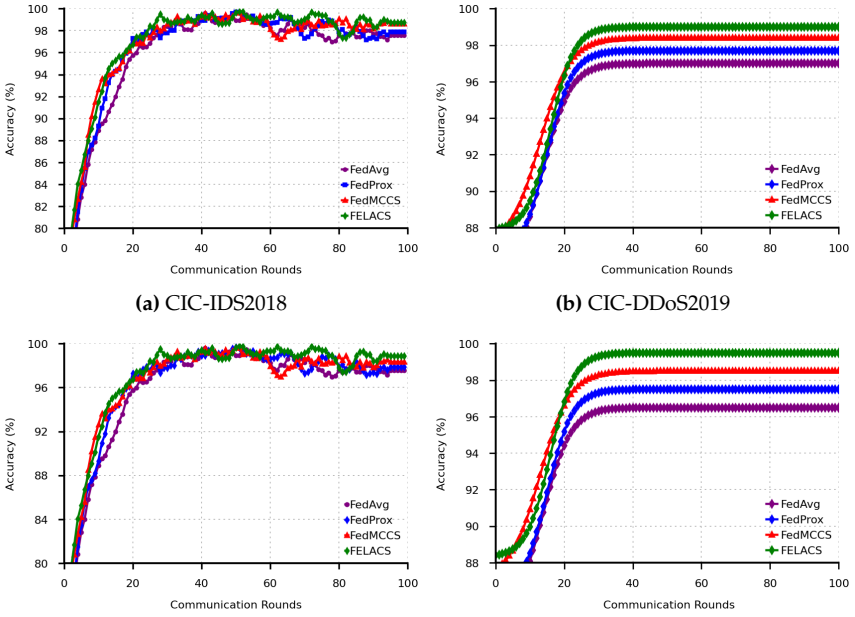
Across our evaluations, selecting approximately 25 clients provides the most favorable trade-off between accuracy and efficiency. This configuration preserves adequate data diversity to avoid both underfitting and overfitting, while preventing the excessive training time, memory usage, and bandwidth consumption associated with requiring many devices to compute or transmit updates for a heavyweight model such as VGG16. These results underscore the need for a carefully designed client selection strategy that jointly considers model complexity and edge resource constraints.

5.3.4 Classification Results

The FL classification results on the CIC-IDS2018 dataset under the FELACS client–selection strategy are shown in Figure 21a. Across all 100 communication rounds, FELACS consistently surpasses FedAvg, FedProx, FedMCCS, and Random selection, maintaining higher accuracy throughout training.

Similarly, Figure 21b reports the accuracy trends for CIC-DDoS2019. FELACS again delivers the strongest performance, achieving accuracy levels above 98.8% and exhibiting stable convergence over the rounds. The results for the BoT-IoT dataset, illustrated in Figure 21c, show all methods approaching near-perfect accuracy, with FELACS obtaining the highest final performance. Its sustained advantage reflects the effectiveness of prioritizing clients based on data relevance and resource conditions in highly heterogeneous IoT settings.

Finally, Figure 21d presents outcomes on CIC-IoT2023. FELACS exceeds 98% accuracy by round 20 and reaches the highest overall accuracy, approximately 99.5%, demonstrating fast convergence and strong steady-state performance across diverse datasets.



(a) CIC-IDS2018 **(b)** CIC-DDoS2019

(c) BoT-IoT **(d)** CIC-IoT2023

Figure 21: Detection accuracies over 100 rounds with different client-selection strategies (FELACS vs others) [4].

5.3.5 Communication Overhead

This section evaluates convergence by examining the number of communication rounds required to achieve a target accuracy. Table 28 reports the rounds needed to reach 90% and 95% accuracy under both IID and non-IID data partitions for all four datasets. As anticipated, heterogeneous (non-IID) distributions lead to increased communication overhead, with all methods requiring additional rounds compared to the IID case. Conventional strategies such as FedAvg and FedProx typically converge within the mid-30s to mid-40s for the 90% threshold and between the high-30s and mid-50s for 95%. Random client selection performs worst, often exceeding 60 rounds under skewed data conditions.

In contrast, the proposed FELACS strategies significantly reduce the

Table 28: Number of communication rounds required to reach 95% and 90% accuracy (IID / non-IID) on the considered datasets.

Approach	CIC-IDS2018		CIC-DDoS2019		BoT-IoT		CIC-IoT2023	
	ToA@95	ToA@90	ToA@95	ToA@90	ToA@95	ToA@90	ToA@95	ToA@90
FedAvg	38/43	32/36	50/58	42/50	46/54	40/46	44/52	36/44
FedProx	36/40	30/34	48/54	40/46	44/50	38/43	42/48	35/40
Random	50/58	42/48	65/75	56/64	60/68	52/56	58/68	50/56
FedMCCS	34/38	28/32	46/52	38/44	42/48	36/40	40/46	34/38
FELACS+CNN	30/35	25/28	40/46	32/38	38/45	32/38	36/42	30/34
FELACS+VGG16	28/32	23/26	38/44	30/36	36/43	30/36	34/40	28/32

number of communication rounds, consistently achieving both accuracy targets within the low-30s to low-40s even under severe non-IID distributions. These findings highlight that by selecting clients whose updates offer the highest informational value while accounting for resource constraints, FELACS enables faster convergence with substantially fewer communication exchanges, an essential requirement for real-time and bandwidth-limited IoT environments [4].

5.3.6 Model Convergence Time Analysis

This subsection examines the convergence time required by the evaluated methods. Table 29 compares the temporal efficiency of the different FL strategies across 100 communication rounds for all datasets. The results show that FELACS consistently outperforms the baseline approaches, achieving lower average communication time per round and requiring fewer rounds to reach convergence. These observations confirm that FELACS effectively reduces communication overhead and improves the overall efficiency of FL in IoT environments, making it a strong candidate for deployment in practical scenarios.

Nevertheless, when FELACS is applied to the deeper VGG16 architecture, the substantially larger number of parameters and increased network depth introduce considerable communication and computational demands for resource-limited IoT and edge devices. Although FELACS+VGG16 achieves the highest accuracy among the tested mod-

Table 29: Per-round time (minutes) required by different FL client selection methods for four datasets.

Method	CIC-IDS2018		CIC-DDoS2019		BoT-IoT		CIC-IoT2023	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
FedAvg	5.45	0.32	6.00	0.35	5.12	0.28	5.00	0.30
FedProx	5.20	0.25	5.80	0.30	4.98	0.22	4.80	0.24
Random	6.30	0.50	7.00	0.55	6.05	0.45	6.10	0.48
FedMCCS	9.20	0.87	9.80	1.00	8.76	0.79	9.00	0.85
FELACS+CNN	4.82	0.20	5.30	0.22	4.65	0.18	4.50	0.16
FELACS+VGG16	9.50	0.86	11.80	1.35	8.20	0.56	9.00	0.98

els, it also incurs significantly higher bandwidth consumption per round compared with FELACS+CNN. To address these constraints in real-world deployments, model compression techniques that employ 8-bit quantization and structured pruning could be employed to reduce payload sizes and computational requirements. Such optimizations would allow FELACS+VGG16 to retain its convergence advantages while remaining compatible with IoT and edge resource limitations.

FELACS Convergence is analyzed theoretically by building on [183], which establishes that FL converges on heterogeneous data at an $O(1/T)$ rate for sequential federated learning on heterogeneous (non-IID) data with strongly convex objectives, provided each client has a positive probability of selection in every round. Moreover, the works of [184,185] demonstrated the convergence of client selection schemes with bounded gradient estimation. In this context, the multi-criteria optimization scheme of FELACS ensures that every client retains a strictly positive selection probability and keeps the gradient estimation variance bounded. Moreover, the federated multiobjective learning framework [186,187] extends identical guarantees to multigradient aggregation scenarios. FELACS satisfies all core assumptions of these convergence theories and therefore exhibits the predicted convergence behavior in our IoT DDoS detection experiments.

5.3.7 Sensitivity Analysis

The robustness of the proposed method is assessed by adjusting each weight (α, β, γ, δ) by $\pm 20\%$ from its default value and evaluating the resulting detection accuracy and F1 scores on the CIC-IDS2018, CIC-DDoS2019, BoT-IoT, and CIC-IoT2023 datasets. Across all datasets and perturbation levels, the maximum variations in accuracy and F1 score remain bounded within $\pm 5\%$. These results demonstrate that the importance-scoring mechanism used in FELACS is resilient to moderate hyperparameter fluctuations, incurring no substantial degradation in performance. This robustness reduces the need for extensive hyperparameter tuning in practical federated IoT settings and indicates that the method can be reliably deployed under diverse device capabilities and network conditions with minimal recalibration.

5.4 Summary

FELACS is introduced as an adaptive client-selection approach for IoT DDoS detection in FL settings. The method formulates client selection as a multi-objective optimization that accounts for data diversity and importance as well as real-time device performance. In contrast with the existing methods in the literature, the approach integrates a multi-objective optimization framework that evaluates key factors, including data entropy, variance, power consumption, and latency, for each client. This comprehensive approach ensures an effective and balanced client selection process during FL, mitigating client drift by favoring the clients with the most representative data and sufficient computational resources.

The method is validated on four public datasets, achieving higher classification accuracy than a non-federated baseline while requiring fewer communication rounds. In comparison with FedAvg, FedProx, FedMCCS, and random client selection, it attains superior accuracy and faster convergence. Future work will focus on developing robust, scalable, and interpretable models using federated transfer learning to enhance the DDoS attack detection process in IoT environments.

Chapter 6

Robust Poisoning Attack Detection in Federated Learning

6.1 Introduction

The rapid expansion of IoT technologies has reshaped modern digital ecosystems by enabling pervasive connectivity among heterogeneous devices that continuously exchange data and automate critical processes. These devices generate vast volumes of information that support service optimization and predictive analytics. However, the distributed and sensitive nature of such data also exposes IoT infrastructures to increasingly sophisticated cyber threats. Safeguarding these decentralized environments has therefore become essential to maintaining their reliability and operational integrity. In this context, FL has emerged as a viable framework for enhancing IoT security, as it enables collaborative model training across devices without transmitting raw data, thereby preserving privacy and reducing communication overhead [105].

Parts of this chapter are based on the work presented in M. B. Anley, A. Genovese, T. B. Tesema, and V. Piuri, "FLIFRA: Hybrid data poisoning attack detection in federated learning for IoT security," in *Proceedings of the 2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2025)*, Vienna, Austria, October 5–8, 2025, pp. 1–8.

Despite these advantages, the decentralized structure of FL introduces new attack surfaces, particularly for poisoning attacks that exploit client-side vulnerabilities. These attacks can manifest in multiple forms, each designed to degrade or subvert the global model. Their impact is amplified in IoT settings, where heterogeneous and non-IID data distributions create additional challenges. For example, backdoor attacks insert malicious triggers into the training data to force targeted misclassifications under specific conditions [116], potentially causing IoT intrusion detection systems to label malicious flows as benign. Label-flipping attacks intentionally corrupt class labels, disrupting the learning process and reducing model accuracy [117]. Likewise, gradient manipulation attacks tamper with the updates shared during aggregation, subtly injecting harmful modifications that can compromise the integrity of the global model [118].

Most existing defenses against data poisoning in FL concentrate either on client-side detection mechanisms [123, 124] or on server-side aggregation strategies [121, 125, 127, 128], rather than adopting an integrated, end-to-end perspective. This separation limits overall robustness as client-side approaches may be ineffective against sophisticated or stealthy poisoning behaviors, whereas server-side methods lack fine-grained visibility into local data distributions and training dynamics [119]. These shortcomings highlight the necessity for a robust, scalable, and adaptive defense framework specifically designed to address the constraints and heterogeneity of IoT environments [188].

To overcome these challenges, FLIFRA [5] proposes a dual-layer defense architecture that tightly couples local anomaly detection using iForest with a global Dynamic Reputation-Based Robust Aggregation (DRRA) mechanism. At the client level, iForest systematically analyzes local training data and model-update behavior to detect anomalous contributions, while at the server level, DRRA adaptively modulates client influence based on historical reliability, thereby enhancing resilience against poisoning attacks.

Contributions. This chapter makes three primary contributions:

- A formal problem statement is developed for FL defenses against

data-poisoning attacks in IoT environments marked by non-IID data, heterogeneous devices, and tight resource budgets.

- *FLIFRA* (Federated Learning iForest with Robust Aggregation) is a scalable, adaptive dual-layer framework. It integrates client-side iForest anomaly screening with server-side DRRA, enabling robust aggregation that downweights or excludes adversarial updates.
- The proposed work conducts a rigorous empirical evaluation under varying poisoning intensities and non-IID partitions on real-world IoT datasets, demonstrating higher accuracy, faster and more stable convergence, and improved robustness over standard and robust baselines.[1]

This chapter builds on our accepted paper [5].

6.1.1 Data Poisoning Attacks In FL

As depicted in Figure 22, federated learning in IoT-Edge ecosystems spans a broad range of application domains, including smart transportation, healthcare, industrial IoT (IIoT), and smart-city infrastructures. Within this paradigm, IoT-Edge devices that smart sensors, wearable technologies, and industrial controllers collect and process local data to train ML models. Each device performs training independently, ensuring privacy by retaining raw data locally and transmitting only model parameters or gradient updates to a central server. The server aggregates these contributions to construct a global model, which is subsequently distributed back to participating devices for additional local refinement. Despite these privacy-preserving benefits, such decentralized coordination increases vulnerability to data poisoning attacks [119].

In a data poisoning scenario, one or more IoT-Edge devices may be compromised, allowing an adversary to manipulate local training data before updates are sent to the server. For example, in a smart-home environment, an attacker-controlled device can train on malicious samples and submit corrupted model updates. When these poisoned updates

[1]Source code will be released at `https://github.com/mulerkal/FLIFRA`.

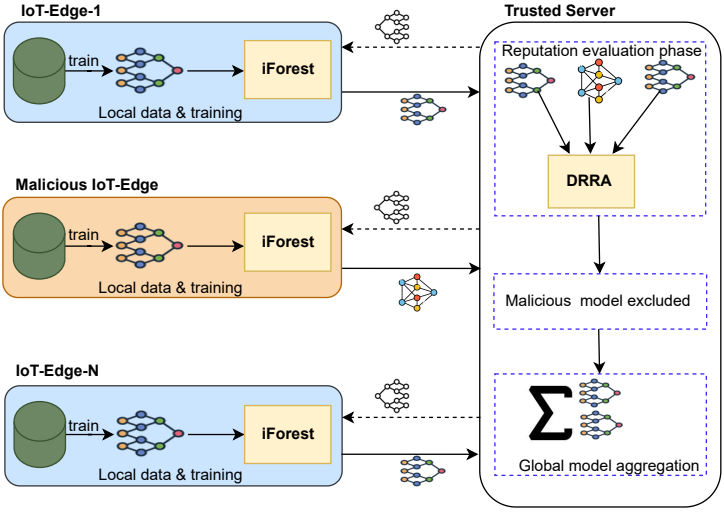


Figure 22: Architecture of an FL-based data poisoning attack in IoT-Edge environments.

are merged with legitimate ones during aggregation, they degrade the integrity and accuracy of the global model. The resulting compromised model is then propagated to all participating devices, amplifying the attack's impact across the entire system [120], [5].

6.1.2 Poisoning Attack Detection in FL

FL remains highly susceptible to data poisoning attacks, in which adversarial clients introduce corrupted samples into local training to impair global model performance. Such attacks pose serious risks to model integrity, particularly within IoT deployments where devices operate under strict resource limitations and rapidly changing conditions that further complicate detection and mitigation [121]. Existing countermeasures generally fall into two categories: client-side and server-side defenses [122]. Client-side approaches, such as iForest-based anomaly detection [123],

attempt to identify abnormal behaviors by analyzing local training data. Nonetheless, these methods often fail to recognize coordinated or stealthy adversaries whose updates resemble benign patterns, limiting their practicality in operational IoT networks. To alleviate this issue, FL-WBC [124] perturbs parameters during local training to diminish the impact of persistent poisoning attacks. While effective in certain cases, such techniques frequently struggle to scale and adapt to dynamic environments.

Server-side defenses employ a variety of strategies to filter, cluster, or robustly aggregate client updates to suppress malicious contributions. Provenance-driven mechanisms [121], for example, track the origins of updates to detect compromised clients; however, they depend on reliable provenance information, which may be infeasible in constrained IoT settings. Other methods incorporate explicit client filtering [125] or clustering-based separation of benign and malicious updates [126]. Robust aggregation rules—including Krum [127], Trimmed Mean, and coordinate-wise median [128] reduce the impact of adversarial updates by focusing on the most consistent gradients across clients. These techniques offer resilience when only a small fraction of clients are compromised, but their effectiveness deteriorates as the number or proportion of adversarial participants increases.

In addition, Fed-LSA [129] introduces a latent-space inspection mechanism using autoencoders to identify poisoned updates, representing a novel direction for detecting adversarial contributions. Nevertheless, its effectiveness diminishes under non-IID data conditions, which reduces its practicality in heterogeneous IoT deployments. Likewise, FreqFed [130] employs frequency-domain features to flag anomalous updates but relies on the assumption of stable data distributions, limiting its robustness in dynamic and evolving IoT environments.

Further, [189] proposed an anomaly-detection framework based on WeiDetect that incorporates a trusted validation set to filter out malicious or low-quality updates before global aggregation. The FLAME method [190] takes a complementary direction by combining model clustering with weight clipping to suppress adversarial influence during training.

Despite these advancements, reliable detection and mitigation of data

poisoning in FL remain challenging. Many existing defenses concentrate solely on either client-side or server-side mechanisms, resulting in fragmented protection. Client-side techniques frequently fail to identify advanced or coordinated attacks [124], whereas server-side defenses generally lack visibility into the underlying local data distributions that contribute to the aggregated updates [119].

6.2 Methodology

This subsection presents the client-side iForest anomaly-detection method, the server-side DRRA mechanism, and the hybrid FLIFRA framework, together with their corresponding algorithms as described in [5].

6.2.1 Client-Side: iForest Anomaly-based Detection

The first layer of the proposed FLIFRA hybrid detection approach employs the iForest algorithm to identify anomalies in model updates. The detailed steps of the algorithm, as applied in our scenario, are described below and outlined in Algorithm 3.

- *Local application of iForest:* after local model training, each IoT device applies the iForest algorithm to its model update. The algorithm randomly selects a feature and a split value to partition the data, constructing trees where outliers are isolated near the root due to their uniqueness. This process is repeated across multiple trees in the ensemble.

- *Anomaly score calculation:* For each model update, iForest calculates an anomaly score based on the average path length required to isolate the point in all trees in the forest. The path length is defined as the number of edges traversed from the root node to the terminating node. The score is calculated using the following (equation 6.1):

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}, \tag{6.1}$$

where $E(h(x))$ is the expected path length of a point x in the iForest, n is the number of samples, and $c(n)$ is the average path length of an unsuccessful search in a Binary Search Tree (BST).

- *Flagging potentially malicious updates:* If a model update's anomaly score exceeds a certain threshold, it is considered an outlier and is flagged as potentially malicious. The proposed work employs a dynamic thresholding mechanism to enhance the detection of malicious activities. The threshold is continuously updated based on the distribution of anomaly scores over time, utilizing a rolling-window approach.

- *Reporting to the central server:* Flagged updates, along with their anomaly scores, are then sent to the central server.

6.2.2 Server Side: Dynamic Reputation-Based Robust Aggregation (DRRA)

Located on the server side, DRRA dynamically adjusts the weight of updates for each client during the aggregation process based on the evolving reputation scores from the training rounds. The algorithm assigns higher weights to reliable clients and penalizes those exhibiting malicious behavior. The details are provided in the following and illustrated in Algorithm 4.

- *Reputation score initialization:* At the beginning of the training process, all clients are assigned equal reputation scores $R_i(0)$, where $R_i(0) = 1$ for each client i. This uniform initialization assumes no prior knowledge about client behavior and treats all clients equally trustworthy. As training progresses, the reputation scores are dynamically updated based on the consistency and reliability of the client's contributions.

Algorithm 3: iForest Anomaly Detection

Input: Local dataset D_i, contamination level η, number of estimators n
Output: Filtered dataset D_i'
Objective: identify and remove the most abnormal samples from D_i before downstream training/analysis.
Expected outcome: $D_i' \subseteq D_i$ retains samples labeled as *normal*, while anomalous samples are excluded.
Initialize iForest model $\mathcal{M}$ with n trees and contamination η
Preprocess D_i (e.g., handle missing values and standardize/normalize features for consistent scoring)
Train $\mathcal{M}$ on D_i
foreach *data point* $x_j \in D_i$ **do**
    Compute anomaly score $s_j \leftarrow \mathcal{M}.score(x_j)$
    (Higher s_j indicates a more anomalous sample.)
Let $\mathcal{S} = \{s_j\}_{j=1}^{|D_i|}$
Compute threshold

$$\theta \;=\; Q_{1-\eta}(\mathcal{S}),$$

the $(1-\eta)$-quantile of $\mathcal{S}$
(This choice flags approximately an η fraction of samples as anomalies.)
Initialize $D_i' \leftarrow \emptyset$
foreach *data point* $x_j \in D_i$ **do**
    if $s_j > \theta$ **then**
        Label x_j as *anomaly* and exclude it from D_i'
    else
        Label x_j as *normal* and append x_j to D_i'
return D_i'

- *Reputation Score Update Mechanism:* Each client i maintains a reputation score $R_i(t) \in [0, 1]$ at training round t.

$$R_i(t+1) = \alpha\, R_i(t) \;+\; (1-\alpha)\, S_i(t), \quad \alpha \in [0,1] \tag{6.2}$$

where α is a decay factor controlling the historical behavior's influence and $S_i(t)$ is a performance score computed based on the similarity of the client's update to the median of all client updates at

round t. We compute $S_i(t)$ by measuring the Euclidean distance between the client's update $U_i(t)$ and the central model update $U_m(t)$, defined as the component-wise median of all client updates:

$$S_i(t) = 1 - \frac{\text{distance}(U_i(t), U_m(t))}{\max_j(\text{distance}(U_j(t), U_m(t)))}\ , \tag{6.3}$$

where: $U_i(t)$ is the update from client i, $U_m(t)$ is the median update, and distance$(\cdot, \cdot)$ is a metric Euclidean distance. Clients with updates closer to the median receive higher scores, while those with significant deviations are penalized.

- *Weighted aggregation of client updates:* After computing the updated reputations $R_i(t+1)$ for all i, the server apply threshold $\tau_r \in [0, 1]$, setting

$$\tilde{R}_i(t+1) = \begin{cases} R_i(t+1), & R_i(t+1) \geq \tau_r, \\ 0, & R_i(t+1) < \tau_r. \end{cases} \tag{6.4}$$

Denote by N the total number of clients. The aggregation weight for client i is then

$$w_i(t+1) = \frac{\tilde{R}_i(t+1)}{\sum_{j=1}^{N} \tilde{R}_j(t+1)}, \quad i = 1, \ldots, N. \tag{6.5}$$

Finally, the new global model update is

$$U_{\text{global}}(t+1) = \sum_{i=1}^{N} w_i(t+1)\, U_i(t). \tag{6.6}$$

6.2.3 FLIFRA: Hybrid Approach

The proposed approach combines client-side anomaly detection via iForest with robust server-side DRRA aggregation. Details are provided in Algorithm 5.

Algorithm 4: Dynamic Reputation–Based Robust Aggregation (DRRA)

Input: Client updates $U_1, \ldots, U_N$, reputations $R_1, \ldots, R_N$, decay α, reputation threshold τ_r, small constant ε.
Output: Global update U_{global}, weights $w_1, \ldots, w_N$.
Objective: Reduce the influence of unreliable client updates by assigning reputation-based weights before aggregation.
Expected outcome: down-weight (or exclude) updates far from the central tendency and aggregate primarily from consistent clients.

$U_m \leftarrow \text{median}(U_1, \ldots, U_N)$;
(Use the coordinate-wise median as a robust reference update.)
foreach $i = 1, \ldots, N$ **do**
  $d_i \leftarrow \|U_i - U_m\|_2$;
  (Larger d_i indicates a more divergent update.)
$D_{\max} \leftarrow \max\big(\max_i d_i,\, \varepsilon\big)$;
(Avoid division by zero and normalize distances.)
foreach $i = 1, \ldots, N$ **do**
  $S_i \leftarrow \min\{1, \max\{0,\, 1 - d_i / D_{\max}\}\}$;
  (Compute a bounded consistency score $S_i \in [0, 1]$; higher is better.)
  $R_i \leftarrow \alpha\, R_i + (1 - \alpha)\, S_i$;
  (Update reputation using an exponential moving average.)
  if $R_i \geq \tau_r$ **then**
    $w_i \leftarrow R_i$
  else
    $w_i \leftarrow 0$
  (Exclude clients below the reputation threshold.)
If $\sum_{j=1}^{N} w_j = 0$, set $w_i \leftarrow 1/N$ for all i (fallback) to avoid undefined normalization.
$\{w_i\} \leftarrow \{\, w_i / \sum_{j=1}^{N} w_j\}$;
$U_{\text{global}} \leftarrow \sum_{i=1}^{N} w_i\, U_i$;
(Aggregate updates using reputation-weighted averaging.)
return $U_{\text{global}},\ \{w_i\}$

Algorithm 5: FLIFRA: Hybrid iForest–DRRA

Input: Client datasets $\{D_i\}_{i=1}^{N}$, contamination η, iForest trees n, reputation decay α, reputation threshold τ_r, rounds T, initial model $M^{(0)}$, server LR γ, small constant ε.
Output: Robust global model $M^{(T)}$.
foreach $i = 1, \ldots, N$ **do**
  $R_i \leftarrow 1$;
$M^{(0)} \leftarrow$ initial model;
for $t = 1$ **to** T **do**
  `// Client-Side: iForest`
  foreach $i = 1, \ldots, N$ **do**
    in parallel
  Train iForest $\mathcal{M}$ on D_i with η, n;
  foreach $x \in D_i$ **do**
    Compute $s_{i,x} \leftarrow \mathcal{M}_i.\text{score}(x)$;
  $\theta_i \leftarrow Q_{1-\eta}(\{s_{i,x}\})$;
  $D_i' \leftarrow \{\, x \in D_i : s_{i,x} \le \theta_i \}$;
  Train local model M_i on D_i' starting from $M^{(t-1)}$;
  $U_i \leftarrow M_i - M^{(t-1)}$;
  Send U_i to server;
  `// Server-Side: DRRA`
  $U_m \leftarrow \text{median}(\{U_i\}_{i=1}^{N})$;
  foreach $i = 1, \ldots, N$ **do**
    $d_i \leftarrow \|U_i - U_m\|_2$;
  $D_{\max} \leftarrow \max\big(\max_i d_i,\ \varepsilon\big)$;
  foreach $i = 1, \ldots, N$ **do**
    $S_i \leftarrow \min\{1, \max\{0,\ 1 - d_i / D_{\max}\}\}$;
    $R_i \leftarrow \alpha\, R_i + (1 - \alpha)\, S_i$;
    if $R_i < \tau_r$ **then**
      $w_i \leftarrow 0$
    else
      $w_i \leftarrow R_i$
  Normalize $\{w_i\}$: $w_i \leftarrow w_i \,/\, \sum_{j=1}^{N} w_j$;
  $U_{\text{global}} \leftarrow \sum_{i=1}^{N} w_i\, U_i$;
  $M^{(t)} \leftarrow M^{(t-1)} + \gamma\, U_{\text{global}}$;
return $M^{(T)}$;

6.3 Experimental Results and Discussion

6.3.1 Threat Models in FL

In the dual-layer defense setting considered here, the adversary seeks to undermine the global NIDS model by submitting poisoned updates

that both reduce overall detection accuracy and embed a covert backdoor, enabling benign IoT traffic to be flagged as malicious or allowing malicious flows to evade detection. The attacker is assumed to have full control over one or more compromised IoT clients, giving them the ability to manipulate local data and gradients and to transmit arbitrary updates to the server. However, the adversary cannot alter the updates of honest participants nor access the server-side validation set used for reputation assessment. Within this threat model, our study concentrates on three poisoning vectors: data injection for embedding backdoor triggers, label-flipping attacks that corrupt class assignments, and broader gradient-based poisoning strategies [5].

6.3.2 Datasets

CIC-IDS2018

The dataset was produced in a controlled network testbed that emulates realistic traffic patterns across diverse applications and services, making it well aligned with contemporary network environments [135]. In this study, we adopt the CIC-IDS2018[2] dataset due to its strong capability to replicate both IoT and non-IoT network behaviors, offering extensive coverage of normal activity as well as numerous attack scenarios. Its comprehensive feature set and balanced distribution of traffic categories make it particularly suitable for addressing the challenges posed by data heterogeneity and non-IID characteristics, which frequently arise in federated learning deployments.

BoT-IoT

The dataset was generated within a controlled testbed that emulated a heterogeneous IoT environment composed of devices such as smart thermostats, webcams, motion detectors, and various connected home appliances. It contains both benign traffic, reflecting normal operational behavior, and malicious traffic produced through multiple cyberattack

[2] https://www.unb.ca/cic/datasets/ids-2018.html

scenarios [138]. The BoT-IoT dataset is publicly accessible[3] and is widely regarded for its realistic portrayal of IoT network activity and its extensive coverage of both legitimate and adversarial conditions.

UNSW-NB15

The UNSW-NB15 dataset, produced by the Australian Centre for Cyber Security (ACCS)[4], serves as a modern benchmark for network intrusion detection. It comprises both benign and malicious traffic spanning a variety of contemporary attack categories. Its rich feature representation and well-defined labels make it particularly suitable for FL-based experimentation. The distribution of samples and classes employed in our experiments is provided in Table 5.

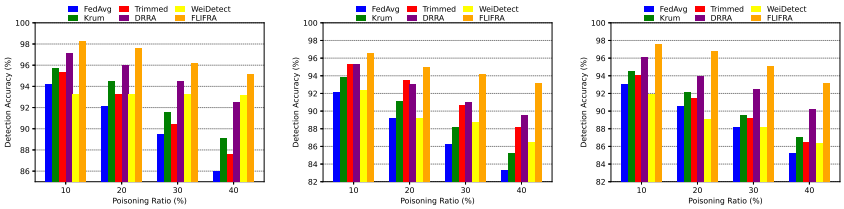
6.3.3 Data Preprocessing

Before applying the proposed methodology, all three datasets underwent a comprehensive preprocessing pipeline to ensure consistency and suitability for model training. Duplicate records and missing values were removed, and irrelevant or low-information features were discarded. Categorical attributes, protocol identifiers, and attack labels were transformed into numerical form using label encoding to facilitate efficient computation. Continuous variables, including flow duration and packet size, were normalized via Min Max scaling to map their values to the $[0, 1]$ interval, thereby promoting stable and efficient model convergence. Dimensionality reduction was performed by selecting the most informative features based on entropy and information-gain criteria, ensuring that essential characteristics were preserved without degrading classification performance [3].

6.3.4 Dataset Distribution

For each dataset, 80% of the samples were allocated for training, while the remaining 20% were reserved for testing. Within the proposed FL frame-

[3] https://research.unsw.edu.au/projects/bot-iot-dataset
[4] https://research.unsw.edu.au/projects/unsw-nb15-dataset



(a) CIC-IDS2018 **(b)** BoT-IoT **(c)** UNSW-NB15

Figure 23: Detection accuracy (%) of baseline and proposed methods across poisoning ratios (10%–40%) on three datasets.

work, the training portion is further partitioned among multiple client nodes to emulate realistic non-IID data conditions. Each client receives a unique subset of the data with differing ratios of benign and malicious samples as well as varying coverage of attack categories, reflecting the diversity observed across operational IoT deployments. To construct these heterogeneous partitions, we employ a Dirichlet-based allocation scheme, a standard approach in FL research for generating controlled class-imbalance and distribution skew [181]. This sampling strategy ensures that the local datasets mirror real-world traffic variability, thereby enhancing the realism and robustness of the experimental evaluation.

6.3.5 Model Architecture

In this research, a CNN model was developed specifically to identify data poisoning attacks within FL frameworks for IoT environments. The model integrates several critical components to tackle the distinct challenges posed by IoT networks. It includes multiple convolutional layers to extract relevant features, followed by pooling layers that help reduce the data's dimensionality while retaining important patterns. Moreover, techniques that include dropout and batch normalization are utilized to improve the model's generalization capabilities and overall robustness.

6.3.6 Hyperparameters

In this study, the CNN models were trained on each client using a batch size of 64, with the Adam optimizer and categorical cross-entropy loss applied for multi-class classification tasks. To mitigate the effects of non-IID data distributions, several hyperparameters were tuned on a per-client basis, including the local learning rate, number of local training epochs, and dropout rate, thereby improving both performance and generalization across heterogeneous nodes. ReLU activations were employed within the hidden layers to facilitate non-linear feature extraction, while a SoftMax layer at the output provided normalized class probabilities. At the server level, a minimum of 100 clients were selected in each communication round to maintain statistical robustness. To further reduce overfitting, early stopping triggered by divergence in validation loss and dropout regularization was incorporated into the training procedure.

6.3.7 Baselines

To assess the effectiveness of the FLIFRA model in detecting data poisoning attacks, we conducted a comparative evaluation against a baseline and several widely adopted aggregation strategies, including Trimmed Mean [128], Krum [127], DRRA, WeiDetect [189], and the standard FedAvg algorithm [105]. These methods were chosen due to their established capability to counter Byzantine behavior and support robust model aggregation in federated learning environments. By incorporating both classical and state-of-the-art aggregation techniques, the evaluation framework provides a comprehensive benchmark for measuring the effectiveness of the proposed approach. Building on this foundation, FLIFRA introduces a dynamic reputation-based scoring mechanism that further strengthens resilience to data poisoning attacks.

6.3.8 Evaluation Metrics

Evaluation metrics include accuracy, precision, recall, TPR, FAR, and F1 score. These metrics are used for both the detection experiments and the

Table 30: Impact of poisoning attack intensity on precision and F1-score values across the considered datasets [5].

Dataset	Int.(%)	FedAvg		Krum		Trimmed Mean		DRRA		FLIFRA	
		Prec.	F1	Prec.	F1	Prec.	F1	Prec.	F1	Prec.	F1
CIC-IDS2018	10	93.8 ± 1.2	94.0 ± 1.1	95.7 ± 1.0	96.0 ± 0.9	95.0 ± 1.1	95.2 ± 1.0	96.8 ± 0.8	97.0 ± 0.	**98.0 ± 0.7**	**98.1 ± 0.7**
	20	91.5 ± 1.6	91.8 ± 1.5	94.5 ± 1.2	94.6 ± 1.2	93.2 ± 1.3	93.4 ± 1.3	95.8 ± 1.0	95.9 ± 1.	**97.3 ± 0.9**	**97.4 ± 0.9**
	30	88.9 ± 2.1	89.2 ± 2.0	91.6 ± 1.8	91.8 ± 1.7	90.4 ± 2.0	90.5 ± 1.9	94.2 ± 1.6	94.3 ± 1.	**96.0 ± 1.2**	**96.1 ± 1.2**
	40	85.5 ± 2.6	85.7 ± 2.5	89.1 ± 2.3	89.3 ± 2.2	87.6 ± 2.4	87.8 ± 2.3	92.2 ± 1.9	92.3 ± 1.	**94.8 ± 1.6**	**94.9 ± 1.5**
BoT-IoT	10	91.8 ± 1.3	92.0 ± 1.2	93.5 ± 1.1	93.7 ± 1.0	92.8 ± 1.1	93.0 ± 1.1	95.0 ± 0.9	95.1 ± 0.	**96.3 ± 0.8**	**96.4 ± 0.8**
	20	88.5 ± 1.9	88.8 ± 1.8	90.8 ± 1.6	90.9 ± 1.5	90.0 ± 1.6	90.1 ± 1.6	92.8 ± 1.2	92.9 ± 1.	**94.8 ± 1.0**	**94.9 ± 1.0**
	30	85.5 ± 2.4	85.7 ± 2.3	87.8 ± 2.1	87.9 ± 2.0	87.3 ± 2.2	87.4 ± 2.1	90.8 ± 1.7	90.9 ± 1.	**94.0 ± 1.3**	**94.1 ± 1.3**
	40	82.5 ± 2.9	82.7 ± 2.8	84.8 ± 2.6	84.9 ± 2.5	83.8 ± 2.7	83.9 ± 2.7	89.3 ± 2.0	89.4 ± 2.	**92.8 ± 1.8**	**92.9 ± 1.8**
UNSW-NB15	10	92.8 ± 1.0	92.9 ± 1.0	94.3 ± 0.8	94.4 ± 0.8	93.8 ± 0.9	93.9 ± 0.9	95.8 ± 0.7	95.9 ± 0.	**96.8 ± 0.6**	**96.9 ± 0.6**
	20	90.2 ± 1.4	90.3 ± 1.4	91.8 ± 1.2	91.9 ± 1.2	91.3 ± 1.3	91.4 ± 1.3	93.8 ± 1.0	93.9 ± 1.	**95.8 ± 0.8**	**95.9 ± 0.8**
	30	87.7 ± 1.8	87.8 ± 1.8	89.3 ± 1.5	89.4 ± 1.5	88.8 ± 1.6	88.9 ± 1.6	92.3 ± 1.3	92.4 ± 1.	**94.8 ± 1.0**	**94.9 ± 1.0**
	40	84.8 ± 2.2	84.9 ± 2.2	86.8 ± 2.0	86.9 ± 2.0	86.3 ± 2.1	86.4 ± 2.1	89.8 ± 1.7	89.9 ± 1.	**92.8 ± 1.5**	**92.9 ± 1.5**

Note. All values as Mean ± Std. dev. (Prec.: Precision, F1: F1-score).

sensitivity analysis.

6.3.9 Varying Poisoning Attack Intensities

In this experiment, we evaluated the framework's resilience to varying levels of data poisoning in federated learning by selecting poisoning client rates of 10%, 20%, 30%, and 40%. These different rates represent a spectrum of attack intensities, ranging from mild to severe, offering valuable insights into the model's robustness. Similar methodologies have been employed in prior research, such as in [123], where poisoning rates of 10% and 30% were used, and in [191], which varied the number of compromised clients to simulate different poisoning scenarios. Furthermore, the work in [119] and [189] examined poisoning rates of up to 50%, reinforcing the use of this approach to assess the impact of adversarial manipulation on model performance.

The results presented in Table 30 show a consistent trend in all three data sets, CIC-IDS2018, BoT-IoT, and UNSW-NB15, where the precision of detection and the F1 score decrease as the intensity of the poisoning attack increases. The FedAvg shows significant performance degradation, especially at higher poisoning rates. In contrast, robust aggregation methods, Krum, WeiDetect, and Trimmed Mean, display improved resilience; however, they still experience notable accuracy losses under severe attack

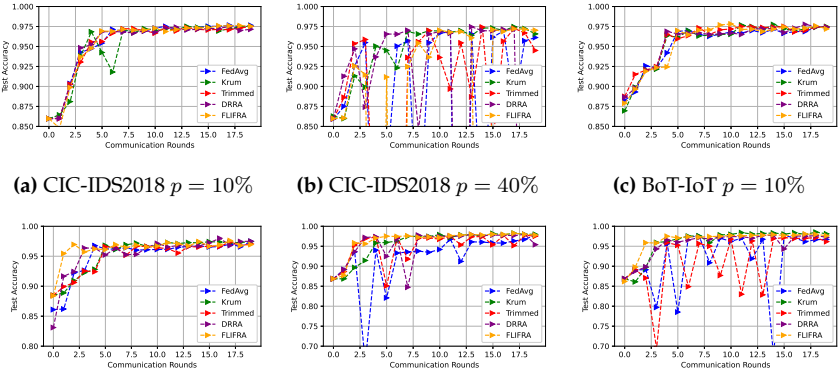
conditions (see Figure 23).

DRRA achieves higher accuracy than both FedAvg and traditional robust aggregators, indicating its effectiveness in mitigating the influence of poisoned updates. In particular, the proposed FLIFRA approach, which integrates iForest-based filtering on the client side with DRRA, consistently outperforms all other methods. Across varying poisoning intensities, the proposed method not only attains the highest detection accuracy but also exhibits lower variance, as evidenced by the smaller standard deviations. This enhanced performance underscores the advantage of the dual-layer defense mechanism in effectively countering data poisoning attacks in federated learning scenarios.

6.3.10 Convergence and Communication Round Analysis

The results in Figure 24 (one plot for each dataset poisoning-rate pair) show clear differences in convergence behavior across the evaluated methods. With lower poisoning (10%), all approaches improve progressively. However, FLIFRA attains a high accuracy much earlier than the baselines (FedAvg, Krum, and Trimmed Mean). When the poisoning rate increases to 40%, these differences become more evident. In particular, FLIFRA not only reaches a higher final (steady-state) accuracy but also maintains greater stability across communication rounds, reflected by the tighter error bands. For instance, on the BoT-IoT dataset (Figure 24), the baseline methods exhibit slower convergence and larger oscillations, whereas FLIFRA stabilizes rapidly, highlighting its robustness even under strong poisoning.

Overall, the convergence analysis suggests that FLIFRA provides notable benefits in both convergence speed and training stability compared with conventional robust aggregation schemes. These findings indicate that combining client-side anomaly detection with server-side dynamic reputation-based aggregation can effectively reduce the impact of adversarial updates, leading to higher final detection accuracy and more consistent performance over time [5].



(a) CIC-IDS2018 $p = 10\%$ **(b)** CIC-IDS2018 $p = 40\%$ **(c)** BoT-IoT $p = 10\%$

(d) BoT-IoT $p = 40\%$ **(e)** UNSW-NB15 $p = 10\%$ **(f)** UNSW-NB15 $p = 40\%$

Figure 24: Convergence behavior under poisoning attacks across communication rounds for the considered datasets, with poisoning rates of 10% and 40% [5].

6.3.11 Effect of the Dataset Heterogeneity

To simulate realistic non-IID conditions in FL, the partitioned datasets are used with a Dirichlet distribution with three distinct concentration parameters (K-values). In our study, $K = 1.0$ represents a near-IID scenario, while $K = 0.5$ and $K = 0.1$ correspond to progressively higher levels of data heterogeneity.

Table 31 summarizes the detection accuracy for each method, including FedAvg, Krum, Trimmed Mean, DRRA, and the proposed FLIFRA in the CIC-IDS2018, BoT-IoT, and UNSW-NB15 datasets with different K-values. Under the low heterogeneity condition ($K = 1.0$), all methods exhibit robust performance, with the FLIFRA approach consistently achieving the highest accuracy. As the K value decreases, indicating more skewed data distributions, a decrease in precision is observed for all methods. The proposed FLIFRA method is more resilient to the adverse effects of data heterogeneity than conventional aggregation strategies. By combining client-side anomaly detection with dynamic reputation-based aggregation, the hybrid approach not only maintains high detection ac-

Table 31: Impact of data heterogeneity (Dirichlet K-values) on detection accuracy for the considered datasets.

Dataset	K-value	Detection accuracy (%)				
		FedAvg	**Krum**	**Trimmed**	**WeiDetect**	**FLIFRA**
CIC-IDS2018	$K = 1.0$	95.0 ± 1.0	96.5 ± 0.8	96.0 ± 0.9	93.06 ± 1.0	$\mathbf{98.5 \pm 0.6}$
	$K = 0.5$	92.0 ± 1.5	94.0 ± 1.2	93.5 ± 1.3	90.09 ± 2.0	$\mathbf{97.0 \pm 0.8}$
	$K = 0.1$	88.0 ± 2.0	90.0 ± 1.8	89.5 ± 1.9	89.64 ± 1.55	$\mathbf{95.0 \pm 1.2}$
BoT-IoT	$K = 1.0$	93.0 ± 1.1	94.2 ± 0.9	94.0 ± 1.0	93.12 ± 1.6	$\mathbf{96.2 \pm 0.7}$
	$K = 0.5$	89.0 ± 1.7	91.0 ± 1.4	90.5 ± 1.5	92.81 ± 1.0	$\mathbf{95.0 \pm 1.0}$
	$K = 0.1$	85.0 ± 2.1	87.0 ± 1.9	86.5 ± 2.0	90.08 ± 1.0	$\mathbf{93.0 \pm 1.3}$
UNSW-NB15	$K = 1.0$	94.0 ± 1.0	95.0 ± 0.8	94.5 ± 0.9	92.80 ± 1.0	$\mathbf{97.05 \pm 0.6}$
	$K = 0.5$	91.0 ± 1.4	92.0 ± 1.2	91.5 ± 1.3	90.98 ± 1.15	$\mathbf{95.0 \pm 0.8}$
	$K = 0.1$	88.0 ± 1.8	89.0 ± 1.5	88.5 ± 1.6	88.22 ± 1.25	$\mathbf{93.0 \pm 1.0}$

Note. All values are presented as Mean $\pm$ Std. dev. Lower K-values correspond to higher data heterogeneity.

curacy under near-IID conditions but also exhibits minimal performance loss as the data becomes increasingly non-IID.

6.3.12 Sensitivity Analysis

A sensitivity analysis is conducted on the dual-layer poisoning-defense framework, combining client-side iForest filtering and server-side DRRA by varying the iForest contamination parameter η over $\{0.01, 0.02, 0.05\}$. For each value of η, which performs FL experiments under diverse data-poisoning attacks, recording overall TPR and FAR. The results found that larger contamination settings yield higher detection sensitivity at the cost of slightly more false alarms: at $\eta = 0.05$ achieved the best balance (TPR >95%, FAR $<$ 3%), at $\eta = 0.02$ we reduced false alarms further (FAR $<$ 1.5%) while maintaining strong detection (TPR $>$ 90%), and at $\eta = 0.01$ minimized false alarms (FAR $<$ 1%) but observed a modest drop in detection (TPR $\approx$ 85%). These results suggest that setting η between 0.02 and 0.05 delivers robust poisoning-attack filtering with a tunable trade-off between sensitivity and specificity.

Table 32: Comparison of computational overhead and convergence speed for the considered datasets.

Method	CIC-IDS2018		BoT-IoT		UNSW-NB15	
	Time (min)	Rounds	Time (min)	Rounds	Time (min)	Rounds
FedAvg	1.3 ± 0.1	57	1.2 ± 0.1	50	1.2 ± 0.1	51
Krum	1.6 ± 0.2	51	1.5 ± 0.2	48	1.5 ± 0.2	48
Trimmed Mean	1.7 ± 0.2	46	1.6 ± 0.2	47	1.6 ± 0.2	47
DRRA	1.9 ± 0.2	48	1.8 ± 0.2	45	1.8 ± 0.2	45
WeiDetect	2.4 ± 0.1	53	2.3 ± 0.1	46	2.2 ± 0.2	42
FLIFRA (proposed)	2.3 ± 0.3	39	2.2 ± 0.3	40	2.2 ± 0.3	40

Note: Values are reported as Mean ± Std. Dev., with average computation time (min/round) and rounds to convergence.

6.3.13 Computational and Communication Overhead

The results presented in Table 32 illustrate that although the proposed FLIFRA method incurs higher computational overhead per communication round – approximately 2.2 min for each round on average for the BoT-IoT dataset compared to 1.2 min for FedAvg the additional processing cost is offset by a faster convergence speed. For example, the FLIFRA approach converges in about 40 rounds on BoT-IoT, whereas FedAvg requires approximately 50 rounds. Similar trends are observed for CIC-IDS2018 and UNSW-NB15, where the hybrid method consistently converges in fewer rounds despite its higher computational expense per round. This overhead-benefit trade-off underscores the effectiveness of the FLIFRA approach in enhancing detection performance and robustness against poisoning attacks.

From a scalability perspective, although the iForest filtering on the client side of the hybrid method and the reputation computation on the server side introduce extra computation and communication overhead, these costs scale moderately with the number of clients and the size of the data set.

6.4 Summary

This study proposed a resilient FL framework to address the critical challenge of data poisoning attacks in IoT security systems. The proposed

FLIFRA approach integrates local anomaly detection using iForest with a global DRRA to enhance robustness against sophisticated adversarial attacks. The framework demonstrated superior resilience and scalability through an extensive evaluation of real-world IoT data sets, including CIC-IDS2018, BoT-IoT, and UNSW-NB15, compared to the baseline methods FedAvg, Trimmed Mean, Krum, and WeiDetect. The FLIFRA approach outperformed these methods in mitigating poisoning attacks, achieving higher accuracy and stability at varying levels of malicious client participation (10%, 20%, 30%, and 40%).

The results demonstrate the effectiveness of combining local anomaly detection with dynamic global aggregation. iForest efficiently filters suspicious updates at the client level, preventing compromised data from reaching the server, while DRRA penalizes malicious clients and prioritizes reliable updates through a reputation-based strategy. This dual-layer defense significantly enhances the robustness of the global model, particularly in heterogeneous and non-IID IoT environments. Moreover, the proposed approach ensures efficient convergence, making it well-suited for resource-constrained IoT systems that demand both computational efficiency and resilience.

Future work will explore additional attack scenarios and varying intensities, evaluate the proposed dual-layer defense in larger and more heterogeneous federated networks, and investigate real-time, privacy-preserving deployment strategies to enhance practicality in real-world environments.

Chapter 7

Conclusion and Future Work

7.1 Summary of Contributions

This thesis presents a robust, AI-based, adaptive, scalable, and privacy-preserving framework for detecting and mitigating evolving DDoS attacks in heterogeneous IoT and complex environments. The research unfolded deliberately in four stages to address concrete operational gaps observed during experimentation:

1. **Centralized DL:** In this thesis, first established strong baselines with carefully *optimized and fine-tuned* DNN/CNN/RNN/LSTM architectures to quantify the representational capacity needed for both binary and multi-class DDoS detection, and to expose bottlenecks due to class imbalance, label scarcity, and distribution shift. The proposed HPO spanned *architecture* (depth, number of filters and hidden units, and residual connections), *temporal context* (window length and sampling rate), and *training* (learning rate schedule, batch size, dropout, and label smoothing). We used efficient search strategies (random and Bayesian search with early-stopping via Hyperband) coupled with stratified cross-validation and seed averaging to obtain stable estimates. Relative to *signature-based* IDS

and *traditional anomaly* detectors over hand-crafted features, the optimized DL models consistently delivered stronger precision–recall trade-offs in high-dimensional, multi-modal traffic by *learning* hierarchical spatial–temporal patterns (1D CNN receptive fields for burst and packet-rate. The DL baselines, reinforced by calibration and cost-sensitive learning data labeling, offered more stable performance across datasets and operating points, thereby providing a principled foundation for the subsequent stages (transfer learning and privacy-preserving federated training).

2. **Transfer Learning:** To address label scarcity and enable rapid adaptation to emerging attack variants, this thesis developed a family of fine-tuning schemes for both custom CNNs and pretrained backbones (VGG, ResNet, EfficientNet, and Xception). These include (i) freezing early layers, (ii) selective re-initialization of higher blocks, (iii) mixed adaptation (partial unfreezing), (iv) last-layer retraining, (v) selective layer fine-tuning, and (vi) differential fine-tuning with layer-wise learning rates. Across datasets, these strategies reduced overfitting, shortened convergence time, and improved cross-domain generalization. Further demonstrated effective cross-domain adaptation of DDoS detection models by transferring representations learned on one dataset to distinct IoT environments with limited labels.

3. **Federated Learning:** To preserve privacy and bandwidth constraints and to cope with device heterogeneity, developed *FELACS*, a multi-criteria, multi-objective client-selection framework that balances model quality, communication cost, and fairness under both IID and non-IID (Dirichlet) partitions, while keeping raw data on-device. At each round, FELACS scores candidates by jointly considering (i) *data value* (entropy, class balance, attack-relevance proxies, and statistical variance and divergence), (ii) *systems cost* (expected compute time, energy budget, and uplink latency and size), and (iii) *participation equity* (to avoid starving weaker devices). The scheduler then selects a feasible, high-impact subset that maximizes expected

update utility subject to round-level resource budgets. By prioritizing clients with richer, attack-relevant representations rather than purely resource-centric criteria, FELACS accelerates convergence, improves DDoS detection accuracy, and enhances responsiveness for real-time IoT threat monitoring, all while preserving privacy and promoting fairness across heterogeneous nodes.

4. **Poisoning-Resilient FL:** To secure collaborative training in adversarial IoT settings, this thesis introduces a dual-layer defense framework, *FLIFRA*, that integrates client-side anomaly screening with server-side trust-aware aggregation. At the edge, iForest scrutinizes local training data and model updates to flag anomalous behavior before transmission; centrally, a DRRA mechanism adaptively weights or excludes client contributions based on historical reliability, update consistency, and their effect on convergence. To formalize the poisoning defense problem for federated DDoS detection under non-IID partitions and stringent resource constraints, which demonstrates, on real IoT intrusion datasets and across a spectrum of poisoning intensities and attack models, that this defense-in-depth design restores global performance more effectively than standard robust aggregators while maintaining acceptable FPR for honest clients and modest systems overhead. (Implementation available at:[1].)

Why the staged progression (DL $\rightarrow$ TL $\rightarrow$ FL $\rightarrow$ robust FL). Empirically, tuned centralized DL provides the representational backbone but remains *fragile* to label scarcity, imbalance, and drift. TL mitigates these weaknesses by reusing transferable flow representations, *cutting label demand and time-to-adaptation* when attack patterns evolve. FL then becomes *necessary* for real deployments: it preserves privacy by data locality, respects bandwidth and compute limits, and via FELACS, achieves better accuracy cost fairness trade-offs under non-IID sampling than uniform participation. However, collaborative training is *vulnerable* to malicious

[1] `https://github.com/mulerkal/FLIFRA`

clients; hence, the final stage hardens the system with defense-in-depth client-side anomaly screening plus trust-aware aggregation (FLIFRA), which demonstrably restores global performance under poisoning. Each step, therefore, removes the dominant failure mode revealed by the previous one, culminating in a secure, adaptable, and deployable IoT DDoS detection pipeline.

7.2 Research Questions and the Discussions

RQ1. *What detection accuracy do centralized DL architectures (DNN, CNN, RNN, LSTM), tuned via HPO, achieve on binary (attack vs. benign) and multi-class (specific DDoS types) classification?*
Under thorough HPO, 1D–CNNs and lightweight LSTMs achieved the strongest centralized baselines on flow-based tabular inputs. *Binary detection* attained high AUROC with low FNR once decision thresholds were calibrated; *multi-class recognition* showed lower accuracy for low-sample attack types (e.g., in CIC-DDoS2019), where class-weighted training improved minority-class recall. Generalization was most sensitive to (i) receptive-field design (kernel size, dilation) and moderate depth to capture short- to mid-range temporal correlations, (ii) learning-rate scheduling with decoupled weight decay to stabilize optimization, and (iii) regularization (e.g., dropout) to curb overfitting under class skew. RNN and LSTM stacks converged more slowly and tended to be under-calibrated on tabular flows unless front-ended by convolutional blocks. Overall, centralized DL provides a reliable foundation, yet its effectiveness is ultimately bounded by label scarcity, dataset bias, privacy constraints, and distribution drift—motivating transfer learning and federated learning.

RQ2. *How can transfer learning via targeted fine-tuning strategies be optimized to detect evolving DDoS patterns with limited labels and minimal retraining, while maximizing accuracy and computational efficiency on resource-constrained devices?*
Selective layer adaptation (freezing early layers, re-initializing higher layers, or mixed strategies) reduced convergence time and overfitting rela-

tive to training from scratch, while improving cross-domain performance when feature schemas and traffic statistics diverged across datasets. TL preserved useful low-level temporal filters and delivered stable gains under label scarcity, validating TL as a practical bridge between environments and as a mechanism for rapid adaptation to novel attacks.

RQ3. *How can FL (client selection + aggregation) preserve privacy, sustain accuracy under IID and non-IID data, minimize communication, and maintain fairness across heterogeneous IoT nodes?*
The multi-objective client-selection approaches improved convergence stability and fairness under non-IID partitions by prioritizing (i) device capability, (ii) data representativeness, diversity, and (iii) communication budget. Compared with uniform participation, the approach reduced rounds-to-target and avoided starving weaker devices, while update compression and scheduled participation curbed bandwidth without harming accuracy. Thus, FL can match centralized and TL baselines under realistic constraints while preserving data confidentiality.

RQ4. *How effective is a dual-layer defense combining local anomaly filtering with global trust-based aggregation in identifying and mitigating poisoning attacks in federated IoT DDoS detection, and what is its impact on overall robustness and privacy?*
Client-side iForests filtered anomalous updates pre-aggregation, reducing the adversarial surface, while DRRA down-weighted persistently suspicious clients via dynamic reputation. In controlled poisoning experiments, the dual-layer approach more reliably restored global performance than single-layer robust aggregators (FedAvg, Krum, Trimmed Mean, WeiDetect), with manageable false positives for honest clients. Defense-in-depth is therefore validated as necessary for secure collaborative training.

7.3 Theoretical and Practical Implications

Theoretical. (i) Transferable flow representations learned on related intrusion datasets act as effective regularizers, supporting out-of-domain robustness; (ii) multi-objective client selection operationalizes a Pareto-

efficient trade-off among accuracy, communication cost, and fairness under non-IID sampling; (iii) combining pre-aggregation filtering with trust-aware aggregation yields robustness unattainable by either layer alone, highlighting complementary error-cancellation effects.
Practical. For IoT operators, a deployable blueprint emerges: establish a tuned centralized baseline; adapt rapidly with TL when labels are scarce, or domains shift; migrate to FL for privacy and scalability; and harden FL with client-side anomaly filtering plus reputation-aware aggregation. This sequence aligns with real constraints (privacy, bandwidth, heterogeneity) while maintaining detection quality under evolving threats.

7.4 Limitations

Residual challenges remain the following (see also Section 3.3.5):

- **Device computation constraints.** The proposed deeply adaptive DL approaches achieved strong detection accuracy offline, but realizing these gains on IoT-edge hardware requires substantially more compute and memory than many devices can provide. In realistic deployments, limited CPU throughput, memory footprint, and energy budgets constrain both feature extraction and inference. High-capacity CNN and LSTM backbones that perform well on workstations may neither fit on-device nor meet per-packet and window latency requirements without model compression (quantization, pruning, distillation) and device-specific runtimes. These optimizations can shift the probability calibration and slightly reduce the recall of rare classes.

- **Transfer-learning generalization.** In the proposed work introduced multiple fine-tuning strategies, last layer retraining, selective freezing, mixed adaptation, and differential (layer-wise) learning rates were introduced, and also projected tabular flow features into images to exploit pretrained backbones (VGG/ResNet/EfficientNet). These choices improved data efficiency and in-domain accuracy. However, cross-domain generalization remained limited: models

fine-tuned on one dataset transferred imperfectly to others without a small amount of target-domain adaptation. The degradation was most pronounced in multiclass settings and for evolving attack families, where covariate and label shifts, as well as schema mismatches (including feature definitions, scales, and temporal statistics), dominated errors.

- **Non-IID heterogeneity and fairness in FL.** FELACS improved convergence and reduced communication under non-IID partitions by prioritizing high-utility clients (informative data, sufficient resources). However, when participation was intermittent and attack modes were rare, some clients were systematically under-selected, leaving their local patterns underrepresented in the global model.

- **Robustness in poisoning defense.** FLIFRA (iForest + DRRA) consistently restored performance under data or model poisoning; however, aggressive anomaly thresholds or rapid reputation decay occasionally down-weighted honest but atypical updates. Tuning detection windows and decay rates reduced false positives, at the cost of slower recovery under strong contamination. Our evaluation did not cover adaptive or colluding adversaries that jointly manipulate local statistics and reputation, which remains an open threat model.

7.5 Future Work

- **Hardware-aware deployment and compression-robust calibration.** The next phase moves beyond workstation experiments to on-device evaluation, co-designing models with the target IoT hardware. Building on the deep-learning and transfer-learning baselines and FELACS scheduling, the plan is to apply (i) run latency and energy profiling on representative devices, (ii) apply layer pruning under explicit memory and energy constraints, and (iii) develop *compression-robust calibration* to correct probability shifts introduced by quantization and distillation. An edge–cloud split with early-exit

cascades will be engineered to meet per-packet deadlines while preserving detection quality.

- **Continual and domain-adaptive learning under drift.** To address the cross-domain brittleness observed even with selective fine-tuning, the approach extends transfer learning to continual and test-time adaptation. It incorporates lightweight feature-alignment mechanisms, streaming normalization matching, and regularized parameter updates designed to preserve probability calibration under evolving benign and attack traffic.

- **Fairness-and staleness-aware client selection in FL.** FELACS improved convergence under non-IID data but left a fairness–utility tension for rare-mode and low-resource clients. It will upgrade FELACS to a *constrained, exploration–exploitation* scheduler that enforces minimum exposure for underrepresented modes and accounts for late updates. The objective will jointly optimize accuracy, bandwidth, and a fairness metric (worst-client), with guarantees on participation over horizons relevant to IoT or edge.

- **Adaptive and collusion-resilient robust FL.** FLIFRA restored accuracy under static poisoning, but can over-penalize atypical yet honest clients and does not cover adaptive colluding adversaries. It will extend DRRA with *tempered* reputation decay, cross-round consistency checks, and collusion detectors, and study its interplay with secure aggregation and differential privacy. Benchmarks will include coordinated backdoor attacks under non-IID data to validate resilience without excessive false positives.

7.6 Final Remarks

This thesis demonstrates that robust, adaptive, AI-based, accurate, efficient, and privacy-preserving DDoS detection in complex IoT ecosystems is achievable through a principled, staged approach. Centralized DL provides the representational backbone; TL enables rapid adaptation

with limited labels; FL preserves privacy and scales across heterogeneous nodes; and dual-layer defenses secure collaborative training against poisoning. While challenges persist around drift, cross-domain transfer, and adversarial robustness, the framework, analyses, and artifacts presented here offer a practical foundation for securing next-generation connected systems and a clear path for continued research.

Bibliography

[1] M. B. Anley, A. Genovese, and V. Piuri, "Deep learning for ddos attack detection in iot: A survey," in *International Conference on Security and Cryptography*. Springer, 2023, pp. 99–121.

[2] M. B. Anley, A. Genovese, V. Piuri *et al.*, "Transferedge: Transfer learning approach to detect evolving ddos threats in edge-iiot," in *2025 IEEE 9th Forum on Research and Technologies for Society and Industry (RTSI)*. IEEE, 2025, pp. 11–16.

[3] M. B. Anley, A. Genovese, D. Agostinello, and V. Piuri, "Robust ddos attack detection with adaptive transfer learning," *Computers & Security*, vol. 144, p. 103962, 2024.

[4] M. B. Anley, P. Coscia, A. Genovese, and V. Piuri, "Felacs: Federated learning with adaptive client selection for iot ddos attack detection," *Computers & Security*, p. 104642, 2025.

[5] M. B. Anley, A. Genovese, T. B. Tesema, and V. Piuri, "Flifra: Hybrid data poisoning attack detection in federated learning for iot security," *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 6816–6823, 2025.

[6] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in iot: a survey," *The Journal of Supercomputing*, vol. 76, pp. 5320–5363, 2020.

[7] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art," *arXiv preprint arXiv:1208.0403*, 2012.

[8] B. B. Gupta, R. C. Joshi, and M. Misra, "Defending against distributed denial of service attacks: issues and challenges," *Information Security Journal: A Global Perspective*, vol. 18, no. 5, pp. 224–247, 2009.

[9] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into internet ddos attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2843–2855, 2018.

[10] European Union Agency for Cybersecurity (ENISA), "ENISA Threat Landscape for DoS Attacks," European Union Agency for Cybersecurity (ENISA), Technical Report, Nov. 2023, january 2022 to August 2023.

[11] E. Osterweil, A. Stavrou, and L. Zhang, "20 years of ddos: a call to action," *CoRR*, vol. abs/1904.02739, 2019.

[12] T. Jung, "The most notorious ddos attacks in history– 2021 update," 2021. [Online]. Available: https://www.cloudbric.com/the-most-notorious-ddos-attacks2021

[13] Cloudflare, "Cloudflare ddos threat report 2022 q3," 2022. [Online]. Available: https://blog.cloudflare.com/cloudflare-ddos-threat-report-2022-q3/

[14] R. R. Brooks, L. Yu, I. Ozcelik, J. Oakley, and N. Tusing, "Distributed denial of service (ddos): a history," *IEEE Annals of the History of Computing*, vol. 44, no. 2, pp. 44–54, 2021.

[15] Google Cloud, "How google cloud blocked largest layer 7 ddos attack," 2024. [Online]. Available: https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-

[16] Center, CERT Coordination, "Certr incident note in-99-07 distributed denial of service tools," *CERT Coordination Center, Pittsburgh, Incident Note IN-99-07*, 1999.

[17] ENISA, "Enisa threat landscape: Cyber espionage," 2020, from January 2019 to April 2020.

[18] H. Orman, "The morris worm: A fifteen-year perspective," *IEEE Security & Privacy*, vol. 1, no. 5, pp. 35–43, 2003.

[19] J. Pagliery, "Mafiaboy: The infamous teenage hacker who took down the internet," 2011. [Online]. Available: http://edition.cnn.com/2011/TECH/web/08/15/mafiaboy.hacker/index.html

[20] A. Apostu, S. Gheorghe, A. Hîji, N. Cleju, A. Pătraşcu, C. Rusu, R. Ionescu, and P. Irofti, "Detecting and mitigating ddos attacks with ai: A survey," *arXiv preprint arXiv:2503.17867*, 2025.

[21] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[22] M. S. Blog. Microsoft response to layer 7 ddos. [Online]. Available: https://msrc.microsoft.com/blog/2023/06/microsoft-response-to-layer-7-ddos-attacks

[23] A. Greenberg. The cheap radio hack disrupted poland's railway system. [Online]. Available: https://www.wired.com/story/poland-train-radio-stop-attack/

[24] C. Sparling and S. Rath. (2023, Sep.) Record-breaking ddos in apac. [Online]. Available: https://www.akamai.com/blog/security/record-breaking-ddos-in-apac

[25] J. Greig. (2023) Killnet ddos attacks on u.s. hospitals. [Online]. Available: https://therecord.media/ddos-hospitals-cisa-killnet-limited-effects

[26] R. Tandon, "A Survey of Distributed Denial of Service Attacks and Defenses," *arXiv.org*, 2020.

[27] J. Nazario, C. Czosseck, and K. Geers, "Politically motivated denial of service attacks," 2009.

[28] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[29] V. Chauhan and P. Saini, "Icmp flood attacks: A vulnerability analysis," in *Cyber Security: Proceedings of CSI 2015*. Springer, 2018, pp. 261–268.

[30] I. Oyekunle, "What are the types of ddos attacks?" September 21 2021. [Online]. Available: https://securitygladiators.com/threat/ddos/type/

[31] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

[32] A. Praseed and P. S. Thilagam, "Ddos attacks at the application layer: Challenges and research perspectives for safeguarding web applications," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661–685, 2018.

[33] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn-based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 493–501, 2019.

[34] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30 387–30 399, 2020.

[35] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020.

[36] L. Yang and A. Shami, "A transfer learning and optimized cnn based intrusion detection system for internet of vehicles," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 2774–2779.

[37] A. S. Dina, A. B. Siddique, and D. Manivannan, "Fs3: Few-shot and self-supervised framework for efficient intrusion detection in internet of things networks," in *Proceedings of the 39th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2023, p. 138–149.

[38] J. Li, Z. Zhang, Y. Li, X. Guo, and H. Li, "Fids: Detecting ddos through federated learning based method," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 856–862.

[39] J. Zhang, P. Yu, L. Qi, S. Liu, H. Zhang, and J. Zhang, "Flddos: Ddos attack detection model based on federated learning," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 635–642.

[40] R. Doriguzzi-Corin and D. Siracusa, "Flad: adaptive federated learning for ddos attack detection," *Computers & Security*, vol. 137, p. 103597, 2024.

[41] T. Cai, T. Jia, S. Adepu, Y. Li, and Z. Yang, "Adam: an adaptive ddos attack mitigation scheme in software-defined cyber-physical system," *IEEE Transactions on Industrial Informatics*, 2023.

[42] D. Agostinello, A. Genovese, and V. Piuri, "Anomaly-based intrusion detection system for ddos attack with deep learning techniques," in *Proceedings of the 20th International Conference on Security and Cryptography. 1*. SCITEPRESS, 2023, pp. 267–275.

[43] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting ddos attacks: A systematic review," *Soft Computing*, vol. 27, no. 18, pp. 13 039–13 075, 2023.

[44] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*. IEEE, 2019, pp. 1–6.

[45] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of ddos attacks with feed forward based deep neural network model," *Expert Systems with Applications*, vol. 169, p. 114520, 2021.

[46] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. Narayan, "Distributed denial of service (ddos) attacks detection system for openstack-based private cloud," *Procedia Computer Science*, vol. 167, pp. 2297–2307, 2020.

[47] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, and J. Iqbal, "A deep cnn ensemble framework for efficient ddos attack detection in software-defined networks," *Ieee Access*, vol. 8, pp. 53 972–53 983, 2020.

[48] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020.

[49] M. V. de Assis, L. F. Carvalho, J. J. Rodrigues, J. Lloret, and M. L. Proen Jr, "Near real-time security system applied to sdn environments in iot networks using cnn," *Computers & Electrical Engineering*, 2020.

[50] L. Wang and Y. Liu, "A ddos attack detection method based on information entropy and deep learning in sdn," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 1084–1088.

[51] W. Wei, X. Liu, C. Sheng, and A. Feng, "A ddos attack traffic classification model for industrial internet based on cnn-lstm," in *2022 China Automation Congress (CAC)*. IEEE, 2022, pp. 3443–3448.

[52] A. A. Cunha, J. B. Borges, and A. A. Loureiro, "Classification of botnet attacks in iot using a convolutional neural network," in *Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, 2022, pp. 63–70.

[53] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[54] M. Li, B. Zhang, G. Wang, B. ZhuGe, X. Jiang, and L. Dong, "A ddos attack detection method based on deep learning two-level model cnn-lstm in sdn network," in *2022 International Conference on Cloud Computing, Big Data Applications and Software Engineering (CBASE)*. IEEE, 2022, pp. 282–287.

[55] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, "Deep learning-based intrusion detection for distributed denial of service attack in agriculture 4.0," *Electronics*, vol. 10, no. 11, p. 1257, 2021.

[56] X. Liang and T. Znati, "A long short-term memory enabled framework for ddos detection," in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[57] Y. Zhang, Y. Liu, Y. Zhang, L. Han, J. Zhao, and Y. Wu, "A ddos attack detection method based on lstm neural network in the internet of vehicles," in *Proceedings of the 4th International Conference on Information Technologies and Electrical Engineering*, 2021, pp. 1–5.

[58] V. Hnamte, H. Nhung-Nguyen, J. Hussain, and Y. Hwa-Kim, "A novel two-stage deep learning model for network intrusion detection: Lstm-ae," *IEEE Access*, 2023.

[59] C.-s. Wu and S. Chen, "A heuristic intrusion detection approach using deep learning model," in *2023 International Conference on Information Networking (ICOIN)*. IEEE, 2023, pp. 438–442.

[60] S. Yeom, C. Choi, and K. Kim, "Lstm-based collaborative source-side ddos attack detection," *IEEE Access*, vol. 10, pp. 44 033–44 045, 2022.

[61] H. Aydın, Z. Orman, and M. A. Aydın, "A long short-term memory (lstm)-based distributed denial of service (ddos) detection and

defense system design in public cloud network environment," *Computers & Security*, vol. 118, p. 102725, 2022.

[62] M. V. Assis, L. F. Carvalho, J. Lloret, and M. L. Proença Jr, "A gru deep learning system against attacks in software defined networks," *Journal of Network and Computer Applications*, vol. 177, p. 102942, 2021.

[63] S. ur Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, Z. Jalil, and A. K. Bashir, "Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru)," *Future Generation Computer Systems*, vol. 118, pp. 453–466, 2021.

[64] D. M. B. Lent, M. P. Novaes, L. F. Carvalho, J. Lloret, J. J. Rodrigues, and M. L. Proença, "A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks," *IEEE Access*, vol. 10, pp. 73 229–73 242, 2022.

[65] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Computer Networks*, vol. 180, p. 107390, 2020.

[66] A. Bhardwaj, V. Mangat, and et al, "Hyperband tuned deep neural network with well posed stacked sparse autoencoder for detection of ddos attacks in cloud," *IEEE Access*, vol. 8, pp. 181 916–181 929, 2020.

[67] A. A. Elsaeidy, A. Jamalipour, and K. S. Munasinghe, "A hybrid deep learning approach for replay and ddos attack detection in a smart city," *IEEE Access*, vol. 9, pp. 154 864–154 875, 2021.

[68] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "Ae-mlp: A hybrid deep learning approach for ddos detection and classification," *IEEE Access*, vol. 9, pp. 146 810–146 821, 2021.

[69] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol. 10, p. 100053, 2023.

[70] M. Roopak, G.-Y. Tian, and J. A. Chambers, "An intrusion detection system against ddos attacks in iot networks," *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0562–0567, 2020.

[71] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[72] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[73] S. Makar, A. Dehghantanha, F. Zarrinkalam, G. Srivastava, and A. Yazdinejad, "Systemization of knowledge (sok)-cross impact of transfer learning in cybersecurity: Offensive, defensive and threat intelligence perspectives," *arXiv preprint arXiv:2309.05889*, 2023.

[74] F. Yu, X. Xiu, and Y. Li, "A survey on deep transfer learning and beyond," *Mathematics*, vol. 10, no. 19, p. 3619, 2022.

[75] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*. Springer, 2016, pp. 443–450.

[76] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," *Advances in neural information processing systems*, vol. 19, 2006.

[77] A. Sicilia, X. Zhao, and S. J. Hwang, "Domain adversarial neural networks for domain generalization: When it works and how to improve," *Machine Learning*, vol. 112, no. 7, pp. 2685–2721, 2023.

[78] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation." *Journal of Machine Learning Research*, vol. 8, no. 5, 2007.

[79] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 1855–1862.

[80] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift," *Frontiers in Artificial Intelligence*, vol. 7, p. 1330257, 2024.

[81] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[82] S. Layeghy, M. Baktashmotlagh, and M. Portmann, "Di-nids: Domain invariant network intrusion detection system," *Knowledge-Based Systems*, vol. 273, p. 110626, 2023.

[83] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM workshop on artificial intelligence and security*, 2021, pp. 111–122.

[84] A. S. Li, A. Iyengar, A. Kundu, and E. Bertino, "Transfer learning for security: Challenges and future directions," *arXiv preprint arXiv:2403.00935*, 2024.

[85] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering edge intelligence: A comprehensive survey on on-device ai models," *ACM Computing Surveys*, vol. 57, no. 9, pp. 1–39, 2025.

[86] H. Lu, Y. Zhao, Y. Song, Y. Yang, G. He, H. Yu, and Y. Ren, "A transfer learning-based intrusion detection system for zero-day attack in communication-based train control system," *Cluster Computing*, vol. 27, no. 6, pp. 8477–8492, 2024.

[87] S. Khanal, S. Tirupathi, M. Dzaferagic, and T. B. Pedersen, "Addressing data scarcity and distribution shifts in communication networks using pre-trained transformers and transfer learning," in *Proc. of AAAI*, 2025.

[88] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep transfer learning for iot attack detection," *IEEE Access*, vol. 8, pp. 107 335–107 344, 2020.

[89] B. Farzaneh, N. Shahriar, A. H. Al Muktadir, M. S. Towhid, and M. S. Khosravani, "DTL-5G: Deep transfer learning-based ddos attack detection in 5g and beyond networks," *Computer Communications*, vol. 228, p. 107927, 2024.

[90] F. Yan, G. Zhang, D. Zhang, X. Sun, B. Hou, and N. Yu, "Tl-cnn-ids: transfer learning-based intrusion detection system using convolutional neural network," *The Journal of Supercomputing*, vol. 79, no. 15, pp. 17 562–17 584, 2023.

[91] S. Latif, W. Boulila, A. Koubaa, Z. Zou, and J. Ahmad, "DTL-IDS: An optimized intrusion detection framework using deep transfer learning and genetic algorithm," *Journal of Network and Computer Applications*, vol. 221, p. 103784, 2024.

[92] O. Jouini, K. Sethom, A. Namoun, N. Aljohani, M. H. Alanazi, and M. N. Alanazi, "A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions," *Technologies*, vol. 12, no. 6, p. 81, 2024.

[93] M. Ishtiaq, N. Saeed, and M. A. Khan, "Edge computing in iot: A 6g perspective," *arXiv preprint arXiv:2111.08943*, 2021.

[94] M. I. Pavel, S. Hu, M. Pratama, and R. Kowalczyk, "Onboard optimization and learning: A survey," *arXiv preprint arXiv:2505.08793*, 2025.

[95] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "IoTDefender: A federated transfer learning intrusion detection framework for 5G IoT," in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*. IEEE, 2020, proc. BigDataSE 2020.

[96] W. Guo, F. Zhuang, X. Zhang, Y. Tong, and J. Dong, "A comprehensive survey of federated transfer learning: challenges, methods and applications," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186356, 2024.

[97] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2687–2700, 2021.

[98] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.

[99] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[100] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE communications surveys & tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[101] S. Roy, J. Li, and Y. Bai, "Federated learning-based intrusion detection system for iot environments with locally adapted model," in *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2023, pp. 203–209.

[102] P. Ruzafa-Alcázar, P. Fernández-Saura, E. Mármol-Campos, A. González-Vidal, J. L. Hernández-Ramos, J. Bernal-Bernabe, and A. F. Skarmeta, "Intrusion detection based on privacy-preserving federated learning for the industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1145–1154, 2021.

[103] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022.

[104] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[105] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[106] K. Bonawitz, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[107] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440*, 2020.

[108] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[109] H. Wu, X. Tang, Y.-J. A. Zhang, and L. Gao, "Incentive mechanism for federated learning with random client selection," *IEEE Transactions on Network Science and Engineering*, 2023.

[110] A. Ginanjar, X. Li, P. Singh, and W. Hua, "Random client selection on contrastive federated learning for tabular data," in *arXiv:2505.10759*, 2025, pp. 1–5.

[111] S. Seo, J. Lee, H. Ko, and S. Pack, "Performance-aware client and quantization level selection algorithm for fast fl," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1892–1897.

[112] F. Maciel, A. M. De Souza, L. F. Bittencourt, and L. A. Villas, "Resource aware client selection for federated learning in iot scenarios," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2023, pp. 1–8.

[113] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[114] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, "Fedmccs: Multicriteria client selection model for optimal iot federated learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2020.

[115] K. H. Li, P. P. B. de Gusmão, D. J. Beutel, and N. D. Lane, "Secure aggregation for federated learning in flower," in *Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning*, 2021, pp. 8–14.

[116] Y. Xuan, X. Chen, Z. Zhao, B. Tang, and Y. Dong, "Practical and general backdoor attacks against vertical federated learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 402–417.

[117] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," *arXiv preprint arXiv:2207.01982*, 2022.

[118] X. Zhang, J. Zhang, K.-H. Chow, J. Chen, Y. Mao, and M. Rahouti, "Visualizing the shadows: Unveiling data poisoning behaviors in federated learning," *arXiv:2405.16707*, 2024.

[119] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25*. Springer, 2020, pp. 480–501.

[120] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023.

[121] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, 2021.

[122] R. Xu, S. Gao, C. Li, J. Joshi, and J. Li, "Dual defense: Enhancing privacy and mitigating poisoning attacks in federated learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 70 476–70 498, 2024.

[123] M. Fang, X. Cao, and e. a. Jia, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[124] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective," *Advances in neural information processing systems*, vol. 34, pp. 12 613–12 624, 2021.

[125] D. N. Yaldiz, T. Zhang, and S. Avestimehr, "Secure federated learning against model poisoning attacks via client filtering," *arXiv preprint arXiv:2304.00160*, 2023.

[126] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[127] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[128] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International conference on machine learning*. Pmlr, 2018, pp. 5650–5659.

[129] T. D. Luong, V. M. Tien, N. H. Quyen, P. T. Duy, V.-H. Pham *et al.*, "Fed-lsae: Thwarting poisoning attacks against federated cyber threat detection system via autoencoder-based latent space inspection," *Journal of Information Security and Applications*, vol. 87, p. 103916, 2024.

[130] H. Fereidooni, A. Pegoraro, P. Rieger, A. Dmitrienko, and A.-R. Sadeghi, "Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning," *arXiv preprint arXiv:2312.04432*, 2023.

[131] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *ACM SIGKDD explorations newsletter*, vol. 2, no. 2, pp. 81–85, 2000.

[132] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[133] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[134] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[135] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[136] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[137] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," 2023.

[138] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[139] J. M. Peterson, J. L. Leevy, and T. M. Khoshgoftaar, "A review and analysis of the bot-iot dataset," in *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2021, pp. 20–27.

[140] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: network ton_iot datasets. sustain. cities soc. 72, 102994 (2021)," 2021.

[141] X. Fu, S. Lou, J. Zheng, C. Chi, J. Yang, D. Wang, C. Zhu, B. Huang, and X. Zhu, "Deep learning techniques for ddos attack detection: Concepts, analyses, challenges, and future directions," *Expert Systems with Applications*, vol. 291, p. 128469, 2025.

[142] A. Alfatemi, M. Rahouti, R. Amin, S. ALJamal, K. Xiong, and Y. Xin, "Advancing ddos attack detection: A synergistic approach using deep residual neural networks and synthetic oversampling," *arXiv preprint arXiv:2401.03116*, 2024.

[143] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[144] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, 2020.

[145] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[146] J. Chen, Y.-t. Yang, K.-k. Hu, H.-b. Zheng, and Z. Wang, "Dad-mcnn: Ddos attack detection via multi-channel cnn," in *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, 2019, pp. 484–488.

[147] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1717–1731, 2020.

[148] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Software Networking*, vol. 2017, no. 1, pp. 177–200, 2017.

[149] P. Aggarwal and S. K. Sharma, "Analysis of kdd dataset attributes-class wise for intrusion detection," *Procedia Computer Science*, vol. 57, pp. 842–851, 2015.

[150] D. Akgun, S. Hizal, and U. Cavusoglu, "A new ddos attacks intrusion detection model based on deep learning for cybersecurity," *Computers & Security*, vol. 118, p. 102748, 2022.

[151] C. E. Ogbuanya, "Improved Dimensionality Reduction of Various Datasets Using Novel Multiplicative Factoring Principal Component Analysis (MPCA)," *International Journal of Computer and Communication Engineering*, vol. 10, no. 4, pp. 85–95, 2021.

[152] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "A DDoS attack mitigation framework for IoT networks using fog computing," *Procedia Computer Science*, vol. 182, pp. 13–20, 2021.

[153] P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection," in *2019 IEEE 4th international conference on big data analytics (ICBDA)*. IEEE, 2019, pp. 281–285.

[154] B. Farzaneh, N. Shahriar, A. H. Al Muktadir, and M. S. Towhid, "DTL-IDS: Deep transfer learning-based intrusion detection system in 5G networks," in *Proc. of CNSM*, 2023, pp. 1–5.

[155] A. A. Najar and S. M. Naik, "Cyber-secure SDN: A CNN-based approach for efficient detection and mitigation of DDoS attacks," *Computers & Security*, vol. 139, p. 103716, 2024.

[156] M. Muhammad, A. S. Alshra'a, and R. German, "An IDS for DDoS attacks in SDN using VGG-based CNN architecture," in *Proc. of SoftCOM*, 2023, pp. 1–7.

[157] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[158] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of CVPR*, 2016, pp. 770–778.

[159] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. of ICML*, 2019, pp. 6105–6114.

[160] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. of CVPR*, 2017, pp. 1251–1258.

[161] E. Yvinec, A. Dapogny, K. Bailly, and X. Fischer, "Safer: Layer-level sensitivity assessment for efficient and robust neural network inference," *arXiv preprint arXiv:2308.04753*, 2023.

[162] A. Chartuni and J. Márquez, "Multi-classifier of ddos attacks in computer networks built on neural networks," *Applied Sciences*, vol. 11, no. 22, p. 10609, 2021.

[163] O. Okey, D. C. Melgarejo, M. Saadi, R. L. Rosa, J. H. Kleinschmidt, and D. Z. Rodríguez, "Transfer learning approach to ids on cloud iot devices using optimized cnn," *IEEE Access*, vol. 11, pp. 1023–1038, 2023.

[164] J. Du, K. Yang, Y. Hu, and L. Jiang, "Nids-cnnlstm: Network intrusion detection classification model based on deep learning," *IEEE Access*, vol. 11, pp. 24 808–24 821, 2023.

[165] A. S. Almogren, "Intrusion detection in edge-of-things computing," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 259–265, 2020.

[166] A. Pakmehr, A. Aßmuth, N. Taheri, and A. Ghaffari, "Ddos attack detection techniques in IoT networks: a survey," *Cluster Computing*, vol. 27, no. 10, pp. 14 637–14 668, 2024.

[167] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1717–1731, June 2021, 1937-9234.

[168] S. Aktar and A. Y. Nur, "Towards DDoS attack detection using deep learning approach," *Computers & Security*, vol. 129, p. 103251, 2023.

[169] Q. H. Nguyen, S. Hore, A. Shah, T. Le, and N. D. Bastian, "FedNIDS: A federated learning framework for packet-based network intrusion detection system," *Digital Threats: Research and Practice*, vol. 6, no. 1, pp. 1–23, 2025.

[170] A. Mazid, S. Kirmani, Manaullah, and M. Yadav, "FL-IDPP: A federated learning based intrusion detection approach with privacy preservation," *Trans. on Emerging Telecommunications Technologies*, vol. 36, no. 1, p. e70039, 2025.

[171] M. Rahmati, "Federated learning-driven cybersecurity framework for IoT networks with privacy-preserving and real-time threat detection capabilities," in *arXiv:2502.10599*, 2025, pp. 1–16.

[172] J. Li, T. Chen, and S. Teng, "A comprehensive survey on client selection strategies in federated learning," *Computer Networks*, vol. 251, no. 110663, pp. 1–15, 2024.

[173] L. Fu, H. Zhang, G. Gao, M. Zhang, and X. Liu, "Client selection in federated learning: Principles, challenges, and opportunities," *IEEE Internet of Things Journal*, 2023.

[174] S. Sun, P. Sharma, K. Nwodo, A. Stavrou, and H. Wang, "FedMADE: Robust federated learning for intrusion detection in IoT networks using a dynamic aggregation method," in *Proc. of ICIS*, 2024, pp. 286–306.

[175] J. Zhang, S. Li, and C. Wang, "Utility aware optimal data selection for differentially private federated learning in iov," *IEEE Internet of Things Journal*, 2024.

[176] R. Anand, D. Aggarwal, and V. Kumar, "A comparative analysis of optimization solvers," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 623–635, 2017.

[177] J. Puchinger, G. R. Raidl, and U. Pferschy, "The multidimensional knapsack problem: Structure and algorithms," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250–265, 2010.

[178] M. A. Elfaki, H. M. Alshahrani, K. Mahmood, R. Alabdan, M. Alymani, H. Alshahrani, A. Motwakel, and A. A. Alneil, "Metaheuristics algorithm-based minimization of communication costs in federated learning," *IEEE Access*, vol. 11, pp. 81 310–81 317, 2023.

[179] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Federated learning meets multi-objective optimization," *IEEE Trans. on Network Science and Engineering*, vol. 9, no. 4, pp. 2039–2051, 2022.

[180] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[181] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[182] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *Proc. of ICAIS*. PMLR, 2021, pp. 3403–3411.

[183] Y. Li and X. Lyu, "Convergence analysis of sequential federated learning on heterogeneous data," in *Proc. of NIPS*, vol. 36, 2023, pp. 56 700–56 755.

[184] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[185] S. Wang and M. Ji, "A unified analysis of federated learning with arbitrary client participation," in *Proc. of NIPS*, vol. 35, 2022, pp. 19 124–19 137.

[186] H. Yang, Z. Liu, J. Liu, C. Dong, and M. Momma, "Federated multi-objective learning," *Advances in neural information processing systems*, vol. 36, pp. 39 602–39 625, 2023.

[187] H. Vardhan, X. Yu, T. Rosing, and A. Mazumdar, "Client selection in federated learning with data heterogeneity and network latencies," in *arXiv:2504.01921*, 2025, pp. 1–23.

[188] D.-P. Khuu, M. Sober, D. Kaaser, M. Fischer, and S. Schulte, "Data poisoning detection in federated learning," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024, pp. 1549–1558.

[189] A. Rocha, M. Conti, and et al, "Weidetect: Weibull distribution-based defense against poisoning attacks in federated learning for network intrusion detection systems," *arXiv preprint arXiv:2504.04367*, 2025.

[190] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX security symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[191] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.