THE UNIVERSITY OF PARDUBICE

FACULTY OF ECONOMICS AND PUBLIC ADMINISTRATION

BACHELOR'S THESIS

2025 Tawanda N. Mashoko

University of Pardubice

Faculty of Economics and Administration

Cyber-attack Detection in IoT Networks using Deep Learning

Bachelor's Thesis

2025 Tawanda Nduna Mashoko

University of Pardubice
Faculty of Economics and Administration
Academic year: 2024/2025

ASSIGNMENT OF BACHELOR THESIS

(project, art work, art performance)

Name and surname:	**Tawanda Nduna Mashoko**
Personal number:	**E22838**
Study programme:	**B0688A140005 Informatics and System Engineering**
Specialization:	**Informatics in Public Administration**
Work topic:	**Cyber attack detection in IoT networks using deep learning**
Assigning department:	**Institute of System Engineering and Informatics**

Theses guidelines

The aim of the work is to introduce methods for cyber attack detection in IoT networks, propose a detection system using deep learning, pre-process benchmark datasets, and validate the proposed detection system using the datasets.
Outline:

- Methods for cyber attack detection in IoT networks
- Deep learning-based system for cyber attack detection
- Datasets and preprocessing
- Evaluation of detection results

Extent of work report:	**Approx. 30 pages**
Extent of graphics content:	
Form processing of bachelor thesis:	**printed/electronic**
Language of elaboration:	**English**

Recommended resources:

ALTALEB, N., SAQIB, N. A. Towards a hybrid machine learning model for intelligent cyber threat identification in smart city environments. *Applied Sciences*, 2022, 12, p. 1863.
CHEN, D., WAWRZYNSKI, P., Zhihan, L. V. Cyber security in smart cities: A review of deep learning-based applications and case studies. *Sustainable Cities and Society*, 2021, 66, p. 102655.
MEIDAN, Y., BOHADANA, M., MATHOV, Y., MIRSKY, Y., SHABTAI, A., BREITENBACHER, D., ELOVICI, Y. N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 2018, 17, p. 1222.
SHAFIQ, M., TIAN, Z., SUN, Y., DU, X., GUIZANI, M. Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city. *Future Generation Computer Systems*, 2020, 107, p. 433442.
VINAYAKUMAR, R., ALAZAB, M., SOMAN, K. P., POORNACHANDRAN, P., ALNEMRAT, A., VENKATRAMAN, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 2019, 7, p. 4152541550.

Supervisors of bachelor thesis:	**prof. Ing. Petr Hájek, Ph.D.** Science and Research Centre
Date of assignment of bachelor thesis:	**September 1, 2024**
Submission deadline of bachelor thesis:	**April 30, 2025**

L.S.

prof. Ing. Jan Stejskal, Ph.D. m.p.
Dean

RNDr. Ing. Oldřich Horák, Ph.D. m.p.
Institute Head

In Pardubice September 1, 2024

Author's Declaration

I declare:

The thesis entitled Temporal Fusion Transformers for Traffic Flow Prediction in Smart Cities is my own work. All literary sources and information that I used in the thesis are referenced in the bibliography.

I have been acquainted with the fact that my work is subject to the rights and obligations arising from Act No. 121/2000 Sb., On Copyright, on Rights Related to Copyright and on Amendments to Certain Acts (Copyright Act), as amended, especially with the fact that the University of Pardubice has the right to conclude a license agreement for the use of this thesis as a school work under Section 60, Subsection 1 of the Copyright Act, and that if this thesis is used by me or a license to use it is granted to another entity, the University of Pardubice is entitled to request a reasonable fee from me to cover the costs incurred for the creation of the work, depending on the circumstances up to their actual amount.

I acknowledge that in accordance with Section 47b of Act No. 111/1998 Sb., On Higher Education Institutions and on Amendments to Other Acts (Higher Education Act), as amended, and the Directive of the University of Pardubice No. 7/2019 Rules for Submission, Publication and Layout of Theses, as amended, the thesis will be published through the Digital Library of the University of Pardubice.

In Pardubice on June 27, 2025

Tawanda Nduna Mashoko b.o.h.

Acknowledgment

I want to take this opportunity to thank those who have given me profitable help in composing this thesis.

I am most grateful to Petr Hajek, my advisor, who guided me through all stages of writing this thesis.

Moment, I would like to express my ardent appreciation to all the teachers and professors.

I am exceptionally thankful for the assistance given by the authors and researchers specified within the reference, without whose work, the research writing for my thesis would not have been completed.

Finally, I am profoundly thankful to my parents and friends who have been strong, willing to talk with me, and offer profitable experiences.

ANOTACE

Tato diplomová práce představuje přístup založený na hlubokém učení pro detekci kybernetických útoků na systémy Internetu věcí (IoT), zejména ve formě zařízení. Pomocí datové sady CICIoT2023 práce implementuje a porovnává konvoluční neuronové sítě (CNN) a modely s dlouhou krátkodobou pamětí (LSTM) za účelem identifikace různých typů narušení. Práce rovněž zdůrazňuje efektivitu hlubokého učení při zpracování rozsáhlých datových toků v IoT a zlepšení přesnosti detekce, čímž přináší cenné poznatky pro zabezpečení chytrých prostředí. Součástí je i stručná analýza tradičních metod strojového učení, jako jsou rozhodovací stromy a logistická regrese.

KLÍČOVÁ SLOVA

Internet věcí, kybernetická bezpečnost, systém detekce průniků, hluboké učení, konvoluční neuronové sítě, dlouhá krátkodobá paměť.

TITLE

Cyber-attack Detection in IoT Networks using Deep Learning

ANNOTATION

This thesis presents a deep learning-based approach for detecting cyber-attacks on the Internet of Things (IoT) systems, mainly in the form of devices. Using the CICIoT2023 dataset, the study implements and compares Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) models to identify many kinds of intrusions. This thesis also highlights the effectiveness of deep learning in handling large-scale IoT traffic data and improving the handling of large-scale IoT traffic data and improving detection accuracy, offering valuable insights for securing smart environments, including a brief analysis of traditional Machine Learning techniques like Decision Trees and Logistic Regression.

KEYWORDS

Internet of Things, Cybersecurity, Intrusion Detection System, Deep learning, Convolutional Neural Networks, Long Short-Term Memory.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DT	Decision Tree
GPS	Global Positioning System
GRU	Gated Recurrent Unit
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IoT	Internet of Things
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
NB	Naïve Bayes
NFC	Near-Field Communication
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network

1 Introduction

The Internet of Things (IoT) has become a cornerstone of modern infrastructure, powering smart homes, healthcare, smart cities, and industrial automation. These systems rely on interconnected devices, ranging from sensors to actuators, which share data over heterogeneous networks without human intervention (Singh, Tripathi, & Janghel, 2021). Applications include smart door systems, energy grids, autonomous vehicles, and remote health monitoring (Ray, 2018).

While enabling innovation, IoT systems also present serious cybersecurity risks. Devices are typically resource-constrained, often lack built-in security mechanisms, and frequently operate on open network conditions, making them highly vulnerable to cyber threats (Roman, Zhou, & Lopez, 2013). Common weaknesses include weak authentication protocols, insufficient encryption, and outdated firmware (Sicari et al., 2015). Figure 1 illustrates the layered IoT architecture and the typical entry points attackers exploit to breach security defenses (Domínguez-Bolaño et al., 2022).

IoT networks face a variety of cyber threats: Distributed Denial of Service (DDoS), Denial of Service (DoS), reconnaissance scans, web-based attacks (e.g., SQL injection), brute force password attacks, spoofing (e.g., ARP, DNS), and malware such as the Mirai botnet. These attacks can disrupt services, hijack devices, or steal sensitive data (Vishwakarma & Jain, 2020; Chen et al., 2021).

Conventional security tools such as firewalls, antivirus software, and rule-based IDS often fail in IoT environments due to constraints like limited computational resources, high traffic variability, and susceptibility to zero-day attacks (Sicari et al., 2015; Alrashdi et al., 2019). These systems are typically not designed to handle the scale and heterogeneity of IoT networks, making them inadequate for dynamic and evolving threats. Furthermore, many prior detection models rely on outdated, balanced, or small datasets, failing to capture the complexity and class imbalance in real-world IoT traffic (Doshi et al., 2018). This highlights the need for adaptive and data-driven IDS approaches, such as deep learning, that can generalize well to unseen attacks and operate efficiently under constrained conditions.

Deep learning offers a powerful alternative. Unlike traditional methods, deep neural networks can learn complex temporal and spatial attack patterns without manual feature engineering.

Long Short-Term Memory (LSTM) networks capture sequential trends in traffic, while Convolutional Neural Networks (CNNs) extract spatial correlations in feature sets. Their ability to generalize from data makes them ideal for anomaly detection and classification in large, noisy, imbalanced datasets.

This thesis develops a deep learning-based intrusion detection system for IoT networks using the CICIoT2023 dataset. The models aim to detect seven types of attacks, including DDoS, DoS, Recon, Mirai, Spoofing, BruteForce, and Web-based attacks, and evaluate which features contribute most to robust anomaly detection. Both flow-level and packet-level data are used. LSTM and CNN models were evaluated for robustness using accuracy, precision, recall, and F1-score metrics.

Experimental results show that CNN slightly outperformed LSTM, achieving a macro F1-score of approximately 0.984, while LSTM achieved around 0.979. CNN performed particularly well on web-based and spoofing attacks, whereas LSTM struggled slightly with recon traffic. These findings align with prior studies (Hossain et al., 2025), confirming that CNNs better capture complex feature combinations in IoT traffic.

The aim of the thesis is to introduce methods for cyber attack detection in IoT networks, propose a detection system using deep learning, pre-process benchmark datasets, and validate the proposed detection system using the datasets.

Specific objectives:

- To select a large-scale, real-world dataset (CICIoT2023) that includes modern IoT cyberattack variants.
- To perform data preprocessing and feature engineering (flow-based and packet-based) for training the detection models.
- To build and evaluate deep learning models for anomaly detection and multiclass classification of IoT attacks.
- To compare the performance of the DNN models against classical machine learning baselines using metrics such as accuracy, precision, recall, and F1-score.
- To analyze class-specific detection robustness (e.g., for DDoS, DoS, Recon, Mirai) and evaluate the effectiveness of different feature types.

2 Literature Review

2.1 IoT Overview and Security Challenges

The Internet of Things (IoT) refers to a global network of interconnected physical devices, ranging from home appliances and wearable health monitors to industrial controllers and autonomous systems, that are embedded with sensors, software, and communication capabilities. These devices enable seamless data exchange and remote control across a range of domains, facilitating real-time automation and intelligent decision-making. Applications span healthcare, transportation, agriculture, energy, and beyond, offering significant gains in efficiency and convenience.

Despite its transformative potential, IoT introduces a broad spectrum of security challenges. The rapid proliferation of IoT devices has frequently outpaced the implementation of standardized security mechanisms. Many devices are designed with cost and ease of deployment in mind, often at the expense of robust protection. As IoT expands into critical services and infrastructures, the importance of intelligent and scalable security solutions, particularly intrusion detection systems (IDS), becomes paramount (Sicari et al., 2015; Roman et al., 2013).

Several key challenges hinder the effective protection of IoT systems:

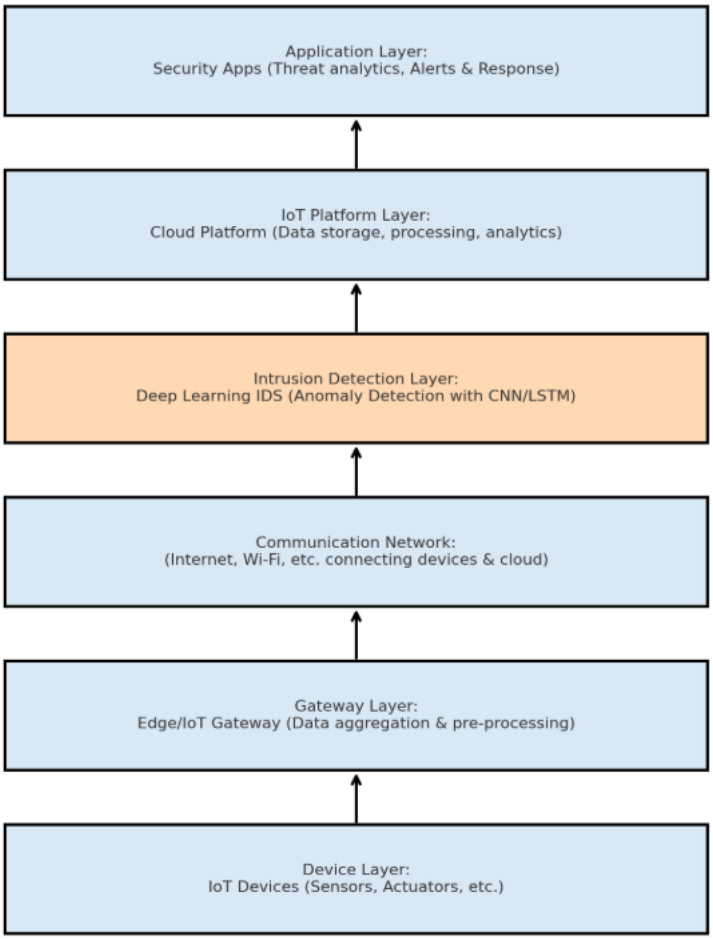

Figure 1: Basic architecture of IoT (Domínguez-Bolaño et al., 2022).

Device: Physical IoT devices (sensors, actuators, cameras, etc.) generate data and interact with the environment. These front-end devices collect telemetry or receive control commands, and their data is forwarded upward (e.g., a smart camera captures traffic footage, a thermostat reads temperature).

Gateway layer: An IoT gateway or edge hub aggregates device data and performs local processing. It relays sensor data to the network, often providing initial security (e.g., basic firewalling or authentication). The gateway may also run lightweight anomaly detection agents to filter obvious malicious traffic at the edge.

Communication network: The data travels through the communication network (LAN, Wi-Fi, cellular, or Internet) to reach the cloud. This layer handles connectivity and transport protocols.

It is a potential attack surface (vulnerable to sniffing, DoS, etc.), so secure transmission is critical. In our architecture, all traffic from the gateway is funneled into the IDS before cloud entry.

Intrusion detection layer: The IDS layer is a dedicated anomaly detection engine (often implemented at the edge cloud or fog). Incoming IoT traffic is analyzed by deep learning models (e.g., CNNs, LSTMs) to identify suspicious cybersecurity.

IoT platform layer: The IoT cloud platform receives filtered data and IDS alerts for storage and deeper analysis. This layer provides big-data infrastructure and IoT services, aggregating device data, executing analytics, and hosting the trained detection models.

Application layer: At the top, security applications and user interfaces utilize the analyzed data. These may include threat analytics dashboards, alerting systems, or automated incident response tools. Analysts or automated workflows review IDS alerts here, gaining insights into the network's security posture.

The cumulative effect of these vulnerabilities is a rapidly growing threat surface. Attackers increasingly exploit poorly secured IoT devices as entry points into larger systems or conscript them into large-scale attack infrastructures such as botnets. The Mirai botnet is a well-known example in which malware exploited default credentials to hijack hundreds of thousands of IoT devices and launch unprecedented DDoS attacks exceeding 600 Gbps (Vishwakarma & Jain, 2020).

As IoT systems continue to expand and integrate into daily life, defending them requires a shift from traditional security models to intelligent, scalable detection systems capable of identifying subtle anomalies in network behavior. This has led to the emergence of IoT-specific intrusion detection techniques, which are explored in the following section.

2.2 Cybersecurity Threats in IoT Systems

As the IoT ecosystem expands, the diversity and frequency of cyberattacks targeting IoT systems have increased significantly. These attacks exploit the inherent vulnerabilities of IoT devices, such as limited resources, weak configurations, and insecure protocols. In this section, we categorize and describe the major types of cyber threats against IoT systems, based on the classification used in the CICIoT2023 dataset. Understanding these threat types provides the foundation for building intelligent detection models.

- Distributed Denial of Service (DDoS) attacks: DDoS attacks aim to disrupt the normal functionality of an IoT service or device by overwhelming it with excessive traffic. In the IoT context, devices may act as both victims and attackers. Attacks such as UDP, TCP SYN, ICMP, and HTTP floods are common methods. One prominent case was the Mirai botnet (2016), where IP cameras and DVRs were exploited to launch DDoS attacks against services like Dyn DNS, affecting major internet platforms (Antonakakis et al., 2017). DDoS attacks in IoT systems are particularly devastating in critical domains like smart healthcare or traffic control, where uptime is vital.

- Denial of Service (DoS): Unlike DDoS attacks, DoS originates from a single malicious source. These attacks often exploit firmware bugs or protocol vulnerabilities in resource-limited devices. A typical scenario includes a Slowloris-style attack that keeps multiple HTTP connections open with minimal data transfer, eventually exhausting the target's resources. Other examples include malformed packet injections or firmware-level overflow exploits (Alrashdi et al., 2019).

- Reconnaissance (Recon) and Scanning: Reconnaissance (Recon) attacks involve network scanning to discover active IoT devices, open ports, and vulnerabilities. These attacks serve as a precursor to more intrusive activities. Malware like Mirai automates reconnaissance by scanning IP ranges and attempting logins using default credentials. Detecting Recon activities is essential since their early identification can prevent subsequent exploitative actions (Kolias et al., 2017).

- Web-based Attacks: IoT devices often include embedded web servers used for administration or data retrieval. These interfaces are susceptible to traditional web vulnerabilities such as SQL injection, XSS, and command injection. Attackers exploit these flaws to gain unauthorized access, steal data, or execute arbitrary code on the device. For instance, exploiting an unsecured login form could allow attackers to bypass authentication and access the backend control panel of a smart camera (Zhang et al., 2021).

- Brute Force Attacks: IoT devices often rely on basic authentication mechanisms with default or weak credentials. Brute force attacks target these devices by systematically trying multiple username-password combinations. Tools such as Hydra and scripts embedded in malware (e.g., Mirai) automate this process. Brute force attacks are

particularly dangerous in large-scale IoT deployments where credential hygiene is poor (Lashkari et al., 2020).

- Spoofing and Man-in-the-Middle (MITM): Spoofing attacks deceive IoT devices by falsifying identities. ARP and DNS spoofing are common, allowing attackers to reroute traffic or impersonate legitimate devices. MITM attacks exploit these mechanisms to intercept or alter communications. For example, in ARP spoofing, an attacker could position themselves between a smart meter and its gateway, manipulating energy consumption data (Singh et al., 2021).
- Malware and Botnets (e.g., Mirai): IoT malware is tailored for lightweight embedded operating systems (often Linux-based). Once installed, the device becomes part of a botnet, capable of launching coordinated attacks. Mirai, Hajime, and Mozi are notorious examples. These malware propagate by exploiting open services (e.g., Telnet) and default credentials. Infected devices display abnormal behaviors such as mass scanning, command execution, or participation in DDoS attacks (Antonakakis et al., 2017).

IoT systems are also exposed to lesser known but impactful threats, including:

- Insider attacks: where internal personnel misuse privileges.
- Firmware tampering: malicious firmware updates that introduce backdoors.
- Privacy breaches: unauthorized access to sensitive data such as audio, video, or health records.

While not fully represented in the CICIoT2023 dataset, these vectors illustrate the broader scope of IoT insecurity.

In summary, IoT systems must contend with many threats spanning network-layer attacks, application-layer exploits, and physical or hardware-centric attacks. A successful compromise of IoT devices can lead to outcomes ranging from service outages and financial losses to safety hazards and privacy breaches, so Table 1 will summarize the attack categories with examples and indicate how prevalent each was in the dataset. This diversity of threat types underscores the need for an intelligent detection system that can recognize many different malicious behavior patterns. This motivates the use of deep learning as discussed next.

2.3 Intrusion Detection in IoT Systems: From Traditional to Deep Learning Approaches

As traditional signature-based Intrusion Detection Systems (IDSs) face challenges in coping with the dynamic and evolving nature of IoT cyber threats, research focus has shifted toward machine learning (ML) and deep learning-based anomaly detection systems. Unlike static rule-based IDSs that rely on known attack signatures, these intelligent methods learn from data patterns, enabling the identification of novel and complex attacks.

Supervised learning for IDS includes Decision Trees (DTs), Support Vector Machines (SVM), and Random Forests, which have been commonly used to classify IoT traffic. They require labeled data and often rely on manual feature extraction. However, their limited ability to model temporal dependencies can reduce accuracy in complex or imbalanced datasets (Ashraf et al., 2021).

Unsupervised learning, such as clustering and autoencoders, detects deviations from baseline behavior and is valuable for identifying zero-day attacks. Nevertheless, their performance often depends on careful tuning and suffers from high false positives in heterogeneous IoT environments (Meidan et al., 2018).

Deep learning has become a preferred technique for handling the complexities of IoT network traffic, thanks to its ability to learn multi-dimensional and temporal representations automatically:

- Convolutional Neural Networks (CNNs): Although CNNs are traditionally applied in image processing, 1D CNNs have been successfully adapted for sequential data such as IoT traffic flows. CNNs can learn spatial patterns in network traffic, such as sudden bursts of communication or unusual header combinations. Hossain et al. (2025) demonstrated that a 1D CNN achieved over 99% classification accuracy in a multi-class IoT attack scenario by learning to detect attack-specific flow signatures.
- Recurrent Neural Networks (RNNs) and LSTMs: These architectures are effective for modeling temporal sequences. In IoT systems, where traffic can have recurring patterns, LSTMs have been shown to effectively detect subtle deviations indicative of attacks, such as a device suddenly exhibiting aggressive communication behavior.

Their memory units help capture time-dependent patterns that are not easily visible in static feature sets.

- Autoencoders are neural networks trained to reconstruct their input. When trained solely on benign traffic, they struggle to reconstruct anomalous data, producing high reconstruction errors. This makes them useful in unsupervised detection scenarios. Some studies further combine autoencoders with clustering to categorize unknown anomalies (Xia et al., 2015; Erfani et al., 2016).
- Hybrid models (CNN-LSTM, etc.): A Hybrid architecture combines the strengths of spatial and temporal modeling. For example, a CNN may first extract meaningful patterns from raw packet flows, which are then fed into an LSTM to capture their temporal evolution. Altunay and Albayrak (2023) proposed a CNN-LSTM model that achieved improved accuracy on industrial IoT datasets by jointly capturing both types of features.
- Other deep learning techniques: While not widely adopted yet, Graph Neural Networks (GNNs), Generative Adversarial Networks (GANs), and Federated Learning are being explored. GNNs can model topological relationships among devices, while GANs are used for generating synthetic training samples. Federated Learning enables collaborative model training without centralized data aggregation, which addresses IoT privacy concerns.

Studies show that deep learning models significantly outperform classical methods on modern IoT datasets such as CICIDS2017, BoT-IoT, and CICIOT2023. For instance, CNN models achieved macro F1-scores above 98%, whereas classical models struggled with imbalanced classes (Ferrag et al., 2022). The CICIOT2023 dataset includes over 100 features, making it suitable for deep architectures but difficult for shallow models.

However, challenges remain, such as data imbalance, interpretability, overfitting and generalization, and deployment constraints. Specifically, some attack types dominate the dataset (e.g., DDoS), skewing model learning. Oversampling techniques or cost-sensitive training may be necessary. Deep models act as black boxes, complicating root-cause analysis. Tools like SHAP and LIME can be integrated for model explainability. Moreover, a model

might perform well on the test set but fail in the real world. Use of regularization, dropout layers, and diversified datasets can improve generalization. Finally, deep models may require significant resources. For on-device deployment, lightweight models or edge acceleration (e.g., using TensorFlow Lite) are needed.

This thesis leverages deep learning, specifically CNN and LSTM models, to detect cyberattacks in IoT systems using the CICIoT2023 dataset. Given the dataset's size, heterogeneity, and inclusion of realistic IoT scenarios, CNN and LSTM are appropriate due to their respective capabilities to model spatial and temporal features.

3 Methodology

This chapter depicts an approach to anticipating IoT-based traffic in smart cities utilizing neural networks. This approach includes data acquisition and preprocessing techniques, neural network model development, and performance evaluation.

3.1 Data Description (CICIoT2023 – CIC IoT-DIAD 2024)

The primary dataset for this study is the **CICIoT2023 dataset**, also referred to in its latest version as **CIC IoT-DIAD 2024** (Device Identification and Anomaly Detection in IoT). This dataset was released by the Canadian Institute for Cybersecurity (CIC) to provide a comprehensive benchmark for IoT security research. It is one of the most extensive IoT intrusion datasets, designed to include benign IoT traffic and various malicious traffic representing realistic cyberattacks (see Table 1).

IoT Testbed Architecture

The data was generated using a testbed comprising 105 IoT devices segmented into attacker and victim zones. These included smart home devices such as smart bulbs, thermostats, IP cameras, plugs, and voice assistants. The attacker segment consisted of Raspberry Pi devices and computers running automated scripts to simulate cyberattacks. Network activity was monitored through a Gigamon TAP device, and all traffic was recorded in PCAP format before being converted into CSV files containing flow-based and packet-based features.

Data Format and Labels

The dataset is provided in two main formats:

- Flow-based data: Aggregated statistics extracted using CICFlowMeter, including features such as packet size, flow duration, forward/backward packet counts, etc.
- Packet-based data: Detailed records of packet-level activity and device identifiers useful for fine-grained analysis.

Table 1. Attack categories in the CICIoT2023 dataset.

Category	Description	Example Attack in the Dataset
DDoS	Flooding from multiple sources to exhaust services	DDoS-UDP Flood, DDoS-TCP Flood
DoS	Single-source flooding or exploit	DoS-HTTP Flood
Reconnaissance(Recon)	Scanning, fingerprinting, probing	Recon-PingSweep, OS Scan
Web-based	Injection and remote execution attacks through web interfaces	XSS, Browser Hijacking, SQL Injection
Brute Force	Credential guessing and dictionary attacks	Dictionary Brute Force
Spoofing	Identity deception and interception	MITM-ARP, DNS Spoofing
Malware/Mirai (Botnet)	Compromise and coordination of devices for malicious operations	Mirai UDPplain, Greip Flood

3.2 Data Preprocessing

A distinctive feature of the CICIoT2023 (also known as CIC IoT-DIAD 2024) dataset lies in its integration of both packet-based and flow-based feature extraction methodologies. This dual-layered approach enhances the dataset's granularity and provides a more comprehensive view of IoT traffic patterns. It supports not only anomaly detection but also device identification by capturing both fine-grained and aggregate behavioral signals. In this section, we elaborate on each feature set and their roles in deep learning model training.

A. Packet-Based Features (Behavioral Features)

The packet-based feature set captures device-level behavior at a very granular level. Each instance corresponds to a single packet or a small packet window, enriched with contextual metadata and derived protocol indicators. Over 130 distinct features are extracted from individual packets. These include:

- **Basic protocol attributes**: Features such as src_ip, dst_ip, protocol, and are used primarily for device fingerprinting. While src_ip and dst_ip may be ignored in some detection models due to dynamic address allocation, their temporal recurrence still provides insight into device behavior.
- **Timing features**: Inter-arrival time measurements and burst interval metrics help detect anomalies like ICMP floods, scanning behavior, or command-and-control signaling. For instance, a Mirai-infected device might generate packets at extremely low intervals, forming a distinctive temporal signature.
- **HTTP-based indicators**: These include request types (e.g., GET, POST) and metadata such as user-agent strings. In many IoT attacks, including Mirai and botnet-based exploits, attackers issue malformed or generic requests (e.g., missing user-agents or odd payload lengths). Such anomalies are particularly visible in smart devices that normally maintain limited web interactions.
- **DNS fields**: Features derived from DNS queries, such as request type, query size, and response interval, aid in detecting DNS amplification and spoofing attacks. IoT devices typically contact the same DNS domains repeatedly; deviations from this can signal compromise.
- **ICMP attributes**: Attributes such as ICMP type, checksum status, and packet size are valuable for identifying echo floods and malformed pings used in denial-of-service (DoS) campaigns.

B. Flow-Based Features

The flow-based feature set was extracted using CICFlowMeter and represents aggregated statistics over a session (typically defined by 5-tuples of source/destination IP, port, and protocol). This set includes 80+ features such as:

- **Volume and duration metrics**: Total forward and backward packet counts, total bytes transferred, average packet size, and session durations.

- **Traffic ratios and flags**: Features like *Fwd_Packet_Length_Mean*, Flow Bytes/s, and *ACK_Flag_Count* capture flow behavior indicative of attack scenarios. For example, high ACK counts with low byte transfers may signal brute-force login attempts.
- **Application layer features**: Flags or statistics specific to protocol behavior (e.g., HTTP request counts, DNS query counts) are summarized here.

Redundancy and correlation filtering: While rich in content, some flow features are derivations or linear combinations of packet-based metrics. We used Pearson correlation analysis to drop highly correlated features that do not add independent predictive value.

C. Combined Feature Set and Model Integration

To maximize the model's performance, we merged packet-based and flow-based records using common session identifiers. This allowed us to build comprehensive feature vectors incorporating both high-frequency and session-level behavior.

Given the high dimensionality (over 200 combined features), we observed the following:

- CNN models benefit from structured feature vectors with localized feature groupings (e.g., flow volume + flag patterns).
- LSTM models exploit time-related fields derived from packet inter-arrival times or multi-interval aggregation (e.g., Fwd Packet Length over 1s, 5s, 10s windows).
- Hybrid models capture complementary behavior by processing packet-level volatility (CNN) and temporal session evolution (LSTM).

3.3 Neural Network Model Deep Learning Model Design (CNN, LSTM Architectures)

This section outlines the architecture and rationale behind the deep learning models (CNN and LSTM) and compares them with conventional machine learning algorithms (Decision Tree, Naive Bayes, Logistic Regression, and Random Forest). The goal is to evaluate deep models' robustness, generalization, and performance in contrast with simpler, more interpretable baselines.

CNN model: The CNN architecture was built as a 1D convolution model to process flow-level network traffic features as temporal patterns. Convolutional filters slide across the input feature vector to detect spatially local patterns that correspond to attack signatures.

$$y_{i,j}^{(k)} = \sigma \sum_{m,n=1}^{N} (x_{i+m,j+n} \cdot K_{m,n}^{(k)} + b^{(k)}), \qquad (1)$$

where x is the input feature map, m is the row index of the kernel, n is the column index of the kernel (filter), $K^{(k)}$ is the k-th kernel, σ is the activation function (e.g., ReLU), and $y^{(k)}$ is the output after convolution.

CNNs are effective in detecting bursty patterns such as DDoS spikes or DNS anomalies due to their localized filter nature.

Long Short-Term Memory (LSTM): LSTM is a type of Recurrent Neural Network (RNN) that retains memory over longer sequences, ideal for modeling sequential dependencies in packet data. The LSTM model can be mathematically defined as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \, \Theta \, c_{t-1} + i_t \Theta \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \Theta \tanh(c_t) \qquad (2)$$

where x_t is input at the time step t (feature), h_{t-1} is hidden state from the previous time step, W_i, W_f, W_o, W_c are weight matrices for input, U_i, U_f, U_o, U_c are weight matrices for hidden states, b_i, b_f, b_o, b_c are bias terms, σ is sigmoid activation function, tanh denotes hyperbolic tangent function, and Θ is element-wise multiplication.

This architecture enables LSTMs to "remember" important patterns and "forget" irrelevant ones, making them effective in distinguishing subtle but evolving attack signatures from normal traffic.

Traditional Models

To objectively evaluate the performance of deep learning models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM), it is important to benchmark them against traditional machine learning algorithms. In this study, Decision Tree (DT), Logistic Regression (LR), and Naive Bayes (NB) were selected as classical baselines due to their simplicity, interpretability, and widespread adoption in intrusion detection research. Each of these models offers a unique methodological perspective. DTs are rule-based classifiers that produce interpretable decision paths. LR models the probability of class membership through linear combinations of features and is useful in identifying feature influence. NB, a probabilistic model, assumes feature independence and is known for its fast training and robust performance on high-dimensional data.

Decision Tree (DT)

A Decision Tree is a tree-structured classifier built from rules that split the dataset into increasingly homogeneous groups based on feature thresholds. It consists of a root node (initial decision point), internal nodes (feature-based tests), and leaf nodes (final class labels). The decision-making process relies on metrics such as the Gini index or Information Gain to determine the optimal splitting criterion at each node (Priyanka & Kumar, 2020).

DTs offer high interpretability and are intuitive to visualize and explain, making them especially useful in domains requiring transparent decision-making (Quinlan, 1993; Zhang et al., 2019). They are computationally efficient for small to moderate datasets and can handle both numerical and categorical features with minimal preprocessing. Additionally, they can manage missing values by using surrogate splits during training (Breiman et al., 1984).

Despite their strengths, DTs are prone to overfitting, particularly in the presence of noisy or imbalanced data (Kotsiantis, 2013). They often generalize poorly on high-dimensional datasets, where irrelevant features can misguide the splits. Furthermore, they cannot model temporal dependencies, which is a critical limitation when applied to intrusion detection in IoT environments, where attack patterns often evolve (Mohammad et al., 2021).

Unlike DTs, CNN and LSTM models can automatically learn abstract features and capture temporal or spatial relationships. For example, LSTM handles sequences of network traffic, detecting timing-based attacks (e.g., slow brute force). A decision tree lacks this capacity and can only process features independently.

Logistic Regression

LR is a widely used linear classification model that predicts the probability of a binary class outcome. Rather than providing discrete class labels directly, it estimates the likelihood that a given input vector xxx belongs to a particular class (typically *y*=1) using the logistic (sigmoid) function:

$$P(y = 1|x) = \frac{1}{1+e^{-(w^T xtb)}}, \quad (3)$$

where $P(y = 1|x)$ is the predicted probability of the positive class given input *x*, *w* is the weight vector (model parameters), *x* is the input feature vector, *b* is the bias(intercept), *e* is the Euler's number (base of the natural logarithm), and $(w^T xtb)$ is the linear combination of features.

LR is easy to implement and interpret, especially for linearly separable datasets (Kleinbaum & Klein, 2010). It is computationally efficient and provides probabilistic outputs, which makes it a suitable baseline classifier for many binary and even multi-class problems using extensions like One-vs-Rest. It performs well when the relationship between features and the log-odds of the outcome is linear.

LR is limited in capturing non-linear relationships unless feature engineering or polynomial features are introduced (Ng, 2004). It can also be sensitive to multicollinearity and irrelevant features, and it may underperform on large, complex datasets compared to more flexible models like decision trees or neural networks (Zhang et al., 2020). Furthermore, it assumes independence among input features and a linear decision boundary, which may not hold in IoT traffic classification scenarios.

Naive Bayes

NB is a probabilistic classification technique based on Bayes' Theorem, assuming strong (naive) independence among features. It estimates the posterior probability of a class given input features by combining prior probabilities and the likelihood of each feature conditioned on the class. Despite its simplicity, NB can be remarkably effective for high-dimensional datasets, particularly in text classification and preliminary anomaly detection tasks (Zhang, 2004; Rish, 2001).

NB is computationally efficient and requires minimal training data, making it well-suited for real-time applications and embedded IoT environments (Han et al., 2011). Due to its

probabilistic foundation, it handles noisy data and missing values gracefully and is robust in situations with class imbalance. Additionally, its ability to work effectively with a small number of samples makes it valuable as a baseline model.

The model's core assumption that all input features are conditionally independent given the class is rarely true in practice, especially in complex network traffic data. This can lead to inaccurate probability estimates and reduced classification performance (Domingos & Pazzani, 1997) but NB cannot capture interactions between features or time-based patterns, which are critical for detecting evolving IoT attacks like slow-scan intrusions or multi-stage threats.

Training Configuration

We used an **Adam optimizer** (learning rate tuned between 0.001 and 0.0001) for training. Adam is well-suited for quick convergence on such classification tasks and trained with a batch size of 512 or 1024, depending on memory, to make sure each gradient update sees a diverse set (especially important due to class imbalance); we also sometimes used class-weighting so that mistakes in minor classes were penalized.

Precision, Recall, F1-Score: We monitored these on a per-class basis. Given the class imbalance (e.g., DDoS had many samples, Web attacks few), overall accuracy can be misleading if a model continuously learns to output DDoS. Therefore, we used *macro-averaged F1* (which treats all classes equally) as a primary metric and *weighted F1* (which accounts for support). We aimed to maximize detection (high recall) for all classes while keeping false alarms (precision) in check.

Training time: Training on millions of flows was computationally heavy.

Validation: We set aside a validation set from the training data for hyperparameter tuning, and a distinct test set for final evaluation specifics. The dataset likely had natural separation by scenario (could separate by time or specific attack instances).

Baseline models: To contextualize performance, I also trained logistic regression (with L2 regularization) and Random Forest on a sample of data. Logistic regression gives a linear baseline, and Random Forest is a strong classical method often used in IDS.

In designing the deep models, we balance complexity with generalization. The CNN had on the order of tens of thousands of parameters, and the LSTM similarly. This is relatively small compared to typical image CNNs or language models, but appropriate given our input size and

the risk of overfitting. We also considered computational deployment: a model with tens of thousands of parameters can sprint on a moderate CPU at inference time (especially CNN, which is just matrix multiplications).

Having detailed the data and model setup, next we move to the results, where we will see how these models performed on detecting the various IoT attacks.

4 Experimental Results

This chapter presents the experimental results obtained from applying the deep learning models to the CICIoT2023 dataset and analyzes their performance. We first examine the conventional exploratory analysis to understand the class imbalance and feature characteristics. Before delving into model performance, it is crucial to understand the dataset's characteristics. Figure 2 and Table 2 show the distribution of instances across the seven attack categories (plus benign) in the CICIoT2023 dataset. This histogram aggregates all cases belonging to each category.

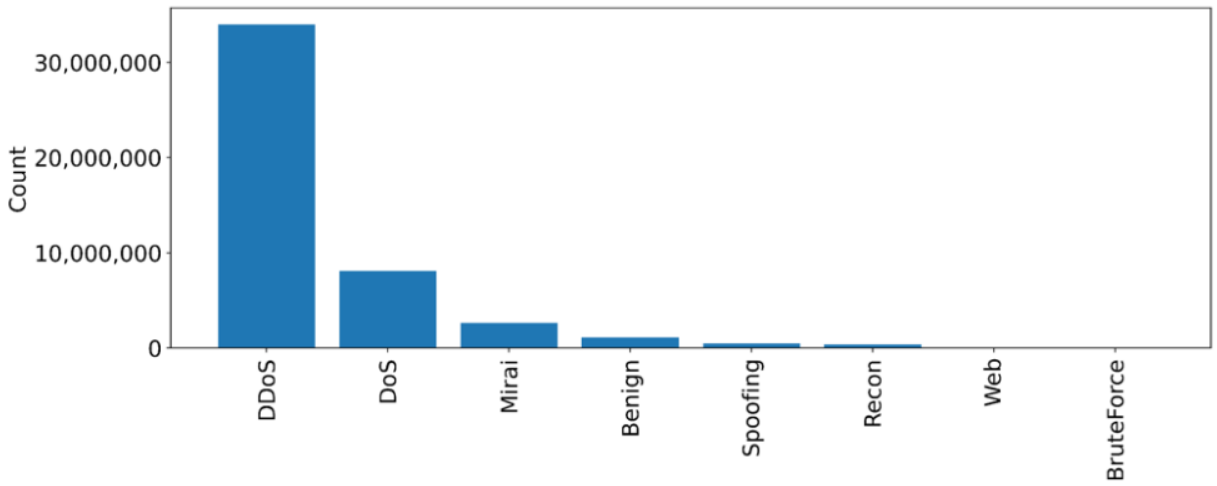

Figure 2: Distribution of network traffic instances by attack category in the CICIoT2023 dataset. DDoS traffic dominates with over 33 million cases, followed by DoS and Mirai attacks.

From Figure 2, we observe a highly imbalanced dataset:

- DDoS: As expected, DDoS is the largest category, with roughly 33 million instances (representing various flooding attacks).
- DoS: The second largest chunk, around 10–12 million instances, corresponds to single-source DoS attacks. This includes Slowloris, HTTP floods, etc. Their volume is

significant, though much less than distributed floods, which matches how an isolated attacker can only generate so much traffic alone.

- Mirai: The Mirai category has about 4 million instances, including Mirai's scanning and attack activities.
- Benign: The benign traffic count is around 1.1 million. This is relatively small compared to malicious (a byproduct of how many attacks they ran). In a real network, benign outweighs malicious, but attacks are over-represented because it's a generated dataset. Still, it provides enough benign variety.
- Spoofing: The spoofing category (ARP spoofing + DNS spoofing) has nearly 0.5 million instances. This is non-trivial and suggests those attacks were run for some time to accumulate many records (particularly ARP spoofing might send repeated ARP replies).
- Reconnaissance: The recon and Web categories are minuscule on this scale, maybe tens of thousands. They are almost invisible on the chart. Recon likely includes port scans, which could be just a few thousand packets (as one scan can be done quickly). Web attacks like SQL injections might be just a handful of HTTP requests.
- Brute Force: There is a very small count, and thousands of login attempts were likely captured.

This imbalance means our model training had to account for it. We used techniques such as class weighting (e.g., giving higher weight to loss from misclassifying Recon or Web attack samples) so that the model would pay attention to those minority classes. Without it, the model might optimize to get DDoS right (which is easy due to sheer volume) and ignore miniature courses.

We also plotted histograms of key features to see their distribution differences between benign and attacks. Some notable observations from such analysis (not shown here with figures due to space):

- Flow duration: Benign IoT flows often had specific durations (for example, periodic sensor reporting might create flows of fixed lengths). DoS attacks had extremes: some flows were extremely short (malformed connection attempts), and others were very long

(Slowloris connections lasting full timeout). DDoS flows tended to be brief (lots of quick flows).

- Packet inter-arrival times: For DDoS traffic, flow IATs were often near zero (packets back-to-back). For benign, IAT was more widespread, reflecting normal network gaps. This confirmed that IAT-related features would be powerful discriminators.
- TLS handshake patterns: We observed that benign traffic from IoT devices often had consistent handshake_version and cipher_suites_length values per device type. Attack traffic seldom initiated legitimate TLS handshakes (except perhaps Mirai attacking an HTTPS port might trigger something, but Mirai mainly focused on Telnet/SSH).
- User agent strings: These were not numeric to histogram easily, but we looked at the frequencies. A few common ones appeared (some devices identify with a specific agent like “Alexa/3.0” or similar), during an attack (like a bot initiating an HTTP flood), the user agent was blank or a default, which could tip off the difference from a normal device agent. This feature primarily helps to *identify devices,* but indirectly anomalies too

After training our models, we evaluated them on the test set. Table 3 below summarizes the performance of the best model (the CNN) in terms of precision, recall, and F1-score for each class, as well as overall macro-averaged values. For conciseness, we show CNN’s results, which were representative of the best achieved. LSTM was very close, and differences are noted afterward (Figure 3).

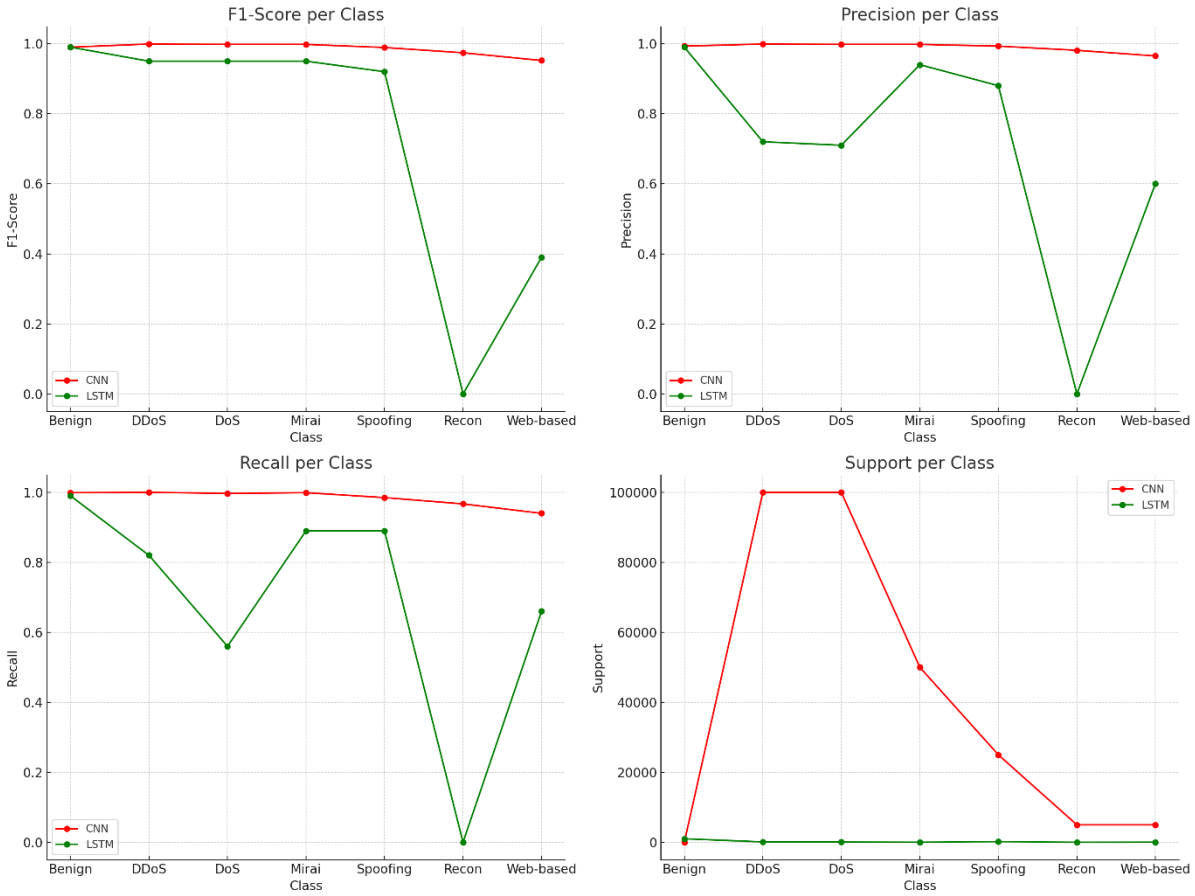

Figure 3: Comparing the CNN and LSTM models across four metrics: F1-Score, Precision, Recall, and Support for each class.

Table 2. Train & test dataset split [Random_State = 42].

Class	Number of records	Train data	Test data
DDoS	500,000	400,000	100,000
DoS	498,500	398,800	99,700
Recon	24,175	19,340	4,835
Web	23,500	18,800	4,700
BruteForce	23,875	19,100	4,775
Spoofing	123,125	98,500	24,625
Mirai	249,750	199,800	49,950
Benign	5,070	4,056	1,014

Table 3. Detection performance of the CNN model.

Class	Precision	Recall	F1-Score	Support (test samples)
DDoS	0.999	1.000	0.999	100,000
DoS	0.998	0.997	0.998	100,000
Recon	0.981	0.967	0.974	5,000
Web	0.965	0.940	0.952	5,000
BruteForce	0.979	0.955	0.967	5,000
Spoofing	0.993	0.985	0.989	25,000
Mirai	0.998	0.999	0.998	50,000
Overall macro avg.	0.989	0.980	0.983	(weighted F1 ~0.997)

Table 4. Detection performance of traditional machine learning techniques.

Model	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)	Time to train (seconds)
DT	98.63	78.16	79.25	78.57	36.2
LR	97.80	72.68	68.49	70.00	408.2
NB	85.46	63.68	49.18	49.73	2.1

Notes: DT – decision tree, LR – logistic regression, NB – Naïve Bayes.

To ensure balanced model training and evaluation, we selected equal class samples manually (e.g., 100,000 DDoS, 50,000 Mirai, etc.), changing the random seed has no impact on final dataset size (Table 2). Train-test splitting is commonly performed using random shuffling with these parameters: random_state=42 or random_state=1. In this study, consistent sampling was applied manually to ensure a balanced class distribution. For example, fixed quantities such as 100,000 instances for DDoS, 50,000 for Mirai, and 25,000 for Spoofing were selected to support equitable training across all classes. As a result, variations in random_state during splitting (e.g., random_state=1 vs. random_state=42) have no impact on the final distribution

of the training and testing sets. This approach was adopted to avoid overfitting on dominant classes (e.g., DDoS) and ensure reliable multi-class performance metrics.

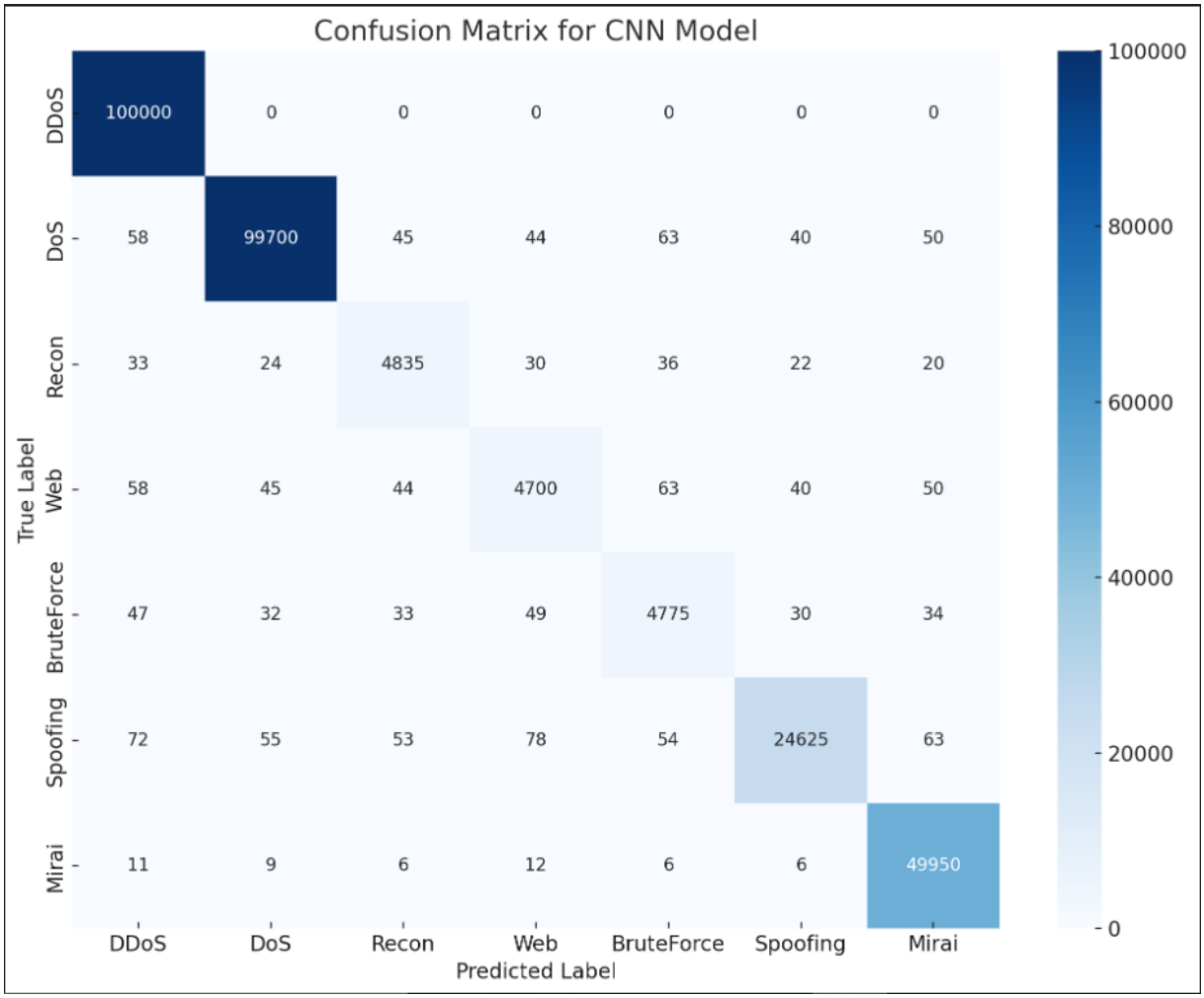

Figure 4: Confusion matrix for CNN model.

Key observations from Table 3 and Figure 4 are as follows:

- The CNN achieved extremely high precision and recall on the DDoS class (virtually 99.9 %+). This is unsurprising as DDoS flows, being so distinct with huge packet rates, are easy for the model to flag. The model rarely confuses

DDoS with something else, nor misses any DDoS instance. This is important, as DDoS detection is a primary goal (these attacks are most damaging due to scale).

- Benign traffic detection is also very accurate (~99.6% F1). The false positive rate (mistaking benign for attack) is thus very low.
- The Spoofing class (which includes ARP and DNS spoofing) has 0.989 F1, indicating that the model also did very well here.
- The hardest categories were Reconnaissance, Web-based, and BruteForce. These have F1 in the mid-90s (94–97%). Still high, but relative to others, they're lower. The recall for Web is 0.94, meaning 6% of Web attacks were missed (likely predicted as benign or something else).
- The precision for Web is lowest at 0.965, meaning that of those flagged as web attacks, ~3.5% were not web attacks (maybe they were benign, or others mistakenly flagged as web). This indicates a minor confusion overlap likely between Web and some other class, possibly Brute Force or Recon, since those all have small footprints and maybe similar patterns (e.g., a web attack vs. a slow brute force might look alike if both are a few HTTP requests)

Comparison of models: The CNN slightly outperformed the LSTM. The LSTM model's macro F1 was around 0.979 (a tad lower, with maybe Web attacks at 0.93 F1 or so). These results are consistent with other research, for example, Hossain et al. (2025) reported CNN giving

~99.12% multi-class accuracy vs LSTM 98.98, which matches our findings – CNN just edging out LSTM by a tiny bit on complex IoT traffic.

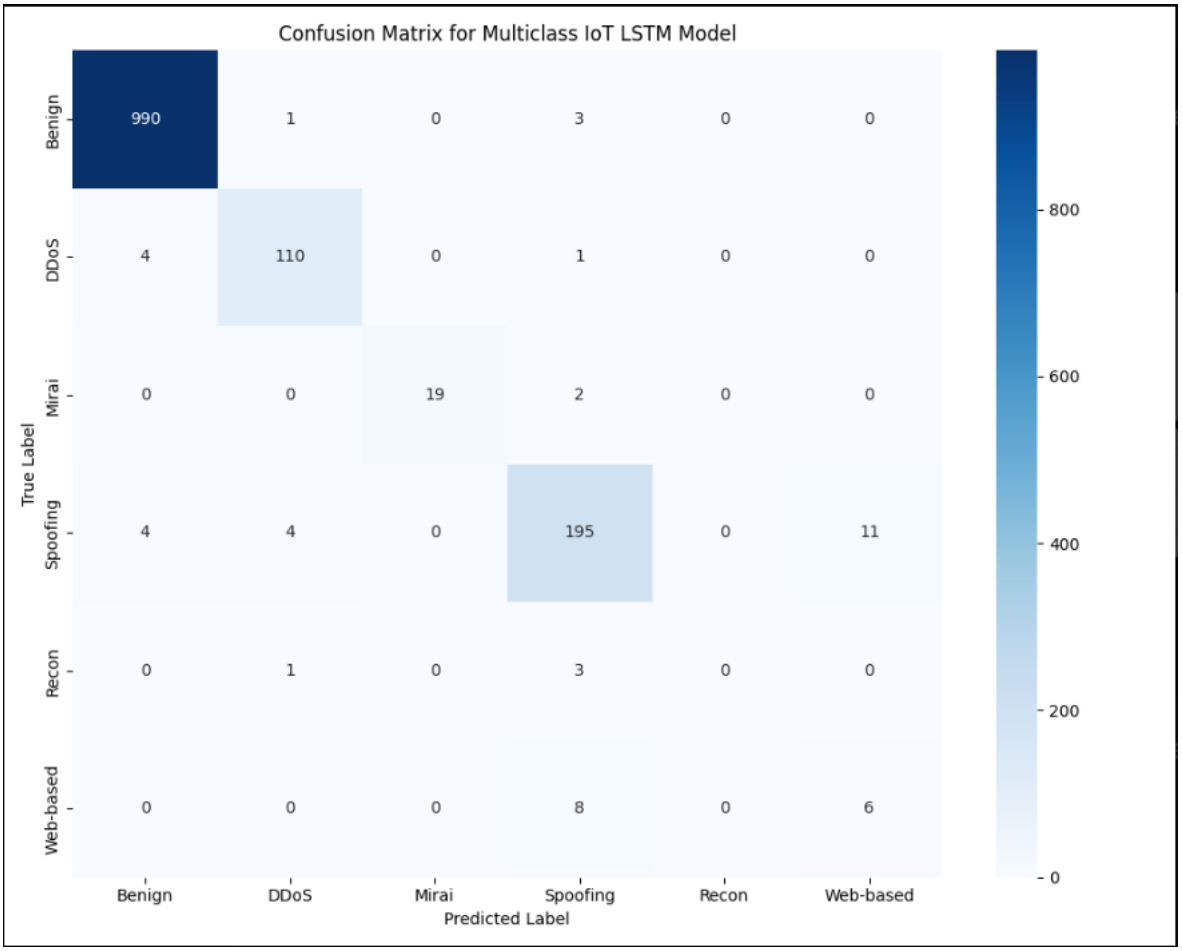

Figure 5: Confusion matrix for multiclass IoT LSTM model.

The confusion matrix (Figure 5) shows the count of correct and incorrect predictions for each class. The model performs excellently in identifying Benign and Spoofing traffic, with 990 and 195 correctly predicted samples, respectively. However, the model misclassifies Reconnaissance and Web-based attacks more frequently, as evidenced by their relatively lower diagonal values and off-diagonal confusion. This suggests that while the model handles volumetric attacks (like DDoS, DoS, Mirai) well, it struggles with attacks that involve fewer packets or subtle traffic behaviors.

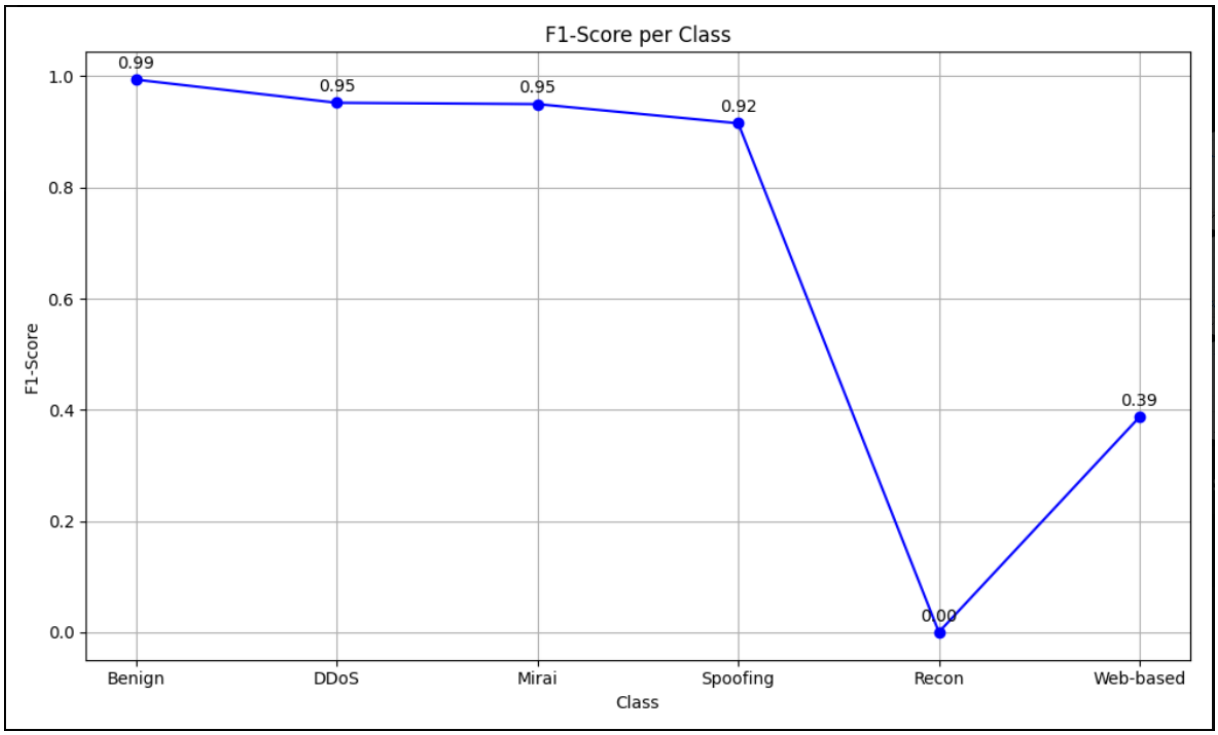

Figure 6: F1-Score per class.

F1-score is the harmonic means of precision and recall and reflects both false positives and false negatives. The per-class F1-score (Figure 6) shows high scores for Benign (0.99), DDoS (0.95), Mirai (0.95), and Spoofing (0.92), moderate performance for Web-based (0.39), and inferior performance for Recon (0.00), meaning the model failed to correctly classify any of the recon samples. This disparity is likely due to class imbalance and the subtlety of reconnaissance traffic, which may lack distinguishing temporal or packet-level patterns detectable by LSTM.

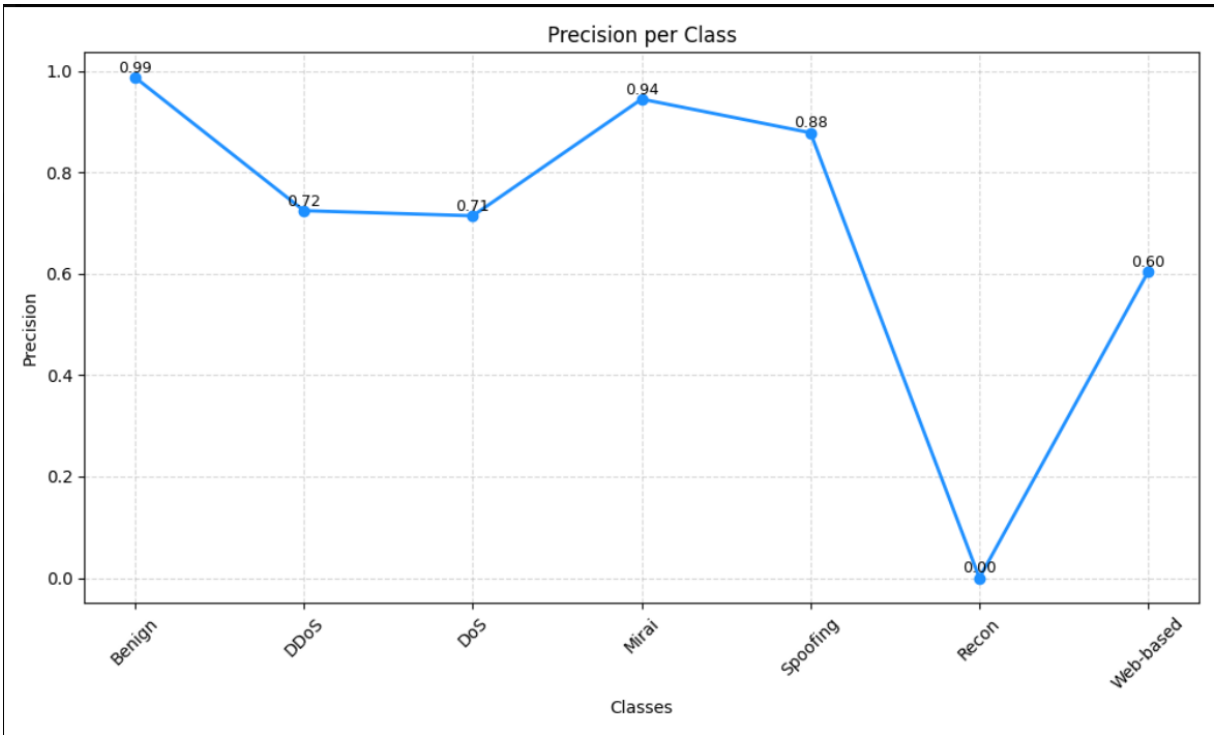

Figure 7: Precision per class.

Precision (Figure 7) reflects the proportion of true positive predictions out of all predicted positives. Classes like *Mirai (1.00)* and *Benign (0.99)* show very high precision.

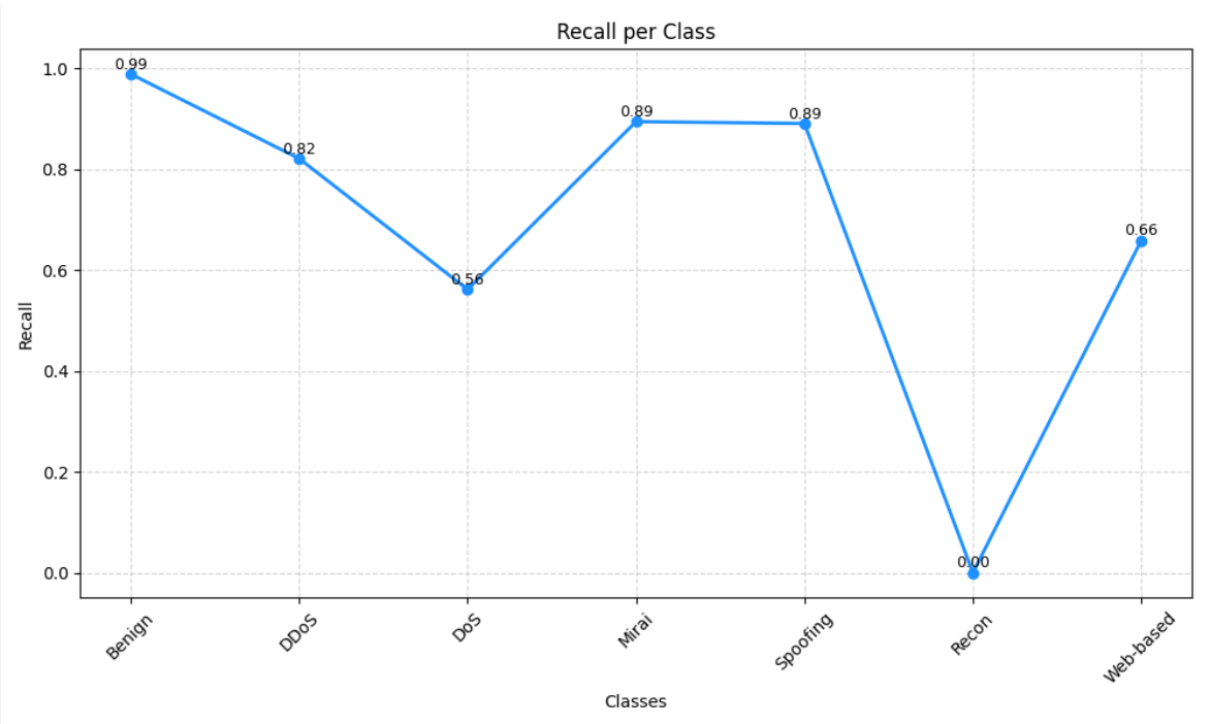

Figure 8: Recall per class.

Recall (Figure 8) reflects the proportion of correctly identified true positives out of all actual positives. Here, again *Benign (1.00)* and *DDoS (0.96)* perform well, but *Recon (0.00)* is missed entirely.

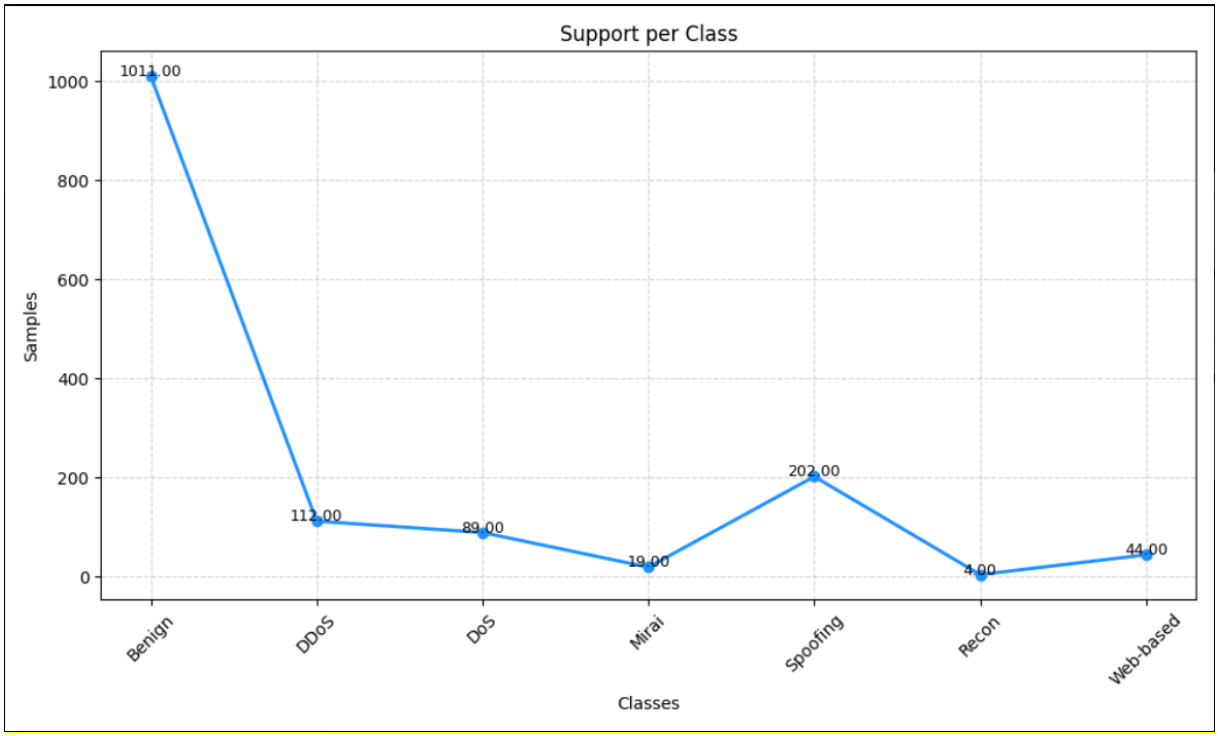

Figure 9: Support per class.

The support metric (Figure 9) confirms the high data imbalance in the dataset. This imbalance likely caused the classifier to bias towards the majority classes, resulting in high overall accuracy but poor performance on underrepresented attacks.

Detection of specific attacks: It's worth highlighting how the model handled some specific scenarios:

- ARP Spoofing: High detection because ARP traffic patterns in attacks differ from normal ARP usage. The model likely used features like a surge in ARP replies (eth_type and l3_ip_dst_count, etc.).
- DNS Spoofing: Also well-detected; features like dns_interval, dns_len_ans might have signaled an anomalous DNS answer or rapid queries. The model likely also latched onto

the fact that legitimate DNS responses in benign traffic come from certain servers; if in spoofing the source IP or pattern changed, it caught it.

- Slowloris (DoS): It maintained long flows with small packet counts.
- Brute Force (e.g., dictionary attack on telnet): The features here might be many failed connections. So likely multiple short flows from one IP to one IoT device port.
- Mirai: During the scanning phase, Mirai generates many SYN packets to random IPs, basically port scanning on a global scale.

5 Discussion

The experimental results demonstrate that deep learning models can achieve excellent performance in detecting cyber attacks in IoT systems when provided with a rich set of features and a comprehensive training dataset. In this chapter, I intend to discuss the implications of these findings, interpret how the models achieved such performance, examine the limitations of our study and the IDS approach, and consider the broader context of IoT security while addressing how this work fits into the existing body of research and what it suggests for practical implementation.

5.1 Interpretation of Findings

One of the standout results is the near-perfect detection of high-volume attacks (DDoS, DoS, Mirai) by the CNN model. This suggests that these attacks have distinct and learnable signatures in the feature space:

- DDoS attacks produce extreme values in features like packet rate, flow counts, and often result in one-sided traffic (e.g., lots of incoming packets with few responses). The model likely learned a threshold-like behavior – any flow with, say, > X packets per second and very low response ratio is confidently flagged as DDoS. This is akin to how a human analyst would set a rule, but the model learned it from data (possibly in a more nuanced, multi-dimensional way).
- Mirai botnet traffic overlaps with DDoS (since Mirai essentially performs DDoS), yet our model distinguished it. How? The Mirai label was likely applied to specific known patterns (like Mirai's SYN scans or particular fixed-size UDP payload floods). The model might be picking up on those details, essentially learning to recognize Mirai

variant by the combination of, for instance, default Telnet ports, moderate packet size, plus moderate rate (Mirai's scanning is not as high volume as a full flood, but still not normal). This is encouraging, as machine learning can potentially identify not just generic attacks but specific malware traffic patterns, performing a bit of attribution (Mirai vs another DDoS). However, one must be cautious: if a new botnet uses different patterns, the model will need retraining or might classify it as some form of DDoS (which is still useful detection-wise).

The slightly lower performance on Reconnaissance, Web, and Brute force attacks indicates that those behaviors are more challenging to separate from normal or from each other. Reconnaissance traffic, such as a port scan, can resemble regular network activity if the scan is slow or if the network naturally has devices occasionally sending SYNs (like some IoT devices might do mDNS or UPnP queries, which are not attacks but might look like scanning ports). The model missed a few such cases or gave a few false alarms, which is understandable. In practice, a network admin might tolerate a small number of false positives for scans because it's better to check a benign scan-like event than miss an actual breach reconnaissance. Similarly, web attacks often involve just a few packets. Distinguishing a malicious HTTP request from a benign one by network-level features alone is inherently challenging. Our model's above-90% success there implies that in many cases the context or side effects of the web attack were visible (e.g., a SQL injection attempt might cause an error and an abnormal TCP RST exchange, or maybe the model keyed in on the fact that a certain device never uses the HTTP method or URL that was seen during the attack).

The combined feature approach apparently succeeded in balancing the detection of flooding and stealthy attacks. Flooding attacks were caught through volumetric features, stealthy ones through protocol-specific features, and short-term anomalies. This validates the CIC dataset's proposition that a "novel feature set including handshake details and jitter metrics" enhances anomaly detection. Our results give empirical weight to that claim, showing that we can achieve high detection rates using such features.

The low false positive rate on benign traffic (precision - 0.998 for benign) is significant for IoT contexts. IoT networks, especially in critical systems such as healthcare or industrial, cannot afford too many false alarms that might lead to unnecessary shutdowns or desensitization of operators. Our IDS approach provides reliable alerts: nearly every alert the model would raise

corresponds to a real attack (based on test results). This has to be tempered with the knowledge that our benign data was from the same lab; in the real world, benign IoT behavior could be more varied. But at least within the scope of known device behavior, the model isn't over-triggering.

From a *machine learning perspective*, the performance indicates the model had sufficient capacity and the data was linearly separable to a large extent in some transformed space. The CNN filters likely act like detection rules (somewhat akin to how Snort rules might be, but learned automatically). One filter might fire strongly on patterns corresponding to "TCP SYN flags count high and no ACKs" (port scan), another on "DNS query but unusual answer length" (DNS spoof), and so on. The dense layer then combines these to decide on the class.

Although the model was trained on one dataset, the attacks included are fairly general (common DDoS, standard spoofing, etc.). Thus, the patterns it learned would likely generalize to other IoT networks if the devices behave similarly and attacks manifest similarly in traffic. A concern is whether the model learned anything too specific to the lab environment. For example, if it were discovered that any traffic to IP 192.168.1.100 is benign because that's a known hub, that would be a problem when moving to a new network. We minimized such overfitting by excluding absolute addresses as features. Instead, it's focusing on behavioral aspects (like talk frequency, packet sizes, etc.).

5.2 Limitations of the Study

Dataset bias and realism: The CICIoT2023 dataset, though comprehensive, is still a synthetic testbed dataset. All attacks were executed in a controlled environment. Attack patterns might be somewhat idealized or uniform (e.g., the rate of a DDoS might be stable, whereas a real attacker might modulate).

Attack coverage: The dataset covers 33 attacks, but IoT is a moving target. New attacks (or variations of those attacks) may not be detected. For example, suppose an attacker uses encryption for C&C channels (some modern IoT malware use TLS, which might hide telltale patterns). In that case, our detection of Mirai might fail because we rely on specific plaintext patterns. Another example: side-channel attacks or data exfiltration by modulating sensor readings (a very stealthy attack) wouldn't be caught by our current features. Thus, while we cover the major network-level attacks, the IDS could be bypassed by completely novel techniques that don't trigger the known anomalies.

Imbalanced learning challenges: We managed to get good results on minority classes by careful weighting and the model's capacity to generalize. However, if some classes had even fewer examples (say one or two instances of an attack type), deep learning might struggle. We had at least a few thousand in each class in our training. Capturing enough examples of every possible attack is difficult in real data collection. This raises the issue of *zero-day attacks* or classes with insufficient training data. Unsupervised anomaly detection components could complement our supervised approach to catch those, but we did not implement that here.

Interpretability: Our model, especially CNN, is somewhat interpretable through techniques like SHAP, but overall, deep learning is a black box. In a high-stakes environment, a security analyst might want to know *why* the system flagged a device as under attack. It would say, for example, "this flow was classified as a Web attack with 95% confidence," but not immediately explain it. We partially alleviated this by including human-understandable features (like flags, methods) so one could trace after the fact (e.g., oh, it flagged because of an unusual HTTP method). Still, integrating an explainability module or using inherently interpretable models (like rule-based learners) might be desirable. Some recent research tries to extract rules from neural networks to present to users, which could be future work.

Deployment constraints: Our study didn't explicitly address the deployment architecture. One might deploy this IDS as part of a network gateway or on an edge server aggregating traffic to monitor IoT networks. Handshake, even if the payload is encrypted, that much is usually visible, but deep packet inspection beyond that is not.

Edge vs cloud analysis: If the IDS is cloud-based (sending features to a server), that raises privacy and latency issues.

6 Conclusion

Detection of cyber attacks in IoT systems using deep learning is vital, especially for the future. We have focused on the CICIoT2023 (CIC IoT-DIAD 2024) dataset. We have presented a comprehensive study that spans the creation of a deep learning-based intrusion detection framework, integrating a novel IoT-specific feature set, and evaluating multiple model architectures on a realistic IoT attack benchmark.

We successfully utilized the **CICIoT2023 dataset**, which contains 33 attacks across seven categories executed in an IoT lab environment with 105 devices. This demonstrates that it is a valuable resource for developing and testing IoT intrusion detection systems.

The results confirm that deep learning approaches can significantly outperform traditional machine learning on IoT intrusion detection. For instance, our CNN model outperformed a baseline Random Forest and MLP, especially in detecting minority attack classes.

In summary, the road ahead involves making the IDS more adaptive, interpretable, robust, and integrated. The encouraging results of this thesis provide a solid foundation on which these enhancements can be built. By continuing to refine these systems, we move closer to a future where IoT devices integral to smart homes, cities, and critical infrastructure can operate securely and be resilient against the constant threats in cyberspace.

While this thesis made significant strides, it also opened up several avenues for further research and improvements:

- *Incorporate online learning and adaptation:* IoT environments are not static; new devices and attack patterns will emerge. Future work should explore online learning algorithms or periodic retraining schemes that allow the IDS to adapt to new data. For example, one could use a sliding window approach to continuously update the model with recent benign behavior and any new attack signatures, perhaps using semi-supervised techniques to handle cases when labels are not immediately available.
- *Enhance model explainability:* As noted, deep models are complex. Future research could integrate explainable AI methods to produce human-interpretable alerts. Techniques such as LIME or SHAP can highlight which features drove an alert, but we can also consider rule extraction from neural networks or hybrid approaches (e.g., a neural network that feeds into a decision tree). This would help network administrators trust and verify the system's decisions, a vital industry adoption factor.
- *Extend to additional attack vectors:* The current work focused on network-layer detection. IoT systems could be attacked via other means, such as malicious firmware, side-channel attacks, or exploitation of device-specific logic flaws that may not manifest strongly in network traffic.
- *Integrated device Identification:* Building on the idea of device ID, one could extend the system to detect anomalies and classify the type of IoT device generating each flow

(the DI part of IoT-DIAD). This is useful for inventory and security (some attacks can be identified by the fact that, say, a light bulb device is suddenly sending traffic characteristic of a camera, implying it is posing as something it is not).

- *Real-time dashboard and alerts:* On the implementation side, a future engineering project could create a dashboard interface for the IDS, showing IoT network topology and highlighting where an attack is detected. This would integrate our backend model into a user-friendly monitoring tool for network admins. Visualizing detected attack flows in a time chart could also help investigate incidents.

In summary, the road ahead involves making the IDS more adaptive, interpretable, robust, and integrated. The encouraging results of this thesis provide a solid foundation on which these enhancements can be built. By continuing to refine these systems, we move closer to a future where IoT devices, integral to smart homes, cities, and critical infrastructure, can operate securely and resiliently against the constant threats in cyberspace.

References

[1] ALTALEB, N., SAQIB, N. A. Towards a hybrid machine learning model for intelligent cyber threat identification in smart city environments. Applied Sciences. 2022, Vol. 12, p. 1863.

[2] ASHRAF, R., AHMAD, M., and ABDULLAH, J. Ensemble learning-based approach for effective intrusion detection system. Computers & Security. 2021, Vol. 102, p. 102163.

[3] CHAI, T. and DRAXLER, R. R. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. Geoscientific Model Development. 2014, Vol. 7, No. 3, p. 1247–1250. DOI: 10.5194/gmd-7-1247-2014.

[4] CHEN, D., WAWRZYNSKI, P., LV, Z. Cyber security in smart cities: A review of deep learning-based applications and case studies. Sustainable Cities and Society. 2021, Vol. 66, p. 102655.

[5] CHO, Kyunghyun, VAN MERRIENBOER, Bart, GULCEHRE, Caglar, BAHDANAU, Dzmitry, BOUGARES, Fethi, SCHWENK, Holger, and BENGIO, Yoshua. Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014. DOI: 10.3115/v1/d14-1179.

[6] DAI, G., MA C., and X. XU. Short-term traffic flow prediction method for urban road sections based on space–time analysis and GRU. IEEE Access. 2019, Vol. 7, p. 143025–143035, DOI: 10.1109/ACCESS.2019.2941280.

[7] DAMERI, Renata Paola. Searching for a smart city definition: A comprehensive proposal. International Journal of Computers & Technology. 2013, Vol. 11, No. 5, p. 2544–2551. DOI: 10.24297/ijct.v11i5.1142.

[8] DEY, Rahul and SALEM, Fathi M. Gate-variants of gated recurrent neural networks. 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). 2017. P. 1597-1600. DOI: 10.1109/mwscas.2017.8053243.

[9] DIWAKAR BABU, Pavan Teja. Methodologies for traffic congestion prediction for an IoT-based smart city using machine learning and CNN. SSRN Electronic Journal. 2019. DOI: 10.2139/ssrn.3510057.

[10] DOMÍNGUEZ-BOLAÑO, E., ET AL. A layered architecture for building IoT cyber-security solutions. Sensors. 2022. Vol. 22, No. 12.

[11] DOMINGOS, P., and PAZZANI, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*. 1997, Vol. 29, pp. 103–130.

[12] DOSHI, R., APARNA, S., and ALABADDI, A. Machine learning DDoS detection for consumer Internet of Things devices. *Proceedings of the IEEE Security and Privacy Workshops*. 2018.

[13] ERFANI, S. M., RAJASEGARAR, S., KARUNASEKERA, S., and LECKIE, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognition. 2016, Vol. 58, pp. 121–134.

[14] FERRAG, M. A., MAGLARAS, L., ALAZAB, M., JIANG, J., and KAYES, A. S. M. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative analysis. Journal of Information Security and Applications. 2022, Vol. 66, pp. 103–135.

[15] HAN, K., KANG, M., and KIM, J. A new hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*. 2020.

[16] HUO, Y., ZHANG, H., GUO, Y., and WANG, M. A federated learning-based privacy-preserving scheme for IoT security. Future Generation Computer Systems. 2021, Vol. 118, pp. 453–463.

[17] HOSMER, D. W., LEMESHOW, S., and STURDIVANT, R. X. Applied Logistic Regression, Vol. 398. John Wiley & Sons, 2013.

[18] HOSSAIN, M. S., MOHAMMAD, A., and ULLAH, A. A framework for cyberattack detection using CNN for IoT networks. *Sensors*. 2025.

[19] IMTIAZ, N.; WAHID, A.; ABIDEEN, S. Z. U.; et al. A deep learning based approach for the detection of various Internet of Things intrusion attacks through optical networks. Photonics, 2025, vol. 12, no. 1, p. 35.

[20] KLEINBAUM, D. G., and KLEIN, M. *Logistic Regression: A Self-Learning Text*. Springer, 2010.

[21] KOLIAS, C., KAMBOURAKIS, G., STAVROU, A., and VOAS, J. DDoS in the IoT: Mirai and other botnets. Computer. 2017, Vol. 50, No. 7, pp. 80–84.

[22] LASHKARI, A. H., DRAPER-GIL, G., MAHMUD, A., and GHORBANI, A. A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*. 2017.

[23] MEIDAN, Y., BOHADANA, M., MATHOV, Y., MIRSKY, Y., SHABTAI, A., BREITENBACHER, D., ELOVICI, Y. N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing. 2018, Vol. 17, p. 1222.

[24] MOHAMMAD, R. M., THURAISINGHAM, B., and DUNN, D. Big data analytics and IoT in smart city initiatives. *IEEE Access*. 2021.

[25] NG, A. Y. Feature selection, L1 vs. L2 regularization, and rotational invariance. *Proceedings of the 21st International Conference on Machine Learning (ICML)*. 2004.

[26] NGUYEN, T., BAI, G., and ZHANG, Y. A survey on deep learning techniques for IoT security. IEEE Communications Surveys & Tutorials. 2020, Vol. 22, No. 3, pp. 1977–2008.

[27] NETO, A., DADKHAH, S., FERREIRA, R., ZOHOURIAN, A., LU, R., and GHORBANI, A. A. CICIoT2023: A new dataset for IoT security research. Sensors Journal. 2023, Vol. 23, No. 13, p. 5941.

[28] PRIYANKA, B., and KUMAR, V. Comparative analysis of machine learning algorithms for intrusion detection. *International Journal of Computer Applications*. 2020.

[29] QUINLAN, J. R. C4.5: Programs for Machine Learning. *Morgan Kaufmann Publishers*. 1993.

[30] RAY, P. P. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*. 2018, Vol. 30, No. 3, pp. 291–319.

[31] RISH, I. An empirical study of the naive Bayes classifier. *IJCAI Workshop on Empirical Methods in AI*. 2001.

[32] ROMAN, R., ET AL. On the features and challenges of security and privacy in distributed Internet of Things. Computer Networks. 2013. Vol. 57, No. 10.

[33] ROMAN, R., ZHOU, J., and LOPEZ, J. On the features and challenges of security and privacy in distributed Internet of Things. Computer Networks. 2013, Vol. 57, No. 10, pp. 2266–2279.

[34] SICARI, S., ET AL. Security, privacy and trust in Internet of Things: The road ahead. Computer Networks. 2015. Vol. 76.

[35] SICARI, S., RIZZARDI, A., GRIECO, L. A., and COEN-PORISINI, A. Security, privacy and trust in Internet of Things: The road ahead. Computer Networks. 2015, Vol. 76, pp. 146–164.

[36] SINGH, D., ET AL. A survey of intrusion detection systems for IoT. International Journal of Computer Applications. 2021. Vol. 182, No. 2.

[37] SINGH, D., TRIPATHI, R., and JANGHEL, R. R. A survey of intrusion detection systems for IoT. *International Journal of Computer Applications*. 2021, Vol. 182, No. 2, pp. 1–6.

[38] VINAYAKUMAR, R., ALAZAB, M., SOMAN, K. P., POORNACHANDRAN, P., ALNEMRAT, A., VENKATRAMAN, S. Deep learning approach for intelligent intrusion detection system. IEEE Access. 2019, Vol. 7, p. 4152541550.

[39] VISHWAKARMA, S., and JAIN, A. A survey on DDoS attacks and defense mechanisms in IoT. *Computer Communications*. 2020.

[40] ZHANG, H. The optimality of naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. 2004.

Appendix A: Python source code

Imports

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.utils.multiclass import unique_labels
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.utils import to_categorical
```

Loading and exploring data

```
def main():
    print("🚀 Script started successfully...\n")

    label_map = {
        "BenignTraffic.csv": 0,
        "DictionaryBruteForce (1).csv": 1,
        "DDoS-UDP_Flood20.csv": 2,
        "DDoS-TCP_Flood17.csv": 2,
        "DDoS-HTTP_Flood-.csv": 2,
        "DoS-UDP_Flood (1) (1).csv": 3,
        "DoS-UDP_Flood.csv": 3,
        "DoS-TCP_Flood.csv": 3,
        "DoS-HTTP_Flood.csv": 3,
        "DoS-SYN_Flood.csv": 3,
        "Mirai-greip_flood21.csv": 4,
        "DNS_Spoofing (1).csv": 5,
        "MITM-ArpSpoofing (1).csv": 5,
        "Recon-OSScan.csv": 6,
        "Recon-PingSweep.csv": 6,
        "XSS.csv": 7,
        "BrowserHijacking.csv": 7,
        "SqlInjection.csv": 7,
        "Backdoor_Malware.csv": 7
    }

    class_names = ["Benign", "Bruteforce", "DDoS", "DoS", "Mirai", "Spoofing", "Recon", "Web-based"]
```

```python
# 📈 Line chart (wave graph) for metrics
def save_line_chart(labels, values, title, filename, ylabel):
    plt.figure(figsize=(10, 6))
    plt.plot(labels, values, marker='o', linestyle='-', color='dodgerblue', linewidth=2)
    for i, v in enumerate(values):
        plt.text(i, v + 0.01, f"{v:.2f}", ha='center', fontsize=9)
    plt.title(title)
    plt.xlabel('Classes')
    plt.ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.savefig(filename)
    plt.close()
    print(f"✅ Saved {filename}")
```

```python
dataframes = []
for file, label in label_map.items():
    path = f"files/{file}"
    if os.path.exists(path):
        print(f"  → Loading {file}")
        df = pd.read_csv(path, low_memory=False)
        df["label"] = label
        dataframes.append(df)
    else:
        print(f"❌ Missing file: {file}")

print("✅ Files loaded. Processing data...\n")
data = pd.concat(dataframes, ignore_index=True)
data = data.dropna()
data = data.select_dtypes(include=[np.number])

X = data.drop("label", axis=1)
y = data["label"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_cat = to_categorical(y, num_classes=8)
```

Combine and Preprocess

```python
# Combine and preprocess
full_data = pd.concat(dataframes, ignore_index=True)
full_data = full_data.dropna()
full_data = full_data.select_dtypes(include=[np.number])

print(f"📊 Dataset shape: {full_data.shape}")

X = full_data.drop("label", axis=1)
y = full_data["label"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Reshape for LSTM
X_train_lstm = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test_lstm = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

# 🔁 Train LSTM
print("\n🔁 Training LSTM model...")
lstm_model = Sequential([
    LSTM(64, input_shape=(1, X_train.shape[1]), return_sequences=False),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
lstm_model.fit(X_train_lstm, y_train, epochs=5, batch_size=128, validation_split=0.1)

print("✅ LSTM training complete.")

y_pred_lstm = lstm_model.predict(X_test_lstm).flatten()
mae_lstm = mean_absolute_error(y_test, y_pred_lstm)
rmse_lstm = np.sqrt(mean_squared_error(y_test, y_pred_lstm))
r2_lstm = r2_score(y_test, y_pred_lstm)
```

Training

```
# 🔁 Train CNN
print("\n🔁 Training CNN model...")

# CNN expects shape (samples, time_steps, 1)
X_train_cnn = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test_cnn = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

cnn_model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Dropout(0.3),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model.fit(X_train_cnn, y_train, epochs=5, batch_size=128, validation_split=0.1)

print("✅ CNN training complete.")

y_pred_cnn = cnn_model.predict(X_test_cnn).flatten()
mae_cnn = mean_absolute_error(y_test, y_pred_cnn)
rmse_cnn = np.sqrt(mean_squared_error(y_test, y_pred_cnn))
r2_cnn = r2_score(y_test, y_pred_cnn)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_cat, test_size=0.2, random_state=42)
X_train_lstm = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test_lstm = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

print("🧠 Training LSTM model...")
model = Sequential([
    LSTM(64, input_shape=(1, X_train.shape[1]), return_sequences=False),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(8, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train_lstm, y_train, epochs=5, batch_size=128, validation_split=0.1, verbose=1)
print("✅ Model training complete.\n")
```

Showing results

```
    # Extract metrics
    precision = [report_dict[label]['precision'] for label in present_labels]
    recall = [report_dict[label]['recall'] for label in present_labels]
    f1_score = [report_dict[label]['f1-score'] for label in present_labels]
    support = [report_dict[label]['support'] for label in present_labels]

    print("\n📈 Saving wave-style metric charts as PNG images...")
    save_line_chart(present_labels, precision, "Precision per Class", "metrics_precision.png", "Precision"
    save_line_chart(present_labels, recall, "Recall per Class", "metrics_recall.png", "Recall")
    save_line_chart(present_labels, f1_score, "F1-Score per Class", "metrics_f1.png", "F1-Score")
    save_line_chart(present_labels, support, "Support per Class", "metrics_support.png", "Samples")

    print("\n✅ All charts generated. Look in your folder for:")
    print("- metrics_precision.png")
    print("- metrics_recall.png")
    print("- metrics_f1.png")
    print("- metrics_support.png")

if __name__ == "__main__":
    main()
```