A Framework for Scalable Dataset Generation and Deep Learning-Based IoT Device Identification: Redefining the Future Paradigm

Abdul Fareed Jamali

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Information Systems Security) at

Concordia University

Montréal, Québec, Canada

December 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Abdul Fareed Jamali**

Entitled: **A Framework for Scalable Dataset Generation and Deep Learning-Based IoT Device Identification: Redefining the Future Paradigm**

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

Master of Applied Science (Information Systems Security)

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

Signed by the Final Examining Committee:

_____ Chair
Dr. Manar Amayri

_____ Examiner
Dr. Arash Mohammadi

_____ Supervisor
Dr. Andrea Schiffauerova

Approved by _____
Dr. Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering

_____ 2025

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

A Framework for Scalable Dataset Generation and Deep Learning-Based IoT Device Identification: Redefining the Future Paradigm

Abdul Fareed Jamali

The rapid evolution of automation, ubiquitous connectivity, and remote management capabilities has driven a significant increase in the deployment of Internet of Things (IoT) devices across sectors such as industry, agriculture, transportation, smart cities, smart homes, and healthcare. Despite their growing prevalence, many IoT devices possess limited computational capabilities and rely on lightweight or weak cryptographic mechanisms, leaving them highly vulnerable to a wide range of cyberattacks. As a result, reliable device identification has become a fundamental requirement for enforcing appropriate security policies and strengthening the cybersecurity posture of modern IoT ecosystems.

This thesis addresses the problem of accurate and scalable device identification in smart home IoT environments. While prior research has predominantly explored traditional machine learning techniques and comparative model evaluations, the present study proposes a deep learning–based approach tailored specifically for the heterogeneous and rapidly expanding nature of smart home devices. The research objectives include designing a high-performance identification model, developing realistic and balanced dataset, and enabling incremental learning to support newly added devices without full model retraining.

A detailed examination of widely adopted public datasets reveals key limitations, notably complex network traffic capture setups, inconsistent annotation practices, and significant class imbalance, all of which hinder reliable model training and evaluation. To overcome these challenges and to support both the research community and smart home users collaboration, two traffic-capture frameworks, ScanIoT and DroidScour, were developed. These frameworks simplify device-specific

traffic collection, annotation, and enable the generation of balanced, scalable datasets suitable for both device identification and behavioral analysis.

Recognizing the limitations of existing datasets, especially their laboratory-controlled environments, limited human interaction, and lack of diversity in network conditions, this thesis introduces the Concordia University IoT Device Identification Dataset (CU2025). CU2025 is a well-balanced, high-quality dataset that includes few instances of identical devices operating under real human interaction and varied network environments, making it suitable not only for device identification but also for detailed device behavior analysis.

To perform device identification, a Deep Feedforward Neural Network (DFNN) model is proposed and evaluated on two distinct datasets. The model demonstrates exceptional performance, achieving 99.9% accuracy for both device category and device type classification. To address the continuous integration of new devices in smart homes, an incremental learning strategy is introduced, enabling the model to incorporate additional device types with only marginal performance degradation. An analysis of incremental update sizes further illustrates their effect on accuracy and model stability.

To evaluate practical generalizability, the model's performance is assessed using both conventional train–validation–test splits and a Leave-One-Subject-Out Cross-Validation (LOSO-CV), the latter simulating real-world scenarios involving unseen device instances. The LOSO-CV evaluation yields an accuracy of 98.40%, demonstrating strong robustness and real-world applicability.

Overall, this thesis contributes a comprehensive and scalable approach to smart home IoT device identification, introducing new dataset, traffic-capture frameworks, and a high-performing deep learning model. The findings offer substantial advancements toward practical and adaptive IoT environments capable of addressing the growing challenges of device heterogeneity and cybersecurity.

Dedication

This thesis is dedicated to my family, whose unwavering support sustained me throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The number of interconnected devices on the Internet has been rapidly increasing due to advancements in embedded systems, which have enabled computational power at a relatively lower cost [4, 5]. The ability to remotely control and monitor various sensors, combined with their processing capabilities, has driven the widespread adoption of IoT devices [6]. IoT devices are widely deployed across various sectors, including business, manufacturing, airports, agriculture, industry, smart cities, transportation, and smart homes [7–9].

Recent years have witnessed incidents ranging from compromised user privacy [10, 11] to large-scale DDoS attacks [12, 13], yet the growth of IoT devices remains unaffected [14]. It is expected to reach a market size of 4 trillion dollars by 2032 [15]. The widespread adoption of IoT devices in smart homes has been remarkable, as the physical and privacy security of individuals increasingly relies on smart homes [16]. While various solutions exist to protect enterprise networks [17, 18], securing home networks depends on proper policy enforcement after accurate device identification.

1.2 Motivation

Device identification serves as the foundation for securing small networks by enforcing relevant policies [19]. Identification of IoT devices using machine learning requires a dataset to train and

test the developed model. There are various implementations aimed at generating datasets for device identification, both in trusted [20, 21] and compromised [3] environments. However, the underlying complexities of implementation hinder collaboration in the effort to generate an expanding dataset.

We have explored the limitations of the existing datasets used for device identification and have proposed two frameworks to collect IoT device traffic for smart homes. ScanIoT [22, 23] and DroidScour [24, 25] are two data collection and annotation frameworks developed to capture and label IoT device traffic in smart homes. Although both tools are designed specifically for IoT traffic collection and labeling, they can also be used to collect traffic from any device connected to Wi-Fi.

ScanIoT and DroidScour are designed to enable collaboration between smart home users and the academic community in creating scalable datasets. Their user-friendly interfaces allow both technical and non-technical users to collect network traffic with ease. These tools revolutionize the device identification problem by overcoming limitations in the number of IoT devices and incorporating replicated devices across diverse networks. This helps build robust machine learning models capable of identifying IoT devices regardless of network-specific characteristics.

IoT device identification can be achieved through physical, network, or application layer-based techniques. Device identification through the physical layer relies on intrinsic hardware characteristics associated with the system's circuit design [26, 27]. Network layer-based identification of IoT devices relies on communication protocols and data transmission patterns [28, 29]. Device identification on the application layer depends on a combination of application-layer information, blockchain, and cryptographic techniques [30, 31]. The scope of our study includes features for device identification from the TCP/IP layer, relying on network headers, statistical extraction of meaningful data, and cross-layer analysis. Our analysis does not rely on the hardware of IoT devices or any cryptographic techniques.

The lack of a scalable dataset poses a significant hurdle to the real-world implementation of device identification. With billions of devices connected to the Internet—many of which are IoT devices—the absence of tools to scale dataset collection limits device identification efforts to theoretical research. The adoption of ScanIoT and DroidScour frameworks paves the way for the creation of scalable datasets. By incorporating a wide range of IoT devices and their replicas across diverse network environments, these tools support the development of robust and adaptable machine-learning

algorithms.

A deep feedforward neural network (DFNN) has been proposed to identify IoT devices along with their respective categories. To accommodate the continuous introduction of new devices, incremental learning techniques have been incorporated into the model. The DFNN performs classification of both individual devices and their categories using separate, distinct labels. This study also examines the evolving standards for IoT device identification and highlights prospective advancements, emphasizing the critical features that future identification frameworks should integrate.

1.3 Objectives

The primary objectives of this thesis are as follows:

- To develop traffic-capture framework that eliminate the need for complex network deployments typically required for device identification research.
- To collect and curate a well-balanced, high-quality dataset suitable for IoT device identification and behavioral analysis.
- To design a deep learning model capable of accurately identifying IoT devices and to validate its robustness by evaluating it on an additional, independent dataset.
- To extend the proposed model with an incremental learning capability that allows it to incorporate new devices without full retraining, and to assess how the scale of newly added devices impacts overall performance.
- To establish an evaluation methodology for handling duplicate devices, enabling assessment of the model's effectiveness in identifying previously unseen device instances.

1.4 Contributions

The contributions of our study are outlined below.

- To our knowledge, this is the first study to examine both the shortcomings of existing IoT datasets and the intricate network constraints that hinder collaborative efforts to scale datasets

for device identification. The lack of genuine human interaction and the restricted growth of these datasets limit their ability to adapt to the dynamically evolving IoT ecosystem.

- We propose two open-source frameworks, ScanIoT and DroidScour, designed to collect and label IoT traffic without requiring complex network deployments. These tools can capture device-specific traffic within any network and are capable of simultaneously collecting traffic from multiple devices. These frameworks does not require any complicated and expensive network deployments and enable the generation of scalable datasets that can adapt to the dynamically evolving IoT ecosystem.

- Using our proposed frameworks, we collected traffic from various IoT devices to construct the Concordia University IoT Device Identification Dataset (CU2025), a well-balanced and high-quality dataset.

- The study presents a Deep Feedforward Neural Network (DFNN) model designed for the identification of Internet of Things (IoT) devices and their respective categories. In addition, an incremental learning approach is also proposed to enable the integration of newly introduced devices without necessitating complete retraining of the model. Furthermore, the research investigates the impact of varying the number of incrementally added devices on the overall performance of the model accuracy.

- To assert the effectiveness of our proposed model, we used the Leave-One-Subject-Out Cross-Validation (LOSO-CV) technique for assessing IoT device identification in real-world scenarios.

1.5 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2** provides an overview of the foundational concepts and principles relevant to this thesis.

- **Chapter 3** presents a literature review on device identification techniques and discusses the limitations of existing datasets.
- **Chapter 4** discusses the proposed ScanIoT and DroidScour frameworks, which were developed to collect and annotate IoT device traffic.
- **Chapter 5** presents the transformation of the CICIoT2023 dataset into device identification focused dataset and details the Concordia University IoT device identification dataset (CU2025).
- **Chapter 6** provides a comprehensive overview of the methodology, encompassing the preprocessing pipeline applied to the datasets, the procedures for data preparation, the design and implementation of the model architecture, the incremental learning framework, and the evaluation strategies employed to assess model performance.
- The results of our findings are presented in **Chapter 7**.
- Discussion of findings are presented in **Chapter 8**.
- **Chapter 9** presents the conclusion of our work, discusses its limitations, and outlines potential directions for future research.

Chapter 2

Background

This chapter provides a comprehensive overview of the foundational concepts that underpin this research, encompassing the Internet of Things (IoT), IoT device identification, and the machine learning methodologies and their corresponding applications.

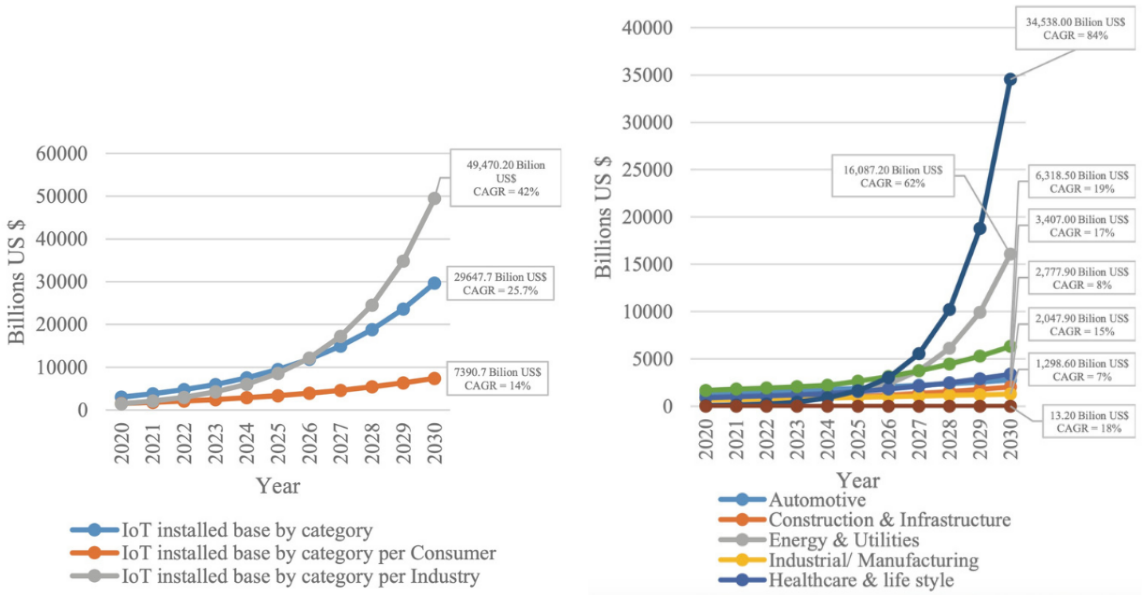
2.1 Internet of Things

This section offers an examination of Internet of Things (IoT) devices, addressing their market evolution, application spectrum, essential security terminologies, common attacks, and prominent incidents that highlight the security challenges inherent to IoT environments.

2.1.1 Overview

Kevin Ashton first introduced the concept of the Internet of Things (IoT) in 1999 during a presentation at Procter & Gamble [9]. The IoT architecture integrates sensors and actuators that operate collaboratively, enabling data acquisition and control through Internet. Equipped with heterogeneous sensors and onboard processing units, IoT devices are inherently versatile and can be readily deployed in a wide range of application environments [32]. This inherent adaptability has positioned IoT systems as a foundational technology across multiple sectors, encompassing industry, agriculture, transportation, smart cities, smart homes, and healthcare [33].

Rapid advancements in embedded systems have catalyzed the unprecedented growth of the IoT



(a) Expected growth of IoT per consumer and industrial applications.

(b) IoT spending by sector.

Figure 2.1: IoT market analysis and forecast 2020-2030 [1].

ecosystem, with more than 120 devices joining the Internet each second. Forecasts predict that by 2030, the global network of interconnected IoT devices will exceed 125 billion [34]. Al-Sarawi et al. [1] conducted a comprehensive analysis of the IoT market, as depicted in Figure 2.1. The projected expansion of IoT in industrial applications is shown in Figure 2.1a, whereas Figure 2.1b presents the anticipated IoT adoption trends across different sectors by 2030 including healthcare and lifestyle, automotive, construction and infrastructure, energy, and industry.

While IoT devices enable remote execution and monitoring of processes, many are rushed to market to maintain competitive timing, often at the expense of robust security measures. Although their computational capabilities are increasing, these resources are largely dedicated to core sensing and actuation functions, with limited attention to safeguarding communications or adhering to standardized security protocols [35]. Consequently, the widespread deployment of IoT devices has created complex technical, security, and operational challenges that must be addressed to ensure safe, reliable, and scalable adoption across domains [36].

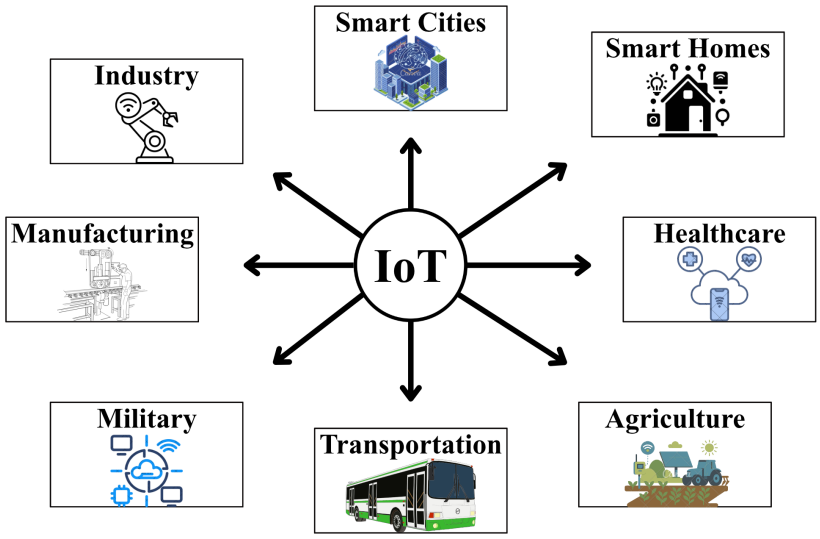


Figure 2.2: Internet of Things application.

2.1.2 IoT application

Figure 2.2 provides an overview of the diverse application domains of the Internet of Things (IoT). Within the industrial domain, IoT devices play a critical role in automating asset tracking, robotics, and process optimization. They are instrumental in monitoring parameters such as vibration and temperature of industrial machines to support predictive maintenance and minimize downtime. Moreover, IoT-based systems enable remote supervision of energy consumption in industrial equipment. These specialized industrial IoT applications are collectively known as the Industrial Internet of Things (IIoT) [37, 38].

In the agricultural sector, the integration of IoT technologies has led to substantial improvements in crop yield and production efficiency. By facilitating real-time monitoring of environmental factors such as soil moisture, pH, and temperature, IoT devices support precision farming and data-driven decision-making. Furthermore, the use of unmanned aerial vehicles (UAVs) for crop health assessment and livestock monitoring enables early detection of issues, optimizing productivity and resource utilization [39–41].

In the transportation domain, IoT devices play a vital role in enabling fleet telematics and vehicle tracking through Global Positioning System (GPS) integration. Such connectivity allows vehicles to interact with transit infrastructure, enhancing route optimization and operational efficiency.

Additionally, IoT-driven communication frameworks contribute to intelligent traffic management systems, enabling dynamic control of traffic flow and congestion mitigation [42].

Within smart city infrastructures, IoT devices play a pivotal role in optimizing resource management and service delivery. Smart grids leverage IoT connectivity to automate energy distribution, balance load demands, and enhance overall energy efficiency. Furthermore, IoT-based solutions such as adaptive street lighting, smart parking systems, air quality monitoring, and waste-bin tracking exemplify the diverse applications that contribute to sustainable and intelligent urban ecosystems [43].

In the military and defense domain, IoT technologies play a crucial role in enhancing operational capability and situational awareness. Applications include wearable health monitoring systems for soldiers, drone-based surveillance and targeting, voice-assisted communication, and remote equipment control. Furthermore, IoT-enabled smart logistics and asset tracking systems streamline supply chain management and resource allocation, contributing to more efficient and responsive defense operations [44].

In healthcare, IoT technologies have become increasingly significant, facilitating remote patient monitoring through smartwatches, wearable sensors, and biosensing devices. Beyond patient care, IoT systems streamline hospital asset tracking and inventory management, improving operational efficiency. Furthermore, applications such as smart beds, patient tracking, telemedicine, virtual consultations, early diagnostic systems, and intelligent medication dispensers exemplify how IoT integration is transforming healthcare delivery and patient outcomes [45, 46].

The adoption of IoT devices in smart homes has been increasing rapidly. Figure 2.3 illustrates a typical smart home, where diverse IoT devices are interconnected to manage lighting, plugs, doors, temperature, moisture levels, smart wearables, cameras, and Internet connectivity. As shown in Figure 2.3, smart cameras, smart plugs, and other purpose-built devices often have limited processing power and memory, which constrains their capabilities and results in Internet connections with minimal or basic security configurations [47]. These limitations make IoT devices attractive targets for adversaries, increasing the risk of cyberattacks that exploit their vulnerabilities [48]. Beyond the aforementioned domains, IoT applications have also emerged in sectors such as aviation [49], education [50], and forestry [51], providing enhanced monitoring, operational efficiency, and intelligent

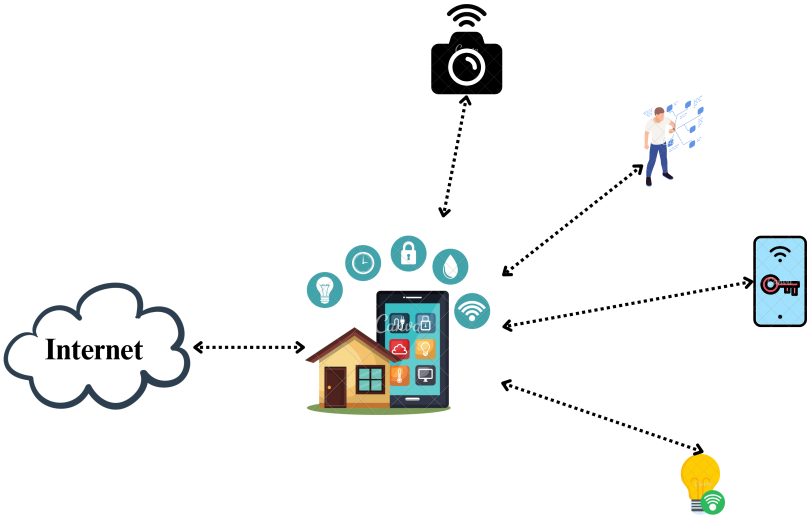


Figure 2.3: A typical smart home.

resource management.

2.1.3 IoT Security

The exponential growth of IoT deployments has presented substantial challenges in effective device management and operation [52]. While IoT technologies facilitate remote monitoring and control, their rapid and widespread adoption has exposed significant vulnerabilities, particularly in terms of security. Competitive market dynamics often compel manufacturers to accelerate product releases, frequently neglecting thorough security evaluation. As a result, many IoT devices are introduced with inadequate or minimal cyber protection mechanisms, rendering them susceptible to exploitation and making them prime targets for cyberattacks.

In addition to design and deployment challenges, constraints such as limited processing power and inadequate cryptographic [53, 54] implementations further weaken the cybersecurity posture of IoT systems. Furthermore, the increasing prevalence of the Bring Your Own Device (BYOD) paradigm across various network environments enlarges the attack surface, as potentially insecure devices gain access to sensitive network infrastructures [55].

Security terminologies

This section briefly defines fundamental security terminologies, including confidentiality, integrity, availability, vulnerability, exploit, threat, risk, and malware.

- **Confidentiality** refers to the property of safeguarding information so that it is accessible exclusively to authorized individuals or systems, thereby preventing unauthorized disclosure or exposure.

- **Integrity** refers to the assurance that information is maintained in a correct, complete, and unmodified state during storage, processing, and transmission, preserving its trustworthiness from origin to destination.

- **Availability** refers to the property of ensuring that information, applications, and IoT systems remain accessible and functional for authorized users whenever needed, minimizing downtime and service disruptions.

- A **vulnerability** can be defined as a weakness or deficiency in the design, implementation, operation, or configuration of an IoT device that may be exploited by adversaries to compromise the device's confidentiality, integrity, or availability.

- An **exploit** refers to a technique, program, or tool that leverages a vulnerability within a software application or device to induce unintended or malicious behavior. These behaviors may manifest as unauthorized access to system resources, data exfiltration, or disruption of essential services, thereby compromising the confidentiality, integrity, or availability of the system.

- A **threat** refers to any event, entity, or condition that has the potential to exploit a vulnerability in an IoT device or system, thereby causing harm to its operations, compromising its data, or disrupting its intended functionality.

- A **risk** represents the potential adverse impact or loss that may result from a threat exploiting a vulnerability in an IoT device or system, thereby affecting its confidentiality, integrity, or availability.

Table 2.1: IoT Scenarios with Associated Threats, Vulnerabilities, and Risks

Scenarios	**Threat**	**Vulnerability**	**Risk**
Compromised smart home camera	Cybercriminal exploiting weak credentials	Default admin password	Unauthorized surveillance, user privacy breach
Failure of Industrial IoT sensor	Malware infection	Unpatched firmware	Production downtime, safety hazard
Compromised healthcare IoT devices	Network attack	Unsecured wireless communication	Patient data theft, potential harm to patient
Smart thermostat manipulation	Hacker exploiting weak network security	Default or weak Wi-Fi password	Unauthorized control of thermostat, energy loss, discomfort
Autonomous vehicle sensor hack	Cybercriminal injecting false sensor data	Insecure communication protocols	Vehicle collision, traffic disruption, injury
Smart factory robot tampering	Insider threat or malware	Lack of access control and unpatched software	Production errors, equipment damage, safety hazard
Compromised wearable health tracker	Malware or phishing attack	Weak encryption of transmitted health data	Exposure of sensitive health information, privacy violation
Smart grid energy system	DDoS attack on IoT controllers	Poor network segmentation	Power outages, grid instability, economic loss
Agricultural IoT sensor breach	Remote attacker exploiting firmware vulnerability	Outdated firmware, unsecured IoT gateway	Crop mismanagement, resource waste, financial loss

- **Malware** refers to any software intentionally created to compromise the normal operation of systems or networks, enabling unauthorized access, disrupting services, or inflicting damage on hardware, software, or data.

Table 2.1 illustrates several scenarios to elucidate the relationships between threats, vulnerabilities, and risks in IoT environments.

Common attacks

This section provides a brief overview of several well-known attacks targeting IoT devices and systems.

- Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks represent some of the most significant threats to IoT infrastructures, aiming to disrupt service availability

by inundating devices or networks with a massive volume of requests. Common techniques employed in these attacks include ACK fragmentation [56], Slowloris [57], ICMP, HTTP, UDP, and TCP floods [58–60], as well as RST-FIN floods [61], PSH-ACK floods [62], UDP and ICMP fragmentation [63, 64], SYN floods [65], Synonymous IP floods [66], and mirai attack [12, 67].

- Reconnaissance attacks represent an initial phase in which an adversary systematically collects information about a target to inform and facilitate subsequent exploitation. Typical techniques employed during reconnaissance include host discovery [68], ping sweeps [69], port scanning [70], operating-system scanning [71], and vulnerability scanning [72]; these methods aim to reveal network topology, exposed services, software versions, and known weaknesses that can be leveraged in later attack stages.

- Spoofing constitutes another category of cyberattacks in which adversaries impersonate legitimate users or systems to gain unauthorized access, exfiltrate sensitive information, or propagate malicious software [73]. Among the most prevalent forms of spoofing are Domain Name System (DNS) spoofing [74] and Address Resolution Protocol (ARP) spoofing [75], both of which manipulate network communication protocols to redirect traffic or intercept data for malicious purposes.

- Web-exposed services hosted on Internet-of-Things (IoT) devices are frequent targets for attackers who exploit application-layer vulnerabilities. Common web-based attack vectors include SQL injection [76], command injection [77], cross-site scripting (XSS) [78], browser hijacking [79], backdoor malware [80], and unauthorized file-upload [81]; these techniques can facilitate data exfiltration, remote code execution, session compromise, establishment of persistent footholds, and malware distribution.

Reported incidents

This section presents a concise overview of several well-documented incidents involving the compromise of IoT devices.

- The Mirai botnet, first identified in 2016, specifically targeted a broad spectrum of Internet of Things (IoT) devices, such as Internet Protocol (IP) cameras, digital video recorders (DVRs), and routers, by exploiting default or weak authentication credentials. This widespread exploitation enabled adversaries to conscript compromised devices into a large-scale botnet, which was subsequently leveraged to execute distributed denial-of-service (DDoS) attacks of unprecedented magnitude [12].

- The Jeep Cherokee cybersecurity incident demonstrated a critical vulnerability that allowed security researchers to remotely manipulate the vehicle's control systems through its Internet of Things (IoT) connectivity features. This exposure highlighted severe risks associated with connected vehicle technologies and consequently prompted the recall of approximately 1.4 million vehicles to implement essential security updates [82].

- The WannaCry ransomware attack exploited unpatched vulnerabilities in industrial Internet of Things (IIoT) systems as well as IoT-connected healthcare devices and networks. This large-scale cyber incident caused widespread disruption to critical infrastructure, including hospital operations, and resulted in an estimated global economic loss of approximately $4 billion [83].

- The well-known casino aquarium thermometer breach exploited a vulnerability in a network-connected IoT temperature sensor, enabling attackers to infiltrate the casino's internal network and gain unauthorized access to the high-roller customer database. This incident resulted in significant reputational damage and the exposure of sensitive client information [84].

- In the Ring doorbell incident, attackers exploited inadequate security configurations in IoT-enabled cameras, allowing them to harass users and invade their privacy through unauthorized access. This breach underscored critical security weaknesses in consumer-grade smart home IoT devices and emphasized the urgent need for stronger privacy and access control mechanisms [85].

- Garmin experienced a significant ransomware attack that disrupted the operation of its fitness-related Internet of Things (IoT) ecosystem. The incident resulted in the encryption and temporary inaccessibility of critical services, including those supporting fitness tracking and aviation systems. The operational downtime severely impacted users worldwide and reportedly culminated in a ransom payment estimated at approximately $10 million to restore functionality [86].

- Another notable incident targeting critical energy infrastructure was the colonial pipeline ransomware attack, in which threat actors exploited compromised credentials to infiltrate the organization's IoT-integrated operational systems. The intrusion led to the temporary shutdown of pipeline operations, disrupting fuel distribution across multiple U.S. states. The company reportedly paid a ransom of approximately $4 million to regain access and restore normal functionality [87].

- In the Verkada camera breach, a large-scale compromise of IoT-based surveillance systems occurred when attackers obtained administrative credentials, granting unauthorized access to approximately 150,000 cameras. The incident resulted in the exposure of live video feeds and recorded footage from sensitive environments, including hospitals, correctional facilities, and industrial sites such as Tesla factories, thereby highlighting severe privacy and security vulnerabilities within cloud-managed IoT ecosystems [88].

- The Oldsmar Water Treatment Plant cyberattack represents one of the most critical incidents targeting public utilities, wherein adversaries gained unauthorized access to the facility's IoT-enabled Supervisory Control and Data Acquisition (SCADA) systems. The attackers attempted to manipulate the chemical composition of the drinking water supply, posing a significant potential threat to public health. Fortunately, the intrusion was detected promptly, and corrective actions were taken to restore normal operations before any harm could occur [89].

- One of the most recent and significant cybersecurity incidents is the MOVEit data breach, which targeted a managed file transfer application by exploiting a zero-day vulnerability in systems associated with IoT environments. This breach resulted in the exposure of millions

of sensitive records and incurred financial penalties exceeding $100 million [90].

2.2 Device Identification

This section provides an overview of device identification, emphasizing the granularity of identification in IoT environments and examining the machine learning approaches employed for accurate device recognition.

2.2.1 Overview

As the incidence of cyber threats within the Internet of Things (IoT) landscape continues to escalate, accurate device identification emerges as a fundamental requirement for recognizing and managing individual devices across interconnected infrastructures [91]. The implementation of reliable identity validation mechanisms is imperative to uphold system security, trust, and operational resilience. A range of methodologies have been developed for device identification, encompassing approaches based on intrinsic physical hardware properties, radio frequency (RF) signal profiling, and network traffic characterization and pattern analysis. Brief description of each methodology follows.

- Device identification at the physical layer relies on the intrinsic behavior of hardware components to distinguish IoT devices. These characteristics are produced by the analog circuitry, which varies across different devices. Variations in manufacturing processes introduce subtle deviations in circuitry, resulting in unique signatures that can be leveraged to identify individual IoT devices [26].

- Device identification at the network layer relies on data received from the physical layer, which is subsequently transmitted to higher protocol layers. The network layer serves as the backbone of the IoT infrastructure, encompassing routing and transmission information that can be analyzed to facilitate device identification [28].

- The application layer receives data from the network layer and represents the outermost layer

responsible for fulfilling operational requirements across diverse domains. The information contained within this layer enables analytical processes that derive meaningful insights from device-related tasks and interactions. Device identification at this layer typically leverages application-specific communication protocols, blockchain-based frameworks, and cryptographic techniques to ensure secure and reliable recognition of IoT devices [92].

For device identification to serve as an effective and robust solution, several desirable characteristics must be considered. Firstly, it should enable the unique identification of each device within the networked environment. Secondly, it should provide meaningful information, reflecting both the device's identity and its functional role within the system. Thirdly, the identification mechanism should be concise, facilitating ease of management and operational simplicity. Finally, it must be secure and efficient, particularly in lightweight or resource-constrained applications, allowing the system to scale without compromising performance or disrupting normal operations [9].

2.2.2 Granularity of device identification

The requirements for IoT device identification vary according to environmental and operational demands. Depending on the context, device identification can be classified into three primary levels: device category identification, device type identification, and device instance identification. The brief explanation of each level of device identification is presented below.

- **Device category identification** provides information about the general category of the device, such as cameras, audio devices, hubs, lighting systems, power outlets, or home automation equipment.
- **Device type identification** offers more detailed information, specifying particular devices within a category, for example, Google Home Mini, Nest Camera, Amazon Alexa, Amazon Echo Dot, or D-Link Camera.
- **Device instance identification** provides the most granular level of detail, capturing not only the device type but also its specific generation or model. Examples include Google Nest Mini (2nd generation), Amazon Echo Dot (3rd generation), or Blink Mini 2 Camera,

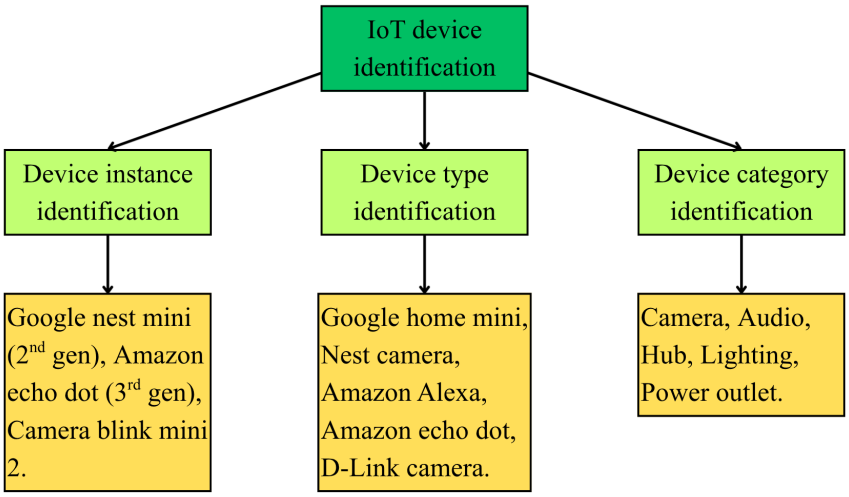


Figure 2.4: IoT device identification granularity.

Figure 2.4 illustrates the granularity of device identification in relation to network requirements and operational demands.

2.2.3 Significance

Device identification holds significant importance within networked environments, serving as a foundation for functions ranging from continuous monitoring to the enforcement of security policies. It plays a pivotal role in enhancing network security across various applications. Accurate identification of device category, type, and instance contributes to strengthening the security posture of the evolving IoT ecosystem, where such devices often represent the initial targets of cyberattacks due to their limited computational capabilities and susceptibility to compromise for large-scale attacks [93]. Several significant aspects of IoT device identification follows.

- **Policy enforcement:** Within a network, device category identification facilitates precise policy enforcement, thereby enhancing the protection of individual devices and, consequently, the overall network [2]. Moreover, device type and instance identification enable the detection of related or potentially compromised devices across interconnected systems, supporting effective network management and incident response. When IoT device identification systems

are integrated within larger, interconnected networks, they can further contribute to global security efforts by providing timely and relevant information regarding compromised devices.

- **Device validation:** Device identification can play a crucial role in mitigating incidents caused by identity spoofing. Device forgery, commonly referred to as identity spoofing, occurs when an adversary obtains a legitimate device identifier, such as a MAC address, to impersonate the victim device. A naive approach to verification involves authenticating devices solely based on their MAC addresses; however, this method is inherently weak, as such identifiers can be easily forged or manipulated, rendering it ineffective for robust device identity validation [94].

- **Device authentication:** While user authentication has long been an integral component of network security, the need to authenticate devices within a network is becoming increasingly critical. Device identification can address this requirement by verifying the known devices within the network, thereby preventing unauthorized access and mitigating potential adversarial activities [95].

- **Device restriction:** The Bring Your Own Device (BYOD) policy permits employees to connect their personal devices to organizational networks, often resulting in the presence of "shadow devices" that may not be fully regulated or monitored [96]. These devices introduce significant security challenges to the network infrastructure. However, through effective device identification mechanisms, unauthorized or unmanaged devices attempting to access organizational servers can be detected and subsequently restricted, thereby enhancing the overall security posture of the organization.

- **Tracking:** Traditionally, user tracking has relied on the use of cookies; however, the increasing deletion of cookies by users, growing privacy concerns, and the default blocking of cookies by modern browsers have significantly limited their effectiveness. Device identification offers an alternative approach by enabling the tracking of user activity through unique device attributes, thereby providing a more persistent and privacy-aware mechanism for online activity analysis [97].

- **Operation Continuity:** The increase in cyber incidents targeting Industrial Control Systems

(ICS), which manage critical infrastructure, has grown exponentially. Such incidents can have devastating consequences, including machine downtime, unwanted blackouts, economic losses, and environmental disasters. Implementing a centralized device identification system allows organizations to monitor all connected devices, identify those requiring attention, and proactively alert cybersecurity experts, thereby reducing the likelihood and impact of cyber incidents [98, 99].

2.3 Machine and Deep learning

Machine Learning (ML), a subfield of Artificial Intelligence (AI), enables computers to learn patterns from data to make decisions or predictions without explicit programming. Common ML techniques include Decision Trees (DT), Support Vector Machines (SVM), Random Forests (RF), and Neural Networks (NN). Deep Learning (DL), a specialized branch of ML, employs Artificial Neural Networks (ANNs) composed of multiple layers, hence the term deep, to automatically learn hierarchical data representations. DL methods are widely applied to complex tasks such as speech recognition, image classification, and natural language processing. In the subsequent sections, we examine the application of these techniques in the context of IoT device identification.

2.3.1 Supervised Machine Learning

Supervised machine learning is a type of machine learning that relies on labeled data to train algorithms to make predictions on unseen testing data. The labeled dataset provides the necessary input-output pairs, allowing the algorithm to learn the relationship between features and the corresponding outputs. During training, the algorithm iteratively processes the data to adjust its parameters and optimize performance, ultimately achieving accurate predictions. Supervised learning is widely used for both classification and regression tasks. Commonly applied algorithms include Decision Trees, Random Forests, Support Vector Machines, k-Nearest Neighbors, Naïve Bayes, Logistic Regression, and Gradient Boosting. A brief description of each algorithm follows.

Decision Tree

Decision Tree (DT) algorithms, as the name suggests, operate through a hierarchical series of if–then decision rules derived from labeled data, such as network traffic patterns, to determine device types. DTs can be employed for both classification and regression tasks [100]. DTs are commonly employed for device classification tasks using features including packet size, packet inter-arrival time, and communication protocols. DTs are relatively simple, non-parametric, interpretable, and computationally efficient; however, they are prone to overfitting if not appropriately pruned. DT are used for both classification and regression purposes.

Random Forest

Random Forest (RF) is an ensemble learning method that constructs multiple Decision Trees (DTs) and generates predictions based on majority voting among them [101]. RF models are employed not only for device identification but also for detecting rogue or anomalous devices within networks by analyzing aggregated network statistics. They generally exhibit high accuracy and demonstrate robust performance even in the presence of noise. However, similar to DTs, RF models are susceptible to overfitting and their performance can degrade when dealing with highly imbalanced datasets.

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning technique that determines the optimal decision boundary, known as a hyperplane, between different classes based on features extracted from network data [102]. SVMs can be applied to both classification and regression tasks. In the context of IoT identification and security, SVMs are utilized to detect abnormal or unauthorized devices by distinguishing their traffic patterns from those of known, labeled network traffic.

k-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm classifies unknown devices by comparing them with the k most similar known devices based on feature distance metrics [103]. kNN enables rapid classification, making it well-suited for lightweight IoT gateway applications that require real-time device identification. The algorithm is simple, intuitive, and easily interpretable; however, its performance degrades with large datasets due to increased computational complexity and it remains sensitive to noisy data.

Naïve Bayes

Naïve Bayes (NB) is a probabilistic algorithm that assumes feature independence and computes the likelihood of each class based on this assumption [104]. NB models are commonly applied to identify known device types when network features, such as network ports, packet sizes, and protocols, are conditionally independent. They are computationally efficient, require minimal resources, and are therefore well-suited for resource-constrained IoT environments. However, the simplifying independence assumption may reduce accuracy in more complex or highly correlated feature spaces.

Logistic Regression

Logistic Regression (LR) is a probabilistic algorithm used to determine the likelihood that a device belongs to a particular class. LR is widely utilized for binary classification tasks, such as distinguishing between authorized and unauthorized devices or differentiating IoT and non-IoT devices [105]. It serves as a simple yet effective baseline model; however, its applicability is limited in scenarios involving complex or nonlinear relationships among features.

Gradient Boosting

Gradient Boosting (GB) is an ensemble learning technique that constructs a strong predictive model by sequentially combining multiple weak Decision Trees (DTs) [106]. GB has demonstrated

high accuracy in device-type identification based on corresponding network traffic patterns and behavioral logs. It effectively handles mixed feature types and exhibits excellent overall performance. However, GB requires extensive hyperparameter tuning and is computationally intensive, which may limit its suitability for resource-constrained IoT environments.

2.3.2 Unsupervised Machine Learning

Unsupervised machine learning, unlike supervised learning, does not rely on labeled datasets for training. Instead of learning from predefined input-output pairs, it identifies inherent patterns and structures within the data to extract meaningful insights and make predictions. Unsupervised learning techniques are commonly applied in domains such as fraud detection, customer behavior analysis, and market segmentation. These approaches aim to uncover natural groupings or detect anomalies within data, making them particularly suitable for applications such as IoT device identification, rogue device detection, and behavioral profiling. A brief overview of several unsupervised learning techniques is presented below.

Clustering

Clustering, as the name suggests, groups data points based on similarity, where each point may either belong exclusively to one cluster or have varying degrees of membership across multiple clusters. In k-means clustering, each data point is assigned to a single cluster, resulting in a straightforward classification approach with relatively low computational complexity. In contrast, Fuzzy c-means clustering assigns membership weights to each data point for all clusters, thereby capturing uncertainty and overlap among groups, albeit at a higher computational cost due to iterative membership updates. Other notable clustering techniques include DBSCAN, a density-based method capable of automatically identifying clusters and anomalies; hierarchical clustering, which constructs a tree-like cluster structure; and Gaussian Mixture Models (GMM), a probabilistic approach that models overlapping groups using a combination of Gaussian distributions [107].

Principal Component Analysis

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and feature extraction [108]. It transforms the data into a set of principal components that capture the maximum variance within the dataset, thereby enabling efficient visualization and preprocessing of IoT traffic data for subsequent clustering or classification tasks. In this way, PCA can help highlight device-specific features. The method is simple, interpretable, and computationally efficient; however, its primary limitation is that it captures only linear relationships among features.

Isolation Forest

Isolation Forest is a powerful and widely adopted unsupervised machine learning algorithm for anomaly and outlier detection, particularly in Internet of Things (IoT) and Industrial Control System (ICS) environments. It operates as an ensemble of Decision Trees (DTs) and is based on the principle that anomalies are easier to isolate than normal data points. Isolation Forest performs effectively on high-dimensional and large-scale datasets, offering scalability and robustness even in the presence of noise [109]. However, it has certain limitations, while it excels at detecting global anomalies, it may fail to identify local outliers within dense clusters. Additionally, the inherent randomness in tree construction can introduce minor variability in results, and the model's lack of interpretability makes it challenging to pinpoint the specific cause of detected anomalies.

2.3.3 Deep Learning

Deep learning, a subset of machine learning, utilizes Artificial Neural Networks (ANNs) composed of multiple layers, hence the term "deep", to automatically learn hierarchical representations of data. It is particularly valued for its ability to capture complex patterns, handle high-dimensional datasets, and model nonlinear relationships, making it suitable for a wide range of applications. Deep learning techniques have been extensively applied to challenging tasks such as speech recognition, image classification, natural language processing, IoT device identification, cybersecurity, healthcare, and autonomous systems [110]. A brief overview of several deep learning techniques is provided below.

Feed-Forward Neural Networks

The Feed-Forward Neural Network (FNN) is a foundational deep learning architecture and represents the simplest form of an Artificial Neural Network (ANN). In an FNN, information flows unidirectionally from the input layer through one or more hidden layers to the output layer, without any loops or feedback connections. Each neuron in a given layer is fully connected to all neurons in the subsequent layer, and the network learns by adjusting these connection weights during the training process. Through iterative optimization, FNNs learn to map input data to output predictions by recognizing underlying patterns in the data.

FNNs have been widely applied across diverse domains, including computer vision, finance, healthcare, cybersecurity, manufacturing, and IoT systems, particularly for tasks involving pattern recognition, classification, and prediction [111]. The network accepts input data, processes it through hidden layers using weighted transformations and activation functions, and produces predictions at the output layer. During training, the network updates its weights to minimize prediction error.

FNNs offer several advantages, including their simple design, ease of training and interpretation, and their theoretical capability as universal approximators, able to model any continuous function given sufficient neurons and layers. They also enable fast inference, as trained models can process new data efficiently. Furthermore, FNNs are versatile, supporting both classification and regression tasks, and serve as the foundational architecture for more advanced neural network models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep learning architecture specifically designed to learn spatial hierarchies and automatically extract meaningful patterns from structured data such as images, signals, or time-series data. Unlike the previously discussed Feed-Forward Neural Network (FNN), which treats all inputs equally, CNNs employ convolutional operations to focus on localized regions of the input, mimicking the way the human visual system perceives and processes visual scenes.

A typical CNN consists of multiple layers. The input layer receives raw data, while the convolutional layers apply filters (also referred to as kernels) that slide across the input to extract relevant features, with each filter learning a specific pattern or attribute. The activation layers introduce nonlinearity through activation functions, enabling the network to model complex relationships. The pooling layers perform dimensionality reduction, enhancing computational efficiency by retaining only the most salient features. The fully connected layers integrate the extracted features to produce the final classification or detection, and the output layer generates the corresponding prediction [112].

CNNs are highly effective for data with spatial, temporal, or structural dependencies, making them widely applicable in computer vision, healthcare, cybersecurity, autonomous systems, IoT and smart city infrastructures, and manufacturing domains. Their advantages include automatic feature extraction directly from raw input data, translation invariance through spatial feature learning, high accuracy, reduced parameter complexity compared to FNNs due to parameter sharing, and scalability for large, high-dimensional datasets. However, CNNs also have limitations: they require large datasets for effective training, are computationally intensive, exhibit limited interpretability (often regarded as "black-box" models), and perform less effectively on tabular data lacking spatial or sequential structure.

Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a deep learning architecture designed to process sequential or time-dependent data in which the order of information is critical. Unlike Feed-Forward Neural Networks (FNNs), RNNs incorporate feedback connections that allow information from previous time steps to influence the current output, thereby maintaining a form of memory across the sequence.

A typical RNN consists of an input layer that receives sequential data, followed by one or more hidden layers containing feedback loops that retain past information. Each neuron in the hidden layer processes the current input along with the hidden state from the previous time step, enabling temporal context to be preserved. The output layer then generates predictions based on the accumulated information.

However, standard RNNs struggle to retain long-term dependencies due to issues such as vanishing or exploding gradients. To address these limitations, several advanced architectures have been developed, including Long Short-Term Memory (LSTM) networks, which capture long-range dependencies and mitigate the vanishing gradient problem; Gated Recurrent Units (GRU), a simplified and computationally efficient variant of LSTMs; and Bidirectional RNNs (Bi-RNN), which process sequences in both forward and backward directions to capture contextual information from past and future data points [113].

RNNs are particularly effective in domains where sequential patterns and temporal context are important, such as Natural Language Processing (NLP), finance (e.g., stock price forecasting), healthcare (e.g., patient monitoring), cybersecurity (e.g., log sequence analysis), IoT systems (e.g., sensor data processing), and industrial automation (e.g., predictive maintenance).

Key advantages of RNNs include their temporal awareness, ability to handle variable-length input sequences, effectiveness in time-series prediction, and capability to detect gradual behavioral changes. Nonetheless, they present several challenges: training is computationally intensive and relatively slow, parallelization is limited due to their sequential nature, and large amounts of temporal data are typically required to achieve optimal performance.

Transformer Neural Networks

The Transformer Neural Network (TNN) is a deep learning architecture specifically designed to process sequential data without relying on recurrence. Instead of the iterative processing used in Recurrent Neural Networks (RNNs), Transformers utilize a mechanism known as self-attention, which enables the model to simultaneously consider all elements within a sequence and capture global dependencies among them. This architecture was first introduced in the seminal 2017 paper "Attention Is All You Need" [114] and has since revolutionized sequence modeling and representation learning across diverse domains.

A standard Transformer consists of two primary components: an encoder and a decoder. The encoder processes the input sequence and generates contextualized embeddings by employing self-attention to model relationships between all positions in the input. The decoder, in turn, generates

the output sequence by attending to relevant portions of the encoder's output through a combination of self-attention and encoder–decoder attention mechanisms. Each encoder and decoder layer typically comprises a multi-head attention mechanism that captures multiple types of relationships in parallel, a feedforward network that processes intermediate representations, and residual connections coupled with layer normalization to enhance training stability and model performance.

Transformers have been widely adopted across numerous deep learning domains. In Natural Language Processing (NLP), they form the foundation of state-of-the-art systems for machine translation, text summarization, and conversational agents (chat bots). In computer vision, Vision Transformers (ViTs) are used for image classification and object detection. In speech processing, they enable speech recognition and voice synthesis. Furthermore, Transformers have gained traction in cybersecurity for log anomaly detection and threat behavior modeling, in IoT and edge systems for sequence and device behavior modeling, and in time-series forecasting for applications such as energy consumption prediction and network traffic estimation.

The key advantages of Transformers include parallel processing, which allows simultaneous handling of all input elements, resulting in significantly faster training compared to RNNs, along with an exceptional ability to capture long-range dependencies through self-attention. They are also highly scalable, perform effectively on large datasets, and exhibit versatility across various data modalities including text, images, audio signals, and graphs. Moreover, Transformers consistently outperform traditional RNN- or LSTM-based models on most sequence-based tasks.

However, the architecture also presents several limitations. Transformers entail high computational and memory costs, often requiring substantial hardware resources for both training and inference. They typically demand large-scale datasets for effective learning and possess a complex architecture compared to simpler models such as Feed-Forward Neural Networks (FNNs) or Convolutional Neural Networks (CNNs). Consequently, their resource-intensive nature limits deployment in constrained or edge environments where computational capacity and energy availability are restricted.

Chapter 3

Literature Review

This chapter provides a literature review on device fingerprinting and limitations in the existing dataset for device identification. Although there is extensive literature on device identification, we highlight a selection of supervised machine learning works along with their limitations, as discussed below.

3.1 Supervised Machine Learning Techniques

Wang et al. [115] proposed IoT-Portrait that automatically identifies IoT devices by relying on a transformer neural network to learn the corresponding device's traffic behavior. Their study utilized class incremental learning while training the new model to preserve the old class features, mitigating catastrophic forgetting. They rely on both active and passive device information extraction- their approach becomes intrusive, hence unsuitable for real-time deployments. Their approach relies on a stable network connection and has only been evaluated against 15 devices. Also, the network fluctuations could potentially affect the device identification accuracy of the model.

IoTTFID, proposed by Hao et al. [116], uses a transformer framework to identify devices after feature extraction. The model performs well on the UNSW dataset, achieving an accuracy of 98.09%, and on the YourThings dataset, achieving an accuracy of 98.29%. However, when the datasets are merged to incrementally add devices, the overall accuracy of the model reduces to 80.40%. In addition to the reduced accuracy of the model, it also begins to consume excessive

resources, which limits the model's applicability.

In another study conducted by Ortiz et al. [117], a stacked autoencoder-based approach was proposed to learn traffic features from corresponding devices, termed DeviceMien. They utilize TCP flows in their study with the objective of identifying devices with limited observations. Although their study worked well in distinguishing between IoT and non-IoT devices. However, the limited observations produced less precise results in device identification in certain cases.

An intrusive method for extracting device annotations through application-layer response was proposed by Feng et al. [118], terming it Acquisitional Rule-based Engine (ARE). Although the intrusive approach, also called active scanning-based IoT device fingerprinting, does not require a dataset. However, the intrusive approach introduces overhead on limited computational devices, degrading their performance even further. Since intrusive approaches involve unnecessary interactions with devices on a network, they are generally discouraged due to security concerns.

IoT Sentinel by Miettinen et al. [2] proposed device identification and securing the vulnerable devices' communication utilizing Software-defined Networking (SDN). They observed the communication of the devices and extracted 23 features to fingerprint the corresponding IoT devices. For each device, they trained a classifier to provide a binary decision for the fingerprint matching. In cases of several classifiers accepting the new IoT device, they utilized an edit distance-based metric to make the decision. They achieved an accuracy of 81.5% with the considered 27 IoT devices.

IoTDevID, a study conducted by Kostas et al. [119], proposed the use of a Decision Tree to improve device identification performance by balancing the trade-off between inference time and accuracy. They considered six machine learning algorithms in their study: RF, kNN, GB, DT, NB, and SVM. An overall aggregated F1 score of 81% was achieved on the Aalto dataset, while a score of 94% was observed on the UNSW dataset for device identification. Decision Tree poses various challenges in a limited dataset that was not explored in their research, including underfitting, overfitting, and strong correlation between non-correlated features [120].

Bezawada et al. [121] conducted a study to identify both the devices and their categories. In their study, they included fourteen devices in seven categories. Their approach relied on extracting behavioral fingerprinting of devices based on the traffic sessions and the corresponding statistical analysis. They used several classifiers, including k-nearest-neighbors, Gradient boosting, Decision

Tree, and majority voting. However, they observed consistent, promising results from Gradient boosting. Their limited feature extraction and their dependency on consecutive packets limit applicability for real-life deployments. Furthermore, their limited feature extraction might not be practical for fingerprint devices in a large dataset.

The device identification and anomaly detection within the Internet of Things environment was studied by Rabbani et al. [122]. They extracted over 130 features from packet captures, modified from the CICIoT2023 dataset, for both device identification and anomaly detection. They studied RF, XGBoost, LR, MLP, and DT classifiers, finding that the DT-based classifier outperformed the rest. Since the CICIoT2023 dataset is highly imbalanced, they did not consider the limitations offered by the DT classifier [120]. However, they limited the samples from all the devices to 100k.

Kostas et al. [123] addressed the limited generalizability of IoT device identification models, which have not been tested in heterogeneous networks [124–127]. The authors proposed a two-stage framework that employs a genetic algorithm with external feedback for feature and model selection, applied across multiple network environments, followed by a comprehensive evaluation phase. However, the study focused only on active traffic, included limited device diversity, and did not verify if identically labeled devices across datasets shared the same hardware or firmware, factors that can affect the validity of generalization.

Device identification using the hash of the IoT device traffic has been proposed by Charyyev et al. [128] in their study termed as Locality-Sensitive IoT Fingerprinting (LSIF) system. This approach eliminates the need for explicit feature extraction by utilizing the Nilsimsa hash of network traffic to identify IoT devices. The system was evaluated in a controlled laboratory environment using 22 IoT devices across various categories, including home appliances, cameras, doorbells, smart plugs, and lighting, achieving a precision of 93% and a recall of 90%. However, the reliance on continuous hash computation imposes limitations on its applicability in resource-constrained environments, such as low-power or embedded IoT devices.

Yin et al. [129] proposed a deep learning-based approach for IoT device identification that involves converting network traffic into images and subsequently extracting features using neural

networks. This method eliminates the need for manual feature engineering and facilitates the identification of both known and unknown devices. However, the approach relies heavily on trial-and-error for model parameter tuning, underscoring the need for systematic data analysis tools to support threshold selection and network architecture optimization.

In a related study [130], the authors investigated the problem of determining whether a device belongs to the IoT category, followed by its device-type classification. They employed multiple deep learning approaches for device type identification, including Deep Feed-Forward Neural Networks (DFNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformer Neural Networks (TNN). The study utilized the IoT-Sentinel dataset; however, the inherent class imbalance in the dataset significantly degraded model performance. Although the authors implemented certain mitigation strategies, the reported accuracy of their DFNN-based approach was approximately 92%. It is important to note that the model incorporated only a 20% neuron dropout rate, which likely caused premature convergence and potential overfitting, especially given the extensive training regime of 100 to 150 epochs. Furthermore, the proposed methodologies did not include any mechanism for integrating new or previously unseen devices, an essential consideration in IoT environments characterized by rapid and continual device proliferation.

3.2 Limitations in existing datasets

Several datasets have been proposed over time for IoT device identification; however, most are private, and the publicly available ones present significant challenges. Either these datasets are deployed in a restricted environment with pre-defined device communication, or deployed in large networks that depend on a lot of networking devices that do not exist in the real world.

The UNSW dataset [124] consists of 28 different commercial IoT devices. These commercial IoT devices range from health monitoring devices, cameras, appliances, lights, plugs, and motion sensors. The dataset was collected in 2016 for a time period of six months. Other than being outdated, the dataset also contained limited samples for a few devices, as reported in the study [115]. Wang et al. [115] neglected six of the devices from the dataset reporting the samples to be too limited for their study. This highlights the nature of an imbalanced UNSW dataset that was built without

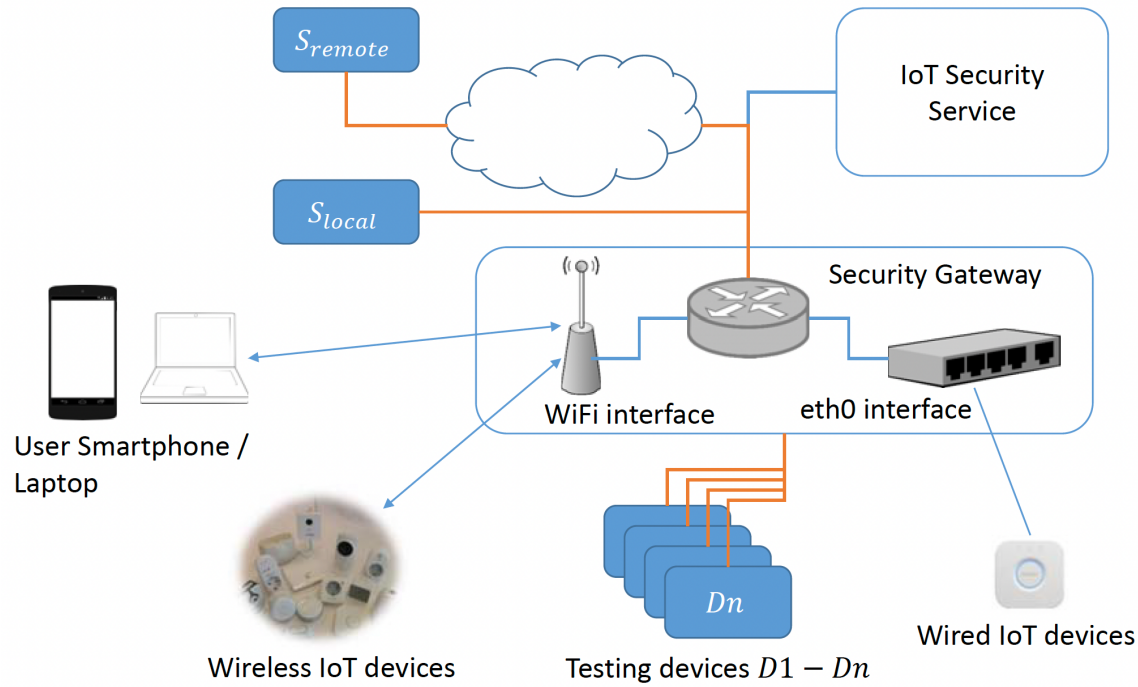


Figure 3.1: IoT Sentinel testbed [2].

consideration.

As mentioned earlier, some device identification datasets are private; one such dataset was reported in the DeviceMien study [117], which collected traffic from 72 IoT devices in 2017. They only mentioned the forty-four devices from the private lab while not sharing the dataset with the public in their study. The dataset is considered outdated in addition to being private.

The IoT Sentinel project introduced a publicly available dataset comprising thirty-one IoT devices, of which twenty-seven were unique [2]. The authors detailed their methodology for dataset generation, which involved capturing device fingerprints twenty times per device to ensure adequate sample representation. Although the dataset was released publicly in 2017, the data collection process was conducted in 2016. The IoT devices included in the dataset span multiple categories such as security cameras, health monitoring systems, lighting, home automation, and general household appliances.

The corresponding testbed setup for IoT Sentinel is illustrated in Figure 3.1. The dataset was originally designed to facilitate research on device-type identification and security policy enforcement. Device traffic was captured under controlled conditions and repeated multiple times

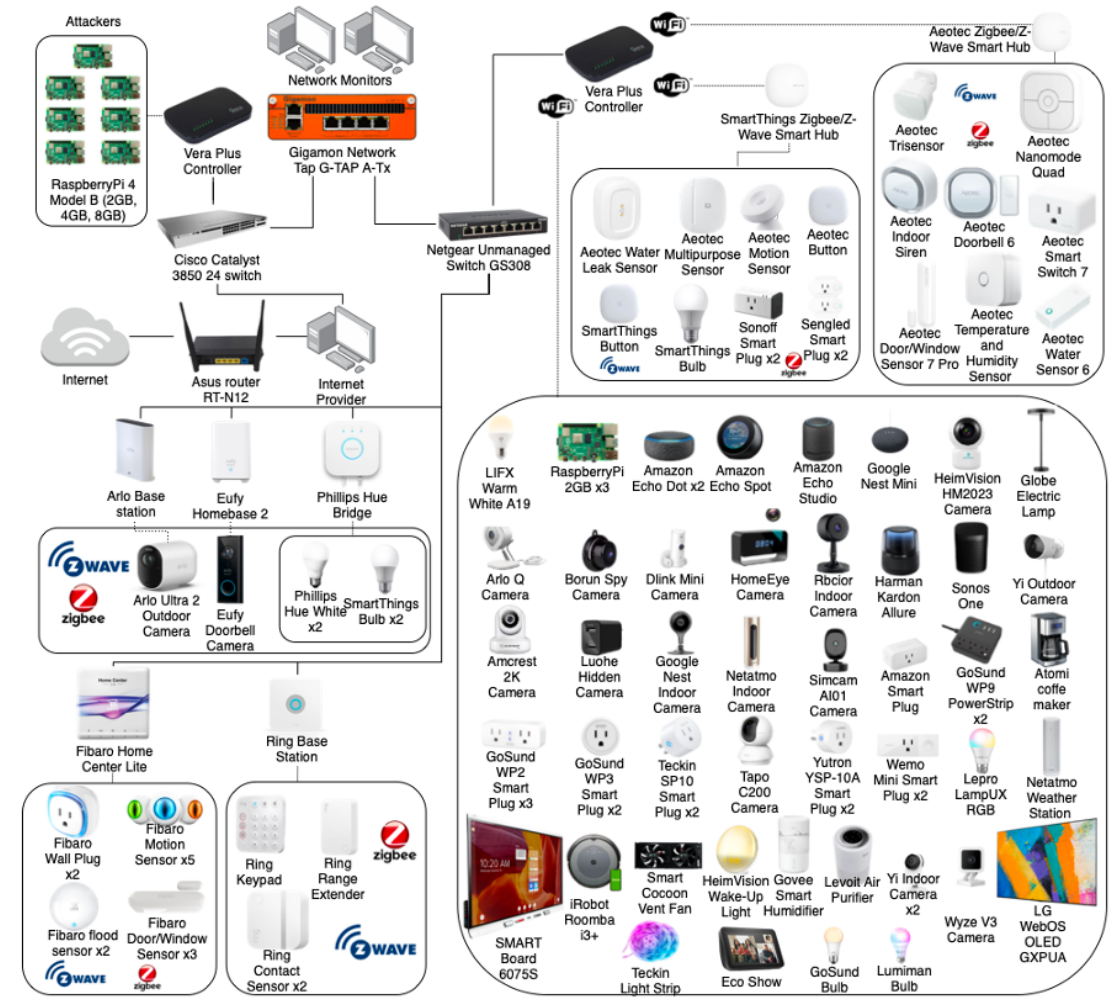


Figure 3.2: CICIoT2023 testbed [3].

to achieve sufficient sample diversity. Despite its popularity and frequent adoption in IoT device identification research, the dataset has several limitations. It is now considered outdated due to its age, and its structure is more suitable for studies focused on device setup characteristics rather than dynamic, behavior-based device identification.

Bezwada et al. [121] collected data from fourteen IoT devices, with sample counts ranging from 1,000 to 10,000 per device. Categories of these IoT devices include colored lights, cameras, socket hubs, music players, and outlets. They considered a limited number of IoT devices in their study. Their work was also published in 2018, highlighting that the dataset was outdated. Given the range of samples, it is evident that the dataset was unbalanced.

The CICIoT2023 dataset [3] was originally developed to facilitate anomaly detection within

Internet of Things (IoT) environments. Subsequently, it was extended and restructured into the CICIoT-DIAD 2024 dataset [122], which supports both device identification and anomaly detection tasks. The corresponding testbed configuration, used for network expansion, represents one of the largest and most comprehensive IoT network environments introduced in the literature, as illustrated in Figure 3.2.

The network architecture incorporated a diverse range of IoT and networking devices, reportedly encompassing approximately one hundred commercial IoT devices along with simulated attacker nodes. Data collection was conducted over a continuous 16-hour period, during which the IoT devices generated randomized network traffic. Analysis of the dataset revealed a significant imbalance in traffic distribution, one device alone accounted for more than 45% of the total network traffic, while only five devices collectively contributed over 70% of the total traffic, as summarized in Table 3.1. Although the expanded testbed enhanced the dataset's scope and heterogeneity, it also introduced substantial communication overheads among the devices. Furthermore, the dataset exhibits pronounced class imbalance, as numerous IoT devices within the network produced minimal or negligible traffic throughout the data collection period.

Table 3.1: Most active IoT devices in CICIoT2023 benign traffic.

Sr. No.	**Device Name**	**Traffic Contribution (%)**
1	Arlo Q Indoor Camera	45.4
2	Nest Indoor Camera	12.4
3	Amazon Echo Show	5.6
4	Amazon Alexa Echo Dot	5.3
5	Amazon Alexa Echo Studio	1.9
6	Google Nest Mini Speaker	1.6
7	LG Smart TV	1.4
8	Harman Kardon	1.3
9	Home Eye Camera	1.2
10	Wyze Camera	1.2

Continued on next page

Table 3.1 (continued)

Sr. No.	Device Name	Traffic Contribution (%)
11	Netatmo Camera	1.1
12	SmartThings Hub	0.9
13	AeoTec Smart Home Hub	0.9
14	HeimVision Smart WiFi Camera	0.6
15	Racior Camera	0.6
16	Yi Indoor Camera	0.6
17	Rufy Home Base	0.6
18	Netatmo Weather Station	0.5
19	TP-Link Tapo Camera	0.5
20	AMCREST WiFi Camera	0.4

The pronounced imbalance in the CICIoT2023 and CICIoT-DIAD 2024 datasets presents notable challenges for developing reliable and generalizable machine learning models. The dominance of traffic from a small subset of devices skews the learned representations, leading models to overfit toward the high-traffic devices while failing to adequately capture the behavioral characteristics of underrepresented ones. Such disproportionate data distribution undermines the robustness of device identification and anomaly detection tasks, as the models tend to exhibit biased performance and poor generalization to unseen devices or real-world heterogeneous network environments. Moreover, the presence of devices that generated little to no traffic further complicates the learning process by introducing sparsity, which limits the ability to infer meaningful behavioral patterns. Addressing these limitations requires rigorous data preprocessing, balancing strategies, or synthetic data augmentation to ensure fair representation of all devices and to enhance model stability across diverse IoT ecosystems.

Most datasets reported in the literature have been generated within controlled laboratory environments employing complex network configurations. While such setups allow precise control over experimental conditions, they also introduce artificial latency due to the intricate interconnections

between multiple networking devices. Replicating these complex infrastructures across research institutions remains a major challenge, thereby limiting the scalability and collaborative development of IoT datasets. Laboratory-generated datasets are inherently constrained by the limited number of IoT devices available for experimentation, in stark contrast to the vast and diverse deployments observed in real-world smart home environments. Unlike these controlled testbeds, smart home networks encompass heterogeneous IoT devices operating across distinct sub-networks and configurations, offering a far more dynamic and realistic data distribution.

Another recurring issue associated with laboratory generated datasets is the pronounced class imbalance, as summarized in Table 3.1. Our analysis revealed that certain IoT devices exhibited minimal or even no traffic generation during the 16-hour data collection period, further amplifying dataset imbalance and limiting model generalizability.

Recognizing these limitations, this research emphasizes the need for scalable and diverse data collection across heterogeneous network environments. Detecting and profiling similar IoT devices across multiple real-world networks can enable the development of more robust and transferable machine learning models. To facilitate this objective, we introduce two traffic-capturing tools, ScanIoT and DroidScour, designed to support scalable, distributed, and realistic IoT traffic collection for improved device identification and behavioral analysis.

Chapter 4

Traffic Capturing Frameworks

This chapter presents two laboratory-developed tools—ScanIoT and DroidScour—designed for the collection and annotation of IoT device traffic. Both tools feature user-friendly graphical interfaces that facilitate ease of use and enable systematic traffic capture from IoT devices. ScanIoT [22, 23, 131] and DroidScour [24, 25] are publicly available and have been specifically developed to gather network traffic from IoT devices operating within realistic smart home environments. These environments incorporate natural human–device interactions to ensure the authenticity and representativeness of the collected data. The following sections provide a detailed overview of the design, functionality, and implementation of both tools.

4.1 ScanIoT

ScanIoT is a network traffic collection and annotation framework designed to facilitate the systematic capture of IoT device traffic. It provides an intuitive user interface accessible via both web and mobile applications, with both platforms engineered to operate coherently for seamless user interaction and data synchronization. The subsequent sections detail the hardware requirements, system architecture, and functional components of ScanIoT.

4.1.1 Hardware Components

The latest version of the Raspberry Pi 5 [132], equipped with 8 GB of RAM, was utilized as the backend platform for developing ScanIoT. In addition to the Raspberry Pi kit, an optional external dual-band USB adapter was integrated to enhance mobility and facilitate wireless connectivity. Alternatively, the built-in ethernet port of the Raspberry Pi can be employed for wired communication. The standard Raspberry Pi kit included an SD card, card reader, protective casing, and cooling fan. The external SD card served as the primary storage medium for installing the operating system and executing the necessary software components.

The Raspberry Pi was chosen as the foundational hardware platform owing to its versatility, cost-effectiveness, and high degree of customization. Within the ScanIoT framework, the Raspberry Pi can be configured either to replace the existing access point or to operate as an intermediary node positioned between the Internet Service Provider (ISP) access point and the connected IoT devices. This configuration ensures that all network traffic is routed through the device for comprehensive traffic annotation and collection. Moreover, the proposed architecture is inherently scalable, as multiple Raspberry Pi units can be interconnected to extend Wi-Fi coverage and support larger, more distributed deployment environments.

The Raspberry Pi was chosen as the foundational hardware platform owing to its versatility, cost-effectiveness, and high degree of customization. Within the ScanIoT framework, the Raspberry Pi can be configured either to replace the existing access point or to operate as an intermediary node positioned between the Internet Service Provider (ISP) access point and the connected IoT devices. This configuration ensures that all network traffic is routed through the device for comprehensive traffic annotation and collection. Moreover, the proposed architecture is inherently scalable, as multiple Raspberry Pi units can be interconnected to extend Wi-Fi coverage and support larger, more distributed deployment environments.

The Raspberry Pi was configured to function as a gateway, enabling all network traffic to traverse through it for collection and annotation. This configuration can be applied to both wireless and wired (LAN) network interfaces on the Raspberry Pi, providing flexibility for diverse deployment environments. The following section elaborates on the system design of the ScanIoT framework.

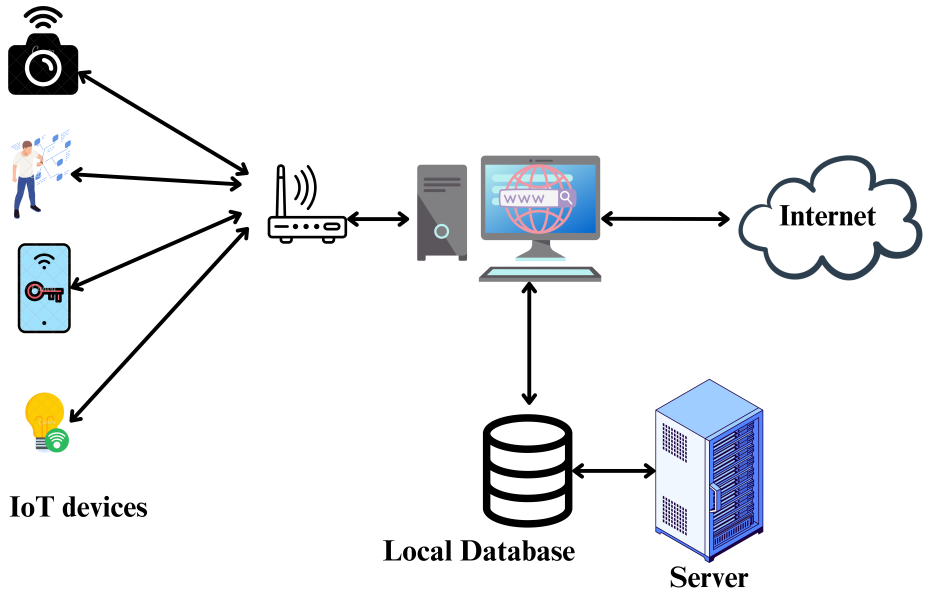


Figure 4.1: ScanIoT, an access point in a Smart Home.

4.1.2 System Design

Figure 4.1 depicts the architectural configuration of the ScanIoT framework operating as an access point within a smart home network environment. In this architecture, Wi-Fi serves as the medium for wireless communication between the other access point and the interconnected IoT devices, whereas the core computational and control functionalities are hosted on a Raspberry Pi platform. The ScanIoT system design section elaborates on the underlying web and mobile application technologies, followed by a comprehensive description of the functional capabilities offered by the system.

Web Interface

The ScanIoT web application is developed using Flask, a lightweight yet extensible Python-based microframework well suited for the rapid prototyping of web systems and RESTful APIs. Flask provides the underlying routing architecture, manages Hypertext Transfer Protocol (HTTP) requests and responses, and dynamically renders web templates, thereby ensuring efficient client

server communication. The system employs PostgreSQL, an open-source relational database management system (RDBMS), to ensure reliable data persistence, integrity, and efficient query execution for IoT device records and network activity logs. Network traffic analysis is supported through Scapy, a powerful Python library designed for low-level packet manipulation, which enables real-time packet capture, inspection, and sniffing functionalities within the ScanIoT framework. The graphical user interface (GUI) is implemented using Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS), providing a structured and visually coherent front-end design.

To enhance interactivity and responsiveness, JavaScript is integrated within the web interface to support dynamic user interactions. This includes asynchronous data retrieval from the back-end through API calls, modal generation for contextual information display, and event-driven operations such as controlling and halting network traffic collection for individual IoT devices. The combination of these technologies results in a modular, scalable, and interactive system architecture that facilitates seamless integration between the user interface, application logic, and underlying network monitoring components.

Mobile Application Interface

Given the widespread adoption and accessibility of mobile applications compared to traditional web interfaces, a dedicated mobile application for the ScanIoT framework was developed using the Flutter framework. While the web application is accessible within the local network through a standard web browser, the mobile counterpart offers enhanced usability, portability, and user engagement. The ScanIoT mobile application has been made publicly available via the GitHub repository [131], thereby promoting transparency and reproducibility of the system. Flutter was selected due to its capability to deliver high-performance, cross-platform applications with a responsive and interactive user interface, making it an optimal choice for IoT-oriented mobile systems.

The overall system adheres to a client–server architecture, wherein the mobile application operates as the client, communicating with the back-end services through standardized HTTP-based APIs. The server component, deployed locally on the Raspberry Pi device, processes client requests and returns the corresponding responses in real time. This architecture ensures efficient communication, low-latency data exchange, and scalability across heterogeneous IoT environments. The

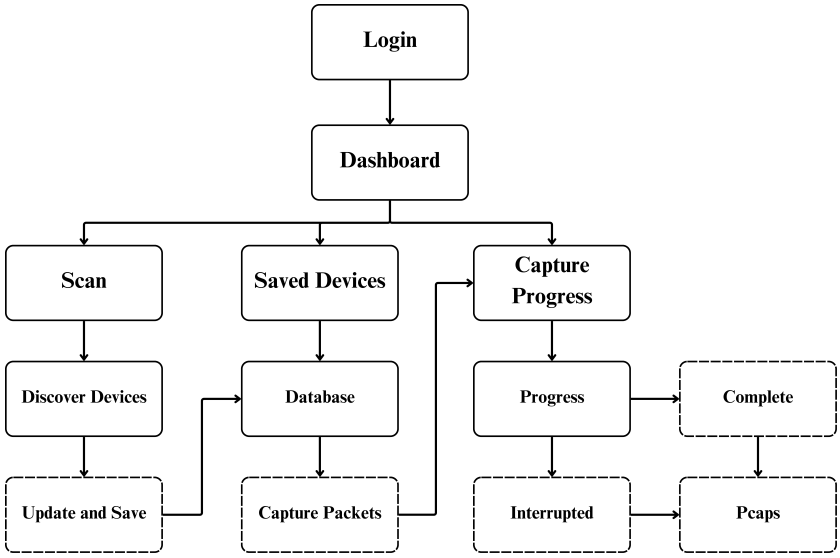


Figure 4.2: Block Diagram representation of ScanIoT.

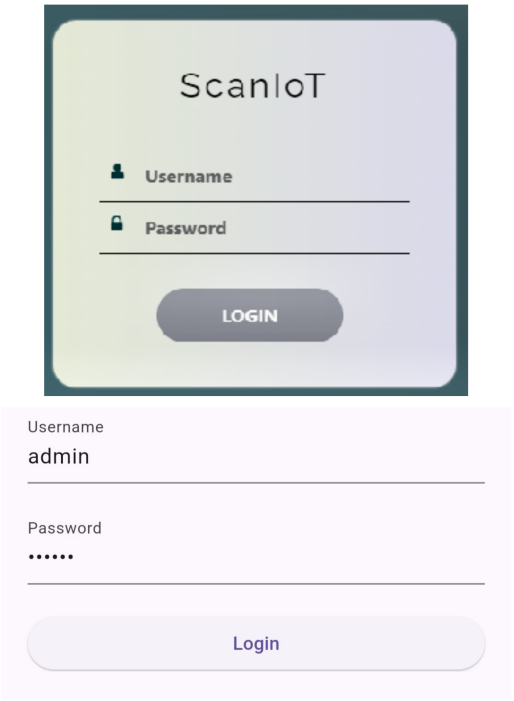
subsequent section provides a detailed explanation of the functionalities offered by both the web and mobile applications within the ScanIoT system.

Functionality

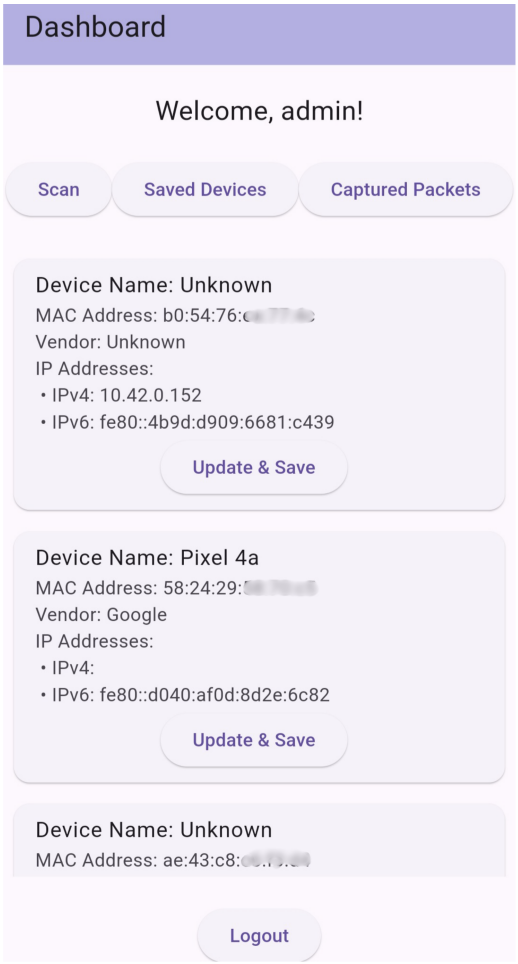
The web and mobile applications operate in a synchronized manner, adhering to an identical functional workflow, as illustrated in Figure 4.2. As previously discussed, the ScanIoT system is hosted locally on a Raspberry Pi platform, which serves as the central access point for user interaction. To ensure secure access, users are required to authenticate through a login interface prior to utilizing the system's functionalities. The corresponding login interfaces are presented in Figure 4.3a, where the upper image represents the web application interface and the lower image depicts the mobile application interface. Upon successful authentication, users are granted access to the system dashboard, which provides an integrated view of the operational features and data analytics components of ScanIoT.

Following successful user authentication, the system dashboard becomes accessible, as illustrated in Figure 4.3b, Figure 4.4, and Figure 4.5. The dashboard serves as the primary interface through which users interact with the core functionalities of the ScanIoT system. It comprises three

main navigation tabs—**Scan**, **Saved Devices**, and **Capture Packets**—each corresponding to a distinct operational module within the framework. These tabs enable users to initiate network scans, manage and review previously identified IoT devices, and monitor or capture network packets, respectively.



(a) Web (top) and Mobile (bottom) Login screen for ScanIoT.



(b) ScanIoT Dashboard.

Figure 4.3: Login and dashboard views of ScanIoT.

The **Scan** navigation tab provides a comprehensive view of all devices currently connected to the ScanIoT system, which functions as a dedicated access point for IoT devices within the network. Device discovery is performed on the back end using the Linux command *ip neigh*, which enumerates all active network endpoints associated with the ScanIoT access point. For each detected device,

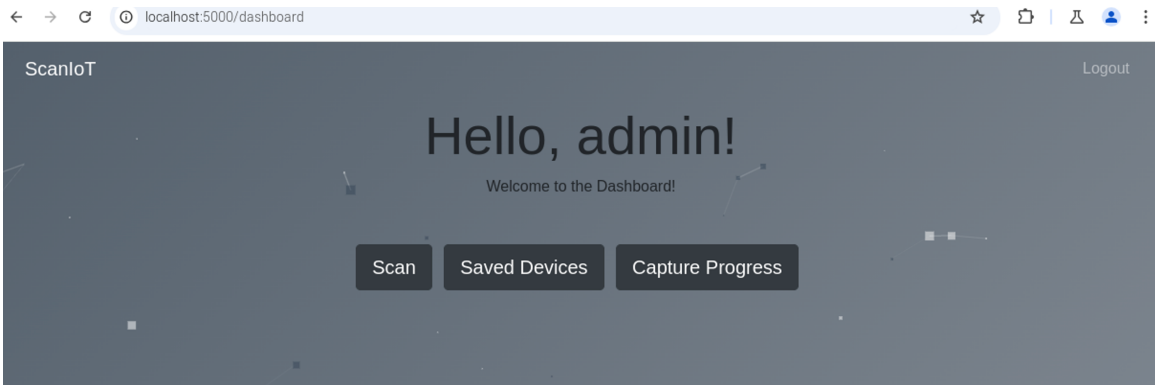


Figure 4.4: ScanIoT web home page.

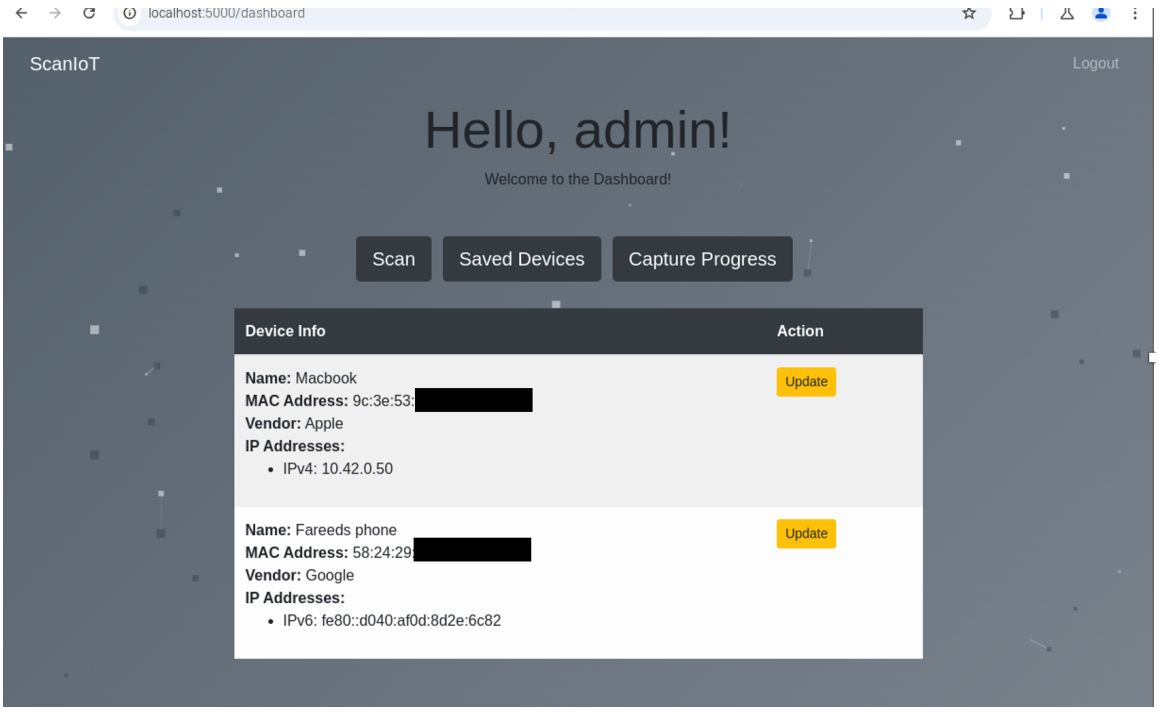


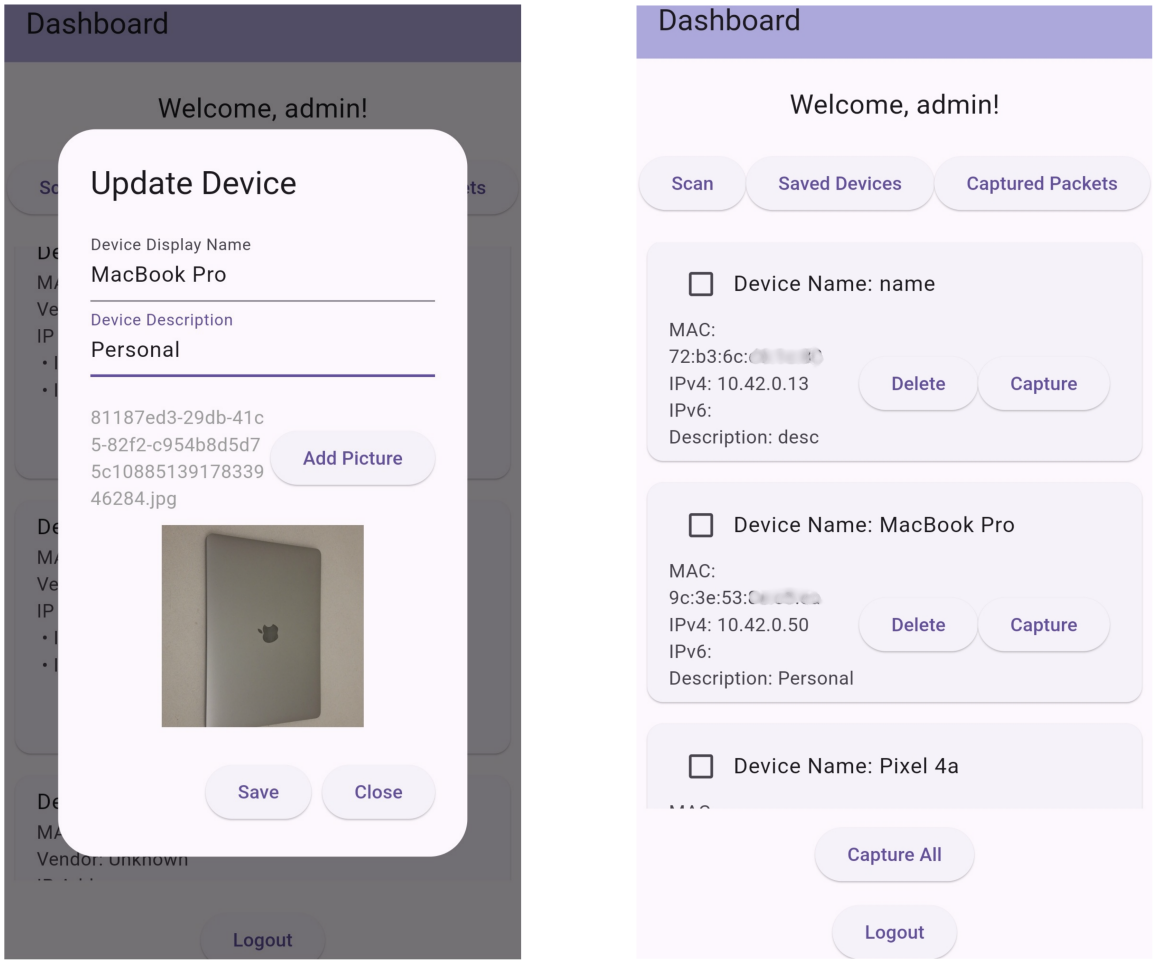
Figure 4.5: ScanIoT scan function on web.

the interface displays the device name (if previously saved), the Media Access Control (MAC) address, vendor information, and both IPv4 and IPv6 addresses. Vendor details are derived from the MAC address, which contains Organizationally Unique Identifiers (OUIs) [133], and are retrieved programmatically using the *manuf* Python library. This approach enables accurate identification and classification of connected devices, facilitating streamlined network monitoring and management.

Within the **Scan** navigation tab, users are provided with the capability to update and save device information through an interactive form. This functionality allows users to assign a name and provide a description for each corresponding device, thereby maintaining an organized and informative device database. To facilitate accessibility for non-technical users, the system also supports the upload of device images, which can subsequently assist technical users in device identification. The image upload interface is illustrated in Figure 4.6a and Figure 4.7. Once the updates are submitted and saved, the revised device information is reflected in subsequent network scans, ensuring that the database remains current and accurately represents all connected devices.

Following the submission of updates through the **Scan** tab, all modifications are stored in the system database and reflected within the **Saved Devices** navigation tab. Upon accessing this tab, the application retrieves the complete set of updated device information from the database and presents it to the user, as illustrated in Figures 4.6b and 4.8. Users are provided with options to manage the device inventory, including the ability to remove devices that are no longer active within the network. Additionally, the tab incorporates a traffic capture feature, enabling users to collect network packets associated with specific devices. Packet capture can be performed on an individual device or concurrently across multiple devices. To initiate a capture session, users must specify a file name for the capture and define the number of packets to be collected, as depicted in Figure 4.9a. This functionality facilitates detailed monitoring and analysis of device-level network activity.

When multiple devices are selected for traffic capture, the ScanIoT system employs multithreading to ensure efficient and uninterrupted collection of network packets. All captured packets are stored locally on the Raspberry Pi, with individual files organized according to the MAC addresses of the corresponding devices. Specifically, the packet capture (PCAP) files are saved within the designated document directory of the Raspberry Pi, and each device maintains a separate PCAP file to prevent data overlap. The traffic capture algorithm filters network traffic based on



(a) Updating devices in ScanIoT.

(b) Save devices navigation tab.

Figure 4.6: Updating devices and fetching them from the database.

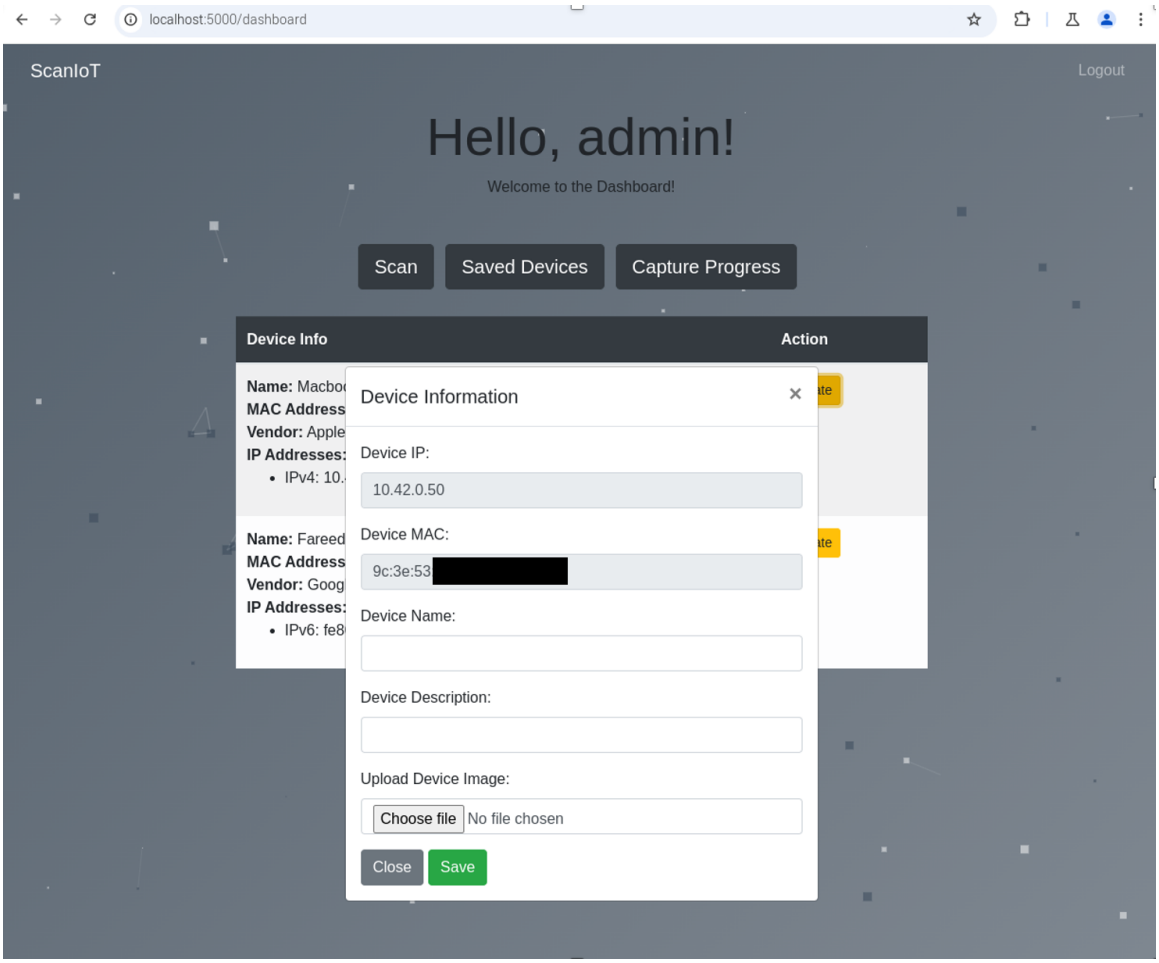


Figure 4.7: ScanIoT update function on web.

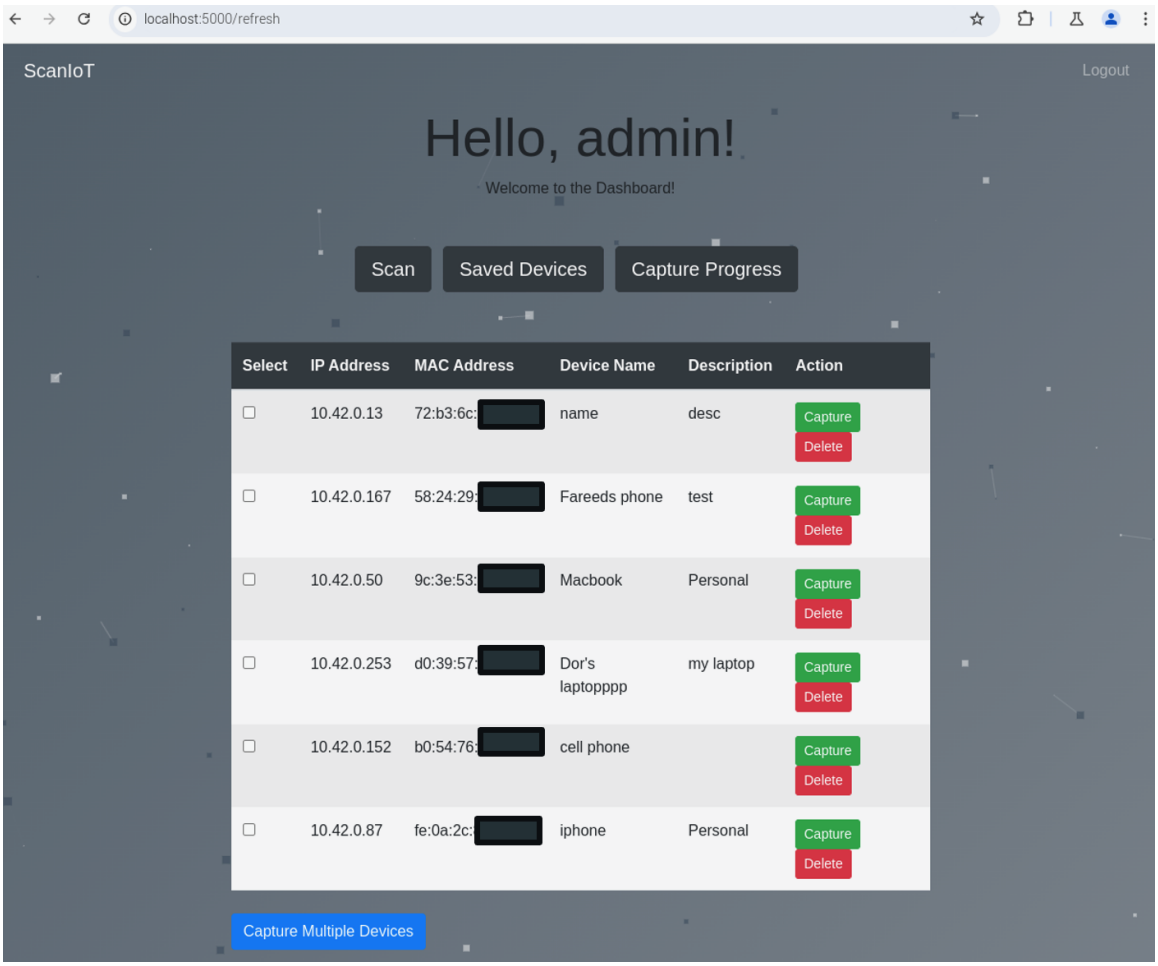
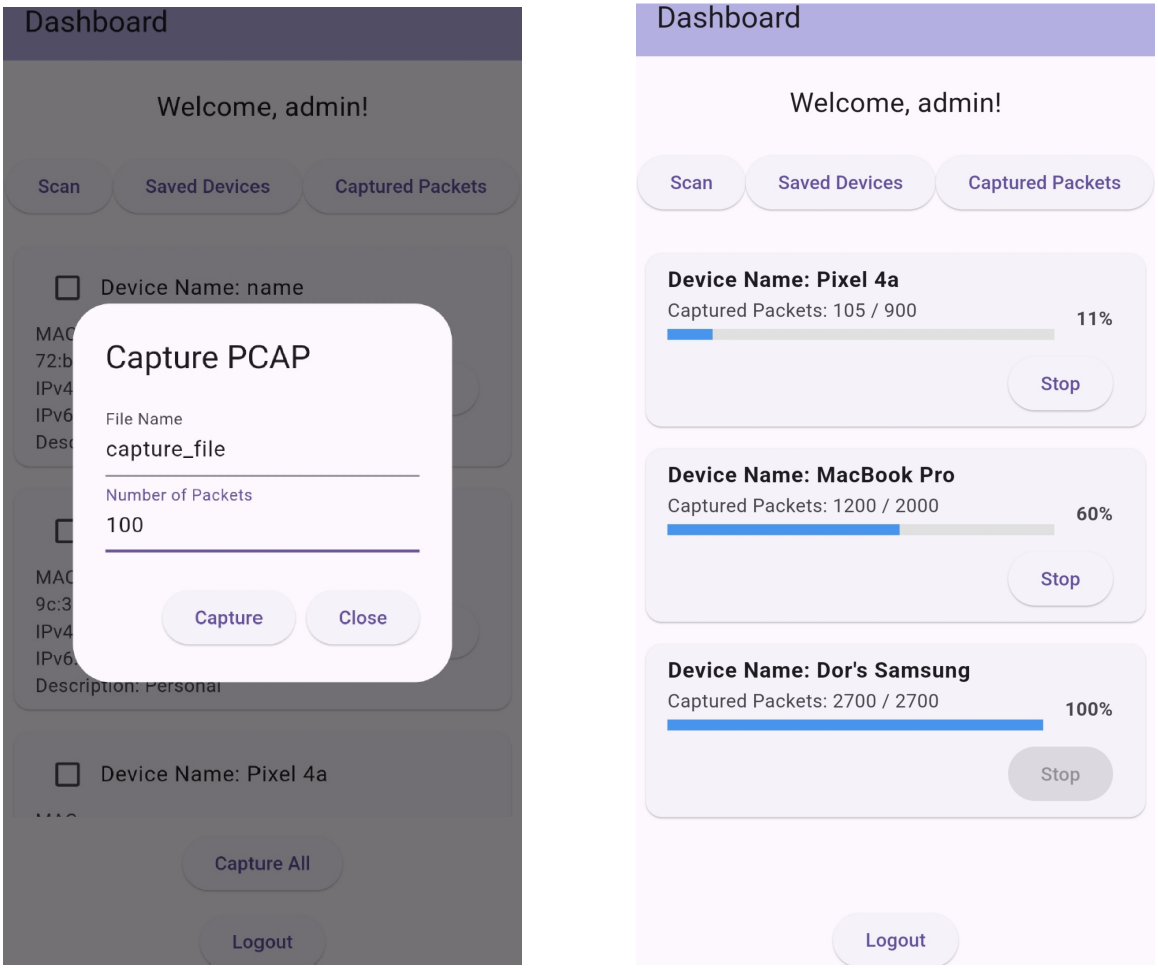


Figure 4.8: ScanIoT update function on web.

device-specific MAC addresses, guaranteeing that each device's data is recorded in a distinct PCAP file.

Upon initiating the traffic capture process, users are directed to the **Capture Packets** navigation tab, where the real-time progress of packet collection for each device is displayed, as illustrated in Figure 4.9b. Users may refresh the interface to monitor the ongoing capture status, and a dedicated stop action allows termination of the capture at any point. When this action is invoked, the system saves the corresponding PCAP files, containing the number of packets collected up to that moment, thereby ensuring no loss of captured data.



(a) Setting name and packet count to capture traffic.

(b) Traffic capture progress for each device.

Figure 4.9: Packet counts and capture progress for ScanIoT.

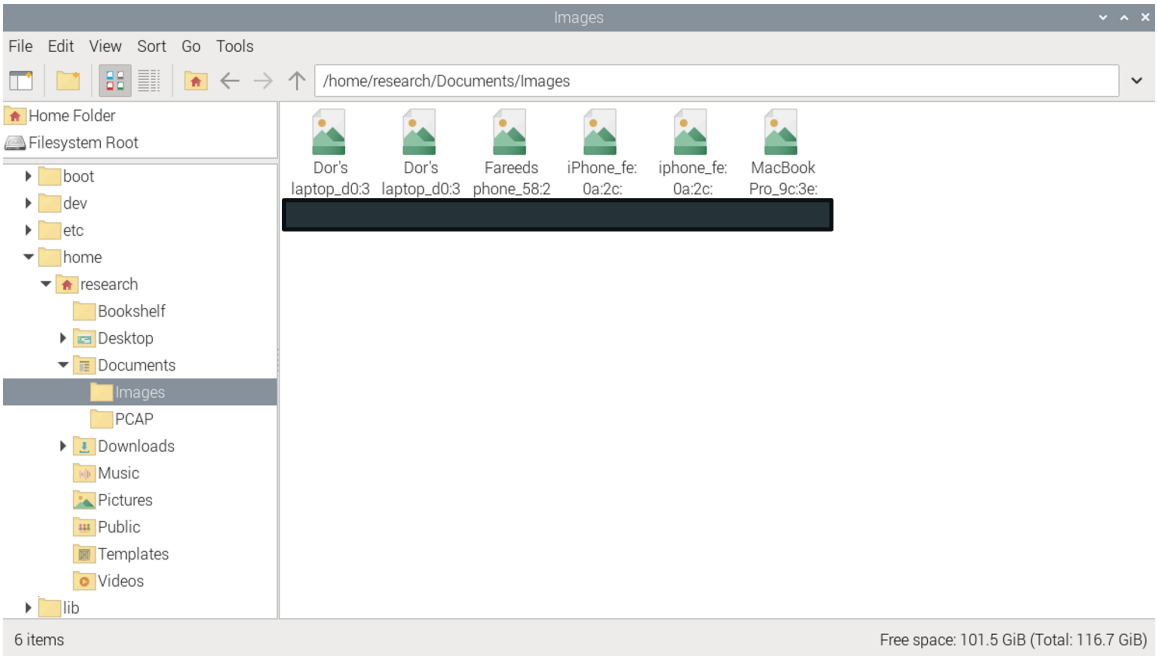


Figure 4.10: ScanIoT sample images.

All device and traffic information collected by ScanIoT is stored locally on the Raspberry Pi, facilitating subsequent retrieval and sharing. Captured PCAP files are organized within the document directory using a hierarchical naming convention, in which the user-provided name is supplemented by the device's MAC address and the capture timestamp. Users are only required to specify the capture name and the number of packets to be collected, while all additional metadata, including MAC addresses and network information, is automatically extracted and recorded by the system. Representative examples of the saved figures and PCAP files on the Raspberry Pi are shown in Figures 4.10 and Figure 4.11, respectively.

The ScanIoT deployment relies on a minimal set of cost-effective hardware components, in contrast to the large-scale and often expensive network infrastructures commonly employed in related research. The datasets generated by ScanIoT are scalable, enabling contributions from independent researchers or smart home users, and are designed to reflect realistic user interactions with IoT devices. By capturing device traffic across diverse network environments, ScanIoT facilitates the creation of high-fidelity datasets suitable for the development and evaluation of robust machine learning algorithms for IoT device identification and classification.

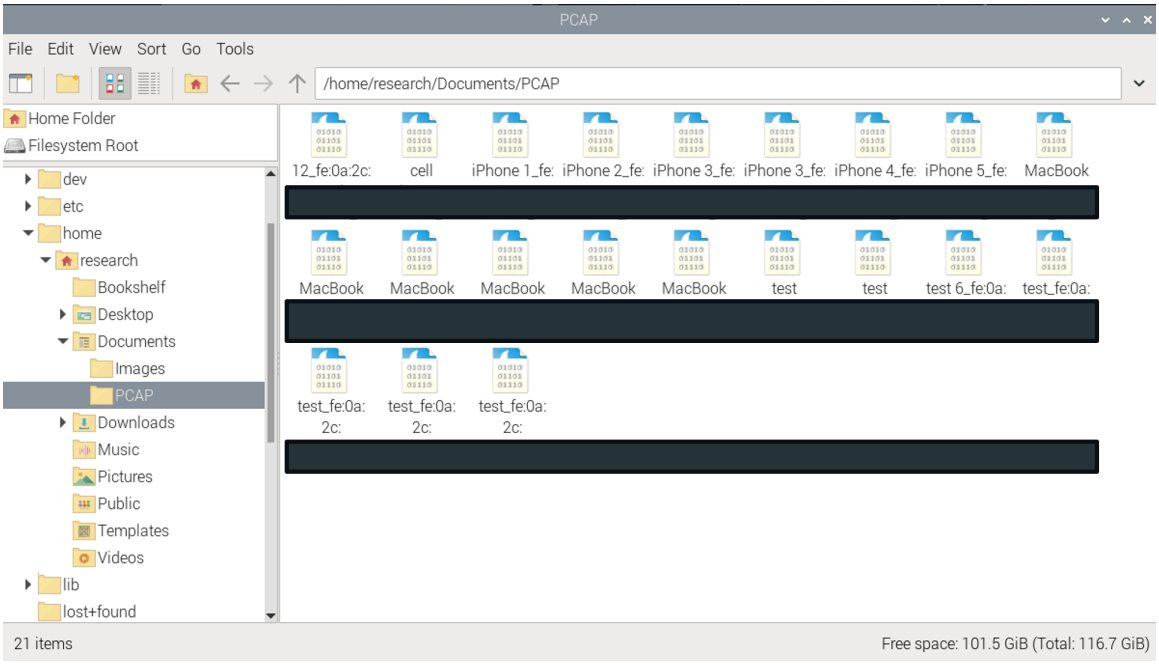


Figure 4.11: ScanIoT PCAP samples.

4.2 DroidScour

DroidScour is an Android-based network traffic–capture application designed to replicate the complete functionality of the ScanIoT framework, including its data–sharing capabilities, while eliminating the need for any additional hardware components such as a Raspberry Pi, external wireless adapter, or auxiliary kits. Although ScanIoT provides comprehensive functionality for IoT traffic collection, DroidScour achieves comparable capabilities using only an Android device, thereby enhancing accessibility and ease of deployment. The captured network traffic can be seamlessly shared through a cloud-based server implemented using Firebase. The subsequent section presents a detailed description of the system architecture and operational workflow of the DroidScour application.

4.2.1 Architecture

The architectural design of the DroidScour application is illustrated in Figure 4.12, which depicts its operation as an inline system positioned between IoT devices and the internet. In this configuration, DroidScour functions as a Wi-Fi access point for the IoT devices whose network

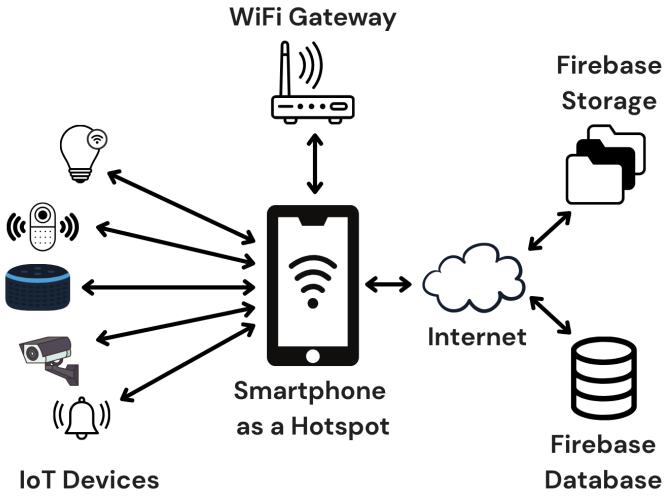


Figure 4.12: Architecture and Working of DroidScour in a network.

traffic is to be captured. Unlike the ScanIoT framework, DroidScour does not rely on any external hardware components; however, its operation requires a rooted Android device [134] to enable low-level packet inspection and control. The application was experimentally validated on a Google Pixel 4a smartphone. The DroidScour application is implemented using the Kotlin programming language [135], which provides robust support for concurrent network operations and modern Android development practices.

Although device rooting is generally discouraged for non-expert users due to potential security and stability concerns, the Android operating system inherently restricts access to network traffic from external devices unless root privileges are granted [136]. As the primary objective of DroidScour is to capture IoT device traffic by functioning as a mobile hotspot, rooting the Google Pixel 4a device was a necessary prerequisite to enable low-level packet interception. The detailed operational flow of the DroidScour application is illustrated in Figure 4.13. In comparison to the ScanIoT framework, DroidScour automates the packet-sharing process by utilizing Firebase for seamless cloud synchronization, thereby eliminating the need for manual file transfer. Furthermore, DroidScour supports time-based capture initiation, allowing users to specify both the capture duration and the desired number of packets, which enhances flexibility and user control during data collection.

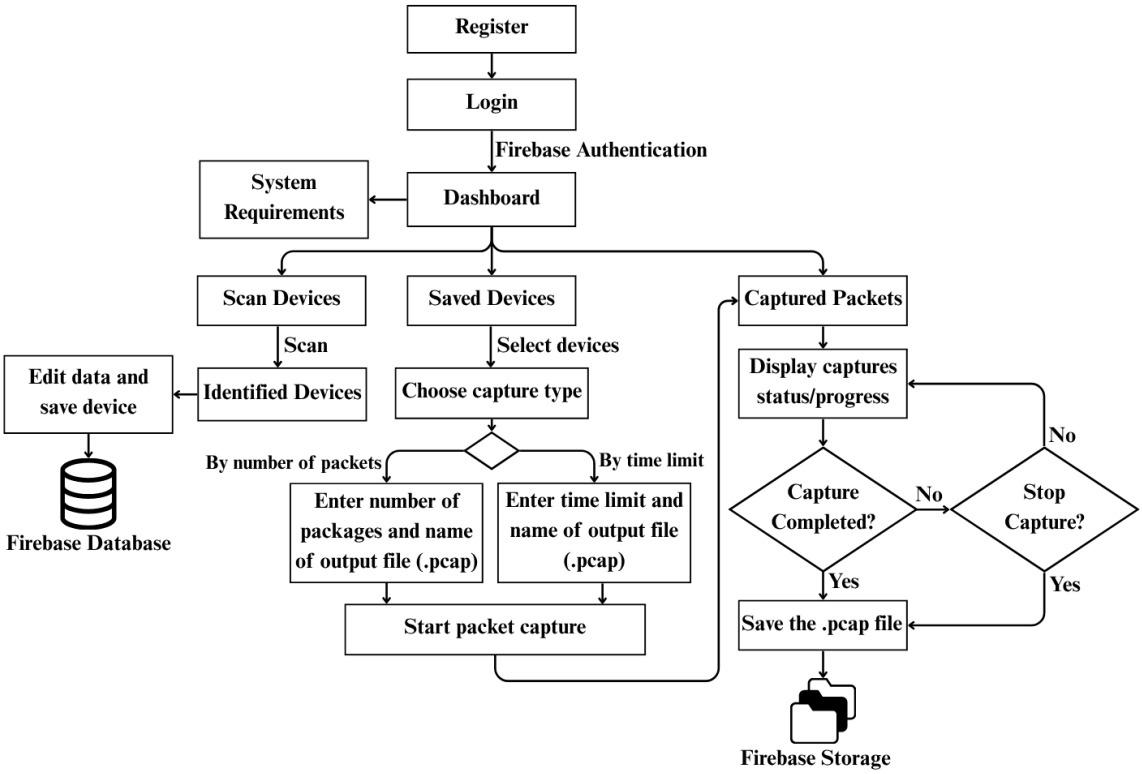
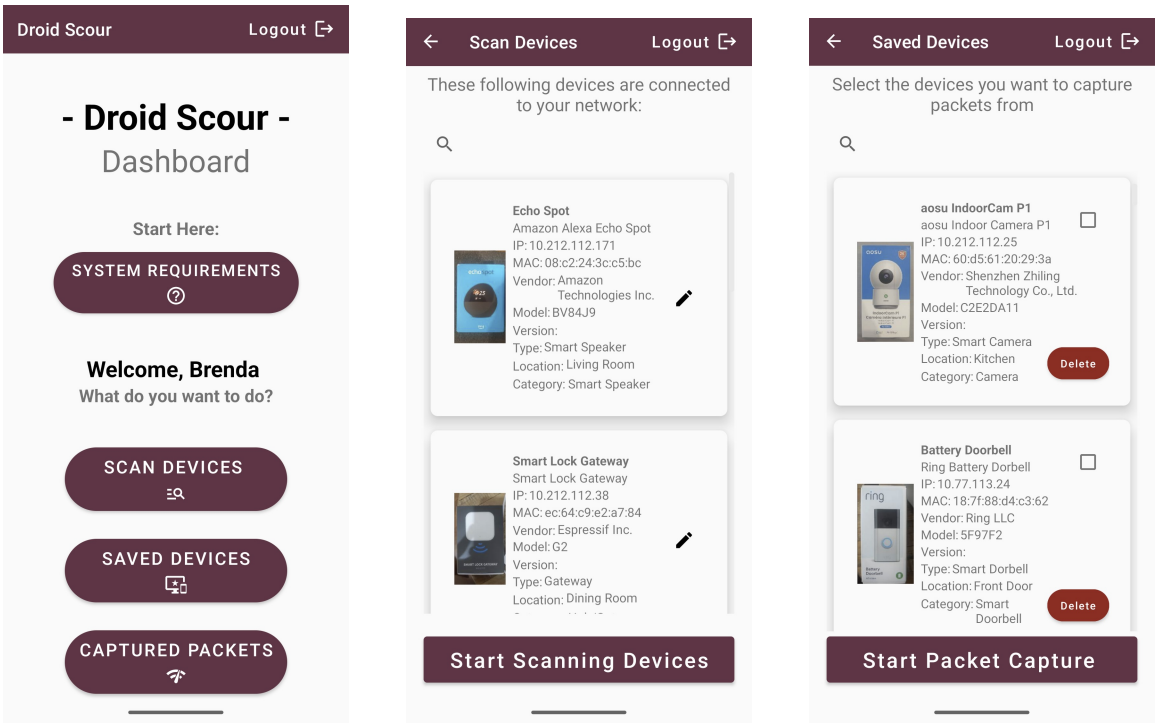


Figure 4.13: DroidScour flow diagram.

4.2.2 System Design

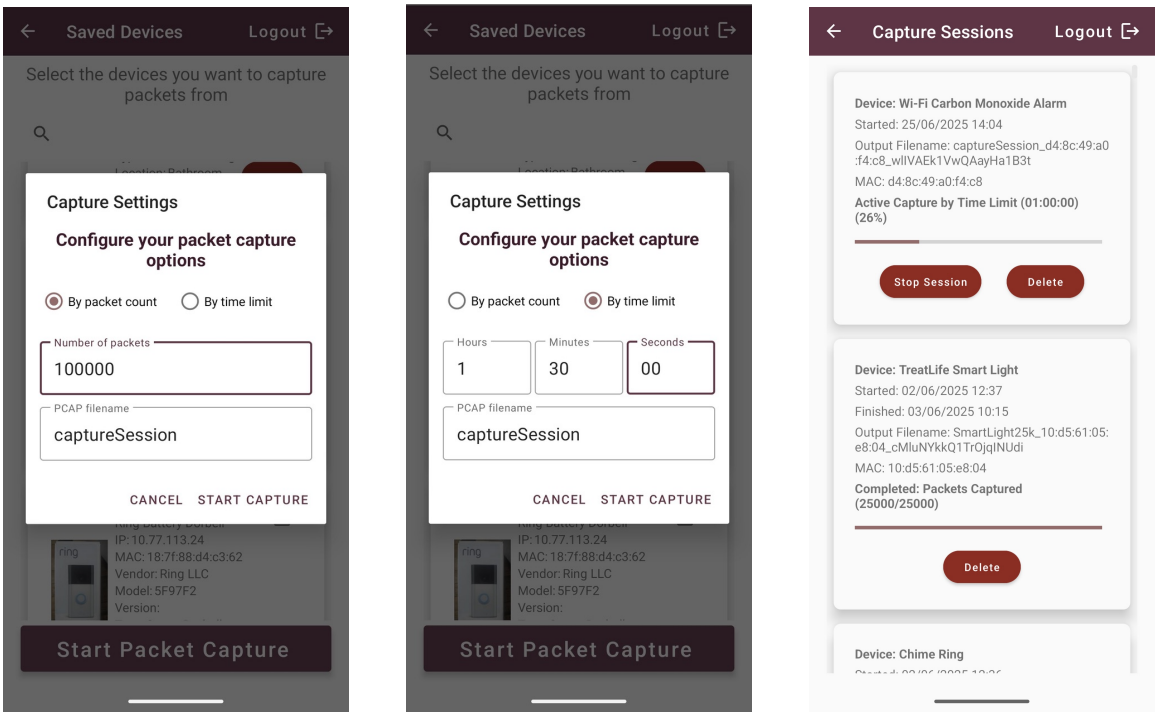
In addition to obtaining root privileges, DroidScour leverages Termux [137] to access a suite of Linux command-line utilities for network discovery and packet capture. Termux exposes standard networking tools on the Android platform, enabling DroidScour to execute reliable, low-level operations that are otherwise unavailable on unmodified devices. Device discovery is performed via the *ip neigh* command within the Termux environment, which interrogates the kernel neighbor table to obtain the IP and MAC addresses of active hosts. This method yields an accurate listing of connected devices without requiring active probing (e.g., periodic ICMP pings), thereby supporting near real-time population of the connected device list with minimal network disturbance. Packet capture is accomplished by invoking *tcpdump* from Termux; captures are performed using device-specific filters (for example, MAC-address filters) to isolate traffic belonging to individual IoT endpoints. Captured traffic is then persisted as PCAP files for subsequent analysis and sharing.



(a) DroidScour dashboard. (b) DroidScour scanned devices. (c) DroidScour saved devices.

Figure 4.14: DroidScour dashboard, scan devices, and saved devices.

Figure 4.14a illustrates the main dashboard interface of the DroidScour application, which enables users to scan for connected IoT devices, view previously saved devices, and monitor the progress of ongoing traffic captures. Upon initiating a device scan, the interface depicted in Figure 4.14b is presented to the user. This interface provides functionality analogous to that of the ScanIoT system, allowing users to edit device names, add descriptive metadata, and capture images of the corresponding IoT devices for visual reference. All user modifications and associated metadata are securely synchronized using Firebase services [138]. Within the DroidScour framework, Firebase facilitates user authentication for account creation and secure access control, while also serving as a cloud-hosted database for the storage of device information, packet capture data, and related metadata. This integration ensures data integrity, accessibility, and secure sharing across devices and users.



(a) DroidScour capture packets by count.

(b) DroidScour capture packets by time.

(c) DroidScour capture progress.

Figure 4.15: DroidScour capture settings and capture progress.

All user edited or newly saved information is synchronized and displayed within the **Saved Devices** navigation tab, as shown in Figure 4.14c. This interface dynamically retrieves the updated

device information from Firebase to ensure consistency across user sessions. From this tab, users are provided with the option to remove devices that are no longer active or initiate traffic capture sessions for one or multiple devices simultaneously. DroidScour supports two distinct modes of packet capture: (i) capture based on a predefined packet count and (ii) capture constrained by a user-specified time duration, as illustrated in Figures 4.15a and 4.15b, respectively. The time-based capture mode is particularly useful for users with limited availability near the target IoT devices, offering flexibility in data collection without compromising dataset completeness. The ongoing capture activity for each device can be monitored under the **Capture Progress** navigation tab. As depicted in Figure 4.15c, users are also provided with controls to manually terminate an active capture session or delete previously collected packet data, ensuring efficient management of storage and network resources.

DroidScour requires an active internet connection to perform user authentication, update device information, and maintain real-time synchronization with Firebase services. Similar to the ScanIoT framework, DroidScour employs a systematic process for naming and managing packet capture (PCAP) files to ensure traceability and consistency across sessions. Upon the completion or manual termination of a capture session, the resulting PCAP file is initially stored locally on the Android device and subsequently uploaded to Firebase for centralized storage and accessibility. In scenarios involving simultaneous captures from multiple devices, DroidScour utilizes MAC address based filtering to isolate network traffic streams, ensuring that a distinct PCAP file is generated and stored for each device, as presented in Figure 4.16. This design supports organized data management and facilitates accurate, device-specific traffic analysis.

4.3 Comparison between ScanIoT and DroidScour

This section delineates the comparative analysis of the ScanIoT and DroidScour frameworks, highlighting their architectural similarities as well as the key distinctions in functionality, deployment requirements, and operational scope.

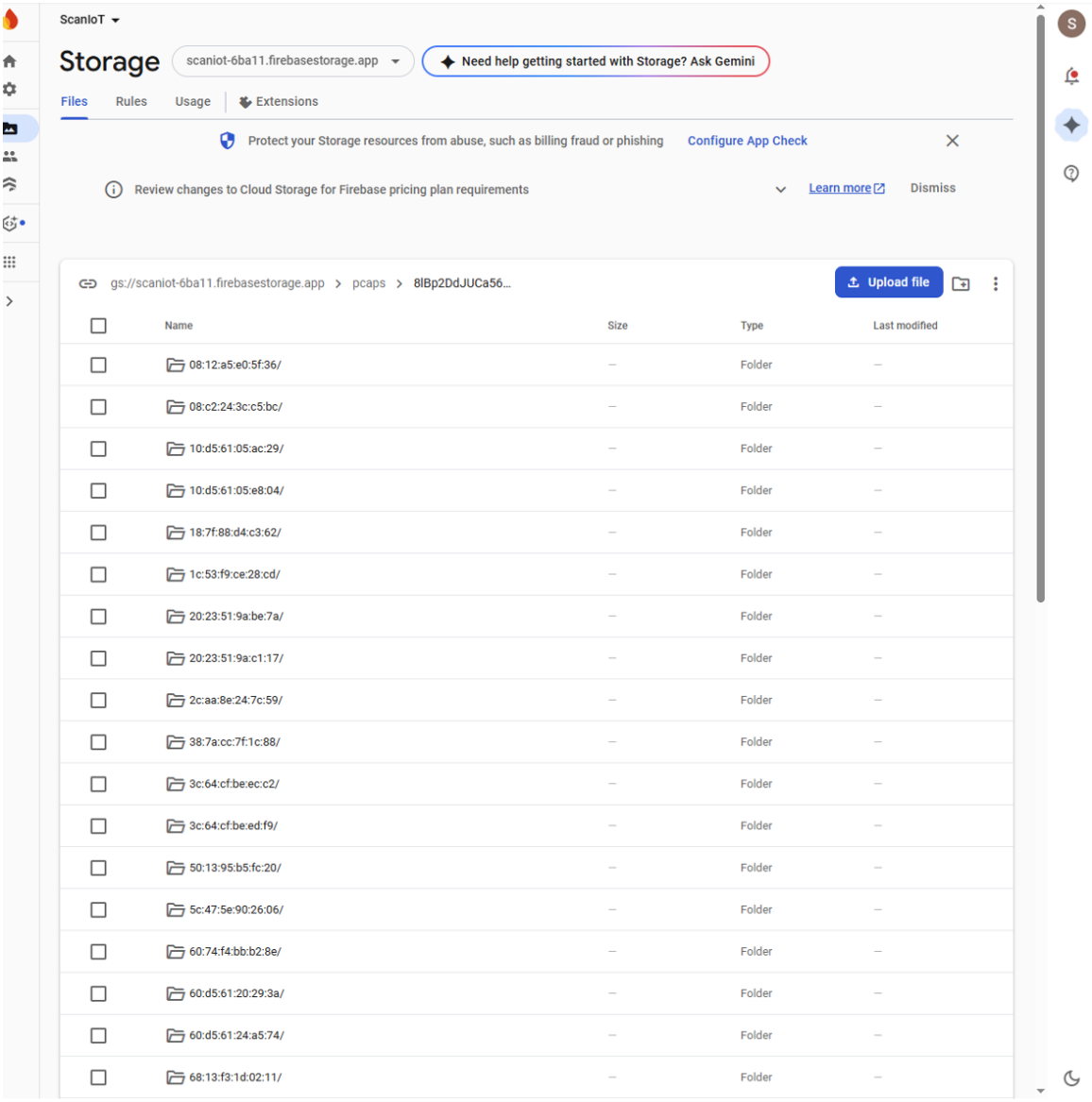


Figure 4.16: Captured traffic PCAP files saved in Firebase.

4.3.1 Similarities

From a functional standpoint, the majority of features provided by ScanIoT and DroidScour are analogous. Both frameworks offer an intuitive and interactive Graphical User Interface (GUI) that enables seamless device-specific traffic capture and management. Each system allows users to record and store traffic traces for individual IoT devices, even when multiple devices are monitored concurrently. Additionally, both frameworks can be deployed in any Wi-Fi network environment to collect traffic from connected devices while allowing users to annotate relevant device information. A further common feature is the ability to capture and store images of IoT devices with uncertain identities, facilitating subsequent expert verification. Both systems support scalable dataset generation, allowing new devices to be easily captured, annotated, and incorporated into the database. Unlike traditional approaches, they do not require large-scale network deployments and can accommodate new traffic collection following firmware updates to existing devices. Although primarily developed for smart home environments, ScanIoT and DroidScour are equally applicable to any network scenario involving Wi-Fi connected devices, making them versatile tools for real-world IoT data collection and labeling.

4.3.2 Differences

Despite their functional similarities, ScanIoT and DroidScour differ significantly in terms of deployment architecture, hardware dependency, and data management mechanisms. ScanIoT operates on a Raspberry Pi platform, requiring minimal yet dedicated hardware components such as a Wi-Fi adapter and local storage, thereby making it suitable for continuous, standalone monitoring in smart home environments. In contrast, DroidScour is entirely software-based and operates on a rooted Android device, eliminating the need for additional hardware infrastructure. While ScanIoT stores packet capture files locally on the Raspberry Pi, DroidScour integrates with Firebase for real-time synchronization, cloud-based storage, and secure data sharing. Furthermore, DroidScour introduces mobile-centric features such as time-bound traffic capture and remote data accessibility, enhancing user convenience and portability. Conversely, ScanIoT offers greater flexibility for large-scale or static deployments, particularly where persistent data collection and local network control

are required. These distinctions underscore the complementary nature of the two frameworks, with ScanIoT optimized for fixed network monitoring and DroidScour designed for lightweight, on-the-go traffic analysis.

4.4 Significance

The rapid proliferation of Internet of Things (IoT) devices across residential and industrial environments has intensified the need for accurate device identification and behavioral profiling. However, most existing IoT datasets reported in the literature have been generated within tightly controlled laboratory settings employing complex network topologies. While such environments allow precise manipulation of experimental parameters, they often introduce artificial latency and lack the heterogeneity and scalability observed in real-world IoT ecosystems. Moreover, replicating these sophisticated infrastructures across research institutions presents a major challenge, limiting collaborative dataset generation and comparative analysis.

Another critical limitation of laboratory generated datasets lies in their inherent class imbalance and restricted device diversity. Many controlled experiments involve only a limited number of IoT devices, with several exhibiting minimal or no traffic during data collection. This imbalance not only constrains dataset representativeness but also hampers the generalizability and robustness of machine learning models trained for IoT device identification and anomaly detection. Consequently, existing approaches may perform well in confined testbeds but fail to adapt to the dynamic and heterogeneous conditions of real-world networks.

Recognizing these limitations, this research contributes two novel traffic capturing fraameworks, ScanIoT and DroidScour, that collectively address the challenges of scalability, diversity, and realism in IoT traffic collection. ScanIoT and DroidScour are designed to facilitate distributed and automated discovery of IoT devices across heterogeneous networks, enabling large scale identification of connected devices in varied environments. Also, these frsmeworks complement this process by collecting detailed and structured network traffic traces from discovered devices. Together, these tools support the development of scalable, cross domain datasets that more accurately reflect real world IoT deployments.

By enabling consistent and distributed traffic collection across multiple smart home and enterprise networks, the proposed frameworks empower researchers to build more transferable, generalizable, and robust machine learning models for IoT device identification, behavioral profiling, and anomaly detection. The significance of this study, therefore, lies in bridging the gap between controlled laboratory datasets and realistic IoT environments, promoting reproducibility, collaboration, and advancement in the broader field of IoT cybersecurity research.

Chapter 5

Datasets

This chapter provides a comprehensive overview of the processes involved in constructing device identification datasets from existing IoT traffic data. Specifically, it details the methodology for converting the CICIoT2023 dataset into a device identification dataset, describing the steps taken to extract, organize, and annotate traffic for each IoT device, along with a characterization of the devices included. In addition, the chapter introduces the Concordia University IoT device identification dataset (CU2025), highlighting the devices represented and the structure of the dataset. These datasets serve as critical resources for developing and evaluating machine learning algorithms for IoT device identification, enabling researchers to model realistic network environments. By documenting both the dataset preparation processes and device level details, this chapter establishes a foundation for subsequent analysis and comparisons, facilitating reproducibility and scalability in IoT research.

5.1 CICIoT2023 dataset

The CICIoT2023 dataset [3] was originally developed to facilitate anomaly detection in Internet of Things (IoT) environments. It was subsequently extended and restructured into the CICIoT-DIAD (Device Identification and Anomaly Detection) 2024 dataset [122], which supports both device identification and anomaly detection tasks. Although the dataset conversion methodology

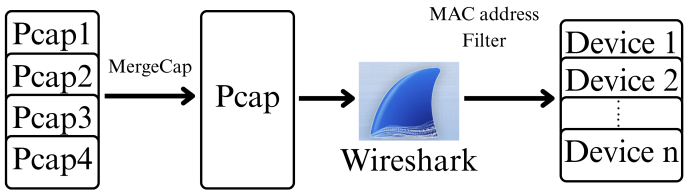


Figure 5.1: CICIoT2023 dataset processed to extract device-specific traffic.

was later provided by the original authors, the accompanying repository included limited documentation. Furthermore, our research had already independently undertaken the dataset conversion process prior to the public release of their methodology. Therefore, the approach described in this work outlines our own comprehensive procedure for transforming the CICIoT2023 dataset into a device identification dataset, ensuring methodological transparency and reproducibility for subsequent analysis.

The CICIoT2023 dataset was initially designed for anomaly detection in IoT networks; however, it was later repurposed by the same research group for device identification, without addressing several inherent limitations. Prior to the publication of their updated version, our research investigated these limitations and developed an independent conversion workflow. The benign traffic within the CICIoT2023 dataset is distributed across four separate packet capture (PCAP) files, collectively representing approximately sixteen hours of network activity. Since the dataset provides the MAC addresses of all IoT devices present in the testbed, we utilized these identifiers to extract device-specific traffic after merging all benign PCAP files. The merging process was performed using MergeCap, a command line utility capable of combining multiple PCAP files efficiently. Although Wireshark [139] also supports merging, it requires iterative manual operations through its graphical interface, merging only two files at a time.

Following the merge, a single consolidated PCAP file containing traffic from all devices was produced. This unified capture was then processed in Wireshark, where device specific MAC address filters were applied to isolate the traffic corresponding to each IoT device. This extraction process was repeated for forty-seven devices, and the resulting filtered traffic for each device was saved separately, as illustrated in Figure 5.1. The extracted device metadata including MAC addresses, device names, and functional categories is summarized in Table 5.1.

Table 5.1: CICIoT2023 devices extracted details.

Sr.	MAC Addresses	Device	Category
1	1C:FE:2B:98:16:DD	Amazon Alexa Echo Dot 1	Audio
2	2C:71:FF:05:F1:15	Amazon Echo Show	Audio
3	08:7C:39:CE:6E:2A	Amazon Alexa Echo Studio	Audio
4	CC:F4:11:9C:D0:00	Google Nest Mini Speaker	Audio
5	B0:F1:EC:D3:E7:98	harman kardon (Ampak Technology)	Audio
6	48:A6:B8:F9:1B:88	Sonos One Speaker	Audio
7	9C:8E:CD:1D:AB:9F	AMCREST WiFi Camera	Camera
8	3C:37:86:6F:B9:51	Arlo Base Station	Camera
9	44:01:BB:EC:10:4A	HeimVision SmartWiFi Camera	Camera
10	34:75:63:73:F3:36	Home Eye Camera	Camera
11	70:EE:50:68:0E:32	Netatmo Camera	Camera
12	10:5A:17:97:A5:C6	Rbcior Camera	Camera
13	6C:5A:B0:44:1D:90	TP-Link Tapo Camera	Camera
14	7C:78:B2:86:0D:81	Wyze Camera	Camera
15	84:7A:B6:64:62:58	Yi Indoor Camera	Camera
16	84:7A:B6:62:3A:6C	Yi Indoor 2 Camera	Camera
17	2C:D2:6B:66:D2:87	Yi Outdoor Camera	Camera
18	8C:85:80:6C:B6:47	Eufy HomeBase 2	Hub
19	68:57:2D:56:AC:47	Atomi Coffee Maker	HA
20	C4:DD:57:13:07:C6	GoSund Bulb	Lighting
21	84:E3:42:42:ED:0B	Lumiman bulb	Lighting
22	00:17:88:60:D6:4F	Philips Hue Bridge	Hub
23	18:69:D8:EB:D4:3E	Teckin Light Strip	Lighting
24	D4:A6:51:76:06:64	Teckin Plug 1	PO
25	D4:AD:FC:29:C8:A2	Govee Smart Humidifer	HA

Continued on next page

Table 5.1 – continued from previous page

Sr.	MAC Addresses	Device	Category
26	D4:A6:51:20:91:D1	Yutron Plug 1	PO
27	D4:A6:51:21:6C:29	Yutron Plug 2	PO
28	AC:F1:08:4E:00:82	LG Smart TV	HA
29	70:EE:50:6B:A8:1A	NetatmoWeather Station	HA
30	DC:A6:32:C9:E6:F4	RaspberryPi	NextGen
31	44:BB:3B:00:39:07	Nest Indoor Camera	Camera
32	50:02:91:1A:CE:E1	Gosund Power strip (1)	PO
33	40:5D:82:35:14:C8	Arlo Q Indoor Camera	Camera
34	B8:F0:09:03:29:79	GoSund Smart plug WP2 (1)	PO
35	B8:5F:98:D0:76:E6	Amazon Plug	PO
36	08:3A:F2:1D:BC:68	Cocoon Smart HVAC Fan	HA
37	C4:DD:57:0C:39:94	GoSund Smart Plug WP3 (1)	PO
38	B0:C5:54:59:2E:99	DCS8000LHA1 D-Link Mini Camera	Camera
39	D4:A6:51:30:64:B7	HeimVision SmartLife Radio/Lamp	Lighting
40	F4:CF:A2:34:48:6B	LampUX RGB	Lighting
41	1C:9D:C2:8C:9A:94	Levoit Air Purifier	HA
42	28:6D:97:7A:2B:2D	SmartThings Hub	Hub
43	28:6D:97:9E:F4:D5	AeoTec Smart Home Hub	Hub
44	50:14:79:37:80:18	iRobot Roomba	HA
45	D0:73:D5:35:FB:C8	LIFX Lightbulb	Lighting
46	00:02:75:F6:E3:CB	Smart Board	HA
47	30:23:03:F3:57:CB	Wemo smart plug 2	PO

The IoT devices included in the CICIoT2023 dataset were categorized into six primary functional groups: Camera, Hub, Audio, Power Outlet, Lighting, and Home Automation. Although a Raspberry Pi was present within the IoT network, it was not assigned to any of the aforementioned

categories, as it primarily served as part of the network infrastructure. Nonetheless, the Raspberry Pi was treated as a distinct device instance for the purpose of device classification.

5.2 CU2025 dataset

Although both the ScanIoT and DroidScour frameworks are capable of facilitating IoT device data collection and annotation, the Concordia University IoT Device Identification Dataset (CU2025) was collected exclusively using the DroidScour framework (Section 4.2). The dataset comprises a comprehensive set of IoT devices, as listed in Table 5.2. In total, traffic was captured from twenty-seven devices, including several instances of duplicate models. The dataset encompasses a diverse range of IoT device categories, such as cameras, audio devices, smart lighting systems, home automation units, smart plugs, and smart sensors. Device models were used to differentiate among devices sharing identical product names but representing distinct hardware configurations or firmware versions. For example, the WiFi Smart Camera category includes four devices corresponding to two unique models, each with two duplicate instances. Similarly, duplicate instances were recorded for the WiFi Smart Light, Kasa Smart WiFi Plug, Blink Mini 2 Camera, and Tapo Pan/Tilt Home Security WiFi Camera during traffic collection. This redundancy was intentionally preserved to capture intra model variations and enhance the dataset's representativeness for real world IoT device identification tasks.

Table 5.2: Devices details in CU2025 dataset.

Sr.	MAC address	Device Names	Device Model
1	08:12:A5:E0:5F:36	Amazon Echo Dot (3rd Gen)	C78MP8
2	18:7F:88:D4:C3:62	Ring Battery Doorbell	5F97F2
3	E8:4C:4A:B7:1B:DB	Blink Video Doorbell	BDM00200U
4	DC:A0:D0:E1:AC:68	Blink Outdoor Camera	BCM00500U
5	68:13:F3:5E:E0:0A	Blink Sync Module	BSM00401U
6	5C:47:5E:90:26:06	Chime Ring	5F67E9

Continued on next page

Table 5.2 – continued from previous page

Sr.	MAC address	Device Names	Device Model
7	1C:53:F9:CE:28:CD	Google Nest Hub	GUIK2
8	9C:C8:E9:82:95:31	Amazon Echo Pop	C2H4R9
9	BC:DF:58:0A:CA:54	Google Nest Mini (2nd Gen)	H2C
10	60:74:F4:BB:B2:8E	Goove Life Thermo-Hygrometer	H5103
11	C4:82:E1:25:0A:2A	Rain Point WiFi Water Timer	TWG004WRF
12	AC:9F:C3:38:BF:58	Ring StickUp Camera	5UM7E5
13	EC:64:C9:E2:A7:84	Smart Lock Gateway	G2
14	2C:AA:8E:24:7C:59	Wyze Camera Pan	WYZECP1
15	50:13:95:B5:FC:20	YI 1080p Home Camera	YYS.2016
16	A4:86:DB:5C:1F:D1	WiFi Smart Camera	T-CP8082LF-W3M
17	A4:86:DB:59:B6:C2	WiFi Smart Camera	T-CP8082LF-W3M
18	A4:86:DB:84:DB:4C	WiFi Smart Camera	T-CP8050LF-W3M
19	A4:86:DB:84:DB:86	WiFi Smart Camera	T-CP8050LF-W3M
20	10:D5:61:05:AC:29	Treat Life WiFi Smart Light	SL20
21	10:D5:61:05:E8:04	Treat Life WiFi Smart Light	SL20
22	20:23:51:9A:C1:17	Kasa Smart WiFi Plug	HS103
23	20:23:51:9A:BE:7A	Kasa Smart WiFi Plug	HS103
24	9C:C8:E9:C2:89:76	Camera Blink Mini 2	BCM00700U
25	68:13:F3:1D:02:11	Camera Blink Mini 2	BCM00700U
26	3C:64:CF:BE:EC:C2	Tapo Pan/Tilt Home Security WiFi Camera	Tapo C210
27	3C:64:CF:BE:ED:F9	Tapo Pan/Tilt Home Security WiFi Camera	Tapo C210

All IoT devices were deployed within a realistic home network environment, allowing for natural human device interactions during data collection. The data acquisition process for the CU2025 dataset is ongoing, with new devices being periodically integrated and their corresponding network

Figure 5.2: IoT devices in CU2025 dataset.

traffic continuously captured. Traffic collection for the devices listed in Table 5.2 has been conducted over a period exceeding two months. The volume of traffic generated by each device varies significantly depending on its functionality and activity level. Smart home automation units and camera devices typically produce high traffic volumes, whereas smart plugs, smart lights, and smart sensors generate comparatively lower volumes. For high traffic devices, around 300,000 samples have been collected per device, while for low traffic devices, around 100,000 samples have been collected per device. Figure 5.2 provides a visual overview of the IoT devices comprising the CU2025 dataset, depicting the diversity of device types used during data collection.

Given the heterogeneous nature of IoT traffic, contributors may capture and share datasets consisting of multiple smaller traffic segments. To ensure consistency, we developed an aggregation script that consolidates all packet capture (PCAP) files within each device directory and produces a unified PCAP file containing the combined traffic. This process enables the seamless integration of fragmented captures into a coherent dataset, thereby facilitating scalability and reproducibility across different data collection instances.

5.3 Remarks

Most existing IoT datasets have been generated within controlled laboratory environments employing complex and often highly specialized network configurations. Although such settings enable precise manipulation of experimental variables, they also introduce artificial latency arising from the intricate interconnections among multiple networking devices. Furthermore, reproducing these sophisticated infrastructures across different research institutions poses significant challenges, thereby limiting the scalability, reproducibility, and collaborative advancement of IoT datasets. Laboratory generated datasets are inherently constrained by the limited range of IoT devices available for experimentation, which stands in stark contrast to the extensive and heterogeneous device deployments commonly observed in real world smart home ecosystems. Unlike these controlled testbeds, smart home networks encompass diverse devices operating across distinct network conditions, resulting in highly dynamic and representative traffic patterns.

Another persistent limitation of laboratory generated datasets is the pronounced class imbalance, as summarized in Table 3.1. Our analysis shows that several devices in CICIoT2023 dataset generate minimal or, in some cases, no meaningful traffic within the recorded 16-hour collection window. Such imbalance substantially restricts the applicability of machine learning models trained on these datasets and reduces their ability to generalize across unseen environments.

Motivated by these shortcomings, this research underscores the necessity for scalable, diverse, and distributed data collection across heterogeneous, real world networks. Capturing and profiling similar IoT devices across multiple naturally occurring environments enables the development of

more robust and transferable machine learning models. To address this need, we introduce two traffic capturing frameworks, ScanIoT and DroidScour, designed to support scalable, and realistic IoT traffic acquisition for device identification and behavioral analysis. Using the DroidScour framework, we collected the Concordia University IoT Device Identification Dataset (CU2025). Unlike prior datasets, CU2025 was intentionally constructed to achieve a well-balanced representation of device traffic, ensuring sufficient traffic samples for each device.

The CU2025 dataset empowers researchers to develop more transferable, generalizable, and resilient machine learning models for IoT device identification, behavioral profiling, and anomaly detection. One of the significant contribution of this work lies in bridging the gap between traditional laboratory based datasets and real world IoT environments, thereby advancing reproducibility, fostering cross institutional collaboration, and contributing to progress in the broader domain of IoT research.

Chapter 6

Device Identification Methodology

This chapter presents the methodology adopted in this study for IoT device identification. It provides a detailed description of the Deep Feedforward Neural Network (DFNN) model employed for both device type and device category identification. The chapter also outlines the complete labeling methodology used for the CICIoT2023 and CU2025 datasets, ensuring consistency and reproducibility across both data sources. Furthermore, the incremental learning strategy designed to enhance the DFNN's adaptability to newly introduced devices is thoroughly explained. Finally, the chapter specifies the evaluation metrics used to assess model performance and to report the results of the study.

6.1 Model Selection

The initial objective of this study was to identify a suitable existing model and subsequently enhance its capabilities for IoT device identification. However, after replicating several published approaches, it became evident that many reported results were either not reproducible or were highly dependent on narrowly tailored feature selection procedures that worked only within the authors' own experimental settings. Our aim, by contrast, was to develop a methodology capable of generalizing across multiple datasets rather than performing well on a single, isolated benchmark. This observation is consistent with findings in the literature, where numerous studies conduct experiments

under tightly controlled conditions and strong assumptions, resulting in models that lack generalizability and cannot be reliably reproduced on datasets collected from different environments [123].

To validate these concerns, we implemented and evaluated several models including LSTM-based [140] architecture that had previously been reported to achieve strong performance. Contrary to the results presented in the literature, none of the tested models reproduced the expected accuracy or behavior when applied to our datasets. These inconsistencies formed a key motivation for our work. Consequently, our proposed methodology was evaluated on two distinct datasets, rather than a single controlled testbed, to ensure robustness and reproducibility. As demonstrated in the Chapter 7, the proposed approach generalizes effectively across heterogeneous network environments and exhibits strong and consistent performance, thereby affirming the validity and reliability of our methodological framework.

Motivated by the strong performance of the Deep Feedforward Neural Network (DFNN) reported in [141], we implemented the baseline DFNN architecture to validate its suitability for our device identification task. The DFNN aligns closely with the requirements and characteristics of our datasets, making it an appropriate foundational model. Its capacity to learn complex nonlinear patterns, coupled with its hierarchical feature representation capabilities, enables the extraction of subtle patterns that may not be apparent through manual feature engineering. Additionally, the DFNN offers excellent scalability for large and diverse datasets containing numerous device classes and high-dimensional feature spaces. It also demonstrates strong generalizability, robustness to noise, and rapid inference speeds suitable for real-time device identification. Moreover, its lightweight architecture facilitates efficient deployment on resource-constrained environments. For these reasons, the DFNN was selected as the primary model for both IoT device-type and device-category classification in this study.

6.2 Deep Feedforward Neural Network (DFNN)

A Deep Feedforward Neural Network (DFNN) is a class of artificial neural network (ANN) in which information propagates unidirectionally, from the input layer through successive hidden layers to the output layer, without forming cyclic connections. DFNNs represent a foundational

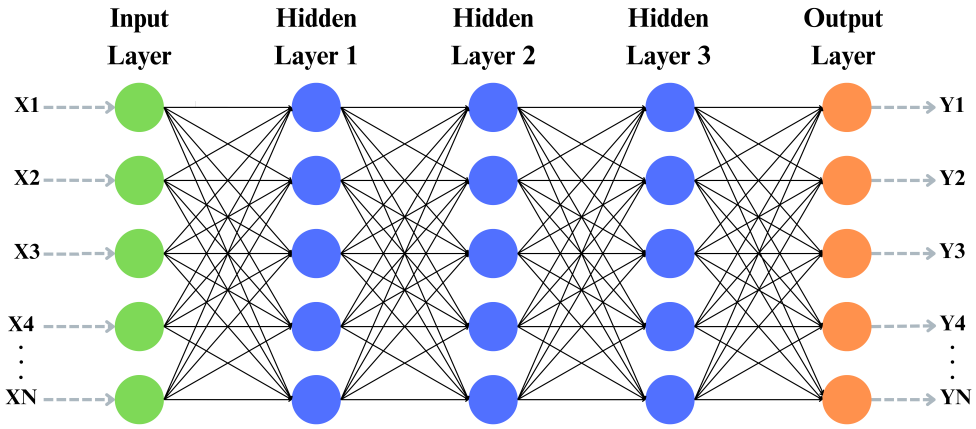


Figure 6.1: DFNN.

architecture in deep learning and are widely adopted for regression, classification, and feature extraction tasks. A representative DFNN architecture comprising an input layer, three hidden layers, and an output layer is shown in Figure 6.1. Owing to their ability to learn progressively abstract feature representations across layers, DFNNs offer substantial advantages for classification problems. With sufficiently large hidden layers and an appropriate number of neurons, DFNNs are capable of modeling highly complex, nonlinear decision boundaries. Moreover, when combined with appropriate regularization techniques, DFNNs exhibit strong robustness to noisy or partially incomplete input data, making them well-suited for real-world IoT traffic classification scenarios.

6.2.1 Feature Extraction

Feature extraction [142] is essential for deriving meaningful information from Packet Capture (PCAP) files, enabling machine learning algorithms to learn effectively and achieve accurate classification. The feature set selected for this study is designed to capture the distinctive communication patterns and protocol level characteristics that differentiate devices within a network environment. Devices often exhibit consistent and unique behaviors across the TCP/IP stack ,including connection formation, protocol usage, timing patterns, and header configurations, necessitating a diverse set of features spanning multiple network layers.

At the application layer, features derived from protocol usage and DNS-related attributes provide insight into device interactions with external services, application dependencies, and query

frequency. These behaviors often form stable, device-specific signatures influenced by operating systems, firmware, and built-in applications. Payload length and related metrics serve as indirect indicators of device functionality and routine communication patterns without inspecting the actual content.

Transport-layer features offer additional discriminatory value by describing how devices manage end-to-end communication. Device classes often exhibit consistent usage patterns in transport protocols, preferred ports, TCP flag sequences, and connection setup behaviors. TCP sequence and acknowledgment values, as well as segment lengths, can reveal manufacturer- or OS-specific stack implementations. Incorporating these features enables the DFNN to capture subtle but reliable communication fingerprints not immediately apparent at higher layers.

Internet-layer attributes, including IP packet length, TTL values, and protocol identifiers, further enrich device representations by reflecting routing decisions and stack level defaults. TTL values, often determined by operating system settings, provide a reliable signal for distinguishing device families. ICMP related fields contribute additional context by reflecting diagnostic responses and error messaging behaviors that vary with system configuration.

At the link layer, MAC addresses and ARP related fields provide insights into device identity and local network resolution behaviors. Even when MAC addresses are randomized, ARP interactions—how devices announce themselves and respond to address resolution—can reveal patterns tied to hardware or firmware. Including these features improves understanding of device local interactions, particularly when higher layer information is limited.

Statistical and cross-layer features are critical for capturing temporal and volumetric characteristics of device traffic. Time based measures, session duration, inter-arrival times, and packet-size statistics collectively describe the rhythm and consistency of network communications. Devices often generate predictable traffic patterns due to automated tasks, background services, or scheduled updates. By integrating these temporal dynamics with protocol header information, the DFNN can learn holistic behavioral signatures that differentiate devices, even when protocol usage appears similar.

Several candidate features were initially considered and systematically evaluated to determine the combination that produced the most accurate classification results. Feature selection was guided

not only by quantitative analysis of the dataset but also by expert knowledge of IoT device behaviors and networking protocols. This ensured that the final feature set was both meaningful and discriminative, enabling the model to identify devices accurately across heterogeneous network environments.

Collectively, these features provide a multidimensional representation of device behavior that aligns with the strengths of a DFNN. By combining semantic, structural, and temporal characteristics, the model can learn nuanced relationships distinguishing devices at multiple layers of abstraction. This comprehensive approach ensures accurate and robust device identification in heterogeneous network environments, where single-layer or single-feature approaches are insufficient.

For practical implementation, network traffic was parsed and structured into Comma-Separated Values (CSV) files. Preprocessing and feature extraction were conducted using the Scapy framework, enabling flexible parsing of diverse network protocols. Feature extraction included both packet-level and session-level metrics, applied consistently for device identification and category classification tasks. The process followed a sequential parsing of PCAP files to extract features across the TCP/IP layer model [143], as well as cross-layer and statistical features. A detailed description of each feature and its corresponding TCP/IP layer is provided in Table 6.1. All extracted features were structured in CSV format for each device, capturing temporal, statistical, and protocol-layer information to provide comprehensive contextual input for the DFNN.

Table 6.1: Feature extraction details.

TCP/IP Layer	**Feature Name(s)**	**Description**
Application Layer	app_protocol	Inferred application-level protocol (HTTP, DNS, etc.)
	dns_id, dns_qr, dns_opcode, dns_count, dns_query_name	DNS metadata: ID, query/response flag, opcode, etc.
	payload, raw_payload	Payload length from TCP/UDP and total raw payload

Continued on next page

Table 6.1 – continued from previous page

TCP/IP Layer	Feature Name(s)	Description
Transport Layer	protocol	Transport protocol (TCP or UDP)
	src_port, dst_port	Source and destination port
	flags, seq, ack	TCP flags, sequence, and acknowledgment numbers
	tcp_len, udp_len	Lengths of TCP segment or UDP datagram
Internet Layer	src_ip, dst_ip	Source and destination IP addresses
	ip_len	Length of the IP packet
	ttl	Time-to-live (hop limit)
	ip_proto	Protocol number in IP header
	icmp_type, icmp_code	ICMP message type and code
Link Layer	src_mac, dst_mac	Source and destination MAC addresses
	arp_op, src_arp_mac, dst_arp_mac, src_arp_ip, dst_arp_ip	ARP fields: op code, MAC and IP addresses
Cross-Layer / Statistical	timestamp, time_diff	Packet timestamp and time difference since last packet
	pkt_len	Total packet length (all layers)
	session_duration	Time since session start
	avg_pkt_size, max_pkt_size, min_pkt_size	Session-level packet size statistics

6.2.2 Data Preparation

After feature extraction, the resulting CSV files must be prepared for machine learning by performing several preprocessing steps to ensure the data is suitable for training the DFNN model. The Pandas library was used to efficiently load and manipulate the CSV files.

Categorical string values in the dataset, such as protocol types or device identifiers, were converted into numerical form using label encoding from the sklearn library. This conversion allows the neural network to process categorical data effectively while preserving distinct class information.

To ensure robust model evaluation, the dataset was split using stratified shuffle split, maintaining class distribution across the splits. Specifically, 90% of the data was allocated for training and testing, while the remaining 10% was reserved for validation. The 90% training/testing set was further subdivided into 90% for training and 10% for testing. This hierarchical split ensures that both training and evaluation sets adequately represent all device classes, preventing bias in the model's learning process.

For numerical stability and to improve convergence during training, the feature values were normalized using StandardScaler from sklearn. Standardization ensures that all features have zero mean and unit variance, which helps the DFNN learn more effectively and prevents features with larger ranges from dominating the learning process.

By combining label encoding, stratified splitting, and normalization, the dataset is transformed into a structured and well-scaled format, optimized for accurate and efficient learning by the DFNN.

Since the objective of the study was to perform device type and category identification, appropriate labels were appended to each data. For every CSV corresponding to a specific device, a Device Type Label (DTL) was included to uniquely identify the device, while a Device Category Label (DCL) was added to indicate the type or category of the device (e.g., camera, smart plug, sensor). These labels provide the ground truth necessary for supervised learning, enabling the DFNN to learn not only to distinguish individual devices but also to generalize across device categories.

CICIoT2023 dataset

Table 6.2 provides the labeling details for the CICIoT2023 dataset. The IoT devices in this dataset were organized into six distinct categories: camera, hub, audio, power outlet, lighting, and home automation. The Raspberry Pi device was excluded from category classification, as it functioned as part of the underlying IoT network infrastructure; however, it was included for device type identification. This labeling scheme ensures that the dataset supports both granular device-level recognition as well as broader category-level classification for supervised learning tasks.

Table 6.2: CICIoT2023 device labels.

Sr.	MAC Addresses	Device	Category	DCL	DTL
1	1C:FE:2B:98:16:DD	Amazon Alexa Echo Dot 1	Audio	0	0
2	2C:71:FF:05:F1:15	Amazon Echo Show	Audio	0	1
3	08:7C:39:CE:6E:2A	Amazon Alexa Echo Studio	Audio	0	2
4	CC:F4:11:9C:D0:00	Google Nest Mini Speaker	Audio	0	3
5	B0:F1:EC:D3:E7:98	harman kardon (Ampak Technology)	Audio	0	4
6	48:A6:B8:F9:1B:88	Sonos One Speaker	Audio	0	5
7	9C:8E:CD:1D:AB:9F	AMCREST WiFi Camera	Camera	1	6
8	3C:37:86:6F:B9:51	Arlo Base Station	Camera	1	7
9	44:01:BB:EC:10:4A	HeimVision SmartWiFi Camera	Camera	1	8
10	34:75:63:73:F3:36	Home Eye Camera	Camera	1	9
11	70:EE:50:68:0E:32	Netatmo Camera	Camera	1	10
12	10:5A:17:97:A5:C6	Rbcior Camera	Camera	1	11
13	6C:5A:B0:44:1D:90	TP-Link Tapo Camera	Camera	1	12
14	7C:78:B2:86:0D:81	Wyze Camera	Camera	1	13
15	84:7A:B6:64:62:58	Yi Indoor Camera	Camera	1	14
16	84:7A:B6:62:3A:6C	Yi Indoor 2 Camera	Camera	1	15
17	2C:D2:6B:66:D2:87	Yi Outdoor Camera	Camera	1	16

Continued on next page

Table 6.2 – continued from previous page

Sr.	MAC Addresses	Device	Category	DCL	DTL
18	8C:85:80:6C:B6:47	Eufy HomeBase 2	Hub	2	17
19	68:57:2D:56:AC:47	Atomi Coffee Maker	HA	3	18
20	C4:DD:57:13:07:C6	GoSund Bulb	Lighting	4	19
21	84:E3:42:42:ED:0B	Lumiman bulb	Lighting	4	20
22	00:17:88:60:D6:4F	Philips Hue Bridge	Hub	2	21
23	18:69:D8:EB:D4:3E	Teckin Light Strip	Lighting	4	22
24	D4:A6:51:76:06:64	Teckin Plug 1	PO	5	23
25	D4:AD:FC:29:C8:A2	Govee Smart Humidifer	HA	3	24
26	D4:A6:51:20:91:D1	Yutron Plug 1	PO	5	25
27	D4:A6:51:21:6C:29	Yutron Plug 2	PO	5	26
28	AC:F1:08:4E:00:82	LG Smart TV	HA	3	27
29	70:EE:50:6B:A8:1A	NetatmoWeather Station	HA	3	28
30	DC:A6:32:C9:E6:F4	RaspberryPi	NextGen		29
31	44:BB:3B:00:39:07	Nest Indoor Camera	Camera	1	30
32	50:02:91:1A:CE:E1	Gosund Power strip (1)	PO	5	31
33	40:5D:82:35:14:C8	Arlo Q Indoor Camera	Camera	1	32
34	B8:F0:09:03:29:79	GoSund Smart plug WP2 (1)	PO	5	33
35	B8:5F:98:D0:76:E6	Amazon Plug	PO	5	34
36	08:3A:F2:1D:BC:68	Cocoon Smart HVAC Fan	HA	3	35
37	C4:DD:57:0C:39:94	GoSund Smart Plug WP3 (1)	PO	5	36
38	B0:C5:54:59:2E:99	DCS8000LHA1 D-Link Mini Camera	Camera	1	37
39	D4:A6:51:30:64:B7	HeimVision SmartLife Radio/Lamp	Lighting	4	38
40	F4:CF:A2:34:48:6B	LampUX RGB	Lighting	4	39
41	1C:9D:C2:8C:9A:94	Levoit Air Purifier	HA	3	40
42	28:6D:97:7A:2B:2D	SmartThings Hub	Hub	2	41

Continued on next page

Table 6.2 – continued from previous page

Sr.	MAC Addresses	Device	Category	DCL	DTL
43	28:6D:97:9E:F4:D5	AeoTec Smart Home Hub	Hub	2	42
44	50:14:79:37:80:18	iRobot Roomba	HA	3	43
45	D0:73:D5:35:FB:C8	LIFX Lightbulb	Lighting	4	44
46	00:02:75:F6:E3:CB	Smart Board	HA	3	45
47	30:23:03:F3:57:CB	Wemo smart plug 2	PO	5	46

Abbreviations

DCL Device Category Label **DTL** Device Type Label **HA** Home Automation **PO** Power Outlet

CU2025 dataset

Table 6.3 presents the labeling scheme adopted for the CU2025 dataset, detailing the device-specific labels used for machine learning tasks. Unlike the CICIoT2023 dataset, which includes both device type and device category labels, the CU2025 dataset does not incorporate category labels. This omission is intentional and reflects the evolving nature of modern IoT devices, many of which exhibit multi-functional capabilities that blur traditional category boundaries. As a result, assigning a single functional category to these devices is neither representative nor methodologically sound. A comprehensive discussion of this shift in IoT device categorization and its implications for dataset construction is provided in Chapter 8.

Table 6.3: Device classification result on CU2025 dataset.

Sr.	Device Names	Device Type Label
1	Amazon Echo Dot (3rd Gen)	0
2	Ring Battery Doorbell	1
3	Blink Video Doorbell	2
4	Blink Outdoor Camera	3

Continued on next page

Table 6.3 – continued from previous page

Sr.	Device Names	Device Type Label
5	Blink Sync Module	4
6	Chime Ring	5
7	Google Nest Hub	6
8	Amazon Echo Pop	7
9	Google Nest Mini (2nd Gen)	8
10	Goove Life Thermo-Hygrometer	9
11	Rain Point WiFi Water Timer	10
12	Ring StickUp Camera	11
13	Smart Lock Gateway	12
14	Wyze Camera Pan	13
15	YI 1080p Home Camera	14
16	Wifi Smart Camera	15
17	Wifi Smart Camera2	16
18	Treat Life Wifi Smart Light	17
19	Kasa Smart WiFi Plug	18
20	Camera Blink Mini 2	19
21	Tapo Pan/Tilt Home Security WiFi Camera	20

6.2.3 DFNN architecture and design

The proposed DFNN architecture is well aligned with the requirements of IoT device identification, offering a balanced capacity for learning both low-level and high-level patterns embedded within network traffic. Effective device identification depends on modeling subtle, often non-linear relationships across packet-level and session-level attributes—including timing behavior, protocol usage, header configurations, and implementation-specific communication traits. The hierarchical structure of the three hidden layers enables the network to progressively learn and refine these relationships. The first hidden layer, composed of 128 neurons, provides sufficient representational

capacity to process the full set of extracted features and capture broad communication patterns. The subsequent hidden layers with 64 and 32 neurons compress these learned representations into increasingly abstract and discriminative forms, enabling the extraction of device specific communication signatures.

Multiple architectural configurations were evaluated during model development, including variations in the number of neurons and hidden layers. Neuron counts were gradually reduced while monitoring model performance to determine the minimal architecture capable of maintaining classification accuracy. The depth of the network was further reformed by incremental learning requirements, ensuring that the architecture remained effective across evolving datasets and device variations.

ReLU activation functions were applied across all hidden layers to improve training efficiency, promote sparsity, and mitigate vanishing gradient issues properties that are essential for learning from heterogeneous network traffic. A dropout layer with a rate of 0.3 was incorporated to reduce overfitting and enhance model generalization, preventing the network from over relying on specific traffic patterns or temporal variations. This regularization is particularly important in IoT environments, where device behavior may vary due to firmware updates, operational states, or environmental conditions. The output layer was configured to match the number of device classes, allowing the network to directly map learned representations to unique device identities.

Training was performed using the categorical cross-entropy loss function, which is well suited for multi-class classification tasks. The Adam optimizer was employed for gradient based optimization. Several learning rates, ranging from 0.01 to 0.00001, were systematically evaluated to identify a configuration that ensured stable convergence while maintaining responsiveness to gradient updates. Through empirical analysis, a learning rate of 0.0001 was selected, as it consistently provided the most reliable performance on both the training and validation sets. Model training was conducted using a batch size of 32 over 10 epochs. To further mitigate overfitting, an early stopping mechanism with a patience parameter of 2 was applied, ensuring that training terminated when no additional improvement was observed in validation performance.

Overall, the design choices in the DFNN architecture, activation functions, regularization strategy, and training configuration collectively produce a robust and efficient model capable of learning

rich, device specific communication signatures. These characteristics enable high accuracy device identification across diverse and evolving network environments.The details of the model architecture are summarized in Table 6.4.

Table 6.4: DFNN model architecture.

Layer	Type	Activation	Neurons	Purpose
Hidden Layer 1	Fully Connected	ReLU	Input_dim ->128	Receives Input Features
Hidden Layer 2	Fully Connected	ReLU	128 -> 64	Extracts deeper patterns
Hidden Layer 3	Fully Connected	ReLU	64 -> 32	Further feature refinement
Dropout Layer	Regularization	-	30% Neurons dropped	Prevents overfitting
Output Layer	Fully Connected		32 -> output_dim	Outputs class probabilities

6.2.4 Incremental Learning

As new IoT devices are continuously integrated into networks, effective device identification requires a model capable of incorporating previously unseen devices without retraining from scratch. This motivates the use of an incremental learning framework, which enables the classifier to assimilate new information while preserving knowledge of previously learned devices. The introduction of new devices into an existing classifier often leads to the problem of catastrophic forgetting, wherein the integration of new classes leads to the erosion of prior knowledge. Incremental learning addresses this issue to some extent by enabling the model to retain previously acquired knowledge while adapting to new data.

To support incremental device identification, the proposed DFNN architecture was extended with dynamic output-layer expansion and continual learning mechanism. When new devices are introduced, the output layer must accomodate an enlarged set of classes. Accordingly, the model dynamically expands the output layer at each incremental stage to support the classification of newly added devices. The parameters associated with previously learned classes are preserved, ensuring that the model retains existing knowledge. In contrast, weights corresponding to newly added classes are initialized randomly and subsequently optimized during incremental learning.

To further mitigate catastrophic forgetting, the framework integrates rehearsal memory, feature freezing, and knowledge distillation:

(1) Rehearsal Memory:

A memory buffer retains 10% of the previously observed samples from each device. During incremental learning, newly collected samples are combined with the rehearsal memory to retain the model. This prevents the decision boundaries from shifting excessively due to disproportionate exposure to new classes, thereby stabilizing perfromance across previously learned devices.

(2) Feature Freezing:

After the initial training phase, the weights of the first hidden layer, which capture low-level, device-agnostic communication features, are frozen. This ensures that foundational feature representations remain stable across all incremental steps. Only the higher-level layers are updated as new devices classes are introduced, allowing the model to adapt while preserving essential learned behavior.

(3) Knowledge Distillation:

Knowledge distillation further reduces catastrophic forgetting by transferring information from the previously trained model (teacher) to the updated model (student). A distillation loss is computed using the Kullback-Leibler (KL) divergence between the softened logits of the teacher and student models. In parallel, a hard target loss is computed using cross-entropy between the student's prediction and the ground-truth labels for newly added classes. The total loss function is a weighted combination of both components, enabling the student model to learn new devices patterns while maintaining consistency with the teacher model's output behavior.

The resulting framework offers substantial adaptability by incorporating new device classes without requiring architectural redesign beyond the output-layer expansion. Through the combined use of memory replay, feature preservation, and distillation-based knowledge transfer, the model effectively mitigates catastrophic forgetting and maintains stable performance across incremental updates. Moreover, because only the final layer grows during expansion while the core network remains compact, the framework is computationally efficient and scalable to the diverse and rapidly frowing populations of IoT devices commonly found in smart homes and enterprise networks.

6.3 Evaluation

To assess the effectiveness of the proposed DFNN model for IoT device identification, we conducted a comprehensive evaluation using standard classification metrics as well as Leave-One-Subject-Out Cross-Validation (LOSO-CV). This dual evaluation strategy provides insight into both overall classification performance and the model's ability to generalize to previously unseen devices.

6.3.1 Evaluation metrics

To evaluate the device type and device classification performance of the DFNN model, we used a confusion matrix [144] to gain valuable insights into the classifier's effectiveness. For correctly or incorrectly identifying the classification tasks, a confusion matrix relies on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). A device belonging to class X that is correctly identified as class X is considered a True Positive (TP). A device that does not belong to class X and is correctly predicted as not being class X is a True Negative (TN). A device that does not belong to class X but is incorrectly identified as class X is a False Positive (FP). Similarly, a device of class X that is incorrectly predicted as belonging to another class is a False Negative (FN).

The combination of TP, TN, FP, and FN is used to calculate accuracy, precision, recall, and $\mathbf{f}_1$ score, which further define the model's performance and classification details [145]. **Accuracy** presents the correct predictions from the total proportion and is evaluated as follows.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision measures the proportion of true positive predictions among all positive predictions made by the model, and is computed as follows.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall measures the model's ability to identify all instances of a specific device type correctly. It represents the proportion of actual devices of a given class that are correctly classified by the system.

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

The F1 Score is the harmonic mean of Precision and Recall, providing a single metric that balances both false positives and false negatives. It reflects how accurately and comprehensively the model identifies each device type, especially in cases of class imbalance, and is calculated as follows.

$$\text{F}_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

6.3.2 Leave-One-Subject-Out Cross-Validation

We employed Leave-One-Subject-Out Cross-Validation (LOSO-CV) to evaluate the model's generalization performance. This approach ensure that, in each iteration, all data from one device or subject is held out as the test set while the model is trained on the remaining devices. By repeating this process for all devices, LOSO-CV provides a rigorous measure of the model's ability to correctly identify previously unseen devices, reflecting realistic deployment scenarios in dynamic IoT networks. The advantages of LOSO-CV are as follows:

- **Tests generalization to unseen devices:**

  LOSO-CV simulates a real-world scenario where the model encounters a completely new device. By leaving out one subject during training, it evaluates how well the model generalize to devices it has not seen before.

- **Prevents data leakage:**

  The device-specific features could cause the model to leak some data that could appear both in the training as well as testing sets. LOSO-CV keeps all data from a device in the test set during its iteration, eliminating any overlap between training and testing for that device.

- **Reflects realistic deployment scenarios:**

  In practice, new IoT devices may be added to a network after a model has been deployed. LOSO-CV simulates this situation by ensuring the model is tested on completely unseen

devices.

- **Handles device specific features robustly:**

  LOSO-CV forces the model to rely on behavioral pattern and protocol-level features rather than memorizing identifiers, leading to a more robust identification model.

- **Reduces overfitting:**

  LOSO-CV prevents the model from relying too heavily on device-specific patterns in the training set because the test device is always unseen. This helps ensure that learned features are generalizable.

- **Insigt to per-device performance:**

  Since each device is left out once as the test set, the model is evaluated across all devices. This provides a more robust and unbiased estimate of overall performance than random train-test splits.

Chapter 7

Results

This chapter presents the results derived from the methodologies and observations detailed in Chapter 6. The findings include both device-category and device-type identification performance for the CICIoT2023 dataset, followed by device identification results for the CU2025 dataset. In addition, this chapter reports the outcomes of the incremental learning experiments and the Leave-One-Subject-Out Cross-Validation (LOSO-CV), providing a comprehensive evaluation of the proposed framework across multiple dimensions.

7.1 Device identification on CICIoT2023 dataset

7.1.1 Device-category identification on CICIoT2023

The evaluation outcomes for the device category classification task are summarized in Table 7.1. The model exhibited rapid convergence, with early stopping activated upon reaching a training accuracy of 99.95% and an associated test accuracy of 99.97%. These results indicate that the DFNN demonstrates highly effective performance in distinguishing IoT device categories within the CICIoT2023 dataset.

The DFNN demonstrated outstanding performance in classifying camera devices within the CICIoT2023 dataset, achieving a precision of 0.9998, a recall of 0.9999, and an F1 score of 0.9999. In contrast, the lowest performance was observed for the lighting device category, with a precision of 0.9945, a recall of 0.9964, and an F1 score of 0.9954. Among the evaluation metrics, the F1

Table 7.1: Device-category classification results

Device category	Precision	Recall	F_1 Score
Audio	0.9996	0.9995	0.9996
Camera	0.9998	0.9999	0.9999
Hub	0.9979	0.9983	0.9981
Home Automation	0.9984	0.9963	0.9973
Lighting	0.9945	0.9964	0.9954
Power Outlet	0.9993	0.9989	0.9991

score is typically regarded as the most representative, as it provides a balanced measure of precision and recall. Based on the F1 scores, the classification performance across device categories is ranked as follows: camera, audio, power outlet, hub, home automation, and lighting. Overall, the DFNN exhibits exceptionally strong classification capabilities, and the corresponding confusion matrix for device-category classification using the CICIoT2023 dataset is shown in Figure 7.1.

7.1.2 Device-type identification on CICIoT2023

The DFNN exhibited robust performance on the multi-class device-type classification task, attaining a test accuracy of 99.73%. Model convergence occurred rapidly, with early stopping triggered at epoch 6 upon reaching a training accuracy of 99.56%. Early stopping was integrated into the training procedure as a safeguard against overfitting and to ensure optimal generalization. Comprehensive per-class evaluation metrics are reported in Table 7.2. The majority of device types within the CICIoT2023 dataset achieved F1 scores exceeding 0.95, with several classes demonstrating performance levels above 0.99.

Notably, the Amazon Alexa Echo Dot achieved a precision of 0.9953, a recall of 0.9967, and an F1 score of 0.9960, while the Amazon Echo Show attained a precision of 0.9988, a recall of 0.9982, and an F1 score of 0.9985. Similarly, the Amazon Alexa Echo Studio achieved a precision of 0.9974, a recall of 0.9970, and an F1 score of 0.9972, highlighting consistent performance across device variants within the same product family. In the camera category, the Netatmo Camera achieved a precision of 0.9962, a recall of 0.9946, and an F1 score of 0.9955. The Nest Indoor Camera attained near-perfect performance, with a precision of 0.9996, a recall of 0.9997, and an F1 score of 0.9997. The Arlo Q Indoor Camera further demonstrated exceptional discriminative capability, achieving a

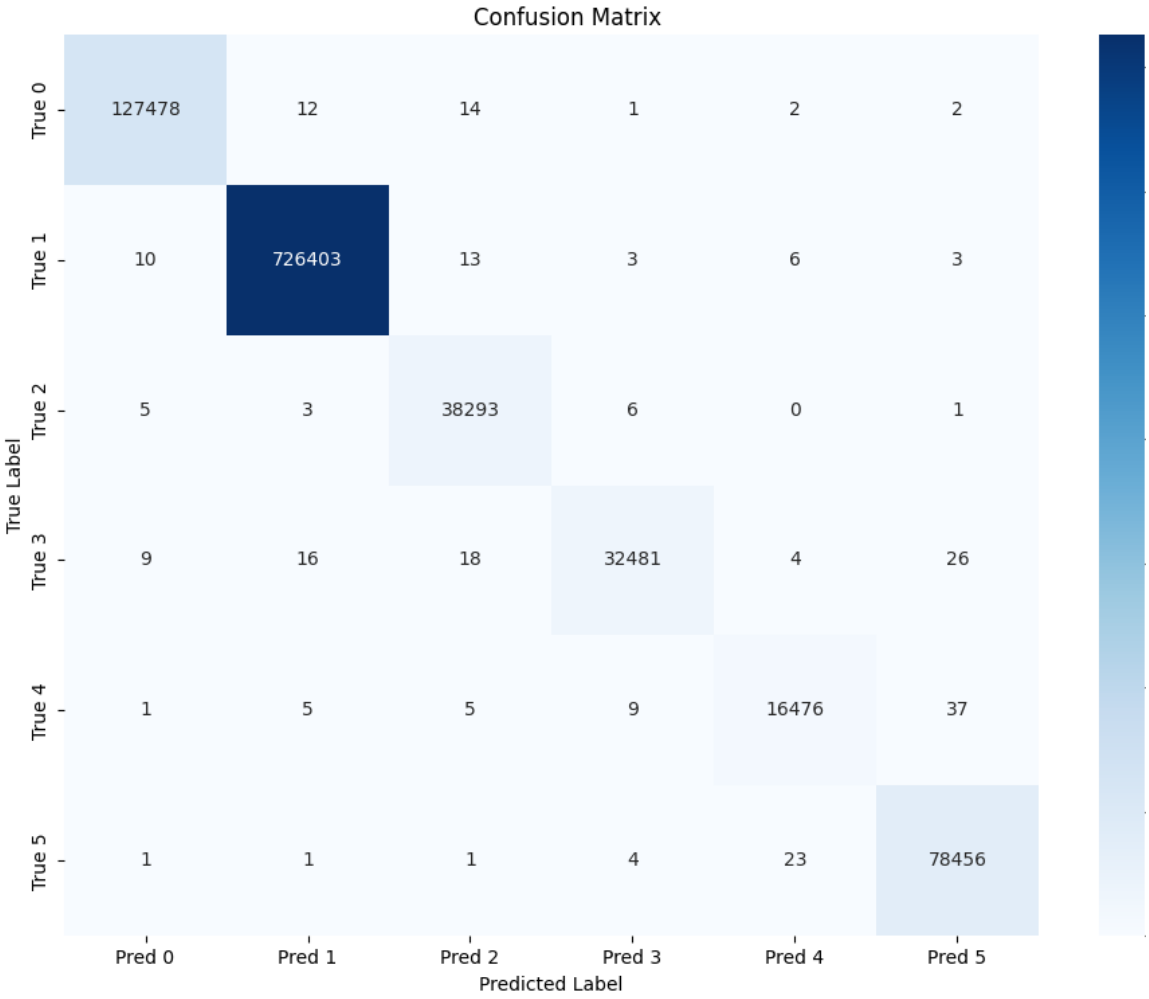


Figure 7.1: Confusion matrix for device-category identification.

precision, recall, and F1 score of 0.9999.

Collectively, these results underscore the effectiveness of the DFNN architecture in capturing distinctive feature representations necessary for fine-grained IoT device-type classification within the CICIoT2023 dataset. Overall, the DFNN exhibits exceptionally strong classification capabilities, and the corresponding confusion matrix for device-type classification using the CICIoT2023 dataset is shown in Figure 7.2.

Table 7.2: CICIoT2023 devices classification result using DFNN

No.	**Device**	**Precision**	**Recall**	**F_1 Score**
1	Amazon Alexa Echo Dot	0.9953	0.9967	0.9960
2	Amazon Echo Show	0.9988	0.9982	0.9985
3	Amazon Alexa Echo Studio	0.9974	0.9970	0.9972
4	Google Nest Mini Speaker	0.9931	0.9973	0.9952
5	harman kardon (Ampak Technology)	0.9937	0.9954	0.9946
6	Sonos One Speaker	0.9831	0.9701	0.9766
7	AMCREST WiFi Camera	0.9764	0.9836	0.9800
8	Arlo Base Station	0.9810	0.9769	0.9790
9	HeimVision SmartWiFi Camera	0.9913	0.9900	0.9907
10	Home Eye Camera	0.9961	0.9935	0.9948
11	Netatmo Camera	0.9962	0.9946	0.9954
12	Rbcior Camera	0.9917	0.9899	0.9908
13	TP-Link Tapo Camera	0.9818	0.9879	0.9848
14	Wyze Camera	0.9872	0.9789	0.9830
15	Yi Indoor Camera	0.9488	0.9756	0.9620
16	Yi Indoor 2 Camera	0.9756	0.9382	0.9566
17	Yi Outdoor Camera	0.9835	0.9846	0.9840
18	Eufy HomeBase 2	0.9863	0.9899	0.9881
19	Atomi Coffee Maker	0.9761	0.9793	0.9777

Continued on next page

Table 7.2 – continued from previous page

Sr.	Device	Precision	Recall	F_1 Score
20	GoSund Bulb	0.9858	0.9780	0.9819
21	Lumiman bulb	0.9861	0.9672	0.9766
22	Philips Hue Bridge	0.9920	0.9951	0.9935
23	Teckin Light Strip	0.9837	0.9843	0.9840
24	Teckin Plug 1	0.8732	0.7353	0.7984
25	Govee Smart Humidifer	0.9627	0.9261	0.9441
26	Yutron Plug 1	0.9776	0.9728	0.9752
27	Yutron Plug 2	0.7746	0.9006	0.8329
28	LG Smart TV	0.9951	0.9959	0.9955
29	NetatmoWeather Station	0.9860	0.9950	0.9905
30	RaspberryPi	0.9561	0.9230	0.9392
31	Nest Indoor Camera	0.9996	0.9997	0.9997
32	Gosund Power strip (1)	0.9780	0.9913	0.9846
33	Arlo Q Indoor Camera	0.9999	0.9999	0.9999
34	GoSund Smart plug WP2 (1)	0.9907	0.9856	0.9882
35	Amazon Plug	0.9615	0.9596	0.9605
36	Cocoon Smart HVAC Fan	0.9854	0.9812	0.9833
37	GoSund Smart Plug WP3 (1)	0.9753	0.9858	0.9805
38	DCS8000LHA1 D-Link Mini Camera	0.9765	0.9506	0.9634
39	HeimVision SmartLife Radio/Lamp	0.9720	0.9871	0.9795
40	LampUX RGB	0.9794	0.9919	0.9856
41	Levoit Air Purifier	0.9749	0.9846	0.9797
42	SmartThings Hub	0.9896	0.9941	0.9918
43	AeoTec Smart Home Hub	0.9936	0.9888	0.9912
44	iRobot Roomba	0.9741	0.9798	0.9769

Continued on next page

Table 7.2 – continued from previous page

Sr.	Device	Precision	Recall	F_1 Score
45	LIFX Lightbulb	0.9649	0.9442	0.9544
46	Smart Board	0.9239	0.7364	0.8196
47	Wemo smart plug 2	0.9438	0.9329	0.9383

Despite the overall strong performance of the DFNN, certain device types exhibited comparatively lower classification metrics, indicating challenges in distinguishing these classes from others. For example, the Teckin Plug 1 attained a precision of 0.8732, a recall of 0.7353, and an F1 score of 0.7984. Similarly, the Yutron Plug 2 achieved a precision of 0.7746, a recall of 0.9006, and an F1 score of 0.8329. The Smart Board also demonstrated reduced performance, with a precision of 0.9239, a recall of 0.7364, and an F1 score of 0.8196. These comparatively lower scores suggest that the model encounters difficulty in learning sufficiently discriminative representations for these specific device types, potentially due to overlapping behavioral patterns, feature similarity, or limited intra-class variability within the dataset.

7.2 Device identification on CU2025 dataset.

The performance of the DFNN on the Concordia University IoT Device Identification Dataset (CU2025) further demonstrates its effectiveness in fine-grained device-type classification. Several devices—including the Amazon Echo Pop, RainPoint WiFi Water Timer, Smart Lock Gateway, and WiFi Smart Camera 2—achieved a perfect F1 score of 1.0000, indicating flawless classification. Performance across the dataset is highly consistent, with the lowest F1 score recorded at 0.9996 for the Blink Sync Module and the Ring Chime. The precision, recall, and F1 score for each device are comprehensively reported in Table 7.3. The DFNN exhibits exceptionally strong classification capabilities, and the corresponding confusion matrix for device-type classification using the CU2025 dataset is shown in Figure 7.3.

Figure 7.2: Confusion matrix for device-type identification.

Table 7.3: Device classification result on CU2025 dataset.

Sr.	Device Names	Precision	Recall	F_1 Score
1	Amazon Echo Dot (3rd Gen)	0.9998	0.9999	0.9999
2	Ring Battery Doorbell	0.9998	0.9998	0.9998
3	Blink Video Doorbell	1.0000	0.9999	0.9999
4	Blink Outdoor Camera	1.0000	0.9999	0.9999
5	Blink Sync Module	0.9996	1.0000	0.9996
6	Chime Ring	0.9998	0.9995	0.9996
7	Google Nest Hub	0.9999	0.9998	0.9998
8	Amazon Echo Pop	1.0000	1.0000	1.0000
9	Google Nest Mini (2nd Gen)	0.9998	0.9998	0.9998
10	Goove Life Thermo-Hygrometer	0.9998	0.9998	0.9998
11	Rain Point WiFi Water Timer	1.0000	1.0000	1.0000
12	Ring StickUp Camera	0.9999	0.9999	0.9999
13	Smart Lock Gateway	1.0000	1.0000	1.0000
14	Wyze Camera Pan	0.9999	0.9999	0.9999
15	YI 1080p Home Camera	0.9999	0.9998	0.9998
16	Wifi Smart Camera	1.0000	0.9999	0.9999
17	Wifi Smart Camera2	0.9999	1.0000	1.0000
18	Treat Life Wifi Smart Light	0.9996	1.0000	0.9998
19	Kasa Smart WiFi Plug	1.0000	0.9998	0.9999
20	Camera Blink Mini 2	1.0000	0.9997	0.9998
21	Tapo Pan/Tilt Home Security WiFi Camera	0.9998	1.0000	0.9999

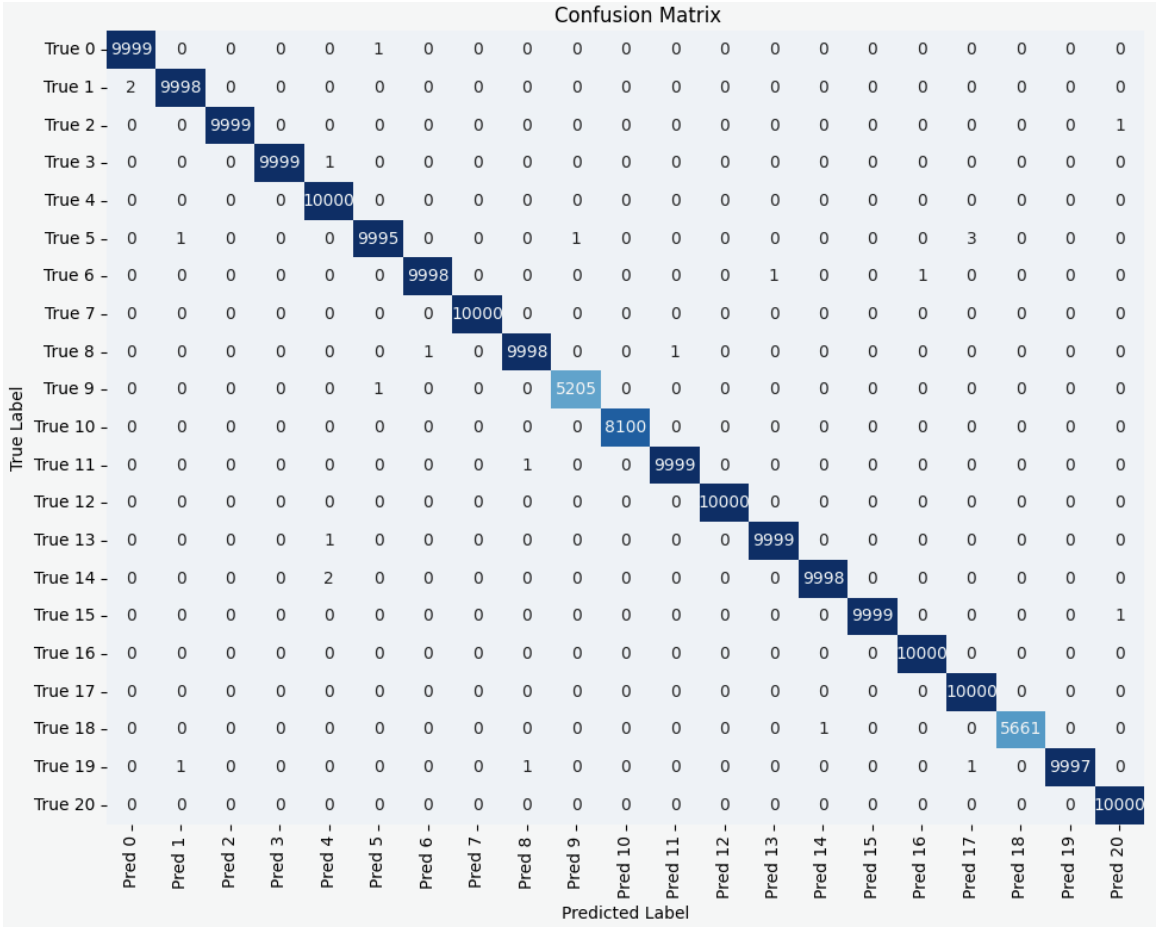


Figure 7.3: Confusion matrix for device-type identification.

Figure 7.4: The impact of big batch sizes on model accuracy.

7.3 Incremental learning

The incremental learning component of this study evaluates the DFNN's capacity to incorporate newly introduced device types following the initial training phase. Specifically, we examine the impact of varying incremental batch sizes on overall model accuracy, with the goal of establishing a baseline for assessing the effectiveness of the proposed incremental learning strategy. The base model was trained using data from 10 devices, after which additional devices were introduced in incrementally sized batches.

Two incremental learning configurations were investigated. In the first configuration, five additional device types were added in two successive batches. In the second configuration, new devices were introduced across four smaller batches consisting of 2, 2, 3, and 3 devices, respectively. In both cases, Batch 1 refers to the base model, while subsequent batches represent incremental updates.

Figure 7.4 presents the results of the first configuration, where the base model, trained on the initial 10 devices, achieved an accuracy of 99.92%. After two incremental additions of five devices each, the accuracy decreased to 90.15%. This outcome highlights the impact of introducing a

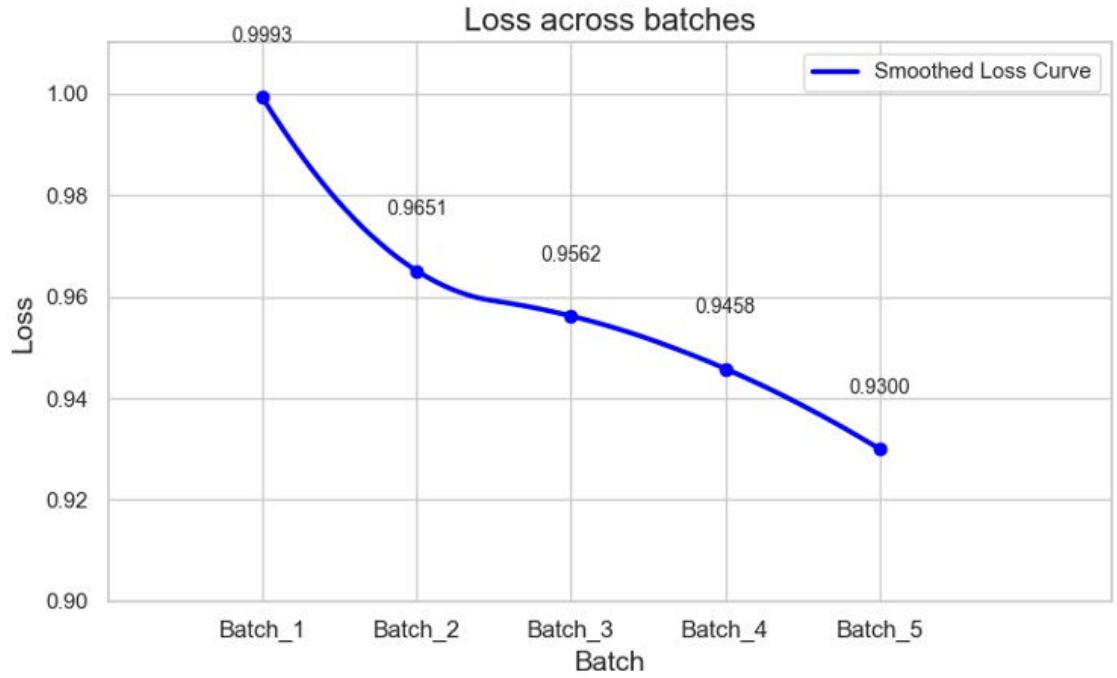


Figure 7.5: The impact of small batch sizes on model accuracy.

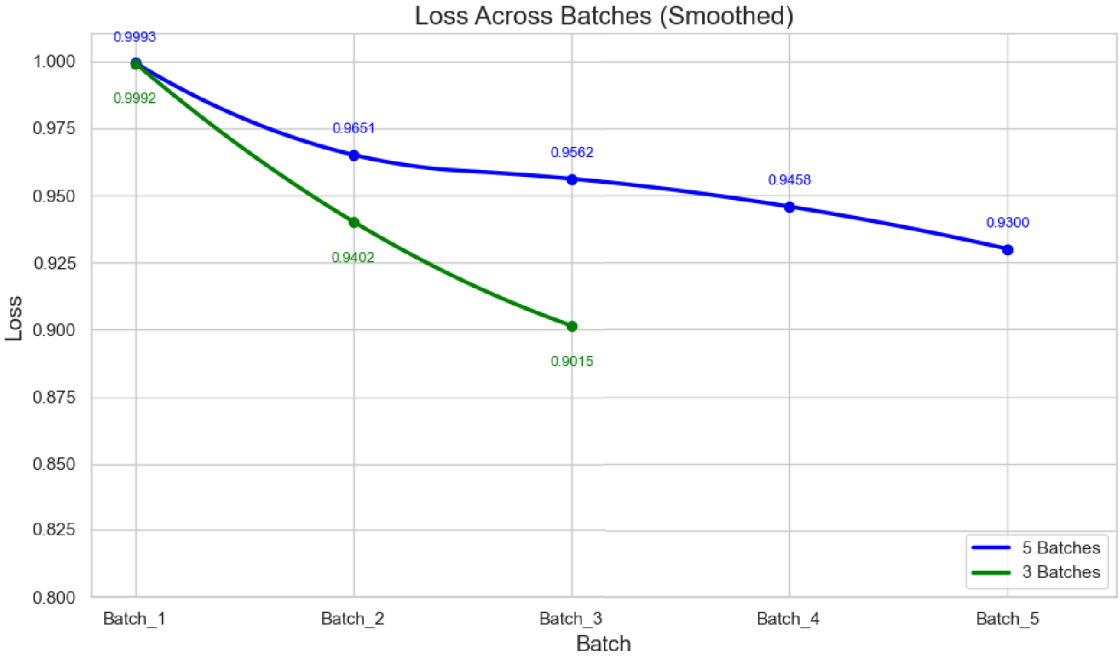


Figure 7.6: The impact of varying batch sizes on model accuracy.

relatively large number of devices per incremental step.

Similarly, Figure 7.5 illustrates the performance of the second configuration, which employed five batches: the initial batch of 10 devices, followed by batches of 2, 2, 3, and 3 devices. In this case, the accuracy declined from 99.93% to 93% after the addition of four incremental batches, demonstrating that smaller, more frequent increments yield comparatively less degradation.

A combined comparison of both configurations is depicted in Figure 7.6. The blue curve represents the four-batch incremental strategy involving smaller batch sizes, while the green curve corresponds to the two-batch strategy with larger increments. Although both approaches ultimately incorporated the same number of new devices, 10 additional devices on top of the original 10, the resulting accuracy trajectories differ slightly.

As shown in Figure 7.6, the model's accuracy declines to 90.15% when larger batches are used, whereas the degradation is limited to 93% when devices are introduced gradually across smaller incremental batches. These results indicate that the DFNN is capable of effectively integrating new device types in an incremental learning setting, with only moderate performance degradation. Moreover, they demonstrate that finer-grained incremental updates mitigate accuracy loss relative to larger, less frequent updates.

7.4 Leave-One-Subject-Out Cross-Validation (LOSO-CV)

The devices listed in Table 7.4 were selected for the Leave-One-Subject-Out Cross-Validation (LOSO-CV) experiment, with only device types that had multiple instances included in the evaluation. The requirement for duplicate devices arises from the structure of LOSO-CV itself: when one physical device is held out as the test subject, at least one other device of the same class must remain in the training set to ensure that the model can learn the characteristic feature patterns associated with that device type. If a class is represented by only a single physical device, removing it during the LOSO process would eliminate all training samples for that class, preventing the model from learning a meaningful representation and rendering the evaluation invalid for that device type. Moreover, the presence of duplicate devices allows the LOSO-CV procedure to capture natural variations that may occur across different physical units of the same model, such as differences arising

from manufacturing tolerances, firmware updates, or environmental conditions, thereby providing a more realistic measure of the model's ability to generalize to unseen devices of the same type. Under this configuration, the DFNN achieved an overall LOSO-CV accuracy of 98.40%, demonstrating strong generalization capability across replicated device types. The detailed per-device identification results are presented in Table 7.5.

Table 7.4: Devices considered for LOSO cross-validation.

Sr.	MAC address	Device Names	Device Model
1	A4:86:DB:5C:1F:D1	WiFi Smart Camera	T-CP8082LF-W3M
2	A4:86:DB:59:B6:C2	WiFi Smart Camera	T-CP8082LF-W3M
3	A4:86:DB:84:DB:4C	WiFi Smart Camera	T-CP8050LF-W3M
4	A4:86:DB:84:DB:86	WiFi Smart Camera	T-CP8050LF-W3M
5	10:D5:61:05:AC:29	Treat Life WiFi Smart Light	SL20
6	10:D5:61:05:E8:04	Treat Life WiFi Smart Light	SL20
7	20:23:51:9A:C1:17	Kasa Smart WiFi Plug	HS103
8	20:23:51:9A:BE:7A	Kasa Smart WiFi Plug	HS103
9	9C:C8:E9:C2:89:76	Camera Blink Mini 2	BCM00700U
10	68:13:F3:1D:02:11	Camera Blink Mini 2	BCM00700U
11	3C:64:CF:BE:EC:C2	Tapo Pan/Tilt Home Security WiFi Camera	Tapo C210
12	3C:64:CF:BE:ED:F9	Tapo Pan/Tilt Home Security WiFi Camera	Tapo C210

Table 7.5: LOSO cross-validation results.

Sr.	MAC address	Device Names	Accuracy
1	A4:86:DB:5C:1F:D1	WiFi Smart Camera	1.0000
2	A4:86:DB:59:B6:C2	WiFi Smart Camera	0.9999
3	A4:86:DB:84:DB:4C	WiFi Smart Camera	1.0000

Continued on next page

Table 7.5 – continued from previous page

Sr.	MAC address	Device Names	Accuracy
4	A4:86:DB:84:DB:86	WiFi Smart Camera	1.0000
5	10:D5:61:05:AC:29	Treat Life WiFi Smart Light	0.9999
6	10:D5:61:05:E8:04	Treat Life WiFi Smart Light	1.0000
7	20:23:51:9A:C1:17	Kasa Smart WiFi Plug	0.9914
8	20:23:51:9A:BE:7A	Kasa Smart WiFi Plug	0.9842
9	9C:C8:E9:C2:89:76	Camera Blink Mini 2	0.9974
10	68:13:F3:1D:02:11	Camera Blink Mini 2	0.9986
11	3C:64:CF:BE:EC:C2	Tapo Pan/Tilt Home Security WiFi Camera	0.9248
12	3C:64:CF:BE:ED:F9	Tapo Pan/Tilt Home Security WiFi Camera	0.9116

Chapter 8

Discussion

8.1 Effectiveness of DFNN

The results highlight the effectiveness of the DFNN model in accurately classifying both device-categories and device-types, demonstrating its suitability for practical deployment. The model's performance across different labels and datasets further affirms its generalizability. The near perfect F1 scores observed in both device-category and device-type classification underscore the discriminative power of the selected features, which enable the model to effectively distinguish between classes. This strong performance indicates good class separability within the feature space, as well as a well-balanced data distribution during training.

The DFNN exhibits remarkable performance in device-category classification, as illustrated in Figure 7.1 and detailed in Table 7.1. Despite this overall effectiveness, the model showed reduced classification accuracy for a few devices, including Teckin Plug 1, Smart Board, and Yutron Plug 2, as shown in Table 7.2. This reduced performance can be attributed primarily to class imbalance within the CICIoT2023 dataset; limited samples of these devices resulted in weaker representations during training. Machine learning models are known to be biased toward classes with higher sample sizes in imbalanced datasets [146]. Additionally, feature similarity across different device classes contributed to misclassifications. Lower recall scores for certain devices indicate that the model occasionally confuses instances with other classes, emphasizing the need for larger and more balanced datasets. Frameworks such as ScanIoT and DroidScour play a critical role in this regard,

enabling systematic collection of sufficient traffic samples for each device-type, thereby supporting the development of more robust and accurate classification models.

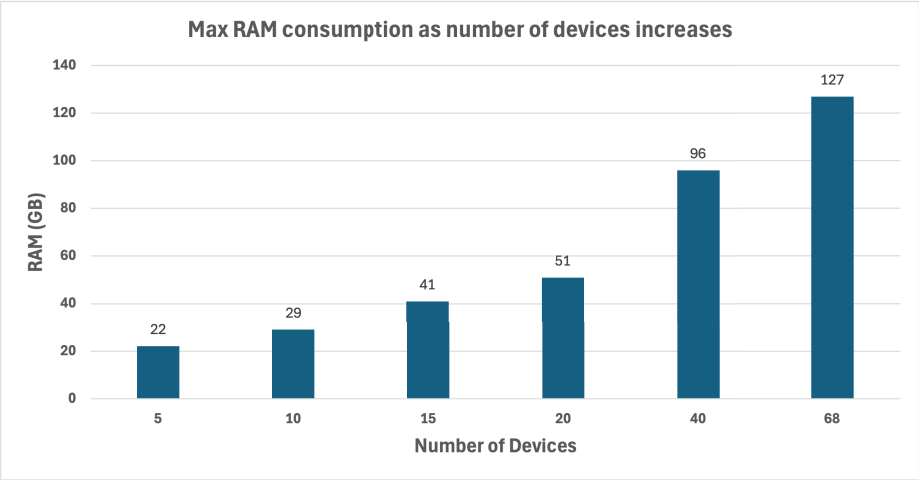
Even with a limited number of samples for certain devices, the DFNN demonstrated high reliability and robustness in classification tasks. The model's strong performance, as presented in Figure 7.1 and Table 7.1, highlights its ability to accurately classify device categories, despite significant class imbalance resulting from the aggregation of multiple devices into broader categories. This indicates that the model's predictions remain practical and reliable for real-world deployments, though a bias toward devices with larger sample sizes remains observable.

It is important to note that the DFNN model imposes considerable computational demands. System resources, particularly RAM, increase proportionally with the number of devices and corresponding data samples during training, as shown in Figure 8.1a. Similarly, training time grows with the addition of more devices, as illustrated in Figure 8.1b. To further assess resource requirements, Figure 8.1 presents a combined analysis of the CICIoT2023 and CU2025 datasets, which expands the total number of IoT devices under evaluation. These results indicate that, while deep learning models such as the DFNN achieve high accuracy in IoT device identification, this performance is associated with increased computational cost, highlighting the inherent trade-off between model effectiveness and resource consumption.

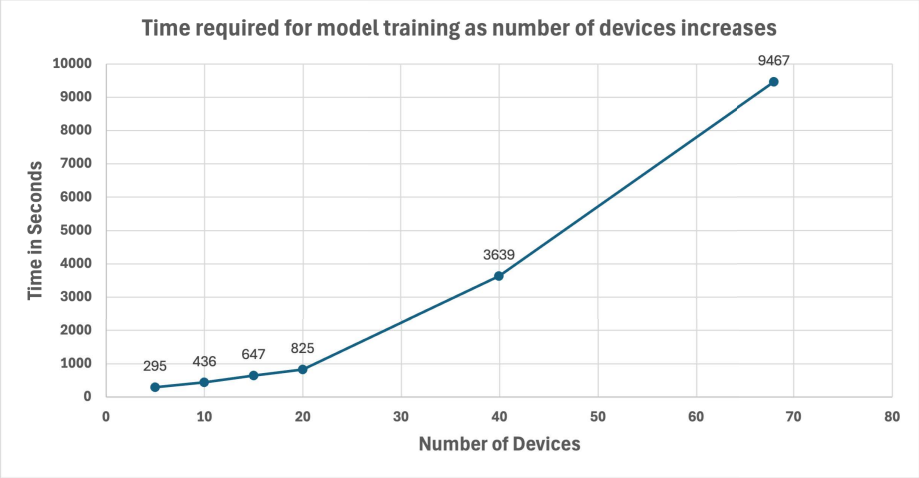
8.2 Need for traffic capturing frameworks

Since the performance of most classifiers is hindered by dataset imbalance, the existing datasets present significant challenges, prompting the proposal of the CU2025 dataset. The CU2025 dataset was collected with careful attention to ensuring sufficient sample representation from each IoT device, aimed at optimizing classifier performance. The proposed frameworks, ScanIoT and DroidScour, provide research and smart home users an opportunity to gather a scalable dataset for device identification. Both the laboratory frameworks allow the user to predefine the number of packets to be captured for the corresponding devices. The packet-capturing process terminates only when the required number of packets is collected by the tool or interrupted with a stop action by the users.

It is crucial to mention that not all IoT devices generate the same volume of traffic. Some of



(a) DFNN Resource Consumption



(b) DFNN training time

Figure 8.1: Resource consumption of the DFNN model.

the IoT devices may require human interaction to generate any traffic in the first place. It is evident that the IoT devices like camera generates a sheer volume of traffic, whereas the temperature sensor and carbon monoxide sensors would generate the least amount of traffic. These limitations are not considered in the collection of devices for IoT device identification problems. To meet the required samples, research mostly follows the oversampling [147] and undersampling [148] to handle the imbalanced dataset. These approaches often lead to information loss and issues such as underfitting and overfitting, which are further amplified in minority classes with limited samples due to the lack of new or diverse information.

The results of the DFNN model on the Concordia University IoT Device Identification Dataset

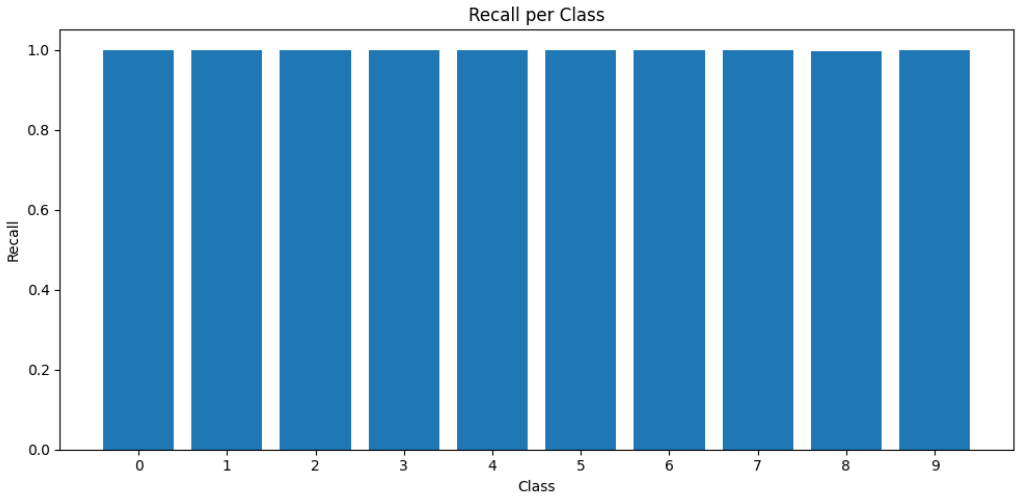
(CU2025) demonstrate its effectiveness on a balanced dataset, as shown in Table 7.3. Even the lowest performance, with an F1 score of 0.9996, is notably high and demonstrates the robustness of the model. Several devices achieved a perfect F1 score of 1.0000. A comparison between the results on the CICIoT2023 dataset and the CU2025 dataset highlights the impact of dataset imbalance. All machine learning models exhibit a bias toward devices with a higher number of samples. The best-performing devices on the CICIoT2023 dataset, as shown in Table 7.2, correspond to those with the highest number of samples, as indicated in Table 3.1. In contrast, the results from the CU2025 dataset exhibit greater consistency, with minimal variation observed in the classification performance across devices.

While a balanced dataset is generally recommended, it is important to recognize that not all IoT devices generate the same volume of network traffic. For instance, camera devices typically produce a large amount of traffic, whereas other IoT devices, such as smart plugs or sensors, generate significantly less. Some devices, like toxin alarm sensors, transmit traffic only under specific conditions. To ensure sufficient sample collection across all device types, frameworks such as ScanIoT and DroidSource are essential, as they continuously capture traffic until a minimum number of samples is acquired. In machine learning, devices with a large number of samples may lead to overfitting, while those with limited samples risk underfitting, highlighting the need for careful dataset construction and sampling strategies.

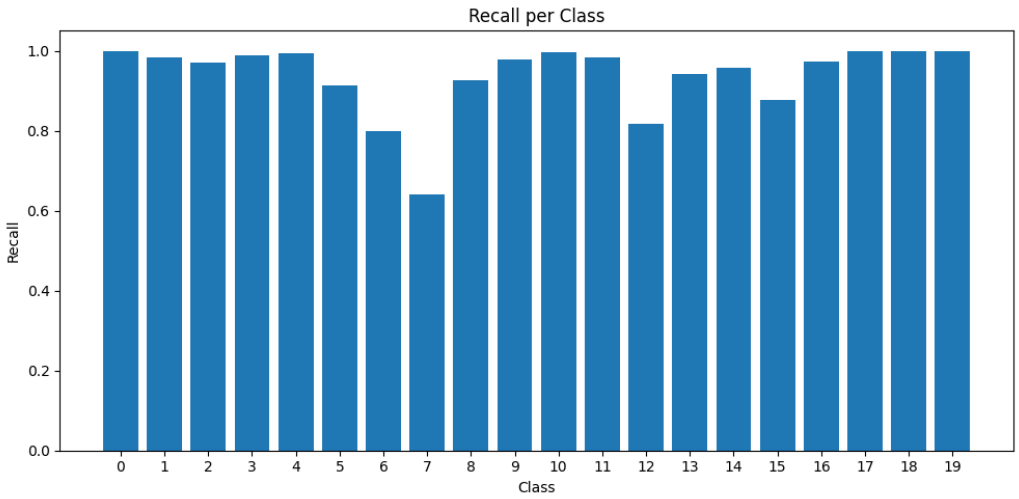
8.3 Incremental learning

In the context of incremental learning, the results indicate that introducing a large number of devices within a single batch leads to greater degradation in classifier performance compared to smaller, more frequent incremental batches. This decline is primarily due to the substantial disruption caused by incorporating many new classes simultaneously, which exacerbates catastrophic forgetting by making it increasingly challenging to maintain a balanced representation of previously learned and newly introduced samples.

Despite this performance reduction, the incremental learning results demonstrate the effectiveness of the proposed incremental DFNN implementation. While some decrease in classifier accuracy is expected as new devices are added, the model remains sufficiently accurate for practical device identification. Nevertheless, detailed classification reports and overall accuracy trends suggest that full retraining may become necessary over successive increments. To manage this, a predefined threshold, based on either the recall score of individual devices or overall classifier accuracy, can be established to trigger a complete retraining of the model using all available devices, thereby refreshing the base model and maintaining consistent performance in an incremental learning scenario.



(a) Base model batch 1



(b) Incremental learning batch 5

Figure 8.2: Per-device recall score in incremental learning.

Monitoring per-device recall provides a practical mechanism for deciding when retraining is required. Recall, which measures the model's ability to correctly identify actual positive instances, can indicate underperforming devices that may require additional training samples. For example, Figure 8.2 illustrates this approach: Figure 8.2a shows the recall scores for devices after training the base model (Batch 1), while Figure 8.2b depicts the recall scores for each device following Batch 5. Such monitoring enables the establishment of device-specific or overall thresholds to inform retraining decisions, ensuring sustained accuracy in incremental learning deployments.

8.4 Essence of LOSO-CV

To further evaluate the effectiveness of DFNN for device identification, Leave-One-Subject-Out Cross-Validation (LOSO-CV) has been employed. LOSO-CV is used to assess the model's ability to generalize to unseen devices in real-world scenarios. It is a preferred form of cross-validation, as it minimizes data leakage by ensuring that the training and testing sets do not share temporal or behavioral similarities. Although LOSO-CV is computationally intensive, it provides strong evidence of the model's generalization capability in practical deployments.

8.5 IoT revolution impact on categorization

The rapid expansion of the Internet of Things (IoT) ecosystem has fundamentally challenged traditional frameworks for device categorization. Earlier generations of networked devices were functionally constrained, allowing clear distinctions between categories such as sensors, cameras, and actuators. In contrast, contemporary IoT devices increasingly integrate heterogeneous sensing, communication, and processing modules, resulting in multifunctional behavior that spans multiple conventional categories. Additionally, cloud-centric architectures, continuous firmware updates, and software-defined feature extensions enable devices to acquire new capabilities over time, further blurring category boundaries. Consequently, classification based solely on a single dominant function no longer accurately reflects operational behavior. This evolution has driven a shift towards device-specific identification and behavioral modeling, approaches that provide a more precise representation of the complex and dynamic functionality exhibited by modern IoT systems.

8.6 Critical features for future identification frameworks

While device identification has been the primary focus of many research efforts, recent work such as the study in [123] has shown that several proposed models inadvertently learn network-specific characteristics rather than device-specific features. As a result, their performance degrades significantly when evaluated on datasets collected from different network environments. It is therefore essential to ensure that device identification models learn features that are independent of network conditions and truly representative of device behavior.

In this study, we incorporated two datasets collected from networks with differing levels of complexity to validate the robustness of our approach under varying conditions. Ensuring network-agnostic feature learning remains a critical requirement for future device identification techniques. Moreover, the adoption of evaluation strategies such as Leave-One-Subject-Out Cross-Validation (LOSO-CV) is important, as it enables assessment of a model's ability to generalize to previously unseen device instances. The CU2025 dataset is specifically designed to support such evaluations.

Chapter 9

Conclusion, Limitation, and Future work

9.1 Conclusion

The widespread integration of Internet of Things (IoT) devices across smart homes, smart cities, healthcare, industry, agriculture, transportation, and commercial sectors has made them indispensable components of modern digital infrastructures. However, their limited computational capabilities and reliance on weak or lightweight cryptographic mechanisms leave many IoT devices highly vulnerable to cyber threats. Ensuring their security therefore relies fundamentally on accurate device identification, which enables effective access control, intrusion prevention, and enforcement of network policies.

This thesis addresses the challenges associated with IoT device identification by critically examining the limitations of existing datasets and introducing a more realistic and balanced alternative—the Concordia University IoT Device Identification Dataset (CU2025). CU2025 incorporates pairs of identical device models operated under real human interaction, providing a practical basis for evaluating a model's ability to generalize to unseen device instances. Sufficient traffic samples were collected for each device to ensure dataset balance and quality.

To support high-quality dataset generation, two traffic-capture frameworks, ScanIoT and DroidScour, were designed and implemented. These frameworks eliminate the need for complex network

configurations by offering automated and efficient mechanisms for collecting device-specific traffic in real smart home environments. Both frameworks continuously capture traffic until an adequate number of samples is obtained, enabling their use not only for device identification but also for behavioral or security-focused analysis. ScanIoT provides an independent setup for traffic collection and annotation, allowing users to gather and share datasets, though it requires specific hardware components. In contrast, DroidScour leverages an Android application that automatically uploads captured traffic via Firebase integration, enabling rapid and seamless dataset sharing without additional equipment.

A Deep Feedforward Neural Network (DFNN) model was developed and rigorously evaluated across two datasets. On the CICIoT2023 dataset, the model achieved 99.73% accuracy in device-type identification and 99.97% accuracy in device-category classification. On the CU2025 dataset, the DFNN attained an even higher device-type identification accuracy of 99.99%, underscoring the value of a well-balanced and high-quality dataset in improving classifier performance. To accommodate the natural expansion of smart home ecosystems, an incremental learning framework was introduced, allowing the model to incorporate new devices without the need for complete retraining. The model sustained strong performance across incremental updates, with accuracy remaining at 90.15% after two large increments and 93% when smaller increments were applied. These findings demonstrate that introducing devices in smaller increments minimizes performance degradation compared to adding many devices simultaneously.

To further evaluate real-world applicability, the model was assessed using Leave-One-Subject-Out Cross-Validation (LOSO-CV), achieving an accuracy of 98.40%. This result highlights the robustness and generalization capability of the proposed approach in identifying previously unseen devices under realistic and variable operating conditions.

In conclusion, this thesis delivers a comprehensive and effective solution for IoT device identification through the creation of realistic dataset, the development of versatile traffic-capture frameworks, and the design of a high-performing deep learning model with incremental adaptability. Together, these contributions advance the state of IoT security and support the development of more resilient, adaptive, and trustworthy smart home environments.

9.2 Limitation

Although the proposed deep learning model demonstrates strong performance across the evaluated datasets, the feature selection process may require further refinement as the number and diversity of IoT devices increase. While the model was tested on both the CICIoT2023 and CU2025 datasets, and even on a combined version of the two, where it continued to perform effectively, the scope of available public datasets remains limited. In real-world environments, thousands of IoT device models exist, many of which are not yet publicly accessible for research or dataset construction. Ensuring that the selected features remain robust and discriminative across such a wide device spectrum is therefore essential, especially while avoiding features that inadvertently capture network-specific characteristics or deployment complexities.

Furthermore, the scarcity of publicly available, well-annotated IoT datasets poses an additional constraint. To address this challenge, broader adoption of the developed ScanIoT and DroidScour frameworks is encouraged, as their use for traffic collection and annotation can facilitate the creation of diverse, high-quality datasets. Expanding the range of available datasets will be critical for advancing IoT device identification research and improving model generalizability to the large variety of devices encountered in practical deployments.

9.3 Future work

As part of future work, we plan to continue collecting and annotating network traffic from a broader and more diverse set of IoT devices. The expanded dataset will also be made publicly available and will maintain the principles of balance and realism established in the CU2025 dataset, with traffic generated through genuine human interactions to capture authentic device behaviors. Although the primary aim of this dataset is to support IoT device identification research, its richness and variability also make it well-suited for device behavior analysis and other IoT security studies.

We encourage the research community to adopt the proposed traffic-collection frameworks, ScanIoT and DroidScour, to develop scalable, interaction-driven datasets that more accurately reflect real-world IoT usage patterns. Both frameworks are flexible and can be used not only for

device identification but for any application requiring balanced, device-specific data collection. Additionally, while we adapted the CICIoT2023 dataset for device identification, the same frameworks can be used to collect benign traffic from IoT devices prior to incorporating attack traffic, enabling the creation of datasets suitable for intrusion detection, anomaly detection, or behavioral profiling.

Finally, although the CU2025 dataset has primarily been utilized for device identification in this thesis, it also holds significant potential for broader applications. Its structure and granularity make it valuable for device behavior analysis, enabling future research in areas such as activity recognition, pattern modeling, device state inference, and security analytics.

Appendix A

List of Publications

The following publications have resulted from the research conducted in this thesis.

- A. F. Jamali and C. Fung, ”Demo-ScanIoT, an Application to collect IoT dataset for HomeGuard,” 2024 7th Conference on Cloud and Internet of Things (CIoT), Montreal, QC, Canada, 2024, pp. 1-2, doi: 10.1109/CIoT63799.2024.10757166.

- A. F. Jamali, D. Rostami and C. Fung, ”IoT Device Identification using Deep Learning,” 2025 16th International Conference on Network of the Future (NoF), Montreal, QC, Canada, 2025, pp. 46-54, doi: 10.1109/NoF66640.2025.11223318.

- Schussler, B.S., Jamali, A., Cordeiro, W. and Fung, C., DroidScour: an Android Application for IoT Device Data Collection. 2025 21st International Conference on Network and Service Management (CNSM).

Appendix B

Credit to contributors

Two students provided valuable contributions to this research, as detailed below.

- Dorreen Rostami was responsible for the development of the mobile application component of the ScanIoT tool. Although the ScanIoT system comprises backend services, a web application, and a mobile application, only the mobile application module was completed by Dorreen Rostami as part of the COMP6971 project.

- Inspired by the ScanIoT tool, Brenda Schussler, participating in the project through the MITACS program, developed the DroidScour Android application under the academic supervision of Abdul Fareed Jamali. In addition to leading the development of the application, Brenda also made a substantial contribution to the data collection process. The contributions to the DroidScour application are as follows: Abdul Fareed Jamali contributed to the conceptualization, methodology, formal analysis, and supervision, while Brenda Schussler contributed to the methodology, software development, investigation, data curation, and visualization.

Although they contributed to the overall project, a substantial amount of time and effort was required on my part to familiarize them with the domain and to transfer the necessary technical knowledge. This guidance was essential to ensure that they were able to effectively understand the project requirements and successfully complete their respective components. All code developed as part of this work has been made publicly available through the respective contributors' GitHub profiles to

ensure proper credit and recognition for their efforts and contributions to the project.

Bibliography

[1] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, "Internet of things market analysis forecasts, 2020–2030," in *2020 Fourth World Conference on smart trends in systems, security and sustainability (WorldS4)*, pp. 449–453, IEEE, 2020.

[2] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pp. 2177–2184, IEEE, 2017.

[3] 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," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[4] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 2002.

[5] T. N. Theis and H.-S. P. Wong, "The end of moore's law: A new beginning for information technology," *Computing in science & engineering*, vol. 19, no. 2, pp. 41–50, 2017.

[6] Y. Kim, Y. Park, and J. Choi, "A study on the adoption of iot smart home service: using value-based adoption model," *Total Quality Management & Business Excellence*, vol. 28, no. 9-10, pp. 1149–1165, 2017.

[7] J. Kaur, Jaskaran, N. Sindhwani, R. Anand, and D. Pandey, "Implementation of iot in various domains," in *IoT based smart applications*, pp. 165–178, Springer, 2022.

[8] R. R. Chowdhury and P. E. Abas, "A survey on device fingerprinting approach for resource-constraint iot devices: Comparative study and research challenges," *Internet of Things*, vol. 20, p. 100632, 2022.

[9] C. Saadouni, S. El Jaouhari, N. Tamani, S. Ziti, L. Mroueh, and K. El Bouchti, "Identification techniques in the internet of things: Survey, taxonomy and research frontier," *IEEE Communications Surveys & Tutorials*, 2025.

[10] S. Kim, M. Park, S. Lee, and J. Kim, "Smart home forensics—data analysis of iot devices," *Electronics*, vol. 9, no. 8, p. 1215, 2020.

[11] V. Sivaraman, H. H. Gharakheili, C. Fernandes, N. Clark, and T. Karliychuk, "Smart iot devices in the home: Security and privacy implications," *IEEE Technology and Society Magazine*, vol. 37, no. 2, pp. 71–79, 2018.

[12] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[13] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[14] S. Vetrivel, B. Bouwmeester, M. van Eeten, and C. H. Gañán, "{IoT} market dynamics: An analysis of device sales, security and privacy signals, and their interactions," in *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 7031–7048, 2024.

[15] "Internet of things (iot) market size, share & industry analysis, by component (platform and solution & services), by deployment (on-premise and cloud), by enterprise type (smes and large), by industry (bfsi, retail, government, healthcare, manufacturing, agriculture, sustainable energy, transportation, it & telecom, and others), and regional forecast, 2024-2032." https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307, 2025.

[16] E. Zeng and F. Roesner, "Understanding and improving security and privacy in {multi-user} smart homes: A design exploration and {in-home} user study," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 159–176, 2019.

[17] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, p. 109116, 2022.

[18] J. Manzoor, A. Waleed, A. F. Jamali, and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source siem solutions for smes," *Plos one*, vol. 19, no. 3, p. e0301183, 2024.

[19] Z. B. Celik, G. Tan, and P. D. McDaniel, "Iotguard: Dynamic enforcement of security and safety policy in commodity iot.," in *NDSS*, 2019.

[20] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, pp. 506–509, 2017.

[21] Y. Wang, B. P. Rimal, M. Elder, S. I. C. Maldonado, H. Chen, C. Koball, and K. Ragothaman, "Iot device identification using supervised machine learning," in *2022 IEEE international conference on consumer electronics (ICCE)*, pp. 1–6, IEEE, 2022.

[22] A. F. Jamali and C. Fung, "Scaniot, an application to collect iot dataset for homeguard," in *2024 7th Conference on Cloud and Internet of Things (CIoT)*, pp. 1–2, IEEE, 2024.

[23] A. F. Jamali, "Fareed-jamali/scaniot: Scaniot is a tool designed to collect traffic." https://github.com/Fareed-Jamali/ScanIoT, Jun 2025.

[24] B. Schussler, "Brendaschussler/scan_iot_android." https://github.com/brendaschussler/scan_iot_android, 2025.

[25] B. Schussler, "Brendaschussler/droidscour-android-app: Android app for scanning hotspot-connected devices and capturing network traffic.." https://github.com/brendaschussler/DroidScour-Android-App, 2025.

[26] B. Danev, D. Zanetti, and S. Capkun, "On physical-layer identification of wireless devices," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–29, 2012.

[27] J. Zhang, S. Rajendran, Z. Sun, R. Woods, and L. Hanzo, "Physical layer security for the internet of things: Authentication and key generation," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 92–98, 2019.

[28] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEe Access*, vol. 7, pp. 82721–82743, 2019.

[29] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE communications surveys & tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[30] M. Abbasi, M. Plaza-Hernandez, J. Prieto, and J. M. Corchado, "Security in the internet of things application layer: requirements, threats, and solutions," *IEEE Access*, vol. 10, pp. 97197–97216, 2022.

[31] L. Meddahi, A. Meddahi, P. Sondi, and F. Zhou, "Leveraging blockchain for a robust and scalable device identification in lorawan," in *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, IEEE, 2023.

[32] O. O. Kazeem, O. O. Akintade, L. O. Kehinde, O. Akintade, and L. Kehinde, "Comparative study of communication interfaces for sensors and actuators in the cloud of internet of things," *Int. J. Internet Things*, vol. 6, no. 1, pp. 9–13, 2017.

[33] S. J. Ramson, S. Vishnu, and M. Shanmugam, "Applications of internet of things (iot)–an overview," in *2020 5th international conference on devices, circuits and systems (ICDCS)*, pp. 92–95, IEEE, 2020.

[34] M. Anjum, M. A. Khan, S. A. Hassan, A. Mahmood, H. K. Qureshi, and M. Gidlund, "Rssi fingerprinting-based localization using machine learning in lora networks," *IEEE Internet of Things Magazine*, vol. 3, no. 4, pp. 53–59, 2021.

[35] M. Azrour, J. Mabrouki, A. Guezzaz, and A. Kanwal, "Internet of things security: challenges and key issues," *Security and Communication Networks*, vol. 2021, no. 1, p. 5533843, 2021.

[36] K. Shaukat, T. M. Alam, I. A. Hameed, W. A. Khan, N. Abbas, and S. Luo, "A review on security challenges in internet of things (iot)," in *2021 26th international conference on automation and computing (ICAC)*, pp. 1–6, IEEE, 2021.

[37] K. M. Hou, X. Diao, H. Shi, H. Ding, H. Zhou, and C. de Vaulx, "Trends and challenges in aiot/iiot/iot implementation," *Sensors*, vol. 23, no. 11, p. 5074, 2023.

[38] O. Peter, A. Pradhan, and C. Mbohwa, "Industrial internet of things (iiot): opportunities, challenges, and requirements in manufacturing businesses in emerging economies," *Procedia Computer Science*, vol. 217, pp. 856–865, 2023.

[39] M. S. Farooq, O. O. Sohail, A. Abid, and S. Rasheed, "A survey on the role of iot in agriculture for the implementation of smart livestock environment," *IEEe Access*, vol. 10, pp. 9483–9505, 2022.

[40] G. Virupaxappa and S. Thangam, "Smart agriculture and role of iot," in *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 1–6, IEEE, 2021.

[41] A. H. A. Hussein, K. A. Jabbar, A. Mohammed, and L. Jasim, "Harvesting the future: Ai and iot in agriculture," in *E3S Web of Conferences*, vol. 477, p. 00090, EDP Sciences, 2024.

[42] P. Whig, A. Velu, R. R. Nadikattu, and Y. J. Alkali, "Role of ai and iot in intelligent transportation," in *artificial intelligence for future intelligent transportation*, pp. 199–220, Apple Academic Press, 2024.

[43] H. Nguyen, D. Nawara, and R. Kashef, "Connecting the indispensable roles of iot and artificial intelligence in smart cities: A survey," *Journal of Information and Intelligence*, vol. 2, no. 3, pp. 261–285, 2024.

[44] P. H. Rettore, J. Mast, T. Aurisch, A. C. Viana, P. Sevenich, and B. P. Santos, "Military iot from management to perception: Challenges and opportunities across layers," *IEEE Internet of Things Magazine*, vol. 8, no. 2, pp. 25–31, 2025.

[45] S. Krishnamoorthy, A. Dua, and S. Gupta, "Role of emerging technologies in future iot-driven healthcare 4.0 technologies: A survey, current challenges and future directions," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 1, pp. 361–407, 2023.

[46] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling iot based healthcare applications," *IEEE Access*, vol. 9, pp. 38859–38890, 2021.

[47] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine learning for the detection and identification of internet of things devices: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 298–320, 2021.

[48] W. Fei, H. Ohno, and S. Sampalli, "A systematic review of iot security: Research potential, challenges, and future directions," *ACM computing surveys*, vol. 56, no. 5, pp. 1–40, 2023.

[49] M. Baláž, K. Kováčiková, A. Novák, and J. Vaculík, "The application of internet of things in air transport," *Transportation Research Procedia*, vol. 75, pp. 60–67, 2023.

[50] R. G. Saadé, J. Zhang, X. Wang, H. Liu, and H. Guan, "Challenges and opportunities in the internet of intelligence of things in higher education—towards bridging theory and practice," *IoT*, vol. 4, no. 3, pp. 430–465, 2023.

[51] A. Salam, "Internet of things for sustainable forestry," in *Internet of Things for sustainable community development: Wireless communications, sensing, and systems*, pp. 147–181, Springer, 2024.

[52] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[53] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, "Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities," *IEEE access*, vol. 9, pp. 28177–28193, 2021.

[54] M. N. Khan, A. Rao, and S. Camtepe, "Lightweight cryptographic protocols for iot-constrained devices: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4132–4156, 2020.

[55] M. I. Ali, S. Kaur, A. Khamparia, D. Gupta, S. Kumar, A. Khanna, and F. Al-Turjman, "Security challenges and cyber forensic ecosystem in iot driven byod environment," *IEEE Access*, vol. 8, pp. 172770–172782, 2020.

[56] P. Kumari and A. K. Jain, "A comprehensive study of ddos attacks over iot network and their countermeasures," *Computers & Security*, vol. 127, p. 103096, 2023.

[57] I. Duravkin, A. Loktionova, and A. Carlsson, "Method of slow-attack detection," in *2014 First International Scientific-Practical Conference Problems of Infocommunications Science and Technology*, pp. 171–172, IEEE, 2014.

[58] H. Harshita, "Detection and prevention of icmp flood ddos attack," *International Journal of New Technology and Research*, vol. 3, no. 3, p. 263333, 2017.

[59] I. Sreeram and V. P. K. Vuppala, "Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied computing and informatics*, vol. 15, no. 1, pp. 59–66, 2019.

[60] A. A. Acharya, K. Arpitha, B. S. Kumar, *et al.*, "An intrusion detection system against udp flood attack and ping of death attack (ddos) in manet," *International Journal of Engineering and Technology (IJET)*, vol. 8, no. 2, 2016.

[61] F. S. Cebeloglu and M. Karakose, "A cyber security analysis used for unmanned aerial vehicles in the smart city," in *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, pp. 1–6, IEEE, 2019.

[62] E. Y. Chen, "Detecting tcp-based ddos attacks by linear regression analysis," in *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.*, pp. 381–386, IEEE, 2005.

[63] C. Kaufman, R. Perlman, and B. Sommerfeld, "Dos protection for udp-based protocols," in *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 2–7, 2003.

[64] Y. Gilad and A. Herzberg, "Fragmentation considered vulnerable," *ACM Transactions on Information and System Security (TISSEC)*, vol. 15, no. 4, pp. 1–31, 2013.

[65] M. Bogdanoski, T. Suminoski, and A. Risteski, "Analysis of the syn flood dos attack," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 5, no. 8, pp. 1–11, 2013.

[66] G. E. Raptis, C. Katsini, and C. Alexakos, "Towards automated matching of cyber threat intelligence reports based on cluster analysis in an internet-of-vehicles environment," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 366–371, IEEE, 2021.

[67] S. G. Abbas, F. Hashmat, G. A. Shah, and K. Zafar, "Generic signature development for iot botnet families," *Forensic Science International: Digital Investigation*, vol. 38, p. 301224, 2021.

[68] M. Wolfgang, "Host discovery with nmap," *Exploring nmap's default behavior*, vol. 1, p. 16, 2002.

[69] J. Shun and H. A. Malki, "Network intrusion detection system using neural networks," in *2008 fourth international conference on natural computation*, vol. 5, pp. 242–246, IEEE, 2008.

[70] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Surveying port scans and their detection methodologies," *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.

[71] A. Orebaugh and B. Pinkard, *Nmap in the enterprise: your guide to network scanning*. Elsevier, 2011.

[72] C. deRito and S. Bhatia, "Comparative analysis of open-source vulnerability scanners for iot devices," in *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*, pp. 785–800, Springer, 2022.

[73] J. R. Van Der Merwe, X. Zubizarreta, I. Lukčin, A. Rügamer, and W. Felber, "Classification of spoofing attack types," in *2018 European Navigation Conference (ENC)*, pp. 91–99, IEEE, 2018.

[74] A. A. Maksutov, I. A. Cherepanov, and M. S. Alekseev, "Detection and prevention of dns spoofing attacks," in *2017 Siberian Symposium on Data Science and Engineering (SSDSE)*, pp. 84–87, IEEE, 2017.

[75] S. Whalen, "An introduction to arp spoofing," *Node99 [Online Document]*, vol. 563, 2001.

[76] W. G. Halfond, J. Viegas, A. Orso, *et al.*, "A classification of sql injection attacks and countermeasures.," in *ISSSE*, 2006.

[77] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," *Acm Sigplan Notices*, vol. 41, no. 1, pp. 372–382, 2006.

[78] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *Proceedings of the 30th international conference on Software engineering*, pp. 171–180, 2008.

[79] M. M. S. Kumar and B. Indrani, "A study on web hijacking techniques and browser attacks," *International Journal of Applied Engineering Research*, vol. 13, no. 5, pp. 2614–2618, 2018.

[80] H. Loi and A. Olmsted, "Low-cost detection of backdoor malware," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 197–198, IEEE, 2017.

[81] M. Jensen, N. Gruschka, and R. Herkenhöner, "A survey of attacks on web services: Classification and countermeasures," *Computer Science-Research and Development*, vol. 24, no. 4, pp. 185–197, 2009.

[82] H. Sebestyen, D. E. Popescu, and R. D. Zmaranda, "A literature review on security in the internet of things: Identifying and analysing critical categories," *Computers*, vol. 14, no. 2, p. 61, 2025.

[83] S. Ghafur, S. Kristensen, K. Honeyford, G. Martin, A. Darzi, and P. Aylin, "A retrospective impact analysis of the wannacry cyberattack on the nhs," *NPJ digital medicine*, vol. 2, no. 1, p. 98, 2019.

[84] L. Mathews, "Criminals hacked a fish tank to steal data from a casino." https://www.forbes.com/sites/leemathews/2017/07/27/criminals-hacked-a-fish-tank-to-steal-data-from-a-casino/, Jul 2017.

[85] K. Paul, "Dozens sue amazon's ring after camera hack leads to threats and racial slurs." https://www.forbes.com/sites/leemathews/2017/07/27/criminals-hacked-a-fish-tank-to-steal-data-from-a-casino/, Dec 2020.

[86] C. Cimpanu, "Garmin services and production go down after ransomware attack." https://www.zdnet.com/article/garmin-services-and-production-go-down-after-ransomware-attack/, Jul 2020.

[87] J. Easterly, "The attack on colonial pipeline: What we've learned & what we've done over the past two years." https://www.cisa.gov/news-events/news/attack-colonial-pipeline-what-weve-learned-what-weve-done-over-past-two-
May 2023.

[88] C. Gartenberg, "Security startup verkada hack exposes 150,000 security cameras in tesla factories, jails, and more." https://www.theverge.com/2021/3/9/22322122/verkada-hack-150000-security-cameras-tesla-factory-cloudflare-jails-hosp
Mar 2021.

[89] A. Greenberg, "A hacker tried to poison a florida city's water supply, officials say." https://www.wired.com/story/oldsmar-florida-water-utility-hack/, Feb 2021.

[90] G. Thiyagarajan, V. Bist, and P. Nayak, "The hidden dangers of outdated software: A cyber security perspective," *arXiv preprint arXiv:2505.13922*, 2025.

[91] S. A. Bkheet and J. I. Agbinya, "A review of identity methods of internet of things (iot)," *Advances in Internet of Things*, vol. 11, no. 4, pp. 153–174, 2021.

[92] G. Nebbione and M. C. Calzarossa, "Security of iot application layer protocols: Challenges and findings," *Future Internet*, vol. 12, no. 3, p. 55, 2020.

[93] M. Safi, S. Dadkhah, F. Shoeleh, H. Mahdikhani, H. Molyneaux, and A. A. Ghorbani, "A survey on iot profiling, fingerprinting, and identification," *ACM Transactions on Internet of Things*, vol. 3, no. 4, pp. 1–39, 2022.

[94] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 94–104, 2015.

[95] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based iot device authentication," in *IEEE INFOCOM 2017-IEEE conference on computer communications*, pp. 1–9, IEEE, 2017.

[96] A. T. Ayedh M, A. W. A. Wahab, and M. Y. I. Idris, "Systematic literature review on security access control policies and techniques based on privacy requirements in a byod environment: State of the art and future directions," *Applied Sciences*, vol. 13, no. 14, p. 8048, 2023.

[97] A. Das, N. Borisov, and M. Caesar, "Tracking mobile web users through motion sensors: Attacks and defenses.," in *NDSS*, 2016.

[98] K. Yang, Q. Li, X. Lin, X. Chen, and L. Sun, "ifinger: Intrusion detection in industrial control systems via register-based fingerprinting," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 955–967, 2020.

[99] C. Shen, C. Liu, H. Tan, Z. Wang, D. Xu, and X. Su, "Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 26–31, 2018.

[100] Y.-Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, 2015.

[101] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[102] S. Suthaharan, "Support vector machine," in *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, pp. 207–235, Springer, 2016.

[103] M. Steinbach and P.-N. Tan, "knn: k-nearest neighbors," in *The top ten algorithms in data mining*, pp. 165–176, Chapman and Hall/CRC, 2009.

[104] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, Seattle, USA, 2001.

[105] A. Bremler-Barr, H. Levy, and Z. Yakhini, "Iot or not: Identifying iot devices in a shorttime scale," *arXiv preprint arXiv:1910.05647*, 2019.

[106] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[107] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[108] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.

[109] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, pp. 413–422, IEEE, 2008.

[110] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6, IEEE, 2018.

[111] M. H. Sazlı, "A brief review of feed-forward neural networks," *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, vol. 50, no. 01, 2006.

[112] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[113] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[114] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[115] J. Wang, J. Zhong, and J. Li, "Iot-portrait: Automatically identifying iot devices via transformer with incremental learning," *Future Internet*, vol. 15, no. 3, p. 102, 2023.

[116] Q. Hao and Z. Rong, "Iottfid: An incremental iot device identification model based on traffic fingerprint," *IEEE Access*, vol. 11, pp. 58679–58691, 2023.

[117] J. Ortiz, C. Crawford, and F. Le, "Devicemien: network device behavior modeling for identifying unknown iot devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, pp. 106–117, 2019.

[118] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering {Internet-of-Things} devices," in *27th USENIX security symposium (USENIX Security 18)*, pp. 327–341, 2018.

[119] K. Kostas, M. Just, and M. A. Lones, "Iotdevid: A behavior-based device identification method for the iot," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23741–23749, 2022.

[120] L. Ying *et al.*, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[121] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 workshop on attacks and solutions in hardware security*, pp. 41–50, 2018.

[122] M. Rabbani, J. Gui, F. Nejati, Z. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. Opushnyev, R. Lu, and A. A. Ghorbani, "Device identification and anomaly detection in iot environments," *IEEE Internet of Things Journal*, 2024.

[123] K. Kostas, R. Y. Kostas, M. Just, and M. A. Lones, "Gemid: Generalizable models for iot device identification," *Internet of Things*, p. 101806, 2025.

[124] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[125] K. Kostas, *Behaviour-based security with machine learning on IoT networks*. PhD thesis, Heriot-Watt University, 2024.

[126] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home iot device classification using machine learning-based network traffic analysis," *IEEe Access*, vol. 10, pp. 97117–97141, 2022.

[127] N. Msadek, R. Soua, and T. Engel, "Iot device fingerprinting: Machine learning based encrypted traffic analysis," in *2019 IEEE wireless communications and networking conference (WCNC)*, pp. 1–8, IEEE, 2019.

[128] B. Charyyev and M. H. Gunes, "Iot traffic flow identification using locality sensitive hashes," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[129] S. Yin, W. Zhang, Y. Feng, Y. Xiang, and Y. Liu, "Automatic iot device identification: a deep learning based approach using graphic traffic characteristics," *Telecommunication Systems*, vol. 83, no. 2, pp. 101–114, 2023.

[130] Z. M. Ferdjouni, *Passive IoT Device-Type Identification Using Few-Shot Learning*. PhD thesis, Concordia University, 2023.

[131] D. Rostami, "Dorreenrostami/scaniot." https://github.com/DorreenRostami/ScanIoT, Jan 2025.

[132] M. Fezari and A. Al-Dahoud, "Raspberry pi 5: The new raspberry pi family with more computation power and ai integration," 2023.

[133] IEEE, "Organizationally unique identifier (oui)." https://standards-oui.ieee.org/.

[134] S.-T. Sun, A. Cuadros, and K. Beznosov, "Android rooting: Methods, detection, and evasion," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 3–14, 2015.

[135] M. Moskala and I. Wojda, *Android Development with Kotlin*. Packt Publishing Ltd, 2017.

[136] M.-R. Boueiz, "Importance of rooting in an android data acquisition," in *2020 8th international symposium on digital forensics and security (ISDFS)*, pp. 1–4, IEEE, 2020.

[137] A. A. Kadams, "Comparative evaluation of python integrated development environments (ides) on android mobile devices: Implications for teaching programming concepts," *CaJoST*, vol. 6, no. 1, pp. 35–44, 2024.

[138] P. Chougale, V. Yadav, A. Gaikwad, and B. Vidyapeeth, "Firebase-overview and usage," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 3, no. 12, pp. 1178–1183, 2021.

[139] R. Soepeno, "Wireshark: An effective tool for network analysis," *CYBV-Introd. Methods Netw. Anal*, pp. 1–15, 2023.

[140] K. Kostas, "Lstm-based-iot-device-identification." https://github.com/kahramankostas/LSTM-based-IoT-Device-Identification/issues/1, Sep 2023.

[141] M. Data and M. Aritsugi, "T-dfnn: An incremental learning algorithm for intrusion detection systems," *IEEE Access*, vol. 9, pp. 154156–154171, 2021.

[142] D. Ruano-Ordás, "Machine learning-based feature extraction and selection," 2024.

[143] P. B. Nath and M. M. Uddin, "Tcp-ip model in data communication and networking," *American Journal of Engineering Research*, vol. 4, no. 10, pp. 102–107, 2015.

[144] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics in machine learning algorithms," in *Computer science on-line conference*, pp. 15–25, Springer, 2023.

[145] R. Yacouby and D. Axman, "Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models," in *Proceedings of the first workshop on evaluation and comparison of NLP systems*, pp. 79–91, 2020.

[146] P. Kumar, R. Bhatnagar, K. Gaur, and A. Bhatnagar, "Classification of imbalanced data: review of methods and applications," in *IOP conference series: materials science and engineering*, vol. 1099, p. 012077, IOP Publishing, 2021.

[147] M. Mujahid, E. Kına, F. Rustam, M. G. Villar, E. S. Alvarado, I. De La Torre Diez, and I. Ashraf, "Data oversampling and imbalanced datasets: An investigation of performance for machine learning and feature engineering," *Journal of Big Data*, vol. 11, no. 1, p. 87, 2024.

[148] K. Vijay, J. Manikandan, B. Rajendiran, K. Sowmia, and E. I. Berna, "Deep dive on oversampling and under sampling techniques in machine learning," in *Recent Trends in Computational Intelligence and Its Application*, pp. 423–430, crc Press, 2023.