# Incremental Learning based Identification of IoT devices using Packet-level Behavioral features

***A M. Tech Project Report Submitted in Fulfillment of the Requirements for the Degree of***

**Master of Technology**

*by*

**Param Bharatbhai Patel**
(234101062)

*under the guidance of*

**Dr. Tamarapalli Venkatesh**

**to the**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**
**GUWAHATI - 781039, ASSAM**

# CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Incremental Learning based Identification of IoT devices using Packet-level Behavioral features**" is a bonafide work of **Param Bharatbhai Patel (Roll No. 234101062)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Tamarapalli Venkatesh**

Professor,

June, 2025 — Department of Computer Science & Engineering,

Guwahati. — Indian Institute of Technology Guwahati, Assam.

# Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Prof. Tamarapalli Venkatesh, from the Department of Computer Science and Engineering at IIT Guwahati. His constant support, insightful guidance, and encouragement have been instrumental in shaping this project. Working under his mentorship has not only deepened my understanding but also inspired me to push my boundaries. The lessons I've learned from him will undoubtedly guide me in the years to come.

I am also incredibly thankful to my parents and friends for their unwavering support and motivation throughout this journey. Their belief in me and their constant encouragement have been my driving force. Finally, a heartfelt thank you to Prof. Venkatesh once again, whose dedication and guidance have made this project possible.

# Abstract

The rapid expansion of Internet of Things (IoT) ecosystems presents new challenges in identifying and managing heterogeneous connected devices. Conventional identification models typically rely on static, batch-trained classifiers, which lack adaptability when deployed in real-world, evolving environments. This work proposes an incremental learning-based approach for IoT device identification using packet-level behavioral features. The system processes network traffic in mini-batches and continuously updates the model using a stochastic gradient descent classifier, enabling it to learn from streaming data without full retraining. Experimental evaluations on two benchmark datasets—Aalto and UNSW—demonstrate that the model achieves consistently high accuracy across a range of device types, including those with complex or high-volume traffic patterns. The results show that incremental learning provides a lightweight and scalable alternative for dynamic IoT environments, offering timely adaptation, efficient memory usage, and sustained classification performance.

# Contents

# Chapter 1

# Introduction

The Internet of Things (IoT) represents a transformative network of physical objects or devices embedded with sensors, software, and other technologies. This network facilitates seamless connectivity and communication among devices and systems, enabling automation, monitoring, and intelligent decision-making. Through local networks or the internet, IoT devices exchange data to enhance user experiences and operational efficiency.

## 1.1 IoT Devices

### 1.1.1 What are IoT Devices?

The Internet of Things (IoT) refers to a technological paradigm in which physical devices are embedded with sensors, processing units, and communication capabilities to interact with each other and the surrounding environment. These devices—commonly referred to as IoT devices—enable the automation and optimization of tasks across a wide range of application domains.

IoT devices are diverse in both form and function. They include smart home appliances such as thermostats, plugs, and surveillance cameras; medical wearables like heart-rate monitors and fitness trackers; industrial sensors deployed in manufacturing plants; and infrastructure components in smart cities, including traffic and air quality monitors. Despite their varied use cases, these devices share common features such as the ability to sense data, process it to some extent, and communicate it over a network.

Typically, IoT devices are designed for autonomous operation and are constrained in terms of computational power, memory, bandwidth, and battery life. This makes them suitable for long-term deployment in resource-sensitive environments but also introduces challenges in terms of manageability and security.

### 1.1.2 IoT Device Diversity and Threats

One of the defining characteristics of IoT ecosystems is their heterogeneity. Devices vary widely in hardware configurations, operating systems, communication protocols, and intended usage contexts. Some devices use IP-based protocols such as HTTP or MQTT, while others operate using low-power, non-IP protocols like ZigBee, Bluetooth Low Energy (BLE), or Z-Wave. This diversity allows IoT technology to be embedded in virtually any environment, but it also complicates standardization and security enforcement.

Security issues are further exacerbated by the fact that many IoT devices are shipped with insecure default settings, such as open ports or default passwords, and often lack mechanisms for remote firmware updates. These limitations make IoT networks vulnerable to a range of attacks, including device spoofing, unauthorized access, data theft, and botnet recruitment. For example, attackers may exploit a vulnerable device to gain access to a private network or use it as part of a large-scale distributed denial-of-service (DDoS) attack.

Given the sensitivity of environments in which IoT devices operate—such as hospitals, homes, and factories—ensuring device integrity and trust becomes essential. A fundamental requirement in this regard is the ability to reliably identify the devices connected to a network.

### 1.1.3 What is IoT Device Identification?

IoT Device Identification (DI) is the process of recognizing and classifying devices connected to a network by analyzing their observable characteristics—most commonly their network behavior. Unlike traditional identification methods that rely on explicit identifiers such as IP addresses, MAC addresses, or hostnames, behavior-based DI focuses on how a device communicates rather than who it claims to be.

This shift is especially important in modern IoT environments, where:

- Devices may communicate over anonymized or encrypted channels.
- IP/MAC identifiers can be easily spoofed or may be absent (e.g., in ZigBee or BLE networks).
- Devices can dynamically join and leave the network, sometimes without manual configuration.

Instead of relying on static metadata, behavior-based DI systems build a fingerprint of a device using traffic-level features that reflect its communication habits. These features are extracted from packet headers, payload summaries, and statistical patterns, and typically include:

- **Protocol usage:** Identifying which network and transport protocols are used (e.g., TCP, UDP, DNS, SSDP).
- **Packet sizes:** Analyzing the distribution and range of packet lengths.
- **Inter-arrival times:** Measuring timing patterns between consecutive packets or bursts.
- **Payload entropy:** Estimating randomness in packet contents to characterize control vs. data messages.
- **Port class mappings:** Mapping source/destination ports to logical categories (e.g., HTTP, DNS, dynamic).

Using these features, a machine learning model or rule-based engine can be trained to associate traffic behavior with a specific device type, brand, or model. This process enables device identification even when device vendors use proprietary protocols or non-IP transport layers.

The overall goal of DI is to allow the system to infer the identity of devices in an unsupervised or minimally supervised manner, making it highly applicable in large-scale, distributed networks where manual labeling or enrollment is infeasible.

### 1.1.4 Importance of IoT Device Identification

As the number of connected devices in IoT networks continues to grow, maintaining visibility into what devices are present and how they behave has become a cornerstone of network security and management. Accurate device identification is not just a passive inventory task—it plays a foundational role in enabling multiple layers of defense and control in IoT environments.

First, device identification is essential for access control and policy enforcement. Networks can restrict access based on device type or behavior, isolating unknown or unauthorized devices before they can interact with critical systems. For example, a network may allow communication from a known smart thermostat but block traffic from an unrecognized device attempting to connect to the same gateway.

Second, it enables vulnerability assessment and patch management. Many IoT devices have well-known security weaknesses tied to specific models or firmware versions. If a device can be accurately identified, known vulnerabilities can be cross-referenced, and the device can be flagged for update or mitigation, even if the user is unaware of the risk.

Third, identification forms the basis for anomaly detection. Understanding the normal behavior of a specific device type allows deviations to be detected more reliably. For instance, if a device identified as a motion sensor starts transmitting large volumes of data at unexpected intervals, it may indicate compromise or misconfiguration.

Additionally, accurate DI contributes to network segmentation, allowing administrators to group devices logically and apply differentiated security rules. This becomes especially important in environments like smart factories or hospitals, where different classes of devices have different risk profiles.

In summary, IoT device identification is not an isolated technical feature—it is a key enabler for secure, scalable, and responsive IoT system management. As networks become more dynamic and diverse, the ability to perform accurate and adaptive DI becomes increasingly important to maintaining system integrity.

In dynamic IoT networks, devices may be added, removed, or updated regularly. As a result, device identification systems must not only be accurate but also adaptive. The ability to update models over time—without retraining from scratch—is essential to maintain effectiveness in such evolving conditions. This motivates the integration of incremental learning techniques into DI systems to support long-term scalability and resilience.

## 1.2 Incremental Learning

### 1.2.1 Background

Incremental learning, also referred to as lifelong learning or continual learning, is a machine learning paradigm that enables a model to learn from data presented in sequential chunks or tasks over time. Unlike traditional batch learning, which assumes access to the complete dataset during training, incremental learning relaxes this assumption by allowing the model to be updated as new data arrives—without reprocessing the entire dataset or starting from scratch.

This capability is crucial in scenarios where data is continuously generated or where memory and computational resources are limited. Examples include mobile applications, edge computing, robotics, streaming analytics, and any environment where learning must occur on the fly.

### 1.2.2 Need for Incremental Learning in IoT Device Identification

In real-world IoT environments, the landscape of connected devices is not static—it is constantly evolving. New devices are frequently introduced into networks as homes and industries adopt new technologies, while existing devices may undergo behavioral changes due to firmware updates or shifting usage patterns. Traditional static models for device identification are ill-equipped to handle this fluidity.

Most conventional IoT device identification systems rely on batch learning, where the entire training dataset—consisting of traffic traces from all known devices—is used to build a model in a one-time training process. While effective under controlled conditions, these models require complete retraining when new devices need to be included. This retraining process not only demands access to the full historical dataset but also consumes significant computational resources. More critically, it introduces downtime and delays in deployment, which are unacceptable in security-sensitive environments.

Moreover, in large-scale IoT networks such as smart cities, industrial control systems, or enterprise environments, it is often impractical to assume that every device will be known in advance. Devices may be added without prior registration, and their data may arrive in unpredictable batches. A model that cannot incorporate this new information in real time becomes increasingly outdated and less reliable.

Incremental learning addresses this limitation by allowing the device identification model to update itself continuously as new data arrives. Rather than retraining the model from scratch, incremental learning integrates new patterns or classes into the existing model while attempting to preserve previously acquired knowledge. This makes it well-suited for IoT networks that require real-time adaptability, scalability, and minimal retraining overhead.

In addition, incremental learning reduces the risk of model drift in long-term deployments. It enables the system to adapt to natural behavioral shifts in devices, whether caused by software updates, changes in usage context, or network conditions. This continuous refinement enhances the robustness of the identification system and ensures that it remains effective over time.

## 1.3 Motivation

With the exponential growth of Internet of Things (IoT) networks, the number and diversity of connected devices have increased dramatically across domains such as smart homes, healthcare, manufacturing, and critical infrastructure. These devices, while essential to automation and monitoring, introduce new security and management challenges.

One of the foundational requirements for securing such environments is the ability to accurately identify the devices connected to a network.

Behavior-based device identification techniques, which analyze network traffic patterns to classify device types or models, have shown promise due to their generalizability and resistance to spoofing. However, most existing approaches rely on static batch learning. These models assume that the entire set of device types is known beforehand, and require complete retraining whenever new devices are introduced. This assumption does not hold in real-world deployments, where devices may be added, removed, or updated unpredictably.

The dynamic nature of IoT environments makes it essential to adopt learning paradigms that can adapt over time. Here, incremental learning offers a promising solution. By allowing the model to incorporate new knowledge in stages, it eliminates the need for full retraining and supports real-time model evolution. This adaptability is particularly valuable in environments where device behavior changes due to firmware updates, configuration changes, or usage context shifts.

Moreover, incremental learning enables the identification system to scale gracefully with the addition of new device types. It ensures that the model remains relevant and accurate over long-term deployments, even as the composition of the network changes. At the same time, it reduces computational and storage overhead, making it feasible to deploy on resource-constrained systems such as gateways or edge devices.

However, integrating incremental learning into IoT device identification is not straightforward. It raises key challenges such as catastrophic forgetting, where learning new devices may degrade the model's performance on previously seen ones. Therefore, it becomes crucial to design effective update mechanisms and evaluation protocols to balance new learning with knowledge retention.

This thesis is motivated by the need to build a scalable, adaptive, and efficient IoT device identification system that reflects the realities of operational networks. By combining behavior-based fingerprinting with incremental learning, the goal is to develop a system capable of identifying both existing and emerging devices with high accuracy, minimal overhead, and sustained robustness over time.

## 1.4 Research Objectives

The goal of this research is to design and evaluate an IoT device identification system that leverages incremental learning to address the dynamic nature of real-world IoT environments. Traditional behavior-based identification approaches assume a fixed set of devices and rely on batch learning models trained with complete datasets. However, such models require full retraining to incorporate new data over time, which is computationally expensive and unsuitable for deployment in evolving traffic environments.

This thesis seeks to overcome these limitations by developing a system that can incrementally adapt to incoming data while preserving identification accuracy. The objective is to enable a model that evolves with time, updates continuously using mini-batches of network traffic, and retains prior knowledge without requiring full reinitialization.

The research focuses on adapting a behavior-based classification pipeline to support incremental learning. It involves implementing a structured update process where the model is trained incrementally on temporally ordered mini-batches, simulating real-world traffic arrival. The evaluation considers overall accuracy, per-device classification performance, and stability under different batch size configurations.

By achieving this, the work aims to contribute a practical, scalable solution for real-time IoT device identification—capable of maintaining robustness and adaptability in dynamic and resource-constrained environments without complete retraining.

# Chapter 2

# Literature Review

## 2.1 Overview

The task of identifying IoT devices within a network has attracted significant research attention due to its critical role in securing and managing heterogeneous and dynamic IoT environments. Traditional approaches based on static identifiers like IP and MAC addresses are increasingly unreliable, especially in scenarios involving anonymized communication or non-IP protocols. To address these challenges, recent studies have explored behavior-based methods that fingerprint devices by analyzing patterns in their network traffic. These approaches often leverage machine learning to classify devices based on protocol usage, packet size distributions, and timing features, offering improved generalizability and resistance to spoofing.

This section reviews the evolution of IoT device identification techniques, highlighting the strengths and limitations of existing methods and setting the stage for adaptive, learning-based solutions.

## 2.2 IoT Device Data

An IoT device dataset is a structured collection of data that captures the network behaviors, communication patterns, and physical characteristics of IoT devices. These datasets typically include raw network traffic (.pcap files), extracted features such as packet size, inter-arrival times, and protocol usage, along with behavioral metrics like activity cycles[SSG+17], signaling patterns, and cipher suites[SGL+18]. They are often annotated with labels indicating device types, vendors, or instances, making them suitable for supervised learning tasks. Metadata, including the deployment environment and device configurations, may also be included to provide context. Such datasets are collected through passive monitoring of network traffic, controlled experiments in testbeds, simulated scenarios, or crowdsourced data from IoT gateways, ensuring coverage of diverse devices and settings.

## 2.3 Literature Review : IoT Device Identification

### 2.3.1 Metadata-Based Identification Techniques

Early approaches to IoT device identification primarily relied on *network-layer metadata*, such as IP addresses, MAC addresses, port numbers, DHCP options, and DNS queries. These methods were attractive due to their simplicity and low computational overhead, requiring only basic inspection of packet headers without deep traffic analysis or learning algorithms.

For instance, MAC address-based identification is straightforward: each device has a unique hardware address that can often be mapped to a vendor using Organizationally Unique Identifiers (OUIs). Similarly, static IP address allocations or consistent DHCP lease patterns were used to associate devices with identities in small, controlled environments. In some cases, device types could be inferred from hostname conventions or typical DNS query patterns.

One of the notable works in this space is *IoT Sentinel* by Miettinen et al., which used MAC address-based clustering combined with a lightweight fingerprinting mechanism to identify and isolate vulnerable devices [MMH+17]. Another study by Sivanathan et al. [SGL+19] captured long-term flow-level traffic data and applied statistical analysis on DNS, NTP, and port usage patterns to classify devices based on their communication metadata.

However, these approaches face several major limitations:

- **Susceptibility to Spoofing:** Attackers can easily forge MAC or IP addresses, undermining the reliability of such identifiers.
- **Dynamic Addressing:** In real-world deployments, DHCP and NAT can result in changing IP addresses, making consistent device tracking difficult.
- **Lack of Support for Non-IP Protocols:** Many IoT devices communicate over low-energy or non-IP protocols such as ZigBee, Bluetooth, or Z-Wave, where IP/MAC identifiers are unavailable.
- **Reduced Visibility in Encrypted Networks:** The growing use of encrypted transport protocols like HTTPS or DoH limits the availability of readable metadata.

Due to these limitations, metadata-based methods have largely fallen short in delivering reliable and scalable solutions for modern IoT environments. Their inability to generalize across diverse networks and device types has prompted a shift toward *behavior-based* and *feature-driven* approaches, which infer device identity from communication patterns rather than static identifiers.

### 2.3.2 Behavior-Based Identification Techniques

As the limitations of metadata-based identification became apparent, research in IoT device identification shifted toward behavior-based techniques. These methods do not rely on static attributes like MAC or IP addresses; instead, they analyze how a device communicates over the network. The underlying assumption is that each device type exhibits distinctive communication patterns—such as protocol usage, timing, and traffic structure—that can be used to build a behavioral fingerprint.

Behavior-based approaches typically extract features from packet-level or flow-level network traffic. Common features include protocol types (e.g., TCP, UDP, DNS), packet size distributions, inter-arrival times, port usage patterns, and payload entropy. These features are then used as input to machine learning classifiers to identify device types or models. Compared to metadata-based methods, these techniques are more robust to spoofing and better suited for environments with encrypted or anonymized traffic.

One notable contribution is *IoTDevID*, which uses a multi-stage feature selection process to identify a generalizable set of packet-level features and applies decision tree classifiers to discriminate between devices [KJL22]. It explicitly removes non-generalizable features such as session identifiers and port numbers that are highly specific to certain datasets. Another study, *IoTSense*, builds on the fingerprinting approach of IoT Sentinel but introduces entropy-based features and evaluates performance on multiple devices using session-level traffic segments [BBP$^+$18]. Similarly, Aksoy and Gunes [AG19] extracted over 200 features from packet headers and applied genetic algorithms to identify the most relevant subset for classification.

These methods have demonstrated high classification accuracy, especially in controlled environments with balanced datasets. They are also applicable to both IP and non-IP protocols, making them more versatile for modern IoT networks. Additionally, since behavior-based techniques operate passively and do not require prior device registration or deep packet inspection, they are well suited for scalable, real-time monitoring.

However, despite their improved generalizability, most behavior-based models are trained in a static fashion and assume a fixed set of known devices. They do not inherently support dynamic updates or new class integration after deployment. As such, their effectiveness diminishes over time in real-world networks where new devices are frequently introduced or where device behavior evolves. This highlights the need for adaptable learning frameworks capable of maintaining high accuracy while evolving alongside the network.

### 2.3.3 Machine Learning Models Used in Device Identification

In the domain of IoT device identification, machine learning models are widely used to classify devices based on their communication behavior. These models are typically trained on traffic-derived features such as protocol usage, packet lengths, timing patterns, and entropy-related attributes. Various supervised learning algorithms have been applied across studies, with performance evaluated in terms of classification accuracy, generalizability, and computational feasibility.

Decision Trees (DT) and Random Forests (RF) are the most frequently used models in IoT device identification research due to their transparency, low inference cost, and strong performance on structured network traffic data. Aksoy and Gunes [AG19] achieved high accuracy using Decision Trees in their study on automated IoT device identification based on packet header features. Kostas et al. [KJL22] also selected Decision Trees for their IoTDevID framework after evaluating multiple classifiers, citing its favorable balance between speed and accuracy.

Gradient Boosting (GB) methods have also been explored, although less frequently than DT and RF. In particular, Kostas et al. [KJL22] tested Gradient Boosting as part of their model selection process and found it to be competitive in terms of accuracy, though more computationally demanding than DT.

Support Vector Machines (SVMs) have been evaluated in earlier works, such as those by Bezawada et al. [BBP+18], but were found to be less suitable for multi-class problems involving many device types, as SVMs scale poorly with large and imbalanced datasets.

k-Nearest Neighbors (k-NN) and Naïve Bayes (NB) models have also been included in comparative studies. For example, Aksoy and Gunes [AG19] included both in their evaluation and found that while k-NN could achieve reasonable accuracy, it was significantly slower during inference. Naïve Bayes, on the other hand, consistently underperformed due to its strong independence assumptions, and was mainly used as a baseline classifier.

Several studies also emphasize the role of feature selection in improving classification accuracy. Aksoy and Gunes [AG19] applied genetic algorithms to prune a high-dimensional feature set extracted from packet headers. Similarly, Kostas et al. [KJL22] used a multi-stage process combining feature importance voting and genetic algorithm-based selection to refine their feature pool, focusing on features that generalize across datasets.

While deep learning has been transformative in many areas, it has seen limited application in IoT device identification due to the relatively small size of public datasets and the interpretability and efficiency requirements of practical deployments. Most current studies favor traditional machine learning models that are easier to deploy in real-time, resource-constrained environments such as IoT gateways and routers.

In summary, existing IoT device identification studies predominantly favor interpretable and efficient models such as Decision Trees and Random Forests, often complemented by feature selection techniques to reduce complexity and enhance generalization. Although more complex models like Gradient Boosting and SVMs have been explored, they are less frequently adopted in practical pipelines due to concerns around scalability and inference speed.

### 2.3.4 Datasets for IoT Device Identification

Numerous studies on IoT device identification have relied on publicly available network traffic datasets that capture the behavior of IoT devices under various operational settings. Although newer datasets are emerging each year, the *Aalto University IoT Devices Captures (Aalto)* [MMH+17] and the *UNSW IoT Traffic Traces, Data for IEEE TMC 2018 (UNSW45 DI)* [SGL+18] remain the most prominent and widely adopted benchmarks in this domain. These datasets were both collected in 2016 and continue to be extensively utilized; in fact, more than half of all device identification studies have been performed using these two datasets. Early works predominantly leveraged the Aalto dataset, which offers clean, per-device packet captures in isolated lab environments, ideal for feature extraction and establishing baseline accuracy. In contrast, the UNSW dataset provides long-term, realistic captures from multiple devices operating simultaneously in a smart home setting, enabling better evaluation of model robustness and generalization. More recent research has begun to incorporate newer datasets like *VARIoT* [Pro21], *IoT-23* [G+20], and *CIC IoT 2023* [L+23], which include either benign traffic from a wider variety of device types or traffic mixes containing both normal and malicious activity. The overall trend reflects a shift from static, controlled environments to more dynamic and heterogeneous network settings, with the goal of improving real-world applicability of identification models.

An important factor in evaluating IoT device identification systems is the number and diversity of devices included in the dataset. A larger number of device classes offers a more realistic test of a model's scalability and its ability to generalize across varied hardware and communication patterns. Conversely, datasets with a small number of devices can lead to inflated performance metrics, especially in multi-class classification tasks where the chance-level accuracy is relatively high. For example, the *IoT-23* dataset contains only three benign devices, making it unsuitable for robust device identification benchmarks. Similarly, datasets such as *N-BaIoT*, *IoT-deNAT*, and *IoT-TrafficDataset* include fewer than ten devices, which limits their generalizability to real-world deployments. When evaluating identification models, it is important to distinguish between the total number of physical devices and the number of device classes. Typically, identification is performed at the device type or model level, where multiple physical devices of the same model are grouped under a single label. For instance, in the *Aalto* dataset, EdimaxCam1 and EdimaxCam2 are treated as a single class (EdimaxCam) despite having different MAC addresses and being physically distinct. This distinction plays a critical role in interpreting the classification results and comparing the difficulty levels across datasets.

A critical consideration in the assessment of IoT device identification datasets is the availability of raw packet-level data, typically provided as packet capture (pcap) files. These files are essential for flexible and detailed feature extraction tailored to specific research needs. In contrast, some datasets offer only pre-processed data in the form of CSV files, where the feature extraction has already been performed. While this format simplifies data handling, it inherently restricts researchers to a fixed set of features selected by the original authors—potentially overlooking other informative characteristics. This loss of flexibility limits the ability to experiment with alternative feature sets or refine existing ones. Without access to raw pcap files, reproducing results or adapting the dataset for different methodologies becomes difficult. Consequently, datasets like *ProfilIoT*, *N-BaIoT* [MBM+18], *LwHBENCH-RASPI* [SKIJ20], and *IoT-deNAT* [BKA18], which lack raw captures and provide only extracted features in CSV format, are less suited for in-depth or customized device identification research.

### 2.3.5 Limitations in Existing Work

Despite notable advances in IoT device identification through behavior-based machine learning models, several key limitations persist in existing research. A fundamental assumption in most prior work is the availability of a fixed training dataset comprising known device types. This assumption does not hold in real-world deployments, where new IoT devices are regularly introduced, and device behavior may evolve over time due to firmware updates, configuration changes, or shifting user interactions. As a result, models trained under static settings tend to degrade in performance when exposed to novel or changing device patterns.

Moreover, the majority of studies do not simulate incremental or phased data arrival, which is characteristic of realistic network environments. Instead, train-test splits are typically random or session-based, ignoring temporal continuity. This hinders the evaluation of a model's adaptability and resilience over time. Full model retraining with each new data batch, while possible, is computationally expensive and infeasible in resource-constrained deployments such as edge devices or routers.

## 2.4 Related Work : Incremental Learning-Based IoT Device Identification

As IoT networks continue to grow in scale and complexity, the need for adaptive device identification systems has become increasingly apparent. Traditional methods, while effective in controlled settings, assume access to complete datasets and static device populations—assumptions that often break down in dynamic real-world environments. To address this, a small but growing body of research has explored the use of incremental learning techniques to update models as new data becomes available.

*IoTTFID* is one such approach, leveraging traffic fingerprints and adaptive tree-based classifiers to gradually learn device behavior from streaming data [DIKE23]. While it demonstrates the feasibility of updating identification models over time, it operates on flow-level traffic summaries, which may miss subtle patterns present at the packet level. Other frameworks, such as *ProfilIoT* [FBLM20] and *IoT-deNAT* [BKA18], incorporate behavior modeling under evolving conditions, but offer limited flexibility due to reliance on pre-extracted CSV features and lack of support for deeper feature engineering or model updates.

Overall, existing studies remain limited in terms of data granularity, scale, and long-term evaluation. Most do not account for changes in device behavior due to firmware updates or network reconfiguration, and few provide mechanisms to retain previously learned patterns without retraining. These limitations highlight the need for an approach that combines the expressiveness of packet-level features with the flexibility of incremental learning, allowing models to adapt over time without sacrificing performance or interpretability.

## 2.5 Conclusion of Literature Review

The landscape of IoT device identification has evolved considerably, transitioning from static metadata-based methods to more robust behavior-driven techniques supported by machine learning. These advancements have significantly improved classification accuracy, especially in structured environments, and enabled identification even in the presence of encryption or protocol obfuscation. However, much of the existing work continues to operate under static assumptions, relying on fixed datasets, known device classes, and one-time training processes. This limits their applicability in real-world IoT deployments where data is constantly generated, devices are dynamically introduced, and behaviors evolve due to updates or contextual changes.

While a few studies have attempted to incorporate incremental learning into IoT device identification, they often rely on flow-level data and pre-extracted features, which restrict the granularity and adaptability of models. The lack of access to raw packet-level data, combined with insufficient attention to catastrophic forgetting and long-term model stability, presents clear gaps in the literature. These limitations point to the need for an identification framework that can operate at packet-level granularity while supporting continual updates to accommodate evolving traffic patterns.

This motivates the design of an incremental learning-based identification pipeline that preserves the rich feature set of packet-level analysis while enabling scalable, adaptive learning suitable for dynamic IoT environments.

# Chapter 3

# Methodology

## 3.1 Overview of the Approach

This work proposes a lightweight, adaptive framework for identifying IoT devices based on their network traffic patterns using a batch-incremental learning strategy. Unlike traditional static learning pipelines that require retraining the entire model whenever new data becomes available, the proposed framework processes packet-level features in mini-batches and incrementally updates the model with each batch.

The motivation behind this approach stems from the real-world deployment of IoT devices in dynamic environments, where packet streams from known devices are continuously generated. This necessitates a learning system that can evolve in response to new behavioral patterns without incurring the overhead of full retraining.

The methodology involves extracting protocol-level features from packet capture (PCAP) files using a pre-defined feature extraction pipeline. These features undergo preprocessing steps such as cleaning, selection, and standardization. The processed dataset is then partitioned into fixed-size batches, simulating a streaming environment.

An online-capable classifier—specifically, `SGDClassifier`—is employed to support incremental updates using the `partial_fit()` function. After predicting on each incoming batch, the model is updated using that same batch, thereby enabling continual learning.

Performance evaluation is carried out using prequential accuracy and macro-averaged F1-score over all processed batches. This ensures that the system can respond to behavior changes in real-time traffic without requiring access to past data or reinitializing the model. The framework is fully compatible with pre-recorded datasets but designed to be extensible to real-time deployments on routers or edge gateways.

## 3.2 Dataset Preparation

This work utilizes two publicly available IoT traffic datasets containing labeled, packet-level network traces: the **Aalto dataset** and the **UNSW dataset**. These datasets were selected due to their diverse representation of consumer IoT devices and the availability of extracted packet features in structured formats suitable for machine learning.

#### Aalto Dataset

The Aalto dataset consists of packet traces collected from 31 IoT devices representing 27 distinct device types. These include a variety of consumer devices such as smart plugs, surveillance cameras, lighting systems, and home automation sensors. The traffic captures reflect both active and idle device behavior, recorded in a smart home setting. For this implementation, we used CSV files containing pre-extracted packet features derived from the original `.pcap` captures.

#### UNSW Dataset

The UNSW dataset contains network traffic data from 33 IoT device classes, recorded over a 60-day period in a controlled laboratory environment. The dataset covers a wide range of device manufacturers and functional categories. Similar to the Aalto dataset, preprocessed CSV files were used, where each entry corresponds to a single packet and is labeled with the originating device type.

#### Preprocessing

Each row in the dataset corresponds to a single packet and includes various protocol-level features along with a device label. The following preprocessing steps were applied uniformly to both datasets prior to training:

- Removal of rows containing missing or invalid feature values
- Encoding of device labels into integer class identifiers
- Preservation of the natural class imbalance in the datasets
- No synthetic sampling or resampling was performed
- Data was processed in the original temporal order of packet arrival

To enable staged updates to the model during training, the temporally ordered dataset was divided into fixed-size mini-batches. Each mini-batch contains a contiguous sequence of packets and simulates a time-consistent segment of network traffic. During training, the model predicts labels for the current mini-batch and is then updated using the same batch, reflecting a streaming data scenario where learning occurs incrementally. Multiple batch sizes were explored (e.g., 250, 500, 1000 packets) to evaluate the trade-off between update frequency and overall accuracy.

## 3.3 Feature Extraction and Selection

This work utilizes the same feature extraction and selection pipeline provided with the original dataset preparation process. The features are derived from packet-level network traces and are designed to capture behavioral characteristics of IoT devices without relying on static identifiers such as MAC or IP addresses.

### Feature Extraction

The raw packet traces from both the Aalto and UNSW datasets were processed using a domain-specific feature extraction pipeline that converts `.pcap` files into structured CSV files. Each packet is represented by a set of protocol-layer features extracted from Ethernet, IP, TCP, UDP, DHCP, DNS, and other IoT-relevant protocols.

The extracted features include:

- Packet size, header lengths, and flag values

- IP TTL, type-of-service (TOS), and fragmentation indicators
- TCP flags (e.g., ACK, FIN), window size, and data offset
- DHCP and DNS option fields
- Entropy and payload byte length
- Source and destination port class mappings

All features were extracted from protocol headers without performing deep packet inspection, ensuring compatibility with encrypted or privacy-sensitive traffic.

### Feature Selection

A multi-stage feature selection process was applied to eliminate redundant or non-generalizable features while retaining those most predictive of device behavior. The process consisted of the following stages:

1. **Voting-Based Filter Selection**: Multiple statistical and model-driven techniques such as Information Gain, Chi-square tests, L1-based regularization, and feature importance scores from tree-based models were applied. Features that received zero votes across all techniques were discarded.

2. **Generalization Validation**: The remaining features were evaluated for their stability across different device sessions. Session-dependent features (e.g., TCP sequence numbers, raw port numbers, and IP IDs) were removed if they reduced performance on unseen data.

3. **Wrapper-Based Optimization**: A genetic algorithm-based wrapper selection method was used to optimize the final feature subset for classification accuracy and low redundancy.

The final feature subset contained approximately 32–35 protocol-level and statistical features. This subset was used consistently across all experiments to ensure comparability and efficiency.

## 3.4 Learning Pipeline Design

This section presents the design of the learning pipeline used to identify IoT devices from network traffic in a temporally evolving environment. The pipeline simulates real-world conditions, where network traffic from IoT devices is continuously generated, and the learning model must adapt incrementally without requiring retraining from scratch. The architecture is based on a mini-batch incremental learning strategy using prequential evaluation.

### 3.4.1 Problem Setup and Learning Objective

The task is formulated as a multi-class classification problem, where each network packet, represented as a fixed-length feature vector, must be classified as belonging to one of the known IoT device types. The data is processed sequentially to simulate the natural order of packet arrival in a live network. The model must learn incrementally from streaming input, with updates occurring at regular intervals based on mini-batch divisions.

Unlike traditional models that train on the full dataset at once, the learning process here is designed to process and update the model incrementally, allowing the system to scale to large volumes of data, adapt to evolving traffic patterns, and reduce computational overhead.

### 3.4.2 Mini-Batch Construction

After preprocessing, the dataset is sorted in its original temporal order to reflect real packet arrival sequences. The data is then divided into fixed-size, non-overlapping mini-batches of size $B \in \{250, 500, 1000\}$. Each mini-batch $\mathcal{B}_t$ contains a contiguous sequence of $B$ packets:

$$\mathcal{B}_t = \{(x_{tB}, y_{tB}), (x_{tB+1}, y_{tB+1}), \ldots, (x_{tB+B-1}, y_{tB+B-1})\}$$

This setup emulates real-time learning by ensuring that predictions are made on previously unseen data before the model is updated. It also enables analysis of the effect of batch size on learning dynamics and model stability.

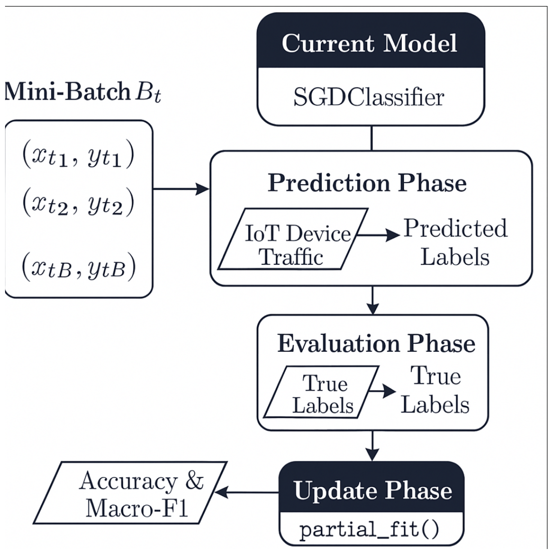

**Fig. 3.1** Architecture of the incremental learning pipeline for IoT device identification. Packet features are processed in mini-batches, passed through a scalable online classifier, and predictions are evaluated and updated sequentially.

### 3.4.3 Classifier Design and Justification

The classifier selected for this pipeline is the `SGDClassifier` from the `scikit-learn` library. This model supports the `partial_fit()` method, enabling it to be updated incrementally using streaming mini-batches. It is configured with the `log_loss` objective for probabilistic multi-class classification.

$$\mathcal{L}_{\log} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{[y_i=c]} \cdot \log(\hat{p}_{i,c}) \tag{3.1}$$

The selection of `SGDClassifier` is based on the following considerations:

- **Online Learning Capability**: It supports true incremental learning through `partial_fit()` without requiring full retraining.
- **Efficiency and Scalability**: Training time is linear in the number of samples, making it suitable for large-scale datasets.

- **Low Memory Footprint**: The model maintains only the weight vector, making it ideal for deployment in resource-constrained environments such as edge devices.
- **Multi-Class Support**: Implements a one-vs-rest strategy to handle classification across multiple device types.
- **Probabilistic Outputs**: Using the log-loss function allows probabilistic predictions, which are useful for confidence scoring and threshold-based decisions.

Alternative classifiers such as k-Nearest Neighbors, Random Forests, and SVMs were not used due to their limitations in streaming settings, including the need for full retraining, high memory requirements, or the inability to handle batch-wise updates efficiently.

### 3.4.4 Training and Evaluation Procedure

The learning procedure follows a prequential (predict-then-update) evaluation strategy. For each mini-batch $\mathcal{B}_t$, the model executes the following steps:

1. **Prediction Phase**: The current model is used to predict the labels for all packets in $\mathcal{B}_t$.
2. **Evaluation Phase**: Predicted labels are compared with the true labels in $\mathcal{B}_t$ to compute accuracy and macro-averaged F1-score.
3. **Update Phase**: The model is updated on $\mathcal{B}_t$ using the `partial_fit()` function.

This evaluation scheme ensures that predictions are always made on unseen data and that the model evolves in response to the actual sequence of network activity.

### 3.4.5 Batch Size Analysis

The batch size $B$ serves as a tunable parameter controlling the trade-off between update frequency and stability. Smaller batch sizes (e.g., $B = 250$) enable more frequent updates, allowing the model to react quickly to new patterns or behavioral shifts in traffic. However, they may introduce noise and variability in gradient updates. Larger batch sizes (e.g., $B = 1000$) provide smoother updates and reduce variance but may delay the model's response to changing conditions.

Empirical results comparing performance across batch sizes are reported in the evaluation section, highlighting this trade-off in both accuracy and model adaptability.

## 3.5 Performance Evaluation Setup

### 3.5.1 Evaluation Metrics

The performance of the proposed incremental learning framework was evaluated using the following classification metrics:

- **Accuracy**: The ratio of correctly predicted device labels to the total number of predictions made.
- **Per-class Precision and Recall**: These metrics provide detailed insights into the classifier's performance for individual device types, helping to identify any bias or degradation in specific classes.

All metrics were computed incrementally after each batch and also reported cumulatively over the full dataset. This dual evaluation helps analyze both local adaptability and long-term learning behavior.

### 3.5.2 Tools and Libraries

The evaluation pipeline was implemented using the following Python libraries:

- `scikit-learn` (`sklearn.metrics`) for computing:
  - `accuracy_score()`
  - `f1_score(average='macro')`
  - `precision_score()`, `recall_score()`
  - `confusion_matrix()`

- `matplotlib` for visualizations, including:
  - Batch-wise accuracy and F1-score plots
  - Heatmaps for confusion matrices
- `NumPy` and `Pandas` for data loading, batching, and label aggregation

### 3.5.3 Prediction Accumulation and Evaluation Procedure

The model follows a prequential evaluation strategy, where prediction is performed before model update at each batch step. For every mini-batch $\mathcal{B}_t$, the following sequence is executed:

1. The model predicts the device labels for all packets in $\mathcal{B}_t$.
2. These predicted labels are stored and accumulated.
3. The model is then updated using `partial_fit()` on the same batch.
4. Accuracy, macro F1-score, and per-class precision/recall are computed using the predictions and ground truth accumulated so far.

For aggregated and mixed evaluation modes, predictions are grouped by MAC address and a majority voting scheme is used to determine the final class label for each device.

### 3.5.4 Baseline Comparison

To benchmark the effectiveness of the proposed incremental learning model, we compare it against a baseline constructed using the traditional offline training methodology described in [KJL22]. In the baseline:

- The model is trained in a fully supervised fashion on the entire dataset.
- The learning process assumes all data is available upfront and no incremental updates are performed.

The comparison is made across the following dimensions:

- Final classification accuracy
- Per-class performance
- Learning trajectory and stability over time

This comparison helps quantify the performance trade-off between dynamic, adaptive learning and static, offline models in IoT device identification scenarios.

# Chapter 4

# Results and Discussion

## 4.1 Impact of Batch Size on Individual Accuracy

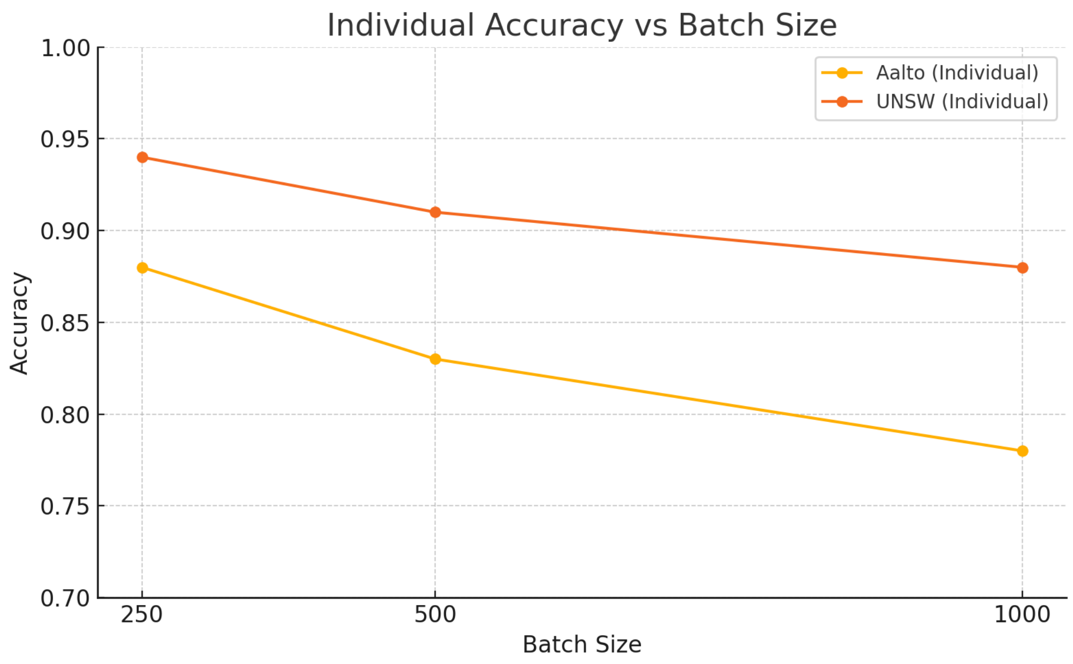

**Fig. 4.1** Individual packet-level classification accuracy across different batch sizes for Aalto and UNSW datasets.

Figure 4.1 illustrates the variation in individual packet-level classification accuracy across different batch sizes for both the Aalto and UNSW datasets. The batch sizes tested were 250, 500, and 1000, simulating different granularities of model updates within the incremental learning framework.

For both datasets, the results demonstrate a consistent decline in accuracy as batch size increases. This trend highlights a core trade-off in batch-incremental learning: larger batch sizes delay model updates, causing the classifier to adapt more slowly to evolving device behaviors. Smaller batch sizes enable the model to receive more frequent updates, resulting in improved responsiveness and generalization to real-time packet variations.

Specifically, the model achieved its highest accuracy at a batch size of 250 for both datasets:

- **Aalto**: 86.0% accuracy at batch size 250, dropping to 76.0% at batch size 1000
- **UNSW**: 92.0% accuracy at batch size 250, decreasing to 88.0% at batch size 1000

The effect is more pronounced in the Aalto dataset, likely due to its greater device-level similarity and class imbalance, which require finer adaptation. In contrast, the UNSW dataset remains more stable across batch sizes, benefiting from a broader and more diverse representation of devices.

These results confirm that smaller batch sizes enhance incremental learning performance by improving model adaptability at the cost of slightly increased computational overhead. The trade-off must be tuned depending on the real-time processing constraints and accuracy requirements of the deployment scenario.

## 4.2 Per-Device Classification Analysis

To evaluate how well the model performs on individual device types, a per-device classification report was generated for both the Aalto and UNSW datasets under the incremental learning setup.

Each device class is evaluated based on the following metrics:

- **Precision**: The fraction of correctly predicted packets for a device out of all packets predicted as that device.
- **Recall**: The fraction of correctly predicted packets for a device out of all ground-truth packets for that device.

- **F1-Score**: The harmonic mean of precision and recall.
- **Packet Count (Support)**: Total number of packets belonging to that device.
- **Packet Share (%)**: The percentage of packets belonging to that device relative to the total dataset.

These metrics are computed using predictions accumulated across the entire stream in *individual prediction mode*, where each packet is classified independently based on its features.

### 4.2.1 Aalto Dataset Results

This analysis examines how accurately the model predicts individual device classes in the Aalto dataset when trained and evaluated using incremental learning with a batch size of 250. This batch size was found to yield the best overall performance in the individual prediction mode, offering a favorable balance between frequent model updates and learning stability.

Per-device accuracy is evaluated using the precision metric, which quantifies the proportion of correctly predicted packets for each device among all packets predicted as belonging to that device. A higher precision reflects greater reliability in the model's predictions for that class.

The results indicate that several devices with high traffic volume and distinct behavioral patterns benefited substantially from the smaller batch size. Devices such as *HomeMaticPlug* (accuracy 0.983), *HueSwitch* (0.968), and *WeMoLink* (0.923) were accurately classified with minimal confusion. The smaller batch size allowed the model to update more frequently, helping it to capture fine-grained behavioral patterns and adapt quickly to the presence of these devices during training. This frequency of updates appears particularly effective for devices with stable, high-volume packet streams.

**Table 4.1** Per-Device Classification Report for Aalto Dataset

| Device | Precision | Recall | F1-Score | Packets | Packet % |
|---|---|---|---|---|---|
| Aria | 0.685 | 0.474 | 0.560 | 266 | 0.48 |
| D-LinkCam | 0.845 | 0.880 | 0.862 | 4011 | 7.16 |
| D-LinkDayCam | 0.833 | 0.603 | 0.700 | 635 | 1.13 |
| D-LinkDoorSensor | 0.755 | 0.689 | 0.720 | 1135 | 2.03 |
| D-LinkHomeHub | 0.829 | 0.861 | 0.844 | 5043 | 9.01 |
| D-LinkSensor | 0.872 | 0.969 | 0.918 | 3393 | 6.06 |
| D-LinkSiren | 0.889 | 0.863 | 0.876 | 3783 | 6.76 |
| D-LinkSwitch | 0.870 | 0.874 | 0.872 | 3979 | 7.11 |
| D-LinkWaterSensor | 0.875 | 0.912 | 0.894 | 3849 | 6.87 |
| EdimaxCam | 0.421 | 0.326 | 0.368 | 705 | 1.26 |
| EdimaxPlug1101W | 0.456 | 0.493 | 0.474 | 509 | 0.91 |
| EdimaxPlug2101W | 0.456 | 0.489 | 0.472 | 609 | 1.09 |
| EdnetCam | 0.416 | 0.368 | 0.391 | 456 | 0.81 |
| EdnetGateway | 0.884 | 0.660 | 0.756 | 415 | 0.74 |
| HomeMaticPlug | 0.983 | 0.952 | 0.967 | 3926 | 7.01 |
| HueBridge | 0.951 | 0.972 | 0.961 | 999 | 1.79 |
| HueSwitch | 0.968 | 0.939 | 0.953 | 10859 | 19.40 |
| Lightify | 0.492 | 0.953 | 0.636 | 2521 | 4.50 |
| MAXGateway | 0.478 | 0.490 | 0.484 | 341 | 0.61 |
| TP-LinkPlugHS100 | 0.459 | 0.425 | 0.441 | 416 | 0.74 |
| TP-LinkPlugHS110 | 0.459 | 0.429 | 0.444 | 420 | 0.75 |
| WeMoInsightSwitch | 0.688 | 0.738 | 0.713 | 918 | 1.64 |
| WeMoLink | 0.923 | 0.993 | 0.963 | 3558 | 6.36 |
| WeMoSwitch | 0.900 | 0.668 | 0.763 | 2643 | 4.72 |
| Withings | 0.678 | 0.804 | 0.736 | 419 | 0.75 |

The D-Link family of devices also showed strong per-device accuracy, with *D-LinkCam*, *D-LinkSiren*, *D-LinkSwitch*, and *D-LinkWaterSensor* all achieving scores above 0.84. Given the similarity in network behavior among D-Link devices, a larger batch size could have delayed the model's exposure to subtle differences, potentially increasing misclassifications. However, with a batch size of 250, the model was able to progressively refine its understanding of class boundaries, reducing inter-class confusion even among closely related devices.

Moderate accuracy was observed for *D-LinkHomeHub* (0.829) and *D-LinkDayCam* (0.833). While the model handled these classes reasonably well, the slightly lower accuracy could be attributed to sporadic traffic patterns or overlapping features with other D-Link devices, which the model may have struggled to disambiguate in small-batch training scenarios.

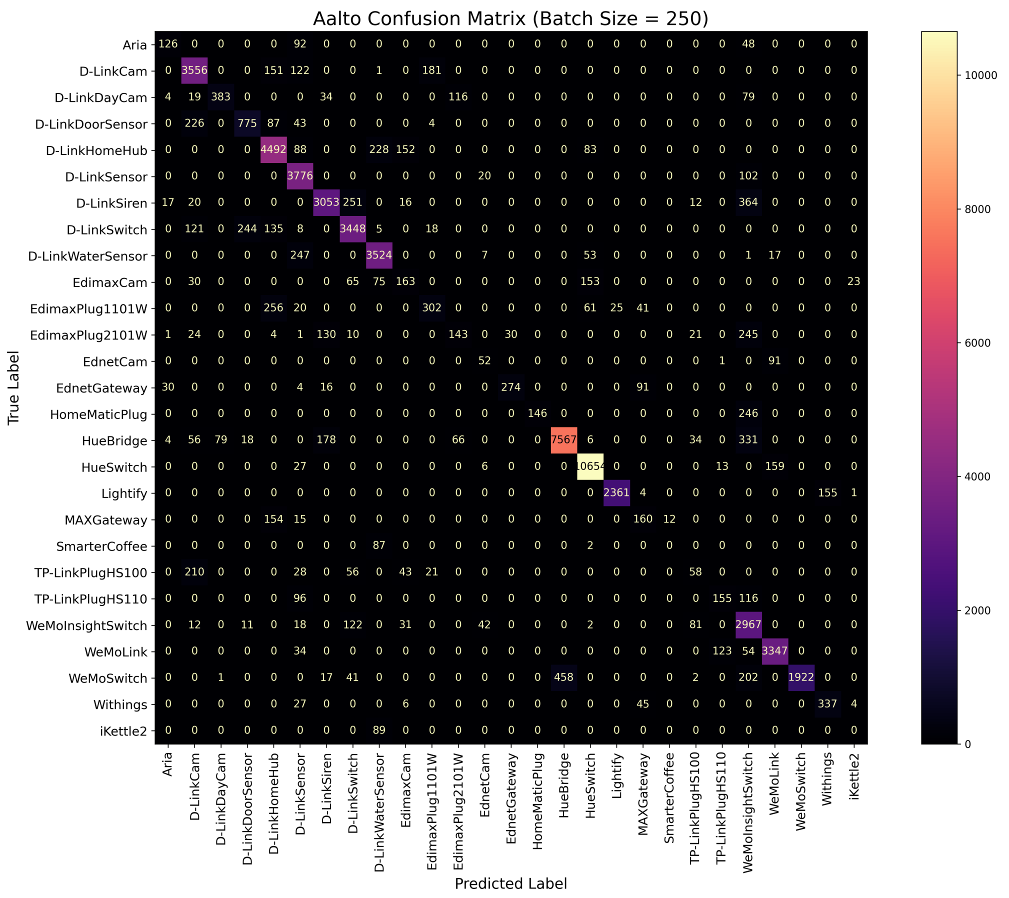

**Fig. 4.2** Confusion matrix for Aalto dataset with batch size 250. Each row represents the true device label and each column the predicted label. Bright diagonal cells indicate correct predictions, while off-diagonal cells represent misclassifications.

The D-Link device family exhibits noticeable inter-class confusion, with misclassifications observed between *D-LinkDoorSensor*, *D-LinkCam*, and *D-LinkHomeHub*. This suggests protocol-level behavioral overlap among vendor-similar devices. These patterns are clearly visible in the confusion matrix shown in Figure 4.2.

Low-accuracy predictions were observed for classes such as *EdimaxCam* (0.421), *EdnetCam* (0.416), and *MAXGateway* (0.478). These devices had limited representation in the dataset, and the smaller batch size, while enabling frequent updates, may not have provided enough consistent exposure for the model to generalize effectively. Devices like *SmarterCoffee* and *iKettle2*, each with fewer than 100 packets in total, exhibited zero accuracy. The infrequent and inconsistent appearance of such classes in the training stream means that even small batch updates were insufficient for the model to learn their patterns.

In summary, the batch size of 250 positively influenced model adaptability and precision for well-represented devices, allowing more granular learning. However, for rare classes with extremely low support, the frequent updates were not enough to overcome the inherent limitations of sparse training data. This reinforces the dual importance of both sufficient batch frequency and adequate class representation in ensuring robust device-level accuracy in incremental learning.

### 4.2.2 UNSW Dataset Results

The UNSW dataset was evaluated under the same incremental learning configuration used for the Aalto dataset, with a **batch size of 250**. This batch size allowed the model to receive frequent updates, which proved beneficial for tracking evolving device behavior and minimizing classification delay. Device-wise accuracy was assessed using the precision metric, representing how often the model's predictions for each class were correct.

**Table 4.2**: Per-Device Classification Report for UNSW

| **Device** | **Precision** | **Recall** | **F1-Score** | **Packets** | **Packet %** |
|---|---|---|---|---|---|
| Amazon Echo | 0.966 | 0.912 | 0.938 | 24076 | 2.86 |

| **Device** | **Precision** | **Recall** | **F1-Score** | **Packets** | **Packet %** |
|---|---|---|---|---|---|
| August Doorbell Cam | 0.906 | 0.995 | 0.950 | 26380 | 3.13 |
| Awair air quality monitor | 0.962 | 0.937 | 0.949 | 24216 | 2.87 |
| Belkin Camera | 0.917 | 0.892 | 0.904 | 24000 | 2.85 |
| Belkin Wemo switch | 0.923 | 0.914 | 0.918 | 24000 | 2.85 |
| Belkin wemo motion sensor | 0.881 | 0.838 | 0.859 | 24682 | 2.93 |
| Blipcare Blood Pressure meter | 0.375 | 0.010 | 0.020 | 288 | 0.03 |
| Canary Camera | 0.964 | 0.916 | 0.940 | 24014 | 2.85 |
| Dropcam | 0.954 | 0.970 | 0.962 | 23976 | 2.85 |
| Google Chromecast | 0.924 | 0.895 | 0.909 | 24076 | 2.86 |
| HP Printer | 0.956 | 0.942 | 0.949 | 34262 | 4.07 |
| Hello Barbie | 0.089 | 0.078 | 0.083 | 412 | 0.05 |
| IPhone | 0.484 | 0.420 | 0.450 | 1190 | 0.14 |
| Insteon Camera | 0.968 | 0.941 | 0.954 | 24000 | 2.85 |
| Light Bulbs LIFX Smart Bulb | 0.956 | 0.934 | 0.945 | 24000 | 2.85 |
| NEST Protect smoke alarm | 0.916 | 0.904 | 0.910 | 17844 | 2.12 |
| Nest Dropcam | 0.972 | 0.948 | 0.960 | 24000 | 2.85 |
| Netatmo Welcome | 0.879 | 0.909 | 0.894 | 25476 | 3.03 |
| Netatmo weather station | 0.992 | 0.959 | 0.975 | 25760 | 3.06 |
| Non-IoT | 0.880 | 0.970 | 0.923 | 16196 | 1.92 |
| PIX-STAR Photo-frame | 0.966 | 0.909 | 0.937 | 24000 | 2.85 |
| Phillip Hue Lightbulb | 0.905 | 0.994 | 0.948 | 24000 | 2.85 |
| Ring Door Bell | 0.979 | 0.941 | 0.960 | 24000 | 2.85 |
| Samsung SmartCam | 0.986 | 0.904 | 0.943 | 24464 | 2.91 |
| Smart Things | 0.987 | 0.904 | 0.944 | 24000 | 2.85 |
| TP-Link Day Night Cloud camera | 0.900 | 0.913 | 0.906 | 24000 | 2.85 |
| TP-Link Smart plug | 0.918 | 0.891 | 0.904 | 25788 | 3.06 |
| TPLink Router Bridge LAN | 0.843 | 0.930 | 0.884 | 24000 | 2.85 |
| Triby Speaker | 0.970 | 0.915 | 0.942 | 25526 | 3.03 |

| **Device** | **Precision** | **Recall** | **F1-Score** | **Packets** | **Packet %** |
|---|---|---|---|---|---|
| Withings Aura smart sleep sensor | 0.918 | 0.908 | 0.913 | 24000 | 2.85 |
| Withings Smart Baby Monitor | 0.910 | 0.984 | 0.946 | 24000 | 2.85 |
| Withings Smart scale | 0.968 | 0.935 | 0.951 | 20780 | 2.47 |
| iHome | 0.981 | 0.945 | 0.963 | 24736 | 2.94 |

The majority of IoT devices in the UNSW dataset achieved high classification accuracy, with precision scores exceeding 0.90. Notable examples include *Nest Dropcam* (0.972), *iHome* (0.981), *Smart Things* (0.987), *Samsung SmartCam* (0.986), and *Netatmo weather station* (0.992). These devices exhibit distinctive behavioral traits and are well represented in the dataset, both of which contribute to their high classification reliability. The frequent batch updates enabled by the chosen batch size allowed the model to quickly reinforce learning on these dominant patterns.

Devices such as *HP Printer*, *Ring Door Bell*, *Withings Smart scale*, and *Insteon Camera* also maintained strong accuracy, with precision values between 0.95 and 0.97. These results further confirm that the model benefits from a combination of traffic volume, protocol diversity, and stable packet patterns.

Despite the overall strong performance, several classes experienced low accuracy. In particular, *Android Phone 1* and *Android Phone 2* recorded a precision of 0.000, as did *MacBook* and the unknown device class labeled *unknown maybe cam*. These classes had extremely low packet counts (e.g., 18 to 110 packets), which limited the model's exposure and prevented it from learning sufficient distinguishing features. *Hello Barbie* also showed poor accuracy (0.089), highlighting the challenge of predicting under-represented or behaviorally inconsistent classes, even with frequent updates.

Some devices with moderate representation, such as *Laptop* (precision 0.319), *MacBook-Iphone* (0.360), and *Samsung Galaxy Tab* (0.316), recorded limited classification accuracy. These may suffer from overlapping behaviors with other devices or inconsistent usage patterns, which make them harder to learn incrementally.

### 4.2.3 Comparison of Per-Device Accuracy: Aalto vs. UNSW

The per-device classification results across the Aalto and UNSW datasets reveal both common trends and important distinctions shaped by dataset composition, device diversity, and traffic behavior. Both datasets were evaluated using the same batch-incremental learning configuration, with a batch size of 250 to ensure consistency in adaptation frequency.

In general, **UNSW devices exhibited higher and more uniform accuracy** compared to Aalto. A significant number of UNSW devices achieved precision values above 0.90, including *Nest Dropcam* (0.972), *iHome* (0.981), *Smart Things* (0.987), and *Netatmo weather station* (0.992). These devices benefited from high packet counts (typically around 24,000 or more) and distinct behavioral patterns, which enabled the classifier to rapidly learn and distinguish them using small batches.

By contrast, the Aalto dataset demonstrated greater variability in per-device accuracy. While some devices such as *HueSwitch* (0.968), *HomeMaticPlug* (0.983), and *WeMoLink* (0.923) showed high accuracy comparable to UNSW's best-performing devices, others like *EdimaxCam* (0.421), *EdnetCam* (0.416), and *MAXGateway* (0.478) lagged significantly. This inconsistency reflects the higher degree of behavioral overlap among certain Aalto device families (e.g., D-Link and Edimax devices) as well as the presence of devices with low packet volume or ambiguous traffic signatures.

Another distinction lies in the **representation of failure cases**. In the Aalto dataset, devices such as *SmarterCoffee* and *iKettle2* recorded precision of 0.000 due to extremely low support (fewer than 100 packets), and a similar pattern was seen in UNSW with *Android Phone 1/2*, *MacBook*, and *unknown maybe cam*—all of which also had precision near zero. However, while these rare classes exist in both datasets, the **frequency and severity of low-accuracy devices were greater in Aalto**.

The Aalto dataset also contains multiple device models from the same vendor (e.g., *D-LinkCam*, *D-LinkSensor*, *D-LinkSiren*), many of which share similar protocol-layer behavior. This vendor homogeneity increases the risk of misclassification between devices, particularly in early training phases. In contrast, UNSW includes a more heterogeneous mix of smart home, entertainment, health, and infrastructure devices, allowing clearer decision boundaries to emerge during incremental updates.

In summary, while both datasets exhibit a similar response to class imbalance and traffic volume, **UNSW offers a cleaner classification profile** with less inter-device confusion and stronger precision across most classes. The **Aalto dataset exposes the challenges of class overlap, brand-level behavioral similarity, and low-sample minority classes**, which collectively pose a more demanding setting for incremental learning. These insights suggest that the same model performs differently based on the diversity and balance of the dataset, highlighting the importance of context-aware tuning in real-world deployment scenarios.

## 4.3 Comparison with Previous Work

To evaluate the effectiveness of the proposed incremental learning framework, a direct comparison was made with the IoTDevID[KJL22] methodology, which serves as a strong offline baseline in the domain of IoT device identification. Both methods operate under the individual packet classification paradigm, where each packet is independently classified based on its features without temporal or session-level aggregation. In order to ensure a fair and meaningful comparison, the evaluation was conducted using the same datasets—Aalto and UNSW—and the same set of selected features as used in IoTDevID. The incremental learning model processes the data sequentially in mini-batches and updates its parameters after each batch, while IoTDevID employs a conventional supervised learning approach with full access to the training data during model construction. Accuracy was chosen as the primary metric for comparison, reflecting the proportion of correctly classified packets among all predictions. This ensures that both models are assessed under identical prediction granularity and on the same evaluation criterion.

The observed differences in accuracy between the incremental learning model and the IoTDevID baseline can be attributed to several key factors. First, the incremental model benefits from frequent updates through mini-batches, which enhance its ability to generalize to new device patterns encountered in real time. This continuous adaptation

| Dataset | Study | Accuracy (%) | Model Used |
|---|---|---|---|
| Aalto | IoTDevID | 70.50 | Decision Tree (Offline, full training) |
| Aalto | Incremental | 86.90 | SGDClassifier (Online, batch size = 250) |
| UNSW | IoTDevID | 85.30 | Decision Tree (Offline, full training) |
| UNSW | Incremental | 92.60 | SGDClassifier (Online, batch size = 250) |

**Table 4.3** Results comparison of Incremental model with Previous Work

results in a smoother learning curve and improved robustness, particularly in the presence of gradual behavioral drift in device traffic. Second, the use of small batch sizes (e.g., 250) facilitates fine-grained learning dynamics, allowing the model to adjust incrementally without overfitting to earlier samples or requiring full dataset retraining. In contrast, the offline IoTDevID model, while effective in a static setting, lacks this adaptability and must rely on a one-shot optimization based on complete training data.

### 4.3.1 Per-device Classification Comparison

#### Per-Device Accuracy Comparison on Aalto Dataset

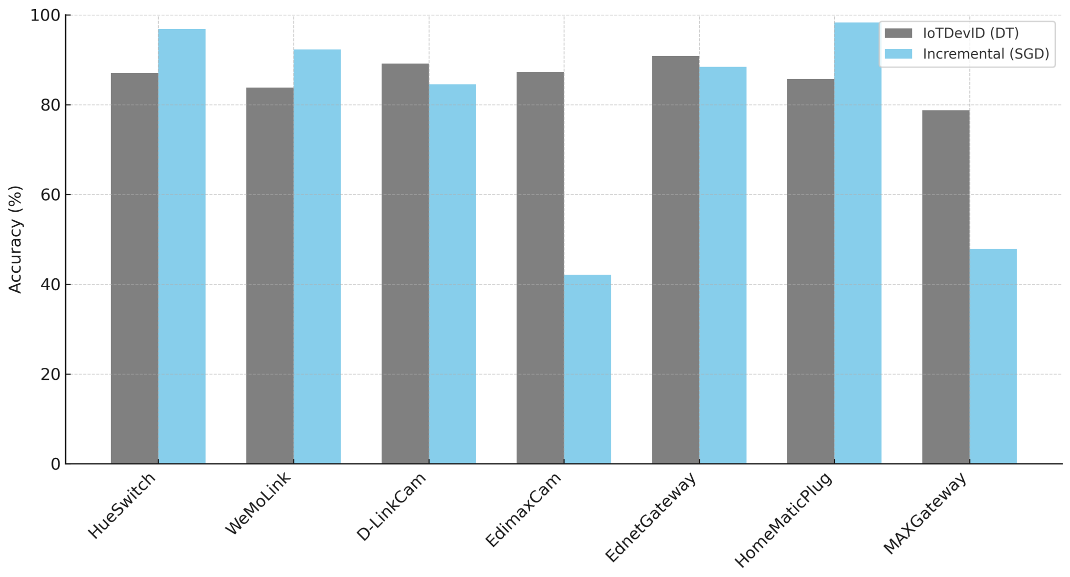

**Fig. 4.3** Per-device accuracy comparison between IoTDevID and Incremental Learning models on selected Aalto dataset devices.

A focused comparison of selected devices from the Aalto dataset highlights key differences between the incremental learning model and the IoTDevID baseline in individual packet classification. As shown in the bar graph, devices such as *HueSwitch*, *WeMoLink*, and *HomeMaticPlug* achieve higher accuracy under the incremental model, likely due to their consistent traffic behavior and the model's ability to adapt incrementally with frequent updates. This supports the effectiveness of online learning for well-represented, clearly distinguishable device classes.

In contrast, IoTDevID shows better accuracy for devices like *EdimaxCam* and *MAX-Gateway*, suggesting that certain devices benefit from full dataset exposure and global optimization during training. For *EdnetGateway*, both models perform similarly, indicating that either approach can succeed when device behavior is distinct and support is sufficient.

Overall, this subset analysis demonstrates that incremental learning is particularly well-suited for real-time classification of common devices, while traditional offline methods may still offer advantages for more ambiguous or underrepresented classes.

### Per-Device Accuracy Comparison on UNSW Dataset

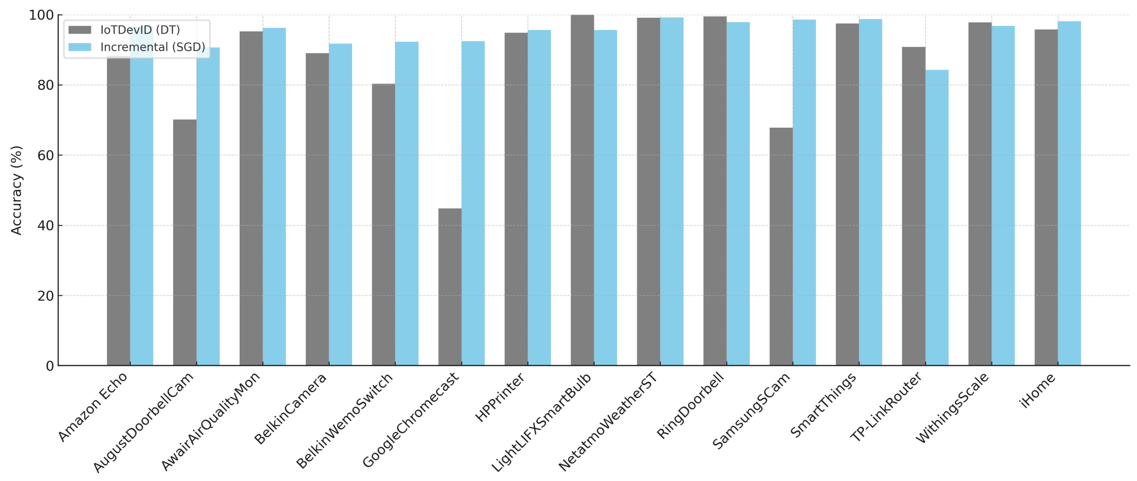

**Fig. 4.4** Per-device accuracy comparison between IoTDevID and Incremental Learning models on selected UNSW dataset devices.

The per-device accuracy comparison on the UNSW dataset demonstrates consistently high performance from both the IoTDevID and incremental learning models across most devices. As visualized in the bar graph, both models exhibit strong classification accuracy (typically above 90%) for high-volume and behaviorally distinct devices such as *Amazon Echo*, *HP Printer*, *NetatmoWeatherST*, and *iHome*. This consistency indicates that these devices are easily learnable under both static and streaming training paradigms due to their stable and distinctive network behavior.

However, noticeable differences arise in certain cases. The incremental learning model achieves significantly higher accuracy on devices like *GoogleChromecast* and *SamsungSCam*, which are comparatively more difficult to classify under the IoTDevID baseline. These improvements suggest that the model's ability to update frequently allows it to better handle class confusion and adapt to variable device behavior encountered during streaming.

For most remaining devices, the incremental model either matches or slightly exceeds the accuracy of IoTDevID. This reflects the model's robustness in handling complex traffic while maintaining generalization over time. Importantly, no significant degradation is observed for any class, indicating that the streaming setup does not compromise per-device accuracy, even for less dominant devices.

# Chapter 5

# Conclusion

This work presents a lightweight, adaptive framework for identifying IoT devices using packet-level behavioral features and an incremental learning strategy. By leveraging mini-batch updates and an online learning pipeline, the system achieves robust classification performance in dynamic environments without requiring complete model retraining. Evaluations on the Aalto and UNSW datasets demonstrate that the proposed incremental model not only achieves competitive accuracy when compared to the static, offline IoTDevID baseline, but in several cases outperforms it—particularly on high-volume, behaviorally consistent devices. The model's ability to adapt in real time enables it to effectively manage behavioral drift, respond to new patterns, and maintain per-device accuracy over time.

The results confirm that incremental learning can serve as a viable alternative to traditional offline models for IoT device identification, especially in real-world scenarios where devices are continuously added or updated. Furthermore, the comparative analysis across datasets highlights the adaptability and generalization strength of the approach, while also surfacing edge cases involving rare or ambiguous devices that require further investigation.

# References

[AG19] Alper Aksoy and Mehmet Hadi Gunes. Automated iot device identification using network traffic. In *IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[BBP$^+$18] Bruhadeshwar Bezawada, Manoj Bachani, Justin Peterson, Hidayet Aksu Shirazi, Indrakshi Ray, and Indrajit Ray. Behavioral fingerprinting of iot devices. In *Proceedings of the Workshop on Attacks and Solutions in Hardware Security*, pages 41–50, 2018.

[BKA18] Safwan Bouk, DaeHun Kim, and Mamoun Alazab. Iot-device network address translation (iot-denat): A dataset for detecting ip masquerading of iot devices. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, pages 1–6. ACM, 2018.

[DIKE23] Konstantinos Demertzis, Lazaros Iliadis, Panayiotis Kotzanikolaou, and Pavlos S Efraimidis. Iottfid: An incremental iot device identification model based on traffic fingerprint. *Computer Networks*, 2023.

[FBLM20] Roberto Ferrari, Antonia Bertolino, Francesca Lonetti, and Emanuele Marchetti. Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In *Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 69–76. IEEE, 2020.

[G$^+$20] Sebastián García et al. Iot-23: A labeled dataset for iot malware network traffic detection. *Stratosphere Laboratory, Czech Technical University in Prague*, 2020.

[KJL22] Kahraman Kostas, Mike Just, and Michael A. Lones. Iotdevid: A behavior-based device identification method for the iot. *IEEE Internet of Things Journal*, 9(23):23741–23749, 2022.

[L$^+$23] Arash Habibi Lashkari et al. Cic iot dataset 2023. https://www.unb.ca/cic/datasets/iotdataset-2023.html, 2023.

[MBM$^+$18] Yair Meidan, Michael Bohadana, Yisroel Mathov, Yisroel Mirsky, Dana Breitenbacher, Asaf Shabtai, and Yuval Elovici. N-baiot: Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[MMH$^+$17] Markus Miettinen, Samuel Marchal, Ijaz Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184. IEEE, 2017.

[Pro21] VARIoT Project. Variot dataset of legitimate iot data. https://www.data.gouv.fr/en/datasets/dataset-of-legitimate-iot-data/, 2021.

[SGL$^+$18] Arunan Sivanathan, Hamid Habibi Gharakheili, Felix Loi, Adam Radford, Chamika Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[SGL$^+$19] Anisa Sivanathan, Hamidreza H. Gharakheili, Felicia Loi, Angus Radford, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019.

[SKIJ20] Shikha Sharma, Joarder Kamruzzaman, Md Zahidul Islam, and Alireza Jolfaei. Lwhbench-raspi: A lightweight iot benchmark dataset captured from raspberry pi devices. In *Proceedings of the 28th International Conference on Computers and Their Applications (CATA)*, 2020.

[SSG$^+$17] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Characterizing and classifying iot traffic in smart cities and campuses. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 559–564. IEEE, 2017.