UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Scalable Network Fingerprinting for IoT Devices

by

Tadani Nasser Alyahya

ORCiD: 0000-0001-8570-5445

A thesis for the degree of
Doctor of Philosophy

December 2024

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

Scalable Network Fingerprinting for IoT Devices

by Tadani Nasser Alyahya

Recognising IoT devices through network fingerprinting contributes to enhancing the security of IoT networks and supporting forensic activities. Network fingerprinting for IoT devices involves analysing the traffic from these devices to accurately identify them without relying on explicit identifiers within the transmitted packets, which can be spoofed. Machine learning techniques have been extensively utilised in the literature to optimise IoT fingerprinting accuracy. Given the rapid proliferation of new IoT devices, a current challenge in this field is around how to make IoT fingerprinting scalable, which involves efficiently updating the used machine learning model to enable the recognition of new IoT devices. Some approaches have been proposed to achieve scalability, but they all suffer from limitations like large memory requirements to store training data and accuracy decrease for older devices.

In this research, we propose a novel, scalable network fingerprinting method for IoT devices that leverages online stream learning and fixed-size session payloads. This approach enables the model to be updated periodically without needing to retain data, ensuring scalability and maintaining high recognition accuracy. Moreover, our method includes a mechanism for detecting unknown IoT devices.

Our contributions are multifaceted, beginning with a comprehensive survey of passive IoT device fingerprinting that leverages machine learning and network characteristics, systematically reviewing the literature and detailing the network traffic features used for device identification. We identify key open research problems and future directions in this domain, highlighting significant challenges and gaps. A notable advancement is the introduction of ScaNeF-IoT, a scalable IoT fingerprinting approach utilising online stream learning and fixed-size traffic payload sessions, demonstrating high accuracy and adaptability. The scalability of the approach lies in its ability to continuously update the machine learning model with minimal resource overhead, allowing for the seamless recognition of new IoT devices without retraining from scratch. We further investigate feature extraction method, which indicates the instances of interest from network traffic, such as packets, flows, or sessions, for

further analysis and feature extraction, finding that fixed-size payload sessions outperform others with an accuracy of over 99.5% and an average false positive rate of 2.25%. Additionally, our scalable system is able to detect unknown IoT devices using online stream learning and z-score analysis, showcasing efficiency and adaptability. Our scalable IoT device fingerprinting approach achieves 100% accuracy in detecting unknown devices and 94% average accuracy in identifying known devices in streaming data.

Contents

List of Figures

List of Tables

List of Algorithms

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

Signed: Tadani Alyahya Date: 19 Dec 2024

Acknowledgements

All praise and thanks are due to Allah, the Almighty, whose infinite mercy and guidance have been my constant companions throughout this journey. Without His blessings, none of this would have been possible. I am eternally grateful for His grace and wisdom, which have illuminated my path and strengthened my resolve every step of the way.

I would also like to extend my sincere thanks to my examiners, Dr. Mohamed Bader-El-Den and Dr. BooJoong Kang. Their thorough evaluation, constructive feedback, and insightful suggestions have been crucial in refining and enhancing the quality of this work. I deeply appreciate the time and effort they have invested in this process.

I am profoundly grateful to my first supervisor, Dr. Leonardo Aniello, for his exceptional insights and guidance. His expertise and support have been instrumental in shaping this work. I am also deeply thankful to my second supervisor, Prof. Vladimiro Sassone, for his continuous encouragement and unwavering support.

I want to express my deepest gratitude to my parents for their endless prayers throughout this journey. My appreciation extends to my siblings, especially Rea'am, whose presence has been a constant source of strength and comfort despite the distances that separate us. The bond we share has been a continuous source of inspiration. I am equally grateful to my sister, Muneerah for her unwavering support and belief in me.

I am truly speechless when it comes to expressing my gratitude to my true love, Abdullah. His sacrifices and steadfast support throughout my years of study have meant the world to me, and his faith in me has been a cornerstone of my success. Finally, to my precious children, Sara and Saud, you are my joy and my rock. Your encouragement, whether cheering me on with "Mum, you can make it!" or helping me catch the earliest bus to Uni, has filled my journey with love and motivation. Every day, you inspire me to strive harder and remind me why I embarked on this path. This achievement would not have been possible without all of you.

To my beloved husband Abdullah and our precious children,
Sara & Saud

Definitions and Abbreviations

ACK	Acknowledgement
AdaBoost	Adaptive Boosting
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AMF	Aggregated Mondrian Forest
ANOVA	Analysis of Variance
ARF	Adaptive Random Forest
BoW	Bag of Words
CE	Cross-Entropy
CFS	Correlation-based Feature Subset
CIL	Class Incremental Learning
CNN	Convolutional Neuron Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DF	Don't Fragment
DNS	Domain Name System
DPI	Deep Packet Inspection
DT	Decision Tree
EL	Ensemble Learning
ET	Extra Trees
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GA	Genetic Algorithm
IAT	Inter Arrival Time
ID	Identification Domain
IG	Information Gain
IIoT	Industrial IoT
IoT	Internet of Things
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IQR	InterQuartile Range

kNN	k-Nearest Neighbours
LSTM	Long Short-Term Memory
MAC	Media Access Control
ML	Machine Learning
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
OEL	Online Ensemble Learning
ORF	Online Random Forests
OS	Operating System
OSL	Online Stream Learning
PCA	Principal Component Analysis
PCAP	Packet Capture file
q1	First quartile
q2	Second quartile
q3	Third quartile
RF	Random Forest
RFID	Radio-Frequency Identification
SHN	Smart Home Network
SMOTE	Synthetic Minority Oversampling Technique
SYN	Synchronise
TCP	Transmission Control Protocol
TF-IDF	Term Frequency-Inverse Document Frequency
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TTL	Time To Live
t-SNE	t-Distributed Stochastic Neighbour Embedding
UNSW	University of New South Wales
VPN	virtual private network
xverse	X uniVerse
YAF	Yet Another Flowmeter tool

Chapter 1

Introduction

Security and forensics in IoT networks have become prominent nowadays, as IoT applications are pervasive in many environments, such as smart homes, university campuses and enterprise networks. Among the existing IoT security and forensics techniques, *network fingerprinting* allows to recognise individual IoT devices by analysing their traffic. More precisely, network fingerprinting of IoT devices (hereinafter, *IoT fingerprinting*) analyses the traffic of a device and estimates the identity of the device itself, without relying on any identifier it might include in the packets it sends. Malicious or suspicious activities can be detected by comparing this estimate with the device identity associated with the identifier reported in the traffic (e.g., its MAC address). For example, IoT fingerprinting allows security practitioners to detect unknown malicious devices that are spoofing the MAC or IP address of legit devices to avoid suspicion. Additionally, authorised devices infected by a malware might start unusual network activities, which can prevent an IoT fingerprinting mechanism from recognising these devices and, consequently, enable security practitioners to identify them as suspicious.

IoT fingerprinting commonly involves passively analysing network traffic, without any direct interaction with the devices, which could otherwise allow an attacker to understand that their behaviour is being analysed. Many techniques have been proposed and developed for IoT device fingerprinting, which leads us to the following research questions:

RQ1. What approaches have been proposed in the literature for passive IoT device fingerprinting?

Most IoT fingerprinting approaches employ machine learning (ML) techniques to train models to recognise known IoT devices based on features extracted from network traffic. Several studies have shown the effectiveness and accuracy of passive IoT fingerprinting based on ML (1; 2; 3). Most of these studies rely on supervised

learning (4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17), where a classifier is trained to recognise a set of IoT devices using a dedicated class for each device. Other ML approaches used for IoT fingerprinting are based on semi-supervised (18; 1; 19; 20) and unsupervised (21; 22) techniques. While these IoT fingerprinting techniques offer significant promise for enhancing network security, they are also accompanied by inherent limitations and challenges, where we pose the following research question:

RQ2. Which specific limitations and challenges exist in the literature regarding IoT fingerprinting techniques, and how do these affect the accuracy and scalability of IoT device identification?

The literature has several key open research problems, including IoT device diversity and imbalance issues in IoT datasets, improving the robustness of the IoT fingerprints, and enhancing IoT device fingerprint scalability. Marchal *et al.* (23) define the scalability of IoT fingerprinting as the ability "to manage a large number of IoT device types and learn to identify new types as they emerge''. An important trend in the IoT field is the staggering and relentless increase of active IoT devices (24). When an ML-based IoT fingerprinting mechanism is used, the classifier should be retrained often to ensure new devices can be recognised accurately. However, this aspect is largely neglected in literature. In fact, most of the IoT fingerprinting approaches proposed in literature are designed and evaluated to handle a fixed number of devices, without assessing their behaviour when new devices are introduced (25), i.e., without assessing their scalability.

A few approaches for scalable IoT fingerprinting have been proposed in literature. Some use *a binary classifier for each device* and introduce further binary classifiers as training data for new IoT devices becomes available. This approach requires an ever growing number of classifiers, which might lead to memory exhaustion. Also, additional mechanisms are needed to break ties when more classifiers return a positive outcome for the same device. Furthermore, the accuracy of this approach tends to degrade over time because existing classifiers are never retrained to ensure they can properly distinguish the devices introduced latest. Other approaches are based on *Class Incremental Learning* (CIL), where the classifier is retrained using both data for new IoT devices and a selection of data for known devices. As time goes by, the selection of data for known devices needs to be capped in size for memory/disk constraints, which implies fewer samples for each device can be retained and, therefore, the classifier accuracy tends to worsen for older devices; this phenomenon is commonly referred to as catastrophic forgetting. To address the scalability of IoT device fingering, we draw the following research question:

RQ3. How can we develop a scalable IoT fingerprinting approach to improve device recognition accuracy and memory consumption?

We introduce *ScaNeF-IoT*, a novel approach for scalable network IoT fingerprinting. It relies on *Online Stream Learning* (OSL) based classifier to achieve scalability enabling on-the-fly retraining without memory exhaustion. Furthermore, it uses fixed-size traffic session payloads as features to feed the classifier, which limits the time needed to fingerprint a session and generally speeds up the detection of unusual network activities. Our approach implements Aggregated Mondrian Forest (AMF) as an OSL classifier and achieved comparable accuracy with AutoIoT (25) and outperform IoT-Portrait (26) using UNSW IoT Traces dataset (12). Our findings indicate that OSL could be an effective scalable solution for IoT fingerprinting. While this is a preliminary evaluation, many other features can be examined to enhance fingerprinting performance. Therefore, we pose the following research question:

RQ4. What features can provide the best performance for fingerprinting IoT devices?

We examine different feature extraction methods, including, packet-based, flow-based, and session-based. Selecting the right features can significantly enhance the accuracy of device identification, reduce error rates, and improve overall system performance. Our evaluation using AMF shows that the payload of size 784 bytes extracted from sessions performs the best in terms of accuracy and error rate using UNSW IoT Traces dataset. However, it is important to note that these features may not consistently offer optimal performance across different ML approaches, as various algorithms may leverage or prioritize features differently.

different machine learning (ML) algorithms indeed respond differently to various types of features.

As the IoT continues to expand, scalability approaches concentrate on the identification of known devices; however, detecting unknown or new devices poses a significant challenge. AutoIoT applies the Kolmogorov–Smirnov test, which necessitates having access to the data distribution of known IoT devices each time it needs to be compared with the data distribution from an unknown IoT device. This approach imposes a significant demand on memory resources. This limitation leads us to ask the following research question:

RQ5. How can we accurately detect unknown IoT devices in a scalable IoT device fingerprinting system while reducing memory consumption?

We propose the application of the z-score measurement in our scalable IoT fingerprinting system. This technique quantifies the extent to which a device's behaviour deviates from those of known devices. It leverages the incoming data maximum probability from the trained OSL model and calculates the z-score accordingly, using the mean and standard deviation (sd) of the known distribution.

The z-score value will be used to determine whether it is known or unknown by comparing it with a predefined threshold. The performance evaluation for detecting known IoT devices, using AMF and fixed-size payload sessions from the UNSW IoT Traces dataset, reveals accuracy comparable to AutoIoT, along with the added benefit of decreased memory consumption. Our approach needs only to retain the mean and sd of the known distribution. We conduct a further evaluation of the overall system using the same dataset by periodically updating the AMF model along with the mean and sd of the known distribution. The results consistently indicate that fixed-size payload sessions surpass those based on other features in performance, with the scalable IoT fingerprinting model achieving high accuracy in identifying known IoT devices and detecting unknown ones.

The Thesis contributions are:

1. a comprehensive study in the area of passive IoT device fingerprinting based on ML and network characteristics over the period 2017-2023, which, to the best of our knowledge, is the first academic review that solely focused on this aspect. It proposes a taxonomy of ML-based approaches for passive IoT device fingerprinting. It provides a detailed study of network traffic features that can be used for generating a device fingerprint and reports the IoT traffic features utilised by the literature. We prepare a draft of the survey to be submitted to ACM Computing Surveys. (Chapter 3)

2. identification of key open research problems, challenges, and further directions in the area of IoT device fingerprinting. (Chapter 3)

3. a novel approach, ScaNeF-IoT, for scalable IoT fingerprinting (27), based on OSL and fixed-size session payloads. We implement the approach using Aggregated Mondrian Forest (AMF) and UDP/TCP payloads and evaluated it using IoT Traces dataset (12). We compare the accuracy of our approach with two other scalable IoT fingerprinting approaches, AutoIoT (25) and IoT-Portrait (26), showing comparable results with the former and outperforming the latter. This contribution is published in *The 19th 2024. International Conference on Availability, Reliability and Security (ARES 2024)* (27). (Chapter 4)

4. a novel method that explores various feature extraction types to create unique fingerprints for IoT devices, employing OSL for IoT device fingerprinting. We implement a feature extraction from packet-based, flow-based, sessions-based. We evaluate each feature extraction type using AMF and the UNSW IoT Traces dataset, by comparing their accuracy, false positive rate, and time efficiency. The results show that fixed-size session payloads extracted from the session-based outperform the others in IoT device fingerprinting with over 99.5% accuracy and 2.25% false positive rate. (Chapter 5)

5. a novel scalable network fingerprinting with unknown IoT device detection, based OSL and z-score using fixed-size session payloads. The implementation and evaluation of unknown IoT device detection on the UNSW IoT Traces dataset show comparable accuracy with AutoIoT with improved resources. The evaluation of scalability through updating the model implementation also shows comparable results with AutoIoT, featuring on-the-fly updates without retraining requirements. (Chapter 6)

6. a novel application of the scalable network fingerprinting system with unknown IoT device detection on periodic streaming data. To the best of our knowledge, this is the first research to demonstrate scalable IoT device fingerprinting on stream data. The evaluation consistently shows the approach based on payload session features outperforms those based on other features with an average accuracy of over 94% in identifying known IoT devices and achieves 100% accuracy in detecting unknown IoT devices from just a single payload session. This showcases not only efficient resource management but also high accuracy in operation on the conducted dataset. (Chapter 6)

The rest of the report is as follows: Chapter 2 presents the background. Chapter 3 presents a comparative survey on IoT device fingerprinting using passive network scanning and Machine Learning. Chapter 4 demonstrates our proposed approach, ScaNeF-IoT, for scalable network fingerprinting for IoT devices. Chapter 5 introduces an analytical study of various feature extraction types to find the features for fingerprinting IoT devices. Chapter 6 introduces our approach for scalable network fingerprinting with unknown IoT device detection. Finally, conclusions and future works are presented in Chapter 7.

Chapter 2

Background

2.1 Introduction

This Chapter provides an overview of several key concepts essential for understanding this research, particularly within the IoT. Section 2.2 explores the concept of IoT, the interconnected devices via the Internet, enabling them to send and receive data. Section 2.3 further delves into IoT fingerprinting, an essential technique for ensuring the security and integrity of IoT networks by identifying and categorising devices based on their unique network traffic patterns. Additionally, the discourse extends to ML and its critical role in interpreting the vast amounts of data generated by IoT devices in Section 2.4. The discussion on ML transitions Section 2.5 into Online Machine Learning, a dynamic adaptation of traditional ML techniques designed to process and learn from data in real-time, is an essential capability in the constantly evolving IoT system. Section 2.6 covers the fundamental understanding of the standard score (z-score), a statistical measure used to gauge data variability and manage outliers effectively. Finally, a summary of this chapter is presented in Section 2.7.

2.2 Internet of Things

The term *"Internet of Things"* was invented by Kevin Ashton in 1999 to describe the Internet-connected cyber-physical systems via sensors, including RFID (Radio-frequency identification Offsite Link) (28). Numerous definitions of the Internet of Things (IoT) have been proposed, including one notable that emphasises the connectivity and sensory demands of components in standard IoT settings (29). The technological advancements, such as cheaper sensors, and widespread internet access, have facilitated the affordability, efficiency, and functionality of IoT devices,

leading to their extensive use across various industries and consumer applications (30).

The pace of interconnected IoT devices is increasing rapidly. According to a Vailshery (24) report, there will be more than 29 billion connected IoT devices worldwide in 2030, with the consumer sector expected to dominate with an estimated 17 billion IoT devices (31). This growth is driven by the impact of IoT on the devices market by facilitating communication and data analysis to provide services and applications (32). IoT has a wide range of applications across different sectors including homes, healthcare, industries, and agriculture, as illustrated in Figure 2.1.

- In healthcare, IoT aids through wearable technologies that monitor patient health metrics and provide critical data to medical professionals, improving the management of chronic conditions and enhancing patient care (32; 33).
- In consumer electronics, smart home technology has revolutionised people's lives. Devices enabled by IoT, such as smart speakers, thermostats, lighting systems, smart meters, and security cameras, are increasingly popular. These devices allow users to manage their homes using voice commands or mobile applications to ensure smart living for people (32; 33).
- In industrial, the Industrial IoT (IIoT) technology is employed through the use of sensors, data analytics, and automation, IIoT boosts efficiency, facilitates predictive maintenance, and improves the use of resources (32; 33).
- In smart cities, IoT technology enhances urban infrastructure, optimizes resource management, and improves the quality of life for residents. IoT devices and sensors are integrated across various city systems, enabling cities to become more interactive and responsive in different technologies, such as traffic management, waste management, energy distribution, and public safety systems (32; 33).
- In agriculture, integrating IoT into agriculture, farmers can achieve higher operational efficiency, reduce wastage, and significantly increase the sustainability of agricultural practices. This smart farming approach involves the use of connected devices and sensors to monitor and manage agricultural operations such as greenhouse automation, crop health monitoring, and precision farming (33).

Even though the evolution of IoT involves many opportunities and the benefits are undeniable, there are challenges in the deployment and maintenance of IoT systems. Security is a significant concern, as the proliferation of IoT devices increases the risks of data breaches and privacy issues. Interoperability is essential; establishing common standards and protocols ensures that devices from various manufacturers can

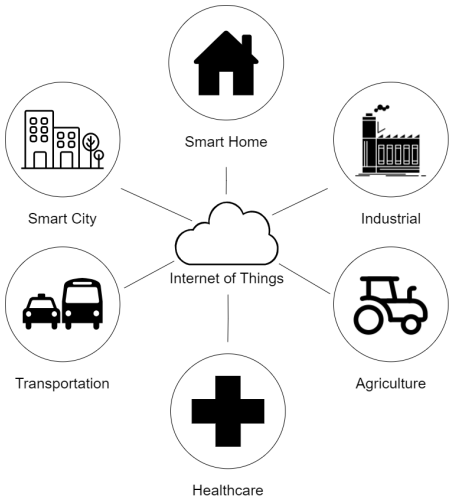


FIGURE 2.1: IoT application

seamlessly communicate. Scalability and management challenges arise as the number of IoT devices grows, complicating the deployment and operation of large-scale solutions. Addressing these areas effectively is vital to meet the complex requirements of IoT systems and ensure their sustainable growth and security. There are essential requirements that should be part of any IoT implementation, illustrated in Figure 2.2, which are (33):

- **Identification.** The capability of identifying all objects connected to a network.
- **Scalability.** The ability of the IoT environment to deliver, manage and maintain its services efficiently even with the growth of connected objects.
- **Security.** Security is a crucial requirement for IoT systems to protect data and prevent unauthorised access. Effective security in IoT involves using strong encryption for data transmissions, regularly updating software to fix vulnerabilities, and setting stringent access controls to ensure data integrity and confidentiality.
- **Energy efficiency.** The ability to minimise energy enables the IoT objects a long life span.
- **Data management.** Ensure that the data generated from an IoT environment can be efficiently retrieved at any time.
- **Interoperability.** The ability to standardise the communication between different IoT objects, that differ in their capabilities such as bandwidth and energy.
- **Self-organising capability.** IoT objects should be intelligent enough to manage themselves and act autonomously in any situation.

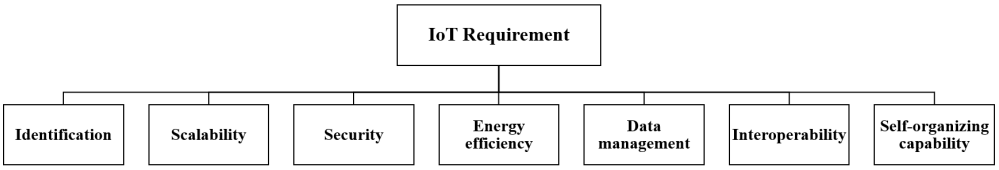


FIGURE 2.2: IoT requirements (33)

2.3 IoT Device Fingerprinting

IoT device fingerprinting involves creating a unique identifier for a device based on its network traffic (3; 34). This identifier, or *fingerprint*, enables device recognition across a network. Shridhar and Nagaveni (35) highlighted the importance of fingerprinting in enhancing network security. Following this, it's pertinent to explore the methodologies employed in device fingerprinting, specifically the *Active* and *Passive* approaches, each of which will be discussed in detail in the following sections.

2.3.1 Active IoT Fingerprinting

Active fingerprinting involves directly interacting with the target device to identify its presence based on its responses. Queso was introduced in 1990 and is considered one of the first tools for active fingerprinting (36). It sends Transmission Control Protocol (TCP) packets that deviate from the protocol's specifications, resulting in various responses from target devices. Another well-known tool, Nmap, is used for discovering operating systems and services (37). Additional fingerprinting tools include Xprobe (38) and SinFP (39) for OS identification, and Scanrand for network scanning (40).

There have been many studies based on probing the network for device identification (e.g.,(41; 42; 43; 44)). Although this method provides additional insights about the devices in the network, it can be detected and potentially blocked by targeted devices, which may also manipulate responses to mislead the fingerprinting process (18). This detectability introduces a vulnerability, as attackers can identify the fingerprinting activity and either avoid detection or engage in deceptive practices to evade security measures (18).

2.3.2 Passive IoT Fingerprinting

Passive fingerprinting relies on analysing the traffic that devices naturally emit, which means it is less intrusive and reduces the risk of being detected by target devices. This method is generally safer in terms of not provoking defensive or deceptive responses

from devices (45; 18). Moreover, it can be used with any network-connected device, allowing administrators to detect device presence, monitor activities, identify misuse, and enforce security policies effectively (46; 47).

Many researchers have explored passive device fingerprinting, providing network administrators with extensive data to enhance technologies for network management, authentication, and intrusion detection. The first exploration of individual device fingerprinting was conducted in 2005 by Kohno *et al.* (48), which utilises the TCP timestamp option to estimate a device's clock skew. This approach, which requires deep packet inspection, applies only to devices having the TCP timestamp option enabled. Gao *et al.* (47) apply wavelet transformation to signals derived from Inter Arrival Time (IAT) features for device identification. They utilise circular cross-correlation to assess the similarity between the target Access Point (AP) and a set of master signatures. However, this technique is specifically designed for APs and does not extend to endpoint IoT devices. Noguchi *et al.* (49) approach involves using the changing time patterns of feature amounts, which are acquired from transmitted signals, to identify individual devices. This method was further evaluated in (50), where it automatically identified the device's type and model based on the similarity of communication features. This technique specifically targets IoT devices specialised for particular functions, such as network cameras. Le *et al.* (51) present a solution for identifying the vendor and type of IoT devices by utilising Natural Language Processing (NLP) techniques. This process involves converting Domain Name System (DNS) names into words and then applying Term Frequency-Inverse Document Frequency (TF-IDF) for identification.

Machine learning (ML) has been recently adopted in device fingerprinting (52; 53). Subsequent studies have demonstrated the effectiveness of applying ML for identifying IoT devices, achieving high accuracy in constructing intelligent identification systems (52; 53). ML techniques are utilised to classify network flows, including encrypted traffic, to recognise individual devices and their types, such as vendor and model (34; 54). This approach provides more comprehensive information, proving advantageous for network management compared to other methods. A more detailed explanation of related studies will be provided in Chapter 3.

2.4 Machine Learning

ML has received several formal definitions in the literature. Arthur Samuel presented a general definition in 1959 that states ML as the "field of study that gives computers the ability to learn without being explicitly programmed" (55). A more precise definition presented by Géron is "the science (and art) of programming computers so they can learn from data" (56). ML is employed in complex problems where

traditional approaches are insufficient, often requiring an extensive set of rules, or in cases that cannot be resolved through traditional methods (56). ML is broadly categorised according to the nature of the data labelling into *supervised learning*, *unsupervised learning*, and *semi-supervised learning*, as illustrated in Figure 2.3 (57). Supervised learning trains samples of data to estimate an unknown instance, that happens to be one of the known samples, called *labels* (56). Unsupervised learning tries to learn without labelling (57). *Semi-supervised learning* deals with partially labelled training data (56).

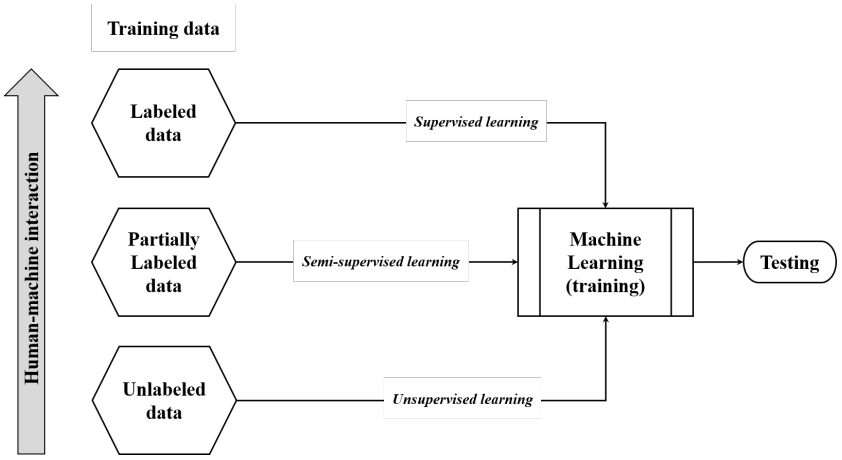


FIGURE 2.3: Categories of machine learning algorithms according to the nature of the data labelling (57).

Traditional ML methods typically employ *batch learning* or offline learning approaches, particularly in supervised learning scenarios (58). In batch learning, a model is trained all at once using a complete set of training data through a specific learning algorithm as shown in Figure 2.4a. Once training is complete, the model is deployed for inference and generally remains static, without updates post-deployment. The batch learning method incurs high re-training costs when new data is introduced, making them less scalable for practical applications. The emergence of big data made the traditional batch learning approaches increasingly constrained, particularly as live data rapidly expands and evolves. Therefore, enhancing ML to efficiently handle ongoing data streams remains a significant and open challenge in the fields of ML and Artificial Intelligence (AI) (58).

Many ML classifiers flow the batch learning method, including Decision Trees, Ensemble Learning, and Artificial Neural Networks. The key is to choose the right classifier for the exact problem. Several factors should be considered such as data size, quality and diversity, accuracy, and time complexity (59).

Decision Tree (DT). It is a supervised ML model that constructs a DT to predict outcomes based on decision rules derived from training data analysis. In this model, the branches represent conjunctions of features leading to a predicted class (56). DT

are powerful and easily interpretable methods. Although they are traditionally sensitive to variations in the training set, such as rotation (56).

Ensemble Learning (EL). It is a single model (strong learner) that combines multiple classifiers (weak learners) to generate better results than an individual classifier (56). *Random Forest* (RF) is an example of EL that trains numerous DT models, where each has a randomly chosen subset training set to find optimal split (56). RF is known for its simplicity powerful and robustness of overfitting (i.e., the model performs well on training data but performs poorly on new data) (56; 9). Another example is Extra Trees (ET), similar to RF, but uses a random tree-splitting method rather than searching for the best split, making it significantly faster by combining randomisation with optimisation (choosing the best split) (9). Therefore, it is much faster than RF as it combines randomisation and optimisation (choose the best split). For that reason, it is called Extremely Randomised Trees and Extreme Random Forests. *Boosting*, another EL technique, involves DTs that are grown sequentially, with each model evolving from its predecessor (56). It particularly focuses on instances that were incorrectly classified by previous models, which are then prioritised in the subsequent model. This process continues until no further improvements can be made or predetermined criteria are met. *Adaptive Boosting* (AdaBoost) is a notable technique within this category, known for enhancing the accuracy of weak classifiers through a focus on correcting errors in predecessor models. AdaBoost trains using weighted instances and continues adjusting until specified conditions are fulfilled (9). All these techniques flow the batch learning method.

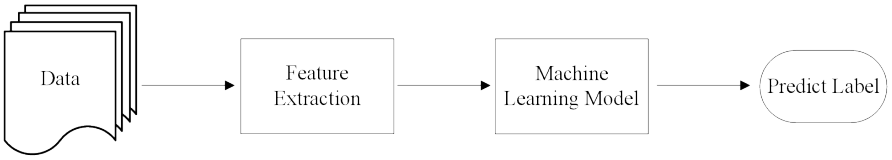
Neural Networks and Deep Learning (DL). An artificial neural network (ANN) consists of interconnected processing layers (60). This technique gains attention as it is inspired by the biological system (60). ANN can learn the right behaviour in a supervised manner or can analyse data automatically to reflect its output as unsupervised learning (61). Its main advantage is the ability to model complex non-linear relationships in high-dimensional data (56; 62). Yet, it has many limitations, including (but not limited to) is inexplicable, which is untrustworthy, and requires sophisticated computational resources (62).

The core of DL is ANN and ML, which comprises multiple layers of non-linear neurons to automatically learn representations with multiple abstraction levels (56; 63). It does not require feature engineering that involves human intervention. Instead, the features are learned from data using general-purpose learning (64). Its prediction depends on a large amount of training data that consumes time (56; 59). DL has achieved huge success in complex problems to process images, speech and natural languages (56; 63).

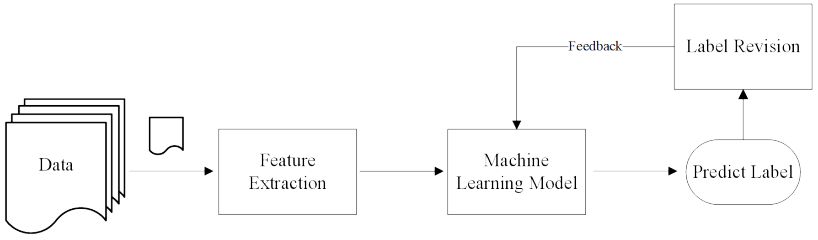
Multi-Layer Perceptron (MLP) utilises ANN, DL and supervised learning techniques (56). It consists of at least three layers of nodes: an input layer, one or more hidden layers and an output layer.

2.5 Online Machine Learning

Unlike traditional ML, *online ML* focuses on developing techniques for incrementally updating models as data arrives as shown in in Figure 2.4b (58; 65). This method addresses the limitations of batch learning by allowing models to be updated immediately and efficiently with new data without waiting for sufficient samples to be trained and retraining the model from scratch every time (65; 58). Therefore, the primary advantage of Online ML is handling vast amounts of data efficiently to achieve *scalability*. Its adaptability allows real-time learning and prediction. Furthermore, it reduces the computational cost by using only relevant, recent data for training, avoiding the need to store large datasets. It addresses the problem of concept drift, where the distribution of the data changes over time and deviates from the one learned by the Online ML model (65). Indeed, these algorithms typically employ adaptive mechanisms to adjust to changes in the data distribution over time, making them suitable for dynamic environments such as IoT networks.



(A) Batch Machine Learning



(B) Online Machine Learning

FIGURE 2.4: Online Machine Learning Versus Batch Machine Learning.

Online ensemble learning (OEL). It is an extension of the online learning framework that combines multiple learning algorithms to improve predictive performance (66). For instance, the Online Random Forests (ORF) algorithm adapts the traditional RF method to an online setting by incrementally building and adjusting DTs as new data flows in. This approach not only maintains the robustness of standard ensemble methods but also adds the flexibility needed for dynamic data environments (66).

2.5.1 Metrics

Evaluating the ML model is an essential part of any solution. This section will cover wildly used types of evaluation metrics for classifying IoT devices.

Confusion matrix. It describes the complete performance of the model (56). It is the basis for the other types of metrics (56). For constructing the matrix, a set of predictions are required to be compared to the actual ones (see Figure 2.5). Each row represents actual values (i.e. class), while each column represents predicted values. There are four main terms that should be considered to understand the matrix. (67):

- *True Positive (TP)*: Correctly classify an instance as denoted, by means predicted YES and the actual output is also YES.
- *True Negative (TN)*: Correctly classify an instance as denoted, by means predicted NO and the actual output is NO.
- *False Positive (FP)*: Incorrectly classify an instance as denoted, by means predicted YES and the actual output is NO.
- *False Negative (FN)*: Incorrectly classify an instance as denoted, by means predicted NO and the actual output is YES.
  Consequently, a perfect classifier would have only TPs and TNs (56).

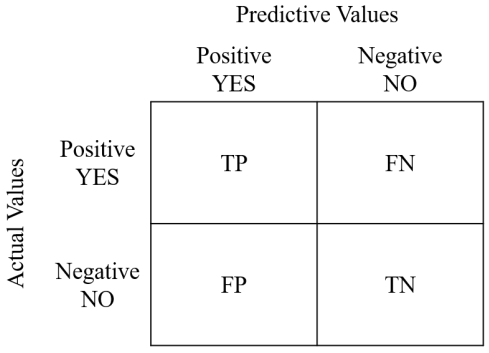


FIGURE 2.5: Confusion Matrix.

Classification accuracy. It is the most straightforward metric, that is the average of correct predictions (68).

$$Accuracy = \frac{TP + TN}{all\ predictions} \tag{2.1}$$

Precision. It is the accuracy of the positive predictions (56). It shows the model's ability to identify positive instances truly.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall. Also called True Positive Rate (TPR) or sensitivity, this is the proportion of the truly positive instances with respect to all positive instances (56).

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

False Positive Rate. It indicates the proportion of negative instances that are incorrectly classified as positive.

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{2.4}$$

For multi-classification problems, the $Average\_FPR$ can be calculated by commuting the FPR_i for each class C_i independently, where $i < N$ and N is the total number of classes.

$$Average\_FPR = \frac{1}{N} \sum_{i=0}^{N} FPR_i \tag{2.5}$$

F1-Score. It is a combination of Precision and Recall. It asses the performance of each class x by taking into account the impact of FP_x and FN_x (56).

$$F1_x = 2 \times \frac{Precision_x \times Recall_x}{Precision_x + Recall_x} \tag{2.6}$$

2.6 Standard Score

The standard score (known as z-score), is a statistical measure that quantifies the number of standard deviations a data point is from the mean of the dataset (69). It is a technique used to understand how far a specific observation x is from the typical values observed in the data X. The z-score of x is defined by the following:

$$z\text{–}score = \frac{x - \mu}{\sigma} \tag{2.7}$$

where x is the value of the element, μ is the mean of X, and σ is the standard deviation of X, which measures the average distance of each data point in X from the mean.

Fundamentally, a z-score measures the number of standard deviations a data point is from the mean of the dataset. A z-score close to 0.0 implies that the data point is near the average, typically reflecting a standard or expected value as shown in Figure 2.6. In contrast, z-scores that are significantly positive or negative indicate values that are well above or below the mean, respectively. A positive z-score indicates an

above-average data point, while a negative z-score points to a below-average data point. This metric is invaluable for assessing the relative position of data points within a distribution, identifying outliers, and standardising scores across different scales for comparative analysis.

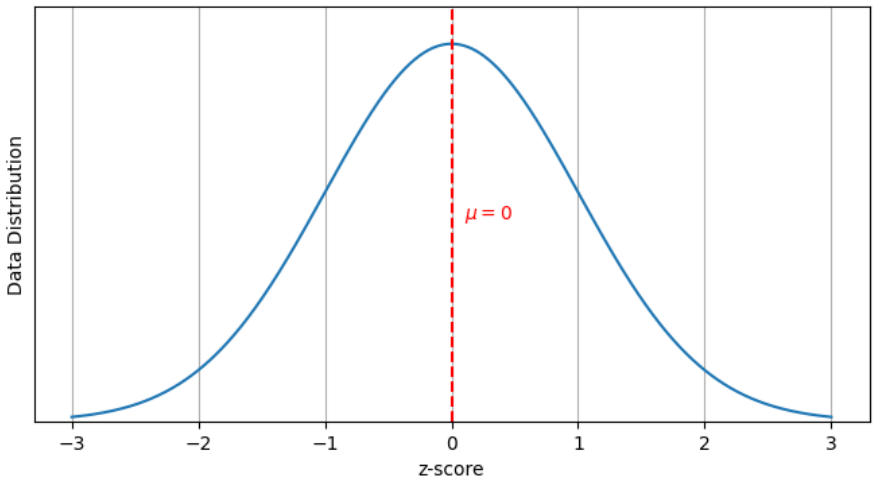


FIGURE 2.6: The z-scores and the standard normal distribution.

2.7 Summary

The Chapter delves into essential concepts in modern technological landscapes, such as the IoT, IoT fingerprinting, ML, online learning, and z-score. IoT has enabled a network of interconnected devices that communicate seamlessly, generating vast data streams. IoT fingerprinting enhances the IoT systems' security by identifying unique device behaviours. ML techniques thrive on this data to predict and automate decisions, but with the dynamic nature of IoT data, traditional ML methods falter, paving the way for online learning to adapt in real-time to new data. Moreover, the z-score is discussed as a critical statistical measure to manage data variability.

Chapter 3

Survey on Machine Learning Techniques for Passive IoT Device Fingerprinting

3.1 Introduction

There are various surveys in the literature devoted to IoT security and privacy issues; however, limited works discuss passive IoT device fingerprinting (70; 4). To our knowledge, there is no direct review study that examines existing efforts to assess the approaches, and challenges or even prevent duplicating efforts. Therefore, the need for a comprehensive survey on this matter is important more than ever. Therefore, the research questions we explore in this work is:

RQ1 What approaches have been proposed in the literature for passive IoT device fingerprinting?

RQ2 Which specific limitations and challenges exist in the literature regarding IoT fingerprinting techniques, and how do these affect the accuracy and scalability of IoT device identification?

This Chapter will expound on the major existing solutions for IoT device fingerprinting, specifically, the passive fingerprinting method. This is because it utilises the network traffic and provides valuable information that enables network administrators to identify any IoT device by its network characteristics. Generated fingerprints can be distinctive to identify the device's identification domain (e.g., vendor, category, and model). This chapter also focuses on studies that applied ML for IoT device identification due to its effectiveness in addressing security challenges in the IoT environment. Therefore, the main contributions of this work are as follows:

- Provides a comprehensive study in the area of passive IoT device fingerprinting based on ML and network characteristics over the period 2017-2023, which, to the best of our knowledge, is the first academic review that solely focused on this aspect
- Introduces a taxonomy of ML-based approaches for passive IoT device fingerprinting
- Provides a detailed study of network traffic features that can be used for generating a device fingerprint
- Reports the IoT traffic features utilised by the literature. This is a novel contribution that can be considered a baseline reference
- Identifies possible key open research problems, challenges, and further directions in the area of IoT device fingerprinting

The rest of the paper is organised as follows: Section 3.2 presents related IoT security surveys; Section 3.3 presents the survey methodology. Section 3.4 introduces the proposed taxonomy of the reviewed passive IoT fingerprinting approaches based on ML. Section 3.5 presents the IoT datasets. Section 3.6 highlights the challenges and further directions. Finally, a summary of this chapter is in Section 3.7.

3.2 Related Surveys

Several surveys in the literature analyse IoT security and privacy concerns. The majority focus on IoT security challenges, including potential issues and solutions for securing IoT (71), IoT attacks and corresponding countermeasures (72), as well as technologies and applications aimed at fostering trust in IoT systems (73). The survey by Yang *et al.* (74) explores IoT device limitations, IoT attack classification approaches, and authentication schemes. Additional surveys delve into IoT trust management techniques (75; 76), authentication protocols (77; 78), and the role of fog/edge computing (79; 80).

Recent studies have demonstrated the effectiveness of machine learning (ML) in enhancing IoT security. Cui *et al.* (70) review how ML applications can be used to develop intelligent IoT systems, including traffic profiling, security measures, and IoT device identification. Similarly, Hussain *et al.* (81) explore the role of ML in providing security and privacy solutions for IoT networks. While Sagu and Gill (82) focus on reviewing ML techniques that secure IoT devices. Tahaei *et al.* (83) discuss IoT traffic classification and its practical applications. It differentiates between regular network traffic and IoT traffic, noting that early classification methods, which relied on port monitoring, could not detect fake ports. Conversely, ML approaches leverage traffic

features like payload, statistics, and behaviour. The study offers a taxonomy of IoT traffic applications, such as device identification, anomaly detection, and user authentication. For device identification, the study summarises various research objectives, ML techniques, traffic sources, and device types. The key challenges highlighted in this survey include (i) quality of Service (QoS), where traffic classification is an essential factor for management to meet QoS, (ii) scalability due to big data from numerous devices, (iii) standardisation issues, and (iv) the need to update traffic patterns for anomaly detection. Tahaei et al. (83) present a framework for analysing mechanisms of fingerprinting IoT devices. They review 31 studies published between March and September 2020, focusing on ML-based approaches and notable Rule-based techniques (such as if-then-else rules). The framework categories the studies into seven categories: fingerprinting methods (passive/active and static/dynamic), fingerprinting input features (MAC, Network, and Application Layers; packet/dataframe/flow; header/payload; and Rule-based or ML-based), and fingerprinting outputs (Class/Type/Unique). The authors conclude that most studies rely on passive network scanning, and both static (e.g., MAC address) and dynamic (e.g., IAT) features were used in many studies to enhance robustness against attacks. Meanwhile, studies employing active and static fingerprinting were less common due to their susceptibility to vulnerabilities. Sanchez *et al.* (84) a comprehensive review studies two main scenarios, device identification and device misbehaviour. It discusses the various identification domains of device types (e.g., IoT devices and general computers), behavioural sources (e.g., network, clock skew, and system calls), and behaviour processing and evaluation methods (e.g., rule-based, ML/DL-based, and statistical-base). The authors emphasise the importance of ML and DL approaches through their remarkable performances using network communication. They draw attention to the need for IoT datasets and the exploration of DL approaches, such as recurrent neural network (RNN), convolutional neuron network (CNN) combination, and Reinforcement learning (RL).

Our scope differs from the reviewed surveys, as we target the objective of IoT device identification using passive device fingerprinting and ML. Also, our novel contribution is to provide a comprehensive review, which is not addressed before. Shridhar and V (35) state that one of the challenges in securing the network is using device fingerprinting. However, Cui et al. (70) review a few studies, conducted between 2013 and 2017, related to IoT device identification using ML. The review highlights various challenges and issues, mainly defending the approach to protect the process of device identification, understanding the influence of different ML techniques and evaluating user privacy. Meanwhile, Yadav *et al.* (85) consider many studies in IoT device fingerprinting, including ML-based, Rule-based, passive, and active approaches, yet, it provides a general categorisation, not a deep inspection of the approaches and the challenges encountered. Yue *et al.* (86) discuss IoT solutions based on DL to address securing and privacy issues from two main perspectives,

system architecture and security. From the security perspective, the survey discusses profiling and fingerprinting, and the ability of DL to build a reliable secure IoT system. In our survey, we review highly ranked studies (Section 3.3), propose a taxonomy of passive IoT deceive fingerprinting based on ML to analyse the studies (Section 3.4), and define the main challenges (Section 3.6).

Sanchez *et al.* (84) addresses device behaviour fingerprinting, including IoT devices. However, they do not discuss network monitoring types, such as active and passive, nor their implications for device fingerprinting. They identify three device identification domains: types, models, and individual devices. We believe that more domains for identification should be explored (discussed in Section 3.4.4). In ML/DL-based techniques, the review is limited to supervised and unsupervised learning approaches, while our review covers a broader range, including semi-supervised and meta-learning (discussed in Section 3.4.3). They mention the types of features for fingerprinting (e.g., flow statistics and header statistics), but in our survey, we inspect the proposed network features and classify them based on their origin (discussed in Section 3.4.2).

Safi *et al.* (87) present an investigation into IoT devices profiling (fingerprinting) in common networks (e.g., smart home). It also discusses the identification techniques, including IoT device type identification, individual device identification, unseen IoT device identification, and anomaly detection. It classifies the IoT traffic features into features related to size, service, time, statistical, network communication protocol long range, and network communication protocol short range, where the practicality and the efficiency of the common features are stated. Our survey generalises the feature categories to encompass all IoT device traffic features (discussed in Section 3.4.2). Moreover, we are the first survey that presents a baseline feature reference for interested researchers to examine the proposed features. The review briefly addresses ML approaches and highlights several challenges in IoT profiling. The effectiveness of existing methods is uncertain due to the rapid expansion of IoT environments. Challenges include the need for less costly yet accurate feature extraction and preprocessing for diverse IoT device profiling. A significant reliance on training data often results in models failing to recognise updated and new devices. The lack of public IoT datasets hinders effective method comparison. Additionally, profiling involves sensitive information, potentially predicting device activities, underscoring the urgent need for techniques that ensure security and protect user privacy. These identified challenges remain unresolved, and we discuss additional challenges in Section 3.6.

Table 3.1 summarises the main contributions of exciting surveys that addressed IoT security and privacy issues.

TABLE 3.1: Related IoT Security and Privacy Surveys Contribution

Year	Authors	Contribution
2013	Zhao and Ge (71)	Discusses security issues and solutions that exist in each layer of IoT structure.
2014	Yan *et al.* (75)	Overview of trustworthy IoT by discussing trust properties and objectives of IoT trust management.
2017	Mosenia and Jha (72)	Summarises IoT security attacks on the edge-side layer of IoT and countermeasures against them.
2017	Ferrag *et al.* (77)	Comprehensive review for IoT authentication protocols, threats, countermeasures and applied verification techniques.
2017	Lin *et al.* (79)	Discusses the integration of fog computing and IoT and related security and privacy issues.
2017	Yang *et al.* (74)	Discussion security and privacy issues in IoT applications, IoT limitations, classification approaches for IoT attacks, IoT authentication schemes and architectures, and IoT solutions.
2017	Yu *et al.* (80)	Survey on using edge computing to improve the IoT network security.
2018	Cui *et al.* (70)	Overview of the major application of ML for IoT techniques.
2018	Seliem *et al.* (88)	Discussion on threats identification and mitigation in IoT environment.
2018	Shridhar (35)	Brief review on lightweight encryption technique and device fingerprinting approaches to secure data transmission and devices.
2019	Ud Din *et al.* (76)	Comprehensive analysis of IoT trust management techniques and limitations.
2019	El-hajj *et al.* (78)	Survey of the IoT authentication schemes and presets a taxonomy of discussed schemes.
2019	Hassija *et al.* (73)	Survey of the major solutions for IoT security, the security-related challenges and sources of threat at different layers of an IoT application.
2019	Hussain *et al.* (81)	Systematic review of ML applications in providing security and privacy services to the IoT networks.
2020	Sagu and Gill (82)	Review of ML approaches to secure IoT devices.
2020	Tahaei *et al.* (83)	A detailed review of the IoT network traffic classification methods.
2020	Yadav *et al.* (85)	A systematic framework for reviewing the literature related to IoT device fingerprinting.
2021	Yue *et al.* (86)	A survey of IoT-based DL solutions in IoT environments for security and privacy concerns.
2021	Sánchez *et al.* (84)	A comprehensive review of device behaviour fingerprinting focusing on device identification and device misbehaviour and discussing several aspects regarding device types, behavioural sources, techniques and datasets.
2022	Safi *et al.* (87)	A comprehensive review of various IoT device profiling techniques in different domains (e.g., Healthcare and Smart Home).
2023	Our contribution	Comprehensive survey of passive IoT device fingerprinting based on ML.

3.3 Methodology

The survey applies systematic literature of relevant papers published on various venues, including *ACM*, *IEEE*, *Science Direct*, and *Elsevier*. The review includes highly ranked journal articles and conference papers published between 2017 and 2023 using the keywords: ”passive IoT device fingerprinting”, ”IoT device fingerprinting”, ”IoT Device type identification”, ”IoT device classification”, ”classifying IoT devices”, ”behavioural IoT fingerprinting”, ”IoT Fingerprinting Technique”, and ”IoT Fingerprinting”.

3.4 ML Based Taxonomy for Passive IoT Device Fingerprinting

This section presents the taxonomy of the reviewed approaches for IoT device fingerprinting based on passive network monitoring and ML. The taxonomy

showcases five major dimensions, as illustrated in Figure3.1. These dimensions capture the main distinguishing aspects of the research works carried out in the field. The passive network monitoring element is characterised by the particular *network communication protocols* observed (Section 3.4.1) and the *network traffic features* extracted (Section 3.4.2). The *ML approaches* element is captured by a dedicated dimension (Section 3.4.3) representing the class of ML algorithms employed to identify which device, or class of devices, corresponds to the computed fingerprint. Indeed, some of the reviewed works aim at identifying one or more properties of a device (e.g., the model) rather than the device itself (i.e., the device instance), which determines the *identification domain* of the fingerprinting approach (Section 3.4.4). Finally, whether the fingerprinting approach supports extending the set of recognisable devices, is captured by the *scalability dimension* (Section 3.4.5).

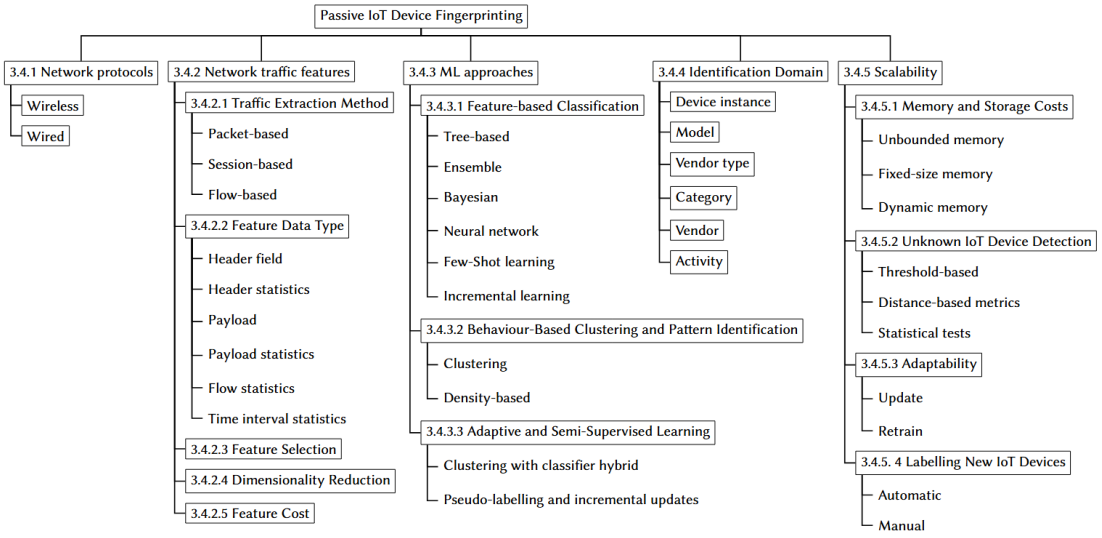


FIGURE 3.1: Taxonomy of machine learning techniques for passive IoT device fingerprinting.

3.4.1 Network Protocols

The reviewed studies employ various communication protocols within IoT systems. Devices utilising wireless features, such as those based on WiFi, ZigBee, and Z-Wave, are prevalent due to their simplicity in setup and flexibility. WiFi-based IoT devices emerged as the most
popular (23; 2; 1; 5; 12; 89; 90; 91; 92; 93; 94; 13; 95; 96; 97; 98; 99; 100; 101; 102). Some studies considered devices based on
ZigBee (2; 12; 21; 95; 91; 97; 93; 13; 23; 90; 103; 89; 98), Z-Wave
(2; 95; 97; 13; 23; 90; 103), Bluetooth (23), Insteon (12; 95; 13), ClearConnect (Lutron's proprietary protocol) (95; 13), and Thread (Nest's proprietary protocol) (95; 13). Other studies analysed Ethernet communication (23; 2; 1; 90; 96; 104; 99).

3.4.2 Network Traffic Features

Network traffic can be scanned continuously and analysed to acquire the device's features (characteristics). The reviewed studies proposed sets of features ranging from tens to hundreds of features representing device fingerprints. These fingerprints form an individual device profile, where the device profile is the input to the ML model for classification and identification. The challenge of this dimension is to find the distinguishing features for the device identification domain, not to mention the costs of feature extraction.

This section details the traffic instance extraction (3.4.2.1), traffic feature types (3.4.2.2), feature engineering and selection (3.4.2.3), dimensionality reduction (3.4.2.4), and related cost (3.4.2.5).

3.4.2.1 Feature Extraction Method

Features are usually extracted from individual packets, flow, and/or sessions to form vector features (105; 34).

Packet-based. It involves extracting and examining the header fields and payload of packets as they are transmitted across a network. This method is widely adopted in early studies
(18; 2; 11; 10; 90; 106; 103; 101; 102; 107; 108; 109; 110; 20; 111; 112; 100; 113; 114).

Session-based. The packets in a sequence within a session have five field values in common, known as a 5-tuple. These include the source IP, destination IP, source port, destination port, and transport-layer protocol, in either
direction. (115; 92; 116; 117; 113; 97; 118). A session could focus on a TCP connection, known as a 4-tuple session, from the synchronisation message (SYN packet) to the finish message (FIN packet) (5; 119; 99).

Flow-based. A window time is predefined to split the traffic flow of each IoT device into a sequence of packets considering that time window (e.g., 5
minutes) (1; 120; 23; 14; 12; 21; 9; 98; 121; 22; 122; 19; 123; 124; 104; 125; 126). There are some studies define a flow as a predefined fixed number of packets without considering time (127; 10; 26).

It is worth noting that it could be used more than one traffic instance type in fingerprinting as in (3). The study used packet-based and single/multiple flow instances.

3.4.2.2 Feature Data Type

Packets consist of a header and a payload. The payload is usually encrypted, which hinders inspection of its content for fingerprinting purposes (9). Consequently, the majority of studies have shifted their focus to analysing traffic features from the header fields, header statistics, payload statistics, flow and time interval statistics.

Another novel contribution of this research is the reporting of the traffic features used by the reviewed papers. This effort could serve as a baseline reference for the traffic features proposed for fingerprinting (see Appendix A). It is worth noting that the features reviewed are only partly reported due to insufficient information in this aspect (11; 20; 114; 119; 26).

Header field. This category includes features extracted from the packet header, such as MAC addresses, ports, IP addresses, protocols, packet size, and flags (see Table A.1) (18; 2; 1; 11; 10; 12; 8; 9; 90; 101; 102; 98; 107; 108; 109; 110; 117; 97; 118; 20; 111; 127; 114). These features can be acquired even from encrypted traffic. Although some header field features, like MAC addresses and IP addresses, are specific to the IoT device identification domain (discussed in Section3.4.4), they do not necessarily accurately fingerprint other domains, such as device model and category. Moreover, these features are vulnerable to spoofing.

Many studies employ the port interval modelling method to categorise source and destination ports into port class numbers. For example, Miettinen *et al.* (2) and Zhang *et al.* (108) use a simplified model where port numbers are classified into four categories: class 0 for no ports, class 1 for well-known ports ranging from 0 to 1,023, class 2 for registered ports from 1,024 to 49,151, and class 3 for dynamic ports from 49,152 to 65,535. In contrast, Kostas *et al.* (109) adopt a more granular approach by categorising ports into 14 distinct classes, including specific services like 53 (DNS) and 80 (HTTP) as individual classes, thereby differentiating them from the broader category of well-known ports. This refined categorisation by Kostas *et al.* acknowledges the diverse and specific functionalities of IoT devices, whose unique network behaviours and protocol uses require more precise identification and analysis of port usage. In fact, destination ports can serve as distinct identifiers for the services they provide, as noted by Zhang *et al.* (108). In their analysis of 24 hours of IoT traffic, they observed that a Smart Things device sends an NTP request every 600 seconds, while an Amazon Echo device sends one every 50 seconds. Additionally, devices from the same vendor, like Belkin Wemo Switch and Motion Sensor, often use the same non-standard/registered ports.

Even though the IP protocol is generally stable, it may not be sufficient for identifying specific applications or services. In contrast, port numbers show more stability, but can vary with dynamic port assignments.Therefore, Pashamokhtari *et al.* (118) combined

the pair of IP protocol and port number as a main feature to improve accuracy and resilience against firmware upgrades and user activity, significantly reducing FP.

While the IP protocol and port numbers provide a stable basis for identifying specific applications or services, the use of IP addresses themselves plays a crucial role in IoT device fingerprinting. For instance, if IoT devices consistently send packets to specific cloud servers owned by different vendors, the destination IP addresses can directly identify the device's vendor without requiring a trained identification model, but rather relying on a database of known server IPs (90). Furthermore, Zhang *et al.* (108) suggest utilising IP options rather than just the IP addresses. They develop a unique fingerprinting method by analysing the frequency of IP address changes, which could indicate various network conditions like failures, changes, roaming, or periodic reassignments. This approach provides additional insights into the dynamic behaviour of devices within the network, enhancing identification strategies.

Additionally, identifying IoT devices in environments using Network Address Translation (NAT) presents another layer of complexity (92; 121; 122; 118). Mainuddin *et al.* (92) demonstrate that remote IP addresses and ports can effectively serve as features for identifying devices behind NAT. They highlight the utility of decomposing the IP address into network prefix and host components, showing how the network prefix, combined with port information, can classify different types of IoT devices effectively, even in these more complex settings.

Packet length and direction vary depending on the implementation of IoT devices (110). While Kuzniar *et al.* (106) focus on these two simple features for fingerprinting, Duan *et al.* (110) argue that this approach could be significantly affected by updates to configurations, software, and firmware.

The TCP window size feature reflects the amount of data a sender can transmit before receiving an acknowledgement (10). This feature can provide insights into the memory or buffer space available on the IoT device; for instance, a larger window size suggests that the device has more buffer space available to manage incoming data (90). For example, light bulbs typically have smaller window sizes compared to smart cameras, as observed by Bezawada *et al.* (7).

Header statistics. It consists of features statistically derived from the packet header based on a flow/session to analyse its characteristics, such as sum, min, and max (14; 10; 1; 8; 21; 120; 103; 128; 98; 22; 122; 129; 99; 125; 127; 123) (See Table A.2). Time-To-Live (TTL) is a feature that determines the lifespan of a packet in a network before it is discarded (10). According to Hamad *et al.* (10), the average TTL value is the most significant feature for fingerprinting $\langle$*Device Instance*$\rangle$ and $\langle$*Model*$\rangle$.

The packet size varies between devices and can change dramatically (130). Consequently, Mainuddin *et al.* (92) derive statistical features from TCP flow packet

size. Similar statistical features were also extracted from Ethernet packet size (10), IP packet size (10; 128; 19), and IP header size (10). Furthermore, Duan *et al.* (110) analyse the frequency distributions of packet lengths and directions, which shows great discrimination. However, they inspect the phenomenon of concept drift, where substantial evolution in traffic patterns over time may affect these analyses.

Packet protocols can be categorised into control packets and user packets, as noted by Bai *et al.* (14). Control packets support protocols such as DNS and ARP, whereas user packets facilitate user data and server communication through protocols like TCP/UDP and HTTP. Bai *et al.* highlight six statistical features from user and control packets as distinguishing features for fingerprinting specific device $\langle Category \rangle$. These features derive from variations in packet size and interval time. Du and Li (22) use features derived from seven protocols to fingerprint IoT devices, including the average packet size and the average rate of incoming and outgoing flows within a one-minute period. They divide the feature vector for each device into several sub-vectors according to different protocols, and these sub-vectors are trained independently to reduce the overlap between devices. Additionally, Sun et al. (19) extract the number of protocol packets sent in less than one minute for their analysis.

Ports are usually distinctive among different devices; thus, the occurrence of each source and destination port is recorded (10). Sivanathan *et al.* (12) observe the uniqueness of this feature, where they specifically determine the number of unique ports for each IoT device. Some studies have established a feature based on the frequency of ports within specified bins (25; 122). Meanwhile, others consider the number of IP destinations as a feature (10; 2).

Packet payload. Even though the encrypted payload is not widely utilised, the plain text payload, such as handshake packets and DNS query, is used (12; 121; 111; 112) (See Table A.3).

Analysing unencrypted payloads is fundamental in IoT device fingerprinting, as it provides critical data that helps distinguish between devices based on their specific network activities and protocol usage. Sivanathan *et al.* (12) collect a bag of unencrypted DNS names because IoT devices from the same vendor tend to query the same domain names. In contrast, Kumar *et al.* (121) extracted unique DNS queries used by IoT devices. However, DNS names are not reliable for fingerprinting behind NAT, as IoT devices often use the same cloud services, according to Ma *et al.* (102). Ammar *et al.* (99) exploit unencrypted payloads and utilised a binary Bag of Words (BoW) model, demonstrating significant performance improvements despite potential vulnerabilities. Jia *et al.* (112) and Zhao and Ge (113) use TCP/UDP payload bytes, focusing only on plaintext payloads to capture keywords of application layer protocols through convolution operations. Jia *et al.* (112) also emphasise the importance of combining payload with flow statistic features for a robust fingerprint.

Sun *et al.* (3) extract 139 features from the unencrypted TLS handshake, including cipher suites (CS), TLS extensions, and public key sizes, noting their significant discriminatory potential but also their vulnerability to attacks. Sivanathan *et al.* (12) and Kumar *et al.* (121) explore the implications of CS for the uniqueness of IoT fingerprinting.

Encrypted payloads have gained the attention of recent scholars for fingerprinting IoT devices. Kotak and Elovici (115) proposed using a session of encrypted packet payloads by transforming them into grayscale images. This approach can be applied for fingerprinting behind NAT as it only requires utilising the initial bytes of the payload without any modification (i.e., not affected by NAT). The approach is tested on TCP session payloads. Similarly, Liu *et al.* (100) also converted payloads into grayscale images, but they employed a packet-based approach. Zhao and Ge (113) utilised the first B bytes of the first K packets as these contain necessary protocol keywords to establish or negotiate the settings for the session, such as the server name and CS.

Payload statistics. Deriving statistics from the payload is also considered such as calculating payload size, min and max (1; 10; 12; 8; 9; 130; 90; 120; 109; 97; 118; 104) (See Table A.4). They depend on the size and the nature of the payload data, but not the data itself. The reviewed studies compute various statistical features gained from protocols flow payloads including IP (34), TCP (7; 10; 130), and DNS (12; 13). In general, the proposed payload-based features can operate on encrypted traffic.

The frequency of payload bytes analysis can gain insights into the nature of traffic. Therefore, Sun *et al.* (3) examine how byte values distribute within payload data over a fixed time window to create a frequency histogram.

The packet payload length, extracted from the first bytes of a TCP/UDP flow by Jia *et al.* (112), reflects the size of the protocol keywords within that flow. While Meng and Li (104) convert the payload lengths into a time series list using a wavelet transform technique, which archives high accuracy for identifying IoT device $\langle Category \rangle$.

TCP payload data offset indicates the start of the data at the top of the network layer. Hamad *et al.* (10) show the significant of this feature in fingerprinting $\langle Model \rangle$ and $\langle Device\ Instance \rangle$.

For DNS flow, Perdisci *et al.* (13) computed the probability of queried domain names in a given time window as well as the inverse document frequency. There are other statistical features derived from DNS flows used for fingerprinting, such as counting the top 10 DNS names, and DNS response size.

The entropy of the payload is correlated with the type and the size of transmitted data (7). Few studies (109; 90) include this feature even though it gives an insight into

the characteristics of the payload by distinguishing between different types of data and identifying encrypted content.

Flow statistics. IoT devices generate different amounts of traffic, where features could be extracted from groups of packets based on flows or sessions (e.g., number of packets per flow)
(10; 1; 12; 9; 21; 124; 103; 128; 129; 120; 25; 92; 121; 107; 108; 122; 131; 97; 99; 127; 3) (See Table A.5). The features are independent of packet headers and payloads, making them the least privacy-intrusive (1). Additionally, they apply to encrypted traffic. Generally, the traffic generated by IoT devices can be originate from either idle or active traffic. The idle traffic, also known as background traffic,consists of automated processes that maintain the device's functionality and ensure it remains updated without direct user interaction. In contrast, active traffic is directly produced by user actions.

IoT devices generate varying levels of traffic along with different durations (92). These variations in transmitted traffic illustrate the differences in the physical functions of IoT devices (92; 120). Mainuddin *et al.* (92) analyse TCP flow-level features in both directions and extract flow size, flow rate, and ratio. In contrast, Zhang *et al.* (120) and Sivanathan *et al.* (12) adopt a different approach by considering a time window, where the latter highlights the significance of flow duration and volume in fingerprinting specific $\langle$*Device Instance*$\rangle$. Further, in subsequent works, Sivanathan *et al.* (124; 21) define features based on multiple time scales by computing the count of packets and bytes across 8 flows. Zhang *et al.* (120) identify burstiness, caused by active traffic, as a distinctive feature and use the z-score to quantify this burstiness.

Periodic traffic is transmitted to maintain the connection between IoT devices and the server without triggering an event (126). To ensure connectivity, IoT devices periodically send packets that either report their status, such as through a KeepAlive mechanism, or send notification messages. Features derived from this type of idle traffic can exhibit a high level of fingerprint discrimination among IoT devices. Zhang *et al.* (120) employ autocorrelation to characterise the periodicity of the traffic, selecting the maximum periodicity value as a distinctive feature. Marchal *et al.* (23) use a 30-minute flow (i.e., packets sent by a given MAC address using a specific protocol) to extract 33 periodic features. They acquire the features during periodic flow inference and divided them into four categories; (i) periodic flow features that related to the quantity and the quality of the flow; (ii) period accuracy features that related to the measure of the accuracy and the noise of the flow; (iii) period duration features where the flow was split manually into four duration ranges; and (iv) period stability features. They evaluate the significance of the features where period duration features are of the highest importance. The count of flows with a static source port also showed its relevance. By means, IoT devices differ in managing periodic communications.

Wu *et al.* (126) calculate cycle duration for each device (i.e., to obtain traffic transmission time and idle time), and then select a time scale that represents the minimum required time to extract traffic rate. The time scale selection is different among IoT devices which reflects their traffic and reduces overhead.

Time interval statistics. These are features related to time interval, including IAT statistics and duration (10; 1; 12; 9; 21; 124; 103; 128; 129; 120; 25; 92; 121; 107; 108; 122; 131; 97; 99; 127; 3) (See Table A.6).

Packet inter-arrival time measures the delay between successive packets (10; 107). It is one of the significant features that is highly affected by hardware configuration and Internet quality (25). For instance, poor Internet connection increases the latency and the retransmission of packets. Zhang *et al.* (108) observe variations in IAT values among devices of the same type and define class numbers mapped to different IAT lengths. Several studies derived statistics from IAT (25; 10; 92). In fact, Babun *et al.* (103) use only IAT to fingerprint Z-based IoT device categories. They derive a density distribution from the IAT values within a specific time window of idle traffic, using this distribution to construct a probability signature.

Packet jitter is a measurement of the variability in the delay time between the arrival of packets at a destination. Luo *et al.* (128; 129) consider five-time windows for generating packet jitter statistics to analyse the stability and consistency in packet timing.

The time span of flows is a widely known feature (120; 12; 121; 25). The activity time feature is proposed by Thangavelu *et al.* (1), which is related to the amount of time it takes for a device to send packets in a specific time window. For instance, if a device sends packets that span 10 minutes within a 15-minute session, then the activity period would be 10 minutes. On the other hand, the sleep time during non-active flows is also considered a feature (120; 12; 121).

3.4.2.3 Feature Selection

Feature vector dimension can significantly impact the performance and efficiency of ML models. Feature vectors having high dimensions increase computational complexity and memory requirements, while noises can lead to overfitting. On the other hand, low-dimension feature vectors might not capture the underlying patterns and relationships in the data, leading to underfitting and can result in poor predictive performance, due to insufficient data. Therefore, the reviewed studies employed feature selection techniques to extract relevant features to improve model performance.

ANOVA (Analysis of Variance) score is a statistical feature selection in (107). It is primarily used to identify the differences between different classes by comparing their means and analysing the variances within and between these classes. In the context of feature selection for machine learning, ANOVA can help in determining which features are significant for predicting the target variable, especially when dealing with numerical features and categorical targets (107).

ReliefF feature selection algorithm assesses feature significance based on the differences in feature values among nearest neighbour pairs. Marchal *et al.* (23) employ ReliefF due to its conceptual similarity to kNN (k-nearest neighbours). It identifies the significance of the count of flows with a static port feature. The counting period flows and period duration features are also important. However, the least relevant features are the mean and the SD of both period per-flow and period inference success; but the authors decided to keep them because they improved the identification accuracy.

Random Forest (RF) feature importance is a valuable technique in feature selection, as it not only provides strong model performance but also ranks features based on their contribution to the classification outcome. By analysing these importance scores, one can select the most relevant features, thereby enhancing model performance. This approach has been effectively utilised in various studies, including the work by Palmese *et al.*(122), where RF feature importance was employed to identify and prioritise critical features for improved classification performance.

Genetic Algorithm (GA) was introduced by Aksoy and Guns (11) to reduce the number of selected features to enhance the IoT fingerprinting accuracy. Chen *et al.* (131) addressed the concept drift that affects the stability of identification accuracy due to the changes in the environment. They proposed a feature selection method based on the degree of drift and GA to ensure high performance and stability. The study demonstrates an increase in accuracy by 8.1%.

xverse (X uniVerse) is a Python package for machine learning for feature engineering, feature transformation, and feature selection (132). It is a feature-importance-based voting method that identifies the most informative features. Kostas *et al.* (109) adopt this method and remove the features that failed to receive votes along with randomly assigned features (e.g., TCP sequence). The remaining features are further passed to GA to find the most relevant features. This method eliminates about 73% of the original feature set.

Several studies propose various techniques for feature selection. Duan *et al.* (110) utilise *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) clustering to identify representative instances in training data. Du and Li (22) focus on evaluating the importance of seven protocols by modelling them for different devices, finding that protocol importance varies based on device functionality. This approach allows to eliminate of unnecessary models, thereby maintaining accuracy while

reducing storage costs and training time. Ma *et al.*(102; 101) propose forming feature vectors from the header field through packet-based extraction, grouping packets into bursts within a one-second window, and selecting sequence profiles to minimise computational overhead. Additionally, Kotak and Liu (115; 133) introduced an approach that avoids complex feature engineering, focusing instead on extracting features from encrypted payloads for fingerprinting.

3.4.2.4 Dimensionality Reduction

Dimensionality reduction is a preprocessing step aimed at eliminating redundant, noisy, and irrelevant features, thereby enhancing feature accuracy and reducing training time in machine learning tasks (134). Traditional methods such as *Principal Component Analysis* (PCA) and *t-distributed Stochastic Neighbour Embedding* (t-SNE) are widely used in the literature for this purpose (108; 22; 21; 25).

While DL methods like *Convolutional Neural Networks* (CNNs) and *Long Short-Term Memory* (LSTM) networks are primarily designed for classification and prediction tasks, they can also be effectively used for dimensionality reduction. Several studies have leveraged CNN models by extracting the activations of specific layers to use as input for classifiers (113; 128). This technique, known as transfer learning, utilises a pre-trained CNN as a feature extractor. For instance, Zhao *et al.* (113) extract protocol keywords from plain text payloads using three convolutional blocks.

In addition, LSTM networks have been employed for dimensionality reduction by capturing temporal relations in sequential data as proposed by Bai *et al.* (14). Jia *et al.*(112) combine a Bi-LSTM with a one-hot layer to construct high-dimensional representations from sequence data, such as TCP/UDP payload length and payload bytes.

The *Autoencoder* (AE) gained the attention of late researchers (as in (20)) which not only reduces the input dimension but also provides a more powerful generalisation than other techniques (135). Wang *et al.* (26) introduce a preprocessing pipeline that first applies PCA, followed by an encoder to reconstruct the features into a lower-dimensional space. Then, clustering is used to determine the centroid of each cluster for each IoT device to represent a feature.

3.4.2.5 Feature Cost

Generally, traffic features are correlated with a variety of costs, such as obtaining and storing them. Chakraborty *et al.* (93) defined three types of costs: (i) the computational costs related to the process of acquiring and computing, (ii) the memory costs involving the measure of memory usage for storing features during computation, and

(iii) the privacy cost concerning the privacy violation while dealing with sensitive information, especially with unencrypted payload traffic. In consequence, the authors address the problem of cost-aware feature selection for IoT device classification and develop a cross-entropy-based algorithm that showed a faster and low-risk score than the brute force approach.

The feature selection can play a vital role in decreasing the amount of memory needed to store and process the data. Yet, few studies address the computational cost of the feature vector obtained in their experiment. Kumar *et al.* (121) demonstrate the high cost of training packet-based compared to other approaches. Palmese *et al.* (122) address storage cost by proposing an optimisation Storage-Accuracy framework. Their goal is to find the required number of features to be extracted and apply lossy feature compression for specific storage using scalar quantisation. The study maintained the feature importance rank provided with the RF design after initial training.

Pashamokhtari (111) design a dynamic technique that distributes complex features across several models, where costly models are required on demand. They proposed three models: (1) TCP SYN/SYN-ACK model that consists of TCP window size, option list and ports, (2) DNS model that inspects query names in DNS requests, and (3) IP model that trains IP addresses of cloud serves. The cost of the computation and inspection of the first two models are low, while the third model is expensive (i.e. packet-based) but only used if the confidence of the previous models is low.

Sivanathan *et al.* (12) discuss the trade-offs between cost, speed, and the performance of their feature vector. The study in (124) examined the cost of features by using a Correlation-based Feature Subset, a selection algorithm, and Information Gain (IG), a DT-based ML. It shows the incoming/outgoing remote flow bytes have the highest impact on the prediction. By eliminating 35 non-redundant features, and considering the 25 most relevant features with 8-byte flow counter the accuracy achieved 97% and saved 50% of flow entries and 37% of space complexity.

Wu *et al.* (126) introduce a times scale selection method to reduce computational resources. Each IoT device will have a different time scale where a single feature is extracted.

Kotak and Elovici (115) and Liu *et al.* (133) focus on encrypted payloads as features that are free from complex feature engineering. These payloads are processed as images, making the features applicable to any protocol.

The reviewed studies are based on passive scanning without performing deep packet inspection (DPI). However, analysing some header and unencrypted payload features can reveal sensitive information. Additionally, these features are vulnerable to being

spoofed. Therefore, most studies avoid using exposed features and instead apply statistical features to create robust fingerprint models.

3.4.3 Machine Learning Approaches

ML is demonstrating robustness in classifying local features of interest, including network feature classification for device identification (7). It provides viable solutions by deeply analysing data, extracting hidden features, and developing intelligent IoT device fingerprinting (70). Therefore, this section will review the potential of ML solutions for IoT device fingerprinting. The studies approaches are categorised into *feature-based classification* in Section 3.4.3.1, *behaviour-based clustering and pattern identification* in Section 3.4.3.2, and *adaptive and semi-supervised learning* in Section 3.4.3.3.

3.4.3.1 Feature-Based Classification

This focuses on approaches that classify IoT devices based on their known behaviour or features observed in training data (see Table 3.2). These approaches fall under supervised methods, as they rely on labelled features to learn patterns and make accurate predictions. The approaches can be further categorised as follows:

Tree-Based:
Decision Trees (DT) are favoured for their interpretability, allowing for clear insights into decision-making processes. They serve effectively in various classification tasks related to IoT device identification (109; 93; 99; 127). *J48 Decision Trees*, *OneR* and *PART* are DT based models. These can either be used as standalone classifiers or as base learners in ensemble methods (11).

Ensemble:
Ensemble techniques can be broadly categorised into *bagging* and *boosting*, both of which have found significant applications in IoT device fingerprinting. Bagging (Bootstrap Aggregation) focuses on reducing model variance by training multiple models on different subsets of the data. One of the most widely-used bagging methods is *Random Forest* (RF), which builds several decision trees on different random subsets of the data and then aggregates their outputs through majority voting. Due to its ability to handle complex and varied IoT device behaviours, RF has become a key method in numerous studies for IoT identification and management systems (2; 5; 103; 10; 122; 111; 124; 125; 12). This makes RF particularly effective for classifying diverse network traffic patterns while maintaining high accuracy and reducing overfitting.

On the other hand, boosting techniques sequentially train weak learners, each focusing on the errors made by the previous models. This iterative improvement helps achieve higher accuracy, particularly for difficult-to-classify IoT device instances. *Gradient Boosting* (GB) (5) and its variants, such as *Extreme Gradient Boosting* (XGBoost) (5; 92; 121) and *Light Gradient Boosting Machine* (LightGBM) (98), are widely adopted in IoT systems for their ability to handle large-scale and complex datasets. By focusing on misclassified instances during training, boosting methods excel in refining IoT device fingerprinting models, making them particularly useful in dynamic and heterogeneous IoT environments.

Bayesian:
Bayesian methods, including *Bayes Net*, utilise probabilistic models to capture relationships between features. This characteristic allows for the integration of uncertainty in device classification, enhancing robustness in dynamic environments (103).

Neural Networks:
Neural networks, especially *Convolutional Neural Networks* (CNN) and *Long Short-Term Memory* (LSTM) networks, are highly effective for high-dimensional data, making them particularly suitable for IoT fingerprinting applications. Their architectures capture intricate temporal and spatial relationships, which are essential for accurately identifying and classifying IoT devices based on their unique characteristics (100; 101). Several studies have highlighted the feasibility of using neural network and deep learning approaches for time series classification in the context of IoT (14). CNNs have been widely applied across various studies related to IoT fingerprinting (100; 101; 102; 121; 116; 104; 126), including lightweight variants explored in (114) that are optimised for resource-constrained devices. LSTMs also play a significant role, with research proposing hybrid models like the *LSTM-CNN* cascade classifier (14), which can effectively process sequential data from IoT devices. Other methods, such as the *Single-Layer Perceptron* (SLP) (115) and *Multilayer Perceptron* (MLP) (128; 3), have been employed to enhance classification performance in IoT environments. Additionally, a two-stage classification model that combines Logistic Regression (LR) with MLP has been proposed for improved accuracy (107). *Graph Convolutional Networks* (GCNs) have also been utilised to enhance classification accuracy in IoT fingerprinting applications (117). These hybrid architectures leverage the strengths of various models, ultimately improving the performance of fingerprinting systems in dynamic IoT environments (14; 107).

Few Shot Learning:
Supervised learning models indeed require extensive labelling of data to train effectively, which can be time-consuming and costly, especially when large datasets are involved. Few-shot learning emerged as an alternative approach to address this challenge. It enables models to learn from a very small amount of labelled data. This

technique is particularly useful when it is impractical or impossible to obtain a large labelled dataset. Few-shot learning leverages prior knowledge, either from related tasks or embedded model pre-training, to make accurate predictions with limited examples. Duan *et al.* (110) examine their ByteIoT approach under insufficient labelled data scenario using Few-shot learning. Jia *et al.* (112) and Zhao *et al.* (113) use a similar framework based on Few-shot learning for IoT traffic classification. After feature extraction, a feature comparator module is used to produce the final prediction via concise similarity in (112) and multiple convolution blocks connected sequentially in (113).

Incremental Learning:
Incremental learning (IL) is an ML technique where a model is continuously trained on new data over time, updating its knowledge and improving its performance. A binary classifier for each IoT device is a viable approach in scenarios where each classifier distinguishes whether a device belongs to a particular class or not. This setup can be incrementally updated when new devices are introduced, allowing the system to adapt to new devices without retraining the entire model from scratch. It was first proposed by Miettinen *et al.* (2) is the first to introduce a scalable approach, named IoT Sentinel, using binary classifiers. IoT Sentinel has a two-fold classification: training a binary classifier for each device type, then breaking the tie of multiple matching unknown fingerprints using edit distance-based.

The *Distance-based* methods, such as *k-Nearest Neighbours* (kNN), play a pivotal role in effectively classifying devices based on their feature representations. Using distance metrics to measure similarity between devices, kNN can rapidly identify new or unknown IoT devices by comparing their attributes with those of known devices in the feature space. This method is particularly beneficial in dynamic environments, where the ability to quickly adapt to the introduction of new devices is crucial for maintaining an accurate fingerprinting system. The simplicity and effectiveness of kNN make it a suitable choice for scalable IoT fingerprinting solutions, enabling real-time identification while minimising computational overhead (110). Therefore, kNN is employed by Duan *et al.* (110) to introduce ByteIoT, an IoT device identification system. ByteIoT employs the Hellinger distance metric to measure the similarity between two discrete probability distributions and utilises DBSCAN for efficiency optimisation.

The *Broad Learning System* (BLS) is an ML framework that offers an efficient alternative to DL models, particularly in terms of time complexity. Unlike traditional DL approaches, which typically involve retraining the entire model when new data is introduced, BLS updates only certain parts of the model, such as the weights, without requiring full retraining. This makes it much faster and more computationally efficient, especially when working with large datasets or IL scenarios. In the context of IoT, BLS can be used to update models with new samples from known devices,

enhancing predictions while maintaining low computational costs as shown by Zhang *et al.* (108).

The *Class Incremental Learning* (CIL) is an ML approach where a model is continuously updated to recognize new classes without forgetting previously learned ones. In the context of IoT, CIL allows the system to integrate new device types into its classification framework without retraining the entire model, ensuring that both new and previously identified devices are accurately classified. This technique enables scalability as new IoT devices are introduced over time. Wang *et al.* (26) utilise a CIL approach, employing a transformer network as a multi-classifier for IoT network traffic. In this framework, training data exemplars are stored in memory, allowing the classifier to identify IoT devices effectively. When new devices are introduced to the network, the classifier is incrementally retrained using both past samples and new data from these devices. This method enables the model to adapt to changes in the environment without forgetting previously learned information, ensuring accurate identification of both existing and new IoT devices.

3.4.3.2 Behaviour-Based Clustering and Pattern Identification

Behaviour-based clustering and pattern identification techniques leverage *unsupervised machine learning* to analyse and categorize IoT devices based on their observable behaviours and interactions within a network. Unlike supervised methods, which require labelled data and prior knowledge of device characteristics, these techniques can autonomously identify patterns and group devices without such constraints. In the context of IoT fingerprinting, these approaches facilitate the detection of device types and anomalies by analysing traffic patterns and behaviours. By exploring the inherent structure of network data, behaviour-based methods enable effective identification and classification of devices, enhancing security and network management. This adaptability is particularly valuable in dynamic environments where new devices frequently connect, allowing for continuous learning and improved identification of IoT devices. The literature approaches can be categorised as clustering and density-based (see Table 3.3).

Clustering:

Clustering approaches are used to group similar IoT devices based on their behavioural patterns and interactions within a network. By organising devices into distinct clusters, these methods facilitate the identification of device types and enhance the performance of IoT fingerprinting. *k-means* clustering is used to create a distinct cluster for each identification domain (one cluster per IoT device) in (120). Additionally, some studies independently train and update each cluster (21; 22). *kNN*, typically a supervised learning model, can be adapted for unsupervised learning

TABLE 3.2: Characterisation of Surveyed Papers Having Feature-Based Classification as ML Approach

Year	Ref.	Network Protocol	ID[1]	ML	Feature			Dataset	Performance[3]	Scalable	Limitation
					#	Source	Type[2]				
2017	(5)	WiFi	C	RF, GB, XGBoost	NA (2)	Session	NA	(5)	ACC 99.281%	NA	The method consumes time as it requires extracting features after the end of the session. Limited to TCP traffic. It can not be considered a device fingerprinting as it does not use traffic features for identification. The dataset is small.
2017	(2)	WiFi, Ethernet, ZigBee and Z-Wave	MF	RF	23	Packet	H	(2)	ACC 81.5%	Yes	The method is applied to IoT device identification during setup. It can not distinguish between devices with identical vendors, functions, hardware, and software versions. It is unfeasible due to the increase of IoT devices, where it is inefficient to have a binary model for each class (identification domain type) as the number of models will increase exponentially with the number of classes. Moreover, each sample should be tested across the models.
2018	(14)	Wireless, wired and ZigBee	C	LSTM-CNN	6	Flow	HS	(12)	74.8%	No	The tendency for accuracy to decrease as the number of categories increases. The dataset is small.
2019	(10)	WiFi, Ethernet,	M	RF	67	Flow	H, HS, FS, TS, PS	(2)	F1 90.3%	NA	The dataset (2) provides traffic traces of device setup. The challenge is to detect devices sharing the same vendor. IT can detect unauthorised devices, but can not update the model with new authorised devices.
			DI						F1 89%		Limitation in dataset.
2019	(11)	WiFi	M	PART	212	Packet	H, FS, PS	(2)	no info	NA	The method can not identify models from the same vendors having similar network behaviour. Applied to only TCP traffic. It is unfeasible due to the increase of IoT devices, i.e. a class for every model. The dataset (2) provides traffic traces of device setup.
				Two-level classification model using PART and J48 DT					ACC 82%		The dataset is restricted to identify devices during device setup. There are difficulties in identifying some devices having the same vendor. It is impractical due to the increase of IoT vendors and devices.
2019	(12)	Wireless, wired and ZigBee	DI	Multi-stage classifier: Naive Bayes Multinomial and RF	12	Flow	H, HS, FS, TS, P	(12)	ACC 99.88%	NA	Complicated multistage architecture. The extraction cost is high for payload-based features (bags of domain names and CS). It is impractical as the entire model requires retraining; when adding a new identification domain or upgrading the firmware of known devices. Features are vulnerable to security threats.
2019	(3)	Wireless, wired and ZigBee	DI	MLP	=	Flow	=	(12)	ACC 99.8%	No	The MLP classification model is computationally expensive, though it provides fast predictions of new samples. The dataset is small.
2019	(9)	WiFi and wired	C	AdaBoost	over 21	Flow	H, HS, FS,	(12)	ACC 95.5%	No	Fingerprinting has to wait until the end flow. Hence, not suitable for real-time detection. The dataset is small.
2019	(125)	Wireless, wired and ZigBee	DI	RF	3	Flow	HS, FS	(12)	F1 96%	No	The analysis showed the similarity of the features across IoT devices that can cause performance degradation.
2020	(103)	ZigBee	C	Bayes Net	1	Flow	T	private	ACC 91%	No	The evaluation showed over 91% in fingerprinting the category of ZigBee and Z-wave separately. The performance of the model in identifying Z-IoT device categories is still questionable. The data set is small.
		Z-wave		RF					ACC 93.25%		
2020	(111)	Wireless, wired	DI	RF	6	Flow	H, P	private	99.9%	No	No info about dataset and device type identification. The approach was tested on a small dataset achieving high accuracy. This might drop when using a larger dataset with different behaviour.
2020	(114)	Wireless, wired	C	lightweight CNN	297	Packet	H	(5)	87.67%	No	Low accuracy. Unable to identify Lamp devices. The dataset is small.
2020	(99)	Wi-Fi and Ethernet	DI	DT	18	Session	H, HS, FS, TS, P	(2) + private	97%	No	based on non-encrypted traffic, where features can be spoofed. Training includes non-IoT (smartphones).
2020	(101)	WiFi	DI	CNN	19	Packet	H	(12)+ private	F1 99.9%	Yes	It uses two CNNs per IoT device: one for continuous detection and another for population estimation upon device detection. It needs robust management and potentially impacts detection times, showing a few minutes of latency to detect IoT devices.

Continue on the next page

[a]ID column: DI-Device Instance, M-Model, MF-Model and Firmware, C-Category, VT-Vendor and Type, V-Vendor.

[b]Feature Type column: H-Header field, HS-Header Statistics, P-Payload, PS-Payload Statistics, FS-Flow Statistics, TS-Time interval Statistics.

[c]Performance column: ACC-Accuracy, F1-F1-Score, TPR- True Positive Rate.

TABLE 3.2: Characterisation of Surveyed Papers Having Feature-Based Classification as ML Approach (Cont.)

Year	Ref.	Network Protocol	ID[1]	ML	Feature			Dataset	Performance[3]	Scalable	Limitation
					#	Source	Type[2]				
2020	(124)	Wireless, wired and ZigBee	DI	Hierarchical structure RF	14	Flow	FS, TS	(12)	F1 98.6%	No	Discussed cost and performance trade-off. The data set is small.
2020	(104)	wireless	C	CNN	1	Flow	PS	private	AP 97.2%	No	The size of the input effect on the processing time. The optimisation solution using PCA reduces the overhead but degrades the accuracy. The dataset is small.
2021	(93)	WiFi	M	DT	69(1)	Flow	No info	(136)	F1 99%	No	
2021	(117)	WiFi	DI	GCN	4	Session	H	(12)	TPR 95%	No	Challenge of using TLS message type under different TLS version. The data set is small.
2021	(131)	WiFi	VT	No info	21	Packet	H, HS	(2) + Private	ACC 3%-11%	No	Feature selection algorithm is time consuming.
			M						94.5%		
2021	(129)	WiFi	M	1-D CNN (with residual connections)	23	Flow	HS, TS	(94)	ACC 93.96%	No	The dataset is small.
2021	(97)	Wireless	DI	RF	28	Session	H, FS, TS, PS	(97)	ACC 96%	No	features are affected by firmware upgrade and dependent on user activity in the same device.
2021	(90)	WiFi, Ethernet and ZigBee	M	GB	19	Flow	H, PS	private	ACC 93.37%	Yes	The approach provides a model for each vendor that trains its devices, which needs to retrain the entire model for new devices belonging to that vendor. The method is vulnerable to security threats. The dataset is small.
2022	(92)	WiFi	DI	single stage Xgboost	70	Session	H, HS, FS, TS	private	BA 99.8%	No	The dataset is small.
2022	(127)	Wireless, wired and ZigBee	DI	DT + RF		Flow	H, HS, FS, TS	(12)	F1 73% or 96%	No	Need further improvement, imbalance data issue,
2022	(128)	WiFi	M	DL	23	Flow	HS, TS	(94)	ACC 96.33% and F1 95.99%	No	It needs labelling normal and malicious traffic along with IoT device models. It can only identify known devices. It requires a long time to achieve better results. The dataset is small.
2022	(100)		DI	CNN	1	Packet	P	(12) + private	ACC 93% and F1 92.96%	No	The model is optimised autonomously to ensure the robustness of recognising known IoT devices, but not unknown devices. The dataset is small.
2022	(102)	WiFi	M	CNN	19	Packet	H	(12)+ private	FS 99.9%	Yes	TCP IoT devices??
2022	(98)		DI	LightGBM	10	Flow	H, HS	(98)	AP 98.48%.	No	The features are specific protocols and services, and problems arise with IoT devices of different protocols (e.g., MQTT TLS). The higher threshold might detect concept drift.
		Wireless, wired and ZigBee						(12)	AP 98.93%		
2022	(121)	Wireless, wired and ZigBee	DI	XGBoost	6	Flow	H, HS, FS, TS, P	(12)	ACC 98.96%	No	Features are vulnerable to spoofing. The dataset is small.
				CNN					ACC 97.52%		
2022	(107)	Wireless, wired and ZigBee	DI	two stage (LR and MLP)	11	Packet	H, TS	(12)	ACC 99.9% - F1 99.6%	No	Features are vulnerable to spoofing. The overhead of monitoring traffic for collecting traces. Time-consuming for training data and computational overhead for model training.
								(96)	ACC 99.8% - FS 99.7%		
2022	(108)	Wireless, wired and ZigBee	DI	BLS (k-means and SVM preporocessing)	21	Packet	H, TS	(12)	ACC 84.8% - F1 80.0%	Yes	Features are vulnerable to spoofing. Features are not strong for fingerprinting. BLS can update the model for new samples of known types only. The data set is small.

Continue on the next page

[a]ID column: DI-Device Instance, M-Model, MF-Model and Firmware, C-Category, VT-Vendor and Type, V-Vendor.

[b]Feature Type column: H-Header field, HS-Header Statistics, P-Payload, PS-Payload Statistics, FS-Flow Statistics, TS-Time interval Statistics.

[c]Performance column: ACC-Accuracy, F1-F1-Score, TPR- True Positive Rate.

TABLE 3.2: Characterisation of Surveyed Papers Having Feature-Based Classification as ML Approach (Cont.)

Year	Ref.	Network Protocol	ID[1]	ML	Feature			Dataset	Performance[3]	Scalable	Limitation
					#	Source	Type[2]				
2022	(126)	Wireless, wired and ZigBee	DI	CNN	2	SW	FS	private + (?)	ACC 99.72% - FS 99.77%	No	It did not discuss the time complexity of selecting the time scale at the offline and online stages.
								(12)	ACC 99.97% - FS 99.97%		
2022	(109)	WiFi, Z-devices, Bluetooth	DI	DT	30	P	H, FS, PS	(2)	ACC 83.3% - FS 86.1%	No	Features are vulnerable to spoofing. The data set is small.
								(89; 12)	ACC 94.1% - FS 93.5%		
2022	(110)	Wireless, wired and ZigBee	DI	kNN (DBSCAN data reduction)	2	Packet	H	(12)	ACC 99.81%	Yes	Identity new device dependent on human intervention. The model requires only 30 minutes to monitor the traffic.
								(95)	ACC 97.91%		
								Private	ACC 96.24%		
2022	(115)	Wireless, wired and ZigBee	DI	SLP	1	Session	Packet	(12)	99%,	No	The scope of this research is limited to TCP protocol traffic. Thus, further tests should be pursued to cover a wider range of network protocols. DL usually is computationally expensive. The dataset is small.
									unknown 98.57%		
2022	(116)	IP-based	DI	CNN	1	Session	H/F?	(12)	ACC 98.828%	No	Limited to IP packet only, the computational overhead of the traffic deep learning module. Can not identify IoT devices from the same vendor and functionality (Fire TV, Echo Dot, Echo Plus, and Echo Spot).
								(96)	ACC 93.941%		The data set is small.
2022	(119)	DL and FL	C	FedCNN	over 297	Session	H, HS	(5)	ACC 87.2% - FS 83.9%	No	The method is based on a decentralised approach that shows decreasing in accuracy with more clients for local training. Although it shows its efficiency, it has lower performance than the centralised approach.
				FedLSTM					ACC 88.2% - F1 84.9%		
2022	(122)	Wireless, wired and ZigBee	DI	RF	209	Flow	HS, TS	(12)	about BA 85%	No	The accuracy is low. Needs an exhaustive search to maximise accuracy without overfitting. The data set is small.
2022	(118)	Wireless, wired	DI	RF	2	Session	H, HS	(97)	ACC 92%	No	based on IP and port address (soft features). Some devices are misclassified. Can not distinguish between Planex cameras.
				2 methods (specific service and fingerprint set)	402				lower FP		Challenge of ID for IoT devices having the same vendor.
2022	(110)		DI	kNN	2	Packet	H	(12)	ACC 99.77%	Yes	It needs 30 minutes for feature extraction. It does not discuss the scalability when adding new devices incrementally.
								(95)	ACC 97.91%		
								private	ACC 96.42%		
				kNN with Few-shot learning				(12)	ACC 98.28%		
2022	(112)	.	DI	Cosine similarity for classification with Few-shot learning	2	Session	H,P	(12)	F1 98.42%	No	.
								(96)	FS 96.62%		
2022	(113)		DI	CNN with Few-shot learning	1	Session	P	(12)	Known class F1 99.01%, Unknown class F1 96.57%	No	Detect unknown traffic but do not classify them.
								(96)	Known class F1 87.43%, Unknown class F1 88.66%		

Continue on the next page

[a]ID column: DI-Device Instance, M-Model, MF-Model and Firmware, C-Category, VT-Vendor and Type, V-Vendor.

[b]Feature Type column: H-Header field, HS-Header Statistics, P-Payload, PS-Payload Statistics, FS-Flow Statistics, TS-Time interval Statistics.

[c]Performance column: ACC-Accuracy, F1-F1-Score, TPR- True Positive Rate.

scenarios through techniques such as clustering, where it identifies the nearest neighbours of a point without using labelled data. Marchal *et al.* (23) adopt this technique to explore data structure and autonomously create clusters and labels in device identification. Their proposed model operates without prior knowledge of device types and requires only four parameters, including traffic capture duration, sampling period, k value, and threshold distance for kNN, to be set before system deployment.

Density-based:
Density-based clustering methods are effective for identifying clusters of varying shapes and sizes by grouping dense data points while distinguishing outliers in low-density regions. These approaches are particularly beneficial for IoT fingerprinting, where device behaviours can vary significantly and noisy or anomalous data is prevalent. For instance, DBSCAN is utilized in studies (106; 123) for effectively identifying clusters of devices based on their interactions while marking points in less dense areas as outliers. This capability enhances the accuracy and robustness of device identification in dynamic IoT environments.

3.4.3.3 Adaptive and Semi-Supervised Learning

Adaptive and semi-supervised learning approaches combine labelled and unlabelled data to enhance IoT device identification. By leveraging large amounts of unlabelled data alongside a smaller set of labelled examples making them valuable in dynamic environments where new devices frequently emerge. The literature approaches can be categorised as *clustering with classifier hybrid* and *pseudo-labelling and incremental learning* (see Table 3.4).

Clustering with Classifier Hybrid:
Semi-supervised approaches often cluster devices initially, then classify them using labelled data. In IoT, clustering-based classifier hybrid approach enhances the identification of IoT devices in complex environments, where labelled data are scarce, yet a large amount of raw data is available. Clustering techniques, such as k-means, *Hierarchical Dirichlet Process* (HDP), and*Ordering PoinTs to Identify the Clustering Structure* (OPTICS), are widely used to organise unlabelled data into clusters. Once clusters are formed, classification algorithms, like RF, are applied to assign labels to the data points (1; 18; 19; 20).

Pseudo-Labelling and Incremental Updates:
In scenarios where manually labelling data is not feasible, pseudo-labeling offers a powerful alternative. In the ByteIoT system proposed by Duan *etal.*(110), a kNN classifier is used to predict and assign labels to unlabelled IoT traffic data, effectively

TABLE 3.3: Characterisation of Surveyed Papers Having Behaviour-Based Clustering as ML Approach

Year	Ref.	Network	ID Protocol	ML Domain[1]	Features			Dataset	Performance[3]	Labelling	Scalable	Limitation
					#	Source	Type[2]					
2019	(21)	WiFi	DI	K-means	16	Flow	HS,TS	(89)	ATP 94.5%	NA	Yes	The challenge to scale with fixed-sized SDN switches (124). The high cost of having a model per IoT device type. The dataset is small.
2019	(23)	WiFi and Ethernet	AT	kNN	33	Flow	FS	(23)	FS 98.2%	A	Yes	The model requires only 30 minutes to monitor the traffic and identify a single fingerprint of a specific abstract device type. It requires a least five fingerprints to train a new abstract type, which requires 2.5 hours of monitoring. Also, it requires a cloud service that could raise latency and user privacy concerns (**?**). Some IoT devices do not produce period flows such as surveillance devices. The proposed system needs human intervention to initialise the parameters before deployment. Moreover, the architecture shows a centralised cloud server that increases the complexity and overhead when updating the model (19).
2021	(120)	WiFi	DI	VAE and K-means	14	Flow	H, HS, FS, TS, PS	(12)	86.7%	A	No	k must be initialised. Some features are vulnerable to attacks. Manual labelling is required by the administrator. The dataset is small.
2021	(22)	Wireless, wired and ZigBee	DI	k-means	9	Flow	H, HS, FS	(12)	98%	A	Yes	the approach consumes time (collecting data and testing). The dataset is small.
2022	(106)	WiFi, Zigbee	DI	DBSCAN	2	Packet	H	(91)	Acc 100%	No info	No	It was tested on TCP sessions. For TLS connection, the study considers only unencrypted TLS record headers. The model reaches 100% around a 5-second timeout, however, it drops to 80% without timeouts. It raises a concern about the negative impact of timeout due to the variation in communication delays between devices. Also, scaling up signatures is a major issue. The dataset is small.
2021	(123)	No info	No info	DBSCAN	3	Flow	HS	simulation	No info	NA	No	It is based on soft features (IP and ports)

[a]ID column: DI-Device Instance, M-Model, MF-Model and Firmware, C-Category, VT-Vendor and Type, V-Vendor.

[b]Feature Type column: H-Header field, HS-Header Statistics, P-Payload, PS-Payload Statistics, FS-Flow Statistics, TS-Time interval Statistics.

[c]Performance column: ACC-Accuracy, F1-F1-Score, TPR- True Positive Rate.

expanding the labelled dataset. This process improves the classifier's accuracy by leveraging both labelled and unlabelled data.

Incremental updates further enable IoT device identification models to evolve with time. AutoIoT (130) is one such system that applies semi-supervised learning to automatically update its model when new devices are introduced. By clustering traffic features and applying probability-based classification, the system can identify and adapt to new device types while keeping manual intervention to a minimum. This adaptive framework ensures that IoT fingerprinting systems can maintain high accuracy even in constantly changing networks.

3.4.4 Identification Domain

Different levels of granularity are used to identify a device, including Vendor (e.g., D-Link), Category (e.g., Camera), Model (e.g., DCS-932L), Device Instance, i.e. serial numbers/MAC or IP address, and Activity, i.e. Behaviour (e.g., live view). Some of the studies use the term *Device Type* to indicate a type or multiple types of identification (e.g., Vendor and Model). To avoid any confusion in the literature, this section will explain the identification domain (ID) of each study.

Table 3.5 shows the types of IoT device identification domains in the literature. Many studies are interested in identifying devices via their ⟨*Device Instance*⟩, where the main challenge is to distinguish between different devices having the same model and functionality based on their behaviour. The ⟨*Model*⟩ and ⟨*Vendor, Type*⟩ are very similar except the latter does not distinguish between models from the same vendor (e.g., D-Link Camera). Yu *et al.* (18) argued the significance of fingerprinting ⟨*Vendor, Type*⟩ regardless of ⟨*Model*⟩ and ⟨*Version*⟩ claiming the vulnerability is associated with the device itself.

Few studies identified the ⟨*Category*⟩ of IoT devices where similar devices are in a single group (e.g., Camera and Lights). There is variation in the approaches of categorising the IoT devices among the reviewed studies as shown in Table 3.6, where mostly follow the datasets categorisation they utilise, such as UNSW IoT Traces dataset (12; 137) and ProfilIoT dataset (5).

The activity or event fingerprinting of IoT devices (e.g., Switch on and off) and traffic relations (e.g., Browsing and File transfer) have gained the attention of many studies such as (106; 138; 125). However, this survey focuses on detecting the presence of IoT devices, rather than their activities or traffic relations.

A different approach proposed by Marchal *et al.* (23) which fingerprints an ⟨*Abstract Type*⟩ that represents IoT devices having the same vendor and similar communication behaviour (140). For example, they classified iKettle2 and

TABLE 3.4: Characterisation of Surveyed Papers Having Adaptive and Semi-Supervised Learning as ML Approach

Year	Ref.	Network	ID Protocol	ML Domain[1]	Features #	Features Source	Features Type[2]	Dataset	Performance [3]	Labelling[4]	Scalable	Limitation
2018	(18)	WiFi and Ethernet	V	HDP Bayesian model	18	Packet	H	(2) + private	ACC 100%	M	Yes	Generating fingerprints using DHCP features will apply to only devices communicating over IP. DHCP is vulnerable to spoofing. The method fails to distinguish between different device types of an identification domain from the same vendor as they have an identical codebase and DHCP option sequence. Human intervention is required for labelling new types. The data set includes non-IoT devices.
			VT						ACC 88%			
			M		23 (2)			(2)	ACC 96%			The method identifies the type during initial communication to the network when the device boots up.
2019	(1)	WiFi and Ethernet	DI	k-means for training the model and RF for classification	111	Flow	HS , FS, TS, PS	private	ACC and FS ≈ 97%	Manual for initial training and automatic for retraining	Yes	It requires to initialise k as an input to k-means clustering. It does not consider the costs of acquiring the features. The architecture shows a centralised approach (i.e., control logic) that increases the complexity when updating the model and extensive router configurations (19; 101).
2020	(20)	.	C	RF,AE and OPTICS	297	Packet	H	(5)	ACC 91.2%	M	No	Human intervention is required to authorise unknown devices. The dataset is small.
2021	(19)	WiFi and Ethernet	DI	K-means and RF	15	Session	FS, SH	(12)	FS 99.2%	initial label, then auto	Yes	It uses an integer number for labelling. Memory resource exhaustion concerns with increasing training data. The dataset is small. Requires retraining the entire model. The detection of unknown samples is not clear.
2022	(25)	WiFi and Ethernet	DI	PCA+ CNN	129	Flow	HS, PS, FS,TS	(12)	ACC 99.83%	Manual for initial training and automatic for new types	Yes	The challenge is a new type and a known type having the same vendor and functionality. The accuracy will be degraded if device instances of the same type join the network (e.g., two Apple TVs 4th Gen). An administrator is required to re-label the new types manually and check the correctness of the classification.
								(95)	ACC 97.20%			
2022	(110)		DI	kNN	2	Packet	H	(12)	ACC 98.84%	Initial manual labelling then use the pseudo-labelling technique	Yes	Each device should train N-labelled samples first. It does not discuss the scalability when adding new devices incrementally.

[a]ID column: DI-Device Instance, M-Model, MF-Model and Firmware, C-Category, VT-Vendor and Type, V-Vendor.

[b]Feature Type column: H-Header field, HS-Header Statistics, P-Payload, PS-Payload Statistics, FS-Flow Statistics, TS-Time interval Statistics.

[c]Performance column: ACC-Accuracy, F1-F1-Score, TPR- True Positive Rate.

[d]Labelling column: M-Manual, A-Automatic, MA-Manual initial and Automatic afterwards.

TABLE 3.5: IoT device Identification Domains

Identification Domain	References
⟨*Device Instance*⟩	(10; 120; 22; 125; 99; 12; 124; 117; 97; 127; 21; 1; 110; 113; 126; 112; 118; 22; 106; 25; 115; 107; 98; 121; 92; 19; 116; 108; 122; 109; 100; 3; 26)
⟨*Model*⟩	(90; 11; 10; 128; 93; 129; 101; 102)
⟨*Model, Firmware*⟩	(2)
⟨*Vendor, Type*⟩	(18; 131)
⟨*Category*⟩	(14; 9; 103; 104; 20; 114; 139; 119)
⟨*Vendor*⟩	(18; 131)
⟨*Device instance, Activity*⟩	(106)
⟨*Device instance, Relation*⟩	(138)

TABLE 3.6: IoT Device Categories Identification Domain

Ref	Categories
(139)	PC/server, Wireless LAN access point, Network attached storage (NAS), Printer, Switching hub
(104)	Video surveillance equipment, Real-time condition monitoring equipment, Remote-Controlled actuator, Event-driven equipment, Non-IoT equipment
(103)	Arrival Sensor, Outlet, Button, Water Sensor, Motion Sensor, Multipurpose Sensor, Lock, Door/Window Sensor, Dimmer, Switch
(137)	Hubs, Cameras, Switches & Triggers, Air quality sensors, Healthcare devices, Light Bulbs, Electronics
(5)	Baby monitor, Lights, Motion sensor, Water Sensor, Security Camera, Smoke detector, TV, Watch, Sockets, thermostat

SmarterCoffee devices, produced by Smarter, as the same abstract type class. Their argument is the identification of an abstract type is sufficient for device fingerprinting compared to an actual device model, which the latter is difficult to keep track of with the rapid growth of IoT device models and vendors. Furthermore, this type of identification can be associated with a set of policies that can be useful for anomaly detection, network resource allocation, and identification and isolation of vulnerable IoT devices. The study in (141) has a different approach, where it defines IoT device classes based on coefficients of variation ratio of network traffic. The study does not fit this survey as a single class includes many devices from different vendors and categories regardless of their functionality or purpose.

3.4.5 Scalability

Scalability in IoT fingerprinting involves managing both the addition of new device types as they emerge and the efficient handling of many devices (23). In this review, we focus on scalability challenges particularly relevant to identifying new devices with minimal resource overhead, such as memory and storage efficiency, retraining and updating strategies, adaptability to unknown devices, and efficient labeling mechanisms.

3.4.5.1 Memory and Storage Costs

Efficient memory usage is crucial in scalable IoT systems, especially when handling vast datasets or real-time data streams. Some approaches fall under *unbounded memory* usage, where models, such as the IoT Sentinel proposed by Miettinen *et al.*(2), use binary classification for each IoT device type. This leads to a linear increase in memory consumption as more devices are introduced, making it impractical at scale due to the high number of required classifiers.

In contrast, *fixed memory size* approaches, such as those employed by Fan *et al.* (25) and Wang *et al.* (26), use transfer learning and CIL, respectively, to minimise memory consumption by storing knowledge representations, which helps mitigate catastrophic forgetting. While this keeps memory usage under control, it requires careful management of stored knowledge to balance memory constraints with scalability.

3.4.5.2 Unknown IoT Device Detection

Detecting unknown IoT devices that were not part of the training data presents significant scalability challenges. As the number of IoT devices grows and new types emerge, systems must efficiently handle and identify these unknown devices effectively. *Threshold-based* approaches involve setting either static or dynamic thresholds for detection. We observed only static threshold techniques for unknown IoT device detection, as seen in studies like Zhao *et al.* (113), that may lead to high false positive or negative rates. Kotak *et al.* (115) follows the same approach by setting a threshold for each IoT device based on extensive experiments to minimise the number of epochs required for maximum accuracy, which is impractical for large-scale IoT environments. *Distance-based metrics* such as Euclidean distance and Edit distance can be used to detect the presence of unknown IoT devices to compare new fingerings against known fingerprints, though these methods face challenges in computational complexity when scaling to numerous devices (2; 19). Lastly, the use of *Statistical tests* such as the Kolmogorov-Smirnov test, as applied by Fan *et al.* (25), allows for the comparison of probability distributions to detect behavioural differences in new devices. However, this approach requires a known distribution of data and may encounter memory constraints in large datasets.

3.4.5.3 Adaptability

Adaptation in IoT device identification systems is crucial for scalability, especially in dynamic environments where new devices frequently emerge or existing devices exhibit changing behaviours. Adaptation mechanisms can be categorised based on

how the system responds to new data, either through *updating* existing models or *retraining* them.

Updating the system with new devices can often be more efficient than retraining from scratch, as it saves time and computational resources. For instance, Miettinen *et al.* (2) allow the system to update when new IoT devices are detected by adding a binary classifier for each type. While this approach facilitates the inclusion of new devices, it does not address the need to improve the performance of existing classifiers.

Retraining machine learning models, particularly in a rapidly evolving IoT landscape, can be computationally expensive and time-consuming. Constant retraining is impractical due to the increasing number of IoT devices, as emphasised by Marchal *et al.* (23). Typically, retraining is performed in periodic batches, either at fixed intervals or when triggered by significant changes, such as the introduction of new device types.

For example, Jiao *et al.* (90) propose a multi-level identification framework that triggers updates only when a new vendor or device type is detected. In this framework, the system updates the vendor model and retrains it to classify only the devices associated with that particular vendor.

To mitigate the drawbacks of frequent retraining, methods like transfer learning can be employed to reuse pre-trained models, adapting them to new tasks with minimal retraining. Techniques such as few-shot learning and knowledge replay further facilitate adaptation to new devices while retaining knowledge of previously learned ones. For instance, Fan *et al.* (25) utilise transfer learning in their AutoIoT framework, reducing the need for complete retraining and helping to mitigate forgetting.

Additionally, CIL enables on-demand retraining for fixed classes using a pre-trained model. When new device types or classes emerge, the system can incrementally update its knowledge while maintaining the performance of older classes (142; 143). CIL leverages techniques like regularisation, replay methods, and knowledge distillation to prevent the loss of past information. However, Wang **et al.** (26) note that employing CIL with fixed-size knowledge replay can lead to catastrophic forgetting. As the system adapts to new devices, the limited memory size restricts its capacity to retain knowledge about older devices, potentially overwriting or discarding this information, which hinders its ability to maintain performance on previously learned classes while accommodating new ones.

3.4.5.4 Labelling New IoT Devices

Labelling new IoT devices is a key challenge in achieving scalability in IoT fingerprinting systems. To handle the ever-growing number of devices efficiently,

scalable approaches must minimise the need for manual intervention. *Automatic labelling* methods offer a more scalable solution by leveraging ML techniques that can handle new devices without human input. For instance, unsupervised clustering techniques like K-means and hierarchical clustering automatically group devices with similar behaviours (23; 120; 22). Semi-supervised learning further enhances scalability by using small amounts of labelled data alongside large pools of unlabelled data to label new devices (1; 110; 19; 25). On the other hand, *manual labelling* methods require human input, which limits their scalability as in supervised approaches.

3.5 Characterisation Based on datasets

This section discusses the IoT datasets used in the reviewed studies regarding availability, type of network, number of IoT devices, size, and the type of raw data.

3.5.1 Public Datasets

Several studies adopt a public dataset to evaluate the proposed IoT device fingerprinting approach (see Table 3.7).

IoT Sentinel Dataset. It is one of the significant datasets among early IoT fingerprinting studies (2). It is designed for a Small-Office/Home-Office (SOHO) environment with 31 IoT devices spanning cameras, hubs, switches, gateways, plugs, motion sensors, and appliances. The dataset consists of 27 different models, where four models represent two distinct types (e.g., TP-Link plugs HS100/TP-Link plugs HS110 and Edimax plugs 1101W/Edimax plugs 2101W). The traffic traces are recorded during initial communication and saved as packet capture (PCAP) files. Each setup is repeated at least 20 times per device model producing 550 raw PCAP files, with a total size of 64 MB. The IoT devices in this dataset use different network protocols, including WiFi, Ethernet, ZigBee, and Z-Wave. The authors developed this dataset for IoT device fingerprinting to identify devices using 23 features, where these features are enclosed with the dataset.

ProfilIoT dataset. The dataset encompasses features extracted from typical smart home devices including 10 different IoT device categories, including a camera, printer, sensors, smart TV, appliances, and smartwatch (5; 148). Wireshark was used to monitor the traffic and save it in a PCAP file. A total of 2405 samples of 297 features were extracted, such as port numbers and packet size.

UNSW IoT Traces. The University of New South Wales (UNSW) in Sydney published this dataset to utilise the traffic flows for IoT device fingerprinting using network traffic characteristics (12). It represents a smart environment, covering human and

TABLE 3.7: Public IoT Datasets

Dataset	Year	Network Type	Network Traffic Type	Dataset Size			Cited Dataset
				# IoT dev.	# PCAP files	Total size	
ProfilIoT dataset (5)	2017	SHN	No info.	22	2405 feature vector	No info.	(5; 20; 114; 119)
IoT Sentinel (2)	2018	SOHO	IoT device setup traffic	31	550	64 MB	(2; 10; 11; 18; 133; 99; 109)
UNSW IoT Traces (12)	2018	CAN	Active and idle	23	20	≈9.5 GB	(12; 14; 9; 115; 120; 100; 101; 102; 121; 107; 108; 22; 110; 116; 122; 19; 117; 113; 124; 125; 127; 126; 98; 3; 109; 25; 26)
N-BaIoT (94)	2018	SHN	Malicious traffic features	9	NA	9.33 GB	(129; 128)
IoT Benign Traces (89)	2019	SHN	Active and idle	27	27	35.267 GB	(21; 109)
The Mon(IoT)r (96)	2019	SHN	Active and idle	81	No info.	No info.	(107; 116; 113)
YourThings Data (95)	2019	SHN	IP traffic	63	3744	309.2 GB	(110; 25)
IoTDNS Data (13)	2019	SHN	DNS traffic	60	2	366.68 MB	-
KDDIIoT-2019 (98)	2019	SHN	IPFIX records	25	23,089,118 records	3,259 MB	(98)
IoT-23 Dataset (benign traffic) (144)	2020	SHN	Active	3	3	≈390 MB	-
LSIF (145)	2020	SHN	Active	22	440	16 GB	-
PingPong (91)	2020	SHN	Active	19	No info.	No info.	(106)
IoT-deNAT (146)	2020	SHN	idle	8	No info.	91.4 GB	-
IoT IPFIX records (97)	2021	SHN	IPFIX records	26	over 9 million records	0	(97; 118)
D-Link IoT (147)	2021	SHN	Active and idle packets	14	704	6 GB	-
			Active and idle MAC frames	11	437	171.4 MB	-
IoT Network Traffic (136)	2020	SHN	111 features extracted from 15 minute traffic interval	16	16 csv	3.769 MB	(1; 93)

ideal activities. The dataset is a collection of 20 PCAP files over 20 days. It has 23 IoT devices and 7 non-IoT devices, including cameras, bulbs, plugs, motion sensors, appliances, and health monitors. The total size of the raw data is approximately 9.5 GB. The authors also made their processed features and additional tools, to derive statistics of IP, NTP, and DNS flows, available for interested researchers.

IoT Benign and Attack Traces. UNSW provides another IoT dataset containing malicious and benign IoT network traffic traces using 10 and 27 IoT devices, respectively (89). The traffic traces are captured over a month and saved in PCAP files, where each corresponds to traffic collection over a day. There are 17 and 27 PCAP files for malicious and benign data, respectively. The authors developed the dataset for detecting anomalous by measuring the network behaviour of IoT devices over the

dataset. For IoT device fingerprinting, the study in (21) used only *IoT Benign Traces* without attacks to study the behaviour of the devices.

The Mon(IoT)r. The Mon(IoT)r Research Group at Northeastern University set up a SHN to construct their IoT testbed (96). It consists of 81 IoT devices with IP connectivity (55 distinct models) located in labs in the US and UK. All the labs are connected via a virtual private network (VPN) to facilitate researchers to experiment with controlled and uncontrolled traffic. It provides labelled traffic traces recorded in PCAP files per device activity (e.g., power on). The testbed encompasses cameras, TVs, audio, hubs, bulbs, plugs, motion sensors and appliances. The dataset was mainly developed for analysing information exposure from IoT devices and is available upon request.

YourThings Data. This labelled dataset encompasses network traces for 63 IoT devices spanning cameras, home assistants, home automation, media, network devices, and appliances, and 4 non-IoT devices (95). Traffic data was collected over 13 days (March 20, 21, 25, 2018, and April 10-19, 2018), with the daily traffic stored across 288 PCAP files, each capturing 5 minutes of data. The total size of the dataset is over 309 GB. It was originally developed for evaluating security.

PingPong. It is a labelled dataset (91) comprising 19 WiFi and Zigbee IoT devices from 16 vendors. The traffic flow was captured during automated interactions, accumulating a total of 100 actions per device. The dataset was developed to identify $\langle Vendor, Type \rangle$ and $\langle Vendor, Type, Activity \rangle$, where it is available upon request. The IoT identification implementation is also accessible to the public.

N-BaIoT. It is a statistical features dataset extracted from 9 IoT devices (cameras, doorbells, a thermostat, and a baby monitor) infected by Mirai and BASHLITE (94). It is composed of 115 features captured from 5-time windows (100 ms, 500 ms, 1.5 sec, 10 sec and 1 min) and obtained from packets originated by having the same source IP, the same source MAC and the same IP address, and have been between the source and destination IPs or TCP/UDP sockets. It considers 8 packet size features (mean/variance) from outbound packets, 4 packet count features, 3 packet jitter features (mean, variance, and number), and 8 packet size (magnitude, radius, covariance, correlation, and coefficient) features from inbound and outbound packets.

KDDIIoT-2019. The dataset consists of a set of Internet Protocol Flow Information Export (IPFIX) records that contain information up to the transport layer (98). The authors captured packets over 108 days from 25 connected IoT devices, both wired and wireless, using Tshark. They saved over 2 million packets in PCAP files, totalling 60,000 MB. These PCAP files were then converted into over 23 million IPFIX records.

IoT IPFIX records. The dataset comprises over 9 million IPFIX records, extracted from 29 IoT devices (97). The network traffic PCAP files were collected in an SHN during

January, March, and April of 2020, and were labelled with MAC and IP addresses. Using the Yet Another Flowmeter (YAF) tool, the PCAP files were converted into IPFIX binary files. Subsequently, the Super Mediator tool was used to transform these binary files into IPFIX JSON files. The IPFIX records include both remote and local flows, with the latter accounting for 70% of the total records.

IoT Network Traffic. This dataset includes features extracted from 16 different IoT devices connected to the internet through a common gateway (136). These include 111 features detailed by Thangavelu *et al.* (1).
It is worth noting that other public IoT datasets, as listed in Table 3.7, have not, to the best of our knowledge, been utilised in the studies reviewed.

IoTDNS Data. It is a labelled dataset that utilises the ground truth of DNS flows for IoT device fingerprinting (13). It spans two months and includes 60 IoT devices and 4 non-IoT devices. It consists of plugs, cameras, home assistants, TVs, streaming boxes, network devices, thermostats, gaming stations, and appliances. The DNS responses are collected over two months and saved in 2 PCAP files with a total size of 366.68 MB.

IoT-23 Dataset. This labelled dataset developed at the Stratosphere Laboratory consists of malicious and benign IoT network traffic (144). The dataset is a collection of 23 PCAP files of network traffic, , of which 20 contain traffic from compromised IoT devices and three from benign devices. The benign traffic traces were captured from 3 IoT devices, including a Somfy smart doorlock, an Amazon Echo, and a Philips HUE smart LED lamp, throughout 1.4, 5, and 24 hours, respectively, producing a total of 427,276 network packers.

LSIF. The dataset includes active traffic traces from 22 distinct IoT device models (145). The traffic of each device was captured for 20 days using Tcpdump and stored separately in a PCAP file.

IoT-deNAT. This dataset includes network traffic recorded over 37 days from home NAT environments within a lab at Ben-Gurion University of the Negev (146). It features data from eight IoT devices and five non-IoT devices, with each traffic flow identified by the device model. The researchers focused solely on idle traffic, which consists of routine messages sent by the devices during periods of inactivity. This consistency and stability in the traffic pattern are expected to have a significant impact on the effectiveness of ML classification.

D-Link IoT. It specifically targets D-Link devices and comprises network packets and MAC frames that were passively gathered in the NSSP laboratory (147). It includes traffic traces from 14 D-Link IoT devices of various types, including, cameras, network cameras, smart plugs, door-window sensors, and home hubs, collected during five months from September 9, 2020, to January 10, 2021.

It is worth noting that a new dataset is anticipated to be released, as stated by Marchal *et al.* (23). The *AuDI* datasets, encompassing both Background and Activity datasets, will cover 33 IoT devices, including IP cameras, plugs, light bulbs, appliances, network devices, and sensors. The Background dataset records packets during the network configuration process and then leaves the devices running without explicit user interaction for at least 24 hours. In contrast, the Activity dataset includes traffic initiated by user interactions and captures the behaviour of 27 single-purpose IoT devices (e.g., ON, OFF, ADJUST). The estimated sizes for the Background and Activity datasets are 226 MB and 239 MB, respectively. This dataset is excluded from Table 3.7 as it is not yet publicly available.

3.5.2 Private Datasets

Several studies introduced private datasets along with public data to evaluate their approaches.

Both Yu *et al.* (18) and Ammar *et al.* (99) leverage the IoT Sentinel dataset in conjunction with their private data to enrich their research on IoT network behaviour. Yu *et al.* (18) utilises their data from 10 IoT and non-IoT devices connected to a TP-Link 2428web gateway via WiFi. They specifically captured DHCP packets using Wireshark to augment their data analysis. On the other hand, Ammar *et al.* (99) capture traffic from six IoT devices in a well-equipped private lab featuring a Broadband Residential Gateway and a Hub, facilitating both Internet and Ethernet connections. A personal computer (PC) monitors and collects the network traffic as IoT devices are connected through an Access Point (AP) or Hub.

Wu *et al.* (126) validate their approach using a combination of their private dataset, collected from 12 IoT devices in an SHN environment (including 9 cameras and 3 speakers, totalling 39.7 GB), and a public dataset provided by the MAWI Working Group (149). The public dataset, which includes background traffic traces from Japan to the US on a 10 Gbps backbone network, is used to simulate IoT traffic in backbone networks. The authors specifically focused on traffic from the 3rd and 10th of June 2020. Given that this dataset does not pertain directly to IoT traffic, it is excluded from the IoT dataset characterisation in our analysis

Liu *et al.* (100) collected traffic from two WiFi IoT devices and combined it with UNSW IoT Traces to enhance the diversity and quality of their testbed. Similarly, the studies by Ma *et al.* (101; 102) also utilises UNSW IoT Traces. Additionally, they captured background traffic at the border of a campus network, which included data from 2,952 unique IP addresses associated with WiFi IoT devices, with a total size of 1.1 TB. Duan *et al.* (110) evaluate their method using UNSW IoT Traces, YourThings dataset, and a private dataset. This private dataset includes traffic collected from 15 IoT

devices over 44 days in an SHN, with device integration facilitated by the XiaoMi Mijia home automation platform and several action trigger rules.

The rest of the studies evaluated their approaches using only private datasets. Meidan *et al.* (5) captured TCP packets from 9 IoT devices, 2 PCs and 2 smartphones via WiFi AP and saved them in PCAP files. The IoT devices in the dataset are typical smart home devices, including a camera, a printer, sensors, smart TV, appliances, and smartwatch.

Jiao *et al.* (90) captured over 20 million packets from 25 IoT devices over four months, from January to April 2019. This data collection was facilitated using Raspberry Pi and Tshark, with a primary focus on WiFi and Ethernet connections (i.e., the presence of ZigBee and Z-Wave IoT devices was accounted for by their associated WiFi hubs). They conducted analyses on 7 IoT devices (such as hubs, cameras, kettles, and sockets) and extracted 26.9 GB of features.

Babun *et al.* (103) focused on idle traffic from Z-based IoT devices and constructed two datasets: one comprising ZigBee devices (17 devices over seven hours) and the other consisting of Z-Wave devices (22 devices over 24 hours). To capture the traffic, they used a Samsung SmartThing hub as the IoT Gateway and employed various sniffer tools, including the KillerBee ZigBee framework, Wireshark, and the Z-Wave 500 Zniffer.

Thangavelu *et al.* (1) configured a SHN using a Raspberry Pi 3 as a gateway to enable internet connectivity for IoT devices through Ethernet or WiFi. This gateway facilitated the capture of traffic, defining a 15-minute interval for each flow. Over seven days, the data collection process successfully recorded 7,584 traffic flows from 16 unique IoT devices, each identified by its MAC address.

Mainuddin *et al.* (92) captured IoT network traffic from 12 IoT devices and 7 non-IoT devices connected to a WiFi router in SHN. The router mounted with Tcpdump and a script to collect traffic for two months. Only Data from one week is selected and used for further analysis or evaluation in their study.

Meng *et al.* (104) collected data from an IoT system in a mining environment, where the IoT devices were connected to the mine LAN. This mine LAN is characterised as a high-speed industrial Ethernet, facilitating communication among 10 IoT devices and 4 non-IoT devices using Ethernet technology. The study focused on extracting packet headers from the network layer, which led to the extraction and labelling of nearly two million headers, identified by their source MAC addresses.

3.6 Open Challenges and Research Questions

This section identifies the main challenges and open questions of the reviewed papers. We concluded that IoT device fingerprinting faces several challenges, specifically, the issue of finding a valid dataset for IoT device fingerprinting (3.6.1). The challenge of standardising IoT device categories (3.6.2). Also, the challenge of finding distinguishing features for each type of identification domain (3.6.3). Moreover, the challenge of developing a model that identifies IoT devices and learns to evolve toward persistent large-scale IoT devices over the years, thus achieving scalability (3.6.4).

3.6.1 IoT Dataset

Traffic traces generated by IoT devices play an essential role in evaluating studies. It requires a real-world IoT dataset collected passively to cover the entire behaviour of devices; in any operational mode. The dataset should contain traffic traces from a wide variety of IoT devices as well as new ones. Each packet should be captured and labelled to be effectively evaluated by ML models.

Most studies utilise public datasets (91%), while a significant portion also includes or relies on private datasets (27%). The primary drawback of using private datasets is that they limit the reproducibility of research findings. Consequently, to mitigate security and privacy concerns associated with publishing real datasets, many are instead generated in controlled lab environments.

As discussed in Section 3.5.1, the availability of IoT datasets to researchers remains limited. IoT Sentinel Dataset gained its popularity among the earliest studies however it includes initial device setup packets. Specialised datasets like IoTDNS focus on DNS traffic, whereas PingPong , LSIF , and IoT-23 target active traffic. In contrast, IoT-deNAT captures exclusively idel traffic. These datasets provide insights into specific IoT interactions. More border datasets like the UNSW IoT Traces and IoT Benign Traces, which include both active and idle traffic, offer a wider view of IoT behaviour, with the former being utilised in over half of the studies reviewed. In fact, Mon(IoT)r is the largest of these datasets, followed by YourThings, providing extensive coverage of IoT interactions.

Despite the size of the datasets, several risks can influence fingerprinting solutions, including being outdated and lacking diversity in IoT devices (Section 3.6.1.1), and the impact of data balance issues (Section 3.6.1.2)

3.6.1.1 Limited IoT Device Diversity

The diversity and quantity of IoT devices within the same model are critical components of IoT datasets. However, a significant concern with public datasets is the limited number of IoT devices, particularly those of identical models, which affects the generalisation of the problem. For example, many researchers achieved high performance in fingerprinting IoT ⟨*Device Instance*⟩, where the dataset includes a single device from different vendors and models. However, the robustness of the approach should be further investigated under various IoT devices sharing the same vendor, model, and functionalities. The issue of the diversity of the dataset is demonstrated by Wang *et al.* (116) where the model achieves high accuracy using UNSW IoT Traces, but not with Mon(IoT)r as it has a TV and three speakers from Amazon (i.e., Fire TV, Echo Dot, Echo Plus, and Echo Spot). These devices are completely different models but have a similar vendor and functionality.

As the IoT market continues to grow, ensuring that the diversity of devices is accurately reflected in real-world scenarios is crucial, especially in real-time applications. This limitation leads to the question of *what are the most effective techniques for developing and regularly updating IoT datasets to reflect the real-world environment?*

3.6.1.2 Imbalanced Datasets

Imbalanced datasets are a common issue in real-world applications, significantly affecting the performance of ML-based solutions. This imbalance complicates the process of detecting devices or instances in the minority classes, often leading the models to be biased towards the majority class. Consequently, many reviewed studies, especially those based on DL, struggle to utilise the entire datasets. In fact, many studies employ various sets of IoT devices to ensure robust fingerprinting solutions, which complicates the process of making fair comparisons among different approaches.

There are several techniques to address imbalanced data and mitigate its impact on model performance. Resampling techniques, such as oversampling the minority classes or downsampling the majority classes, are popular but can lead to biases (overfitting) or the loss of valuable information. Zhang *et al.*(108) introduce an identification method that uses both Synthetic Minority Oversampling Technique (SMOTE) for minority classes and K-means for majority classes. Although the results showed high bias, they generalised well when a new feature was introduced to the model. This outcome prompts the need to explore *how the collection of traffic traces can be optimised to accurately represent diverse device behaviours while addressing dataset imbalances.*

3.6.2 IoT Device Categories

As demonstrated in Section 3.4.4 and Table 3.6, there is a notable variation in the granularity with which IoT device categories are fingerprinted. This variation highlights a lack of standardisation in defining IoT categories, posing challenges for fair comparisons. Consequently, it is essential to explore *how to standardise IoT device categories to facilitate more consistent and comparable research outcomes?*

3.6.3 Network Traffic Features

The lack of standardisation in IoT protocols presents significant challenges for fingerprinting and identifying IoT devices. Effective fingerprinting requires identifying unique features in the network traffic that can identify a specific identification domain. The features without standardisation can vary significantly posing a great complication in determining the most relevant features. Section 3.6.3.1 discusses the complexity of feature extraction strategies to produce a balanced approach that considers the trade-offs between complexity, performance, and cost. Section 3.6.3.2 highlights the effect of the variation of features on fingerprinting different identification domains. Continuing this exploration, Section 3.6.3.3 delves into the robustness of these features. Finally, Section 3.6.3.4 addresses the significant challenges posed by compromised IoT traffic in IoT fingerprinting.

3.6.3.1 Traffic Features Extraction Complexity

Generally, traffic features are correlated with various costs, such as those for acquisition and storage. Packet-based features, which include packet headers, require a higher level of computational resources (141; 1). Consequently, many studies extract features from flows and sessions to reduce overhead. Msadek *et al.* (9) raise concerns that segmenting flow traffic into fixed-length windows can jeopardise the continuity of activity flows, potentially degrading performance. Instead, they recommend allowing traffic transmission to complete using a sliding window approach for traffic extraction, although this method may not be suitable for early identification. Additionally, several studies utilise the first n packets in a connection or flow for real-time device identification (e.g., (2)), which can be more efficient but might overlook significant aspects of IoT device behaviour.

The feature engineering procedure is usually complex, prone to errors, and time consuming (115). Many reviewed studies, such as (1), acquire a large feature vector without considering the associated costs. Indeed, the performance of ML is affected by the feature vector size, where a high-dimensional feature vector increases the computational time and might degrade the model performance. Therefore, feature

selection and reduction techniques are introduced to reduce the feature space. They can improve predictions while reducing costs, by eliminating irrelevant and redundant features. However, Luo *et al.* (128) claim that traditional ML-based methods such as RF and kNN often require specific feature selection algorithms tailored to different scenarios, which can lead to low robustness, as the effectiveness of these models may vary significantly with changes in the data environment.

An interesting method developed by Kotak and Elovici (115) involves transforming encrypted payloads into fixed-size images to serve as features. This technique, which is applied to TCP payloads, is notable for its minimal overhead and has demonstrated high performance in tests. However, one potential drawback is that this approach might result in a loss of semantic information.

Given these challenges, it is crucial to explore *what are the most effective methods for balancing computational efficiency and accuracy in traffic feature extraction for IoT device identification?* Additionally, *what innovative approaches can be implemented to enhance early identification of IoT devices without compromising the continuity and completeness of traffic data analysis?* Addressing these questions will help in developing robust and efficient methods for IoT device behaviour analysis, ensuring both computational efficiency and high accuracy.

3.6.3.2 Identification Domain Behaviour

The network traffic features are used to generate fingerprints to train a classification model for IoT device identification. It is challenging to fingerprint IoT devices efficiently and accurately, especially when there is no standard in IoT protocols. IoT devices from different vendors or different models from the same vendor may use varied communication protocols. This inconsistency leads to different network traffic patterns even for devices performing similar functions.

Reviewed studies demonstrate that fingerprinting IoT devices across various identification domains is challenging, as it is difficult to create a universal model that reliably identifies devices using the same features, as shown in (18; 10; 7). For example, Yu *et al.* (18) report a 100% accuracy using DHCP options for fingerprinting a specific $\langle Vendor \rangle$, which drops to 89% when applied to $\langle Model \rangle$ identification. However, the model's performance improves when utilising IoT Sentinel features. This variability emphasises the importance of thoroughly examining device identification domains and identifying the most relevant distinguishing features for each. Given these challenges, one must ask, *what are the main features that can develop a unique device fingerprint for each device identification domain?*

3.6.3.3 Traffic Features Robustness

As mentioned in Section 3.5, most reviewed studies evaluate the performance using a limited number of IoT devices in a controlled environment dataset. Some studies show the accuracy of identification drops with the increase in the quantity and diversity of IoT devices (e.g., (115; 25)). Consequently, scalability becomes an issue, which is discussed further in Section 3.6.4.

Adding to this challenge, the ability of header field features to differentiate between devices reduces as the IoT dataset becomes diverse. This decline is greatly impacted by the non-standardisation of IoT protocols which affects the reliability of device fingerprinting in extensive, heterogeneous networks. The literature has proposed a broader range of traffic features for fingerprinting as shown in Table A.1 - A.5. Studies such as (10; 109) proposed a large set of features where some of them have a very low contribution to fingerprinting. We observe that certain features, such as IP addresses, ports, DNS, and DHCP, are highly specific to an identification domain and can significantly enhance the accuracy of device fingerprinting, as demonstrated in the study by Hammainen (123). However, these features do not directly reflect the device's behaviour and are vulnerable to attacks. For instance, domain names can be easily changed, for various business reasons by vendors, which potentially compromises the reliability of the fingerprinting model. Hence, it is better to adopt liable features to fulfil the aim of fingerprinting for long-term identification.

Focusing on features that describe the behaviour of IoT devices, such as packet size, TTL, and TCP window time, can provide great discrimination. IAT-related features are particularly crucial; some studies, such as (103), rely exclusively on these for fingerprinting. However, factors like hardware setup and Internet connectivity can significantly influence these features (108). Additionally, periodicity features, which are extracted from the background traffic without triggering the devices, vary depending on the devices' configurations. According to Wu *et al.* (126) and Marchal *et al.* (23), these features require complex extraction procedures.

Considering these observations, traffic traces must be deeply analysed to identify distinguishing features specific to an IoT device identification domain. While a feature vector may prove distinctive within one domain, it may not necessarily hold the same significance across other domains. This variability necessitates a careful evaluation of the importance of each feature, highlighting the need to explore the most effective methods for balancing computational efficiency and accuracy in traffic feature extraction for IoT device identification, as discussed in Section 3.6.3.1. This aspect is critical as it directly impacts the performance of IoT device fingerprinting by addressing the potential trade-offs (i.e., cost, speed, and accuracy). Dimensional reduction methods have shown substantial improvements in fingerprinting by reducing overhead and enhancing performance. Additionally, identifying the main

features that can develop a unique device fingerprint for each device identification domain is essential for ensuring robust and adaptable IoT systems, as discussed in Section 3.6.3.2. These considerations emphasise the importance of meticulously selecting and optimising features that not only cater to domain-specific requirements but also maintain a balance between computational demands and identification precision.

3.6.3.4 Compromised IoT Traffic

Most of the examined studies use benign traffic. This raises the question of how incorporating and analysing features from abnormal or malicious traffic could enhance IoT security. Specifically, *to what extent do dataset traffic traces with attacks impact the development of an IoT device identification scheme?*

3.6.4 Scalability

The scalability of IoT device fingerprinting is associated with the traffic features and ML technique. As mentioned earlier in the previous section, the need to address feature robustness to remain effective in providing consistent predictive performance across different identification domains. This section will focus on the ML methods and their implication in fingerprinting scalability.

An important trend in the IoT field is the staggering and relentless increase of active IoT devices (24). When an ML-based IoT fingerprinting mechanism is used, the classifier should be retrained often to ensure new devices can be recognised accurately. However, this aspect is largely neglected in literature. In fact, most of the IoT fingerprinting approaches proposed in literature are designed and evaluated to handle a fixed number of devices, without assessing their behaviour when new devices are introduced (25), i.e., without assessing their scalability. A naive approach would be to retain all the available training data, augment it whenever training data for new IoT devices is available and, finally, retrain the classifier from scratch. Although sound in principle, this approach is impractical because it requires a large, growing amount of training data; also, training a model from scratch on all the training data can be very time consuming.

A few approaches for scalable IoT fingerprinting have been proposed in literature. Some use a binary classifier for each device and introduce further binary classifiers as training data for new IoT devices becomes available. This approach requires an ever growing number of classifiers, which might lead to memory exhaustion. Also, additional mechanisms are needed to break ties when more classifiers return a positive outcome for the same device. Furthermore, the accuracy of this approach

tends to degrade over time because existing classifiers are never retrained to ensure they can properly distinguish the devices introduced latest. Other approaches are based on CIL, where the classifier is retrained using both data for new IoT devices and a selection of data for known devices. As time goes by, the selection of data for known devices needs to be capped in size for memory/disk constraints, which implies fewer samples for each device can be retained and, therefore, the classifier accuracy tends to worsen for older devices; this phenomenon is commonly referred to as catastrophic forgetting.

The scalability of IoT device fingerprinting and its integration with traffic features and machine learning techniques presents significant challenges and opportunities in enhancing system autonomy without sacrificing precision. As we contemplate advancements in this field, two crucial questions emerge that must be addressed to further our understanding and development of effective IoT identification systems. First, *in what ways do traffic features and ML improve the scalability and autonomy of IoT device identification compared with other scalable approaches?* This question prompts a deeper examination of how machine learning algorithms can be optimised to handle increasing numbers of IoT devices without extensive retraining or memory overhead. Additionally, the balance between system performance and operational cost raises another vital inquiry: *to what extent should we consider trade-offs between performance and memory consumption?* Addressing this will help in crafting strategies that not only ensure efficient and effective fingerprinting but also align with the practical constraints of memory and processing resources

3.7 Summary

This chapter provided a comprehensive review of passive IoT device fingerprinting, emphasising its dependence on machine learning and network traffic analysis. To the best of our knowledge, this is the first academic review exclusively focused on this topic. The research questions we address in this Chapter concern the approaches proposed in the literature for passive IoT device fingerprinting and the main challenges and limitations these approaches face. We introduce a detailed taxonomy of ML-based approaches for device identification and offer an in-depth study of network traffic features essential for generating robust device fingerprints. Our discussion establishes a baseline reference for IoT traffic features utilised in current literature, marking a significant contribution to the field. We also identify key open research problems and suggest directions for future research, including enhancing scalability, improving robustness, and addressing the dynamic nature of IoT environments.

Addressing scalability is essential in IoT device fingerprinting, as it must manage resource requirements while maintaining high performance. This research will focus

on addressing scalability to develop a more efficient approach to IoT device fingerprinting that dynamically adapts to the growing diversity of IoT devices without compromising performance.

Chapter 4

ScaNeF-IoT: Scalable Network Fingerprinting for IoT Devices

4.1 Introduction

Scalability is a critical challenge in IoT fingerprinting due to the rapid growth of connected devices and the continuous generation of diverse. Traditional ML techniques, such as binary classifiers and DL often require extensive datasets and retraining, which becomes computationally expensive as the system grows. The transfer learning and CIL mitigate this issue, where the classifier is retrained using both data for new IoT devices and a selection of data for known devices (knowledge reply). As time goes by, the selection of data for known devices needs to be capped in size for memorydisk constraints, which implies fewer samples for each device can be retained and, therefore, the classifier accuracy tends to worsen for older devices; this phenomenon is commonly referred to as catastrophic forgetting.

In IoT fingerprinting, *Online Stream Learning* (OSL) techniques (150; 65) offer a more scalable solution by learning incrementally from real-time data streams. Unlike DL or CIL, OSL eliminates the need to retrain the entire model on large static datasets or knowledge replies. Instead, it updates the model incrementally as new data arrives, making it highly suitable for dynamic environments where device behaviours change frequently. This approach depends on the availability of labelled data for updates, ensuring scalability while adapting to evolving patterns with minimal computational overhead.

This Chapter aims to address the limitations of existing approaches for scalable IoT fingerprinting, with a specific focus on identifying new IoT devices and updating the model accordingly without extensive retraining or storage demands for old samples which consumes memory. We propose to make use of OSL, which allows for adaptive

model updates while efficiently managing computational resources. This improves scalability as it makes it easier to fingerprint samples from new IoT devices. An OSL classifier can be updated on the fly by providing new training data; previous training data does not need to be retained and replayed, which eliminates storage requirements. The internal model of an OSL classifier is automatically extended to adapt to the new training samples, and each training sample only needs to be processed once to be learned.

While OSL-based IoT fingerprinting inherently supports scalability, its practicality depends on the accuracy it can provide in recognising IoT devices. Our focus is, thus, on assessing the OSL classifier accuracy. Therefore, the research question we explore in this work is:

> **RQ3.** *How can we develop a scalable IoT fingerprinting approach to improve device recognition accuracy and memory consumption?*

This Chapter introduces *ScaNeF-IoT*, a novel approach for scalable network IoT fingerprinting. Besides relying on an OSL-based classifier to achieve scalability, it uses *fixed-size traffic session payloads* as features to feed the classifier. The reason behind this choice is that the literature shows that the packet-based features can be highly variable and may introduce noise (2), while statistical features often require significant feature engineering. These features, such as packet count, average inter-arrival time, or byte distribution, need to be carefully designed and extracted to capture the most relevant patterns in network traffic. The challenge lies in selecting the right statistical metrics that represent device behaviour accurately, which can be both time-consuming and error-prone. Furthermore, as IoT environments evolve, previously engineered features may lose relevance, requiring constant updates to maintain accuracy (see Chapter 3). In contrast, using fixed-size traffic session payloads bypasses much of this complexity. It can avoid the need for intricate feature engineering while still capturing key characteristics of the traffic without manual feature extraction (115). Moreover, it allows to cap the time required to fingerprint a session, which, on average, enables faster detection of unusual network activities.

We implement ScaNeF-IoT using the Adaptive Mondrian Forest (AMF) (151) as OSL algorithm and focusing on UDP/TCP sessions. We evaluate our implementation in terms of IoT device recognition accuracy using the UNSW IoT Traces dataset (12) and comparing the results with two state of the art scalable IoT fingerprinting approaches, namely AutoIoT (25) and IoT-Portrait (26). The authors of these works evaluated their approaches using the same dataset; to enable a fair comparison, our experiments employ the same evaluation strategy as theirs. The former is compared with over the whole dataset, with ScaNeF-IoT showing comparable device recognition accuracy. The comparison with the latter, instead, assesses the accuracy across a number of

retrainings, demonstrating that ScaNeF-IoT performs better than IoT-Portrait most of the time.

The novelty of this Chapter lies in the fact that ScaNeF-IoT is the first approach proposed for scalable IoT fingerprinting that is based on OSL. The main contributions of this work are:

- a novel approach for scalable IoT fingerprinting, based on OSL and fixed-size session payloads;
- an implementation of the approach, based on AMF and UDP/TCP payloads;
- an experimental evaluation based on the UNSW IoT Traces dataset (12), with an accuracy comparison with two other scalable IoT fingerprinting approaches, AutoIoT (25) and IoT-Portrait (26), showing comparable results with the former and outperforming the latter.

The rest of this Chapter is organised as follows. Section 4.2 presents the ScaNeF-IoT approach. Section 4.3 presents the experimental evaluation. Finally, a summary of this chapter is in Section 4.4.

4.2 ScaNeF-IoT approach and implementation

This section introduces the ScaNeF-IoT approach for scalable network fingerprinting of IoT devices. Figure 4.1 highlights the key stages of the proposed approach. Network traffic generated by IoT devices is first preprocessed to (i) identify sessions, (ii) extract their payload and (iii) produce fixed-size payloads. Section 4.2.1 details the operations involved in this step. The resulting payloads are used as feature vectors to feed an OSL classifier, which outputs the identifier of the corresponding IoT device. More details on this step and how it enables scalability are discussed in Section 4.2.2. The implementation of the ScaNeF-IoT approach is described in Section 4.2.3.

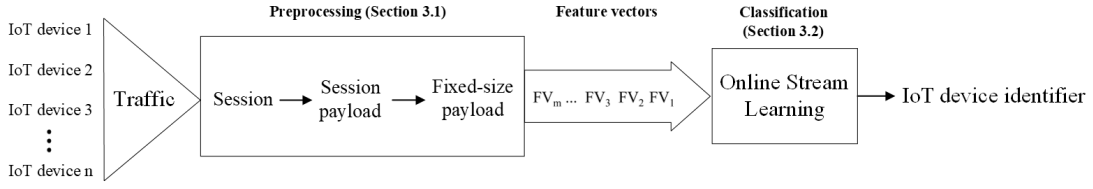


FIGURE 4.1: ScaNeF-IoT approach.

4.2.1 Preprocessing

In this work, we adopt an approach similar to Kotak and Elovici (115), where images are generated from session payloads. We use packet headers to identify bi-directional sessions. The payloads of a session are concatenated and either trimmed or padded to a fixed size to generate the feature vector corresponding to that session. Rather than creating an image, we produce a fixed-size payload ready to be fed to the chosen classifier.

Using fixed-size payloads allows to generate feature vectors without waiting for the end of longer sessions, which, on average, can permit to classify a session earlier. This better fits situations where it is fundamental to take security countermeasures promptly in case anomalies are detected. While, in this way, a possibly large part of the payload might not be considered for the classification, other works in the literature have shown that considering only the initial segment of a network communication is sufficient to accurately fingerprint IoT devices (115).

The preprocessing stage consists of three steps, as shown in Figure 4.2.

1. **Session identification**. By network session, we refer to the traffic exchanged between two different endpoints using the same protocol. For example, in TCP or UDP, a session is uniquely identified by the following 5 values included in each packet header: source IP address, source port number, destination IP address, destination port number and protocol (TCP or UDP).
2. **Session payload extraction**. The payloads of the packets within a same session are extracted and concatenated in a single, session-specific buffer in hexadecimal format. Empty payload sessions are ignored.
3. **Fixed-size payload generation**. Once the buffer reaches a size equal to or greater than a prefixed value S, the payload is trimmed to S and becomes ready for classification. In case the session ends before the buffer reaches a size of S, then it is padded with 0x00 bytes. At this stage, we do not define an explicit timeout for sessions.

A fixed-size payload extracted from a session initiated by an IoT device x (i.e., with x as source of the first packet of the session) is used to train the classifier model to recognise x. Note that payloads might be encrypted.

4.2.2 Classification

The classification stage relies on an Online stream learning (OSL) algorithm to achieve scalability. Indeed, OSL techniques allow for incremental updates of the model as new

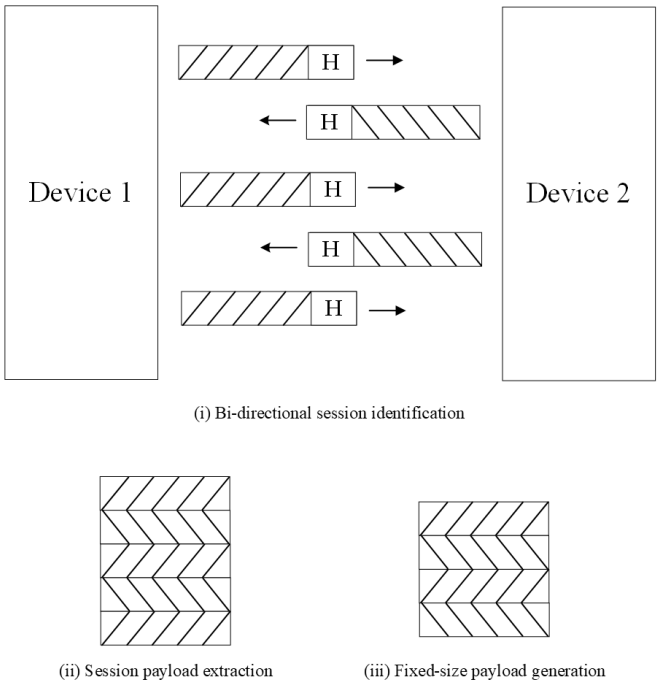


FIGURE 4.2: Preprocessing steps.

training data is available, without the need for retraining the model from scratch every time (65) and, therefore, for storing training data for later retraining. This approach also helps address the problem of concept drift, where the distribution of the data changes over time and deviates from the one learned by the ML model (65). Indeed, these algorithms typically employ adaptive mechanisms to adjust to changes in the data distribution over time, making them suitable for dynamic environments such as IoT networks.

In this context, ensemble-based OSL methods, such as Online Random Forest, can offer additional advantages. Ensemble predictors combine multiple models to form a collective decision, which tends to improve accuracy over individual models by aggregating their strengths (152). This is especially effective in streaming scenarios, where ensembles can scale easily, support distributed processing, and adapt to concept drift by pruning underperforming models and incorporating new ones (152). This adaptability enhances the classifier's robustness and responsiveness to evolving IoT traffic patterns.

When up-to-date labelled data becomes available for new IoT devices or recent behaviours of known devices, it can be used to incrementally update the ensemble-based OSL model. This enables the recognition of new devices or behaviours without relying on previous training data, setting ScaNeF-IoT apart from traditional IoT fingerprinting scalability approaches, which often require retaining the entire dataset or subsets for retraining. By leveraging the adaptive properties of

ensemble-based OSL, ScaNeF-IoT can scale seamlessly to recognise new IoT devices as soon as the corresponding training data is available.

4.2.3 Implementation

To run the experimental evaluation detailed in Section 4.3, we have implemented the ScaNeF-IoT approach as described below.

4.2.3.1 Preprocessing Implementation

We identify UDP and TCP sessions using the 'session' option in the SplitCap (153) tool, which allows splitting network traffic in different sessions. We tested different values for the buffer size S to assess how it affects the accuracy of the classifier (115).

4.2.3.2 Classification Implementation

Random Forest, a popular ensemble method, is particularly effective for classification tasks in IoT environments due to its ability to enhance accuracy and manage complex data patterns without overfitting (154). This robustness makes it ideal for handling the varied and dynamic nature of IoT data. To leverage these strengths, we selected an ensemble-based OSL approach for our classification stage, ensuring both adaptability and scalability. Ensemble methods, like *Adaptive Random Forest* (ARF) (154) and *Aggregated Mondrian Forest* (AMF) (151), are well-suited for OSL because they aggregate predictions from multiple decision trees, which enhances accuracy while enabling the model to adapt efficiently to evolving data. Given these advantages, we selected ARF and AMF for our implementation to leverage their ability to incrementally update individual trees with new data, allowing for seamless adaptation in dynamic environments.

Adaptive Random Forest. ARF extends the traditional RF to handle evolving data streams through an adaptive strategy for concept drift detection and adaptation (154). ARF uses Hoeffding trees (155) as base learners and integrates a drift detector (e.g., ADWIN) to monitor changes in data distribution. When a drift is detected, ARF replaces outdated trees, ensuring the model remains relevant to recent data. Additionally, ARF employs an online bagging approach based on Oza's resampling method (156), introducing diversity across trees. Each tree has a dedicated drift detector, which triggers the training of new background trees in response to early warnings, allowing seamless transitions and reducing model disruption when concept drift is confirmed. This approach allows ARF to retain both historical and new data

patterns, making it highly effective for dynamic environments where data distributions are continuously shifting.

The configuration of an ARF in the river library (157) offers a range of customisable parameters that enhance its flexibility and adaptability in handling evolving data streams. At its core, ARF constructs an ensemble of decision trees (default: 10 trees) where each tree can be tuned by adjusting parameters such as the maximum features considered for each split, determined by options like ”sqrt” or ”log2.” To handle the dynamic nature of IoT data, ARF integrates drift detection and warning mechanisms (commonly ADWIN), which monitor and adapt to concept drift by resetting underperforming trees. Tree-specific settings, such as maximum depth, split criterion (e.g., information gain or Gini), and leaf prediction method (Naive Bayes or majority class), allow ARF to balance computational efficiency and predictive accuracy. Additional parameters manage memory usage and prevent overfitting by enforcing conditions like minimum branch fraction and pre-pruning.

Aggregated Mondrian Forests. AMF is an online learning algorithm for random forests that can efficiently update the model as new labelled data arrives sequentially in a streaming setting (151). It builds upon the Mondrian Forest (MF) methodology introduced by Lakshminarayanan *et al.* (158) and incorporates principles from the Infinite Mondrian Process (IMP) proposed by Mourtada *et al.* (151) to deliver AMF. When AMF classifies a sample, it utilises an ensemble of DT generated through the MP. Each DT in the ensemble provides a prediction for the sample based on its features. These individual predictions are then aggregated to produce a final prediction for the sample. During training, When a new sample is presented for training, AMF updates the structure of its DTs to incorporate information from the sample. This may involve splitting nodes, creating new leaf nodes, or adjusting existing nodes based on the sample’s features. After updating the DT structure, AMF computes a prediction for the sample using each pruned tree. These individual predictions are then aggregated using a variant of the context tree weighting (CTW) algorithm. This aggregation process assigns weights to each prediction based on its reliability, considering factors such as the tree’s performance on past samples. The final prediction for the sample is obtained by computing a weighted average of all the predictions from the pruned trees. By updating its DT structure and prediction function in an online streaming setting, AMF adapts and learns from the data incrementally, without requiring full retraining from scratch. Consequently, enabling it to make accurate predictions even in the presence of changing or evolving patterns in the data.

The configuration of an AMF in the river library (157) involves several key parameters that allow customisation to specific datasets and learning requirements. The *n_estimators* parameter, with a default value of 10, specifies the number of trees in the forest, impacting both accuracy and computational demands. The step parameter, set

by default to 1.0, controls the step size for aggregation weights, optimising for classification tasks using log-loss. The *use_aggregation* boolean, recommended to be set to True, enables the use of aggregation in trees to enhance model robustness across varied data samples. Additionally, the *dirichlet* parameter, serves as a regularisation factor for class frequencies in node predictions, helping to mitigate overfitting in scenarios with numerous classes. It is typically set to the reciprocal of the expected number of classes $\frac{1}{n\_classes}$. The default value for this parameter is set at 0.5, which is optimal for binary classification tasks. Moreover, the *split_pure* parameter, which is False by default, determines whether nodes containing only samples from a single class should be split further, potentially aiding in generalisation by analysing more features, even in homogeneous data segments. Lastly, the *seed* parameter, for reproducibility.

4.3 Evaluation

This section describes how we evaluate ScaNeF-IoT to assess its accuracy in fingerprinting IoT devices, as well as its scalability as the classifier model learns new IoT devices.

Dataset. We use the UNSW IoT Traces dataset (12), which consists of network traffic traces captured in a network with 23 IoT devices and 7 non-IoT devices (e.g., smartphones, tablets, and laptops). The dataset is a collection of 20 pcap files of approximately 9.5 GB total size, spanning over 20 days. It includes 148,788 sessions (97,945 for IoT devices, 50,843 for non-IoT devices). In this research, we only focus on IoT device sessions. Table 4.1 provides further details on the IoT devices in the dataset and how many TCP and UDP sessions are available for each.

Evaluation approach. Our experiments aim to assess ScaNeF-IoT accuracy in fingerprinting IoT devices through comparisons with relevant related works. We first evaluate the ScaNeF-IoT accuracy using two ensemble-based OSL methods: ARF and AMF over the whole dataset for different values of the buffer size S, and compare it with the performance AutoIoT is reported to have on the very same dataset (25) (see Section 4.3.1). Then, we analyse how the ScaNeF-IoT accuracy varies as we incrementally train the classifier model with new IoT devices (see Section 4.3.2); in this experiment, the comparison is made with IoT-Portrait (26), which employs an incremental learning approach and provides an experimental evaluation based on the same dataset we use.

To enable a fair comparison with alternative approaches proposed by other researchers, we employ of the same accuracy metrics they use in their experiments. In particular, we consider the overall *accuracy* of a classifier, defined in Equation 2.1. To assess the accuracy of the classifier in fingerprinting a specific IoT device x, we use the

TABLE 4.1: The Payload Sessions Extracted from the UNSW IoT Traces Dataset

#	IoT Device Name	Sessions	TCP Sessions	UDP Sessions
1	Amazon Echo	3,491	3,407	84
2	Belkin Wemo Motion-Sensor	48,883	48,786	97
3	Belkin Wemo Switch	8,939	7,032	1,907
4	Blipcare Blood Pressure Meter	4	4	0
5	Dropcam	35	35	0
6	HP Printer	150	150	0
7	iHome Power Plug	153	153	0
8	Insteon Camera wired	8,702	4,055	4,647
9	Insteon Camera wireless	102	1	101
10	Light Bulbs LiFX Smart Bulb	52	34	18
11	Nest Dropcam	29	29	0
12	NEST Protect Smoke Alarm	84	84	0
13	Netatmo Weather Station	2,338	2,338	0
14	Netatmo Welcome	2,728	2,688	0
15	PIX-STAR Photo Frame	1,118	1,118	0
16	Samsung SmartCam	10053	9,082	971
17	Samsung Smart Things	24	24	0
18	TP-Link Day Night Cloud Camera	1,541	1,109	432
19	TP-Link Smart Plug	239	232	7
20	Triby Speaker	131	129	2
21	Withings Aura Smart Sleep Sensor	3,584	3,584	0
22	Withings Smart Baby Monitor	5,545	5,545	0
23	Withings Smart Scale	20	20	0

F1 Score defined in Equation 2.6. Furthermore, we ensure using the same set of devices and the same validation strategy they employ.

4.3.1 Accuracy Evaluation

In this experiment, we assess how the accuracy of ScaNeF-IoT (evaluated with both ARF and AMF), as defined in Equation 2.1, is affected by the buffer size. We test three different buffer sizes: 265, 784 and 1,024 bytes. These buffer sizes correspond to the image dimensions used in converting the payloads: 16x16 (265 bytes), 28x28 (784 bytes), and 32x32 (1,024 bytes), which are compatible with standard input sizes frequently used in image processing tasks. The 16x16 size is optimal for low-resolution data, as it efficiently fits small datasets and benefits from being a power of 2, which optimizes computational performance. The 28x28 size is widely recognised in tasks like like the MNIST dataset (159), striking a balance between resolution and computational cost and used in IoT fingerprinting as proposed by Kotak and Elovici (115). The 32x32 size, commonly used in higher-resolution tasks such as CIFAR-10 (160), allows for capturing more data detail while maintaining computational efficiency, also benefiting from its power of 2 structure. Additionally, we make a comparison with the classification performances reported by Fan *et al.* for AutoIoT (25), which is described in Section 3.4.3.3. Like them, we use all 23 IoT

devices included in the UNSW IoT Traces dataset and adopt a 30% holdout validation strategy, splitting the dataset into training and test sets in a 7:3 ratio. We consider a sequential split according to the timestamp by preserving the last 30% of each IoT device payload in the test set.

Figure 4.3 shows the results of this experiment. The accuracy comparison between AutoIoT and ScaNeF-IoT (evaluated with both ARF and AMF) shows interesting insights into their performance on IoT device identification tasks. AutoIoT achieves a very high accuracy of 99.8%, which serves as the baseline for this comparison. ScaNeF-IoT, utilising ensemble-based OSL methods, performs slightly lower than AutoIoT, but still demonstrates strong accuracy, particularly with AMF.

The results indicate that AMF consistently outperforms ARF across all buffer sizes tested (265, 784, and 1024 bytes), with AMF reaching an accuracy of up to 99.29% at 784 bytes, which is the closest to AutoIoT's accuracy. ARF, while slightly less accurate than AMF, still performs reasonably well, achieving accuracies between 96.94% and 97.91%.

Interestingly, AMF maintains a stable accuracy across different buffer sizes (99.29% with 784 bytes against 99.18% with 265 and 99.24% with 1024), suggesting it may be more robust in handling variations in data granularity. In contrast, ARF's accuracy declines slightly as the buffer size increases (97.91% with 265 bytes, 97.44% with 784, and 96.94% with 1024), which may indicate that it is more sensitive to changes in data volume or complexity.

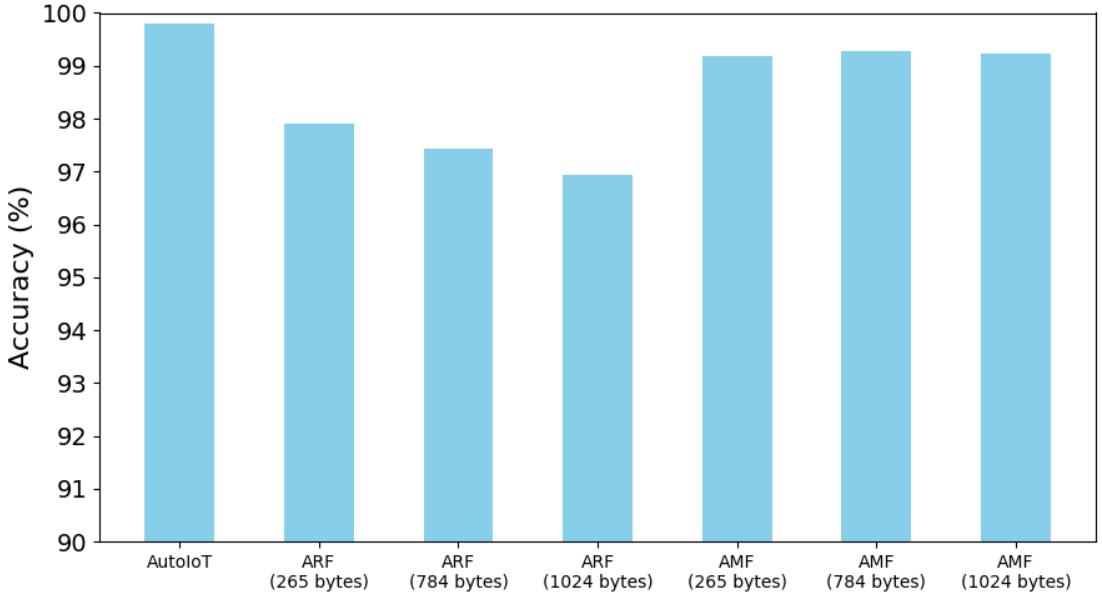


FIGURE 4.3: Accuracy comparison between AutoIoT and ScaNeF-IoT (ARF and AMF) with different buffer sizes, based on a 30% holdout validation strategy.

However, since the difference in accuracy is very small between ScaNeF-IoT with AMF and AutoIoT, we look at the F1 score of each device, as defined in Equation 2.6,

to analyse the extent to which different buffer sizes affect the fingerprinting of individual devices. For each buffer size, we assess how many devices are fingerprinted with the highest F1 score by ScaNeF-IoT when configured with that buffer size. We distinguish between cases where the F1 score is strictly higher than with the other two buffer sizes, and cases where it is equal to the F1 score obtained with any or both of the other two buffer sizes. The results reported in Figure 4.4 show that ScaNeF-IoT configured with a buffer size of 784 bytes provides the highest F1 score for 19 devices out of 23 (strictly higher than the others for 4 devices, equal to the others for 15 devices). Instead, ScaNeF-IoT configured with a buffer size of 1024 bytes achieves the highest F1 score in 17 cases (strictly higher in 2 cases, equal in 15 cases), while using 265 bytes as buffer size leads to the highest F1 score for 14 devices (for 1 strictly higher, for 12 as high as the others). Therefore, for the following experiments, we configure ScaNeF-IoT with a buffer size of 784 bytes.

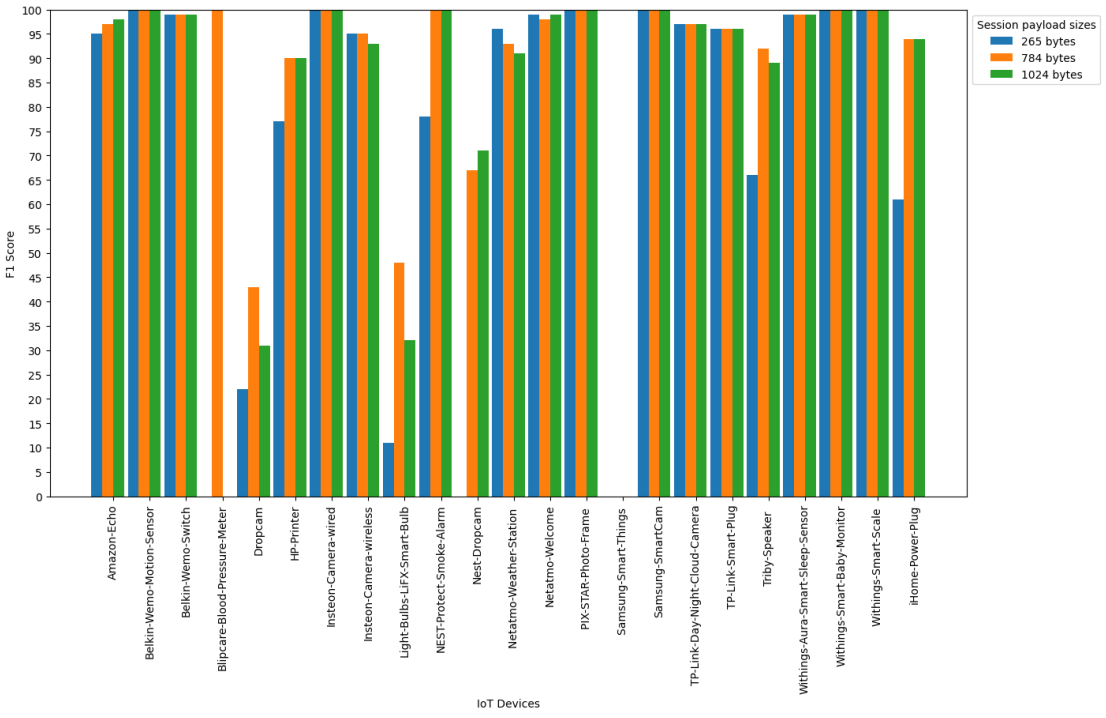


FIGURE 4.4: F1 score comparison for each IoT device using ScaNeF-IoT with AMF and different buffer sizes.

Although the F1 score is at least 90% for the large majority of IoT devices, ScaNeF-IoT exhibits a less than satisfactory performance for a few devices. Table 4.2 details precision, recall and F1 score obtained for each device when using ScaNeF-IoT configured with a buffer size of 784 bytes. Also, the table reports the support for each device, which is the number of feature vectors used for the testing. The dataset is heavily imbalanced, as evidenced by the widely varying support values across devices. This imbalance affects performance, leading to lower F1 scores for devices with less support. As shown in the support column of Table 4.2, devices with a

smaller number of feature vectors have fewer examples available for testing (30% of the data), which contributes to lower precision, recall, and F1 scores.

TABLE 4.2: Precision, recall, F1 score and support for each IoT device using ScaNeF-IoT with 784 bytes buffer size.

#	IoT Device Name	Precision	Recall	F1	Support
1	Amazon-Echo	0.99	0.96	0.97	1047
2	Belkin-Wemo-Motion-Sensor	1.00	1.00	1.00	14665
3	Belkin-Wemo-Switch	1.00	0.99	0.99	2682
4	Blipcare-Blood-Pressure-Meter	1.00	1.00	1.00	1
5	Dropcam	1.00	0.27	0.43	11
6	HP-Printer	1.00	0.82	0.90	45
7	Insteon-Camera-wired	1.00	1.00	1.00	2611
8	Insteon-Camera-wireless	1.00	0.90	0.95	31
9	Light-Bulbs-LiFX-Smart-Bulb	1.00	0.31	0.48	16
10	NEST-Protect-Smoke-Alarm	1.00	1.00	1.00	25
11	Nest-Dropcam	0.83	0.56	0.67	9
12	Netatmo-Weather-Station	0.87	1.00	0.93	701
13	Netatmo-Welcome	0.99	0.98	0.98	818
14	PIX-STAR-Photo-Frame	1.00	1.00	1.00	335
15	Samsung-Smart-Things	0.00	0.00	0.00	7
16	Samsung-SmartCam	1.00	1.00	1.00	3016
17	TP-Link-Day-Night-Cloud-Camera	0.98	0.97	0.97	462
18	TP-Link-Smart-Plug	1.00	0.93	0.96	72
19	Triby-Speaker	1.00	0.85	0.92	39
20	Withings-Aura-Smart-Sleep-Sensor	0.99	0.99	0.99	1075
21	Withings-Smart-Baby-Monitor	0.99	1.00	1.00	1664
22	Withings-Smart-Scale	1.00	1.00	1.00	6
23	iHome-Power-Plug	1.00	0.89	0.94	46

4.3.2 Scalability Evaluation

This experiment aims to assess how ScaNeF-IoT fingerprinting accuracy varies as the classifier model is updated to recognise new devices. We compare ScaNeF-IoTns's performance with a state of the art scalable IoT fingerprinting approach, namely IoT-Portrait (26), which is described in Section 3.4.5.

To ensure fairness in the comparison, we scale up the model in the same way IoT-Portrait does. As explained in Section 3.4.5, IoT-Portrait employs CIL and, therefore, arranges its classification activities over time across a number of sequential tasks. At the beginning of each task, a set of new IoT devices is introduced and the classifier model is retrained to recognise these new devices as well as all the IoT devices introduced in the previous tasks; in the first task (i.e., task 0), the classifier model is trained to fingerprint an initial set of IoT devices.

IoT-Portrait is evaluated using the UNSW IoT Traces dataset with 3 IoT devices in task 0 and 2 new IoT devices in each of the following 6 tasks. Table 4.3 details which IoT devices are new in each task. To assess ScaNeF-IoTns, we update the classifier model

TABLE 4.3: Fingerprinting accuracy of ScaNeF-IoT as the classifier model is scaled up to recognise new IoT devices.

#	Device Name	Task	Accuracy
1	Samsung-SmartCam	0	99.97
2	Belkin-Wemo-Motion-Sensor		
3	Withings-Smart-Baby-Monitor		
4	Belkin-Wemo-Switch	1	96.11
5	Amazon-Echo		
6	Netatmo-Welcome	2	94.33
7	Netatmo-Weather-Station		
8	TP-Link-Day-Night-Cloud-Camera	3	71.15
9	PIX-STAR-Photo-Frame		
10	TP-Link-Smart-Plug	4	94.14
11	HP-Printer		
12	Triby-Speaker	5	93.47
13	Dropcam		
14	Samsung-Smart-Things	6	94.74
15	Withings-Smart-Scale		

using the same sets of IoT devices that Wang *et al.* used to evaluate IoT-Portrait. Some of these IoT devices are under-represented in the dataset, which can lead to poor accuracy, as discussed in Section 4.3.1. Also, we adopt a 6:4 ratio for training and testing since the same is used to assess IoT-Portrait. The test set for each task includes payloads from new devices introduced and payloads from older devices presented in previous tasks. The accuracy is calculated for each task as defined in Equation 2.1, and is reported in Table 4.3.

Figure 4.5 shows the comparison between IoT-Portrait and ScaNeF-IoT in the accuracy they provide across all 7 tasks. ScaNeF-IoT performs better than or equally to IoT-Portrait in all tasks but task 3, where the accuracy drops to 71%.

This outlier is mostly caused by the misclassification of a large proportion of Belkin-Wemo-Motion-Sensor samples as TP-Link-Day-Night-Cloud-Camera, as reported in the task 3 confusion matrix in Figure 4.6. The F1 score for these two devices in task 3 is 73% and 12%, respectively.

In the following task, once the classifier model has been updated to recognise the two new IoT devices, the overall accuracy goes back to 94%. Also, the misclassifications of Belkin-Wemo-Motion-Sensor and TP-Link-Day-Night-Cloud-Camera samples are reduced significantly, as shown in Figure 4.7, with F1 score for these two devices equal to 98% and 73%, respectively.

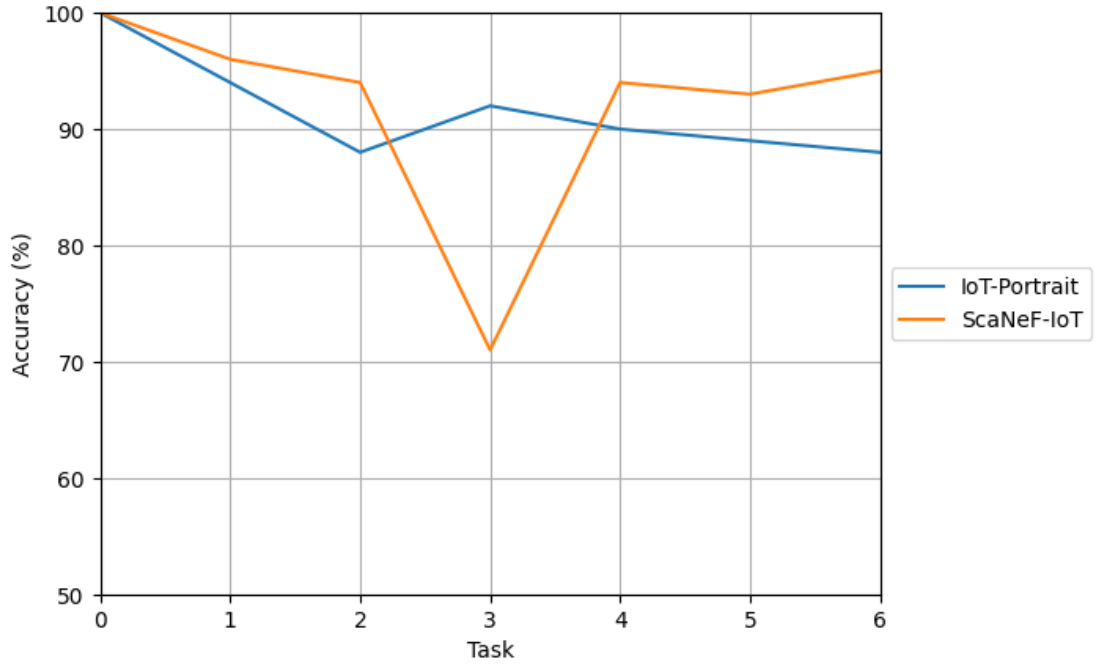


FIGURE 4.5: Accuracy comparison between ScaNeF-IoT and IoT-Portrait as new IoT devices are introduced.

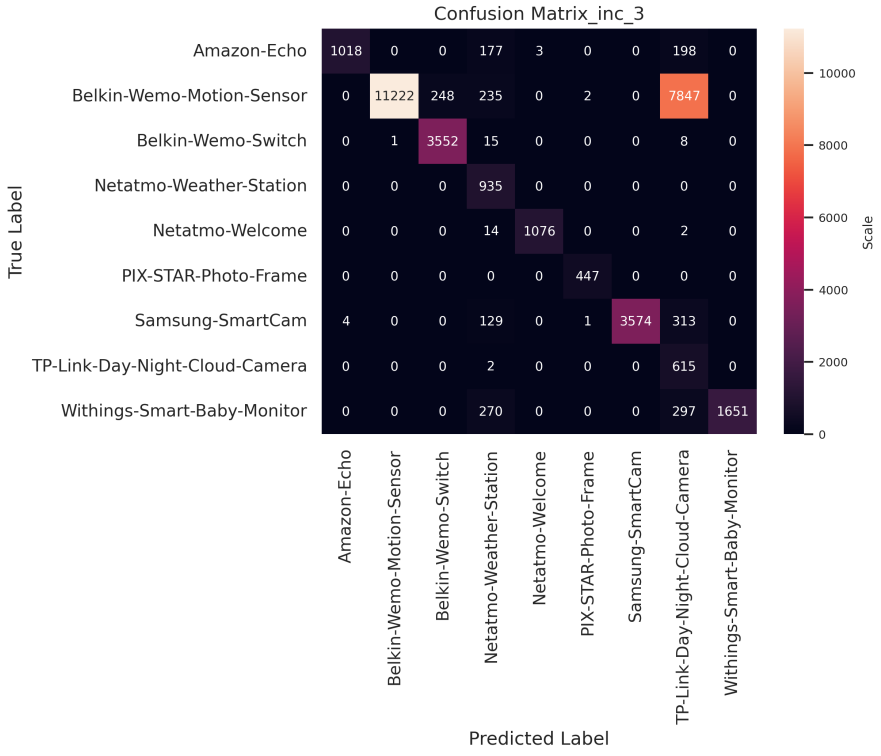


FIGURE 4.6: Task 3 confusion matrix

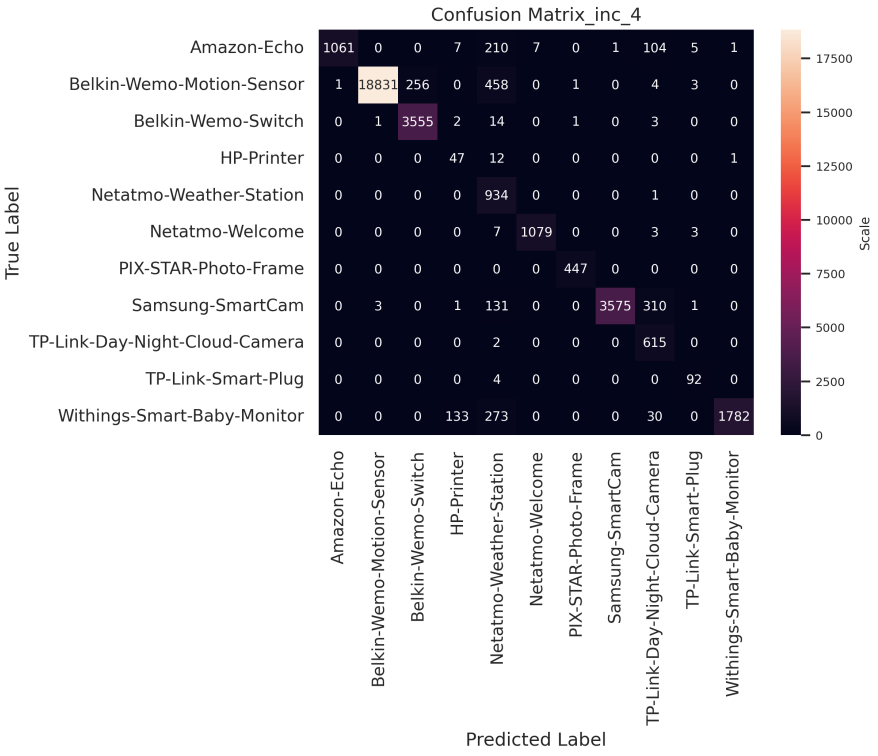


FIGURE 4.7: Task 4 confusion matrix

4.4 Summary

Scalable approaches to IoT fingerprinting are required to cope with the fast pace at which new IoT devices are developed. The novel approach we propose, ScaNeF-IoT, achieves scalability by using an OSL classifier that does not require storage for retaining old samples from known devices. The research question we address in this Chapter is how can we develop a scalable IoT fingerprinting approach to improve device recognition accuracy and memory consumption. Our experiments show that ScaNeF-IoT performs similarly to AutoIoT when assessed on all the IoT devices included in the same dataset, and outperforms IoT-Portrait when incrementally introducing new IoT devices. While this is a preliminary evaluation, the results are promising towards establishing OSL as the ML classifier to use to achieve scalable IoT fingerprinting.

This line of research can be developed further by assessing and comparing the retraining time against existing approaches. Also, other OSL algorithms beyond AMF can be tested, as well as combining different types of features (e.g., statistics of the sessions) to explore the possibility to improve the detection accuracy even further.

Chapter 5

IoT Network Traffic Feature Analysis

5.1 Introduction

The scalable IoT fingerprinting approach proposed in Chapter 4, ScNef-IoT, operates an OSL-based classifier to fingerprint large amounts of IoT devices on the fly while reducing storage requirements and computation time. ScNef-IoT achieves comparable accuracy with AutoIoT (25) for device recognition using session payload features. However, many other features can be examined using OSL to enhance the fingerprinting performance. Therefore, we aim to address the following research question:

RQ4.*What features can provide the best performance for fingerprinting IoT devices?*

This Chapter explores the network traffic features of IoT devices, with a specific aim of understanding how these devices interact with their network environment. It introduces an analytical study of various feature extraction methods to understand the device behaviour and how it can contribute to the detection of IoT devices.

We implement different feature extraction methods, including packet-based, flow-based, and session-based. We evaluate our implementation in terms of IoT device identification accuracy, FPR, and time efficiency using The UNSW IoT Traces dataset (12). Our results indicate that the payload session of size 784 bytes yields the best performance with minimal FPR and latency.

The novelty of this Chapter lies in the analysis of various extraction methods to develop a unique IoT device fingerprint for IoT device fingerprinting that is based on OSL. The main contributions of this work are:

- a novel method that explores various feature extraction methods to create unique fingerprints for IoT devices, employing OSL for IoT device fingerprinting;
- an implementation of feature extraction methods, including packet-based, flow-based, and session-based;
- an experimental evaluation of each feature extraction method based on AMF and the UNSW IoT Traces dataset (12), with an accuracy, FPR, and time efficiency comparison showing the payload session outperforming the other feature extraction methods.

The rest of this Chapter is organised as follows. Section 5.2 discusses the method of selecting instances of interest from network traffic, including packet-based, flow-based, and session-based approaches, to generate feature vectors for IoT fingerprinting. Section 5.3 presents the evaluation of each feature vector. Finally, a summary of this chapter is in Section 5.4.

5.2 Method

This section discusses different approaches for selecting instances of interest from network traffic to generate feature vectors for IoT fingerprinting. Network traffic generated by IoT devices is preprocessed by packet-based as in Section 5.2.1, flow-based as in Section 5.2.2, and session-based as in Section 5.2.3. The preprocessing stage extracts features from each method to form a feature vector.

5.2.1 Packet-Based Features

In this section, we investigate the packet-based traffic features of IoT devices, primarily focusing on header field features. We extracted features proposed by IoT Sentinel (2), which are detailed in Table 5.1. Each IoT device transmits a sequence of packets, denoted as $P = \{p_1, p_2, p_3, ...\}$. For every packet $p_i \in P$, we parse and extract 23 features to construct a feature vector $PkFV = \{f_1, f_2, f_3, ..., f_{23}\}$.

The majority of these features are binary, indicating the use of a communication protocol with a value of 1. This includes 16 protocols spanning various network layers. Additionally, three binary features specifically indicate the use of IP header options, such as padding and router alert, and the presence of raw data. Non-binary features include the size of the packet and the frequency of IP address usage, setting a counter to each unique destination IP address and incremented with each subsequent appearance of that IP address. Lastly, the source and destination ports are classified into predefined classes as follows:

TABLE 5.1: Packet-based feature vector $PkFV$.

Type	Feature	#
Link layer protocol	ARP/LLC	2
Network layer protocol	IP/ ICMP/ ICMPv6/EAPoL	4
Transport layer protocol	TCP/UDP	2
Application layer protocol	HTTP/HTTPS/DHCP/BOOTP/ SSDP/DNS/mDNS/NTP	8
IP options	Padding/RouterAlert	2
Packet content	Size /Rawdata	2
IP address	Destination IP counter	1
Port class	Source/Destination	2

$$f(x) = \begin{cases} 0 & x \notin [0, 65535] \Rightarrow \text{no port} \\ 1 & x \in [0, 1023] \Rightarrow \text{well-known ports} \\ 2 & x \in [1024, 49151] \Rightarrow \text{registered ports} \\ 3 & x \in [49152, 65535] \Rightarrow \text{dynamic ports} \end{cases} \tag{5.1}$$

5.2.2 Flow-Based Features

To analyse flow-based features, we establish a fixed, non-overlapping time window for each device to extract features consistently. Within this window, we derive statistical features to construct three distinct feature vectors $FwFV_{bi}$, $FwFV_{bi,entropy}$, and $FwFV_{bi,uni}$. Further details on these feature vectors and their specific applications are discussed comprehensively in this section.

Within a specified time window T, each IoT device transmits a sequence of packets represented as $P = \{p_1, p_2, p_3, ...\}$. The number of packets within each time window may vary, reflecting changes in device activity or network conditions from one window to another. We analyse bidirectional traffic to construct the initial feature vector $FwFV_{bi}$, which is detailed in Table 5.2. The traffic volume, indicated by the number of packets within T, is analysed through flow statistical features (e.g., min and max). We also extract header statistical features based on packet size and TTL values. Devices typically employ a variety of protocols at different network layers, we track the usage of 18 protocols within this time window, as their relevance to fingerprinting IoT devices is supported by the findings in (19). We investigate protocols not listed in Table 5.1, particularly including protocols for security, such as SSH and TLS. Additionally, we examine time interval features by measuring the intervals between consecutive packets and deriving their statistical properties as well.

We extend the second feature vector, $FwFV_{bi,entropy}$, by including payload entropy along with $FwFV_{bi}$ features, as outlined in Table 5.3. Payload entropy quantifies the

TABLE 5.2: Flow-based feature vector $FwFV_{bi}$ for IoT device fingerprinting.

Type	Feature	#
Flow	packet count	1
IAT	mean, min, max, std, var, median	6
Packet length	mean, min, max, std, var, median	6
TTL	mean, min, max, std, var, median	6
Link layer protocol	ARP count	1
Network layer protocol	ICMP/ ICMPv6/EAPoL count	3
Transport layer protocol	TCP/UDP count	2
Application layer protocol	HTTP/HTTPS/DNS/mDNS/ BOOTP/SSDP/NTP/SMB/ SSH/QUIC/STUN/TLS count	12

TABLE 5.3: Flow-based feature vector $FwFV_{bi,entropy}$ for IoT device fingerprinting.

Type	Feature	#
Flow	packet count	1
IAT	mean, min, max, std, var, median	6
Packet length	mean, min, max, std, var, median	6
TTL	mean, min, max, std, var, median	6
Link layer protocol	ARP count	1
Network layer protocol	ICMP/ ICMPv6/EAPoL count	3
Transport layer protocol	TCP/UDP count	2
Application layer protocol	HTTP/HTTPS/DNS/mDNS/ BOOTP/SSDP/NTP/SMB/ SSH/QUIC/STUN/TLS count	12
Payload length	TCP/UDP payload entropy	2

degree of variability in the bytes of a packet's payload. High entropy is indicative of significant randomness, typically associated with encrypted or compressed data. In contrast, low entropy suggests more predictable and less random data, which is usually seen in plain text or basic data formats. This metric offers valuable insights into the data's nature, which is known to vary across different IoT devices, as reported in studies by Kostas et al.(109) and Jiao et al.(90). Consequently, we assess the entropy for payloads of both TCP and UDP, defining the entropy, H, for a sequence of X bytes, in the following manner:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log(P(x_i)) \tag{5.2}$$

where $P(x_i)$ is the probability of occurring byte x_i in the payload and n is the length of the payload.

In the third feature vector, $FwFV_{bi,uni}$, we delve into granular details by incorporating both uni-directional and bi-directional traffic data within T, as presented in Table 5.4. For each device, we extract and compute statistical features from the packet lengths,

including minimum, first quartile (1q), second quartile (2q), third quartile (3q), maximum, mean, variance, skewness, and kurtosis. We also measure the entropy of these packet lengths, as specified in Equation 5.2, to assess the variability of packet sizes, determining whether they are uniformly low in entropy or diverse and high in entropy.

Furthermore, we track the number of packets (count), the cumulative length of packets, and the average packet size. Source and destination port classifications are analysed according to Equation 5.1, spanning incoming, outgoing, and bi-directional traffic.

Consistent with earlier vectors, $FwFV_{bi,uni}$ processes IAT and TTL statistics along with protocol data from bi-directional traffic. Additional features recorded include TCP window size statistics and the Don't Fragment (DF) flag ratio, which reflects the percentage of packets that have the DF bit set to 1 relative to the total number of packets.

TABLE 5.4: Flow-based feature vector $FwFV_{bi,uni}$ for IoT device fingerprinting.

Type	Feature	#
in/out/bi Flow	packet count	3
in/out/bi packet length	mean, min, max, std, var, kurtosis, skew, 1q, 2q, 3q, entropy	33
in/out/bi packet byte	bytes, median	6
IAT	mean, min, max, std, var, median	6
TTL	mean, min, max, std, var, median	6
TCP window size	mean, min, max, std, var, median	6
DF flag	ratio	1
Link layer protocol	ARP count	1
Network layer protocol	ICMP/ ICMPv6/EAPoL count	3
Transport layer protocol	TCP/UDP count	2
Application layer protocol	HTTP/HTTPS/DNS/mDNS/ BOOTP/SSDP/NTP/SMB/ SSH/QUIC/STUN/TLS count	12
In/out/bi Payload length	TCP/UDP payload entropy	6
In/out/bi Port class	Source/Destination count	8

5.2.3 Session-Based features

In the session-based feature vector, we adopt the approach detailed in Section 4.2.1. We employ a method similar to Kotak and Elovici (115), which involves generating images from session payloads. Instead of images, however, we utilise packet headers to distinguish bi-directional TCP and UDP sessions. The session payloads are then combined and adjusted to a uniform size, either by trimming excess or padding insufficient data, to form a consistent feature vector $PsFV$ for each session.

5.3 Evaluation

This section analyses each feature extraction method by implementing and evaluating their accuracy, and FPR using the OSL approach introduced in Section 4.2.2. It also evaluates the training time, the time taken for the initial training of the model and the testing time for each sample.

5.3.1 Dataset

We use the UNSW IoT Traces dataset (12) as in Section 4.3, to extract feature vectors from different extraction methods. We evaluate the accuracy of each feature vector method over the whole dataset considering packet-based features $PkFV$, flow-based features $FwFV_{bi}$, $FwFV_{bi,entropy}$, and $FwFV_{bi,uni}$ with different values of window time T, and session-based features $PsFV$ with different payload buffer size S. We use all 23 IoT devices included in the dataset and split the data into a 7:3 ratio.

Table 5.5 shows a notable imbalance across all the feature vector extraction methods especially for similar devices. This is particularly evident with Insteon Cameras, where the wired version produces more traffic than the wireless one, which might complicate the identification process. Similarly, in session-based data for Belkin Wemo devices, the Belkin Wemo Motion Sensor exhibits a higher number of payload sessions compared to the Belkin Wemo Switch.

5.3.2 Implementation

For packet-based features, we utilise Scapy (161), a Python library for parsing the network packets. For flow-based features, we first identify packets using Scapy and then experiment with different values for the window time T to evaluate its impact on classifier performance. The implementation of session-based features follows the guidelines outlined in Section 4.2.3.

5.3.3 Evaluation Approach

Our experiments aim to assess each feature vector extracted from different extraction methods by analysing their validity in fingerprinting IoT devices through accuracy comparison. In each experiment, we evaluate the AMF model with all 23 IoT devices in the dataset, where we adjust $n\_classes$ to 23 in the *dirichlet* parameter, $\frac{1}{n\_classes}$, representing the unique number of IoT devices as discussed in Section 4.2.3. The *dirichlet* distribution is used to regulate class frequencies within each decision node, preventing overfitting and stabilising predictions in the model.

TABLE 5.5: The number of traffic instances extracted from the UNSW IoT Traces dataset using packet-based, flow-based and session-based.

#	IoT Device Name	Packet	Flow-based			Session
		based	1 minute	5 minutes	10 minutes	based
1	Amazon Echo	1,343,081	28,168	5,634	2,817	3,491
2	Belkin Wemo Motion Sensor	887,889	28,127	5,634	2,817	48,883
3	Belkin Wemo Switch	730,069	28,126	5,634	2,817	8,939
4	Blipcare Blood Pressure Meter	229	4	4	4	4
5	Dropcam	4,240,606	28,168	5,634	2,817	35
6	HP Printer	210,047	26,241	5,634	2,817	150
7	iHome Power Plug	65,337	10,370	2,082	1,044	153
8	Insteon Camera wired	736,768	20,279	4,058	2,030	8,702
9	Insteon Camera wireless	462	10	4	3	102
10	Light Bulbs LiFX Smart Bulb	172,662	20,291	4,063	2,032	52
11	Nest Dropcam	149,764	36	8	4	29
12	NEST Protect Smoke Alarm	4,302	26	23	23	84
13	Netatmo Weather Station	267,688	10,605	4,061	2,808	2,338
14	Netatmo Welcome	673,754	28,168	5,634	2,817	2,728
15	PIX STAR Photo Frame	68,520	8,432	3,281	1,789	1,118
16	Samsung SmartCam	1,160,255	28,168	5,634	2,817	10,053
17	Samsung Smart Things	571,063	28,164	5,634	2,817	24
18	TP-Link Day Night Cloud Camera	318,295	18,135	3,637	1,820	1,541
19	TP-Link Smart Plug	41,316	7,960	4,197	2,099	239
20	Triby Speaker	200,646	27,796	5,596	2,799	131
21	Withings Aura Smart Sleep Sensor	387,669	20,345	4,070	2,035	3,584
22	Withings Smart Baby Monitor	685,844	21,001	4,202	2,101	5,545
23	Withings Smart Scale	5,581	43	35	35	20
	Total	12,921,847	388,659	84,393	43,162	97,945

5.3.4 Results

Figure 5.1 illustrates the accuracy comparison of feature vectors across different feature extraction methods. In the packet-based experiment, the model uses $PkFV$ and achieves a maximum accuracy of 78.65%. The flow-based feature extractions demonstrate a notable improvement in accuracy, surpassing the packet-based method by over 15%. We evaluate three flow-based feature extraction method, $FwFV_{bi}\_T$, $FwFV_{bi,entropy}\_T$, and $FwFV_{bi,uni}\_T$, using three different time windows T: 1 minute, 5 minutes, and 10 minutes. $FwFV_{bi}\_5m$, $FwFV_{bi,entropy}\_5m$, and $FwFV_{bi,uni}\_5m$ perform best, suggesting that a 5-minute time window likely provides a sufficient data to accurately reflect the behaviour of IoT devices. Even though $FwFV_{bi}\_5m$ and $FwFV_{bi,entropy}\_5m$ show similar accuracy, the latter exhibits a slight enhancement, suggesting the contribution of the payload entropy feature. However, $FwFV_{bi,uni}\_5m$

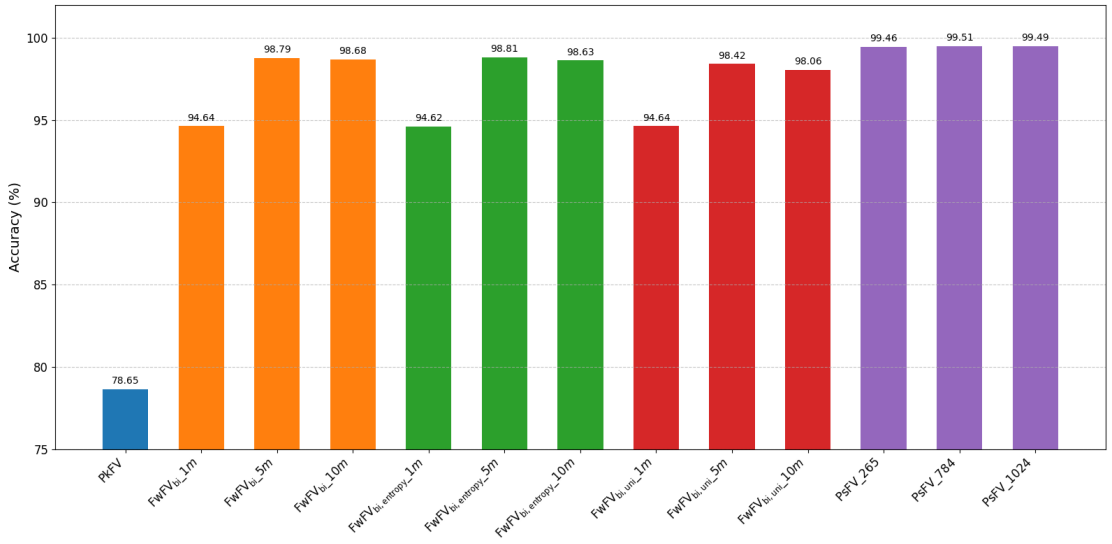


FIGURE 5.1: Accuracy comparison between feature vectors derived from different feature extraction methods, based on a 30% holdout validation strategy

shows lower accuracy compared to other feature vectors that may focus more exclusively on bi-directional features. This decrease could be attributed to the inclusion of unidirectional flows, which potentially dilute the more contextually rich bi-directional data, leading to a reduced ability to detect IoT device behaviours. The presence of unidirectional data might introduce noise, obscuring key patterns and thus affecting the overall effectiveness of the feature vector in complex network environments.

For the session-based features, as in Section 4.3.1, we extracted the payload sessions $PsFV\_S$ with three different buffer sizes S: 265, 748, and 1024 bytes. $PsFV\_784$ outperform other buffer sizes achieving 99.51% accuracy, compared to 99.46% for the 265 bytes payload and 99.49% for the 1024 bytes payload. The $PsFV\_784$ is likely the most effective at capturing relevant data without including too much noise or missing critical information. The results suggest that this specific buffer size efficiently captures the essential characteristics of the payload sessions, demonstrating superior performance compared to other flow and packet-based feature vectors.

Figure 5.2 shows the Average FPR, as defined in Equation 2.4, representing the proportion of negative samples incorrectly classified as positive across all classes. We find the FPR for each class by considering that class as the positive class and all other classes as negative. The average FPR is then computed by averaging the FPRs of all individual classes. The Packet-based features $PkFV$ show notably the highest FPR of 1%. Flow-based $FwFV_{bi}$ shows varying FPRs (0.25%, 0.06%, 0.06%) depending on the window time (1 minute, 5 minutes, 10 minutes). It suggests that shorter analysis periods may result in higher false positives. Similarly, $FwFV_{bi,entropy}$ and $FwFV_{bi,uni}$ exhibit trends where the FPR generally decreases as the observation period increases.

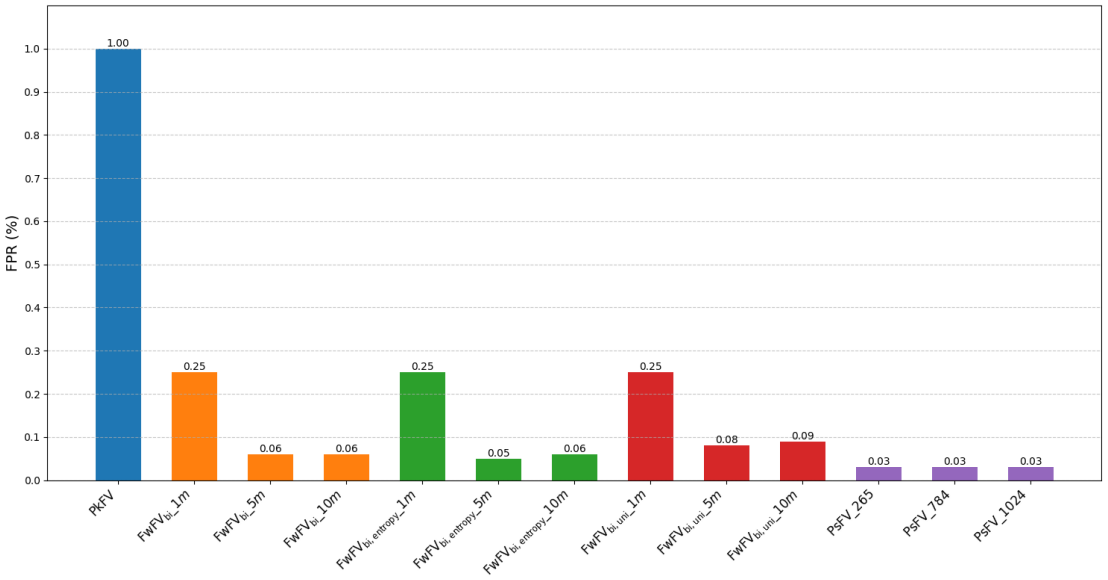


FIGURE 5.2: Average FPR comparison between feature vectors extracted from different feature extraction methods, based on a 30% holdout validation strategy

Notably, $FwFV_{bi,entropy}\_5m$ consistently showcases better performance across all flow-based extraction methods.

Session-based features represented by payload session features, $PsFV\_265$, $PsFV\_784$, and $PsFV\_1024$ show very low FPRs with 0.03%, indicating that the payload session features reliably distinguish between positive and negative classes with minimal error. This type of feature extraction shows excellent performance among all feature extraction methods, which might indicate that payload session features are strong discriminators of IoT devices.

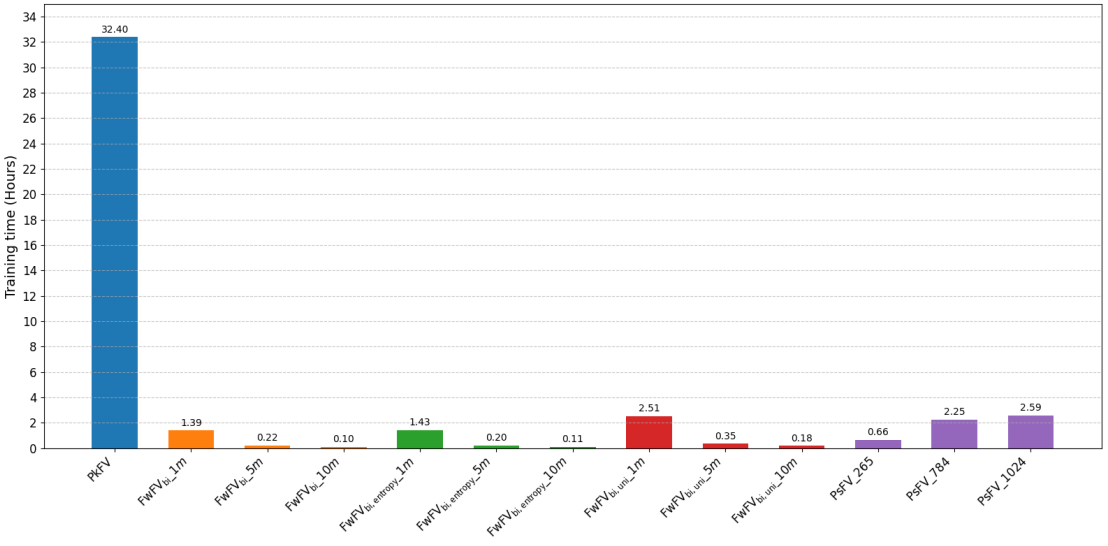


FIGURE 5.3: Training time comparison between feature vectors from different feature extraction methods, based on a 30% holdout validation strategy

Figure 5.3 and Figure 5.4 present the training time, the time taken for the initial training of the model and the testing time for each sample using different feature extraction methods. The training time comparison in Figure 5.3 highlights a significant disparity between the packet-based features *PkFV* and other feature extraction methods. *PkFV* requires an extensive 32.40 hours of training, making it the most time-consuming approach due to the high volume of training samples. In contrast, the flow-based and session-based features exhibit considerably lower training times. Among the flow-based methods, $FwFV_{bi}\_10m$ and $FwFV_{bi,entropy}\_10m$ have the shortest training time at 0.1 hours, whereas $FwFV_{bi,uni}\_1m$ requires 2.51 hours, making it the most time-intensive flow-based extraction type due to its complex features. The session-based features *PsFV* also show efficient training times, even with large buffer sizes, as seen with *PsFV*_784 and *PsFV*_1024 recording training duration of 2.25 and 2.59 hours, respectively, demonstrating efficiency in handling varying buffer sizes.

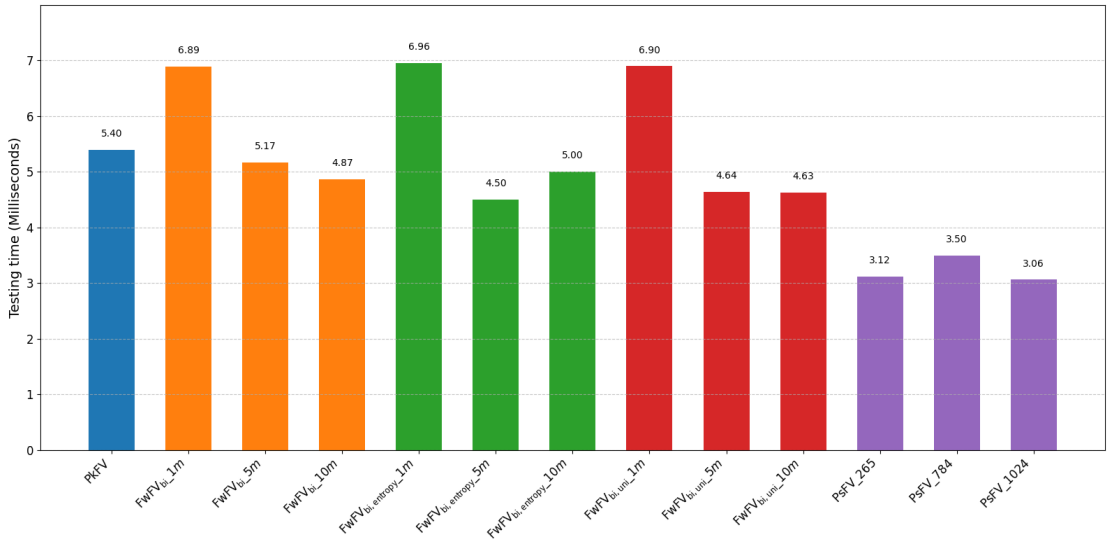


FIGURE 5.4: Testing time comparison between feature vectors extracted from different feature extraction methods, based on a 30% holdout validation strategy

The testing time in Figure 5.4 demonstrates the efficiency of the session-based features *PsFV* with at most 3.5 milliseconds per sample. This performance is closely followed by $FwFV_{bi,entropy}\_5m$ and $FwFV_{bi,uni}\_10m$ which have testing times of 4.5 and 4.63 milliseconds, respectively. These low latency measurements are crucial for practical IoT device fingerprinting systems, where *PsFV* demonstrate low responsive time among all.

Overall, the session-based feature *PsFV*_784 followed by flow-based feature $FwFV_{bi,entropy}\_5m$ significantly reduce both training and testing times compared to other feature extraction methods. This efficiency in training and testing times, coupled with the high accuracy and minimal error rate, makes these feature vectors highly advantageous for IoT device fingerprinting.

5.3.5 Hyperparameter Tuning

This section focuses on tuning the hyperparameter of the AMF classifier, as discussed in Section 4.2.3. Optimising hyperparameters in OSL is a significant challenge due to the dynamic nature of the learning model, which handles each sample individually in a streaming data environment and adapts continuously.

Popular methods for automated hyperparameter optimisation, including grid and random search, and Bayesian optimisation, typically involve multiple iterations over data batches. This process can be resource-intensive and costly because it necessitates storing large volumes of data (162). This requirement conflicts with the stream processing paradigm, which focuses on processing data while minimising the need for extensive data storage. However, optimising hyperparameters in OSL remains an open challenge and addressed as future work (163; 162).

Even though we do not use a direct method for optimising hyperparameters, we analyse different values for two key parameters for their critical impact on model performance: *n_estimators* and *dirichlet*. The *n_estimators* parameter, which denotes the number of trees in the forest, is crucial because it directly influences the model's accuracy. More trees can improve prediction stability by reducing variance, though this comes at the cost of increased computational demand. Additionally, a large number of trees can cause overfitting, where the model becomes too tailored to the training data and performs poorly on new, unseen data. Although the praralisaiton process can mitigate that, the current software library, river, does not support praralisaiton with AMF. Therefore, finding an optimal number of trees balances performance with efficiency. The *dirichlet* parameter, which regulates class frequencies at each node, is essential for managing class imbalance, ensuring the model remains sensitive to less frequent classes and preventing overfitting. This makes *dirichlet* particularly valuable in diverse data environments. In contrast, parameters like *step*, *use_aggregation*, and *split_pure* have more specific roles or recommended settings that render them less variable and thus less critical for broad optimisation. Generally, *step* remains fixed as its default is optimal for most cases, *use_aggregation* is almost always enabled to ensure model robustness, and *split_pure* is concerned with a specific situation, whether to split nodes that contain only samples of one class.

The optimisation targets the classifier, fed by two main feature vectors identified as most effective in Section 5.2. These are $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$. The hyperparameter space considered for *n_estimators* is varied using values derived from the set $\{10, 50, 100\}$. The default value is 10, which serves as a baseline, offering a reference point to assess how increasing the number of trees to 50 and 100 trees affects the model. These increments provide insight into the performance of the model and complexity in terms of training and testing time. The hyperparameter space considered for $dirichlet = \frac{1}{n\_classes}$, where $n\_classes \in \{2, 23, 100, 1000\}$. The default

value is $n\_classes = 2$, which is particularly well-suited for binary classification problems. When $n\_classes = 23$, the *dirichlet* value is tailored to match the number of IoT devices in the dataset. Higher $n\_classes$ values $\{100, 1000\}$ offer a view into how the model behaves under minor regularisation adjustments. We assess the accuracy, training time, and testing time (latency) for each experiment.

Table 5.6 addresses the impact of the number of estimators. Increasing the number of estimators generally improves the accuracy for both feature vectors. For the $FwFV_{bi,entropy}\_5m$, accuracy increases marginally from 97.82% with 10 estimators to 97.92% with 100 estimators. For the $PsFV\_784$, accuracy also increases from 99.29% with 10 estimators to 99.44% with 100 estimators. However, there is a significant increase in training and latency time as the number of estimators rises. The $FwFV_{bi,entropy}\_5m$ shows a notable increase, from 0.2 hours to 2.2 hours with 10 estimators to 20 hours with 100 estimators. For the $PsFV\_784$, training time increases stickily from 2 hours for the same range of estimators. The testing time per sample also rises with the number of estimators. For $FwFV_{bi,entropy}$, it increases from 4.6 ms to 49.46 ms, and for $PsFV\_784$, the time increases from 3.4 ms to 32.7 ms.

TABLE 5.6: Impact of the number of estimators on AMF classifier with default $dirichlet = \frac{1}{2}$

Feature vector type	$FwFV_{bi,entropy}\_5m$			$PsFV\_784$		
n_estimators	10	50	100	10	50	100
Accuracy (%)	97.82	97.89	97.92	99.29	99.39	99.44
Train time/train set (hours)	0.2	1.1	2.2	2	10	20
Test time/sample (ms)	4.6	24.3	49.46	3.4	16.5	32.7

Table 5.8 shows the effect of different *dirichlet* values on the accuracy of the AMF model for each feature vector. The accuracy remains relatively stable for both feature vectors, indicating that the *dirichlet* parameter settings $n\_classes$ to larger values do not significantly affect classifier performance. For both feature vectors, the training and testing times show minimal variation across the different *dirichlet* values. Consequently, the performance of the classifier is robust to variations in the *dirichlet* parameter setting.

TABLE 5.7: Impact of different $n\_classes$ values for $dirichlet = \frac{1}{n\_classes}$ parameter on AMF classifier with the default $n\_estimators = 10$

Feature vector type	$FwFV_{bi,entropy}\_5m$				$PsFV\_784$			
n_classes	2	23	100	1000	2	23	100	1000
Accuracy (%)	97.82	98.81	98.84	98.82	99.29	99.51	99.52	99.52
Train time/train set (hours)	0.2	0.2	0.2	0.2	2	2.25	2.23	2.17
Test time/sample (ms)	4.6	4.5	4.6	4.6	3.4	3.5	3.3	3.3

Table 5.8 shows the impact of different values for *n_estimators* parameter. We set the $dirichlet = \frac{1}{n\_classes}$ parameter to $n\_classes = 23$ indicating the number of IoT devices in the UNSW IoT Traces dataset. It clearly shows the accuracy improvements are marginal when increasing the number of estimators, especially for the $FwFV_{bi,entropy\_5m}$ feature type where there is a negligible decrease beyond 50 estimators. In contrast, the impact on training and testing times is more substantial. This suggests that while increasing estimators can slightly enhance accuracy, it comes at a considerable cost in computational resources, particularly evident in training and testing duration. This trade-off between accuracy and latency indicates that careful consideration is needed when scaling the number of estimators, especially for applications where response time is critical.

TABLE 5.8: Impact of the number of estimators on AMF classifier with the $dirichlet = \frac{1}{23}$

Traffic feature type	$FwFV_{bi,entropy}\_5m$			$PsFV\_784$		
n_estimators	10	50	100	10	50	100
Accuracy (%)	98.81	98.89	98.86	99.51	99.57	99.44
Train time/train set (hours)	0.20	2.3	2.3	2.25	10.4	20.16
Test time/sample (ms)	4.5	50.0	50.6	3.5	16.3	32.2

The configuration of $n\_estimators = 10$ and $dirichlet = \frac{1}{23}$ is particularly advantageous for IoT fingerprint models. This setup not only minimises the computational resources required but also maintains an excellent level of performance, making it an ideal choice for operational environments where both accuracy and efficiency are critical.

5.3.6 Handling Imbalanced Data

In this section, we address the issue of class imbalance in two primary feature vectors used for IoT device fingerprinting: $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$, as detailed in Table 5.5.

For $FwFV_{bi,entropy}\_5m$, we implement SMOTE to oversample the minority classes, creating synthetic samples until the minority class size matches the majority class size of 5,634 samples.

In contrast, the $PsFV\_784$ demonstrated significant class imbalances across IoT devices, evidenced by a wide disparity in sample sizes. The Belkin WeMo Motion sensor has the highest number of samples at 48,883, whereas the Blipcare Blood Pressure Meter has the lowest with only 4 samples. To address this, we employed a combination of under-sampling techniques to reduce the size of the majority classes to 3,000 samples, and then applied SMOTE to enhance the minority classes to the same level. SMOTE is a statistical technique designed to increase the number of samples in a dataset in a balanced way by creating synthetic samples rather than by

over-sampling with replacement. Initial experiments include three distinct undersampling strategies, each followed by the application of SMOTE: random undersampling, K-means to select diverse samples by grouping similar data points, and agglomeration clustering to retain representative samples from distinct clusters within the majority class. Despite these varied approaches, none succeed in achieving an accuracy greater than 18%. This outcome highlights the significant challenges in managing the imbalance without distorting the data, which severely undermines the effectiveness of the classifier. To improve the quality of synthetic samples and remove noise, we subsequently adopted SMOTEENN (164), which combines SMOTE with the Edited Nearest Neighbours (ENN) technique. This approach effectively generates representative synthetic examples for the minority classes and cleans the data by eliminating noise-introducing samples.

Table 5.9 summaries the impact of SMOTE and SMOTEENN sampling techniques on the performance metrics of a classifier across two different traffic feature vectors: $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$, respectively. For $FwFV_{bi,entropy}\_5m$, applying SMOTE slightly reduced accuracy from 98.81% to 98.70%, and marginally increased training and latency time. Similarly, for the $PsFV\_784$, employing SMOTEENN resulted in a slight decrease in accuracy from 99.51% to 98.96%, despite drastically increasing the training time from 2.25 to 28.6 hours; and also marginally increasing the testing time from 3.5 ms to 3.7 ms. These outcomes suggest that, in this context, the use of these advanced sampling techniques may not be beneficial and could potentially introduce inefficiencies in model training and execution without enhancing accuracy.

TABLE 5.9: Handling imbalanced data using data sampling techniques on two feature vector types using UNSW IoT Traces dataset

Feature vector type	$FwFV_{bi,entropy}\_5m$		$PsFV\_784$	
Data sampling technique	Original	SMOTE	Original	SMOTEENN
Accuracy (%)	98.81	98.70	99.51	98.96
Train time/train set (hours)	0.20	0.30	2.25	28.6
Test time/sample (ms)	4.5	5.0	3.5	3.7

5.4 Summary

This Chapter assesses the OSL approach using AMF to examine different feature extraction methods, including, packet-based, flow-based, and session-based. The research question we address is whether key features can be identified to effectively create a unique fingerprint for IoT device identification. Our experiments show that the payload of size 784 bytes extracted from sessions performs the best in terms of accuracy and time efficiency using UNSW IoT Traces dataset. Despite significant data imbalance, the payload feature vector maintains high accuracy with minimal FPR.

Indeed, hyperparameter tuning can enhance the performance of IoT fingerprinting. For AMF classifier, we focus on the number of estimators and the *dirichlet* parameters which are crucial for enhancing IoT device detection by reducing variance and overfitting. We observe that increasing the number of estimators does not significantly improve the accuracy of the classifier, but it does increase the latency by up to ten times. Additionally, the *dirichlet* parameter has minimal impact when setting it to a large value, greater than the actual number of classes in the dataset.

This research can be extended by discovering new traffic patterns for known IoT devices and identifying unknown or newly introduced IoT devices in a network.

Chapter 6

Scalable Network Fingerprinting with Unknown IoT Device Type Detection

6.1 Introduction

This Chapter builds on the insights from previous chapters, which introduced ScaNeF-IoT for scalable network IoT fingerprinting in Chapter 4 and examined different feature extraction types in Chapter 5 to enhance IoT fingerprinting. Here, we shift our focus to a critical issue: the detection of unknown IoT devices for scalable IoT device fingerprinting. As IoT environments continue to grow and evolve, scalability IoT device fingerprinting primarily focuses on recognising known devices; however, the challenge of identifying unknown or new devices remains significant.

The detection of unknown IoT devices poses scalability challenges across several commonly used methods. Threshold-based approaches rely on static thresholds, which can lead to high false positive or negative rates as these thresholds may not adapt well to the growing diversity of devices (113). Distance-based metrics, like Euclidean or Edit distance, can detect unknown devices by comparing new fingerprints against known ones. However, their computational complexity grows exponentially with the number of devices, making them unsuitable for large IoT ecosystems (2; 19). Statistical tests like the Kolmogorov-Smirnov test are useful for detecting behavioural deviations, however, they are limited by the need for pre-existing data distributions and may face memory constraints as datasets grow in size as in AutoIoT (25). In fact, AutoIoT requires 1.5 $\sim$ 2.5 hours of incoming traffic representing 3 $\sim$ 5 samples to detect unknown IoT devices. These limitations highlight the need for more adaptable, scalable methods for detecting unknown IoT devices.

This Chapter addresses the limitations of existing approaches for unknown IoT device detection, which require access to known IoT device data and can lead to memory exhaustion. We propose to utilise the z-score measure to accurately detect unknown IoT devices while minimising memory consumption. The z-score technique falls under the statistical-based category for unknown device detection in Figure 3.1. It measures how far a given data point (incoming sample) deviates from the mean of known samples in terms of standard deviations, allowing for the detection of unknown devices based on their statistical differences from expected behaviour. Therefore, we will address the following research question:

> **RQ5.** *How can we accurately detect unknown IoT devices in a scalable IoT device fingerprinting system while reducing memory consumption?*

This Chapter introduces a novel approach for scalable IoT device fingerprinting with unknown IoT device detection. The output from OSL classification models is a set of probabilities indicating the likelihood/probability that a given sample belongs to known classes. Therefore, when the sample is from an unknown class the probabilities will reflect the confidence of the model with known classes. We exploit the maximum probability value from these outputs. For samples of known classes, the maximum probabilities are usually close to 1, showing high confidence. While samples of unknown classes, the maximum probabilities are lower and more scattered $[0, 1]$.

We will use the maximum probabilities of incoming samples to calculate the z-score, which is a statistical measure that quantifies the distance of a sample from the mean of the known distribution, expressed in terms of standard deviations. To effectively identify unknown samples, we employ a z-score by leveraging the maximum probability of the unknown distribution along with the mean μ and standard deviation (sd) σ of the maximum probability of known distribution. As we only need μ and σ of the known distribution, the approach eliminates the need to repeatedly process large volumes of historical data for each unknown analysis.

We implement our approach using z-score to detect the presence of unknown IoT devices along with AMF as OSL algorithm and two different feature extraction methods. These methods are fixed-size payload sessions and statistical features derived from fixed window time flows, both of which demonstrate high accuracy in IoT device fingerprinting, as discussed in Chapter 5. We evaluate our implementation in terms of the accuracy in detecting unknown IoT devices using the UNSW IoT Traces dataset (12) and comparing the results with the state-of-the-art scalable IoT fingerprinting approach, namely AutoIoT (25). The authors of AutoIoT evaluated their approaches using the same dataset; to enable a fair comparison, our experiments employ the same evaluation strategy as theirs. Our approach shows comparable results with substantially lower computational demand and latency. Furthermore, we

evaluate the whole system periodically using the same dataset achieving over 94% average accuracy in identifying known IoT device types and 100% accuracy for the detection of unknown IoT devices using only one payload session.

The novelty of this Chapter lies in the fact that our approach is the first approach proposed for scalable network fingerprinting with unknown IoT device detection that is based on OSL and z-score. The main contributions of this work are:

- a novel scalable network fingerprinting with unknown IoT device detection, based on z-score, OSL, and fixed-size session payloads;
- implementation and evaluation of unknown IoT device detection, based on z-score, AMF, and two feature vector types UDP/TCP payloads and statistical features from fixed window time flows, using the UNSW IoT Traces dataset (12). The results show comparable results with AutoIoT (25) with more resource-efficient using UDP/TCP payloads.
- an implementation of the scalable network fingerprinting with unknown IoT device detection system using a periodic streaming data.The data is processed periodically based on non-overlapping window stream data. The evaluation consistently shows the UDP/TCP payloads outperform those based on other features with an accuracy of over 94% in identifying known IoT devices and achieves 100% accuracy in detecting unknown IoT devices from just a single payload session. This showcases not only high accuracy but also efficient resource management in operation.

The rest of this chapter is organised as follows. Section 6.2 presents the approach of scalable network fingerprinting with the unknown IoT devices detection. Section 6.3 shows the experimental evaluation of the detection of IoT devices using z-score. Section 6.4 evaluates the overall system. Finally, a summary of this chapter is in Section 6.5.

6.2 Approach

This section discusses a practical approach to a network management situation where IoT devices join an existing network. These devices could be either *known*, which the IoT fingerprinting classifier has seen during training, or *unknown*, which is newly introduced in the network. Detecting unknown devices in the network is crucial and essential for maintaining network integrity and adapting to the evolving landscape of device types within the network.

Figure 6.1 shows our approach for scalable IoT device fingerprinting with unknown IoT device detection. The system utilises OSL, where the model continuously adapts

and updates based on incoming data. This adaptability is crucial in IoT environments, where device behaviours may evolve or new devices might join the network. This characteristic ensures that the system remains effective and current as the IoT network environment evolves, allowing it to adjust and improve its IoT device recognition.

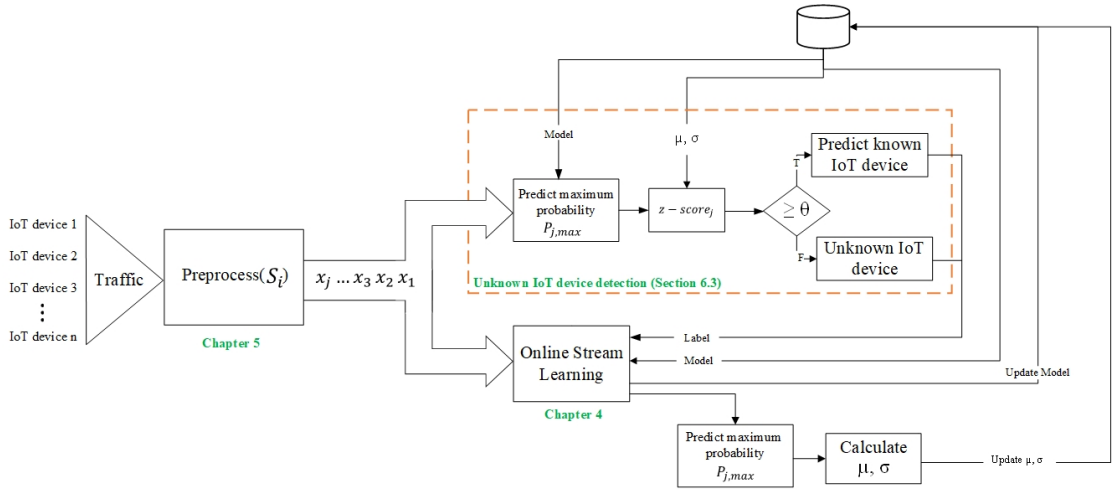


FIGURE 6.1: The scalable network fingerprinting with unknown IoT device type detection approach.

As the IoT devices send traffic in the network, we define a non-overlap sliding window of size w to manage and process the data stream periodically. This method organises the stream data flow into windows, denoted as $S = \{S_0, S_1, ..., S_n, ...\}$, where n is the number of windows.

At each window, we preprocess the data S_i to extract the feature vectors, $X_i = \{x_1, x_2, ..., x_{m_i}\}$, where m_i indicating the number of features in the i-th window, as discussed in Chapter 5. It is important to note that m_i can vary from one window to another, i.e., $m_i \neq m_j$ where $i \neq j$. Initially, we will train an OSL model, $M(X_0, Y_0)$, using the zeroth window of data as discussed in Chapter 4, where Y_0 are the ground truth labels for X_0. Additionally, we will calculate the z-score parameters, μ and σ of the maximum probability of known distribution. The resulting M, μ, and σ are stored on disk for future reference and usage in continuous learning. The OSL training is shown in Algorithm 1.

For each subsequent window S_i, where $i > 0$, the system will apply unknown IoT device detection and then update the model as shown in Algorithm 2. S_i will be preprocessed to extract feature vector X_i. By, utilising M, we find the probability $P_j = \{p_1, p_2, ..., p_{c_k}\}$, where c_k is the number of known classes, for each feature vector $x_j \in X_i$. We compute the maximum probability $P_{j,max}$ and then calculate the z–$score_j$ (Algorithm 1, lines 7 and 8). To detect unknown IoT devices, we compare the z–$score_j$ with a predefined threshold θ as follows:

$$z\text{–}score_j = \begin{cases} \text{unknown} & \text{if } z\text{–}score_j < \theta \\ \text{known} & \text{if } z\text{–}score_j \geq \theta \end{cases} \tag{6.1}$$

Throughout the processing of the i-th window, each recognised IoT device's prediction, denoted by y_j, is added to the label set Y_i, which comprises labels corresponding to the feature vectors within X_i (Algorithm 1, line 14). If known, we will use its prediction as a label y_j, otherwise, a human in the loop is required for labelling, and this limitation is marked as future work. At the end of the ith-window, model M and z-score parameters are updated using (X_i, Y_i) to reflect the expanded knowledge as in Algorithm 1. The procedure will ensure that the model M continuously evolves and improves its predictive accuracy and adaptability based on the newly integrated labels and data insights. Moreover, it is efficient since it minimises storage requirements by only allowing M and z-score parameters to be stored.

Algorithm 1 OSL training

Input:
$\mathcal{M}$: the trained model.
X: the traffic features $X = \{x_1, x_2, \ldots, x_n\}$.
Y: the labels $Y = \{y_1, y_2, \ldots, y_n\}$.
Output:
$\mathcal{M}$: the updated trained model.
μ: The mean of the known distribution data $P_{\max}$.
σ: The standard deviation of the known distribution data $P_{\max}$.

1: **procedure** OSLTRAINING($\mathcal{M}, X, Y$)
2: $\mathcal{M} \leftarrow OSL(\mathcal{M}, X, Y)$;
3: $P \leftarrow \text{predict\_propa}(\mathcal{M}, X)$;
4: $P_{\max} \leftarrow \max(P)$;
5: $\mu \leftarrow \text{mean}(P_{\max})$;
6: $\sigma \leftarrow \text{sd}(P_{\max})$;
7: **return** $\mathcal{M}, \mu, \sigma$
8: **end procedure**

To implement our approach, we identify two feature extraction types: (i) the statistical features derived from a 5-minute window time of flow-based features of bidirectional traffic with payload entropy of $FwFV_{bi,entropy}\_5m$, and (ii) payload session of 784 bytes $PsFV\_784$, detailed in Chapter 5 as they show the best accuracy in IoT device fingerprinting. We use AMF as the OSL algorithm as detailed in Chapter 4. We use the UNSW IoT Traces dataset (12) as in Section 4.3, to extract $FwFV_{bi,entropy}\_5m$, and $PsFV\_784$.

Algorithm 2 Scalable IoT device fingerprinting

Input:
$\mathcal{S}_i$: stream traffic data from IoT devices of of the i-window of size w.
θ: threshold.
Output:
$\mathcal{M}$: the updated trained model.
μ: The updated mean of the known distribution data.
σ: The updated standard deviation of the known distribution data.
Output:
$\mathcal{M}$: the updated trained model.
μ: The updated mean of the known distribution data.
σ: The updated standard deviation of the known distribution data.

procedure SCALABLE_IOT_FINGERPRINTING($\mathcal{S}_i, \theta$)
  Load $\mathcal{M}, \mu, \sigma$;
  $X_i \leftarrow$ preprocessing ($\mathcal{S}_i$);
  $Y_i \leftarrow \varnothing$;
  for all x_j in X_i **do**
    $P_j \leftarrow$ predict_propa($\mathcal{M}, x_j$);
    $P_{j,\max} \leftarrow \max(P_j)$;
    $z\_score_j \leftarrow (P_{j,\max} - \mu)/\sigma$;
    if $z\_score_j \geq \theta$ **then**
      $y_j \leftarrow$ predict($\mathcal{M}, x_j$); ▷ known IoT device
    else
      $y_j \leftarrow$ prompt administrator for a label for unknown IoT device;
    end if
    $Y_i \leftarrow$ add(y_j);
  end for
  $\mathcal{M}, \mu, \sigma \leftarrow$ OSLTraining($\mathcal{M}, X_i, Y_i$);
  store with updated $\mathcal{M}, \mu, \sigma$;
end procedure

6.3 Unknown IoT Devices Detection

In this section, we evaluate the application of z-score measures for unknown IoT device detection detailed in Section 6.2. We define a small subset of feature vectors X_{s_d} of size s produced by IoT device d, where $X_{s_d} \subset X_i$ and $s \geq 1$. We compute the maximum probability $P_{s_d,max}$ for each subset. If the majority of the probabilities in X_{s_d} are less than a threshold θ, i.e., $majority(P_{s_d,max} \leq \theta)$ it is detected as an unknown device.

In this experiment, we assess the accuracy of detection of unknown IoT devices using the z-score measure, as defined in Equation 2.1, and how it is affected by different values of threshold θ and number of samples s from each device d in X_{s_d}. We test two values for θ: -2.5 and -3, indicating that the values are 2.5 and 3 standard deviations below the mean, respectively. These thresholds are typically considered significant for

determining an unknown IoT device (165; 166). For the number of samples in X_{s_d}, we test s: 1 sample and 3 samples in a batch. We evaluate our approach on $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$ with different values of (θ, s), and make a comparison with the unknown IoT device detection performances reported by Fan *et al.* for AutoIoT (25). In AutoIoT, the Kolmogorov-Smirnov test is applied to incoming traffic samples to compare their distribution against the traffic distribution of known devices. By doing this, AutoIoT can determine if the observed traffic is statistically different from what is expected for known devices, which may indicate the presence of an unknown IoT device. It needs $3 \sim 5$ samples to detect the presence of unknown IoT devices.

We adopt AutoIoT evaluation strategy, using 18 IoT devices in the UNSW IoT Traces dataset. The IoT devices are organised into different configurations as outlined in Table 6.1, with each configuration specifying which devices are classified as unknown. The indices of these devices within each configuration are listed in Table 6.2. In our experiments, we treat the devices in each specified configuration as unknown while considering the remaining devices as known. For example, in Configuration 1, device 0 and device 1 are unknown, and devices 2-17 as known. Each configuration is divided into four sets: the known IoT device features are split into $train_{known}$ and $test_{known}$, and the unknown IoT device features are split into $train_{unknown}$ and $test_{unknown}$. Like AutoIoT, we adopt the 30% holdout validation strategy, splitting the known and unknown features into training and test sets in a 7:3 ratio.

TABLE 6.1: Unknown IoT device configurations for the UNSW IoT Traces dataset

Configuration number	Unknown IoT Device Indices
1	0,1
2	2,3
3	4, 5
4	6, 7
5	8, 9
6	10,11
7	12,13
8	14,15
9	16,17
10	1, 13, 14
11	7, 15, 16
12	2, 10, 12
13	0, 3, 17
14	1, 6, 7, 14
15	8, 11, 12, 15
16	4, 10, 13, 16
17	2, 3, 9, 17

Table 6.3 shows the accuracy of $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$ by training an OSL model for each configuration with $train_{known}$ and evaluate it with $test_{known}$ and $test_{unknown}$. The results show $PsFV\_784$ outperform $FwFV_{bi,entropy}\_5m$ in the detection

TABLE 6.2: Device index number for the UNSW IoT Traces dataset

Index of device	IoT Device Name
0	Amazon Echo
1	Belkin Wemo Motion Sensor
2	Belkin Wemo Switch
3	Dropcam
4	HP Printer
5	iHome Power Plug
6	Insteon Camera wired
7	Light Bulbs LiFX Smart Bulb
8	Netatmo Weather Station
9	Netatmo Welcome
10	PIX-STAR Photo-Frame
11	Samsung SmartCam
12	Samsung Smart-Things
13	TP-Link Day Night Cloud Camera
14	TP-Link Smart Plug
15	Triby Speaker
16	Withings Aura Smart Sleep Sensor
17	Withings Smart Baby Monitor

of unknown IoT devices. In general, increasing the number of samples in both feature vector types increases the accuracy. In fact, $PsFV\_784$, the 3 sample batch optimises the accuracy to 100% in most configurations, where the $(-3, 3)$ shows slightly better performance than $(-2.5, 3)$, evidenced by Configuration 17. However, using 1 sample instead as $(-3, 1)$ and $(-2.5, 1)$ parameter configuration for $PsFV\_784$ and $FwFV_{bi,entropy}\_5m$, respectively, show negligible lower accuracy but offers advantages in reducing latency by a factor of 3. It can instantly detect incoming samples by minimising latency and memory usage without significantly impacting accuracy. Increasing the number of samples in a batch also shows its benefit for $FwFV_{bi,entropy}\_5m$ where the parameter setting $(-2.5, 3)$ generally yields higher accuracy across most configurations.

Table 6.4 shows in detail the accuracy for each unknown IoT device and the overall accuracy using 1 sample. In $FwFV_{bi,entropy}\_5m$, Configuration 2 shows a dramatic drop in the unknown accuracy across which is mainly caused by the high similarity of Dropcam with the known data, where most of the samples are above threshold showing high confidence. Similarly, Configuration 10 displays lower accuracy, which is influenced by the similarity of the TP-Link Smart Plug with known data. For $PsFV\_784$, the deficiency in Configuration 12 is attributed to the similarity between the Belkin Wemo Switch and the Belkin Wemo Motion Sensor. Despite these issues, $PsFV\_784$ demonstrates superior and high accuracy overall, making it a robust choice for IoT device detection.

In general, our approach with $PsFV\_784$ shows comparable results with AutoIoT, where they exhibit 100% accuracy across all the configurations. Their method requires

TABLE 6.3: Accuracy of unknown IoT device detection under different θ and s in a batch using two future extraction types on the UNSW IoT Traces dataset

Config	Accuracy (%) in (θ, s)							
	$FwFV_{bi,entropy\_}5m$				$PsFV\_784$			
	(-2.5,1)	(-2.5,3)	(-3,1)	(-3,3)	(-2.5,1)	(-2.5,3)	(-3,1)	(-3,3)
1	99.47	99.82	99.29	99.82	99.69	100	99.54	100
2	59.01	54.84	57.70	53.42	98.92	100	98.89	100
3	99.61	100	99.22	99.78	100	100	100	100
4	96.55	99.59	94.46	99.10	100	100	100	100
5	97.52	99.52	96.84	99.42	100	100	100	100
6	97.42	99.48	95.70	98.58	100	100	100	100
7	99.93	100	99.82	100	100	100	100	100
8	96.46	99.05	94.11	97.52	100	100	100	100
9	99.60	100	98.95	100	100	100	100	100
10	78.79	76.02	78.15	75.50	99.55	99.98	99.35	99.98
11	99.66	100	99.66	100	100	100	100	100
12	99.50	100	98.99	100	80.10	88.86	80.01	88.86
13	99.87	100	99.48	99.87	100	100	100	100
14	95.77	98.42	92.15	94.75	99.74	100	99.62	100
15	90.59	96.10	85.62	91.18	99.95	100	99.95	100
16	98.94	99.68	97.95	99.20	99.84	100	99.84	100
17	96.32	98.69	94.91	97.69	99.42	99.42	99.42	100

$1.5 \sim 2.5$ hours of traffic, representing $3 \sim 5$ samples, for each IoT device to achieve this level of performance. In contrast, our approach involves taking the first 784 bytes from each payload session. Our approach expresses no delays which only need to process a single payload session. Moreover, AutoIoT applies the Kolmogorov–Smirnov test which entails the availability of training data every time to compare it with the unknown data. Unlike AutoIoT, our method does not depend on complex resource requirements and only needs to maintain the μ and σ. This comparison highlights the time and resource efficiency of our approach.

6.4 Overall System

In this section, we evaluate the system presented in Figure 6.1 considering stream data, where samples are introduced periodically. We explore how our approach effectively manages and analyses the periodic streaming data, ensuring continuous, dynamic updates and immediate responsiveness to network changes. To the best of our knowledge, our research is the first to implement a scalable IoT device fingerprinting system that integrates periodic stream data with continuous model updates.

Our experiment aims to assess the accuracy of our scalable IoT fingerprinting approach in Algorithm 1 and Algorithm 2. We evaluate the accuracy of our method in

TABLE 6.4: Accuracy of unknown IoT device detection using $FwFV_{bi,entropy}\_5m$ with (-2.5, 1) parameter and $PsFV\_784$ with (-3, 1) parameter on the UNSW IoT Traces dataset

Config	$FwFV_{bi,entropy}\_5m$ **(%)**		$PsFV\_784$ (%)	
	Unknown	overall	Unknown	overall
1	99.47 - 100, 98.93	96.30	99.54 - 100, 99.50	98.00
2	59.01 - 99.94, 18.05	90.50	98.89 - 98.88, 100	96.88
3	99.61 - 99.94, 98.72	96.03	100 - 100, 100	96.79
4	96.55 - 95.98, 97.13	95.53	100 - 100, 100	96.50
5	97.52 - 95.90, 98.70	95.78	100 - 100, 100	97.97
6	97.42 - 94.82, 98.94	96.54	100 - 100, 100	97.21
7	99.93 - 100, 99.82	95.89	100 - 100, 100	97.05
8	96.46 - 91.82, 99.94	95.73	100 - 100, 100	96.79
9	99.60 - 99.18, 100	96.28	100 - 100, 100	96.80
10	78.79 - 99.41, 99.82, 32.88	93.45	99.35- 99.34, 100, 100	97.83
11	99.66 - 99.51, 99.82, 99.59	96.31	100 - 100, 100, 100	96.75
12	99.50 - 99.88, 97.97, 100	96.55	80.01 - 77.55, 100, 100	94.77
13	99.87 - 99.94, 100, 99.60	96.13	100 - 100, 100,100	97.05
14	95.77 - 99.47, 97.70, 99.75, 85.07	96.21	99.62 - 99.55, 100, 100, 100	98.15
15	90.59 - 85.07, 84.97, 100, 90.77	93.98	99.95 - 100, 100, 100, 94.87	98.17
16	98.94 - 99.76, 95.63, 99.91, 99.59	96.91	99.84 - 100, 100, 99.35, 100	97.14
17	96.30 - 99.88, 99.82, 98.05, 84.54	95.72	99.42 - 98.88, 100, 100, 100	97.30

identifying and classifying known IoT device types, as well as detecting unknown IoT devices across the entire dataset, using $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$ applied over a 1-hour window size. To detect the unknown IoT devices, we set the threshold θ to -2.5 and -3 for $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$, respectively, as they showed their significance in Section 6.3.

In the experiment, we use the UNSW IoT Traces dataset containing 23 IoT devices with 20 days of traffic data. We set a 1-hour window size, during which 14 devices were active and used for initial training (Algorithm 1), while the remaining 9 devices were designated as unknown devices for testing the detection performance and updating the model (Algorithm 2).

Figure 6.2 and Figure 6.3 show the accuracy of our approach using a 1-hour window of stream data from $FwFV_{bi,entropy}\_5m$ and $PsFV\_784$, respectively. They focus on the accuracy of identifying known devices and the detection of unknown IoT devices. The red cross indicates the accuracy of detecting unknown IoT devices, while the blue dots represent the accuracy of classifying known IoT devices. The accuracy of the model for $FwFV_{bi,entropy}\_5m$ in Figure 6.2 maintains a high level of accuracy for classifying known devices consistently with an average of 92.44% over the entire duration. However, the model fails to correctly identify unknown devices, especially at the early stage of the fingerprinting timeline. In contrast, the accuracy of the model for $PsFV\_784$ in Figure 6.3 archives an average of 94.49%. Despite fluctuations and a

significant dip early in the fingerprinting timeline, the model quickly recovers and stabilises, illustrating robust adaptability to changing conditions. Furthermore, the results highlight the capability to detect unknown devices by achieving 100% accuracy with only a single payload session per unknown device. This capability is essential for maintaining reliable operations in dynamic environments where new device types may frequently emerge. This indicates a highly effective mechanism within the system that can instantly detect unknown IoT devices and accurately classify known IoT device types.

Overall, the findings shed light on the novelty of our approach, particularly in providing a scalable IoT device fingerprinting with unknown IoT detection demonstrating high accuracy from a single payload session while requiring less memory to store samples. These distinctive characteristics not only set our approach apart from existing solutions but also showcase its potential for continuous learning and adaptation. By effectively handling dynamic IoT environments with such precision, our approach represents a significant advancement in the field, paving the way for more secure and adaptable IoT systems.

However, it is worth noting that these findings are based on tests conducted with a single dataset. While the results are promising, further research is necessary to validate these findings across diverse IoT datasets to ensure the robustness and generalisation of our approach.

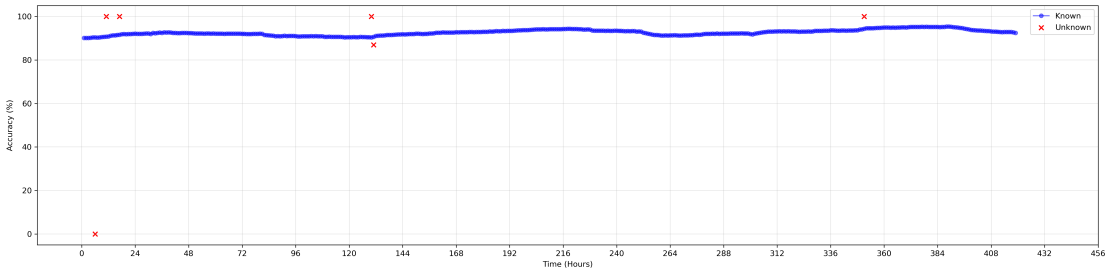


FIGURE 6.2: Accuracy of the scalable network fingerprinting with unknown IoT device detection approach on stream data scenario of $FwFV_{bi,entropy}\_5m$ with a 1-hour non-overlapping sliding window using UNSW IoT Traces dataset

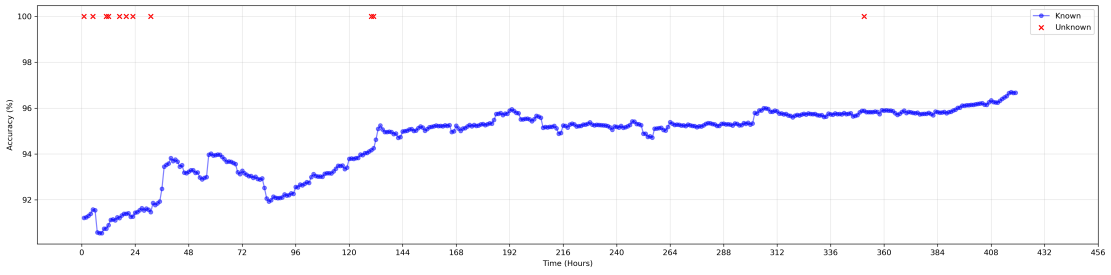


FIGURE 6.3: Accuracy of the scalable network fingerprinting with unknown IoT device detection approach on stream data scenario of $PsFV\_784$ with a 1-hour non-overlapping sliding window using UNSW IoT Traces dataset

6.5 Summary

Scalable approaches to IoT fingerprinting are essential to keep pace with the rapid development of new IoT devices. This scalability is crucial not only for handling the growing volume of devices but also for effectively detecting and integrating unknown IoT devices into the system. The novel approach we propose achieves scalability by using an OSL classifier and z-score measure for unknown IoT device detection, to reduce storage requirements and faster adaptation. The research question we address in this Chapter is how can we accurately detect unknown IoT devices in a scalable IoT device fingerprinting system while reducing memory consumption.

Our experiments evaluate our approach on two types of feature vectors, statistical features with payload entropy derived from a bidirectional flow of a 5-minute window and fixed-size payload sessions of 784 bytes. The detection of unknown IoT devices using the z-score measure demonstrates its best accuracy with payload sessions. It performs similarly to AutoIoT in detecting unknown IoT devices, where the z-score approach has the advancement of handling the resources efficiently. Moreover, we experiment with the application of the approach on periodic stream data by defining a non-overlapping sliding window. The payload session constantly showcases its advancement over other feature-based methods with high accuracy for classifying known IoT devices and detecting unknown IoT devices from a single payload session. It demonstrates the novelty of our approach in providing continuous learning and adaptation in dynamic IoT environments without memory exhaustion.

This research can be expanded by evaluating the approach's effectiveness on various IoT datasets. Moreover, it would be beneficial to test other OSL algorithms beyond the AMF. Investigating dynamic thresholding techniques could also enhance the detection accuracy of unknown IoT devices. Additionally, the use of automatic labelling could significantly improve scalability and reduce the need for manual intervention in model maintenance and updates.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Network fingerprinting is essential for identifying IoT devices, which in turn enhances IoT network security and supports forensic investigations. Machine learning techniques have been widely adopted to optimise the accuracy of this fingerprinting. However, the rapid proliferation of new IoT devices poses a significant challenge in maintaining scalability. Specifically, it becomes necessary to frequently update machine learning models to ensure they can recognise new devices effectively. While several approaches have been proposed to tackle this issue, they often face significant limitations, such as the need for substantial memory to store training data and a decline in accuracy when recognising older devices. These challenges highlight the need for more efficient and adaptable solutions in the realm of IoT fingerprinting.

In this Thesis, we introduce a novel network fingerprinting method for IoT devices that is both scalable and efficient. By employing online stream learning and fixed-size session payloads, our approach allows for continuous updates to the model without the need to store training data, addressing scalability challenges effectively. This ensures that the model remains up-to-date and maintains high recognition accuracy. Furthermore, our method includes a mechanism for identifying unknown IoT devices using z-score, which is significant for preserving network security as the IoT environment continues to grow and evolve rapidly.

7.2 Future Work

This work makes a significant contribution to scalable IoT device fingerprinting. The innovative scalable technique presented in this thesis has the potential for further enhancement. Key areas for improvement are outlined below

- expand our evaluation by utilising a broader range of datasets. This will enable us to further validate the robustness of our approach across different IoT environments and device types.
- investigate the integration of automated labelling techniques to enhance our scalable IoT device fingerprinting methodology. Automated labelling involves the system independently updating and refining device labels as new data is continuously streamed in, without the need for human intervention.
- address the challenges associated with hyperparameter tuning for OSL by developing adaptive methods that can dynamically adjust to concept drifts. These adaptive techniques allow periodically tuning hyperparameters in response to changes in the data stream. This will monitor the performance of the model continuously and make necessary adjustments to the hyperparameters in real time, ensuring that the model remains effective even as the underlying data distribution evolves. This approach will allow our scalable IoT device fingerprinting method to maintain high performance and adaptability, effectively responding to new devices and shifting network behaviours.
- explore the use of dynamic thresholding techniques for improving the detection of unknown IoT devices in the scalable IoT device fingerprinting system. While the current approach leverages the z-score for detecting unknown IoT devices, implementing a dynamic threshold could further enhance the model's adaptability and accuracy. Dynamic thresholding involves adjusting the detection threshold in real time based on the statistical properties of the incoming data stream. This method can better accommodate variations and concept drifts in the data, ensuring that the system remains sensitive to new and evolving device types.

We encourage other researchers to investigate the future directions outlined above.

Appendix A

Characterisation based on traffic features

This section includes lists of features proposed by studies for fingerprinting IoT devices using features retrieved from header field (Table A.1), header statistical features (Table A.2), payloads (Table A.3), statistical payload features (Table A.4), flow features (Table A.5), time interval features (Table A.6).

TABLE A.1: Header Field Features

Features		(2)	(18)	(11)	(10)	(1)	(12)	(9)	(99)	(101; 102)	(90)	(120)	(106)	(98)	(121)	(107)	(108)	(109)	(22)	(110)	(116)	(117)	(131)	(97)	(118)	(111)	(127)	(112)	(92)	(3)	(26)
packet	arrival time									✓																					
	direction				✓					✓			✓							✓											
	raw byte																				✓										
Data link layer	mac_src															✓															
	mac_dst															✓															✓
	EtherType/Len																	✓								✓					
IP header network layer	version																	✓					✓								✓
	IHL			✓						✓								✓					✓								
	service_type (tos)			✓														✓													
	dsfield_dscp			✓																											
	pack_len	✓											✓			✓	✓	✓		✓		✓								✓	✓
	identification			✓														✓													
	flags(DF,MF)			✓														✓													
	Flag_DF			✓								✓											✓			✓					
	Flag_MF																	✓													
	Frag_offset																	✓													
	ttl			✓						✓						✓		✓					✓								
	protocol			✓						✓				✓		✓	✓	✓					✓		✓		✓				✓
	src_ip															✓															
	des_ip	✓														✓										✓					✓
	opt																	✓					✓								
	opt_freq																✓														
	opt_padding	✓									✓																				
	opt_routerAlert	✓									✓																				
External IP	remote_ip_prefix																											✓			
	remote_ip_host																								✓			✓	✓		
TCP	port			✓										✓																	
	src_port	✓														✓	✓	✓					✓			✓			✓		
	dst_port	✓		✓												✓	✓	✓					✓		✓	✓	✓			✓	✓
	Seq_num			✓														✓													
	ack_num			✓														✓													
	offset									✓								✓													
	flags			✓						✓								✓					✓								
	flag_URG																	✓													
	flag_ACK																	✓													
	flag_PSH			✓														✓													
	flag_RST																	✓													
	flag_SYN																	✓													
	flag_FIN																	✓													
	flag_ECE																	✓													
	Reserved																	✓													
	flag_CWR																	✓													
	window len			✓	✓					✓	✓	✓				✓		✓					✓			✓					
	window len_val			✓																											
	window len_scale factor			✓																											
	checksum																	✓													
	opt									✓								✓					✓			✓					
	opt_max_seg											✓																			
	opt_timestamp			✓																											
	timestamp.tsval			✓																											
	timestamp.tsecr			✓																											
	opt_len			✓																											
	hdr_len			✓																											
	payload len													✓														✓			

Continue on the next page

TABLE A.1: Header Field Features (Cont.)

Features		(2)	(18)	(11)	(10)	(1)	(12)	(9)	(99)	(101; 102)	(90)	(120)	(106)	(98)	(121)	(107)	(108)	(109)	(22)	(110)	(116)	(117)	(131)	(97)	(118)	(111)	(127)	(112)	(92)	(92)
UDP	port									✓																				
	src_port			✓														✓												
	dst_port			✓														✓										✓		
	payload len																	✓											✓	
	checksum			✓														✓												
Port Class	sport class																	✓												
	dport class																	✓												
External port	port																								✓				✓	
	set_ports						✓		✓																					
protocol									✓																					
Data Link layer protocol	ARP	✓						✓									✓													
	LLC	✓						✓									✓													
Network layer protocol	IP	✓						✓																						
	IPv6							✓																						
	ICMP	✓						✓			✓						✓		✓											
	ICMPv6	✓						✓			✓						✓													
	IGMP							✓									✓													
	EAPoL	✓									✓						✓													
	ARP																		✓											
	LLC																													
Transport layer protocol	TCP	✓						✓		✓							✓													
	UDP	✓						✓		✓							✓													
Application layer protocol	HTTP	✓						✓		✓							✓	✓	✓				✓	✓						
	HTTPs	✓						✓															✓							
	DHCP	✓						✓									✓						✓							
	SSDP	✓									✓						✓	✓	✓											
	DNS	✓						✓			✓						✓	✓	✓				✓	✓						
	mDNS	✓						✓			✓						✓	✓												
	NTP	✓						✓			✓						✓	✓	✓					✓						
	SMB							✓									✓													
	TLS																✓		✓					✓						
	SSL										✓																			
	BOOTP	✓									✓							✓												
	BOOTP client																	✓												
	ANTLR																	✓												
	FTP																						✓							
	AFP							✓																						
	SSH							✓																						
	tcp/others																							✓						
	udp/others																							✓						
BOOTP msg	op code																	✓												
	Htype (hw type)																	✓												
	Hlen (hw add len)																	✓												
	Hops																	✓												
	xid																	✓												
	secs																	✓												
	flags																	✓												
	SName																	✓												
	file																	✓												
	options																	✓												

Continue on the next page

TABLE A.1: Header Field Features (Cont.)

Features	(2)	(18)	(11)	(10)	(1)	(12)	(9)	(99)	(101; 102)	(90)	(120)	(106)	(98)	(121)	(107)	(108)	(109)	(22)	(110)	(116)	(117)	(131)	(97)	(118)	(111)	(127)	(112)	(92)	(3)
DNS header																													
id																	✓												
qr																	✓												
qr_class			✓														✓												
opcode																	✓												
aa																	✓												
tc																	✓												
rd																	✓												
ra																	✓												
z																	✓												
ad*																	✓												
cd																	✓												
rcode			✓														✓												
qd_count																	✓												
an_count																	✓												
ar_count																	✓												
ns_count																	✓												
flags			✓																										
resp_len			✓																										
opt_list (first 8)		✓																											
opt																													
param_req_list (first 10)		✓																											
ICMP																													
type																	✓												
type_ptr																	✓												
code																	✓												
checksum			✓														✓												
id																	✓												
seq_num																	✓												
data																	✓												
nexthopmtu																	✓												
ts_ori																	✓												
ts_rx																	✓												
ts_tx																	✓												
reserved																	✓												
unused																	✓												
EAPOL																													
version																	✓												
type																	✓												
LLC																													
dsap																	✓												
ssap																	✓												
ctrl																	✓												

TABLE A.2: Header Statistic Features

Features			(14)	(10)	(1)	(12)	(9)	(21; 124)	(120)	(128; 129)	(98)	(25)	(92)	(121)	(22)	(122)	(19)	(131)	(123)	(99)	(125)	(127)	(3)
Packet	Size	mean								✓								✓					
		max																✓		✓		✓	
		min																✓		✓		✓	
		avg													✓					✓			
		var								✓													
		sd																					
		magnitude								✓													
		radius								✓													
		covariance								✓													
		correlation								✓													
		coefficient								✓													
		fft																✓					
		q1																✓					
		q3																✓					
		count									✓												
		sum																				✓	
		skewness																					
		kurtosis																					
	Count	#pkt						✓		✓													
		#sent_pack						✓			✓												
		#rec_pack						✓			✓												
	Remote pkt	#in_pkt						✓															
		#in_byte						✓															
		#out_pkt						✓															
		#out_byte						✓															
	Local pack	#in_pkt						✓															
		#in_byte						✓															
	User pkt	avg_pkt_len	✓																				
		peak_pkt_len	✓																				
		#pkt	✓																				
	Ctrl pkt	avg_pkt_len	✓																				
		peak_pkt_len	✓																				
		#pkt	✓																				
Ethernet layer	Packet size	max					✓																
		min					✓																
		mean					✓	✓															
		var					✓	✓															
		median					✓	✓															
	src2othes	iqr					✓	✓															
		q1					✓																
		iqr					✓																
	Packet size	max					✓																
		min					✓																
		mean					✓																
		var					✓																
		median					✓																
		iqr					✓	✓															
		q1					✓	✓															
		iqr					✓																
Data link protocol		#arp_pkt															✓						

Continue on the next page

TABLE A.2: Header Statistic Features (Cont.)

Features			(14)	(10)	(1)	(12)	(9)	(21; 124)	(120)	(128; 129)	(98)	(25)	(92)	(121)	(22)	(122)	(19)	(131)	(123)	(99)	(125)	(127)	(3)
Network layer	version	#IPv4										✓											
		#IPv6										✓											
	IHL	min		✓																			
		max		✓				✓															
		mean		✓				✓															
		median		✓				✓															
		var		✓				✓															
		q1		✓				✓															
		q3		✓				✓															
		iqr		✓																			
	Packet size	min		✓								✓	✓										
		max		✓								✓	✓										
		mean		✓								✓	✓			✓	✓				✓		
		median		✓												✓					✓		
		avg											✓										
		sd											✓			✓	✓						
		var		✓								✓				✓							
		skewness										✓											
		kurtiosis										✓				✓							
		q1		✓								✓											
		q2										✓											
		q3		✓								✓											
		iqr		✓																			
		min		✓																			
		max		✓																			
		mean		✓																			
		median		✓																			
		var		✓																			
		q1		✓																			
		q3		✓																			
	Packet size (src2others)	iqr		✓																			
		mode														✓					✓		
		count														✓							
		sum														✓							
		entopy										✓											
	Flags	df_ratio										✓											
		#flag_DF																				✓	
		#flag_MF																				✓	
	TTL	min										✓											
		max										✓											
		avg		✓								✓											

Continue on the next page

TABLE A.2: Header Statistic Features (Cont.)

Features			(14)	(10)	(1)	(12)	(9)	(21; 124)	(120)	(128; 129)	(98)	(25)	(92)	(121)	(22)	(122)	(19)	(131)	(123)	(99)	(125)	(127)	(3)
Network layer	#IP protocol	count																	✓				
		avg																	✓				
		mean																	✓				
		min																	✓				
		max																	✓				
		std																	✓				
		median																	✓				
		skew																	✓				
		trend																	✓				
		entropy																	✓				
		unique																	✓				
		mode																	✓				
	IP address	count																	✓				
		avg																	✓				
		mean																	✓				
		min																	✓				
		max																	✓				
		std																	✓				
		median																	✓				
		skew																	✓				
		trend																	✓				
		entropy																	✓				
		unique																	✓				
		mode																	✓				
		#dst_ip		✓																			
		#ip_endpoint														✓							
	Protocols	#ip_pkt							✓														
		#icmp_pkt							✓			✓											
Transport lyr	Protocols	#tcp_pkt							✓				✓				✓						
		#tcp_pkt (in/out)										✓											
		#udp_pkt							✓			✓					✓						
	Remote ports	#ports														✓							
TCP flow	Ports	#src_port										✓				✓							
		#dst_port		✓																			
		#src_port_bin										✓											
		#dst_port_bin										✓				✓							
		#server_port_bag							✓														
		#unique_port				✓																	
		#ports																	✓				
		avg																	✓				
		mean																	✓				
		min																	✓				
		max																	✓				
		std																	✓				
		median																	✓				
		skew																	✓				
		trend																	✓				
		entropy																	✓				
		unique																	✓				
		mode																	✓				

Continue on the next page

TABLE A.2: Header Statistic Features (Cont.)

Features			(14)	(10)	(1)	(12)	(9)	(21, 124)	(120)	(128, 129)	(98)	(25)	(92)	(121)	(22)	(122)	(19)	(131)	(123)	(99)	(125)	(127)	(3)
TCP flow	Flags	#flag_URG																				✓	
		#flag_ACK																				✓	
		#flag_PSH																				✓	
		#flag_RST																				✓	
		#flag_SYN																				✓	
		#flag_FIN																				✓	
	Window len	min										✓											
		max										✓											
		avg										✓											
	Payload len	sum														✓							
		mean														✓							
		median														✓							
		mode														✓							
		var														✓							
		sd														✓							
		kurtiosis														✓							
UDP flow	Payload len	sum														✓							
		mean														✓							
		median														✓							
		mode														✓							
		var														✓							
		sd														✓							
		kurtiosis														✓							
TLS/SSL flow	Count	#pkt			✓												✓						
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
	other fields	unique_CS												✓									
		#ciphersuite_bag							✓														
DNS/mDNS flow	Count	#pkt			✓				✓			✓					✓						
		#dns_byte																					
		#in_dns_pkt						✓															
		#in_dns_byte						✓															
		#out_dns_pkt						✓															
		#out_dns_byte						✓															
		#mdns_pkt															✓						
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
	Qyr	#dns_error			✓																		
		#IN_class_qry			✓																		
NTP flow	Count	#pkt							✓			✓					✓						
		#in_pkt						✓															
		#in_byte						✓															
		#out_pkt						✓															
		#out_byte						✓															
Http/Https flow	Count	#http_pkt			✓				✓								✓						
		#https_pkt																					
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
	other field	#http_rcode			✓																		
		#http_method			✓																		

Continue on the next page

TABLE A.2: Header Statistic Features (Cont.)

Features			(14)	(10)	(1)	(12)	(9)	(21; 124)	(120)	(128; 129)	(98)	(25)	(92)	(121)	(22)	(122)	(19)	(131)	(123)	(99)	(125)	(127)	(3)
SSDP flow	Count	#pkt			✓							✓					✓						
		#out _pkt																					
		#out_byte																					
		#outbyte						✓															
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
BOOTP	Count	#pkt			✓												✓						
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
QUIC flow	Count	#pkt			✓												✓						
		#pkt_type			✓																		
	Packet size	max			✓																		
		mean			✓																		
STUN flow	Count	#pkt			✓												✓						
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
MQTT flow	Count	#pkt			✓																		
		#pkt_type			✓																		
	Packet size	min			✓																		
		max			✓																		
		mean			✓																		
Other flows	App_proto	#app_proto		✓																			✓
	TLNET	#tlnet_pkt																					
	DHCP	#dhcp_pkt							✓			✓											
	SMTP	#smtp_pkt																					
	FTP	#ftp_pkt																					

TABLE A.3: Packet Payload Features

Features		(2)	(3)	(12)	(121)	(109)	(113)	(111)	(99)	(100)	(117)
raw_data	payload (0/1)	✓									
payload	plain									✓	
	bytes					✓	✓				
app_layer_mac	vendor_name								✓		
DNS flow	unique_dns_queries		✓		✓						
	dns_names						✓				
mDns_req	dev_local_name								✓		
	service_name								✓		
	service type								✓		
DHCP	dev_name								✓		
HTTP	dev_os								✓		
	model								✓		
	types								✓		
TLS/SSL	handshake_cs		✓	✓							
	extensions		✓								
	public_key		✓								
	cipher_suite				✓						
	msg_type										✓
XML (UPnP_msg)	vendor_name								✓		
	model								✓		
	friendly_name								✓		
	type								✓		

TABLE A.4: Payload Statistic Features

	Features	(3)	(11)	(10)	(1)	(90)	(120)	(25)	(121)	(109)	(97)	(104)
Payload stat	payload_size					✓						✓
	payload entropy					✓	✓			✓		
	#pkt_less_60B_payload										✓	
	#pkt_more_220B_payload										✓	
	#pkt_nonEmpty_payload										✓	
	payload_size_byte										✓	
	payload_size_first_nonEmpty										✓	
	sd_paload_size										✓	
	max_paload_size										✓	
	Pyaload_byte_freq	✓										
TCP	tcp/udp_payload_len			✓								
	tcp_payload_offset			✓								
TLS/SSL	tls_time_inerval							✓				
	#tls_handshake							✓				
	#ciphersuite_bag						✓					
DNS	#top10_dns_name							✓				
	#top_dns_name_bag						✓					
	#dns_qry				✓		✓	✓				
	mode_dns_qry				✓							
	entropy							✓				
	unique_DNS								✓			
	qry_name_len		✓									

TABLE A.5: Flow Statistics features

	Features	(11)	(10)	(1)	(12)	(9)	(23)	(120)	(25)	(92)	(121)	(22)	(19)	(97)	(99)	(125)	(127)	(116)	(109)	(126)	(3)
Traffic flow	pkt_num (in/out)								✓												
	pkt.Byte (in/out)								✓									✓			✓
	pkt_num/byte (in/out)								✓												✓
	flow_rate				✓			✓	✓											✓	
	mean_flow_rate					✓															
	sd_flow_rate					✓															
	flow_volume				✓																
	#pkt		✓	✓				✓					✓	✓	✓		✓				
	peak_pkt_rate																				✓
	mean_pkt_rate																				✓
TCP	#byte/t													✓		✓					
	#bytes (in/out)									✓											
	#bytes_pkt/sec (in/out)									✓											
	ratio_byte (in/out)									✓											
	ratio_pkt									✓											
	byte/t (in/out)									✓		✓									
	pkt/t									✓		✓									
	tcp.stream	✓																			
	tcp.analysis.acks frame	✓																			
	BoW																				
UDP	udp.stream	✓																			
TLS/SSL	#tls/tcp_keep_alive			✓																	
Application protocol	#http/tcp_keep_alive			✓																	
	dns_len																		✓		
flow	icmp_len																		✓		
Periodic flow	#periodic_flows					✓															
	#periodic_flows (protocol $\leqslant$ layer 4)					✓															
	mean_period_per_flow					✓															
	sd_period_per_flow					✓															
	max_period_per_flow						✓														
	#flows_of_one_period					✓															
	#flows_of_multi_period					✓															
	#flows_of_stat_src_port					✓															
	mean_frequency_srs_port_change					✓															
	sd_frequency_srs_port_change					✓															
Period accuracy	#periods_inferred_all_sub-captures					✓															
	mean_period_inference_success					✓															
	sd_period_inference_success					✓															
Period duration	#periods_in_window					✓															
Period stability (count periods)	#mean(r)_in_window					✓															
	#sd(r)_in_window					✓															
	#mean(m)_in_window					✓															
	#sd(m)_in_window					✓															
	#mean(m)−mean(r)_in_window					✓															
Burstiness	#element_z-score_exceed_threshold						✓														

TABLE A.6: Time Interval Features

Features			(10)	(1)	(12)	(9)	(21; 124)	(103)	(128; 129)	(120)	(25)	(92)	(121)	(107)	(108)	(122)	(131)	(97)	(99)	(127)	(3)
Traffic flow	IAT	IAT				✓		✓						✓	✓						
		min	✓								✓										
		max	✓								✓						✓				
		mean	✓								✓					✓	✓				
		var	✓								✓					✓	✓				
		median									✓										
		sd														✓		✓			
		avg																✓	✓		
		skewness									✓										
		kurtiosis									✓					✓					
		q1	✓								✓										
		q2									✓										
		q3	✓								✓										
		iqr	✓																		
		entropy									✓										
		Mode														✓					
	packet jitter	mean							✓												
		variance							✓												
		num					✓		✓												
	other time stat	duration			✓					✓	✓									✓	
		sleep_time			✓					✓			✓								
		active_time		✓																	
		mean_active_time																			✓
		sd_active_time																			✓
		idel_time																			
		10_highest_fft	✓																		
TCP flow	IAT-10pkt (in/out)	min										✓									
		max																			
		mean														✓					
		sd														✓					
	IAT	var														✓					
		kurtiosis														✓					
		Mode														✓					
	flow duration											✓									
UDP flow	IAT	min																			
		max																			
		mean														✓					
		sd														✓					
		var														✓					
		kurtiosis														✓					
		Mode														✓					
	flow duration																				

TABLE A.6: Time Interval Features (Cont.)

Features			(10)	(1)	(12)	(9)	(21; 124)	(103)	(128; 129)	(120)	(25)	(92)	(121)	(107)	(108)	(122)	(131)	(97)	(99)	(127)	(3)
TLS/SSL	flow duration	mean		✓																	
		min		✓																	
		max		✓																	
NTP flow	IAT	LAT			✓								✓								
		min									✓										
		max									✓										
	flow duration	avg								✓	✓										
		min		✓																	
		max		✓																	
		avg		✓																	
DNS flow	IAT	IAT			✓								✓								
		min									✓										
		max									✓										
		avg		✓																	
	DNS qyr ResTime	min		✓						✓											
		max		✓							✓										
		mean		✓																	
Other flows	http flow duration	min		✓																	
		max		✓																	
		avg		✓																	
	SSDP flow duration	min		✓																	
		max		✓																	
		avg		✓																	
	QUIC flow duration	min		✓																	
		max		✓																	
		avg		✓																	
	MQTT flow duration	min		✓																	
		max		✓																	
		avg		✓																	
	STUN flow duration	min		✓																	
		max		✓																	
		avg		✓																	
	BOOTP flow duration	min		✓																	
		max		✓																	
		avg		✓																	

References

[1] V. Thangavelu, D. Divakaran, R. Sairam, S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, 2019.

[2] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. 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)*, 2017.

[3] J. Sun, K. Sun, and C. Shenefiel, "Automated iot device fingerprinting through encrypted stream classification," in *Security and Privacy in Communication Networks*, 2019.

[4] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018.

[5] 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*, 2017.

[6] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: i see your smart home activities, even encrypted!," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[7] 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*, 2018.

[8] B. A. Desai, D. M. Divakaran, I. Nevat, G. W. Peter, and M. Gurusamy, "A feature-ranking framework for iot device classification," in *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, 2019.

[9] 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)*, 2019.

[10] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "Iot device identification via network-flow based fingerprinting and learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019.

[11] A. Aksoy and M. H. Gunes, "Automated iot device identification using network traffic," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019.

[12] 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*, 2019.

[13] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *2020 IEEE European Symposium on Security and Privacy (EuroS P)*, 2020.

[14] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018.

[15] S. Aneja, N. Aneja, and M. S. Islam, "Iot device fingerprint using deep learning," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 2018.

[16] J. Collins, M. Iorga, D. Cousin, and D. Chapman, "Passive encrypted iot device fingerprinting with persistent homology," in *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

[17] J. Kotak and Y. Elovici, "Iot device identification using deep learning," in *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, 2021.

[18] L. Yu, T. Liu, Z. Zhou, Y. Zhu, Q. Liu, and J. Tan, "Wdmti: Wireless device manufacturer and type identification using hierarchical dirichlet process," in *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2018.

[19] Y. Sun, J. Liu, A. K. Bashir, U. Tariq, W. Liu, K. Chen, and M. D. Alshehri, "E-cis: Edge-based classifier identification scheme in green sustainable iot smart city," *Sustainable Cities and Society*, 2021.

[20] J. Bao, B. Hamdaoui, and W.-K. Wong, "Iot device type identification using hybrid deep learning approach for increased iot security," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020.

[21] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Inferring iot device types from network behavior using unsupervised clustering," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019.

[22] R. Du and S. Li, "Identification of iot devices based on feature vector split," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021.

[23] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, 2019.

[24] L. S. Vailshery, "Number of internet of things (iot) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030," 2023.

[25] L. Fan, L. He, Y. Wu, S. Zhang, Z. Wang, J. Li, J. Yang, C. Xiang, and X. Ma, "Autoiot: Automatically updated iot device identification with semi-supervised learning," *IEEE Transactions on Mobile Computing*, 2022.

[26] J. Wang, J. Zhong, and J. Li, "Iot-portrait: Automatically identifying iot devices via transformer with incremental learning," *Future Internet*, 2023.

[27] T. N. Alyahya, L. Aniello, and V. Sassone, "Scanef-iot: Scalable network fingerprinting for iot devices," The 19th International Conference on Availability, Reliability and Security (ARES 2024), 2024.

[28] "That 'internet of things' thing: In the real world, things matter more than ideas."

[29] A. V. Dastjerdi and R. Buyya, *Internet of things: Principles and paradigms*. Morgan Kaufmann Publishers, 2016.

[30] S. Balaji, K. Nathani, and R. Santhakumar, "Iot technology, applications and challenges: a contemporary survey," *Wireless personal communications*, vol. 108, pp. 363–388, 2019.

[31] L. S. Vailshery, "Number of internet of things (iot) connected devices worldwide from 2019 to 2030, by vertical," 2023.

[32] "Statista technology market insight."

[33] C. Sobin, "A survey on architecture, protocols and challenges in iot," *Wireless Personal Communications*, vol. 112, no. 3, pp. 1383–1429, 2020.

[34] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang, "Iot device fingerprinting for relieving pressure in the access control," in *Proceedings of the ACM Turing Celebration Conference China*, ACM TURC 19, Association for Computing Machinery, 2019.

[35] N. Shridhar and N. V, "A survey on security of iot devices," *International Journal of Engineering Science Invention (IJESI)*, vol. 7, pp. 69–72, 2018.

[36] "Chapter 4 - layer 3: The network layer," in *Hack the Stack* (M. Gregg, ed.), pp. 103 – 150, Burlington: Syngress, 2006.

[37] "Nmap: The network mapper—free security scanner." Accessed: 2020-04-01.

[38] "X probe - active os fingerprinting tool." Accessed: 2020-04-01.

[39] "Sinfp – a new approach to os fingerprinting." Accessed: 2020-04-01.

[40] "scanrand – download stateless tcp scanner with syn cookies." Accessed: 2020-04-01.

[41] X. Wang, J. Huang, and C. Qi, "Fdi: A fast iot device identification approach," in *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, (New York, NY, USA), Association for Computing Machinery, 2021.

[42] L. Hyun-seong, L. Jae-gwang, L. Jae-pil, and L. Jae-kwang, "Design of automatic identification gateway system for different iot devices and services," in *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018.

[43] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Can we classify an iot device using tcp port scan?," in *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, pp. 1–4, Dec 2018.

[44] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, pp. 318 – 327, 2019.

[45] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems.," in *NDSS*, 2016.

[46] Y. Aun, Y.-M. Khaw, and M.-L. Gan, "A holistic iot device classification approach through spatial & temporal behaviors modelling," *Telecommun. Syst.*, vol. 79, no. 4, 2022.

[47] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, 2010.

[48] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, pp. 93–108, April 2005.

[49] H. Noguchi, T. Demizu, N. Hoshikawa, M. Kataoka, and Y. Yamato, "Autonomous device identification architecture for internet of things," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pp. 407–411, 2018.

[50] H. Noguchi, M. Kataoka, and Y. Yamato, "Device identification based on communication analysis for the internet of things," *IEEE Access*, vol. 7, pp. 52903–52912, 2019.

[51] F. Le, J. Ortiz, D. Verma, and D. Kandlur, *Policy-Based Identification of IoT Devices' Vendor and Type by DNS Traffic Analysis*, pp. 180–201. Springer International Publishing, 2019.

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

[53] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1048–1077, 2021.

[54] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning–based approaches for internet traffic classification," *Annals of Telecommunications*, pp. 1–38, 2020.

[55] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[56] A. Géron, *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

[57] I. El Naqa and M. J. Murphy, *What Is Machine Learning?*, pp. 3–11. Springer International Publishing, 2015.

[58] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[59] D. C. Francesca Lazzeri, C.J. Gronlund and J.Martens, "How to select algorithms for azure machine learning," 2020.

[60] B. Yegnanarayana, *ARTIFICIAL NEURAL NETWORKS*. PHI Learning, 2009.

[61] S. Lek and Y. Park, "Artificial neural networks," in *Encyclopedia of Ecology* (S. E. Jørgensen and B. D. Fath, eds.), pp. 237–245, Oxford: Academic Press, 2008.

[62] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996.

[63] W. Zhu, Y. Ma, Y. Zhou, M. Benton, and J. Romagnoli, "Deep learning based soft sensor and its application on a pyrolysis reactor for compositions predictions of gas phase components," in *13th International Symposium on Process Systems Engineering (PSE 2018)* (M. R. Eden, M. G. Ierapetritou, and G. P. Towler, eds.), vol. 44 of *Computer Aided Chemical Engineering*, pp. 2245–2250, Elsevier, 2018.

[64] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[65] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. a. Gama, "Machine learning for streaming data: state of the art, challenges, and opportunities," *SIGKDD Explor. Newsl.*, 2019.

[66] H. Du, Y. Zhang, K. Gang, L. Zhang, and Y.-C. Chen, "Online ensemble learning algorithm for imbalanced data stream," *Applied Soft Computing*, vol. 107, p. 107378, 2021.

[67] "Metrics to evaluate your machine learning algorithm."

[68] "20 popular machine learning metrics. part 1: Classification & regression evaluation metrics."

[69] V. Aggarwal, V. Gupta, P. Singh, K. Sharma, and N. Sharma, "Detection of spatial outlier by using improved z-score test," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 788–790, IEEE, 2019.

[70] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for internet of things," *International Journal of Machine Learning and Cybernetics*, vol. 9, pp. 1399–1417, Aug 2018.

[71] K. Zhao and L. Ge, "A survey on the internet of things security," in *2013 Ninth International Conference on Computational Intelligence and Security*, pp. 663–667, 2013.

[72] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, 2017.

[73] 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.

[74] 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.

[75] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *Journal of Network and Computer Applications*, vol. 42, pp. 120 – 134, 2014.

[76] I. Ud Din, M. Guizani, B. Kim, S. Hassan, and M. Khurram Khan, "Trust management techniques for the internet of things: A survey," *IEEE Access*, vol. 7, pp. 29763–29787, 2019.

[77] M. A. Ferrag, L. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: A comprehensive survey," *Security and Communication Networks*, vol. 2017, 09 2017.

[78] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[79] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[80] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[81] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020.

[82] A. Sagu and N. S. Gill, "Securing iot environment using machine learning techniques," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, pp. 870–873, 2020.

[83] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in iot networks: A survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020.

[84] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 1048–1077, 2021.

[85] P. Yadav, A. Feraudo, B. Arief, S. F. Shahandashti, and V. G. Vassilakis, "Position paper: A systematic framework for categorising iot device fingerprinting mechanisms," (New York, NY, USA), p. 62–68, Association for Computing Machinery, 2020.

[86] Y. Yue, S. Li, P. Legg, and F. Li, "Deep learning-based security behaviour analysis in iot environments: A survey," *Security and Communication Networks*, vol. 2021, pp. 1–13, 01 2021.

[87] M. Safi, S. Dadkhah, F. Shoeleh, H. Mahdikhani, H. Molyneaux, and A. A. Ghorbani, "A survey on iot profiling, fingerprinting, and identification," *ACM Trans. Internet Things*, 2022.

[88] M. Seliem, K. Elgazzar, and K. Khalil, "Towards privacy preserving iot environments: A survey," *Wireless Communications and Mobile Computing*, vol. 2018, p. 15, 11 2018.

[89] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity," in *Proceedings of the 2019 ACM Symposium on SDN Research*, SOSR '19, (New York, NY, USA), p. 36–48, Association for Computing Machinery, 2019.

[90] R. Jiao, Z. Liu, L. Liu, C. Ge, and G. Hancke, "Multi-level iot device identification," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 538–547, 2021.

[91] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-Level Signatures for Smart Home Devices," *Proceedings of the 2020 Network and Distributed System Security (NDSS) Symposium*, February 2020.

[92] M. Mainuddin, Z. Duan, Y. Dong, S. Salman, and T. Taami, "Iot device identification based on network traffic characteristics," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 6067–6072, 2022.

[93] B. Chakraborty, D. M. Divakaran, I. Nevat, G. W. Peters, and M. Gurusamy, "Cost-aware feature selection for iot device classification," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11052–11064, 2021.

[94] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[95] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1362–1380, 2019.

[96] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*, 2019.

[97] A. Pashamokhtari, N. Okui, Y. Miyake, M. Nakahara, and H. H. Gharakheili, "Inferring connected iot devices from ipfix records in residential isp networks," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 57–64, 2021.

[98] N. Okui, M. Nakahara, Y. Miyake, and A. Kubota, "Identification of an iot device model in the home domain using ipfix records," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 583–592, 2022.

[99] N. Ammar, L. Noirie, and S. Tixeuil, "Network-protocol-based iot device identification," in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 204–209, 2019.

[100] S. Liu, X. Xu, Y. Zhang, and Y. Wang, "Autonomous anti - interference identification of IoT device traffic based on convolutional neural network," in *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2022.

[101] X. Ma, J. Qu, J. Li, J. C. Lui, Z. Li, and X. Guan, "Pinpointing hidden iot devices via spatial-temporal traffic fingerprinting," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020.

[102] X. Ma, J. Qu, J. Li, J. C. S. Lui, Z. Li, W. Liu, and X. Guan, "Inferring hidden iot devices and user interactions via spatial-temporal traffic fingerprinting," *IEEE/ACM Transactions on Networking*, 2022.

[103] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. S. Bentley, and A. S. Uluagac, "Z-iot: Passive device-class fingerprinting of zigbee and z-wave iot devices," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2020.

[104] Y. Meng and J. Li, "Research on intelligent configuration method of mine iot communication resources based on data flow behavior," *IEEE Access*, vol. 8, pp. 172065–172075, 2020.

[105] B. Charyyev and M. H. Gunes, "Iot event classification based on network traffic," in *IEEE INFOCOM Workshops*, 2020.

[106] C. Kuzniar, M. Neves, V. Gurevich, and I. Haque, "Iot device fingerprinting on commodity switches," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2022.

[107] A. Hameed, J. Violos, and A. Leivadeas, "A deep learning approach for iot traffic multi-classification in a smart-city scenario," *IEEE Access*, vol. 10, pp. 21193–21210, 2022.

[108] Y. Zhang, B. Gong, and Q. Wang, "Bls-identification: A device fingerprint classification mechanism based on broad learning for internet of things," *Digital Communications and Networks*.

[109] 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.

[110] C. Duan, H. Gao, G. Song, J. Yang, and Z. Wang, "Byteiot: A practical iot device identification system based on packet length distribution," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1717–1728, 2022.

[111] A. Pashamokhtari, "Phd forum abstract: Dynamic inference on iot network traffic using programmable telemetry and machine learning," in *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 371–372, 2020.

[112] W. Jia, Y. Wang, Y. Lai, H. He, and R. Yin, "Fitic: A few-shot learning based iot traffic classification method," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–10, 2022.

[113] Z. Zhao, Y. Lai, Y. Wang, W. Jia, and H. He, "A few-shot learning based approach to iot traffic classification," *IEEE Communications Letters*, vol. 26, no. 3, pp. 537–541, 2022.

[114] G. Qing, H. Wang, L. Guo, and J. Yang, "Device type identification via network traffic and lightweight convolutional neural network for internet of things," *IEEE Access*, vol. 8, pp. 200219–200228, 2020.

[115] J. Kotak and Y. Elovici, "Iot device identification based on network communication analysis using deep learning," *Journal of Ambient Intelligence and Humanized Computing*, 2022.

[116] Y. Wang, X. Yun, Y. Zhang, C. Zhao, and X. Liu, "A multi-scale feature attention approach to network traffic classification and its model explanation," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 875–889, 2022.

[117] J. Yang, Y. Sang, Y. Zhang, P. Chang, and C. Peng, "Exploiting heterogeneous information for iot device identification using graph convolutional network," in *Collaborative Computing: Networking, Applications and Worksharing* (H. Gao and X. Wang, eds.), (Cham), pp. 575–591, Springer International Publishing, 2021.

[118] A. Pashamokhtari, N. Okui, Y. Miyake, M. Nakahara, and H. H. Gharakheili, "Combining stochastic and deterministic modeling of ipfix records to infer connected iot devices in residential isp networks," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[119] Z. He, J. Yin, Y. Wang, G. Gui, B. Adebisi, T. Ohtsuki, H. Gacanin, and H. Sari, "Edge device identification based on federated learning and network traffic feature engineering," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1898–1909, 2022.

[120] S. Zhang, Z. Wang, J. Yang, D. Bai, F. Li, Z. Li, J. Wu, and X. Liu, "Unsupervised iot fingerprinting method via variational auto-encoder and k-means," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.

[121] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "Iot network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 989–1008, 2022.

[122] F. Palmese and A. E. C. Redondi, "A framework for storage-accuracy optimization of iot forensic analysis," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 3779–3784, 2022.

[123] T. Hammaine and J. Kahles, "Clustering unknown iot devices in a 5g mobile network security context via machine learning," in *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–6, 2021.

[124] A. Sivanathan, H. Habibi Gharakheili, and V. Sivaraman, "Managing iot cyber-security using programmable telemetry and machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 60–74, 2020.

[125] A. J. Pinheiro, J. de M. Bezerra, C. A. Burgardt, and D. R. Campelo, "Identifying iot devices and events based on packet length from encrypted traffic," *Computer Communications*, vol. 144, pp. 8–17, 2019.

[126] H. Wu, X. Fan, G. Cheng, and X. Hu, "Identify iot devices from backbone networks using lightweight neural networks," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pp. 140–148, 2022.

[127] B. M. Xavier, R. Silva Guimarães, G. Comarela, and M. Martinello, "Map4: A pragmatic framework for in-network machine learning traffic classification," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4176–4188, 2022.

[128] Y. Luo, X. Chen, N. Ge, W. Feng, and J. Lu, "Transformer-based device type identification in heterogeneous iot traffic," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[129] Y. Luo, X. Chen, N. Ge, and J. Lu, "Deep learning based device classification method for safeguarding internet of things," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[130] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, and J. Yang, "An iot device identification method based on semi-supervised learning," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–7, 2020.

[131] Q. Chen, Y. Song, B. Jennings, F. Zhang, B. Xiao, and S. Gao, "Iot-id: Robust iot device identification based on feature drift adaptation," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[132] S. Krishnan, "xverse," 2019.

[133] J. Liu, Y. Sun, F. Xu, K. Yu, A. K. Bashir, and Z. Liu, "Iis: Intelligent identification scheme of massive iot devices," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1623–1626, 2021.

[134] S. Velliangiri, S. Alagumuthukrishnan, *et al.*, "A review of dimensionality reduction techniques for efficient computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019.

[135] F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, and X. Liu, "A survey on deep learning for human activity recognition," *ACM Comput. Surv.*, vol. 54, no. 8, 2021.

[136] "iot network traffic," 2020.

[137] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 559–564, 2017.

[138] Y. Ren, H. Li, M. Xu, G. Xiong, S. Zhang, H. Zhu, and L. Sun, "Joint classification of iot devices and relations in the internet with network traffic," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 560–565, 2022.

[139] Y. Imamura, N. Nakamura, T. Yao, S. Ata, and I. Oka, "A device identification method based on combination of multiple information," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–4, 2020.

[140] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, "DÏot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, 2019.

[141] I. Cvitić, D. Peraković, M. Periša, and B. Gupta, "Ensemble machine learning approach for classification of iot devices in smart home," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 11, pp. 3179–3202, 2021.

[142] G. M. van de Ven, Z. Li, and A. S. Tolias, "Class-incremental learning with generative classifiers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

[143] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[144] A. Parmisano, S. Garcia, and M. J. Erquiaga, "Stratosphere laboratory. a labeled dataset with malicious and benign iot network traffic.."

[145] B. Charyyev and M. H. Gunes, "Locality-sensitive iot network traffic fingerprinting for device identification," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1272–1281, 2021.

[146] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai, "A novel approach for detecting vulnerable iot devices connected behind a home nat," *Computers Security*, vol. 97, p. 101968, 2020.

[147] R. R. Chowdhury, S. Aneja, N. Aneja, and P. E. Abas, "Packet-level and ieee 802.11 mac frame-level network traffic traces data of the d-link iot devices," *Data in Brief*, vol. 37, p. 107208, 2021.

[148] "Iot-device-type-identification," 2018.

[149] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the wide project," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '00, USENIX Association, 2000.

[150] J. a. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, 2013.

[151] J. Mourtada, S. Gaïffas, and E. Scornet, "Amf: Aggregated mondrian forests for online learning," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2021.

[152] A. Bifet, R. Gavalda, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*. MIT press, 2023.

[153] Netresec, "Splitcap: A tool for network traffic analysis," 2010.

[154] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, pp. 1469–1495, 2017.

[155] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80, 2000.

[156] N. C. Oza and S. J. Russell, "Online bagging and boosting," in *International workshop on artificial intelligence and statistics*, pp. 229–236, PMLR, 2001.

[157] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: machine learning for streaming data in python," *Journal of Machine Learning Research*, vol. 22, no. 110, pp. 1–8, 2021.

[158] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian forests: Efficient online random forests," *Advances in neural information processing systems*, 2014.

[159] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[160] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[161] P. Biondi and the Scapy community, "Scapy," 2024.

[162] T. Lacombe, Y. S. Koh, G. Dobbie, and O. Wu, "A meta-learning approach for automated hyperparameter tuning in evolving data streams," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021.

[163] J. Wilson, S. Chaudhury, and B. Lall, "Homogeneous–heterogeneous hybrid ensemble for concept-drift adaptation," *Neurocomputing*, vol. 557, p. 126741, 2023.

[164] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, p. 20–29, jun 2004.

[165] A. S. Yaro, F. Maly, P. Prazak, and K. Malý, "Outlier detection performance of a modified z-score method in time-series rss observation with hybrid scale estimators," *IEEE Access*, vol. 12, pp. 12785–12796, 2024.

[166] A. S. Yaro, F. Maly, and P. Prazak, "Outlier detection in time-series receive signal strength observation using z-score method with s n scale estimator for indoor localization," *Applied Sciences*, vol. 13, no. 6, p. 3900, 2023.