R+R: IoT Device Identification Under Realistic Conditions

Chakshu Gupta
University of Twente
c.gupta@utwente.nl

Andreas Peter
Carl von Ossietzky Universität Oldenburg
andreas.peter@uni-oldenburg.de

Andrea Continella
University of Twente
a.continella@utwente.nl

Abstract—**Internet of Things (IoT) devices are ubiquitous, yet they often present security issues. The research community has invested substantial effort in designing automated methods for identifying these devices through passive network analysis—an essential step in security applications such as anomaly detection, traffic monitoring, and vulnerability scanning. However, despite the promising results reported in laboratory settings, the effectiveness of these methods under realistic conditions remains unclear. In this work, we systematically review the existing literature on IoT device identification by studying the approaches, features, and evaluation environments. We then design and implement a framework to reproduce and evaluate selected identification methods. We re-implement the selected methods and assess their performance, using our framework, under realistic environmental factors, such as non-IoT traffic, dynamic user activity, and unknown devices. Our study reveals several important insights. We demonstrate that the performances of current identification methods significantly decline under realistic conditions. Furthermore, we highlight these methods' inability to differentiate between known and unknown devices, raising concerns about their effectiveness in security applications such as anomaly detection. We conclude by providing actionable recommendations for future research.**

1. Introduction

Technological advancements have revolutionized our living spaces by connecting home appliances to the Internet. These devices, known as the Internet of Things (IoT), enable users to automate their device usage and personalize their home experiences, increasing their demand. According to predictions, the number of IoT devices is expected to reach around 25 billion by 2030, compared to the 10.07 billion devices in 2021 [1].

While IoT devices offer the convenience of remote control, they pose significant security and privacy risks [2]–[9], making it crucial to monitor their runtime activities to enhance security, e.g., by detecting anomalies [10], monitoring vulnerable devices [11], [12], and identifying potential data leaks [13]. To this end, device identification methods have become vital in security monitoring and threat detection [14], [15]. In the past years, extensive research has aimed to identify IoT devices through automated network traffic analysis [16]–[26]—with several works further leveraging identification techniques to distinguish between known and unknown devices [27]–[33]—predominantly focusing on monitoring local networks and observing traffic through the network router or gateway. However, despite the tremendous effort and advances, the applicability of existing methods to identify (un)known devices in real-world environments remains *unclear*.

In fact, despite reporting high accuracy, most prior studies lack comparative evaluations against existing methods. Variations in their evaluation environments—such as the number and types of devices and the duration of data collection—further hinder direct performance comparisons and *blur* the strengths and limitations of each approach. Prior works often use simplistic lab setups, unlike real smart homes, where IoT and non-IoT devices co-exist and face dynamic conditions like multiple users, (new) device additions, and malfunctions that may alter network behavior.

A few recent studies compare IoT device identification methods through surveys or experimental analysis [34]–[38]. Safi et al. [37] provide a taxonomy of IoT profiling methods like IoT device identification, fingerprinting, and anomaly detection. Whereas Jmila et al. [36] investigate methods for acquiring data for identifying IoT devices, with a focus on feature selection, machine learning algorithms, and different granularities for device identification. These papers highlight challenges such as limited labeled data, imbalanced datasets, and generalization issues in classification. However, they lack a detailed analysis of existing methods or their performance under realistic environmental factors, such as non-IoT traffic, dynamic user activity, and unknown devices.

Kumar et al. [34] conduct an empirical analysis of ML algorithms for IoT device classification based on accuracy, training time, and error rate. They employ network-level features to train the various ML models and compare their performances. However, they do not evaluate existing works or the effects of realistic conditions on their performance. Ahmed et al. [35] study the impact of various data collection methods on the performance of IoT device fingerprinting. Although they analyze the feasibility of an IoT fingerprinting technique, their study does not evaluate existing methods and the impact of different environments on them. Finally, Maali et al. [38] evaluate existing ML-based IoT device identification methods across different modes of operation, spatiotemporal variations, and traffic sampling to identify causes of performance degradation. They further examine the generalizability of different algorithms and the impact of feature selection on their performance. However, their study does not include real-world factors such as non-IoT traffic, unknown devices, and dynamic setups, and it does not explore the security aspects of the investigated methods.

In essence, *our community still lacks a comprehensive understanding* of the existing literature on IoT device identification methods, the environmental factors that influence their performance, and their applicability in security applications such as anomaly detection.

In this paper, we examine the impact of realistic environmental conditions on IoT device identification methods. We consider realistic settings as meeting the following conditions: (1) the network includes both IoT and non-IoT devices; (2) the network includes at least one mobile device to operate the IoT devices; (3) new, previously unseen devices are introduced into the network over time; (4) devices are plugged in and out over time; (5) the network includes a large number of devices—at least 15 [39]. We start by systematizing the existing literature to identify strengths and gaps, with a particular focus on evaluation environments. Then, we categorize prior works based on their approach and the types of features they employ. Through this process, we observe that, due to the challenges in replicating prior work and reproducing their evaluation settings, most studies do not compare their proposed methods with previous ones. To address this gap, we design and implement an open-source framework that facilitates the evaluation and comparison of IoT device identification methods. Our framework provides a consistent evaluation environment and generates standardized result reports, enabling direct comparison. Then, we re-implement and replicate six identification methods to evaluate empirically how environmental factors affect their performance. Furthermore, we investigate how effectively these methods can detect unknown devices, which is crucial for their security applicability.

Our results provide several important insights for our community. First, we observe that the presence of non-IoT devices—standard in real-world deployments, where IoT and non-IoT devices co-exist in the network—significantly affects the performance of existing identification methods. Similarly, contrary to preferred evaluation settings that rely on *static* datasets, device identification methods suffer when evaluated in realistic, *dynamic* environments, which are subject to (multiple) user activity and (new) devices being connected or disconnected. We observe that the performance of some models degrades as the number of devices increases, whereas others scale well. We also highlight that most existing methods fail to effectively identify unknown devices, often misidentifying them with high confidence. In conclusion, despite promising performance in lab settings, our community requires further research on IoT device identification for real-world applications. We provide actionable recommendations for our community.

In summary, we address the following research questions:

RQ1: How does noise from non-IoT devices affect the performance of IoT device identification methods?

RQ2: How does a dynamic environment, where the number of devices or users regularly changes, impact the performance of IoT device identification methods?

RQ3: How well does the performance of the models scale with an increasing number of IoT devices?

RQ4: How well do IoT device identification methods detect unknown devices?

In the spirit of open science, we release our code and dataset at https://github.com/utwente-scs/evaliot.

2. IoT Device Identification: Systematization

We review IoT device identification studies that focus on smart home environments, specifically those that use network traffic analysis to extract patterns or train models for identifying devices in unseen traffic. We exclude work targeting other environments or not relying on network traffic analysis.

We follow two strategies to search for relevant works. First, we perform a literature survey using the keywords: (“IoT Devices” OR “Internet of Things” OR “smart devices”) AND (“traffic modeling” OR “fingerprint” OR “device classification” OR “device identification”). For this literature survey, we mainly utilize Google Scholar [40] and DBLP [41] platforms. For the Google Scholar search, we add the keywords (“network traffic” OR “network flow” OR “network behavior”) to retrieve papers containing these terms anywhere in the text. With these search queries, we find $1,408$ relevant papers in DBLP and $1,940$ in Google Scholar. Second, we review all papers published in the A* conferences (USENIX, CCS, NDSS, and S&P) over the last 8 years related to IoT Security and find 3 relevant papers. Next, we manually inspect these papers and identify 36 that were within the scope of our research, out of which, we exclude 5 since they were either survey papers [36], [37] or studied different aspects of fingerprinting/classification, without presenting a new method [34], [35], [38]. Among the filtered studies, we exclude those that focus on identifying IoT device events rather than the devices themselves, e.g., turning on/off a bulb [42]–[44] since event identification is a different line of research that falls outside the scope of this paper. We also exclude the work by Yu et al. [16] as it focuses on identifying mobile devices in public settings and only tangentially touches on IoT devices. Eventually, through this process, we identify and select a total of 31 works within the scope of our research.

Reviewing the selected works, we identify the three main steps in the IoT device identification process—(i) extract relevant features from the network traffic generated by the target IoT devices, (ii) analyze the extracted features to train a model with ML or extract unique traffic patterns, (iii) use the trained model or extracted pattern to identify the target IoT device from unseen network traffic. Following these steps, we establish criteria to categorize the selected works systematically (Table 1) based on three key aspects: (a) approach, (b) features, and (c) evaluation environment.

TABLE 1: **Systematization of knowledge for IoT device identification.**
?: unclear/unspecified, ∼: partly, **DT**: Decision Tree, **MUD**: Manufacturer Usage Description, **RF**: Random Forest, **GB**: Gradient Boosting, **kNN**: K Nearest Neighbors, **PCA**: Principle Component Analysis, **LSTM**: Long Short-Term Memory, **RNN**: Recurrent Neural Network, **CNN**: Convolutional Neural Network, **MLP**: Multi-Layer Perceptron, **XGBoost**: Xtreme Gradient Boosting, **TF-IDF**: Term Frequency and Inverse document frequency, **MTL**: Multi-task Learning, **LR**: Logistic Regression, **MNB**: Multinomial Naive Bayes, **HDBSCAN**: Hierarchical Density-Based Spatial Clustering of Applications with Noise, **MV**: Majority Voting.

					Feature Types								**Evaluation**								
Approach		**Ref**	**Year**	**Algorithm**	**IP**	**Transport**	**Application**	**Traffic Stats**	**Flow Stats**	**Timing**	**Others**	**IoT vs non-IoT**	**No. IoT**	**No. non-IoT**	**Dataset**	**Real-world Data**	**Dynamic Env.**	**Scalability**	**Reported Acc./ F1-score**	**Compared w/ Works**	**Code available**
	Pattern	[45]	2020	Locality Sensitive Hashes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	✗	22	0	LSIF [45]	✗	✗	✗	100%	[19], [26]	✗
		[46]	2020	MUD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	✗	42	0		✗	✗	✗	-	✗	✗
		[47]	2020	MUD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	✗	28	0		✗	✗	✗	-	✗	∼
Supervised	**One-class**	[19]	2017	RF	●	●	○	○	○	○	○	✗	31	0	IoT Sentinel [19]	✗	✗	✗	81.5%	✗	✗
		[23]	2017	GB, RF, XGBoost	●	●	○	○	○	○	○	✓	9	4		✗	✗	✗	99.28%	✗	✗
		[17]	2018	GB, kNN, Majority Voting, DT	●	●	○	○	○	○	○	✗	14	0		✗	✗	✗	99%	✗	✗
		[48]	2019	RF	●	●	○	○	○	○	○	✗	31	0	IoT Sentinel [19]	✗	✗	✗	90.3%	[19]	✗
		[49]	2020	TF-IDF	○	○	●	○	○	○	○	✗	> 53	> 6	IoTFinder [49]	✓	✗	✓	-	✗	✗
		[50]	2022	CNN, MLP	○	●	○	○	○	○	○	✗	13, 9	0		✗	✗	✗	99% (F1)	✗	✗
	Multi-class	[22]	2017	k-means	○	○	○	●	○	●	○	✓	21	2		✗	✗	✗	95%	✗	✗
		[51]	2018	RF	○	●	○	●	○	○	○	✗	21	0	UNSW [32]	✗	✗	✗	97% (F1)	✗	✗
		[18]	2018	LSTM-CNN	●	○	○	●	○	○	○	✗	15	0	UNSW [32]	✗	✗	✗	99.7%	✗	✗
		[32]	2019	MNB and RF	○	●	○	○	●	●	○	✓	24	6	UNSW [32]	✗	✗	✗	99.88%	✗	✗
		[24]	2019	Adaboost, DT, kNN, LR, and RF	○	○	○	●	●	○	○	✓	20	5		✗	✗	✗	-	✗	✗
		[25]	2019	MLP	○	○	●	●	●	○	○	✓	21	3	UNSW [32]	✗	✗	✗	99%	✗	✗
		[52]	2019	RF, kNN, DT, SVM, MV	○	○	○	●	○	○	○	✓	21	7	UNSW [32]	✗	✗	✗	> 94%	✗	✗
		[53]	2020	LSTM-RNN	●	●	○	○	○	○	○	✗	10	4	YourSmartHome [53]	✗	✗	✗	99.2%	✗	✗
		[54]	2020	Bayesian Network, RF	○	○	○	○	○	●	○	✗	39	0		✗	✗	✗	> 91%	✗	✗
		[29]	2021	2-D CNN and MTL	○	○	○	○	●	●	○	✗	14	6	UNSW [32]	✗	✗	✗	92%	[32], [51]	✗
		[55]	2022	DT	○	○	○	○	○	○	●	✗	24, 31	7	UNSW [32], IoT Sentinel [19]	✗	✗	✗	94.1%	[17], [19], [26], [32], [48]	✓
Unsupervised	**One-class**	[56]	2019	PCA, k-means	○	○	○	○	●	○	○	✗	10	0	UNSW Attack Data [57]	✗	✗	✗	> 94%	✗	✗
		[58]	2020	PCA, k-means	○	○	○	○	●	○	○	✗	10	0	UNSW Attack Data [57]	✗	✗	✗	> 94%	[32]	✗
		[59]	2021	k-means	○	○	○	○	●	○	○	✗	12	0	UNSW & Attack. [32], [57]	✗	✗	✗	98%	✗	✗
	Multi-class	[20]	2019	kNN	○	○	○	●	●	●	○	✗	33	0		✗	✗	✗	98.2%	✗	✗
		[26]	2019	DecisionTable, J48 DT, OneR, and PART	○	○	○	○	○	○	●	✗	23	0	IoT Sentinel [19]	✗	✗	✗	95%	[19]	✗
		[28]	2019	LSTM Autoencoder, Bayesian optimization	○	○	○	○	○	○	●	✓	21, 72	?	UNSW [32], Priv.	✓	✓	✗	100%[1]	✗	✗
		[60]	2020	HDBSCAN	○	○	○	●	○	●	○	✗	21	1	YourThings [2]	✗	✗	✗	99%	✗	✗
		[61]	2021	1-D CNN	○	○	○	●	○	○	○	✗	9	0	IoT Sentinel [19]	✗	✗	✗	93.96%	✗	✗
	Semi	[33]	2019	Seeded k-means and RF	●	○	○	●	●	○	○	✗	16	0		✗	✗	✗	97%	✗	✗
		[62]	2020	CNN and MTL	○	○	●	●	○	●	○	✓	24	6	UNSW [32]	✗	✗	✗	> 99%	[20], [32], [63]	✗
		[31]	2023	CNN and MTL	○	○	●	●	○	●	○	✓	24, 45	7, 4	UNSW [32], YourThings [2]	✗	✗	✗	97.6%	[20], [28], [32], [63]	✗

2.1. Approach Categories

We group existing approaches into the following categories: (1) pattern matching, (2) supervised learning, (3) unsupervised learning, and (4) semi-supervised learning.

2.1.1. Pattern Matching. Pattern-matching approaches utilize a predefined process to identify distinct patterns for each IoT device based on its network traffic. These patterns, such as the hash of a network flow or traffic distribution, create a unique baseline for each IoT device's network behavior. The IoT devices are identified by applying the same process to the unseen traffic and finding a match for the generated pattern.

1. The value is for identifying known devices. This paper identifies previously unseen devices with 82% average F1 score and 70% accuracy.

Two of the 3 works that follow this type of approach leverage the Manufacturer Usage Description (MUD) standard [64] to generate distinctive network patterns [46], [47]. The MUD standard specifies an allow-list of endpoints required for proper IoT device function. This allow-list, known as the MUD profile, is defined by the manufacturer. Hamza et al. [47] propose a tool called MUDgee that generates the MUD profile using a device's network traffic as input. They then generate runtime profiles from real-time network traffic and match them to the previously extracted MUD profiles to identify devices.

Another study using this approach applies a hash function to IoT traffic flows to generate a fingerprint [45]. More specifically, the authors employ the Locality Sensitive IoT Fingerprinting (LSIF) technique, based on the Nilsimsa hash [65], to generate unique hash values for device identification.

Summary. The main strength of this approach is that it uses complete network traffic rather than focusing on extracted features, which may vary depending on the environment. Its key limitation is the assumption that the captured traffic includes all possible device communications. However, if the manufacturers provide concrete MUD profiles for the devices, encompassing the complete set of functionalities, MUD-based methods may overcome this limitation.

2.1.2. Supervised Learning. A supervised learning approach involves training a model using extracted network features and corresponding labels for IoT devices. Then, it utilizes the trained model to predict the devices in unseen network traffic. This approach is the most popular among IoT device identification methods—adopted by 16 works. The works use two different ways of training the classification models: (a) one-class classification (6), and (b) multi-class classification (11).

One-class Classification. The one-class classification method entails training individual classifiers for each device in a one-vs-all approach. Each classifier performs a binary classification to identify if the given input (i.e., features extracted from the network traffic) corresponds to a specific device. The match is the device classifier with the highest probability. A total of 6 works employ this approach (as seen in Table 1).

Miettinen et al. [19] split this approach into two stages: (i) feature extraction and (ii) training classifiers to predict the device. They deploy these stages in a distributed setting, with the first deployed at the gateway and the second at the IoT Security Service Provider (ISSP). The ISSP receives the fingerprints from the gateway and trains binary classifiers for each device. It then predicts the device by selecting the highest-scoring classifier, resolving conflicts using an edit distance-based metric. Meidan et al. [23] use a similar approach involving two stages. The first stage performs binary classification between IoT and non-IoT devices, and the second classifies individual IoT devices. In contrast to Miettinen et al., they select separate machine learning algorithms to train the classifiers of different devices.

Unlike others, Perdisci et al. [49] focus only on the DNS queries. The authors use Term frequency (TF) and Inverse Document Frequency (IDF) on the domains queried by the IoT devices and generate fingerprints for each device. This method works independently of the presence of NAT.

Multi-class Classification. The multi-class classification method involves training a single classifier for all devices. The input for the training consists of a vector of extracted network features and the corresponding IoT device label. The classifier inputs the unseen traffic and assigns it to the device with the highest predicted probability.

Within the works that use this approach, we observe two methods of identifying non-IoT devices in the traffic. The first uses a binary classification to distinguish between IoT and non-IoT devices and then trains a multi-class classifier for individual IoT device identification [22], [25], [52]. The second denotes a separate class for non-IoT devices within the same multi-class classifier [24], [32].

Three works employ varying types of neural networks to train multi-class classifiers [18], [29], [53]. Dong et al. [53] combine NLP techniques with Long Short Term Memory (LSTM) networks for contextual traffic analysis, outperforming traditional methods such as Random Forest.

Aksoy et al. [26] address the problem of distinguishing between devices from the same vendor. They propose a two-level classification method, using the Partition and Regression Tree (PART) algorithm, that identifies the vendor in the first stage and the device in the second. They identify the device directly at the first level if it's the vendor's only device. Babun et al. [54] focus solely on identifying devices that use ZigBee and Z-Wave protocols. Due to the significantly different protocol stacks of ZigBee and Z-Wave, they select different packet types for feature extraction and train separate ML classifiers for each protocol. Kostas et al. [55] approach IoT device identification across all IP and non-IP protocols, using selection methods to identify the relevant features and the optimal ML algorithm (Decision Tree).

Summary. The main strength of the supervised learning approach is that it trains the models with well-defined inputs and outputs. Hence, the models learn the clear trends and distinguishing features between the devices and identify devices with higher accuracy. Similar to pattern-based matching, this approach also faces limitations in identifying behaviors not included in the training. Additionally, the accuracy and robustness of the trained models depend on obtaining a significant amount of high-quality labeled data, which is often labor-intensive and expensive.

2.1.3. Unsupervised Learning. An unsupervised approach trains a model using unlabeled data, relying solely on features extracted from network traffic. The algorithm automatically detects patterns and relationships within the data. Due to the unsupervised nature of this method, the models identify the device type that aligns with the input feature pattern rather than the specific device label. One can implement this approach using one-class and multi-class clustering methods, just like a supervised approach.

One-class Clustering. The unsupervised one-class clustering method trains a separate model for each device. Each model learns to recognize the traffic patterns of a single device while rejecting traffic from all the others using only the extracted features. The matching device type is the model generating the highest probability score.

All the works in this category use principal component analysis (PCA) to reduce the dimensions of the extracted features before inputting them to train the models [56], [58], [59]. The models output the probability of the feature set belonging to that device. When multiple models generate similar probabilities, the works employ conflict resolution mechanisms to identify the ”winner” device type.

Multi-class Clustering. The unsupervised multi-class clustering approach uses unlabeled network traffic features to train a single model that groups similar data into clusters, with each cluster representing a different class. The 4 studies using this method take significantly different approaches.

Marchal et al. [20] propose a distributed design with two components: device fingerprinting and device-type identification. The fingerprinting component monitors network traffic at the gateway to extract features. The identification component, a cloud service, receives these features from multiple gateways to train the model. This system aggregates data from various smart homes, enhancing the model's generalizability. Ortiz et al. [28] propose a method inspired by spectroscopy–utilizing stacked autoencoders, where each encoder's output serves as input for the next. These autoencoders create unique distributions of TCP flows for each device type, enabling differentiation among devices. They apply Bayesian modeling on these flows to generate clusters with discernible classes. Anantharaman et al. [60] present a human-in-loop approach for error handling in IoT devices using WIFI, Bluetooth, and Zigbee. They enhance network traffic features with the received signal strength indicator (RSSI) to localize devices and differentiate between multiple instances of the same device type. The authors employ HDBSCAN due to its ability to handle arbitrary cluster shapes, yielding a more robust identification model than centroid-based methods. Finally, Luo et al. [61] propose a deep learning method using a one-dimensional convolutional neural network (1D-CNN) to extract time series features from network traffic. They also introduce a feature enhancement method that uses residual connections to avoid feature loss problems.

Summary. The primary strength of the unsupervised approach is that it automatically identifies similarities and differences between the devices without needing a labeled dataset. Unlike the supervised learning approach, this approach is not limited to the behaviors seen in the training dataset. However, without the device labels, this method can only predict the type of device rather than its specific label, which reduces the accuracy of the trained models' output. Additionally, in the absence of clear inputs and outputs, the prediction accuracy of this model is generally lower than that of supervised models.

2.1.4. Semi-supervised Learning. A semi-supervised approach combines supervised and unsupervised methods by training a classifier with both labeled and unlabeled data. This approach is handy when large amounts of labeled data are hard to obtain. Typically, the training data includes a small portion of labeled examples alongside largely unlabeled data.

The 3 works that employ this approach are very different in their implementations. First, Thangavelu et al. [33] propose a distributed design where the IoT gateway and controller components are at the gateway and the ISP, respectively. The controller utilizes seeded k-means to cluster network traffic features, using labeled data to provide the initial centroids. The IoT gateway extracts features and classifies devices using the trained model—sending the confidence scores to the controller. These scores help decide if the model needs retraining to handle unknown devices or new functions. Next, Fan et al. [62] propose a method to minimize label loss by adjusting the training parameters of the model, effectively reducing the loss of IoT class labels as well as IoT/non-IoT labels in the labeled data. Finally, Fan et al. continue their research and propose AutoIoT [31], using CNN to reduce the dimensions of the extracted features. Then, they use a multi-task learning-based method to distinguish between IoT and non-IoT devices and further identify specific types of IoT devices. In this work, they also identify new device types (not included in training), label them, and automatically retrain the classifier with these new device types.

Summary. The semi-supervised approach addresses the limitations of supervised and unsupervised methods by introducing a small amount of labeled data during training—providing context to help the model identify patterns in the unlabeled data. However, the accuracy of this approach depends on the quality of the labeled data.

2.2. Feature Types

The works using learning-based approaches all extract features from the network traffic to train their classification models. This feature selection is of critical importance. As shown in Table 1, the types of features vary significantly.

A network packet is composed of various layers, each with different fields. These layers differ based on the packet's protocol. The works extract various combinations of fields or statistics from these layers for their methodology. We categorize these features to compare the various identification methods.

IP Header. The IP header of the packet contains information such as the source and destination IP addresses, protocol, and packet length. The IP addresses provide insight into the servers with which the devices are communicating. We see 7 of the learning-based approaches in Table 1 use fields from the IP header as features. Since MUD profiles consist of an allow-list of endpoints for the devices, the 2 pattern-based approaches [46], [47] also use fields from this header.

Transport Layer. The transport layer in the packet stores the information required for end-to-end communication, such as source and destination port numbers. TCP and UDP are the most widely used transport layer protocols, forming the foundation for higher-level protocols like DNS, NTP, and HTTP. 5 of the works in Table 1 use features from all TCP/UDP packets. Whereas, the work by Bezawada et al. [17] focuses solely on the header fields of TCP packets.

Application Layer. 4 of the works in Table 1 extract features from the application layer. Perdisci et al. [49] use the DNS layers to extract domains queried by the IoT devices. With their specific functionalities, IoT devices interact with a limited set of domains, allowing DNS traffic to provide distinguishable characteristics. Chen et al. [25] and Fan et al. [31], [62] extract features from the TLS layer. This information includes fields such as cipher suites, the client's public key, extensions, and statistics about these fields.

Traffic Statistics. The traffic statistics include features that compute traffic details such as the number of packets, mean rate (bps), average packet length, etc. These statistics provide an overview of the device's network behavior. In total 12 works (Table 1) use these statistics for training their models. While most works use traffic statistics alongside other feature types, 3 works [18], [52], [61] rely solely on statistical values.

Flow Statistics. A network flow includes fields such as source IP address, destination IP address, source port, destination port, and protocol. Relevant statistics for these flows include flow volume, duration, average rate, and mean periods per flow. In Table 1, there are 9 works that utilize these statistics as features, although each work uses a different set of statistics.

Timing. The consistent traffic patterns of IoT devices, particularly during idle periods, allow statistical features—such as inter-packet arrival times—to provide distinctive device characteristics. Sivanathan et al. [22], [32] include device active time and sleep time in their features. Anantharaman et al. [60], Babun et al. [54], and Fan et al. [31] utilize inter-packet arrival time in their methods. Whereas Fan et al. [62] and Marchal et al. [20] compute the statistical features at regular time intervals for training purposes. Finally, Liu et al. [29] extract flow-based features in a temporal order for training their models.

Others. 3 works employ automated feature extraction techniques to select the features that best represent the devices, and therefore do not fit into other categories Aksoy et al. [26] and Kostas et al. [55] use a genetic algorithm to identify the most relevant features, whereas Oritz et al. [28] use LSTM autoencoders for this purpose.

2.3. Evaluation Environment

The works evaluate their IoT device identification methods in varying environments. These environments vary in the number of IoT devices, the presence of non-IoT devices, and other realistic aspects of the environment. The vantage point for data collection in all these works is *local*, i.e., they collect the network traffic produced by devices at the router level, within a LAN. Local traffic provides access to IP and MAC addresses, making it straightforward to obtain ground truth when capturing traffic in a mixed environment. A more realistic setup would involve more central monitoring—such as at an ISP—rather than local monitoring at home, where traffic is behind a NAT and all devices share a single IP address. We define an environment with realistic aspects, such as multiple users in a household, multiple mobile devices with companion apps, and changes to the number of devices, as a dynamic environment. An environment where the number of devices remains unchanging during the data collection period is static. The evaluation environment has a direct impact on the method's performance when identifying IoT devices. A lab environment with only IoT devices connected to the network provides an idyllic setup for evaluating these methods. While the methods perform well in this environment, their performance may not generalize to settings with noise from non-IoT devices or dynamic setups. These differences make it difficult to compare the performance of different works and to identify the state-of-the-art.

A simple evaluation environment consists of IoT devices connected to a gateway for Internet access, with no non-IoT devices. 17 out 31 evaluate their works in such an environment. The number of IoT devices used for evaluation in these works ranges from 9 to 42. Only 9 of all works use more than 30 devices, whereas 10 works use 15 or fewer IoT devices for their evaluation. Some studies evaluate IoT devices alongside varying numbers and types of non-IoT devices, such as phones, tablets, and laptops. Since non-IoT devices generate exponentially more communication traffic than IoT devices, their presence impacts the performance of the methods. 14 of the works have non-IoT devices in their setup, making their evaluation results more realistic than others.

Only 2 of the works use real-world setups for their evaluations. First, Perdisci et al. [49] evaluated their proposed method, IoTFinder, with four datasets. One was collected using 66 devices (IoT and non-IoT), while the other three came from a U.S. ISP, a university campus, and a third-party source. Unlike other works, IoTFinder can identify devices solely through DNS traffic, even when they are behind NAT. Next, Ortiz et al. [28] use two distinct datasets to evaluate their proposed method. One is a public dataset [32], and the other one is from a private lab where they regularly add and remove IoT devices—for a total of 72 devices.

2.4. Summary and Open Gaps

Our systematization reveals that the existing works differ significantly in their approaches. Among all the works listed in Table 1, only 8 compare their performance with existing works. Such a lack of empirical comparisons hinders the identification of state-of-the-art and future research developments. As seen in Table 1, most works use only accuracy as the performance metric, which fails to provide an overview of performance for all devices in the case of imbalanced datasets. Furthermore, performance results obtained in different evaluation settings with varying environmental factors are not comparable. Therefore, to advance our community, we must validate past IoT device identification results with *compelling empirical evidence*.

Besides, we lack a clear understanding of how IoT device identification methods perform in a realistic environment. Unlike typical laboratory settings, realistic environments are unpredictable, dynamic, and often dominated by non-IoT device traffic. Furthermore, under realistic conditions, (new) devices are plugged in and/or removed from the network, and they may experience malfunctions or power outages. All these factors affect the network behavior of IoT devices, impacting the performance of device identification methods. As discussed in Section 2.3, most works rely on simple laboratory setups for their evaluations. Hence, evaluating the impact of realistic conditions on existing identification methods is essential for real-world deployment.

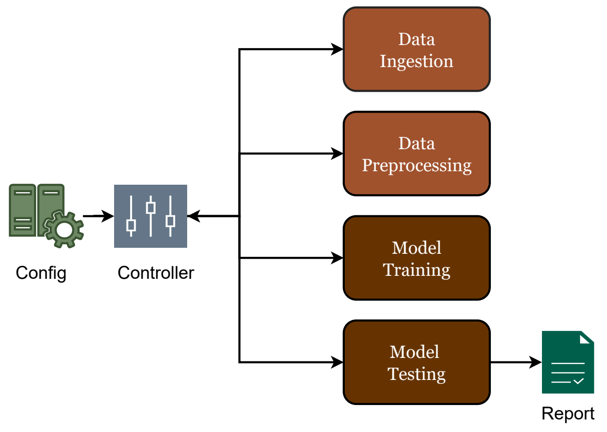


Figure 1: Design overview of the Evaluation Framework.

3. EVALIOT: Evaluation Framework

We present EVALIOT, a framework designed for the systematic evaluation of different IoT device identification methods across diverse environments. Its modular design ensures consistent assessment of methods with varying I/O requirements. At the same time, its plug-and-play mechanism supports seamless integration of datasets and identification methods, allowing for direct comparison of experimental results. We release our framework as open-source.

Overview. We design EVALIOT to be modular and configurable. The framework has four main modules, as shown in Figure 1, representing the different stages of a typical evaluation: a) data ingestion, b) data preprocessing, c) model training, and d) model testing. The *Controller* manages all four modules and uses configuration files to set the evaluation requirements. *Data Ingestion* loads the datasets, and *Data Preprocessing* specifies which are for training and testing. The *Controller* then sets up the training and testing modules, including repository paths, cross-validation, and whether to use saved models or train new ones. It initiates the *Model Training* module to either train or load a model for device identification, followed by the *Model Testing* phase to evaluate the model on test datasets. We will now explore these modules in detail.

Data Ingestion. The *Data Ingestion* module loads the datasets. Since the focus is on IoT device identification via network traffic analysis, the evaluation datasets consist of PCAP files. The configuration file specifies the folder structure used by the dataset, indicating whether traffic is mixed or separated by devices. This module generates a list of all PCAP files in the datasets and returns such a list to the *Controller* module for further processing.

Data Preprocessing. The *Data Preprocessing* module prepares the datasets using the evaluation requirements. The requirements include both training and testing datasets, as well as the decision to test solely on known devices from the training set or to detect unknown devices. For the "known devices" condition, this module removes the unknown device traffic from the test dataset. Method-specific preprocessing, such as feature extraction, takes place in the *Model Training* module.

Model Training. The *Model Training* consists of a base module that provides a framework-compatible skeleton. Each method inherits this module to implement method-specific preprocessing and training using the original codebase. Different methods operate at various granularities, such as per packet, per sequence of packets, or per flow. Utilizing the original codebase enables the framework to accommodate these differences. We store the trained models as files, facilitating faster evaluations with different datasets. The textitController enables cross-validation when the train and test datasets are the same. The *Model Training* module then uses the Stratified K-Fold to split the data and trains `K` separate models.

Model Testing. The *Model Testing* module is a base module with skeleton functions, including a fixed report generation function to ensure consistent evaluation reports across methods. Each method-specific extension customizes the testing function using its original code and stores the test results in the required format for report generation. The report generation function utilizes the `scikit-learn` library to generate the classification report using the test results with *true* and *predicted* values. For cross-validation, this module utilizes the data designated for testing by the training module and assesses the K models with the respective data. On completion of the evaluation, the *Controller* instructs the *Report* module to generate the final report.

Report. The report begins with evaluation requirements and configurations. We utilize the `classification-report` function from the `scikit-learn` library to calculate performance statistics, including precision, recall, and F1-score. This function calculates performance metrics for each device and the aggregated ones for the entire dataset. The per-device metrics allow us to observe performance differences across various devices. The overall metrics provide a broad overview of the method's performance in a specific setting.

4. Empirical Evaluation

While Table 1 shows that existing works report very high-performance metrics—accuracy or F1-score is almost always $> 90\%$—how these methods compare to one another remains *unclear*. As a result, it is difficult to determine the current state of the art and whether any approaches significantly outperform others. To this end, we select and empirically evaluate representative works in the field.

Maali et al. [38] provide empirical comparisons of existing methods and evaluate the generalizability of their ML algorithms and optimal feature sets across different environments. Unlike this study, which examines environmental factors such as operational modes (active versus idle) and transferability (temporal and spatial), we analyze the factors that impact the performance of identification methods in real-world smart homes. These factors include non-IoT traffic, (new) device addition and removal, multiple users, and numerous devices.

For our empirical evaluations, we implement an identification method—genIoTID—using the most generalizable ML algorithm and feature set identified by Maali et al. to assess its generalizability. Next, we select IoTDevID by Kostas et al. [55] because they compare their work with five others across diverse approaches [17], [19], [26], [32], [48]—this is the prior study with the *widest* comparison to other methods—demonstrating improved performances. Additionally, to broaden the scope of our evaluations, we include works published in A/A* conferences and journals [66]: IoT Sentinel [19], MUDgee [47], Your Smart Home [53].

The works selected thus far primarily utilize pattern matching or supervised approaches. To ensure a comprehensive evaluation, we also include an unsupervised approach from Table 1 that aligns with the goals of our study. Devicemein [28] stands out in that regard as it utilizes *real-world data* in a *dynamic environment* for its evaluation.

Due to the unavailability of the source code of the selected works, except for IoTDevID, we first re-implement the proposed methods within our EVALIOT framework. To validate our implementations, we assess the performance of these works on the originally adopted datasets and ensure our results are consistent with those reported in the papers. We report our results in Appendix B.

4.1. Experimental Setups

We conduct distinct experiments for each research question in our university's IoT lab, which simulates a studio apartment with 23 IoT devices, including a smart fridge, smart TV, Google Home Mini, cameras, and more—complete list in Appendix A. Given that the works that we reproduce are intended for local network traffic (i.e., before NAT), we replicate this setup for our experiments. Hence, we capture all TCP/IP traffic at the LAN port of the gateway and interact with the devices to mimic typical smart home user behavior when needed.

For each research question, we assess the influence of a specific environmental factor on IoT device identification methods through a series of experiments. First, we establish a baseline by evaluating the method on network traces captured in a setting that excludes the factors under consideration. Next, we introduce the environmental factors and collect new traces. We train the methods on the first set of traces and test them on the second set to study the impact of introducing environmental factors on performance. Finally, we evaluate the methods using only the second set of traces to determine how the models perform when the studied factors are part of the training dataset. For comparing the performances, we use *weighted average* values of precision, recall, and F1-score to address label imbalance.

4.2. RQ1: IoT vs. non-IoT Devices

RQ1: *How does noise from non-IoT devices affect the performance of device identification approaches?*

Aim. With this research question, we aim to determine whether the presence of non-IoT devices in the network impacts the performance of identification methods. IoT devices are everyday appliances with specific functions that connect to the Internet via network protocols, whereas non-IoT devices are phones, tablets, and laptops. However, the distinction between IoT and non-IoT traffic is not always clear-cut. Most IoT devices have companion apps—regardless of whether they connect via Bluetooth or other protocols—that sync and transmit device data through mobile devices. As a result, mobile traffic often includes IoT-related data. Therefore, such traffic cannot be considered purely "non-IoT." We tackle this concern by splitting the devices into three categories: a) IoT devices, b) mobile devices, and c) IT devices. Mobile devices, such as smartphones, interact directly with IoT devices, whereas IT devices refer to general computing devices, such as laptops and PCs.

Experiment Settings. For this RQ, we collect the network traffic in three different settings. First, we connect all 23 IoT devices to the network and capture the traffic from normal device usage. Then, we add a smartphone and a tablet to the network. We send commands to the devices from both devices during this experiment. We also regularly sync the Bluetooth devices (in addition to the 23 devices) to their servers to add more noise to the data. Finally, we add laptops to the network and work as usual, avoiding security-sensitive tasks such as banking or social media. After collecting the data in all three settings, we evaluate the methods using different combinations of the datasets.

Observations. First, we obtained the baseline by evaluating the methods with only IoT devices. *IoTDevID* displayed the most promising results with a near-perfect F1-score of 0.994, whereas *IoT Sentinel* got 0.729, *MUDgee* 0.79, *Your Smart Home* 0.927, *genIoTID* 0.867, and *Devicemein* 0.602. *Devicemein*, being the only unsupervised method, understandably performs poorly compared to the others.

We assumed the performance of the methods would decline after introducing non-IoT devices to the network traffic. The performances of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, *genIoTID*, and *Devicemein* followed our assumption. The F1-score of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, *genIoTID*, and *Devicemein* notably drops by 17.5%, 26.98%, 11.8%, 27.92%, and 52.49% respectively, when we test with IoT, mobile, and IT devices connected to the network. Table 2 shows the best and worst performance values in red and green, respectively. Even though these methods—except *genIoTID*— perform the best with only IoT devices, those results do not accurately represent their performance in a realistic environment. We observed an improvement in the performance of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, and *Devicemein* when we train with IoT, mobile, and IT devices connected to the network, whereas

TABLE 2: The **weighted average** of Precision, Recall, and F1-score for evalutations with IoT, IoT + Mobile, and IoT + Mobile + IT traffic. For each method, red highlights the lowest performance values and green the highest values.

Train	Test	IoT Sentinel			MUDgee			Your Smart Home			IoTDevID			genIoTID			Devicemein		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
IoT	IoT	0.780	0.709	0.729	0.853	0.798	0.790	0.942	0.926	0.928	0.995	0.995	0.994	0.882	0.846	0.845	0.575	0.664	0.602
IoT	IoT + Mobile	0.717	0.582	0.610	0.827	0.848	0.813	0.806	0.764	0.768	0.932	0.902	0.882	0.887	0.793	0.816	0.483	0.521	0.446
IoT	IoT + Mobile + IT	0.697	0.575	0.601	0.891	0.926	0.890	0.686	0.674	0.677	0.878	0.881	0.876	0.805	0.557	0.609	0.316	0.428	0.286
IoT + Mobile	IoT + Mobile	0.770	0.685	0.706	0.851	0.843	0.822	0.888	0.871	0.867	0.992	0.992	0.991	0.857	0.813	0.818	0.511	0.604	0.547
IoT + Mobile	IoT + Mobile + IT	0.741	0.648	0.662	0.808	0.806	0.784	0.748	0.754	0.739	0.952	0.941	0.918	0.639	0.568	0.576	0.500	0.528	0.495
IoT + Mobile + IT	IoT + Mobile + IT	0.760	0.685	0.702	0.803	0.759	0.757	0.892	0.877	0.879	0.980	0.980	0.979	0.898	0.860	0.867	0.412	0.582	0.477

genIoTID performs the best in that scenario. These observations indicate that including non-IoT traffic offers a more accurate assessment of identification methods in real-world conditions. Among the five methods, *genIoTID* is least affected by mobile devices, while *IoTDevID* is most robust to all non-IoT devices. *genIoTID* relies on time-based features like sleep and interarrival time, which are more susceptible to noise from non-IoT traffic, explaining its performance drop. Table 2 shows that *IoTDevID* achieves an F1-score of 0.97 or more when trained with traffic containing non-IoT devices. Hence, we empirically confirm that *IoTDevID* is not impacted by the inclusion of non-IoT devices.

MUDgee's performance defied expectations. It achieved its highest F1-score (0.890) when trained on IoT-only traffic and tested with all devices, but performed worst (F1-score: 0.756) when trained and tested with all devices connected to the network. This behavior is due to how *MUDgee* generates MUD profiles, i.e., with a single input `pcap` file. Our dataset contains `pcap` files of 15-minute durations. During same-dataset evaluations, *MUDgee* analyzes a single 15-minute `pcap` file per device, making its performance highly dependent on the traffic captured in that brief window. In contrast, cross-dataset evaluations utilize merged `pcap` files that aggregate all traffic for each device, resulting in more comprehensive MUD profiles and better performance.

In Section 2.3, we note that most studies evaluate with only IoT devices. While they report high accuracy, performance may decline in environments with non-IoT devices—highlighting the need for noisy datasets in future research.

Answer: *The noise from non-IoT devices deteriorates the performance of IoT device identification methods. The performance of IoT Sentinel, Your Smart Home, IoTDevID, genIoTID, and Devicemein was impacted by the presence of non-IoT devices, resulting in a drop in the F1-score by* 17.5%, 26.98%, 11.8%, 27.92%, *and* 52.49%, *respectively. However, when we included non-IoT devices in the training phase, IoTDevID performed with an F1-score higher than* 0.97. *MUDgee's performance was not impacted by the presence of non-IoT devices.*

4.3. RQ2: Static vs. Dynamic Environments

RQ2: *How does a dynamic environment, where the set of devices, or the number of users regularly changes, impact the performance of IoT device identification methods?*

Aim. A static environment is where the number of IoT and controlling mobile devices remains unchanged throughout data collection. Conversely, a dynamic environment is one where there are changes in the number of IoT devices, mobile devices, or both during data collection, such as when devices or users are added or removed. The number of IoT devices changes as we plug in or unplug (new) devices, while the number of mobile users changes as we install companion apps on more devices or add new users. Since the dynamic environment more accurately reflects a real smart-home environment, we aim to understand the impact this will have on the IoT device identification methods.

Experiment Settings. We collect the network traffic in two different settings: static and dynamic. We set up the lab in a static setting with 18 IoT devices and a smartphone, maintaining this configuration throughout the data collection period. For the dynamic setting, we introduce numerous changes during data collection, including randomly connecting and disconnecting the devices, adding companion apps to a secondary mobile device (iPad) to operate the devices, and adding new IoT devices to the network. We evaluate all the methods using the data from these two settings.

Observations. First, we established the baseline by evaluating the methods in the static setting (Table 3). *IoTDevID* and *Your Smart Home* performed the best with an F1-score of 0.995 and 0.983, respectively. Whereas, *IoT Sentinel*, *MUDgee*, *genIoTID*, and *Devicemein* achieve an F1-score of 0.796, 0.734, 0.879, 0.813, and 0.648 respectively.

We assumed a dynamic setting would impact the performance of identification methods, and the six methods confirmed this expectation. We see a notable drop in F1-score of *IoT Sentinel*, *MUDgee*, *Your Smart Home*, and *Devicemein* when tested in a dynamic setting—by 47.45%, 14.28%, 16.48%, and 32.7% respectively. Whereas that of *IoTDevID* and *genIoTID* dropped only by 4.7% and 7.5%, respectively. From these values, we can infer that *IoTDevID* and *genIoTID* are more robust to these environmental changes, compared to *IoT Sentinel*, *MUDgee*, *Your Smart Home*, and *Devicemein*. Furthermore, we observed an improvement in the performance of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, *genIotID*, and *Devicemein* when we trained them in the dynamic setting. Whereas the performance of *MUDgee* displayed a minor drop, which is within the error margins experienced with every run. Out of the six methods, *IoTDevID* performed the best (F1-score of 0.986) when trained using the dynamic dataset, whereas *genIotID* gave its best performance (F1-score of 0.897) in that setting. While *MUDgee* did not display the same trends as others, we believe that extended data collection during the MUD profile generation phase could lead to more comprehensive profiles—improving performance.

TABLE 3: The **weighted average** of Precision, Recall, and F1-score for static and dynamic traffic.
For each method, red highlights the lowest performance values and green the highest values.

Train	Test	IoT Sentinel			MUDgee			Your Smart Home			IoTDevID			genIoTID			Devicemein		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Static	Static	0.860	0.769	0.797	0.776	0.744	0.735	0.984	0.983	0.983	0.996	0.996	0.996	0.900	0.877	0.879	0.654	0.684	0.648
Static	Dynamic	0.490	0.443	0.419	0.804	0.716	0.689	0.839	0.831	0.821	0.942	0.963	0.949	0.835	0.828	0.813	0.385	0.539	0.436
Dynamic	Dynamic	0.750	0.629	0.668	0.669	0.686	0.665	0.899	0.895	0.892	0.986	0.986	0.986	0.930	0.885	0.897	0.619	0.638	0.612

Most existing research uses static evaluation environments (as seen in Table 1). Dynamic environments introduce realistic network variability, resulting in more diverse device traffic patterns than static environments—reflecting better real-world conditions. Hence, future research should evaluate their methods in this environment to ensure real-world applicability.

Answer: *A dynamic environment impacts the performance of IoT device identification methods. The impact on the performance of IoTDevID is the least, where the F1-score drops by only* 4.7%. *Whereas, IoT Sentinel experienced the most impact, with F1-score dropping by* 47.45%.

4.4. RQ3: Scalability

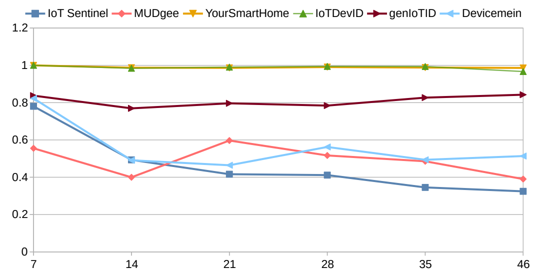
RQ3: *How well does the models' performance scale with an increasing number of IoT devices?*

Aim. With this RQ, we aim to evaluate the scalability of the IoT device identification methods.

Experiment Settings. We perform this experiment with two separate datasets. First, the MonIoTr dataset [67] was collected in their US lab (45 devices). Second, the dataset was collected in our lab (23 devices). Since an average smart home contains 7 IoT devices [44], we start the evaluations with 7 devices and add traffic from 7 more devices in subsequent evaluations.

Observations. Our experiments show that increasing the number of devices affected the six methods differently(Figure 2). *IoT Sentinel* displayed a gradual decrease in the F1-score as the number of devices increased. With the MonIoTr dataset, the F1-score dropped by 58.4% as the number of devices increased from 7 to 46. In contrast, with our Lab dataset, the F1-score dropped by 11.44% as the number of devices rose from 7 to 23. *MUDgee*'s performance showed some inconsistency with the MonIoTr dataset. The F1-score dropped from 0.55 to 0.39 when the number of devices increased from 7 to 14, but rose to 0.59 when the number of devices increased to 21. After this point, we observed a consistent decline in the F1-score with increasing device numbers. Further investigation indicated that the devices used in the experiments with 7 and 14 devices had limited traffic within the MonIoTr dataset—hence the poor performance. When evaluating *MUDgee* with our Lab dataset, we observed a notable drop in the F1-score (by 20.7%) when the number of devices increased from 7 to 14. After that, the F1-score stabilized around 0.6 and displayed only minor variations as we increased the number of devices.

(a) Using the MonIoTr US dataset [67]



(b) Using Lab Dataset

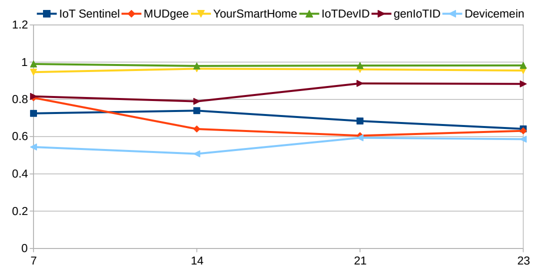


Figure 2: Weighted average of F1-score for all methods, with increasing number of devices.

Next, *genIoTID* and textitDevicemein exhibited more consistent performances. While the *genIoTID*'s F1-score stayed between 0.85 and 0.76—maximum value at 0.842 with 46 devices—with the MonIoTr dataset, *Devicemein*'s F1-score dropped from 0.82 to 0.49 when the number of devices increased from 7 to 14, but stayed between 0.46 and 0.56 for the rest. With our Lab data, *genIoTID*'s performance improved by 8.2% as the device count went from 7 to 23, *Devicemein*'s F1-score ranged from 0.54 to 0.59—achieving slightly higher values with more devices.

Finally, the performance of *Your Smart Home* and *IoTDevID* showed limited impact as the number of devices increased. Both started with a perfect F1-score with 7 devices. While they experience minor drops in performance with an increasing device count, the F1-score stays between 0.98 and 1.0 for *Your Smart Home*, and between 0.96 and 1.0 for *IoTDevID*. They displayed comparable performance with our Lab dataset, where the F1-score stayed between 0.945 and 0.965 for *Your Smart Home* and between 0.979 and 0.99 for *IoTDevID*.

We conclude that identification models vary in scalability, with some struggling as the number of devices increases. For two of the six methods, the number of devices in the network especially affected their performance(Detailed results in Appendix C)—further evaluations are required to

TABLE 4: The **weighted average** of Precision, Recall, and F1-score to identify unknown devices in the network.

	Baseline	Overall			Unknown		
	F1-score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
IoT Sentinel	0.419	0.580	0.402	0.402	0.123	0.777	0.213
MUDgee	0.689	0.865	0.710	0.683	1.0	0.068	0.127
Your Smart Home	0.821	0.796	0.626	0.686	0.014	0.016	0.015
IoTDevID	0.949	0.874	0.867	0.864	0.144	0.057	0.081
genIoTID	0.813	0.825	0.711	0.718	0.201	0.749	0.317
Devicemein	0.436	0.335	0.309	0.280	0.147	0.657	0.240

explore their scalability. Despite an initial decline, *Devicemein*'s performance on the MonIoTr dataset stayed stable regardless of the number of devices but remained notably lower due to its unsupervised nature. The remaining three methods demonstrated scalability, with *Your Smart Home* and *IoTDevID* displaying commendable performance.

Answer: *Some methods fare well with an increasing number of devices, while the performance of others deteriorates. The performance of IoT Sentinel and MUDgee deteriorated significantly with the increasing number of devices from 7 to 46, leading to an overall drop in F1-score by* 58.4%, 29.7%*, respectively. The negligible impact on the performance of Your Smart Home, IoTDevID, genIoTID, and Devicemein demonstrated their scalability.*

4.5. RQ4: Detecting Unknown Devices

RQ4: *How well do IoT device identification methods identify unknown devices?*

Aim. This experiment evaluates each method's ability to detect unknown devices—a key step in threat detection, where distinguishing known from unknown devices is essential.

Experiment Settings. We use a training dataset with fewer devices than the testing dataset. Device identification methods match devices using similarity scores, probabilities, or cluster distances. If this value falls below a set threshold, indicating low confidence in the prediction, we classify it as *unknown*. Since value distributions vary by method, we evaluate each method separately for thresholds from 0.1 to 0.7. For each method, we select the threshold value that optimizes the F1-score for unknown device detection. The resulting performance metrics provide insight into the method's ability to detect unknown devices and its impact on identifying known ones.

Observations. First, we obtained the baseline performance values by treating the unknown devices in the test dataset as noise and disregarding them in the device identification process. Then, we evaluated each method with thresholds from 0.1 to 0.7. When adopting low thresholds, we observe significant false negatives for unknown device identification and false positives for known device identification. The opposite occurs when high thresholds are adopted. As the optimal value varied for each method, we chose method-specific thresholds that minimize these misclassifications.

Table 4 shows that the overall performance of all the methods, except *MUDgee*, declines when detecting unknown devices. The F1-score of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, *genIoTID*, and *Devicemein* decreased by 4.05%, 16.6%, 8.9%, 11.7%, and 35.7% from the baseline. Although *Devicemein* has the lowest overall performance, it ranks second in detecting *unknown* devices, just behind *genIoTID*. In contrast, the two options with the best overall performance, *Your Smart Home* and *IoTDevID*, perform the worst when detecting unknown devices. Nonetheless, none of the six works performs sufficiently well when distinguishing between known and unknown devices. Further investigation revealed that these methods assign similarly high probability values to correct and incorrect predictions. As a result, they often fail to detect *unknown* devices in the network, incorrectly matching them to one of the known devices—explaining the poor performance reported in Table 4.

Answer: *IoT device identification methods often fail to distinguish between known and unknown devices. They tend to show equal confidence in both correct and incorrect classifications, frequently misclassifying unknown devices as known ones.*

4.6. Summary and Comparison

In response to our four research questions, we evaluated how various realistic factors affect the performance of the selected IoT device identification methods. Our experiments revealed that non-IoT devices generate traffic noise that disrupts time-based features, and that mobile devices contribute additional noise to IoT traffic through local interactions. Furthermore, dynamic environments further disrupt the ordered and predictable interactions of static setups, affecting the feature combinations used by identification models. These variations in the network traffic showed a noticeable decline in the performance of the identification methods.

The observed decline in the F1-score of *IoT Sentinel* in the presence of realistic factors highlights the challenges associated with a real-world environment. While the performance of *MUDgee* was not affected by the presence of non-IoT devices, the same was not the case for other factors. Nonetheless, the decline in its F1-score was lower compared to *IoT Sentinel*, demonstrating more robustness toward realistic environmental factors. However, *MUDgee* has a limitation that *IoT Sentinel* does not: it requires more ground truth data to create MUD profiles, including the MAC address of the device and the MAC and IP addresses of the gateway, making deployment more challenging.

While *Your Smart Home* was affected by non-IoT devices and dynamic traffic, there was a significant increase in performance when we included these factors in training. Similarly, *IoTDevID*'s performance dropped only when non-IoT devices and dynamic traffic were excluded from training, demonstrating resilience to these factors. *genIoTID* showed a similar pattern but performed best when we included non-IoT devices and dynamic traffic in the training. These three approaches—*Your Smart Home*, *IoTDevID*, and *genIoTID*—exhibited the ability to scale effectively with an increasing number of devices. *Devicemein*'s performance was also affected by non-IoT devices and a dynamic environment, but improved when we included these factors in the training. Although *Devicemein* scored lower across all metrics, its unsupervised nature requires far less manual effort. With more data, it can likely adapt better to real-world environmental factors.

Lastly, we found that, during our experiments, all six methods had difficulty distinguishing between known and unknown devices on the network. Given the prominent role of identification methods in threat detection (especially anomaly detection) [14], confidently misclassifying unknown devices poses serious risks, which future works needs to investigate.

5. Limitations and Future Work

Despite our results and contributions, our research is not exempt from limitations. Due to the unavailability of their source code, we reimplemented four methods for our experiments—*IoT Sentinel*, the runtime profile generation and device identification part of *MUDgee*, *Your Smart Home*, and *Devicemein*. Hence, our findings about these four works depend on our understanding and implementation, which may or may not be exact to the original work.

Additionally, our findings on MUDgee depend on the automatic generation of MUD profiles from network traffic, which carries inherent limitations [11]—results may differ when using MUD profiles provided by manufacturers.

Another notable limitation is that our conclusions about unknown-device detection rely on a threshold-based approach, which may not be the most effective solution to this problem. We recommend further investigation into alternative techniques—such as one-class classification or anomaly-scoring methods—to enhance the ability of device-identification tools to detect unknown devices. Such investigations would also help assess the feasibility of applying IoT device-identification methods for security purposes.

Our study also highlights less explored areas, such as identifying devices behind NAT. Analyzing local traffic raises potential privacy concerns for users, and inspecting network traffic after NAT complicates the identification process, as it becomes challenging to separate device-specific flows. We recommend further research in this direction.

Since our study demonstrated that existing methods suffer from realistic environmental conditions (because of their noisy and dynamic nature), future research should investigate the adoption of more robust features and explore Continual Learning (CL) to adapt to behavioral changes.

Finally, our framework, EVALIOT, allows future researchers to benchmark their identification methods against others while also investigating the impact of other realistic factors, such as firmware updates or device malfunctions.

6. Key Takeaways

Impact of non-IoT Devices. We observed a significant drop in the F1-score of *IoT Sentinel*, *Your Smart Home*, *IoTDevID*, *genIoTID*, and *Devicemein* —17.5%, 26.98%, 11.28%, 11.8%, 27.92%, and 52.49% respectively—demonstrating the impact of mixing IoT and non-IoT traffic. Over half (51.7%) of the works we systematized do not include non-IoT devices when evaluating their IoT device identification methods—raising concerns about their applicability in real-world scenarios where IoT and non-IoT devices coexist. We recommend future research to *include* a realistic number of non-IoT devices in their evaluation environment.

Impact of Dynamic Environments. The performance of all studied works declined when evaluated in a dynamic environment—with the drop in F1-scores ranging from 4.7% (IoTDevID) to 47.45% (IoT Sentinel). These observations highlight the importance of realistic environments for accurately evaluating identification methods. Among the works we review, only two use a dynamic setting, highlighting the importance of adopting such settings in future evaluations.

Importance of Scalable Methods. Our scalability experiments showed that larger device sets can reduce identification performance, depending on the method's robustness. *IoT Sentinel* and *MUDgee* experienced notable drops in their F1-scores (54.8% and 29.6%) when the device count increased from 7 to 46. In contrast, *Your Smart Home* and *IoTDevID* had minor drops of 1.3% and 4%, respectively. *genIoTID* maintained a stable performance regardless of device count, while *Devicemein*'s F1-score stabilized after an initial decline. We recommend that future research prioritize evaluating the scalability of the proposed methods, as this aspect has often been overlooked.

Detection of Unknown Devices Our experiments showcased that IoT device identification methods perform poorly when distinguishing between known and unknown devices. If these methods fail to detect unknown devices, their effectiveness in security applications such as threat detection is questionable. We recommend that future research investigate this further to determine the extent to which IoT device identification is still relevant for threat detection.

7. Conclusion

We conducted a comprehensive study on IoT device identification by analyzing existing literature and investigating the impact of datasets, features, and deployment environments on the performance of these methods. Our experiments and observations demonstrated that, despite the reported excellent performance in laboratory settings, the applicability of existing methods under realistic conditions is limited and requires further research. In particular, we discovered that noise from non-IoT devices and environmental changes—such as user activity or the addition/removal of (new) devices—negatively impact the performance of device identification methods. Additionally, existing models often struggle to handle a large number of devices. Finally, we observed the inability of the existing IoT device identification methods to differentiate between known and unknown devices. To conclude, we provided actionable takeaways for future research.

Acknowledgements

We are grateful to our reviewers for their valuable inputs and feedback. The INTERSECT project has supported this work, Grant No. NWA 1160.18.301, funded by the Netherlands Organization for Scientific Research (NWO). Part of the research has been funded by the German Federal Ministry for Economic Affairs and Energy (BMWE) under agreement no. 03EI4101D (TRACEY). The findings reported herein are the sole responsibility of the authors and do not necessarily reflect the views of NWO or BMWE.

References

[1] L. S. Vailshery, "Number of internet of things (iot) connected devices worldwide from 2019 to 2030," 2022. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security evaluation of home-based IoT deployments," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2019.

[3] A. Girish, T. Hu, V. Prakash, D. J. Dubois, S. Matic, D. Y. Huang, S. Egelman, J. Reardon, J. Tapiador, D. Choffnes, and N. Vallina-Rodriguez, "In the room where it happens: Characterizing local communication and threats in smart homes," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2023.

[4] 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 ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2020.

[5] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is anybody home? inferring activity from smart home network traffic," in *Proceedings of the IEEE Security and Privacy Workshops (SPW)*, 2016.

[6] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[7] D. Wood, N. Apthorpe, and N. Feamster, "Cleartext data transmissions in consumer IoT medical devices," in *Proceedings of the ACM Workshop on Internet of Things Security and Privacy*, 2017.

[8] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proceedings of the USENIX Security Symposium*, 2017.

[9] C. Tagliaro, M. Komsic, A. Continella, K. Borgolte, and M. Lindorfer, "Large-scale security analysis of real-world backend deployments speaking iot-focused protocols," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2024.

[10] A. Chatterjee and B. Ahmed, "Iot anomaly detection methods and applications: A survey," *IEEE Internet of Things Journal*, vol. 19, p. 100568, 2022.

[11] L. M. Zangrandi, T. van Ede, T. Booij, S. Sciancalepore, L. Allodi, and A. Continella, "Stepping out of the MUD: Contextual threat information for IoT devices with manufacturer-provided behaviour profiles," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2022.

[12] B. Zhao, S. Ji, J. Xu, Y. Tian, Q. Wei, Q. Wang, C. Lyu, X. Zhang, C. Lin, J. Wu *et al.*, "One bad apple spoils the barrel: Understanding the security risks introduced by third-party components in iot firmware," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 3, pp. 1372–1389, 2023.

[13] S. Hui, Z. Wang, X. Hou, X. Wang, H. Wang, Y. Li, and D. Jin, "Systematically quantifying iot privacy leakage in mobile networks," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7115–7125, 2020.

[14] A. M. Mandalari, H. Haddadi, D. J. Dubois, and D. Choffnes, "Protected or porous: A comparative analysis of threat detection capability of iot safeguards," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2023.

[15] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[16] L. Yu, B. Luo, J. Ma, Z. Zhou, and Q. Liu, "You are what you broadcast: Identification of mobile and IoT devices from (public) WiFi," 2020.

[17] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," in *Proceedings of the ACM Workshop on Attacks and Solutions in Hardware Security*. Association for Computing Machinery, 2018, p. 41–50.

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

[19] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[20] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AuDI: Toward autonomous IoT device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[21] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: a context-aware security framework for smart home systems," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2019.

[22] 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 *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM)*, 2017.

[23] 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 ACM Symposium on Applied Computing (SAC)*, 2017.

[24] A. Hsu, J. Tront, D. Raymond, G. Wang, and A. Butt, "Automatic IoT device classification using traffic behavioral characteristics," in *Proceedings of the IEEE SoutheastCon*. IEEE, 2019, pp. 1–7.

[25] J. Sun, K. Sun, and C. Shenefiel, "Automated IoT device fingerprinting through encrypted stream classification," in *Proceedings of the Security and Privacy in Communication Networks*, 2019.

[26] A. Aksoy and M. H. Gunes, "Automated IoT device identification using network traffic," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2019.

[27] R. R. Chowdhury, A. C. Idris, and P. E. Abas, "A deep learning approach for classifying network connected IoT devices using communication traffic characteristics," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 26, 2023.

[28] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: network device behavior modeling for identifying unknown IoT devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation (IoTDI)*, 2019.

[29] X. Hu, H. Li, Z. Shi, N. Yu, H. Zhu, and L. Sun, "A robust IoT device identification method with unknown traffic detection," in *Proceedings of the Wireless Algorithms, Systems, and Applications*, 2021, vol. 12937, pp. 190–202.

[30] M.-y. Zhu, Z. Chen, K.-f. Chen, N. Lv, and Y. Zhong, "Attention-based federated incremental learning for traffic classification in the internet of things," *Computer Communications*, vol. 185, pp. 168–175, 2022.

[31] 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*, vol. 22, no. 10, 2023.

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

[33] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "DEFT: A distributed IoT fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2019.

[34] 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.

[35] D. Ahmed, A. Das, and F. Zaffar, "Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices," in *Proceedings on the Privacy Enhancing Technologies Symposium (PETS)*, 2022.

[36] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home iot device classification using machine learning-based network traffic analysis," *IEEE Access*, 2022.

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

[38] E. Maali, O. Alrawi, and J. McCann, "Evaluating machine learning-based iot device identification models for security applications," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2025.

[39] C. Blinder and V. Velasquez, "Average number of smart devices in a home," 2024. [Online]. Available: https://www.consumeraffairs.com/homeowners/average-number-of-smart-devices-in-a-home.html

[40] "Google scholar." [Online]. Available: https://scholar.google.com/

[41] "Dblp." [Online]. Available: https://dblp.org/

[42] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "PINGPONG: Packet-level signatures for smart home devices," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2020.

[43] Y. Wan, K. Xu, F. Wang, and G. Xue, "IoTAthena: Unveiling IoT device activities from network traffic," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 651–664, 2022.

[44] T. Hu, D. J. Dubois, and D. Choffnes, "BehavIoT: Measuring smart home IoT behavior using network-inferred behavior models," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2023.

[45] B. Charyyev and M. H. Gunes, "IoT traffic flow identification using locality sensitive hashes," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2020.

[46] Y. Afek, A. Bremler-Barr, D. Hay, R. Goldschmidt, L. Shafir, G. Avraham, and A. Shalev, "NFV-based IoT security for home networks using MUD," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.

[47] A. Hamza, D. Ranathunga, H. H. Gharakheili, T. A. Benson, M. Roughan, and V. Sivaraman, "Verifying and monitoring iots network behavior using mud profiles," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, p. 1–18, 2022.

[48] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "IoT device identification via network-flow based fingerprinting and learning," in *Proceedings of the IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*, 2019.

[49] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "IoTFinder: Efficient large-scale identification of IoT devices via passive DNS traffic analysis," in *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[50] 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*, vol. 30, no. 1, pp. 394–408, 2022.

[51] M. R. P. Santos, R. M. C. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in iot ecosystems," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, 2018.

[52] 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, 2019.

[53] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun, and K. Zhang, "Your smart home can't keep a secret: Towards automated fingerprinting of IoT traffic," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2020.

[54] 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 *Proceedings of the IEEE International Conference on Communications (ICC)*, 2020.

[55] 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. 23 741–23 749, 2022.

[56] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Inferring IoT device types from network behavior using unsupervised clustering," in *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*, 2019.

[57] 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 ACM Symposium on SDN Research*, 2019.

[58] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Detecting behavioral change of IoT devices using clustering-based network traffic modeling," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7295–7309, 2020.

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

[60] P. Anantharaman, L. Song, I. Agadakos, G. Ciocarlie, B. Copos, U. Lindqvist, and M. E. Locasto, "IoTHound: environment-agnostic device identification and monitoring," in *Proceedings of the ACM International Conference on the Internet of Things (IoT)*, 2020.

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

[62] 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 *Proceedings of the International Conference on Network and Service Management (CNSM)*, 2020.

[63] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA: IEEE, 2018.

[64] "Rfc 8520: Manufacturer usage description specification." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8520

[65] E. Damiani, S. De Capitani Di Vimercati, S. Paraboschi, and P. Samarati, "An open digest-based technique for spam detection," in *Proceedings of the International Conference on Parallel and Distributed Computing Systems (PDCS)*, D. Bader and A. Khokhar, Eds., 2004.

[66] "Icore conference portal: Core2023 ranking." [Online]. Available: https://portal.core.edu.au/conf-ranks/

[67] 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 ACM Internet Measurement Conference (IMC)*, 2019.

Appendix

1. List of Devices in our Experiments

Table 5 lists all the devices used in the experimental setup.

TABLE 5: **Here is a comprehensive list of devices, along with their MAC addresses, used in our experiments.**

Device name	MAC Address
Samsung Fridge	40:ca:63:cf:0d:6a
Google Nest Cam	20:1f:3b:08:7f:e3
Bosch Dishwasher	c8:d7:78:52:31:90
Smartplug - KP105	00:31:92:e1:7b:17
Smartplug - HS110	74:da:88:5d:4c:3d
Philips HUE	ec:b5:fa:a8:e5:09
iPad 2022	ea:42:72:ea:f0:18
SALCAR Radiatorthermostaat TRV801W	1c:90:ff:16:a7:fe
Philips Luchtreiniger 600-Serie	e4:bc:96:03:8d:bb
Philps TV	10:b2:32:d2:df:48
Chromecast v3	66:39:cf:73:04:d5
PNI Safe House PG600LR Safety Alarm	94:b9:7e:06:1a:57
Echo dot	28:73:f6:67:73:a9
Calex motion sensor	a4:cf:12:e7:06:24
Samsung phone	56:61:8d:20:86:b0
Ring doorbell	34:3e:a4:71:a0:1f
Nest Smokealarm	18:b4:30:f4:0e:58
Google Home Mini	d4:f5:47:36:e0:c1
Amazon Echo	fc:49:2d:58:bd:3c
Princess Smart Aerofryer	3c:39:e7:28:fd:5c
Calex Slimme LED Vloerlamp	1c:90:ff:e7:ca:f1
ilife t10s	bc:fd:0c:d1:ad:5f
WellToBe pet feeder	b8:06:0d:78:0f:8a
Watersensor	1c:90:ff:5c:1e:52
PS5	ec:74:8c:5f:4b:90

2. Validation of Re-implemented Works

2.1. IoT Sentinel. We evaluated our re-implementation of IoT Sentinel using the original dataset, since it was openly available. The performance values obtained per device are in line with the values reported in the paper. Since we randomly split between the training and testing datasets, the values obtained will not be the same as those reported. These differences lead to differences in the overall performance values: **Precision: 0.731773, Recall: 0.712909, F1-score: 0.708118** The performance for each device is as shown in the figure 3.

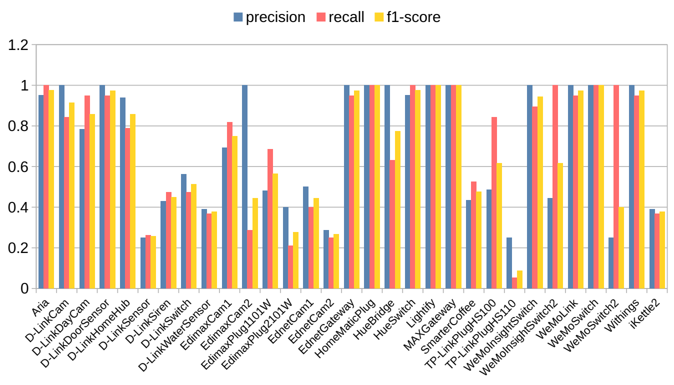


Figure 3: Validation result of IoT Sentinel

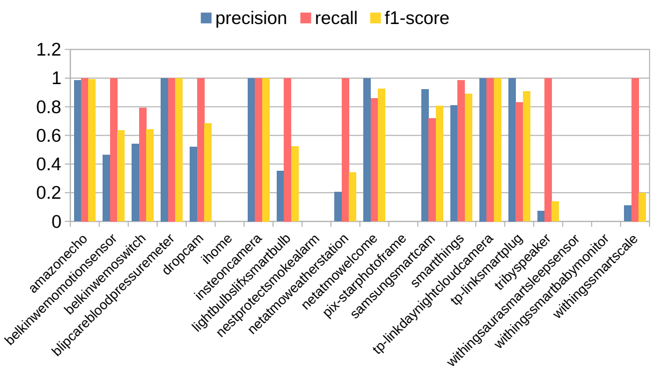


Figure 4: Validation result of MUDgee

2.2. MUDgee. The original dataset used to evaluate MUDgee is not publicly available; therefore, we utilized a different dataset provided by the same research group. Since we are working with a different dataset, we could not compare our performance to the reported values. The overall performance values obtained from our evaluations are: **Precision: 0.82123, Recall: 0.89465, F1-score: 0.83953**. The performance of each device is shown in Figure 4.

2.3. Your Smart Home. We evaluated the re-implementation of Your Smart Home using the original dataset, since it was openly available. The performance results for the bi-directional LSTM were in line with the values reported in the paper, i.e.: **Precision: 0.95853, Recall: 0.95840, F1-score: 0.95838** The performance for each device is shown in the figure 5.

2.4. Devicemein. We implemented the machine learning models outlined in the Devicemein paper [28] for our experiments. The authors evaluated the original approach using the UNSW dataset [32] and a private dataset, focusing solely on identifying unknown devices. As a result, the paper did not present any results related to identifying known IoT devices, which means we could not compare our performance metrics with theirs. We used the UNSW dataset to evaluate our re-implementation, and the overall performance values are

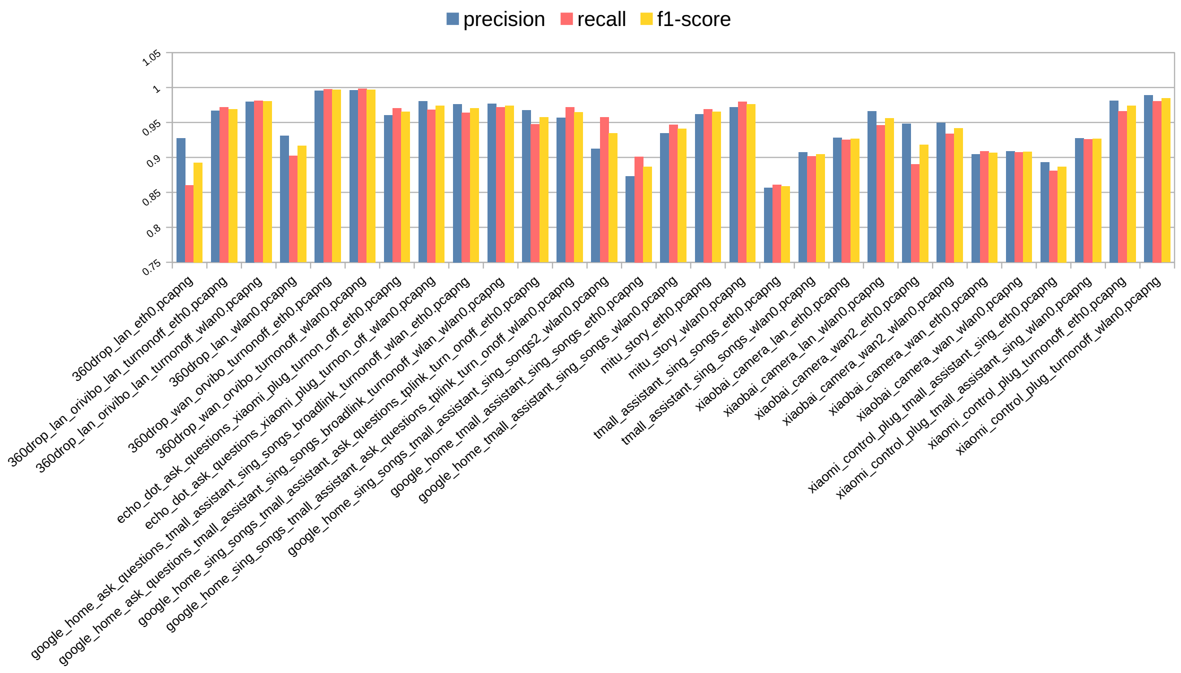


Figure 5: Validation result of Your Smart Home

as follows: **Precision: 0.74655, Recall: 0.79929, F1-score: 0.76525**

3. Detailed Results of Scalability Test

Here, we report the detailed performance metrics for the scalability experiments described in Section 4.4. Table 6 reports on the results when using the MonIoTr-US dataset for the scalability experiment. Table 7 reports on the results when using our Lab dataset for the experiment.

4. Transferability Across Countries

Aim. With these experiments, we aim to identify how transferable the trained models of the IoT device identification methods are across different countries. The transferability depends on the extent to which the patterns in the network traffic vary in different countries. These variations can include changes in the servers contacted by the devices, changes in port numbers, and more. Maali et al. [38] conduct experiments to evaluate Spatial Transferability using datasets collected in the US and the UK. Here, we compare our results with the ones reported by Maali et al. [38].

Experiment Settings. We use publicly available data from the MonIoTr lab [67], which collected network traffic in similar laboratory settings in the US and the UK. The data collected in the US lab consists of network traffic from 46 IoT devices. The data from the UK lab consists of network traffic from 35 IoT devices. The two datasets have 26 devices in common. We evaluate all the methods using these two datasets. The results of these evaluations are in Table 8.

Observations. First, we establish a baseline performance for the six works. Since we have data from the US and UK, we established the baseline performance in both countries by training and testing with data from the same country. Table 8 shows that *IoT Sentinel*, *MUDgee*, and *Devicemein* perform poorly on both datasets, with F1-scores between 0.313 and 0.515. In contrast, *genIoTID* achieves F1-scores above 0.83, while *Your Smart Home* and *IoTDevID* perform best, with F1-scores over 0.95.

Next, we evaluate all the works with data from another country, i.e., training with US data and testing with UK data, and vice versa. *IoT Sentinel*'s F1-score drops when trained on UK data and tested on US data. Conversely, training on US data and testing on UK data slightly improves the F1-score but reduces precision. This discrepancy is due to the UK dataset having more traffic (i.e., more packets) for most devices. *IoT Sentinel*'s performance depends on the amount of available traffic, explaining its behavior on the UK dataset. *MUDgee*'s precision and F1-score experience a drop ranging between 11% and 39.8% respectively when tested with data from another country. Due to these differences in the performance variations, we cannot extract a pattern to generalize the transferability of *IoT Sentinel* and *MUDgee*.

The behavior patterns of *Your Smart Home*, *IoTDevID*, *genIoTID*, and *Devicemein* are similar when trained and tested with different country datasets. While the F1-score drops more for *Your Smart Home*, *IoTDevID*, and *genIoTID* when trained with US data and tested with UK data than vice versa, the opposite is true for *Devicemein*. When evaluated on the combined US and UK dataset, all methods showed improved performance, with Your Smart Home achieving its

TABLE 6: The **weighted average** of Precision, Recall, and F1-score for different number of devices using MonIoTr-US data ([67]).

No. devices	IoT Sentinel			MUDgee			Your Smart Home			IoTDevID			genIoTID			Devicemein		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
7	0.859	0.729	0.780	0.715	0.571	0.555	0.999	0.999	0.999	1.0	1.0	1.0	0.845	0.841	0.837	0.805	0.861	0.823
14	0.585	0.449	0.493	0.648	0.372	0.399	0.987	0.987	0.987	0.985	0.985	0.984	0.786	0.780	0.769	0.461	0.562	0.495
21	0.602	0.378	0.416	0.727	0.576	0.597	0.986	0.986	0.986	0.991	0.990	0.990	0.807	0.796	0.796	0.481	0.530	0.464
28	0.602	0.367	0.411	0.686	0.483	0.517	0.991	0.991	0.991	0.995	0.994	0.994	0.799	0.784	0.784	0.607	0.624	0.562
35	0.616	0.300	0.345	0.680	0.480	0.485	0.988	0.988	0.988	0.994	0.993	0.993	0.842	0.828	0.827	0.498	0.551	0.493
46	0.613	0.275	0.324	0.685	0.352	0.390	0.986	0.986	0.986	0.968	0.967	0.966	0.857	0.843	0.842	0.504	0.586	0.514

TABLE 7: The **weighted average** of Precision, Recall, and F1-score for different number of devices using Lab Data.

No. devices	IoT Sentinel			MUDgee			Your Smart Home			IoTDevID			genIoTID			Devicemein		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
7	0.758	0.726	0.725	0.967	0.720	0.809	0.948	0.946	0.946	0.990	0.990	0.990	0.833	0.818	0.816	0.521	0.596	0.544
14	0.817	0.701	0.739	0.852	0.557	0.641	0.966	0.964	0.964	0.979	0.979	0.979	0.815	0.785	0.789	0.470	0.581	0.507
21	0.779	0.641	0.684	0.753	0.579	0.605	0.963	0.961	0.961	0.989	0.982	0.981	0.911	0.877	0.885	0.595	0.627	0.594
23	0.746	0.595	0.642	0.741	0.609	0.631	0.957	0.954	0.955	0.983	0.983	0.982	0.909	0.875	0.883	0.576	0.628	0.586

TABLE 8: The **weighted average** of Precision, Recall, and F1-score for data from US and UK Labs (MonIoTr lab [67]).
For each method, red highlights the lowest performance values and green the highest values.

Train	Test	IoT Sentinel			MUDgee			Your Smart Home			IoTDevID			genIoTID			Devicemein		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
US	US	0.609	0.265	0.313	0.626	0.378	0.364	0.986	0.986	0.986	0.968	0.968	0.967	0.862	0.847	0.846	0.495	0.593	0.515
US	UK	0.438	0.325	0.341	0.513	0.375	0.324	0.609	0.557	0.569	0.581	0.592	0.581	0.608	0.499	0.495	0.318	0.284	0.274
UK	UK	0.605	0.470	0.498	0.556	0.328	0.343	0.988	0.988	0.988	0.963	0.957	0.951	0.848	0.838	0.836	0.447	0.523	0.461
UK	US	0.352	0.280	0.276	0.335	0.288	0.296	0.854	0.832	0.824	0.610	0.699	0.632	0.684	0.645	0.617	0.225	0.394	0.265
US + UK	US + UK	0.609	0.375	0.415	-	-	-	0.989	0.989	0.989	0.949	0.943	0.938	0.863	0.845	0.844	0.414	0.512	0.426

highest F1 score in this context. Ren et al. [67] found that US lab devices primarily communicated with US servers, while UK lab devices communicated with US and EU servers. Thus, the UK dataset is, in a way, a superset of the US dataset. Our results also align with those observed by Ahmed et al. [35] in their study to understand the impact of fingerprinting devices across different geological locations. They used the same dataset as us (MonIoTr [67]) for their evaluations. While they only studied the impact of different country datasets on the generalizability of features used for fingerprinting, we identified how it impacts the performance of existing methods. These findings show that all methods experience performance drops when trained and tested on data from different countries. However, the impact varies based on the diversity of traffic in the training dataset, as seen with the UK dataset.

Comparison. Maali et al. [38] perform these spatial transferability experiments using datasets collected in their labs in the US and UK. They establish their baseline using a combined US and UK dataset, contrasting our separate baselines for each country. The works they selected for empirical evaluations differ from the ones we selected, with only two in common, i.e., *Devicemein* and *Your Smart Home*. Our observations for these two works in a multi-country setting do not align with those of Maali et al. Maali et al. observed the complete inability of these two works to generalize across countries, with their AUCPR reducing by 73.68%. However, we observed the F1-score of *Your Smart Home* and *Devicemein* dropped by 42.2% and 46.6% when trained on US data and tested on UK data. Both works experience a lower drop in the opposite scenario: *Your Smart Home* at 16.6These differences may be attributed to variations in our re-implementations of these works, the datasets used for evaluations, or both.

Two of the works we evaluate, *IoT Sentinel* and *genIoTID*, use the Random Forest algorithm, similar to two works evaluated by Maali et al. Due to inconsistent performance from *IoT Sentinel*, we will only use *genIoTID* for comparison. The performance patterns of *genIoTID* across the two countries match that of Sivanathan et al. [32], as shown by Maali et al.