Revisiting IoT Device Identification

Roman Kolcun
University of Cambridge
roman.kolcun@cl.cam.ac.uk

Diana Andreea Popescu
University of Cambridge
diana.popescu@cl.cam.ac.uk

Vadim Safronov
University of Cambridge
vadim.safronov@cl.cam.ac.uk

Poonam Yadav
University of York
poonam.yadav@york.ac.uk

Anna Maria Mandalari
Imperial College London
anna-maria.mandalari@imperial.ac.uk

Richard Mortier
University of Cambridge
richard.mortier@cl.cam.ac.uk

Hamed Haddadi
Imperial College London
h.haddadi@imperial.ac.uk

***Abstract*—**
Internet-of-Things (IoT) devices are known to be the source of many security problems, and as such, they would greatly benefit from automated management. This requires robustly identifying devices so that appropriate network security policies can be applied. We address this challenge by exploring how to accurately identify IoT devices based on their network behavior, while leveraging approaches previously proposed by other researchers.

We compare the accuracy of four different previously proposed machine learning models (tree-based and neural network-based) for identifying IoT devices. We use packet trace data collected over a period of six months from a large IoT test-bed. We show that, while all models achieve high accuracy when evaluated on the same dataset as they were trained on, their accuracy degrades over time, when evaluated on data collected outside the training set. We show that on average the models' accuracy degrades after a couple of weeks by up to 40 percentage points (on average between 12 and 21 percentage points). We argue that, in order to keep the models' accuracy at a high level, these need to be continuously updated.

***Index Terms*—IoT, network traffic, machine learning, random forests, neural networks**

I. Introduction

Internet-of-Things (IoT) devices are the source of many security threats, particularly in domestic deployments [1]. To counter such threats, these devices would benefit from active and, in particular, automated management. However, automating such tasks requires robustly identifying devices to be able to apply appropriate policies, actions, and updates. In this environment, the natural way of identifying IoT devices is to analyze their network behavior at the home router: devices cannot hide behavior as, by definition, they must interact over the network in order to provide functionality. Performing analyses of network behavior at the home router is robust in terms of privacy, scalability and not relying on dependencies from manufacturer-provided cloud-services. Furthermore, there is already nascent support for summarizing such analyses' results via the MUD standard [2].

Previous work has resorted to machine learning to carry out IoT device identification. The usual approach entails training machine learning models offline or in a cloud environment [3]–[6], and run inference to identify the devices at the home routers. However, the training and validation of these models is done on a particular set of devices, and for a limited time period, thus achieving high accuracy. These models' accuracy may drop when evaluated on different IoT datasets or require continuous retraining to maintain the level of accuracy needed.

To investigate it in more detail, we evaluate three IoT device identification approaches from the literature on the same dataset: (*i*) a two-stage Random Forest classifier using features extracted from a 1 hour window of collected traffic [5], (*ii*) a 2D Convolutional Neural Network on a stream of raw packets [7], [8]; and (*iii*) Random Forest and Decision Tree classifiers on features extracted from 1 second window of network traffic [9]. We also propose new IoT device classification models, a Random Forest classifier and a Fully Connected Neural Network trained on features extracted from TCP/UDP flows. We evaluate in total six algorithms using four sets of features. We train and evaluate them on a 27 week-long dataset that we gathered from a large IoT test-bed comprising 41 IoT devices, split into three time periods for training, while the evaluation was done on the whole period. We show that IoT device identification models have high accuracy only when the training and inference is run on the same dataset. Our findings prove that static, pre-trained models cannot be used for identification across different home IoT networks while ensuring high accuracy. Thus, we conclude that it is paramount to update the models at the edge with new incoming data.

The main contributions of the paper are as follows:

- We gather a large measurement dataset of 27 weeks from a large IoT test-bed containing 41 IoT devices.
- We study and compare six different machine learning models for IoT device classification in terms of their accuracy using the dataset for training and inference.
- We show that in all six cases the accuracy decays over time, demonstrating the need for updating models at the edge.
- We release the extracted features (pre-processed IoT data) used to train the machine learning algorithms in order to further research in this area in the community[1].

978-3-903176-40-9

[1]https://github.com/DADABox/revisiting-iot-device-identification

II. Related Work

In the last decades, a vast number of machine learning-based network monitoring and Internet traffic classification techniques, both in a distributed and centralized manner, have been explored [10]–[12]. However, not all methods are suitable for IoT, and some of these techniques are adopted and customized for IoT; therefore, in this section, we focus only on techniques used for analyzing IoT traffic.

Traffic Classification for IoT. Offline IoT network traffic analysis is used for understanding various IoT device or user behaviors [13]–[15]. For example, Yadav *et al.* [16] studied traffic from a dozen IoT devices in a lab environment to understand network service (*e.g.,* DNS, NTP) dependencies and robustness of device function when connectivity is disrupted. Apthorpe *et al.* [14] analyzed the traffic rates of four IoT devices, showing that observations about user behavior can be inferred even from encrypted traffic. Similarly, traffic categorization using both statistical and machine learning techniques has been performed by Amar *et al.* [17].

Device Identification and Anomaly detection in IoT. The IoT device identification is the first step towards finding any malicious or unknown IoT device in the network. Generally, many IoT devices have a unique identifier assigned during manufacturing such as MAC address or hardware serial numbers. Even though these unique addresses could reveal some information about the device manufacturer, still the full identification of malicious/abnormal devices in the network using only these unique addresses is not possible. Thus, behavior-based IoT device identification methods, which use traffic classification mechanisms have gained attention recently [3], [4], [18]–[20]. The IoT applications, *e.g.,* anomaly detection and prediction, require low latency and privacy at the edge, and traffic based behavior identification is needed to be done in the real-time at the gateway level for security and data privacy purpose [21]–[23].

Machine Learning for Device Identification. Machine Learning in IoT at the edge is still in its infancy, due to partly lack of available network data in the wild and lack of compact machine learning models. The recent uptake in resource-constrained machine learning [24]–[27] has led to a renewed interest in applying machine learning to IoT network-related problems, specifically network traffic classification [28]–[30], anomaly detection [27], [31] and device identification [3]–[5], [18], [32]. Sivanathan *et al.* [5] used multi-stage classifiers (Naive Bayes Multinomial and Random Forest Classifier (RFC)) for IoT device classification and achieved accuracy from 99.28% to 99.76% with classifiers trained on 1 to 16 days data from 28 unique IoT devices. The high accuracy achieved by this algorithm makes it the first choice for evaluation in our work.

Nguyen *et al.* [31] trained Gated Recurrent Network (GRU) for federated learning for anomaly detection using 33 devices categorized in 27 categories and for evaluation, deployed 13 devices and found only 5 are vulnerable to the Mirai attack when the attack is injected in the local network. The attack was detected within 30 minutes. Many identification works train machine learning models offline or in a cloud environment [3]–[6] and run inference to identify IoT devices on local gateways. The training and evaluation is done only for a set of devices for a limited time period, thus inference achieves a good accuracy when testing data is similar to the training data. However, for real world scenarios, a pre-trained model on a small set of devices would not work on a large set of unknown IoT devices. The identification accuracy may drop when the inference data is different from the training dataset, therefore, requiring retraining of the model for the local setup. None of the works above have looked at or addressed this problem; we are not only investigate retraining requirements for maintaining high identification accuracy over an extended period but also investigate the comparative performance of the IoT identification algorithms on the same dataset.

TABLE I: Categorized IoT devices in our test-bed.

Category	Device Name
Surveillance	Blink camera, Bosiwo camera, D-link camera, Reolink camera Ring doorbell, UBell doorbell, Wansview camera, Yi camera, LeFun camera, ICSee doorbell
Media	Apple TV, Fire TV, Roku TV, LG TV, Samsung TV
Audio	Allure speaker, Echodot, Echospot, Echoplus, Google home
Hub	Insteon hub, Lightify hub, Philips hub, Smartthings hub, Xiaomi hub, Switchbot hub, Blink security hub
Appliance	Smart Kettle, Smarter coffee machine, Sousvide cooker, Xiaomi rice cooker
Home automation	Honeywell thermostat, Nest thermostat, Netatmo weather station, TP-link bulb, TP-link plug, Wemo plug, Xiaomi plug, Smartlife remote, Smartlife bulb, Meross door opener

III. Dataset and Models

A. Dataset

To capture data, we built a test-bed that currently comprises 41 different IoT devices. Table I describes the devices in our test-beds, by category. In each category, these devices were chosen as the most popular one by a large online retailer.

In the test-bed, in addition to the devices, a Linux server running Ubuntu 18.04 with two Wi-Fi cards for 2.4 GHz and 5 GHz connections, plus two 1 Gbps Ethernet connections for LAN and Internet connectivity are part of the setup. The server sits outside of any firewall and has a public IPv4 address. However, to match a regular home network environment, all IoT devices are behind a NAT setup and cannot be accessed directly from the Internet. The monitoring software automatically detects the connection of a new device to the network, assigns it a local IP address, and starts capturing packets using *tcpdump*. Each device's traffic is filtered by MAC address into separate files. Each file is stored in a `pcap` format.

Each device is assigned a unique *device ID*. In some cases, we extract features from TCP/UDP flows. Each flow is

identified by a 5-tuple (`source IP address, source port, destination IP address, destination port, transport protocol`). Each device generates multiple flows. When a flow is extracted from the `pcap` file, it is assigned a *device ID*. Each flow is classified independently of the other flows generated by the same or other devices.

Data were collected over a period of 27 weeks. During this period the interaction with the devices was rather sporadic. Given that the devices were deployed in the lab with people present during the daytime, some of the devices might have been occasionally activated, *e.g.,* cameras with motion detection or a smart speaker reacting to the user's request. However, these activations were rather rare. Researchers did not perform any software update that required manual intervention, *i.e.,* confirming the update using the accompanying app. However, some devices might have updated themselves without researchers' knowledge (*e.g.,* Amazon Alexa devices update themselves automatically). Researchers also sporadically checked whether the devices are still connected to the Internet and functioning properly. This check has been done via a cell phone using an accompanying app.

We are aware that the test-bed setup is not an approximation of a usual household environment. In the household environment the interaction with the devices could be expected to occur more often and probably follow some pattern. In our test-bed scenario, these devices were mostly idle. Additionally, our test-bed contains IoT devices only, *i.e.,* non-IoT devices such as laptops, computers, or cell phones are not present. The only cell phones connected to the test-bed are used to control the IoT devices. The traffic generated by these cell phones are stored in separate `pcap` files and are not included in the training dataset.

To evaluate whether the model can reliably classify IoT devices, we split the collected data into three periods, each 9 weeks long. We then train each ML model using data from only one period, while evaluating them on the whole dataset of 27 weeks.

B. Processing Traces

In this subsection, we list four different IoT device identification models that we analyzed and describe how data were pre-processed for each of them.

1) Multi-stage RFC with 1 Hour Window: This is one of the pioneering work in IoT device identification by Sivanathan *et al.* [5]. This classifier extracts features from a traffic collected over the duration of one hour. This two-stage approach uses the combination of Naive Bayes Multinomial (NBM) classifier and the Random Forest classifier (RFC). In the first stage, NBM is used to classify bag of words, *i.e.,* contacted domain names, port numbers, and cipher suites used. In the second, stage the output from the first stage and the following features from the 1 hour window of traffic are extracted: flow volume, flow duration, flow rate, sleep time, DNS interval, and NTP interval. These features are summarized in Table II. After processing the captured traffic, our dataset contains 130,460 records.

TABLE II: List of features extracted from 1 hour window of network traffic used for training of two-phase RFC models.

Feature Name	Feature Description
First Stage	Using Naive Bayes Multinomial Classifier
bag_of_ports	list of ports contacted
bag_of_domains	list of domains contacted
bag_of_ciphers	list of used cipher suites
Second Stage	Using Random Forest Classifier
flow_volume	volume of flow
flow_duration	duration of flow
flow_rate	rate of flow
sleep_time	sleep time
dns_interval	interval between DNS requests
ntp_interval	interval between NTP requests
ports_class	class for bag of ports (1^{st} stage)
ports_confidence	confidence for bag of ports (1^{st} stage)
domain_class	class for bag of domains (1^{st} stage)
domain_confidence	confidence for bag of domain (1^{st} stage)
cipher_class	class for bag of cipher suites (1^{st} stage)
cipher_confidence	confidence for bag of cipher suites (1^{st} stage)

TABLE III: List of features extracted from 1 second window of network traffic for training of RFC and DTC models.

Feature Name	Feature Description
bytes_sum	sum of bytes of all packets
bytes_avg	average size of packets
bytes_std	standard deviation of size of packets

2) 2D Convolutional Neural Networks on Raw Packet Data: Convolutional networks (CNN) were used in image classification for a very long time. Their advantage is that they are capable of automatic extracting of features. Researchers have been using them to process raw packets to either classify an IoT device [7] or a network flow [8]. For each TCP or UDP

TABLE IV: List of features extracted from TCP/UDP flows used for training of RFC and FC-NN models.

Feature Name	Feature Description
src_port	source port
dest_port	destination port
bytes_out	number of bytes sent
bytes_in	number of bytes received
pkts_out	number of packets sent
pkts_in	number of packets received
ipt_mean	mean of inter-packet interval
ipt_std	standart deviation of inter-packet interval
ipt_var	variance of inter-packet interval
ipt_skew	skewness of inter-packet interval
ipt_kurtosis	kurtosis of inter-packet interval
b_mean	mean of packet sizes
b_std	standard deviation of packet sizes
b_var	variance of packet sizes
b_skew	skewness of packet sizes
b_kurtosis	kurtosis of packet sizes
duration	duration of the stream
protocol	protocol ID
domain	second and top level domain

flow first X packets are order in rows. Then for each packet first Y bytes are extracted. If the packet is smaller than Y or there are fewer than X packets in the flow, the empty space is padded with zeroes. Additionally, fields from the IP header which can uniquely identify the device, such as source MAC or IP address, are replaced with zeroes. This yields an input grid of the size $X \times Y$ which is passed to the CNN. In this paper we used $X = 10$ and $Y = 250$ [7]. After processing all TCP/UDP flows, our dataset consists of 19,771,368 input grids.

The evaluated convolutional network consisted of the following layers: Convolutional, MaxPooling, Convolutional, MaxPooling, Flatten, Dropout, and an output Dense layer.

3) Multiple Classifiers on 1 Second Window: In this case researchers extracted three features from each second of traffic generated by an IoT device [9]. These features are the sum of the size of all packets, the average size of a packet, and the standard deviation of the size of a packet. These features are summarized in Table III. The researchers then evaluated five different classifiers: Random Forest, Decision Tree, Support Vector Classifier, K-Nearest Neighbors, and Voting Classifier. The Voting Classifier used four previous classifiers to classify the 1 second network trace by choosing the device with most votes. After processing the captured traffic, our dataset contains 75,396,781 records.

4) RFC and FC-NN on TCP/UDP Flows: We also propose a new IoT device classification system based on features extracted from TCP and UDP flows. We processed each `pcap` file using *joy*[2] utility which extracts the following features from each TCP/UDP network flow (summarized in Table IV): source and destination IP address, source and destination port number, number of packets sent and received, bytes of packets sent and received, starting and ending time of the flow. Additionally, *joy* extracts DNS request and replies which can be later analyzed. Flow features are extracted if the network flow is inactive for more than ten seconds, or if the network flow is active for more than 30 seconds. If the network flow continues, a new record is created. It means, that a set of features is extracted at latest after 30 seconds.

The extracted features contain also information about the first up to *N* packets. We used the default value of $N = 50$. This information includes data about packet sizes and inter-packet intervals. Using information about packets, additional features are computed, *i.e.,* duration of the flow, and for both, packet sizes and inter-packet intervals, mean, standard deviation, variance, skew, and kurtosis is computed. Each flow is assigned the *device ID*.

The list of DNS responses is used to map IP addresses to domain names. We chose not to use IP addresses as a feature because they may not be consistent due to the nature of the services running in cloud. A virtual server may migrate to another physical server and its IP may change. Or a new server might be temporarily started to balance the load. Additionally, many large manufactures are using DNS load balancing where the same domain is translated to different IP addresses. Therefore, we decided to use the domain name as a feature. However, we noticed that many times the domain name differs on the third or further level. This is especially common when a content delivery network is contacted. Therefore, we decided to use only the second and top level domain name as a feature. In our dataset we identified only 153 unique second and top level domain names. The final dataset contains 60,653,581 records.

The dataset is evaluated using Random Forest classifier and Fully Connected neural network. The fully connected network consists of a Dense input layer, two hidden Dense layers, and an output Dense layer.

[2]https://github.com/cisco/joy

IV. Evaluation

In this section we evaluate the selected machine learning models on various data. The dataset spans over the period of 27 weeks. The dataset is split into three equally-sized periods covering weeks 1-9, 10-18, and 19-26 respectively. Each model is trained on one period using stratified sampling with 80%-20% split between the training and testing set. For the evaluation, the whole dataset spanning 27 weeks is split into 1-week chunks and each week is evaluated as a whole.

Machine learning based classifiers (NBM, RFC, and DTC) are implemented using python scikit-learn library[3]. All classifiers were using default settings. Neural network based classifiers were implemented using Keras library[4]. All models were trained for the duration of 50 epochs and the model with the highest accuracy was chosen for the evaluation.

Data pre-processing and feature extraction is described in (§III-B). All models were evaluated on a server using two Intel Xeon Gold 6132 CPU @ 2.6 GHz with 256 GB of RAM running Ubuntu 18.04 operating system.

We used a standard evaluation metric [33], F_1 score for the overall measure of the models accuracy and is defined as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

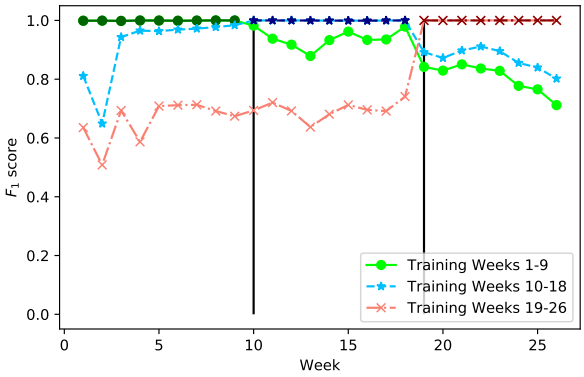
$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, F_1 \in \langle 0, 1 \rangle$$

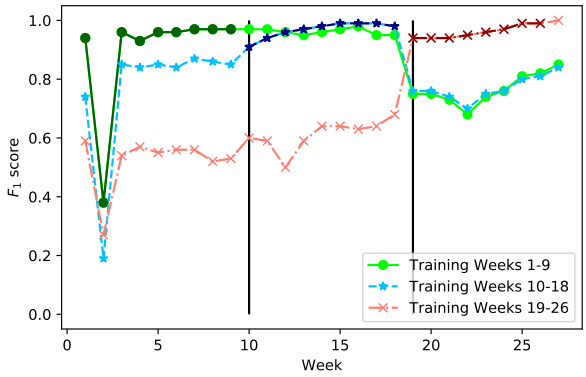
where $True\,Positive$ represents the number of times when a device was correctly classified, $False\,Positive$ the number of times when the device was classified as some other device, $True\ Negative$ the number of times when the device is correctly not classified, and $False\ Negative$ the number of times when a device is classified instead of a correct device. F_1 score represents a harmonic mean of precision and recall.

[3]https://scikit-learn.org/

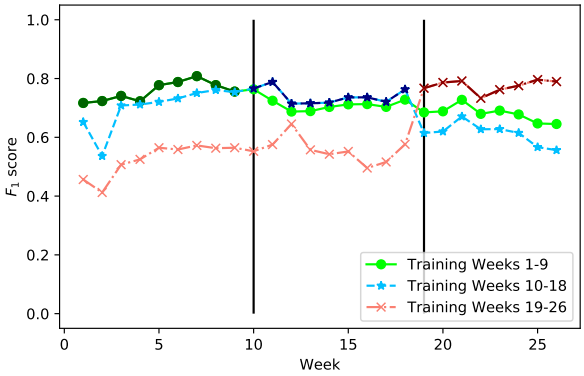
[4]https://keras.io/



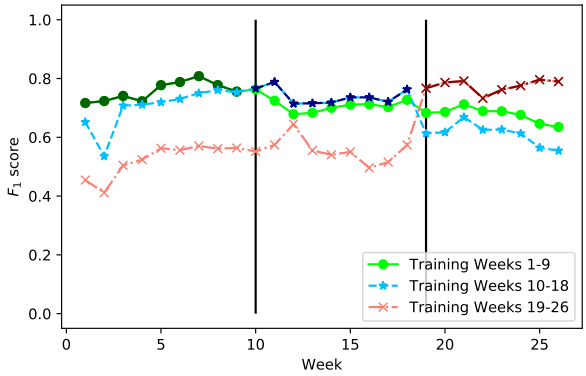
(a) 1 hour window with 2-stage NBM + RFC



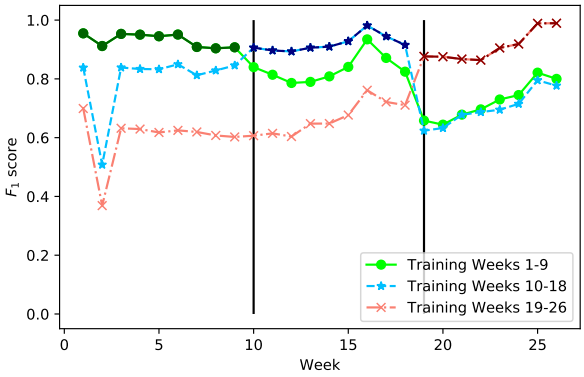
(b) Raw packets with 2D Convolutional Network



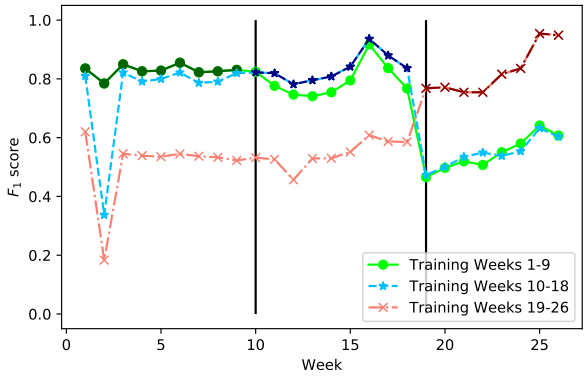
(c) 1 second window with RFC



(d) 1 second window with DTC



(e) TCP/UDP flows with RFC



(f) TCP/UDP flows with Fully Connected NN

Fig. 1: F_1 score of various classifiers on various datasets. Each line corresponds to a different training window (*i.e.,* weeks 1-9, 10-18, or 19-26). The darker color shows the period when the model is evaluated on the training data. Vertical bars splits the figures into three separate periods.

A. Multi-stage RFC with 1 Hour Window

The model relies on two stage evaluation and is evaluated on statistical data collected over the period of 1 hour [5]. The evaluation of this model can be seen in Figure 1a. This model achieves the near-perfect F_1 score when evaluated on the same data as the training set. Even if the dataset is split into 80%-20% between training and testing data, because the testing data is chosen from the same time period as the training data, the model achieves very high F_1 score.

Because the features are extracted from a network traffic over a longer period of time (*i.e.,* 1 hour), the model can achieve higher accuracy for devices with a relative small and regular network footprint, *e.g.,* a weather station. However, devices which rely more on the interaction with a user or its environment, such as smart speakers or doorbell cameras with a motion detection, can produce various amount of traffic during different parts of a day. Additionally, this approach relies on knowing all the contacted domains, port numbers, and used cipher suites in advance. This knowledge cannot always be obtained in advance as a firmware update might change any or all of these variables. Additionally, domain names might be automatically generated by a load balancer.

However, when the same model is evaluated on a dataset on another time period than the training period, the accuracy of the model degrades with the absolute time difference between the training and the testing dataset. This applies to all three training periods. On average, the F_1 score decreases by 18 percentage points when comparing the performance of models evaluated on the training period vs. other periods.

B. 2D Convolutional Neural Networks on Raw Packet Data

Several researchers have used 2D Convolutional Neural Networks (CNN) to automatically extract features and perform classification of the IoT devices or network flows [7], [8]. These approaches are based on ordering the packets into a grid of a fixed size and passing it to the 2D CNN. Usually, these multiple CNN layers are followed by other layers which can slightly increase the accuracy of classification. Because of the computational complexity and time restrictions, we evaluated a simpler model consisting of 2D CNN layers only. However, we believe that the results can be extrapolated to other more complex models.

Figure 1b shows performance of the CNN in IoT device classification. Similar to the two-stage RFC, the best F_1 score is achieved when evaluated on the dataset from the same period as the training dataset. Surprisingly, the accuracy is rather high and is very close to the two-stage RFC. Additionally, CNN can achieve such high F_1 score using only a small subset of packets from each flow, as opposed to collecting and analyzing traffic from an hour long window. CNN also do not require to know any information in advance (*e.g.,* contacted domain names) or manual extraction of features. The results show that CNNs are a promising technique for IoT device classification.

However, similar to other approaches, the high accuracy is achieved only when evaluated on the same period as the training set. Outside of the training period, the accuracy gradually degrades, even though not as quickly as with other models. The average difference in the F_1 score between the models evaluated on the training dataset and other periods is 21 percentage points.

The models were trained for 50 epochs with batch size of 128. They achieved their peak accuracy very close to the 50^{th} epoch.

C. Multiple Classifiers on 1 Second Window

Authors of this model showed that even a simple statistics on a 1 second window worth of network traffic can achieve rather high classification accuracy [9]. The advantage of this approach is its simplicity, very low computational and memory overhead, and the fact that it does not need to know any prior information (*e.g.,* contacted domain names). The researchers showed, that RFC is the best single model, however, by combining results from multiple different machine learning models, it is possible to achieve even higher accuracy. In this case they combined four different ML models: RFC, DTC, SVC, and K-NN. Even though we tried to replicate these results, because our dataset was incomparably larger than the one used by authors, we were not able to finish the training of the SVC and K-NN models. Therefore, we present here the results using the RFC (Figure 1c) and the DTC (Figure 1d) models only.

Figures show that the F_1 score achieved by both models is virtually the same. However, it can be seen the similar trend as previously. The highest accuracy models achieve when they are evaluated on the data from the same period as they were trained on. However, when they are evaluated on the dataset outside their training period, their accuracy degrades over time. This is most visible with the model trained on the last period. When this model is evaluated on first two periods, it achieves significantly lower F_1 score. On average, the F_1 score is lower by 12 percentage points in the case of RFC and 13 percentage points in the case of DTC.

D. RFC and FC-NN on TCP/UDP Flows

The last evaluated approach is based on TCP/UDP flows. We have trained two classifiers on these datasets: RFC (Figure 1e) and the Fully Connected NN (Figure 1f). As can be seen in the figures, the overall performance of both models is rather similar, with the difference that RFC performs on average by 7 percentage points better than the NN model. As expected, also these two models show similar F_1 score degradation when evaluated outside their training period. On average, the RFC model achieves F_1 score 20 percentage point lower when compared to evaluation on the training period, while the F_1 score of Fully Connected NN is lower on average by 21 percentage points.

Interestingly, the size of the RFC models ranged between 100-150 GB which made it challenging to evaluate. Even though the neural network models were trained for 50 epochs, they achieved their peak accuracy no later than in 11^{th} epoch.

Since then, the accuracy was constantly decreasing and by the 50^{th} epoch it was more than 10 percentage points lower.

V. Discussion

Figure 1 shows that no matter which classifier is used, either based on classical machine learning techniques or on neural networks, they all lose their accuracy over time. While their accuracy is reasonably high when evaluated on the data from the same time period as they were trained on, once they are used for evaluation on data outside of this period, their accuracy decreases with the absolute difference in time.

The same applies to the set of extracted features (or the lack of them). Whether the features are extracted from an hour long window with some prior knowledge of data, a simple one second window statistics, a set of raw bytes from packets, or an elaborate features extracted from TCP/UDP flows - all of these features lead to accuracy degradation when evaluated outside of the training dataset.

What is also important is that the data produced for this experiment are from an academic test-bed deployed in a rather controlled environment with a very small interaction of researchers with the test-bed. Thus, we can assume that data produced by various sets of devices in users' home will vary much more and therefore the models might degrade even faster.

Because the dataset was obtained from a test-bed in a controlled environment, the network traces contain mostly background traffic of these devices. In a home environment with a regular interaction with the devices the number of different network flows will increase, which will lead to potentially larger search space and lower accuracy.

Interesting behavior can be seen when comparing the F_1 score of *all* models outside their training period. This fact is the most visible with the models trained on data from period one and two (green and blue line, respectively) and evaluated on data from period three. For *all* models and *all* various features, these two lines are almost parallel. Similar behavior can also be seen with red and blue lines in period one and red and green line in period two. This fact suggests that the network traffic produced by the devices changes over time and is not a problem of a particular feature extraction or a machine learning model.

Another interesting data point is the second week where the F_1 score of most of the models suddenly drops. However, if the data are included in the training set (dark green line), most of the models are capable to learn this anomaly and perform with high accuracy. This has one obvious exception - 2D Convolutional Neural Network (Figure 1b) whose F_1 score drops to less than 40%. Again, because this behavior is consistent across various models and features, it can be assumed that the pattern of the network traffic of the devices has significantly changed.

Because the similar decrease in classification accuracy over time can be seen across different machine learning models and using different set of features, it suggests that the problem of IoT device identification is not a problem of a single approach, but rather a bigger problem that requires further research. It appears that the network traffic generated by these devices changes over time and therefore a single model cannot stay accurate for a longer period of time. In this paper, we have not investigated the root cause of the accuracy decline. We have not found any obvious reason for this behavior and change in network traffic. We plan to defer the further investigation for the future work. We also plan to investigate whether certain class of devices is more susceptible to change of network traffic patterns than the others.

In this paper we have used out-of-the-box parameters for ML models and we did not do any fine-tuning of the parameters. Fine-tuning of the parameters is a rather time consuming activity that would also require significant computational resources. We believe, that fine-tuning of the parameters would lead to marginally higher accuracy, but it would not affect the trend of decreasing accuracy over time.

It can also be expected that because these devices also communicate among themselves, a model] created for a device in one network setting might not be accurate for the same device in another network setting. Additionally, users' usage pattern change between households and therefore it can be expected that a device might have a significantly different network traffic pattern, even when it is deployed in the same network setting.

In order to study this scenario, we are working on a home router device, that will allow users to deploy small test-beds in their homes. The user would connect their IoT devices to the router which would collect the traffic generated by these devices. The router would send anonymised packet headers (without payloads) to the server, where we would be able to analyze them.

One of the possible solutions that we would like to investigate in the future is to keep updating the model with new data at the edge, meaning retraining or tuning of models on edge devices [34], [35]. This could potentially not only solve the problem of changing traffic pattern over time, but also the problem of a device deployed in different network settings.

VI. Conclusion

In this paper we revisited four different approaches to IoT device identification based on the network traffic. For that purpose we have collected network traffic from a test-bed containing 41 different IoT devices over the period of 27 weeks. We have split these data into three periods, each containing 9 non-overlapping weeks. Then we have used three various approaches found in the literature. First, we evaluated a two-stage Random Forest classifier using features extracted from a 1 hour window of collected traffic [5]. Next, we used 2D Convolutional Neural Network on a stream of raw packets [7], [8]. Later, we used Random Forest and Decision Tree classifiers on features extracted from a 1 second window of network traffic [9]. Finally, we proposed and evaluated models using Random Forest classifier and Fully Connected Neural Network on features extracted from TCP/UDP flows.

Each model was trained using data from one period, while it was evaluated on data from the whole dataset (*i.e.,* all three periods). We have shown that while the accuracy of these models is high when tested on the dataset from the same period as the training dataset, the accuracy degrades over time when evaluated on dataset collected outside of the training period. The average degradation of the models' accuracy ranged between 12 and 21 percentage points, with an average of 17 percentage points.

Because this behavior is consistent across all models and various features extracted, we believe that the data generated by the devices change over time and cannot be captured by a single model. We propose a possible solution for this problem is by updating the model at the edge [34], [35].

ACKNOWLEDGMENT

We would like to thank Pinheiro *et al.* [9] for providing us with their source code so the evaluation of their approach could be fair and complete. We thank the anonymous reviewers and our shepherd, Maciej Korczynski. Authors were partially funded by EPSRC DADA: Defence Against Dark Artefacts (EP/R03351X/1) and EPSRC Databox: Privacy-Aware Infrastructure for Managing Personal Data (EP/N028260/1).

REFERENCES

[1] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2019.

[2] E. Lear, R. Droms, and D. Romascanu, "Manufacturer Usage Description Specification," RFC 8520, Mar. 2019.

[3] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2177–2184.

[4] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "Iot-keeper: Detecting malicious iot network activity using online traffic analysis at the edge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 45–59, 2020.

[5] 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, 2019.

[6] A. Pashamokhtari, H. H. Gharakheili, and V. Sivaraman, "Progressive monitoring of iot networks using sdn and cost-effective traffic signatures," in *2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT)*, 2020, pp. 1–6.

[7] F. Yin, L. Yang, Y. Wang, and J. Dai, "Iot etei: End-to-end iot device identification method," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, 2021, pp. 1–8.

[8] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[9] A. Pinheiro, J. Bezerra, C. Burgardt, and D. Campelo, "Identifying iot devices and events based on packet length from encrypted traffic," *Computer Communications*, vol. 144, pp. 8–17, 05 2019.

[10] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: Association for Computing Machinery, 2005.

[11] "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.

[12] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[13] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach," in *Proc. of the Internet Measurement Conference (IMC)*, 2019.

[14] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *Workshop on Data and Algorithmic Transparency (DAT'16)*, 2016. [Online]. Available: http://arxiv.org/abs/1705.06805

[15] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in iot networks: A survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804520300126

[16] P. Yadav, Q. Li, R. Mortier, and A. Brown, "Network service dependencies in commodity internet-of-things devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, ser. IoTDI '19. New York, NY, USA: ACM, 2019, pp. 202–212.

[17] Y. Amar, H. Haddadi, R. Mortier, A. Brown, J. A. Colley, and A. Crabtree, "An analysis of home iot network traffic and behaviour," *CoRR*, vol. abs/1803.05368, 2018. [Online]. Available: http://arxiv.org/abs/1803.05368

[18] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 506–509.

[19] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home device events," in *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020.

[20] S. J. Saidi, A. M. Mandalari, R. Kolcun, H. Haddadi, D. J. Dubois, D. Choffnes, G. Smaragdakis, and A. Feldmann, "A haystack full of needles: Scalable detection of iot devices in the wild," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 87–100.

[21] C. Yang, W. Jiang, and Z. Guo, "Time series data classification based on dual path cnn-rnn cascade network," *IEEE Access*, vol. 7, pp. 155 304–155 312, 2019.

[22] S. A. Magid, F. Petrini, and B. Dezfouli, "Image classification on iot edge devices: profiling and modeling," *Cluster Computing*, 2019.

[23] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 9031–9042.

[24] "Tensorflow lite," https://www.tensorflow.org/lite, [Online; accessed Jan 2020].

[25] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J. sun Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, "Benchmarking tinyml systems: Challenges and direction," in *Proceedings of the 3rd MLSys Conference*, ser. MLSys'20, 2020.

[26] A. Painsky and S. Rosset, "Lossless compression of random forests," *Journal of Computer Science and Technology*, vol. 34, pp. 494–506, 2019.

[27] A. Feraudo, P. Yadav, V. Safronov, D. A. Popescu, R. Mortier, S. Wang, P. Bellavista, and J. Crowcroft, "Colearn: Enabling federated learning in mud compliant iot edge networks," in *In 3rd International Workshop on Edge Systems, Analytics and Networking (EdgeSys'20)*. New York: ACM, April 2020.

[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*, ser. IoTDI '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 106–117.

[29] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing," *Proc. VLDB Endow.*, vol. 7, no. 13, pp. 1529–1540, Aug. 2014.

[30] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-thing devices," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18. USA: USENIX Association, 2018, pp. 327 – 341.

[31] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, "DÏot: A federated self-learning anomaly detection system

for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[32] R. Kolcun, D. A. Popescu, V. Safronov, P. Yadav, A. M. Mandalari, Y. Xie, R. Mortier, and H. Haddadi, "The case for retraining of ML models for iot device identification at the edge," *CoRR*, vol. abs/2011.08605, 2020. [Online]. Available: https://arxiv.org/abs/2011.08605

[33] G. Hackeling, *Mastering Machine Learning With Scikit-Learn*. Packt Publishing, 2014.

[34] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *SysML 2019*, 2019, to appear. [Online]. Available: https://arxiv.org/abs/1902.01046

[35] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluivers, R. van Dalen, C. W. Lau, L. Carlson, F. Granqvist, C. Vandevelde, S. Agarwal, J. Freudiger, A. Byde, A. Bhowmick, G. Kapoor, S. Beaumont, Áine Cahill, D. Hughes, O. Javidbakht, F. Dong, R. Rishi, and S. Hung, "Federated evaluation and tuning for on-device personalization: System design & applications," 2021.