Rapid IoT Device Identification at the Edge

Oliver Thompson
Imperial College London

Anna Maria Mandalari
Imperial College London

Hamed Haddadi
Imperial College London

ABSTRACT

Consumer Internet of Things (IoT) devices are increasingly common in everyday homes, from smart speakers to security cameras. Along with their benefits come potential privacy and security threats. To limit these threats we must implement solutions to filter IoT traffic at the edge. To this end the identification of the IoT device is the first natural step.

In this paper we demonstrate a novel method of rapid IoT device identification that uses neural networks trained on device DNS traffic that can be captured from a DNS server on the local network. The method identifies devices by fitting a model to the first seconds of DNS second-level-domain traffic following their first connection. Since security and privacy threat detection often operate at a device specific level, rapid identification allows these strategies to be implemented immediately. Through a total of 51,000 rigorous automated experiments, we classify 30 consumer IoT devices from 27 different manufacturers with 82% and 93% accuracy for product type and device manufacturers respectively.

CCS CONCEPTS

• **Security and privacy** → *Network security*; • **Networks** → *Network monitoring*; **Public Internet**; **Network measurement**.

KEYWORDS

Internet of Things, IoT identification, IoT security and privacy, Internet Measurement, machine learning, neural networks

1 INTRODUCTION

The consumer Internet of Things (IoT) space has experienced a significant rise in popularity in recent years. From smart speakers, to baby monitors, these devices are becoming increasingly common in households [1]. Since there are no strict compliance and regulations in this ecosystem, IoT malware, botnets, and device abuse (e.g., external access leading to domestic abuse) is increasingly becoming a recurring and major security and privacy issue [2–4]. On the other hand, due to the presence of several middleboxes, gateways and traffic sampling at ISPs; it is practically impossible to identify, detect, and isolate the misbehaving devices or households. This means devices at the edge are best positioned to defend against these attacks.

IoT devices would benefit from automated management of these privacy and security threats [5, 6]. The first natural step is to automate the identification of the device at the edge. There have been several solutions proposed for IoT device identification [7–12]. Those approaches rely on training machine learning models offline or in a cloud environment, using network traffic. However, the training and validation of these models is achieved using a list of features of a particular set of devices, over a long time period. Moreover, these approaches rely on continuous and complete packets capture and data collection, which is not feasible on a device at the edge which likely has limited computational resources.

In this paper we propose a novel method of rapid IoT device identification using neural networks trained on device DNS log traffic that can be captured from a DNS server on the local network. Our method is able to accurately identify the device type from the first few seconds of traffic after the device is connected. The method identifies devices by fitting a model to the first minute of DNS second-level-domains traffic following their first connection.

It is important that device identification occurs rapidly so that device-specific privacy and security threats can be mitigated immediately following a device's first connection. By only considering the traffic following the first connection we are able to fingerprint the product's traffic in a more repeatable way; as the user will not have much time to introduce variation into the traffic by using the product, and there may be certain consistent traffic behavior triggered by the first time connection.

Through a total of 51,000 rigorous automated controlled experiments, collected during different periods of time from 30 IoT devices, we characterize the minimum amount of time necessary to identify the device. Results demonstrate that the model reaches maximum accuracy when trained on only 30 seconds of data after the device is connected, and can therefore perform accurate classification very quickly. We also demonstrate that the model retains high accuracy when tested with data collected several days after the training period. At product level granularity the accuracy and macro f1 score is 85% and 0.87 respectively, when classifying the devices manufacturer rather than product type, the accuracy and f1 score were 95% and 0.91.

We sample the design space of neural networks and compare 1,800 model configurations to determine the best neural network architecture. we ascertain that a model with an input dimension of 32 and 2 hidden layers is highly accurate on the training data and can retain this accuracy when tested on unseen data collected several days after the training period. Unlike similar approaches, this method does not require full packet capture, which is computationally expensive and may pose privacy concerns.

Our main contributions are as follows:

- We develop a methodology for identifying IoT devices using the first 30 seconds of DNS traffic.
- We show that it is possible to detect device product with an accuracy of 82%, device manufacturers can be predicted with an accuracy of 93%.
- We demonstrate that accuracy is retained for at least one week following training data collection.

Category	Device Name
Audio	Echo Spot, Echo Plus, Google Home
Camera	Blink Cam, Bosiwo Cam, Wansview Cam, Yi Cam
Home Automation	Anova Sousvide, Cosori Cooker, Gosund Bulb, Govee Strip, Honeywell T-stat, Levoit Humidifier, Magichome Strip, Meross Door Opener, Netatmo Weather, Smarter Coffee Machine, Smartlife Remote, TP-Link Bulb, TP-Link Plug, Wemo Plug
Smart Hubs	Insteon, Lightify, Philips Hue, Sengled, Smartthings, SwitchBot, Xiaomi
Video	Fire TV, Samsung TV

Table 1: IoT devices under test.

2 METHODOLOGY

In this section we cover the data collection and experiment methodology. We describe the testbed we use for conducting the experiments, the IoT devices under test, and the neural network architecture we use to identify the devices.

2.1 Testbed and IoT Devices

Our methodology rely on a controlled environment for testing IoT devices. Our testbed consists of:

- A *router* that offers IP connectivity to the IoT devices under test, and the ability to capture network traffic for each device;
- A *DNS server* under our control, that serves as a proxy for the ISP's DNS server.
- *Smart-plugs* which can be turned on and off programmatically;
- A set of *support scripts* to control the smart plugs to turn on and off an IoT device automatically.

The support scripts are used to systematically switch the IoT devices off and on through the smart-plugs and collect the device's first traffic following reboot as a PCAP file into a structured directory.

Table 1 describes the IoT devices we use in our experiments, by category. We consider in total 30 IoT devices from 27 manufacturers, chosen for the popularity and prevalence in homes.

2.2 Dataset Generation

We perform 51,000 on-off experiments. Each experiment turns the devices on and off 100 times every two minutes, and this process is scheduled to run every 12 hours during one week. We use Python 3's Scapy library [13] to parse each PCAP file by identifying the timestamp of the DHCP discover packet and creating a list of all outbound DNS queries and their respective timestamps, following this DHCP discovery. We then save the Python objects containing the DNS data, alongside their source PCAP file.

In order to use the DNS traffic as input to a predictive model it is necessary to reduce the wide range of possible URLs to a set of discrete buckets. To achieve this, we pass the SLD (second-level domain) to a hash function, the result of the hash function is reduced into h buckets using the modulo operator, where h is the hash resolution and the hashing function is Python 3's built in hash function. See equation 1.

$$URL = 'time1.google.com'$$
$$SLD = 'google' \tag{1}$$
$$datapoint = hash(SLD) \% h$$

We test different values of hash resolutions between 4 and 64. A higher hash resolution reduces the chances of unrelated SLDs colliding, and therefore results in smaller information loss. However, a lower hash resolution reduces the complexity of the model and size of the dataset.

We choose the SLD as the input to the hash function because it treats related DNS queries whose only difference is in their sub-domain (for example, 'time1.google.com' and 'time2.google.com'), as the same query. If we were to consider the entire URL, each member of this URL 'family' would hash to a different value and the dataset would not represent that these queries are related. This is particularly important as the sub-domain is often the part of the URL that changes most frequently and by ignoring it the model will be more robust to changes in device behavior. On the other hand, considering only the top-level domain (TLD) would be too general, and completely unrelated domains would be hashed to the same value.

Each dataset is associated with a time delta value $t\Delta$ between 1s and 60s and is filtered to contain only the DNS queries whose timestamps fall between the DHCP timestamp and the sum of the DHCP timestamp and the time delta as shown in equation 2.

$$Dataset_{t\Delta} = \{Device, \{DNS_{hash}, DNS_t\}\}$$
$$s.t. : DHCP_t < DNS_t <= (DHCP_t + t\Delta) \tag{2}$$

Once the data is filtered by DNS timestamps, it is converted from storing (DNS_{hash}, DNS_t) pairs to storing the DNS and its associated frequency. The frequency is calculated as the average number of times per second the DNS_{hash} occurs between $DHCP_t$ and $(DHCP_t + t\Delta)$, see the final schema of the dataset in equation 3.

$$Dataset_{t\Delta} = \{Device, \{DNS_{hash}, DNS_{frequency}\}\} \tag{3}$$

In order to test how well the predictive models retain accuracy over time, we also generate and save datasets over a restricted set off dates, for example only containing experimental data captured between 2 days, rather than every experiment. This allows us to train models on data from particular dates and analyze its performance over other, unseen time periods.

2.3 Data Pre-processing

We split the dataset into features and labels, the features are normalized between 0 and 1 using a minmax scaler and the label categories are encoded using one hot encoding [14].

This process must occur both for training and when using the models to make predictions on unseen data. For the purposes of training the model, we split the data into training and testing data in a $80 : 20$ ratio and we split the training data into training and validation data with the same ratio again.

Parameter	Values tested
Number of Hidden Layers	1,2,3
Hash Resolution	4, 8, 16, 32, 64
Time Delta	1 to 60
Number of output Classes	27 and 30

Table 2: Parameters and hyperparameters for the neural networks under consideration.

Hyperparameter	Value
Learnig Rate	0.001
Beta 1	0.9
Beta 2	0.999
Epsilon	1e-7

Table 3: Hyperparameters for the Adam optimizer.

2.4 Neural Network Architecture

We choose neural networks for the predictive model as they can learn complex behavior including both the presence of particular DNS queries and patterns that appear in the time domain. It is also possible to update a neural network's weights when more data becomes available [15] and different configurations of network can be readily compared [16]. We generate and compare neural networks across a design space of 4 parameters and hyperparameters shown in Table 2, in order to find the optimal values.

The dimension of the input layer of the neural network matches the hash resolution h used to generate the dataset. This is because the input feature is the frequency with which each hashed domain is visited, and each hashed domain corresponds to one of the input neurons.

The remaining layers consisted of a varying number of dense hidden layers with 64 neurons followed by the output layer whose dimension is equal to the number of devices in the experiment (27 for manufacture granularity and 30 for device granularity). All layers use rectilinear activation except for the final layer which use softmax activation. The softmax activation $\sigma(z_i)$ is given by equation 4 where z is the input vector and K is the number of classes.

We choose categorical cross-entropy as the loss function as given by equation 5 where N is the number of observations, C is the number of categories and $p_{model}[y_i \in C_c]$ is the probability predicted by the model that observation i belongs to class c. We chose categorical accuracy as the target metric and an Adam optimizer whose hyperparameters are shown in Table 3. The architecture of an example neural network is shown in Figure 1.

$$\sigma(z_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e_j^x} \qquad (4)$$

$$-\frac{1}{N} \sum_{N}^{i=1} \sum_{C}^{c=1} 1_{y_i \in C_c} log\, p_{model}[y_i \in C_c] \qquad (5)$$
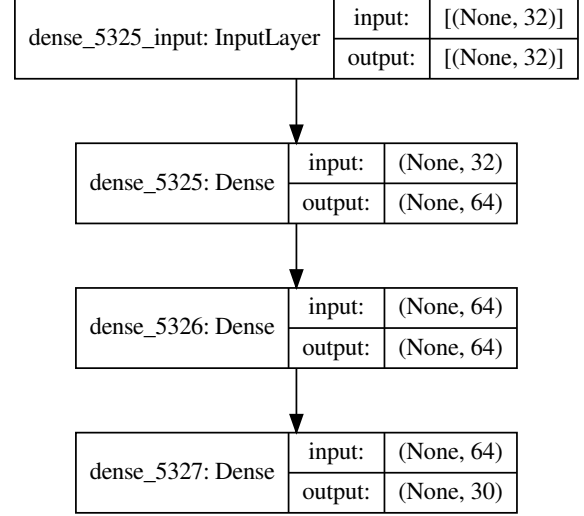

Figure 1: Example architecture of a neural network with 2 hidden layers and a hash resolution of 32.
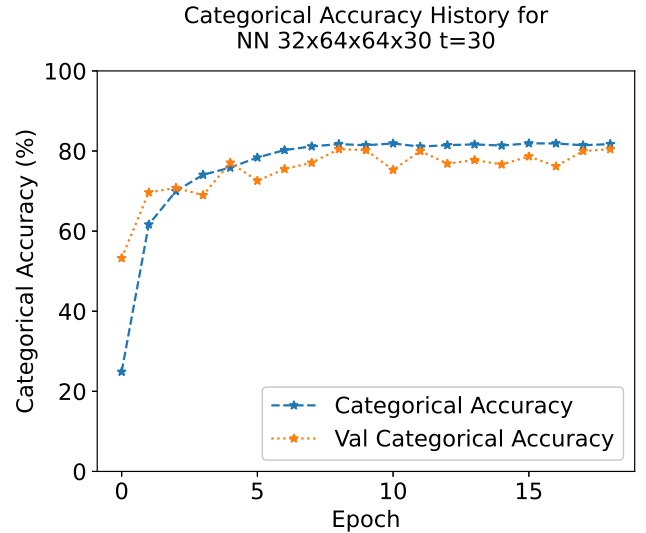

Figure 2: Categorical accuracy of a neural network with 2 hidden layers, a hash resolution of 32 and a time delta of 30 over number of epochs. Early stopping stops training at epoch 17.

2.5 Model Training

We train neural networks with 1, 2 and 3 hidden layers for each hash resolution and time delta. We run the training over 100 epochs, however we implement early stopping to maximise the categorical accuracy and training would usually conclude at a much lower number of epochs. An example of the categorical accuracy training history for the training and validation data of a particular neural network is shown in Figure 2.

When comparing large numbers of neural networks it is important to reduce the role that the random initialization of weights
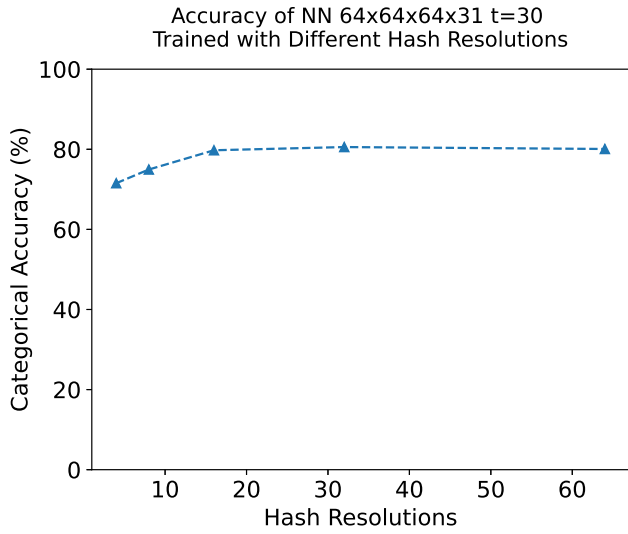
Figure 3: The categorical accuracy of she neural network with 2 hidden layers and a time delta of 30s. We train and test with 5 different hash resolutions to explore the effect of hash resolution value on model accuracy.

plays in the networks results. To accommodate this, we train each of the 1,800 neural network configurations four times with initialization weights initialized from different random seeds for reproducibility. We then average the accuracy of these four networks and use it for comparison between other network configurations. We then use the model with the highest accuracy to generate a macro f1 score which is the harmonic mean between precision and recall.

If p_j and r_j are the precision and recall for a given class j within a set of classes Q, macro f1 score is given by equation 6.

$$MacroF1 = \frac{1}{Q} \sum_{j=1}^{Q} \frac{2 \cdot p_j \cdot r_j}{p_j + r_j} \tag{6}$$

Following training and evaluation, we collect each neural network object in a directory labelled with the time delta, hash resolution and number of hidden layers. The object acts as a wrapper for the 4 averaged models and contains the dataset and results of the best performing model, including the loss, categorical accuracy and macro f1 score. We save the best predictive model itself separately.

3 EVALUATION

In this section we evaluate the highest performing networks, and we compute the minimum amount of time necessary to reach maximum accuracy from the first few seconds of traffic after the device is connected.

3.1 Highest Performing Networks

By sorting the 1,800 models of different configurations by categorical accuracy, we can critically evaluate the neural network design space. The highest performing networks all have a hash resolution
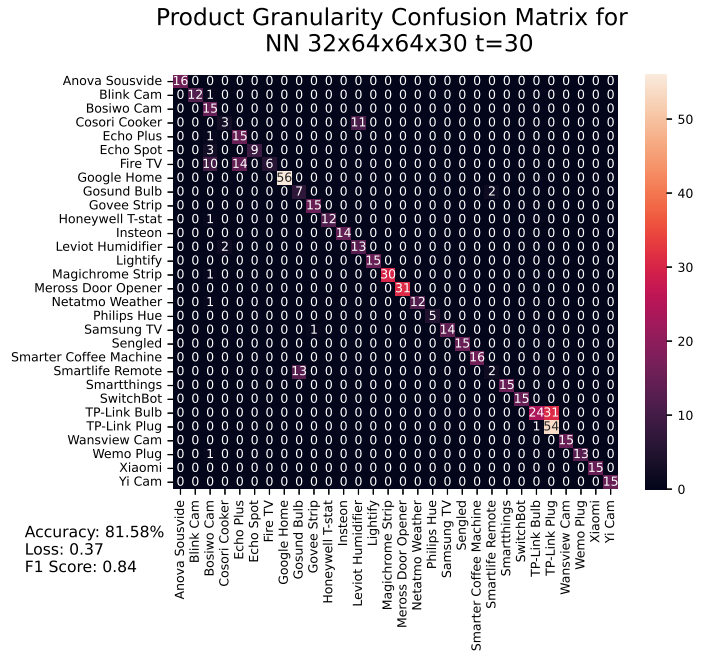

Figure 4: Product level granularity confusion matrix of a highly performing neural network with 2 hidden layers and a time delta of 30s and hash resolution of 32.
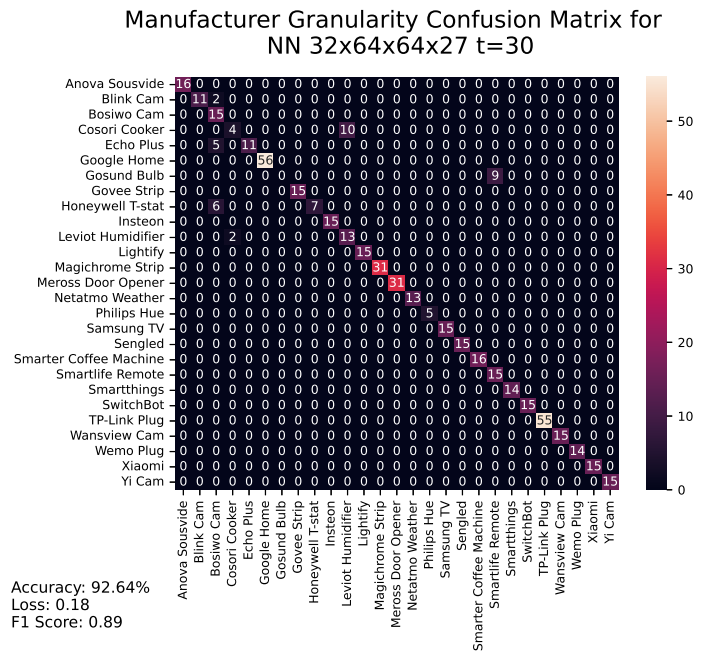

Figure 5: Manufacturer level granularity confusion matrix of a highly performing neural network with 2 hidden layers and a time delta of 30s and hash resolution of 32.

of at least 32. Figure 3 shows that below 32 there is a reduction in

accuracy and hash resolutions above 32 do not result in a significant increase in accuracy.

Results show that there is no notable increase in accuracy between neural networks with more than 2 hidden layers. This suggests that the function that classifies a device from its DNS traffic is not complex enough to require more than 2 hidden layers and that using more complex models may therefore increase the chance of over-fitting.

Figure 4 shows the confusion matrix of a neural network trained on 30 seconds of data at product level granularity. Its categorical accuracy and macro f1 score are 82%, and 0.84. Figure 5 shows the confusion matrix for the same experiment conducted at manufacturer level granularity. Its categorical accuracy and macro f1 score are 93% and 0.89. Both confusion matrices are constructed with the predicted device along the y-axis and the actual device along the x-axis. Classifications that do not lie on the main diagonal are misclassifications.

It can be seen that false positive identification occurs between the 2 TP-Link devices and between the 3 Amazon devices. Since these devices share a common manufacturer, they tend to query the same destinations which makes product level identification more difficult. The Smartlife Remote, the Cosori Cooker and the Gosund bulb devices are the most frequently misclassified. By examining the dataset we conclude that this is because these 3 devices rarely make any DNS requests when compared to the other devices, so there is not enough data to make an accurate prediction. It may be necessary to measure DNS traffic over a longer time period before a reliable classification can be made for these devices.

3.2 Comparison of Different Time Deltas

Once we establish the optimum neural network architecture and hash resolution, we consider the categorical accuracy of a network against the time deltas over which it is trained. The value of the time delta determines how many seconds of traffic the neural network is trained on.

Figure 6 shows that the categorical accuracy of the neural network increases quickly between time deltas of 1 and 10 seconds. After this, the categorical accuracy increases at a slower rate until roughly 30 seconds, after which it does not increase significantly. When making a prediction on new data, we now know that 30 seconds of DNS traffic should be captured in order to maximize the chances of accurate classification.

3.3 Model Reliability Over Time

To validate the model further it is important to test the behavior on unseen data. It is understood from the literature [12] that device identification models trained on data acquired through packet capture quickly become inaccurate as device behavior changes, for example if a device receives a software update from its manufacturer. We restrict the training data of the neural networks to data collected over 2 days and use the unseen data from the following days to test the networks performance. We test the method over different date ranges.

Figure 7 shows that even when a model is only trained on a single day's data the neural network maintains its high accuracy over the following week. This suggests that the DNS traffic of IoT
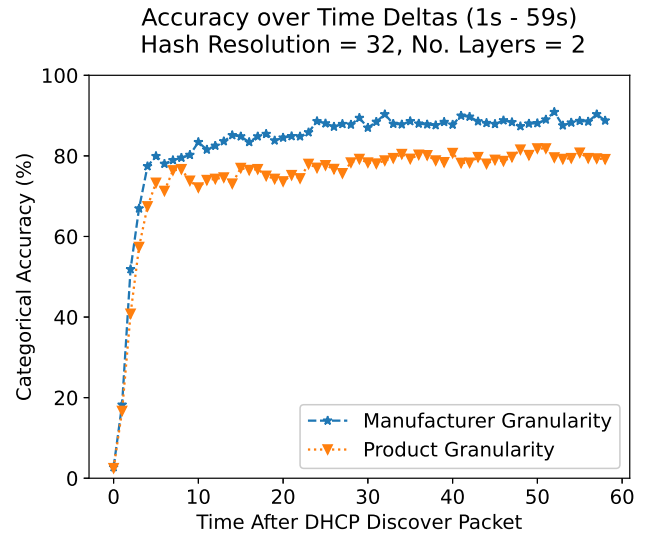

Figure 6: Accuracy against different time deltas. The neural network has a hash resolution of 32 and 2 hidden layers. The model's accuracy stops increasing significantly at around 30s, at both product and manufacturer granularity.
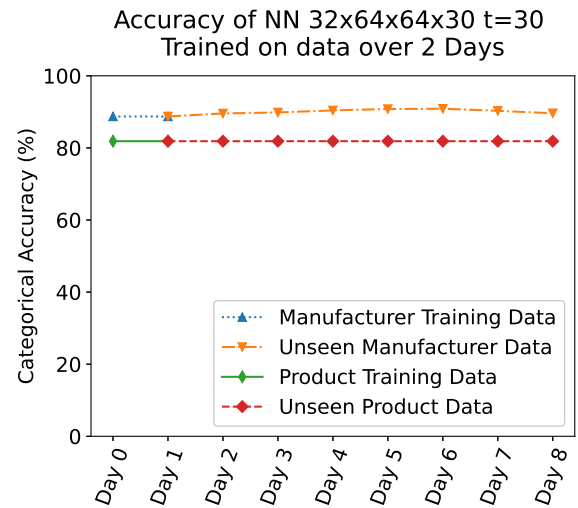

Figure 7: Plot of model accuracy over a week when testing on unseen data. A network trained on data for 2 days maintains its accuracy when tested on data from the following 7 days.

devices does not change as frequently as the contents of the network packets.

4 DISCUSSION

4.1 Limitations

Our methodology has some limitations.

Model degradation. Although we show that the model retains reliability over a week long time period, we can assume that there

will be a point in time where the model's accuracy will degrade. It would be beneficial to test the accuracy degradation over a larger timescale in order to understand how often the model weights must be updated to retain accuracy. Past works have demonstrated that IoT DNS traffic is consistent over a period of at least 6 months [6], which suggests the model may only require updating very rarely.

Scalability. We demonstrate effective identification over a dataset of 30 devices with a relatively simple, 4 layer neural network. We do not yet understand if this model architecture will scale to a larger set of devices. It would also be useful to understand if the model could be used to accurately predict the presence of a non-IoT device.

Unexpected behavior. In very rare cases an IoT device may use hard-coded IP addresses rather than making DNS requests. The method of identification cannot be used for these types of devices.

4.2 Future Directions

One of the possible solutions that we would like to investigate in the future is to use this method with on-device training to update the model locally to fit the latest DNS behavior. By measuring the DNS traffic of connected devices of known product and manufacturer, this data could be used to adjust the weights in the model and perpetually maintain a high accuracy.

This methodology may be suitable for a crowdsourcing approach, where changes in model weights are shared between devices to benefit from information from a much larger dataset. This process would be privacy preserving by nature because all information about device traffic would be passed through the hashing function after which the original data could not be retrieved.

In order to support more fine-grained intra-manufacturer distinguishability, we aim to consider more seconds of initial traffic to improve the accuracy in these cases.

5 RELATED WORK

In recent years, a vast number of techniques for IoT device identification using machine learning have been investigated.

The approach laid out by Meidan *et al.* [17] is able to predict a device's brand and model with 99% accuracy. Their approach firstly distinguishes IoT devices from non IoT devices by examining the user agent property of the HTTP header from captured PCAP files. The second session identifies the type of IoT device through logical characteristics of the captured packets. Similarly, Aksoy *et al.*'s method [18] achieves device classification with an accuracy of over 95% through analysis of a single captured packet. They also observed that when devices share a manufacturer the accuracy of the classification is reduced.

A different approach from Kotak and Elovici [19] achieves IoT device identification of 99% also using deep neural networks. The network makes a prediction based on patterns found in a device's traffic; the raw binary PCAP file is truncated and converted into 28 x 28 greyscale images which are used as input to a simple convolutional neural network. This approach is efficient in that it utilizes the PCAP file in its raw form and does not need to explicitly extract features.

The methodology used by Kolcun *et al.* [12] explores various predictive models including a decision tree classifier, random forest classifiers and 3 different neural networks. The data features consist of various statistical properties from PCAP files, for example kurtosis and skewness of packet size and inter-packet gap, source and destination ports and domains. The results show the method with the best accuracy was the random forest classifier, however the methods do not retain their accuracy over longer periods of time. All the aforementioned techniques use packet capture to extract features from an IoT device's traffic, while our aim is to use only DNS log traffic.

The approach from Perdisci *et al.* [20] however, resembles our method of feature extraction as it uses the query URLs of a device's DNS requests. This approach uses the whole URL rather than the SLD and uses a naive document retrieval algorithm to match URLS to devices. By building on top of this methodology with advanced machine learning techniques, we are able to learn behavior in the time domain in addition to matching DNS queries and achieve high accuracy over a larger set of devices, without requiring full packet capture. Moreover, Perdisci *et al.* approach set the time window length to be one hour, whereas our methodology uses only the first 30 seconds of DNS traffic.

6 CONCLUSION

Consumer IoT devices are already very popular, and their usage is expected to grow further. There is a need to track their deployment without deep packet inspection or active measurements, both intrusive and unscalable methods for large deployments on a device at the edge. Our insight is that many IoT devices contact a small number of domains and it is possible to detect such devices at scale from sampled DNS measurements following a devices first boot.

Our method is able to detect 30 IoT products at manufacturer granularity with 93% accuracy and at product granularity with 82% accuracy. While this detection may be useful to perform device-specific DNS filtering of IoT devices at home, it raises concerns about the general detectability of such devices and the corresponding human activity.

We have also established that roughly 30 seconds of IoT device DNS traffic should be observed in order to maximize the accuracy of the prediction and that models can retain this accuracy over the course of a week. We show that product level identification is highly accurate but may mis-classify devices that share a manufacturer. Conducting the experiment at the manufacturer level eliminates this confusion and can classify nearly every manufacturer accurately.

REFERENCES

[1] IoT Analytics. IoT 2020 in review: The 10 most relevant IoT developments of the year. https://iot-analytics.com/iot-2020-in-review/. [Online; accessed Sept. 2021].

[2] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information exposure for consumer IoT devices: A multidimensional, network-informed measurement approach. In *Proc. of the Internet Measurement Conference (IMC)*, 2019.

[3] Hooman Mohajeri Moghaddam, Gunes Acar, Ben Burgess, Arunesh Mathur, Danny Yuxing Huang, Nick Feamster, Edward W. Felten, Prateek Mittal, and Arvind Narayanan. Watching you watch: The tracking ecosystem of over-the-top TV streaming devices. In *CCS'19*, 2019.

[4] Janus Varmarken, Hieu Le, Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. The TV is smart and full of trackers: Measuring smart TV advertising and tracking. *20st Privacy Enhancing Technologies Symposium (PETS 2020)*, 2020.

[5] Ibbad Hafeez, Markku Antikainen, Aaron Yi Ding, and Sasu Tarkoma. IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge. *IEEE Transactions on Network and Service Management*, 17(1):45–59, 2020.

[6] Anna Maria Mandalari, Daniel J Dubois, Roman Kolcun, Muhammad Talha Paracha, Hamed Haddadi, and David Choffnes. Blocking without breaking: Identification and mitigation of non-essential IoT traffic. *21st Privacy Enhancing Technologies Symposium (PETS 2021)*, 2021.

[7] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan. AUDI: Towards autonomous IoT device-type identification using periodic communication. *IEEE J. on Sel. Areas in Comm.*, 37(6), 2019.

[8] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019.

[9] Arman Pashamokhtari, Hassan Habibi Gharakheili, and Vijay Sivaraman. Progressive monitoring of IoT networks using SDN and cost-effective traffic signatures. In *2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT)*, pages 1–6, 2020.

[10] Feihong Yin, Li Yang, Yuchen Wang, and Jiahao Dai. IoT ETEI: End-to-end IoT device identification method. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2021.

[11] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access*, 5:18042–18050, 2017.

[12] Roman Kolcun, Diana Andreea Popescu, Vadim Safronov, Poonam Yadav, Anna Maria Mandalari, Richard Mortier, and Hamed Haddadi. Revisiting IoT device identification. *Network Traffic Measurement and Analysis Conference (TMA)*, 2021.

[13] Scapy. Packet crafting for python2 and python3. https://scapy.net/. [Online; accessed Sept. 2021].

[14] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International journal of computer applications*, 175(4):7–9, 2017.

[15] CL Philip Chen and John Z Wan. A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(1):62–72, 1999.

[16] David Hunter, Hao Yu, Michael S Pukish III, Janusz Kolbusz, and Bogdan M Wilamowski. Selection of proper neural network sizes and architectures—a comparative study. *IEEE Transactions on Industrial Informatics*, 8(2):228–240, 2012.

[17] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*, pages 506–509, 2017.

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

[19] Jaidip Kotak and Yuval Elovici. IoT device identification using deep learning. In *Álvaro Herrero, Carlos Cambra, Daniel Urda, Javier Sedano, Héctor Quintián, and Emilio Corchado, editors, 13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pages 76–86, Cham, 2021. Springer International Publishing.

[20] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. IoTFinder: Efficient large-scale identification of IoT devices via passive dns traffic analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 474–489, 2020.