A Precise Flow Representation for Autonomous IoT-Devices Reconnaissance

Govinda M. G. Bezerra*‡, Tadeu N. Ferreira‡, Diogo M. F. Mattos*
*LabGen/MídiaCom, ‡LAProp – PPGEET/TET/UFF
Universidade Federal Fluminense (UFF) – Niterói, Brazil
{govindamgb, tadeu_ferreira, diogo_mattos}@id.uff.br

***Abstract*—Devices from the Internet of Things increasingly mediate a significant number of essential everyday activities. IoT devices empower homes, industries, and offices, monitoring, sensing, and acting ubiquitously and stealthily. However, each device produces a network fingerprint that leaks information about users' behaviors and routines. This paper proposes a flow representation method for precise recognition of different types of IoT Devices. Our proposal relies on a tensor representation of the network flows to retrieve spatial and temporal correlation of flows. We show that our proposal achieves up to 99% precision on classifying IoT network flows using machine learning algorithms, such as Convolution Neural Networks, Recurrent Neural Networks and boosted decision trees.**

***Index Terms*—IoT Device, CNN, LSTM, Classification, Machine Learning**

I. Introduction

The 5th generation (5G) mobile networks aim to provide superior performance to the previous generations in terms of mobile broadband, massive Machine-Type Communications (mMTC), and latency. The COVID-19 pandemic boosted the demand for reliable broadband connectivity to support home office, homeschooling, and social/everyday interaction due to the constraints imposed by lockdown and social isolation measures adopted by many countries in 2020. In this way, the increased demand for connectivity stimulates the deployment of the 5G network infrastructure at a faster pace. While only 10% of Communications Service Providers (CSPs) commercialized 5G services in 2020, the current trend is that the share grows to 60% by 2024 [1].

The 5G network's major use case is the massive Machine-Type Communications, designed to support the connection of many Internet-of-Things (IoT) devices, such as sensors, smart wearables, domestic appliances, and industrial machinery. As IoT devices are increasingly present in everyday activities and, driven by this widespread of devices, the number of attacks has also grown intensely, with an expansion of more than 100% in the first semester of 2021 [2]. Hereupon, the autonomous connection that 5G networks offer to these devices raises new security and privacy concerns that must be addressed, considering the IoT devices' restrictions.

IoT devices present power, processing, networking, and storage specifications that significantly differ from traditional Internet nodes, such as servers, computers, and notebooks [3]. Meanwhile, due to the heterogeneity of purpose and manufacture designs, IoT devices are also heterogeneous, hindering a universal approach to securing the devices' connection. Furthermore, due to the demand for low-cost devices, manufacturers' design neglects security and privacy concerns. As a consequence, IoT devices manifest restricted capability to resist cyberattacks.

This paper proposes a method to disclose IoT devices as monitors network traffic and identifies the IoT devices' type. We extract flow-level statistical information from real IoT device network data. The proposed data representation structure is a tensor that captures the spatial and temporal correlation between network flows. We deploy device classification models based on Machine Learning, using three different approaches: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Extreme Gradient Boosting (XGBoost). We investigated each machine learning model using different configurations in order to select the best parameters and architecture.

The ubiquitous introduction of IoT devices raises privacy concerns since these devices are embedded with communication capabilities and interact with users or remote servers. The devices' network activity indicates users' behaviors and routines, even without access to network traffic payload [4], [5]. Moreover, users are unaware that IoT devices collect information even when they are not actively used and may send data to first and third parties [6]. Unlike previous work, our proposal focuses on data representation to enable precise device recognition. Our results show that machine learning models achieve accuracy and precision higher than 97% for the evaluated neural network models and 99% for the best tree-based configuration.

The remainder of the paper is organized as follows. In Section II, we summarize related works. Section III presents the IoT devices dataset and the characterization of device traffic. In Section IV, we describe our proposal for data representation. Section V presents the machine learning approaches used in this paper and Section VI describes the evaluation and results. Section VII concludes the work and presents future directions.

II. Related Work

Previous works focus on classifying IoT devices and discuss approaches to enhance accuracy and precision in recognizing

The authors would like to thank CNPq, CAPES, FAPERJ, FAPESP (2018/23062-5), and RNP for the funding that made the research possible.

IoT devices or users' behaviors. Li *et al.* analyze university-campus network traffic to extract flow features that rely on the names of the contacted hosts, flow statistics and geographic information implicit in IP addressing [7]. Considering these features, they predict users' gender and education level with an accuracy of 82 and 78%, using an XGBoost model.

Sivanathan *et al.* aim to identify IoT devices through a two-stage classification method [8]. The first classification stage deploys the naive Bayes algorithm to classify devices based on contacted domains, communication ports, and cryptographic cipher suites shared during TLS handshake. The second stage classification combines first-stage results with more features, such as duration, volume, flow rate, DNS request intervals, and NTP request intervals. The proposed method achieves 99.88% accuracy using one-hour long time windows. A downside of the approach is that the time window is long for most IoT device classification applications, such as intrusion detection or network management. Shahid *et al.* aim to classify device flows according to information such as packet size and packet interarrival time [9]. The proposed approach classifies the devices with 99.9% accuracy, but the proposal analysis only considers four devices with clearly distinct behavior: a camera, a lamp, a sensor, and a plug. Besides, Meidan *et al.* identify IoT devices with 99.28% accuracy considering features extracted from TCP sessions [10]. However, the approach fails as some IoT devices do not use the TCP protocol.

Bai *et al.* propose an automatic method for identification of IoT devices categories, which consists of clustering packets in time windows and extracting features, such as the number of received and transmitted packets and packet counts for different protocols [11]. The work combines Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) to achieve an average accuracy of 74.8%. Similarly, Lopez-Martin *et al.* propose different combinations of CNN and LSTM networks to classify services/applications, e.g., HTTP and SIP [12]. The proposed method input is a 20×6 matrix representing 20 packets with six features: source and destination ports, number of bytes, TCP window size, interarrival time, and flow direction. The proposed method achieves 96% accuracy and 95% F1-score. In turn, Abbas *et al.* adopt a more straightforward configuration and perform IoT device classification using the K-Nearest Neighbors approach and obtain a 95% accuracy and a 91% F1-score [4].

Unlike previous approaches, our proposal achieves high accuracy and precision in the IoT devices classification, with a short time window, within 60 seconds, suitable for online applications. Furthermore, our model uses only statistical flow features, which allow device classification regardless of the transport and application protocols or the use of encryption, such as TLS and HTTPS. Besides, our method is flexible enough to correctly distinguish similar devices, as our analysis considers a dataset of more than 30 devices, with six device categories and different device models for each category.

III. Preprocessing the IoT Dataset

This paper uses a real network traffic of IoT devices to train and evaluate machine learning algorithms. The dataset used to validate the proposal is a subset from the one proposed by Ren et al. [6], which contains the traffic of 33 IoT devices captured using `tcpdump`[1] and stored as raw `pcap` files. The traffic of each device is composed of flows between network peers, using various transport and application-layer protocols with several flow patterns.

The traffic is generated through multiple experiments, divided into three categories: power, interaction, and idle. The power experiment traffic contains network packets collected for two minutes after launching the device. The interaction experiments generate traffic through the interaction with the device, such as physical interaction, voice commands, or companion mobile applications. The idle traffic is collected when no user is interacting with the device. The idle experiments last 8 hours and are performed three times, whereas the power experiments last for two minutes with, at least, three repetitions for each device. The interaction experiments vary in duration according to the type of interaction and vary the repeatability according to the automation viability of the test. Automated interactions, such as the use of companion application and voice command, were performed at least thirty times, while non-automated interactions, such as physical interactions, were performed at least three times.

The dataset includes two device sets of experiments, one performed in USA and another performed in UK. The devices used in each set are different, and, in the present paper, we focus on the UK subset. Table I summarizes the monitored devices and their categories.

TABLE I
Categories and models of monitored IoT devices.

Cameras	Smart Hubs	Home Automation
Blink Cam	Blink Hub	WeMo Plug
Bosiwo Cam	Insteon	Honeywell T-stat
Ring Doorbell	Lightify	Magichome Strip
Spy Camera	Philips Hub	Nest T-stat
Wansview Cam	Sengled	TP-Link Bulb
Xiaomi Cam	Smartthings	TP-Link Plug
Yi Cam	Xiaomi hub	
TV	**Audio**	**Appliances**
	Allure speaker	
Apple TV	Echo Dot	Anova Sousvide
Fire TV	Echo Spot	Netatmo Weather
Roku TV	Echo Plus	Smarter Brewer
Samsung TV	Google Home Mini	Xiaomi Cleaner
	Google Home	

A. Data Preprocessing

The raw data consists of several packets scattered in several files containing rough and irregular data. Thus, a preprocessing step is mandatory to analyze and use them correctly. The dataset `pcap` files are parsed using the PyShark library[2] and,

[1] Available at https://www.tcpdump.org/.

[2] Avaliable at https://github.com/KimiNewt/pyshark/.

through this, the target fields are extracted and saved as a DataFrame structure from the Pandas library[3].

The considered packets in this analysis use IP protocol in the network layer and UDP or TCP as the transport layer protocol. Datalink layer packets and control packets, such as ICMP, were discarded. The fields extracted from the packets are source and destination ports, source and destination IP address, source and destination MAC address, transport layer protocol, packet size, TCP flags, TCP segment size, timestamp, application layer protocol. For UDP packets, the fields referring to the TCP protocol were set to zero. The resulting dataset contains more than six million packets.

The preprocessing step does not consider packet payload because we focus on characterizing the devices according to their flow pattern rather than the data content. Besides, the data may be private due to protection with cryptographic protocols, such as HTTPS and TLS.

We assort packets into flows according to their source and destination IP addresses, source and destination ports, and transport-layer protocol. Each packet receives a flow identification number and a direction label that can assume either the value "forward", if the device sends the packet, or "backward", otherwise.

B. Traffic Characterization

Given the diversity of applications and manufacturers, the device's network features vary in application protocols, communication ports, contacted servers, and flow statistical metrics. To illustrate these differences, Figures 1(a) and 1(b) show the Sankey diagram of the network activity of two IoT device hubs: the Blink Security Hub and the Xiaomi Hub. Figure 1(a) shows that most Blink Security Hub packets use the transport protocol TCP and HTTPS as the application protocol, using UDP to resolve hostnames and to synchronize clocks. The communication endpoints consist of DNS and NTP server and a manufacturer host. Figure 1(b) shows the communication of the Xiaomi Hub, which, on the other hand, is very different from the Blink Security Hub. Xiaomi Hub uses multicast DNS to resolve local devices hostnames and uses transport ports in the range of non-privileged user ports, ranging from 1024 to 49151.

In our approach, we aggregate the traffic as flows. Thus, we first investigate the duration of the flows to choose an appropriate monitoring period. We consider as the flow duration the interval between the first and the last received or sent packet. Figure 2 reveals that the duration of flows ranges from milliseconds to a few minutes, but more than 80% of flows last only up to 10 seconds, and 60 seconds encompass 97% of all flows. We consider 60 seconds as the time window for our traffic analysis for the rest of this paper.

IV. The Proposed Data Representation

We exploit the privacy vulnerabilities of the presented dataset using machine learning algorithms to identify and classify the IoT devices using their traffic flow information. Our methodology uses statistical information from the devices' network flows. Hence, our methodology also applies to encrypted network traffic, since it does not rely on the packet payload data. Another advantage of the statistical approach is the independence of the port numbers and host IP addresses, hardening the model against protocol changes or application server updates.

IoT devices behave differently, depicted in Figure 1, and the flow durations last from a few milliseconds to several minutes, Figure 2. To extract the flow statistics, we divide the analysis into time windows, which tend to capture the main flow features and are still usable in online classification environments [13]. Figure 2 shows that using a 60 s time window encompasses the total duration of more than 97% of the flows present in the dataset and allows long-lasting flows to be well represented, without excessive fragmentation.

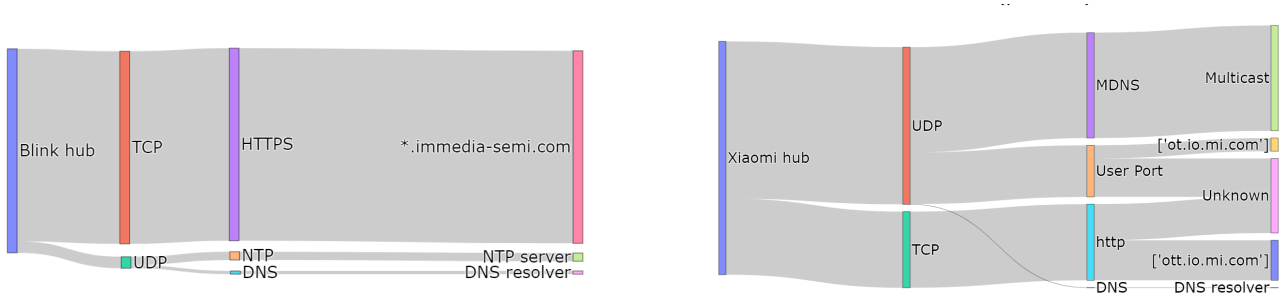
To analyze the data, we extract the network packets from the `pcap` file, and divide them into time windows based on their timestamp and the flow identifiers assigned in the preprocessing step. Then, we calculate the flow statistics for each time window and use these statistics as features to the machine learning algorithms. Our approach uses four types of features: data volume, temporal behavior, speed characteristics, and TCP protocol attributes. Each flow contains these features calculated for received and sent packets and the sum of the two subsets. Thus, the overall flow representation contains two-way traffic features.

The first set of features is volume-based, consisting of the packets' number and size in bytes. As the packets vary in size, we calculate the minimum, maximum, mean, variance, and standard deviation values to represent the metric, in addition to the total number of bytes. Furthermore, temporal features, comprising the flow duration and the inter-arrival times between packets, are represented with statistical values as the volume-based features. We also deploy ratio features, such as the number of packets per second, the number of bytes per second, and the ratio between reception and transmission rates. Lastly, features related to the TCP protocol correspond to the number of packets whose each flag in the TCP header is active and the statistical values of the TCP segment length.

The feature extraction outcome is a tensor dataset representing 76 features for 294 thousand flows of 34 devices[4]. We observed that some features have outliers that could negatively affect the performance of machine learning models. Thus, we upper limit the maximum value of each feature as the value of the 90th percentile and bottom limit it to the 10th percentile. This procedure is mandatory before the normalization, since, during the normalization procedure, outliers may lead median values to an insignificant range of values, which hampers the classification algorithms.

[3]Avaliable at https://pandas.pydata.org/.

[4]The device Insteon Hub does not have enough traffic data to be analyzed, and it was excluded from the analysis.



(a) Network traffic diagram for a Blink Security Hub.

(b) Network traffic diagram for a Xiaomi Hub.

Fig. 1. Comparison of different devices' network activities. Although both devices are smart thing hubs, (a) connects to multiple NTP servers and use unicast DNS requests, while (b) deploys multicast communication to propagate DNS requests locally.

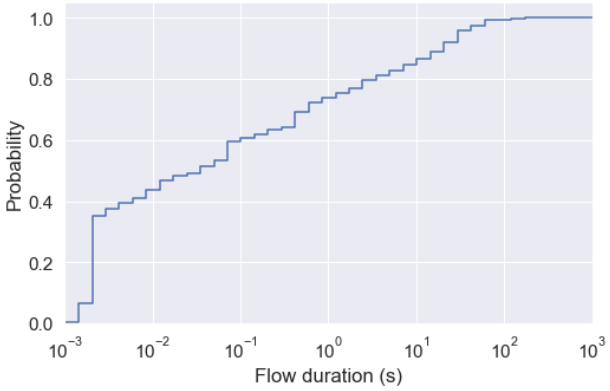


Fig. 2. Cumulative Distribution Function (CDF) for the flow duration. A 60 s time window encompasses up to 97% of all flows in the dataset.

V. Machine Learning Classification

We choose three well-known machine learning models from the literature to perform device identification and to verify the suitability, strengths and weaknesses of each implementation. The chosen models are: Extreme Gradient Boosting (XGBoost) decision trees, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). All models use the same data with the same flow features, but data organization differs for each model to match the model's input specifications. Next, we present a brief description of each evaluated algorithm, its implementations, and how we represent the data in each scenario.

A. Extreme Gradient Boosting Algorithm

Extreme Gradient Boosting Algorithm (XGBoost) is a machine learning system that uses decision trees to perform classification and regression tasks [14]. A decision tree is a structure of nodes, branches, and leaves. The nodes represent a test on some attribute, the branches represent the test results, and the leaves contain the output value used to perform prediction. The boosting technique consists of using several weak learners combined to create a more robust learning system. The algorithm creates and adds a tree at a time to the model to improve the prediction that the existing trees perform. The final classification prediction consists of the sum of values of the leaves corresponding to the class in the trees. An important XGBoost parameter is the maximum depth value, which higher values increase the ability to fit the data, whereas increases the complexity and the risk of overfitting.

The data structure used to train the model is a vector of feature values for each flow in a time window. The data structure implementation is a `DataFrame` from Pandas library. Labels containing class names are mapped to an integer for compatibility with the XGBoost implementation in the XGBoost[5] Python Package. In our approach, each algorithm input sample is a vector that corresponds to one flow containing its 76 features.

B. Convolutional Neural Network

Convolutional Neural Network (CNN) is a powerful machine learning mechanism that has been widely used in image classification, achieving state-of-the-art results. The input samples of a CNN may be interpreted as images that are three-dimensional arrays, where two dimensions represent the width (w) and height (h) of the image and the third dimension represents the color channels (c). In this way, an image in an RGB color space contains three color channels (red, blue, and green). A grayscale image, however, has only one channel.

The CNN model relies on the concept of the mathematical operation known as two-dimensional convolution, where an $n \times n$ filter sweeps across the image to extract image features, such as borders and color. In a convolution layer, the filter is applied to an area of the image, called receptive fields, then the dot product between the filter and the pixels is calculated and entered into a matrix. The output matrix of the convolution layer is known as a feature map or activation map. In addition to convolution layers, CNN deploys pooling layers and fully connected layers. The goal of pooling layers is to reduce the dimensions of the feature map and, for that, an $m \times m$ filter traverses the feature map, aggregating the values and generating an output matrix with a smaller dimension than

[5]Available at https://xgboost.readthedocs.io.

the input matrix. The fully connected layers are responsible for performing image classification. In these layers, all nodes connect to every node of the previous layer.

Our data representation creates an $w \times h$ image in which each flow is a row and features are columns. We set the image width w to 76, representing the flow features. To choose the height h, we use the number of active flows in the same time window. Figure 3 represents the CDF of active flows, in which it is possible to observe that 90% of the time windows contain at most 15 active flows. Thus, we set the image height h to 15. If a time window contains less than 15 flows, it does not have enough flows to fill image rows, so the remaining rows are padded with zeros. On the other hand, time windows with more than 15 flows are adjusted by randomly discarding flows.

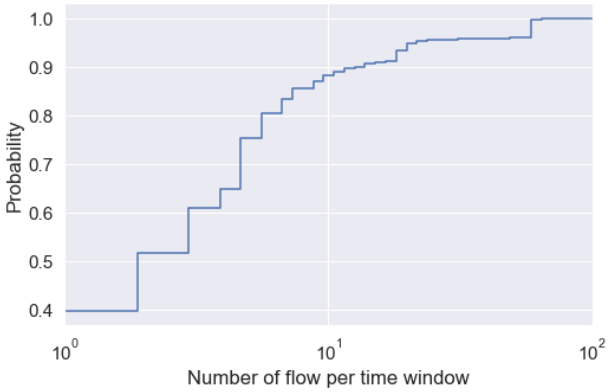


Fig. 3. Cumulative Distribution Function (CDF) of the number of flow per window. A 60 s windows reaches 0.9 probability of containing up to 15 flows.

The CNN model is built using the Keras[6] API in Python. The input format of this network has four dimensions: batch size, height, width, and depth. As our data do not have the dimensional depth, we reshape our time window image to the format (15, 76, 1) and concatenate all matrix to get a tensor shaped as (44.645, 15, 76, 1).

C. Long Short-Term Memory

Long Short-Term Memory (LSTM) network is a specialized Recurring Neural Networks (RNN). RNN differs from traditional neural networks in storing and using states or data from previous inputs to calculate subsequent outputs. This feature allows RNNs to learn temporal relationships among data, making the model popular for applications with time-varying data, such as voice recognition, language translation, and time series prediction. RNNs use the back-propagation through time algorithm to calculate the gradients and adjust the weight of the network nodes. Thus, each time step is considered as a layer. The major drawback of the method is that the gradient exponentially vanishes as it propagates through the layers, making the first layers always suffer a minor adjustment and, consequently, holding less learning capacity. Thereby, traditional RNNs are considered short-lived memory networks.

[6]Available at https://keras.io/.

The Long Short-Term Memory (LSTM) network is a specialized RNN capable of learning long-term dependencies. The model has memory units in hidden layers, each containing three gates: an input, an output, and a forget gate. The input gate is responsible for loading values from the input into memory. The output gate decides what memory information is presented as an input to the learning mechanism of the node. Lastly, the forget gate controls which information is discarded from memory.

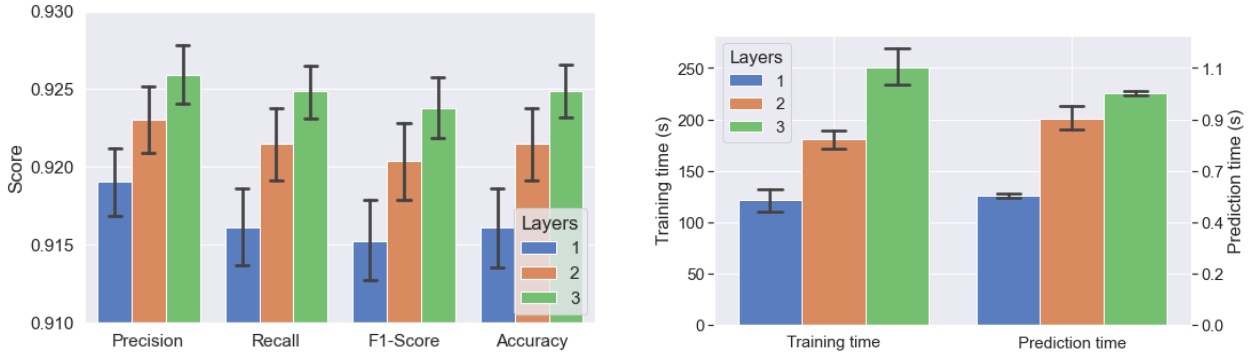
We use two different approaches to train LSTM neural network. In the first approach, the input is a single flow. Hence, flows are presented, processed and classified individually, in the same way that is done in XGBoost. In the second approach, we group flows of the same time window as the LSTM input. Therefore, the algorithm processes and classifies time windows containing 15 flows, similarly to the image processing of CNN. In both approaches, LSTM processes the ordered data to learn the temporal patterns. We built the LSTM model using Keras.

VI. Proposal Evaluation and Results

All experiments were performed on a computer equipped with an Intel i7 9700k CPU @ 3.0 GHz, NVIDIA GeForce GTX 1660 Super (6 GB) graphics card, and 32 GB memory.

We apply a 10-fold cross-validation approach for model training to allow its evaluation with different dataset arrangements and to reproduce the training several times, verifying the statistical relevance of the obtained result. For training the CNN and LSTM, we configure a limit of 100 epochs and an early stop callback with patience of 10 epochs, which prevent the model from overfitting. The validation set for these two neural networks contains 10% of the training set and is used to control and trigger the early stop callback. We evaluate model performance according to the following metrics: accuracy, precision, recall, and F1-score. We also evaluate the models for processing time, represented by training and prediction time. The latter is defined as the time needed to classify all samples in the test set. The former is the time for training the model until reaching the stop condition.

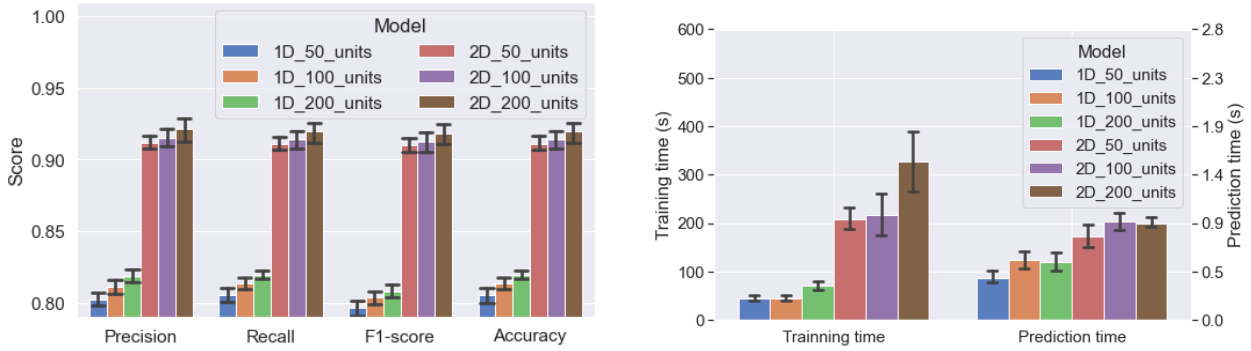
To evaluate the performance of CNN with the network flow data and choose the best neural network architecture that best fits the device identification problem, we implement three CNN models, and we evaluate the impact of adding layers on the overall outcome in terms of classification performance, training and prediction time. Hence, we define a basic block containing three layers. The first layer is a convolutional layer with a 3×3 kernel and filter size 32, followed by a max-pooling and batch normalization layers. The simplest model has only one copy of the basic block; the second one presents two blocks; and the third model, three blocks. All three models have a flatten layer, a fully connected layer containing 50 nodes, with linear rectification function (ReLu) as the activation function, and a batch normalization layer. The last layer is a fully-connected layer with 33 nodes with a softmax activation function that gives the classification outcome as the probability of each class.



(a) Classification performance.

(b) Training and prediction time.

Fig. 4. Comparison between the CNN models with 1, 2 and 3 layers of convolution, max pooling and batch normalization. (a) shows the classification performance and (b) shows the training and prediction time. The CNN model with 3 layers achieves the best classification results with a small increase in training and prediction time.



(a) Classification performance.

(b) Training and prediction time.

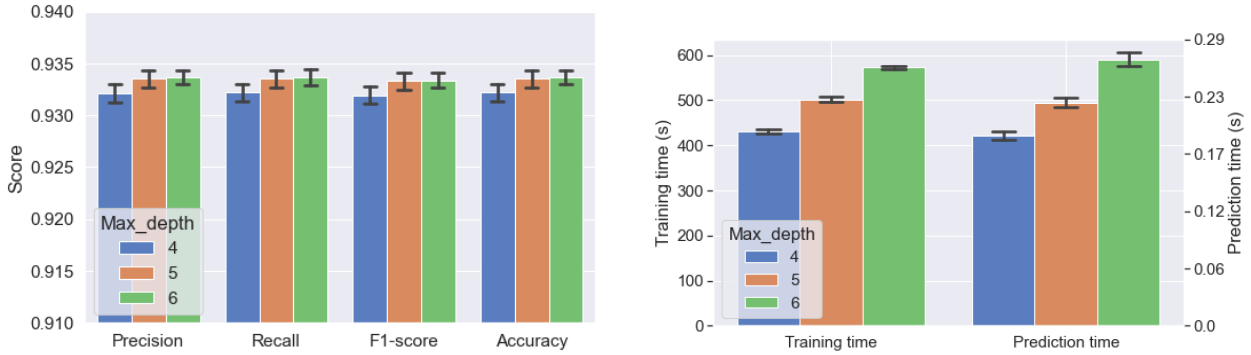
Fig. 5. Comparison between the LSTM models with 1D and 2D inputs, with 50, 100 and 200 units. (a) shows the classification performance and (b) shows the training and prediction time. The LSTM model with 2D input and 200 units achieves the best classification results with a increase in training time.

Figure 4(a) shows the results for the three analyzed CNN models. It is worth noting that all networks performed over 90% for device classification on all four metrics and that the performance gradually improves with the addition of the 3-layers blocks. Training and prediction time, as seen Figure 4(b), show that the performance growth is due to the increase in the network's complexity and processing time. Although adding complexity to the network architecture improves the classification performance, the additional processing time may be harmful to applications that work with a large amount of data, such as real-time network flow analysis.

We evaluate the LSTM network model following two different approaches. The first approach deploys a two-dimensional input, which is similar to the image concept used in the CNN network. The second approach uses a one-dimensional input, representing a single flow and features similar to the XGBoost model. Both models contain an LSTM layer and a fully connected layer with 33 nodes running a softmax activation function. The classification output of the LSTM model is the probability of the flow belonging to each class. In both approaches, we change the number of LSTM layer units and analyze the performance to choose the best configuration.

Figure 5(a) shows the result of the tested LSTM models. The two-dimensional input model performs up to 10% better in all analyzed metrics than the one-dimensional input LSTM model. This result demonstrates that, for LSTM networks, grouping flows into time windows optimizes the model performance. Regarding processing time, Figure 5(b) shows that the training time of the two-dimensional input LSTM is at least four times greater than the training time of one-dimensional input LSTM.

We train the XGBoost model with three different values, 4, 5, or 6, to set the maximum depth limit for the decision trees. We highlight that finding the best value for the parameter is essential because increasing its value may improve the classification performance at the cost of making the model more prone to overfitting. The evaluation metric used is multiclass logistic loss, and the other parameters remain with their default values. The XGBoost model deploys the approach of individual flow classification and, thus, the model input is composed of a 76-feature vector representing a flow. Figure 6(a) shows the results achieved with the different configurations. It is possible to observe that the model using 5 and 6 levels as the



(a) Classification performance.

(b) Training and prediction time.

Fig. 6. Comparison between the XGBoost models with maximum depth values of 4, 5 and 6. (a) shows the classification performance, and (b) shows the training and prediction time. The models with maximum depth values of 5 and 6 achieve comparable classification results, but the former is less complex and has shorter training and prediction time than the latter.

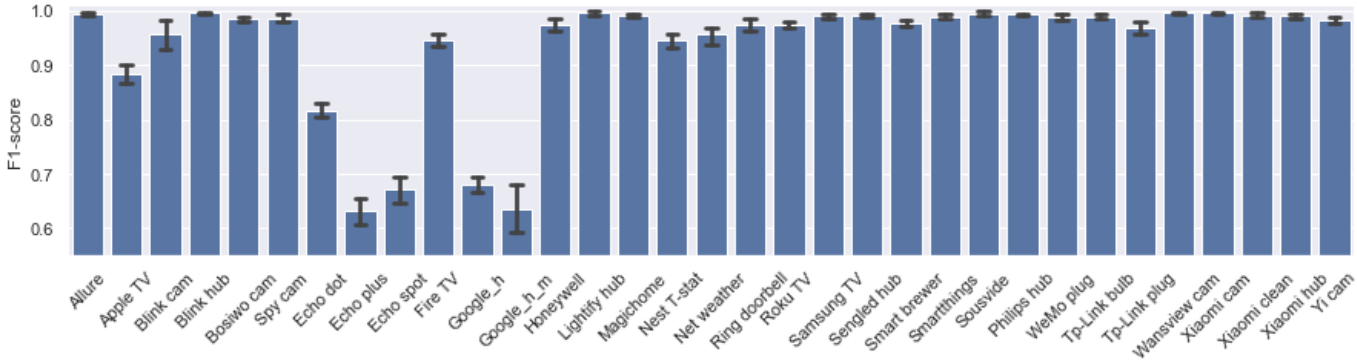


Fig. 7. F1-Score of each device class for CNN classification model. Smart home assistants, such as, Google Home (`Google_h`), Google Home Mini (`Google_h_m`), Echo Dot and Echo Plus, are the classes that represent the lowest performance for all classification models.

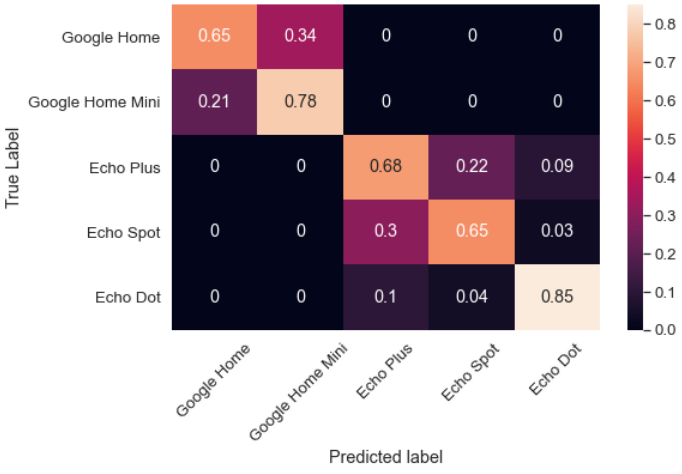
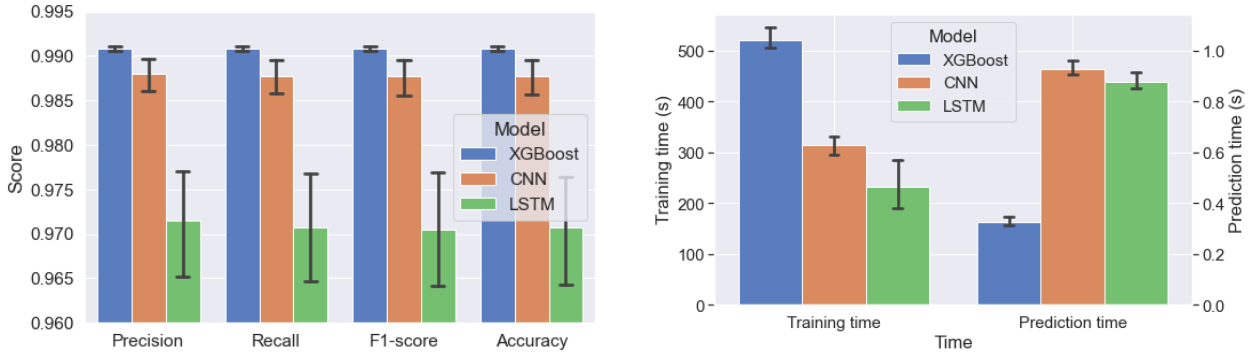


Fig. 8. Confusion matrix of the top five misclassified devices. Home assistant devices introduce misclassifications due to different versions of each device.

maximum tree size achieve similar performance. However, in the comparison of training and prediction time illustrated in Figure 9(b), the model with depth limit equals to 5 requires shorter processing time.

All models scored higher than 90% on all evaluated metrics and performed similarly for most devices. Figure 7 shows the classification result, detailed by device, of the CNN with three layers of convolution. It is possible to observe that the model had the worst performance in classifying the home assistant devices: *Google Home*, *Google Home Mini*, *Echodot*, *Echoplus* and *Echospot*. The confusion matrix, summarized in Figure 8, shows that the model confuses between *Google Home* and *Google Home Mini* devices and among *Echodot*, *Echoplus*, and *Echospot* devices. These devices have very similar behavior, as they deploy the same services and the most notable difference between devices in each group is the user-interfacing features. For simplicity, we do not present detailed results of each model. The *Google Home* and *Google Home Mini* devices are grouped into a new class called *Google Home Devices* and *Echodot, Echoplus, Echospot* devices are grouped into a new *Echo Devices* class.

We compare XGBoost, CNN, and LSTM models that achieved the best performance in former experiments: (i) CNN with three layers of convolution, max-pooling, and batch normalization; (ii) LSTM network with 200 units; and (iii) XGBoost with depth limit equals to 5. The input data are the grouped classes for assistant devices, resulting in a set of 30



(a) Classification performance.

(b) Training and prediction time.

Fig. 9. Comparison among XGBoost, CNN and LSTM models with 30 device classes. (a) shows the classification performance and (b) shows the training and prediction time. The XGBoost model achieves the best classification results and shortest prediction time, even though it has the longest training time.

different classes for classification.

Figure 9 shows the results achieved with the three models. All models show considerable classification improvement when the assistant devices are grouped. The results obtained with XGBoost are slightly superior to the CNN results, while the LSTM model had much lower performance. In addition to the best performance, the XGBoost algorithm performs a fine-grained device classification at flow-level resolution, while the CNN algorithm presents a coarse-grained classification at a time window resolution. A downside of the XGBoost model consists of its high training time, whose average value is greater than twice the training time of an LSTM network. Moreover, when comparing the prediction time, XGBoost is almost three times faster than CNN or LSTM due to the lower number of arithmetic operations XGBoost performs than neural network approaches.

VII. Conclusion

In this paper, we investigated privacy vulnerabilities of Internet of Things devices using a real network dataset. We propose a method to classify IoT devices using machine learning techniques based on statistical flow features that also apply to encrypted network traffic. Our proposal analyses only the network traffic data to recognize IoT devices in the network and, thus, reveals users' behavior and preferences. Furthermore, our approach employs a short time window to extract flows statistics, which allows online operations. We trained our proposal using three machine learning models with a dataset containing 33 devices. The results show that our method can achieve 99% accuracy, precision, and recall for the best tree-based configuration. Our future work will focus on recognizing device operating modes and, from that, infer user behavior patterns.

References

[1] "Gartner forecasts worldwide 5g network infrastructure revenue to grow 39% in 2021," Aug 2021. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-08-04-gartner-forecasts-worldwide-5g-network-infrastrucutre-revenue-to-grow-39pc-in-2021

[2] R. Ryan Daws — 7th September 2021 — TechForge Media Categories: IoT, 5G, Ai, and C. Vehicles, "Kaspersky: Attacks on iot devices double in a year," Sep 2021. [Online]. Available: https://iottechnews.com/news/2021/sep/07/kaspersky-attacks-on-iot-devices-double-in-a-year/

[3] D. M. F. Mattos, P. B. Velloso, and O. C. M. B. Duarte, "An agile and effective network function virtualization infrastructure for the internet of things," *Journal of Internet Services and Applications*, vol. 10, no. 1, p. 6, Mar 2019.

[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 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[5] J. Li, Z. Li, G. Tyson, and G. Xie, "Your privilege gives your privacy away: An analysis of a home security camera service," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 387–396.

[6] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 267–279.

[7] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of wi-fi traffic," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1033–1047, 2017.

[8] 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.

[9] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "Iot devices recognition through network traffic analysis," in *2018 IEEE international conference on big data (big data)*. IEEE, 2018, pp. 5187–5192.

[10] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, 2017, pp. 506–509.

[11] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd conference on local computer networks (LCN)*. IEEE, 2018, pp. 1–9.

[12] 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.

[13] M. Andreoni Lopez, D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "A fast unsupervised preprocessing method for network monitoring," *Annals of Telecommunications*, vol. 74, no. 3, pp. 139–155, Apr 2019.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.