IoT Event Classification Based on Network Traffic

Batyr Charyyev
School of Systems and Enterprises
Stevens Institute of Technology
bcharyye@stevens.edu

Mehmet Hadi Gunes
School of Systems and Enterprises
Stevens Institute of Technology
mgunes@stevens.edu

***Abstract*—The Internet of Things (IoT) consists of sensors and actuators that facilitate many aspects of our daily life. Compared to typical computing devices such as laptops and smartphones, these devices have a very limited set of functionalities and states. Researchers have shown that it is possible to infer the device type from its network traffic. In this paper, we show that an external observer that sniffs the network traffic of an IoT device can further classify device events and hence infer user actions by employing machine learning classifiers. We evaluate and compare the performance of ten machine learning algorithms in classifying 128 device events from 39 different devices. We analyze the impact of the user interaction through LAN and WAN as well as controllers such as Alexa voice assistant on the correct classification of device actions. We also inspect whether the region from which the device is impacts the performance of classifiers as researchers have shown that differing privacy restrictions lead to different external communications.**

I. Introduction

With the integration of IoT devices into our households, simple devices such as locks, bulbs, and plugs become sensors and actuators interconnected with the user through the Internet. These smart devices improve many aspects of our daily life by monitoring sensors such as temperature and energy as well as automating actions such as controlling the A/C and illumination. Data gathered from these devices can be employed for the realization of the Smart Environments and Smart Urban Ecosystems.

On the other hand, IoT devices increase security and privacy concerns. Since these devices have simple computing resources with almost no security protection, an attacker may use the IoT devices to gain unauthorized access to a network or even use them as botnets for larger attacks leveraging the pervasiveness of these devices. IoT devices also compromise the privacy of users as they continuously monitor users. An attacker can infer user actions, such as daily routines and sleeping patterns using the traffic generated by the devices. For instance, Kennedy et.al. showed that it is possible to infer voice commands on smart home speakers from the traffic rate of the device by employing website fingerprinting attacks [19]. This shows that even if the network traffic of the device is encrypted, meta-data of the packet, such as packet length and traffic rate still enables an external user to classify the state changes in the device such as turning on or off, user actions such as dimming the light of the light bulb.

In general, network traffic flow could be inspected to understand the characteristics and use of a computing system [9]. While there exist studies on IoT device type identification through network traffic [2], [14], there are few studies on device event classification [19]. Additionally, many devices have binary functionality such as being on/off [18]. There are also studies focusing on semantic behaviors (e.g., heartbeat, firmware check) of the IoT devices [23].

Studies on network traffic classification widely use machine learning algorithms as machine learning is very suitable for the classification problems. Previous studies on network traffic analyses use machine learning algorithms to classify operating systems [3], [12], smart phones [24], devices [15], and applications (e.g., Chrome, Spotify) [21], [22]. Recent studies on IoT device traffic analyses use machine learning for device identification [2], [14], and botnet/anomaly detection [10]. To get most out of the machine learning algorithms, it is important to choose the right classifier and fine-tune the parameters.

In this paper, we demonstrate that an external observer passively sniffing the network traffic is able to infer IoT device activities. We show that machine learning algorithms achieve high accuracy in classifying IoT device interactions and state. Different from previous studies on IoT device event classification [18], [19], [23], our study does not focus on a specific device or devices with binary functions. We also analyze the impact of different user interaction modes such as controlling the device through Alexa voice assistant, with an android application, or physically interacting with the device. We also inspect the change in classification accuracy when the device is controlled through LAN or WAN. Furthermore, we evaluate if the performance of classifiers is impacted by the device's superfluous communications due to less stringent privacy regulations. Contributions of this paper are as follows:

- Evaluate and compare the machine learning algorithms to classify IoT device events.
- Analyse the impact of different interaction modes with devices, on the performance of classifiers.
- Determine the influence of LAN vs WAN interaction with the device.
- Ascertain the effect of the region from which the device is at.

In the rest of the paper; Section II presents studies focusing on network traffic fingerprinting of IoT devices and evaluation of machine learning algorithms in the classification of network traffic. Section III describes the data collection, feature extraction, and parameter tuning used in this study. Section IV shows the evaluation results and Section V concludes the paper.

II. Related Work

In this section, we present studies focusing on IoT device and event identification through network traffic and evaluation of machine learning algorithms for traffic classification.

A. IoT Device Identification

Device identification can be performed by analyzing the network traffic of the device [4]. IoT Sentinel automatically identifies the type of device from the device's initialization communication [14]. It uses 23 network packet header features such as packet length, port number, list of protocols to classify the type of the device. IoTSense proposes to build a behavioral model of static and dynamic network features for fingerprinting devices [6]. The static model is based on protocols used by the device and the dynamic model considers the session interactions of the device. SysID builds a self-learning automated system for device identification from traffic flows [2]. SysID uses a genetic algorithm to determine relevant features in protocol headers and then use machine learning algorithms to classify device type using selected features. Charyyev et.al. [7] proposes LSIF that uses locality sensitive hashes of the traffic generated by IoT device for device identification. The LSIF does not require feature extraction from the traffic and frequent update the model. Valdez et.al. identifies a device by examining the Transport Layer Security (TLS) handshake protocol [11]. Authors build term frequency-inverse document frequency (TFIDF) models of the textual TLS features and determine the device from the principal component analysis. AUDI uses background communication traffic of IoT devices to identify the device type [20]. It uses a discrete Fourier transform to identify candidate periods for a given flow and employs k-Nearest Neighbors (k-NN) for device classification with extracted features.

B. IoT Event Classification

Device event classification is a process of classifying user interactions with the device such as watching the smart security camera, or state changes of the device such as from idle to active. Most of the IoT devices have a limited set of functionalities and states that could be recognized from their network traffic. Acar et.al. proposes a multi-stage privacy attack on IoT devices by identifying the actions, states, and ongoing user activities with machine learning approaches in a cascading style [1]. Antonio et.al. use a binary classifier to distinguish between IoT and non-IoT devices, then use a multi-class classifier to determine the type of device and device event [5]. Authors use the mean and standard deviation of packet length and the number of transmitted bytes with machine learning algorithms such as k-NN, Random Forest, Support Vector Machine (SVM), and Majority Voting. PingPong extracts packet-level signatures from the network packet sequence and flow direction to identify device events such as light bulb turning on and off [18]. Kennedy et.al. utilizes website fingerprinting attacks to classify the traffic of voice commands on smart home speakers [19]. Authors also evaluate an existing padding method as a countermeasure under voice command fingerprinting attacks. HomeSnitch classifies IoT device semantic behavior such as heartbeat, firmware check, motion detection to report device behaviors [23].

This study extends prior works by focusing on devices with multiple functionalities and analyzing the impact of different user interaction modes, the effect of LAN vs WAN interaction, and the influence of the region from which device is at.

C. Comparison of Machine Learning Approaches

Some studies compared different machine learning approaches in the analysis of network traffic flows for device, system, or application identification. Nizar et.al. compares the existing machine learning algorithms for IoT device fingerprinting [15]. Authors show that sliding windows can be efficiently used to identify the type of IoT device and increase the accuracy of classifiers by 18% compared to the [27] study that uses Dominant Protocol Analysis (DPA) feature set, static segmentation, and Random Forest. In their evaluation, AdaBoost achieves 95% accuracy, extra-trees 79.5%, random forest 78%, k-NN %78, and SVM 39%, respectively. Song et.al. compare k-NN, Decision Tree, and Artificial Neural Network (ANN) to classify operating system (OS) of computers in terms of accuracy and processing time [12]. Their evaluations showed that ANN has the highest classification performance with 94% followed by the Decision Tree 90% and k-NN 87%, respectively. ANN also had the lowest processing time followed by Decision Tree and k-NN. Sengupta et.al. evaluates 7 machine learning algorithms on classifying 175 different android applications with packet-level features augmented with bit sequence features [21]. They observe that Random Forest and Decision Tree achieve the highest accuracy with an accuracy of 89% and 87%, respectively, while SVM and AdaBoost achieve the lowest accuracy. Sina et.al. compares eleven machine learning algorithms to identify different applications (such as Google Chrome, Google Drive, One Drive, OneNote, Spotify, and WhatsApp) from the network traffic [22]. As features, authors use packets length, inter-arrival time, TCP window size, port numbers, and source and destination IP addresses. Evaluation results show that Random Forest, Bagged Trees, and XGBoost are the top-scoring machine learning algorithms achieving a precision of 90%.

III. Methodology

In this section, we provide details of the dataset used in our study along with the feature extraction process and parameter tuning of the machine learning algorithms.

A. Dataset

In our study, we use a dataset collected to analyze the information exposure of IoT devices [13]. The dataset consists of 81 different IoT devices deployed in two different testbeds: 46 devices deployed in the US and 35 deployed in the UK. Out of 81 devices, 54 of them are unique. In our analyses, we treated the set of common devices as the same device since they have the same set of functionalities and design implementation. We excluded devices that have less than 40

TABLE I: Parameters of classifiers with optimum value.

K-Nearest Neighbours (k-NN)

parameter	*neighbor*	*weight*	*algorithm*	*metric*	*ps*	Finds k closest samples for a given input x and labels x with the most common label among the k neighbors [12]
optimum	6	distance	auto	minkowski	1	

Support Vector Machine (SVM)

parameter	*C*	*kernel*	*degree*	*gamma*	*shrinking*	Finds the optimal boundary between different classes by constructing a hyperplane that separates the classes [21]
optimum	1.8	poly	3	0.4	False	

Decision Trees (DT)

parameter	*criterion*	*splitter*	*max depth*	*max features*	Produces decision rules from the data and creates a tree structure where for a given input x, the class label is assigned based on the decision rules within the tree [12]
optimum	entropy	best	15	auto	

Random Forest (RF)

parameter	*criterion*	*estimators*	*max features*	*bootstrap*	*warm start*	Combines multiple decision trees using several bootstrapped samples of the data and randomly chooses subset of features to find the best split for each node [15]
optimum	entropy	60	auto	False	True	

Extra Trees (ET)

parameter	*criterion*	*estimators*	*max features*	*bootstrap*	*warm start*	Trains multiple tree classifiers independently similar to the Random Forest classifier, but splits in a top-down manner [15]
optimum	entropy	140	auto	False	True	

AdaBoost (AB)

parameter	*algorithm*	*estimators*	*learning rate*	Iterates over classification weights by updating the weight of instances so that in the subsequent iteration the classifier focuses on instances that were incorrectly classified [22]
optimum	SAMME	90	1.25	

Gradient Boosting (GB)

parameter	*criterion*	*estimators*	*loss*	*learning rate*	*max depth*	Generates a prediction model from weak prediction models and then iteratively refines a new predictor to the residual errors made by the previous predictor [23]
optimum	friedman_mse	10	deviance	0.5	6	

Stochastic Gradient Descent (SGD)

parameter	*alpha*	*learning rate*	*shuffle*	*loss*	*penalty*	Implements regularized linear models with stochastic gradient descent learning [22]
optimum	0.01	optimal	True	log	L1	

Bernoulli Naive Bayes (B-NB)

parameter	*alpha*	*binarize*	*fit_prior*	Assumes that features are independent naive assumptions and produces a Bernoulli distribution to classify data based on the Bayesian theorem [22]
optimum	0	0.2	False	

Gaussian Process (GP)

parameter	*kernel*	*optimizer*	*multi class*	*warm start*	Determines the distribution that provides a good fit for the observed data [21]
optimum	Rational Quad	fmin_l_bfgs_b	OneVsOne	False	

measurements, and hence, in our evaluation, we ended up with 39 devices and 128 different device events. Interaction with the devices was through a companion app running on an Android Nexus 5X smartphone, an Echo Spot powered by Alexa, and physically interacting with the device. Dataset also consists of LAN and WAN experiments. In the LAN experiment, the companion app is run on the phone within the same network of the device allowing for direct communication (i.e., layer 2) between the phone and the device. In the WAN experiment, the phone is connected to a different network and hence traffic is directed to the cloud infrastructure to communicate with the device. Dataset also contains network traffic when a device is connected to the Internet through VPN tunnels, enabling devices in the UK testbed to connect to the Internet through the US testbed, and vice versa.

B. Machine Learning Classifiers

In this study, we test ten machine learning classifiers from the scikit learn package (`https://scikit-learn.org`) in Python. To find optimal parameters of the classifiers, we first collected a set of tunable parameters from the scikit learn documentation and explored all possible combinations of the parameters. Table I shows the list of classifiers with the optimum set of tuned parameters for each classifier. Note that, we only focused on the classification accuracy of the classifiers and did not consider the speed and computational resource requirement of the algorithms. When evaluating the classifiers we used four-fold cross-validation.

C. Feature Extraction

Features for traffic fingerprinting can either be extracted from individual packet headers or the sequence of packets. While individual packet headers are used do derive which protocols are employed in the communication, sequence features focus on statistical traits derived from the flow. In this study, we use widely employed traffic classification features and protocols [26]. In particular, we employ seven statistical values (i.e., sum, min, max, mean, interquartile range, standard deviation, and variance) on four different features of *ip.len*, *tcp.len*, *ip.ttl*, and *inter arrival time* of the frames. We also considered the presence of ten widely used protocols of *data*, *ssl*, *dns*, *eapol*, *icmp*, *x509sat*, *http*, *ntp*, *ssdp*, and *llc*. Note that, we did not consider protocols such as *arp*, *tcp*, *udp* and *bootp* as they were observed in most devices' traffic flow. To extract features such as ip.len, tcp.len and to check the existence of a protocol in the flow, we used the tshark tool (`https://www.wireshark.org`).

IV. Evaluation

In this section, we present the average performance of the classifiers, the impact of the interaction through ancillary devices (i.e., Alexa smart home assistant and Android App), and the effect of the interaction medium (i.e., LAN and WAN) on the accuracy of the classifiers.

	Random Forest	GradientBoosting	Extra-Trees	Decision-Trees	k-NN	AdaBoost	Gaussion Process	SVM	B-Naive Bayes	SGD	Random
Allure Alexa (4)	67	68	**69**	61	60	57	58	52	53	41	25
Amazon Cam (2)	**100**	99	99	99	99	99	99	99	99	99	50
Amcrest Cam (3)	**73**	63	65	65	62	69	57	51	33	38	33
Anova Sousvide (2)	52	49	52	48	59	56	**59**	45	48	49	50
Blink Cam (3)	**97**	93	96	91	89	91	87	65	96	76	33
Blink Hub (3)	**81**	79	78	76	79	70	77	67	58	65	33
Bosiwo Cam (3)	**100**	**100**	100	98	94	**100**	91	73	93	64	33
Echo Dot (4)	66	67	66	63	66	**69**	60	62	50	45	25
Echo Plus (6)	78	**85**	82	82	66	76	62	51	36	44	17
Echo Spot (4)	69	**74**	71	69	68	25	64	68	58	48	25
Flux Bulb (4)	**57**	54	55	52	53	49	50	45	36	40	25
Google Home (2)	**100**	**100**	**100**	81	73	**100**	77	81	51	49	50
Google Home Mini (2)	76	84	83	**97**	82	86	82	52	58	52	50
Honeywell T-stat (3)	75	73	74	71	**77**	67	73	54	58	55	33
Insteon (2)	71	**85**	77	75	69	80	71	72	61	25	50
Invoke Cortana (2)	**100**	**100**	**100**	89	91	**100**	85	98	80	50	50
Lefun Cam (3)	97	96	**100**	95	**100**	98	98	99	69	61	33
Lightify (4)	**72**	66	63	62	60	55	56	48	27	33	25
Luohe Cam (3)	**82**	79	74	67	64	71	63	27	38	48	33
Magichome Strip (4)	**58**	55	55	54	53	51	50	42	34	34	25
Nest T-stat (3)	**69**	68	68	64	60	65	58	40	41	41	33
Philips Bulb (4)	**64**	61	59	50	55	48	50	51	25	22	25
Philips Hue (4)	71	**75**	65	58	58	49	48	32	36	27	25
Ring Doorbell (3)	**98**	96	96	95	93	98	93	77	87	80	33
Samsung Fridge (4)	95	99	97	93	94	**99**	94	89	93	94	25
Sengled (4)	77	**79**	76	71	74	68	72	35	33	49	25
Smart-things (6)	53	51	51	49	50	49	48	44	**54**	49	17
Smarter Brewer (3)	78	87	**90**	82	76	90	77	57	62	55	33
Smarter iKettle (3)	62	**68**	60	56	52	55	47	42	33	29	33
TP-Link Bulb (4)	**88**	86	81	78	78	78	76	36	46	37	25
TP-Link Plug (2)	68	**69**	60	59	62	58	64	49	53	51	50
Wansview Cam (3)	**77**	71	66	63	58	67	56	36	53	23	33
WeMo Plug (2)	**78**	71	73	71	70	72	64	51	74	47	50
Wink-2 (4)	69	**71**	69	66	63	57	64	35	22	36	25
Xiaomi (3)	**54**	50	53	48	50	46	46	46	38	35	33
Xiaomi Cam (3)	82	**87**	86	80	75	81	71	52	48	59	33
Xiaomi Strip (4)	**64**	63	57	57	56	51	50	45	31	24	25
Yi Cam (3)	**87**	84	86	77	79	76	63	57	31	58	33
ZModo Doorbell (3)	65	63	63	65	51	**66**	48	43	38	49	33
Average	**76**	**76**	**75**	**71**	**70**	**70**	**67**	**56**	**52**	**48**	**33**

Fig. 1: Precision of the classifiers.

A. Classification of IoT Events

We employed machine learning algorithms with the identified optimal parameters reported in Table I. Figure 1 presents a heatmap for the average precision of classifiers along with random classification where the best performing classifier(s) for each device is indicated with a bold font. The recall values are mostly similar and not shown. Device names are followed by the number of different events for that device. Note that, due to the limited number of measurements for certain events of IoT devices in the dataset, the figure does not contain all functionalities of devices. While some devices yield high precision with the majority of classifiers, certain classifiers perform better than others. The tree-based classifiers of Extra-Trees and Random Forest as well as the Gradient Boosting produce an average both precision and recall of 75% to 76%. Sina et.al. [22] had similar poor performance with Gaussian Process and Bernoulli Naive Bayes, which indicates that IoT traffic flow does not follow Gaussian or Bernoulli distributions. Results also show that the limited number of packets impacts the performance of the Stochastic Gradient Descent (SGD) as it requires a random selection of a subset of the data at each iteration to optimize the gradient descent.

For 10 of the 39 devices, there is at least one classifier reaching 90% or better precision, and 5 others have a precision above 85%. We observe that classifiers have the highest precision on camera and surveillance devices achieving 76% on average and lowest on home automation devices (e.g., bulbs and plugs) achieving an average precision of 57%. This might be due to the amount of traffic generated by different devices as we did not observe any correlation between the number functionalities of a device and the performance of classifiers for the device. For instance, for surveillance device Yi Cam when device an event happens the average size of generated traffic is 270KB whereas for home automation device TP-Link Plug the average flow size is 25.6KB.

B. Impact of Interaction Mode

There are different ways to control an IoT device. For instance, a user can interact with the device physically, use smart home assistants such as Alexa, or through an accompanying application. In the previous section, we presented the results when the functionalities were performed under different interaction modes (such as turning ON the device with Alexa and Android app) fall into the same category. In this section, we further analyze if it is possible to distinguish the interaction modes and analyze its impact on classifier performance. With ancillary devices, the number of classes increases. For instance, if turning the device ON represented one class, considering the ancillary device yields three classes of turning ON by physically interacting with the device, with Alexa voice assistant, and with an accompanying application. Note that, however, in some cases, the number of classes does not increase as there are few instances to be analyzed (i.e., less than 40 measurements) when data is split by the interaction medium.

Considering the interaction through ancillary devices and separating the functionalities of the IoT devices into subclasses reduces the precision of the classifiers by 6% on average. Naive Bayes is the most impacted classifier with a precision drop of 14% while other classifiers precision reduces by 3% to 7%. Figure 2 presents classification accuracy to determine *interaction* mode (i.e., Android app, Alexa voice assistant or physical interaction), *event* type (e.g., turning ON/OFF, locking, unlocking etc.), and both *event and interaction* at the same time (e.g., unlocking with Alexa). The figure shows the devices for which there exist at least two different interaction modes. Device names are followed by the number of different instances for that device. For instance, Xiaomi Strip (4, 8) indicates that events fall into 4 classes without an ancillary device and 8 with an ancillary device. Note that there exist devices with an equal number of classes with and without consideration of ancillary devices such as Allure Alexa (4-4) due to the filtering of devices with insufficient data.

On average, the accuracy of classifying only the interaction mode is high achieving 93% precision and recall. Thus, it is possible to classify both event and interaction mode by

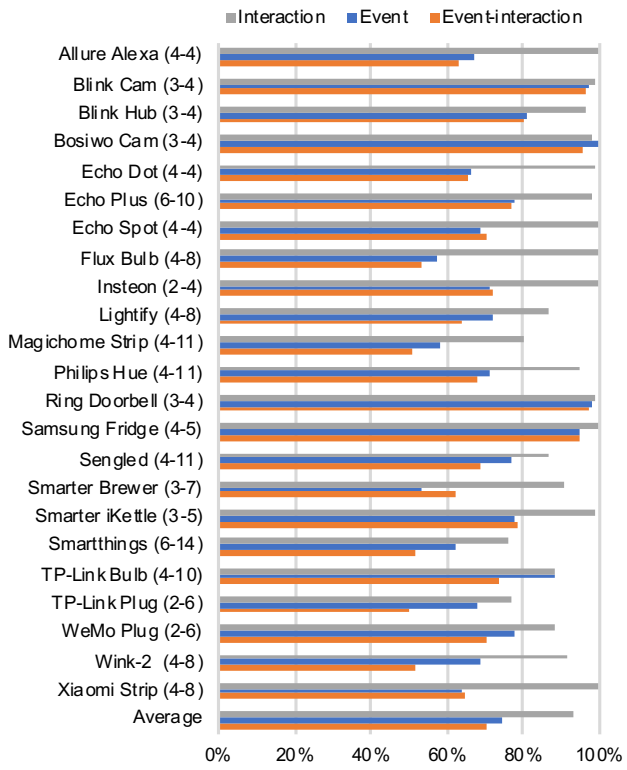

Fig. 2: Classification of *interaction* mode (i.e., Alexa assistant, android application, physical interaction), device *event* (e.g., turning ON/OFF) and both *event and interaction* (e.g., turning ON with Alexa).

cascading event classifier. This multistage classifier will have a little less precision compared to the event only classifier while it can identify interaction mode in addition to the device event. Overall, the classification of Android, Alexa, and physical interaction only events achieve nearly equal precision. However for the top five devices with the highest difference in *Event* and *Even-interaction* precision, we observed that classifying physical interactions within each other has less precision compared to classifying Android only or Alexa only interactions. This is most probably due to less amount of traffic generated when interacting physically with the device compared to interaction through the accompanying app or voice assistant.

TABLE II: Average Precision and Recall for Interaction Mode Classification

Classifier	Interaction	Event	Event-Interaction
Individual Best	94%, 93%	77%, 77%	72%, 70%
Random Forest	93%, 93%	75%, 74%	70%, 70%
Gradient Boosting	93%, 93%	74%, 74%	71%, 70%
Extra-Trees	93%, 92%	73%, 72%	69%, 69%
Decision-Trees	91%, 90%	68%, 68%	65%, 65%
k-NN	90%, 89%	68%, 68%	64%, 64%
AdaBoost	91%, 90%	64%, 62%	58%, 56%
Gaussian Process	86%, 84%	65%, 64%	59%, 57%
SVM	68%, 68%	45%, 44%	41%, 39%
B-Naive Bayes	72%, 68%	50%, 47%	36%, 35%
SGD	73%, 72%	44%, 44%	36%, 35%

Actual \ Predicted	on_android	off_android	lock_android	unlock_android	dim_android	color_android	on_alexa	off_alexa	lock_alexa	unlock_alexa	dim_alexa	color_alexa	on_local	off_local
on_android	35	35	5	5	0	0	9	8	0	0	0	0	2	1
off_android	34	42	4	6	1	1	5	4	0	0	1	0	1	1
lock_android	3	3	49	37	0	0	0	0	7	1	0	0	0	0
unlock_android	3	3	38	47	0	0	0	0	2	6	0	0	0	0
dim_android	1	1	1	2	77	15	0	1	0	1	2	0	0	0
color_android	0	0	1	0	12	87	0	0	0	0	0	0	0	0
on_alexa	14	10	2	0	1	1	36	26	1	0	7	1	2	0
off_alexa	12	8	0	0	1	0	24	41	1	1	8	2	2	1
lock_alexa	0	1	19	8	0	0	0	0	53	19	0	0	0	0
unlock_alexa	0	3	11	34	0	1	0	0	9	42	0	0	0	0
dim_alexa	3	3	1	0	0	0	21	20	0	3	50	0	0	0
color_alexa	0	1	0	0	0	1	5	5	1	1	1	85	0	0
on_local	25	17	3	0	0	0	5	9	2	0	0	0	26	14
off_local	23	22	7	2	0	0	7	7	0	0	0	0	20	13

Fig. 3: Confusion matrix for Smartthings, considering interaction mode (i.e., Android phone, Alexa voice assistant, and physical interaction).

Actual \ Predicted	on	off	lock	unlock	dim	color
on	48	43	3	2	3	1
off	42	47	2	5	4	1
lock	1	4	56	38	0	0
unlock	2	4	38	56	0	0
dim	10	12	2	3	68	7
color	2	2	1	0	5	89

Fig. 4: Confusion matrix without consideration of the ancillary device.

Additionally, we observed that choosing the best classifier for each individual device increases the precision only by 1% for *Interaction* and 2% for both *Event* and *Event-interaction* classification compared to the Random Forest. Table II presents the average classification precision and recall for each of the machine learning algorithm compared with the best individual device results to determine interaction medium, device event, and event-interaction pair. Overall performance of the top 3 algorithms are very similar while they are often in the same order of precision and recall.

While it is possible to distinguish between different types of user interactions, classifiers mostly confuse to distinguish a pair of events with opposite functions such as turning ON/OFF or locking and unlocking. Figure 3-4 presents the confusion matrix for the Random Forest classifier on the Smart-things device with and without an ancillary device. We can see that classifiers can separate commands through Alexa and Android app. However, they confuse commands with opposite functionalities such as turning ON and OFF or locking and unlocking. Similar behavior was observed for other devices. For example, for Echo Plus speaker, classifiers confuse to distinguish turning ON and OFF, which are a pair of commands with the opposite functionalities. This is probably because both commands carry the same packet header including the packet size with some changes in the payload, which is not analyzed.

C. Impact of Communication Medium

IoT devices can be controlled locally or remotely with the help of an accompanying application. We analyze the

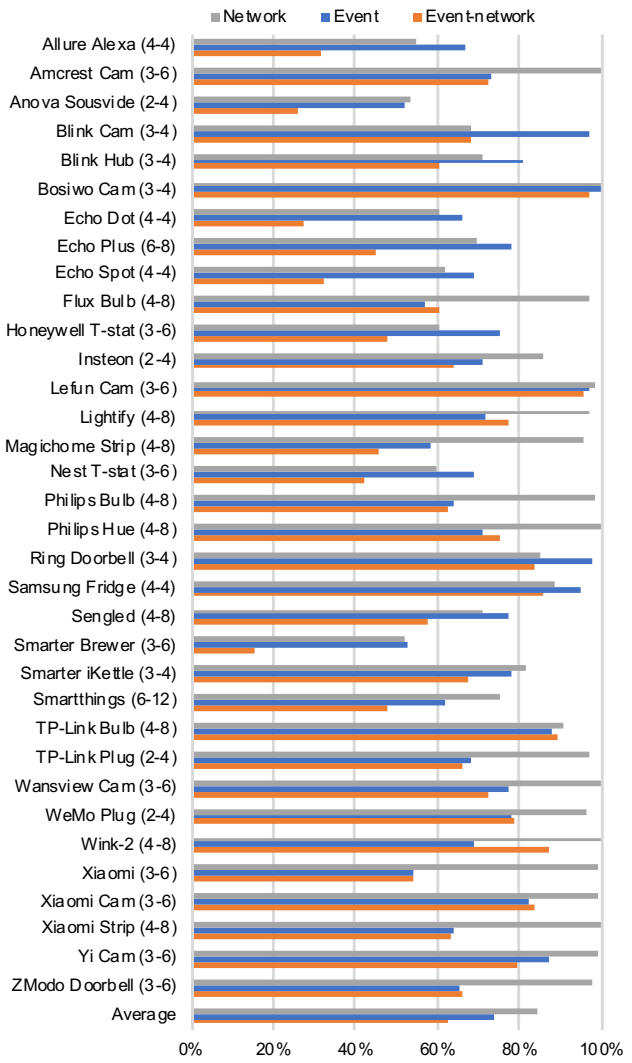

Fig. 5: Classification of the *network* medium (i.e., LAN or WAN), device *event* (e.g., locking and unlocking) and both *event and network* (e.g., locking remotely through WAN). Device names are followed by change in number of instances.

impact of communication medium (i.e., local area network and wide area network) on the performance of machine learning algorithms. Figure 5 shows the classification accuracy of Random Forest, the best performing algorithm for event classification, to classify the *network* (i.e., LAN or WAN) from which the device is controlled, the device event (e.g., locking, unlocking, etc.), and both event and network (e.g., locking remotely through WAN). Device names are followed by the number of different instances for that device. Network bars indicate the precision of the Random Forest when it infers the network type. Event bars indicate the precision when the algorithms infer the device events where data from different networks are combined. Finally, event-network bars indicate precision when both the network and event labels are inferred.

Overall, the classifiers can distinguish LAN and WAN interaction with a precision of 84% on average, events with an average of 74% and event and networks with an average of 63%. Similarly, average recall is 84% for the network, 74% for the events, and 62% when both are classified by the Random Forest. We observed that there is a negligible difference in accuracy to distinguish LAN only (average precision and recall of 72%) and WAN only (average precision and recall of 74%) interactions, meaning that the classifier's accuracy does not drop because of the LAN interactions or WAN interactions. We also observe that for most of the surveillance devices, (e.g., cameras and doorbells) classifiers are able to distinguish interactions performed through LAN and WAN except the Blink Camera. This implies that for surveillance devices an external observer can infer if the user is interacting with the device located in the same network or remotely. Table III presents the average classification precision and recall for each of the machine learning algorithm along with the best individual device results for network type, device event, and event-network. We observe that the overall performance of the top 3 algorithms are very similar, but remain to be in the same order of Random Forest, Gradient Boosting, and Extra-Trees.

TABLE III: Average Precision and Recall for Communication Medium Classification

Classifier	Network	Event	Event-Network
Individual Best	85%, 84%	76%, 76%	64%, 65%
Random Forest	84%, 84%	74%, 74%	63%, 62%
Gradient Boosting	83%, 83%	74%, 74%	62%, 62%
Extra-Trees	83%, 83%	72%, 72%	61%, 61%
Decision-Trees	81%, 81%	69%, 69%	56%, 56%
k-NN	80%, 80%	68%, 68%	55%, 55%
AdaBoost	83%, 83%	67%, 67%	55%, 55%
Gaussian Process	78%, 78%	65%, 64%	52%, 51%
SVM	65%, 66%	53%, 53%	31%, 32%
B-Naive Bayes	71%, 68%	50%, 48%	32%, 31%
SGD	66%, 65%	47%, 46%	29%, 30%

D. Impact of the Regional Difference

Countries have different regulations for network traffic, thus the traffic sent from and to the IoT device might change based on the region [13]. Authors observed that US devices send a higher volume of traffic compared to devices setup in UK testbed. Authors point to the more relaxed privacy regulations in the US with respect to the EU for the difference. Hence, we analyze if the region from which a device was setup impacts the performance of the classifier. We compare the precision of the Random Forest classifier on the set of common devices in the UK and US testbeds. Then, we compare devices in the UK testbed connecting to Internet through US testbed via a VPN tunnel (i.e., UK→US) and with devices in the US testbed connecting to the Internet through the UK testbed (i.e., US→UK).

Figure 6 presents the comparison of UK, US, UK→US and US→UK connections on the set of common devices in both testbeds. Note that in both UK and US→UK types of connection, devices access the Internet through the UK and vice versa. We observe that classifiers on average achieve 69% precision for the UK, 80% for the US, 67% for UK→US, and 78% for US→UK. This shows that classifiers achieve high precision for devices that were setup in US testbed (US

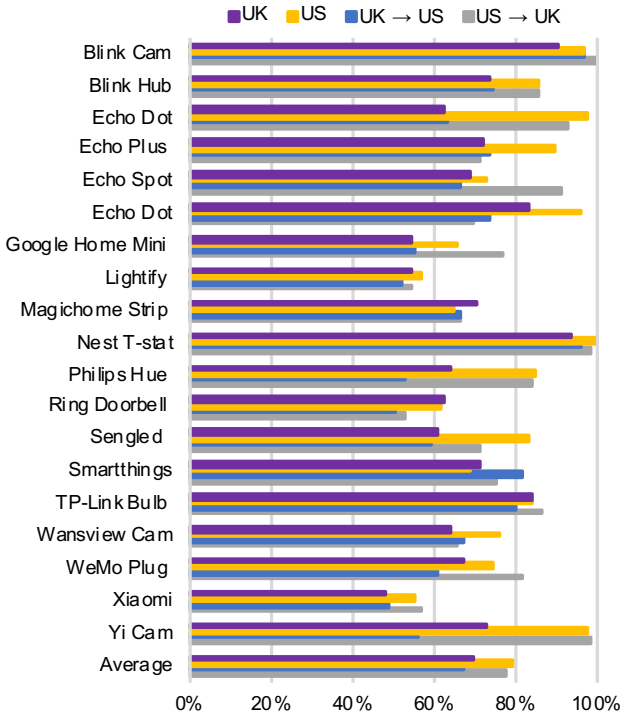

Fig. 6: Impact of the regional difference from which device accesses the Internet (note that, UK and US→UK connection is through UK).

and US→UK connections). This indicates the additional traffic generated by the devices that were setup in US testbed causes classifiers to have a higher precision compared to devices that were setup in UK testbed which has less traffic.

V. Conclusion

In this paper, we evaluated the ability of machine learning algorithms to classify IoT device events and states. We evaluated ten representative machine learning classifiers and observed that it is possible to infer device events and user interactions with an average of 90% on half of the devices and with 83% on three-quarter of the devices. In general, we observed that the tree based classifiers achieve high accuracy compared to other classifiers, followed by the Gradient and AdaBoost classifiers. We also evaluated the impact of interaction mode and observed that classifiers can distinguish the interaction through an Alexa assistant, an accompanying application, or through direct interaction. However, we observed that classifiers struggle to distinguish between device events with opposite functionalities (such as ON/OFF and locking/unlocking). Evaluation results also show that classifiers struggle to classify interaction medium (i.e., LAN and WAN) except for surveillance devices. Regional analyses showed that performance of the classifiers change significantly for devices in different testbeds. We observed that the country from which the device was set up may impact the accuracy of the classifiers as devices in the US incur additional network traffic due to less stringent privacy protections.

References

[1] A. Acar, et.al. "Peek-a-Boo: I see your smart home activities, even encrypted!" 2018, arXiv:1808.02741

[2] A. Aksoy and M.H. Gunes, "Automated IoT Device Identification using Network Traffic," 2019 ICC pp. 1-7. doi: 10.1109/ICC.2019.8761559"

[3] A. Aksoy, S. Louis, and M.H. Gunes,"Operating system fingerprinting via automated network traffic analysis", 2017 CEC pp. 2502-2509.

[4] A. Aksoy, and M.H. Gunes,"Operating system classification performance of TCP/IP protocol headers", 2017 LCN Workshops, pp: 112-120.

[5] A.J.Pinheiro, J.M. Bezerra, C.A.P.Burgardt, D.R.Campelo "Identifying IoT devices and events based on packet length from encrypted traffic" Computer Communications Volume 144, 15 Aug 2019, Pages 8-17

[6] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, I. Ray, "IoTSense: Behavioral Fingerprinting of IoT Devices", arXiv:1804.03852

[7] B. Charyyev, M.H. Gunes "IoT Traffic Flow Identification using Locality Sensitive Hashes", 2020 ICC, June 2020 Dublin, Ireland

[8] B. Li, G. Bebis, J. Springer, and M. H. Gunes "A Supervised Machine Learning Approach to Classify Host Roles Using sFlow," 2013 ACM Workshop on HPPN.

[9] B. Li, G. Bebis, J. Springer, and M. H. Gunes "A survey of network flow applications," Journal of Network and Computer Applications, vol. 36, no. 2, pp. 567-581. 2013.

[10] E.Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos and P. Burnap, "A Supervised Intrusion Detection System for Smart Home IoT Devices," in IEEE JIOT, vol. 6, no. 5, pp. 9042-9053, Oct. 2019.

[11] E. Valdez, D. Pendarakis and H. Jamjoom, "How to Discover IoT Devices When Network Traffic Is Encrypted," 2019 ICIOT pp: 17-24.

[12] J. Song, C. Cho, Y. Won "Analysis of operating system identification via fingerprinting and machine learning" COMPUT ELECTR ENG Vol.78, Sep.2019, pp:1-10

[13] 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." 2019 IMC, 267-279.

[14] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.R. Sadeghi, S. Tarkoma, "IoT Sentinel: Automated device-type identification for security enforcement in IoT" 2017 ICDCS pp: 2177-2184

[15] M. Nizar, R. Soua and T. Engel, "IoT Device Fingerprinting: Machine Learning based Encrypted Traffic Analysis," 2019 WCNC, pp. 1-8.

[16] P. Junges, J. François and O. Festor, "Passive Inference of User Actions through IoT Gateway Encrypted Traffic Analysis," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management, pp. 7-12.

[17] Pedro E. Carmo, Miguel L. Pardal "IoT Neighborhood Watch: device monitoring for anomaly detection" 2019 comunicação INForum

[18] R. Trimananda, et.al. "PingPong: Packet-Level Signatures for Smart Home Device Events" arXiv:1907.11797

[19] S.Kennedy, H. Li, C. Wang, H. Liu, B. Wang and W. Sun, "I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers," 2019 CNS, pp. 232-240. doi: 10.1109/CNS.2019.8802686"

[20] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi and N. Asokan, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," 2019 JSAC, vol. 37, no. 6, pp. 1402-1412

[21] S. Sengupta, N. Ganguly, P. De, and S. Chakraborty, "Exploiting Diversity in Android TLS Implementations for Mobile App Traffic Classification." 2019 WWW , pp:1657-1668.

[22] F.K. Sina, Y. Kaymak and R. Rojas-Cessa, "Identification of User Application by an External Eavesdropper using Machine Learning Analysis on Network Traffic," 2019 ICC Workshops, pp. 1-6.

[23] TJ OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.R. Sadeghi. "HomeSnitch: behavior transparency and control for smart home IoT devices." 2019 WiSec. pp:128-138.

[24] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic. "Who do you sync you are?: smartphone fingerprinting via application behaviour." ACM WiSec '13. pp: 7-12.

[25] V. Thangavelu, D.M. Divakaran, R. Sairam, S.S. Bhunia, and M. Gurusamy, "DEFT: A Distributed IoT Fingerprinting Technique" Internet of Things Journal Vol. 6, no.1, Feb 2019

[26] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang. "IoT device fingerprinting for relieving pressure in the access control." ACM TURC '19. Article 143, 8 pages.

[27] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in 2017 INFOCOM WKSHPS, May 2017, pp. 559–564.