Discovering Unknown IoT Devices

Franck Le, Mudhakar Srivatsa
IBM T.J. Watson Research Center
{fle, msrivats}@us.ibm.com

Abstract

IoT devices have become a prime target for attacks due to their poor security. Consequently, solutions have been developed to help administrators classify, and identify potentially vulnerable IoT devices. However, with the continuous arrival of new IoT devices, it is critical for solutions to also recognize unknown IoT devices. This is because neural networks traditionally classify instances into one of the known classes, leading to false and missed alarms. To discover unknown classes, the state-of-the-art approach relies on Bayesian Neural Networks. We therefore evaluate the suitability of Bayesian Neural Networks to discover unknown IoT devices on three independent datasets. In addition, we compare the results with a different approach, more specifically, rules extracted from deep models: Instances not matching any rule can be considered unknown. Surprisingly, rules offer comparable performance while providing interpretability, and requiring significantly less storage and compute resources. The results also reveal two major challenges, specific to IoT devices: First, the feature selected as the prediction outcome (e.g., device manufacturer, device type) impacts the performance given the fact that for some manufacturers, different device types (e.g., hub, camera) share the same infrastructure, while for other manufacturers, different device types (e.g., thermostat, camera) exhibit very different behaviors. Second, changes in devices behaviors (e.g., following software updates) can result in false positives.

1 Introduction

Due to their poor security, IoT devices have been the target of many cyber-attacks (e.g., [1–6].) Recently, researchers have analyzed 1.2 millions IoT devices, and revealed that more than half of them were still vulnerable to medium- or high-severity attacks [7]. As result, many solutions have been developed to help network administrators classify and detect potentially vulnerable IoT devices (e.g., [8–13]). However, one key requirement for an operational solution is the ability to recognize unknown IoT devices. This is because new IoT devices are continuously introduced into the market, and traditional neural networks classify every instance into one of the known classes, potentially leading to false or missed alarms. For example, feedforward neural networks may include a softmax layer as its final layer to produce probabilities for each of the classes, and there will always be one category with the maximum value. In other words, traditional neural networks cannot recognize when an instance does not belong to any of the class they have been trained for, and tend to be overconfident in their predictions [14]. The problem of recognizing unknowns is not specific to IoT devices, and the state-of-the-art approach relies on Bayesian Neural Networks. The main idea consists in attaching a probability distribution to all model parameters (e.g., weights and biases). To classify an instance, one would run multiple forward passes of the network (e.g., 100's), each with a new sample of weights and biases. From the probability distribution of the output values, one can then derive the confidence and uncertainty in each of the outputs. In particular, a high uncertainty can indicate that the instance belongs to an unknown class.

In this paper, we seek to answer several questions: First, how effective are Bayesian Neural Networks in discovering unknowns in the IoT context? Second, given that recent studies have demonstrated that rules extracted from deep models can be more suitable for network traffic analysis than deep models, allowing to achieve the required high speeds, how do rules compare to Bayesian Neural Networks? The question is also motivated by the facts that rules not only are intuitive (any instance not matching any rule can be classified as unknown), but also offer interpretability. Finally, do IoT devices present unique challenges, with respect to the discovery of unknowns?

To answer these questions, we train a set of Bayesian Neural Networks, and conduct extensive evaluation to compare their performance with those of rules extracted from deep models on three independent packet traces of IoT devices. We focus on the Domain Name System (DNS) traffic, i.e., the DNS queries (e.g., ntp.amazon.com, updates.netgear.com) that each device submits over a configurable period of time to predict the device manufacturer (e.g., Amazon, Netatmo, etc.), or device type (e.g., Amazon Echo, Netatmo Welcome, etc.) Previous studies also focused on the DNS traffic as it is typically in cleartext even when the subsequent connections are encrypted, and DNS traffic from IoT devices have been found to exhibit distinct patterns. More specifically, IoT devices tend to query few domains [9], and periodically connect to their home networks [10] (e.g., to check for software up-

dates, and report metrics).

The evaluation reveals four main results, and observations.

- The Bayesian Neural Networks (BNN) show promising results: When training the BNNs on the dataset containing the largest number of IoT devices to predict their device type, and applying them to discover unknown IoT device types in an independent and publicly available dataset of IoT devices [9], the BNNs can recognize the unknown IoT device types with a precision of 0.94, a recall of 0.73, and a resulting f1-score of 0.82.
- Extracted rules from an attention-based model can surprisingly achieve comparable, if not even better, results than BNNs. Using the same training dataset, and independent testing dataset, rules detect unknown IoT device types with a precision of 0.96, a recall 0.73, resulting in a f1-score of 0.83. The better performance suggests that as reported in previous studies (e.g., [15]), rules can offer better generalization performance than deep models.
- While the features of IoT devices (e.g., device manufacturer, device type, operating system, etc.) can be seen as falling into a class-hierarchy tree, the class selected as the prediction outcome of the classifiers can have a significant impact on the ability to detect unknown IoTs. For example, when training the deep models to predict, not the device type, but the device manufacturer, the performance of both the BNNs and extracted rules in detecting unknown IoT device manufacturers drop considerably to f1-scores of 0.47, and 0.50 respectively. This is because different devices (e.g., hub, camera) from a same manufacturer can exhibit very different behaviors. As such, when training on some types (e.g., hub), any other device type from the same manufacturer (e.g., camera) can result in false positives. In other words, the device manufacturer may be too high in the class-hierarchy tree. Classes too low in the class-hierarchy tree can also result in poor performance, more specifically leading to false negatives. The evaluation section illustrates and discusses those cases in further details.
- IoT devices may exhibit different behaviors following a software, or firmware update, and consequently be classified as unknown. For example, when applying the BNNs and extracted rules on a much more recent packet trace of IoT devices, many IoTs were incorrectly classified as unknown due to their new behaviors. To reduce the false alarms, solutions should therefore be able to distinguish new behaviors from known IoTs from behaviors from unknown IoTs.

The rest of the paper is organized as follows. Section 2 provides a brief background on the two methods we evaluate to detect unknown IoTs. Section 3 describes the datasets. Section 4 presents the methodology. Section 5 presents the results. Finally, Section 6 concludes the paper, and discusses future work.

2 Background

This section provides a brief overview on Bayesian Neural Networks, and on recent work for extracting rules from attention-based models.

2.1 Bayesian Neural Network

While traditional deep neural networks have been applied successfully to many fields, they tend to be overconfident in their predictions, and overfit especially with smaller training datasets. Bayesian neural networks were designed to address these shortcoming by incorporating a measure of uncertainty in the predictions. More specifically, while traditional neural networks have fixed weights and biases, Bayesian Neural Networks attach a probability distribution to all weights and biases. The probability distributions on the weights can for example be learned through Bayes by Backprop [16]. Then, to classify an instance, multiple forward passes of the network are run, each with a new sample of the weights and biases. The set of output values ultimately provide a probability distribution from which confidence and uncertainty in the outputs can be derived.

2.2 Extracted Rules

In previous work [17], we found that while deep neural networks can achieve high precision and recall, their inference latency can be 4 to 6 orders of magnitude larger than those of pattern matching based rules commonly used in Network Intrusion Detection Systems [18, 19], and prevent the high throughput typically required in network analysis from being attained. As such, we proposed to first train an attention based model [20], the state-of-the-art architecture for working with sequences, and then derive rules from the model [17]. More specifically, we rely on the readily available attention weights to identify important domains, and to form candidate if-then clauses. Finally, we select the rules that achieve the highest performance for the desired metric (e.g., precision, recall, or f1-score). In addition, to decreasing the inference latency and memory storage by 4 orders of magnitude, rules also offer interpretability and better generalization performance.

3 Datasets

We conduct the evaluation on three independent packet traces from IoT devices in different environments. For each trace, we focus on the DNS traffic. More specifically, we define an instance as the sequence of DNS requests submitted by an IoT device over a period of a day. Each instance is labelled with the manufacturer (e.g., *Amazon*), and device type of the IoT device (e.g., *Amazon Echo*). The device type consists of the concatenation of the device manufacturer (e.g., *Amazon*), and device model (e.g., *Echo*). As such, the device type can be considered as more specific than the device manufacturer. We preprocess the DNS traffic by discarding DNS queries which are not conducive to the classification objective, i.e., predicting the manufacturer or device type of the IoT device. This is similar to the elimination of stop words in Natural Language Processing. More specifically, we eliminate DNS queries to common services (e.g., pool.ntp.org, time.nist.gov, google.com, www.google.com), those that terminate in ".arpa", and those that are sent only in the local network (e.g., DNS queries ending in ".local") as they may not be observable at the point of packet capture. Table 1 summarizes the characteristics of each dataset. While datasets 1 and 3 were captured in a private lab, dataset 2 is from a publicly available trace [9].

Dataset	1	2	3
Time of capture	01/01/16-12/31/17	09/22/16-10/11/16	08/01/2019
Number of devices	106	23	34
Number of manufacturers	40	14	21
Number of device types	56	21	27
Number of instances	10298	319	493

Table 1: Summary of datasets

4 Methodology

We assume N independent datasets, and adopt a leave-one-out approach using $N-1$ datasets for training/validating, and one dataset for testing to ensure that the testing dataset and training/validating datasets are independent. More specifically, we split the $N-1$ datasets into training and validating datasets according to a 80:20 ratio, and using stratified sampling. The training portion is used to train the classifiers while the validating portion is used to reduce the risk of overfitting.

• **Bayesian Neural Network:** We train k Bayesian Neural Networks, with one Bayesian Neural Network for each class. For example, if the category to be predicted is the device manufacturer, we train a Bayesian Neural Network for *Amazon*, *Belkin*, *Netgear*, etc. The architecture of the Bayesian Neural Networks consists of two hidden layers of 128 units using the Rectified Linear Unit (ReLu) activation function.

Because instances consist of sequences of DNS queries submitted by an IoT device over a day, we convert each instance into a vector by executing the following three steps:

1. First, for each instance, we identify the top M domains. The goal of this step is to identify the most important domains. More specifically, we compute the Term-Frequency Inverse-Document-Frequency (TF-IDF) of each domain. Compared with traditional Natural Language Processing, each domain is equivalent to a word, each instance is equivalent to a document, and the datasets are equivalent to the corpus of the documents. For each instance, we retain the M domains with the highest TF-IDF values. The set of retained domains form the vocabulary $\mathcal{V}$.
2. Second, for each instance, we retain only the domains present in the vocabulary $\mathcal{V}$ previously computed, and filter all other domains. The goal of this step is to reduce the dimensions of the vectors.
3. Third, we recompute the TF-IDF values for each domain and instance. Compared to the computation in Step 1, only the domains in $\mathcal{V}$ are present. Each instance is converted into a vector where the dimensions represent the domains in $\mathcal{V}$, and the values represent the corresponding TF-IDF values.

We then balance the dataset through oversampling [21], and train the Bayesian Neural Networks on the resulting dataset. At classification time, each Bayesian Neural Network provides a probability representing whether an instance belongs to that class. More specifically, given an instance, each Bayesian Neural Network BNN_i runs M forward passes of the network, each with a new sample of weights and biases, to generate the probability that the instance belongs to the corresponding class. If the probability is lower than a set threshold T_i, the instance is considered as not pertaining to that class. If an instance is not classified as any class, it is marked as unknown. To set the threshold T_i, we gradually decrease the threshold from 1.0 to 0.0, and select the value that produces the highest f1-score on the validating dataset.

Unknown	Bayesian NN	Rules
True Positive	126	126
False Positive	8	5
True Negative	139	142
False Negative	46	46
Recall	0.732	0.732
Precision	0.940	0.961
f1-score	0.823	0.831

Table 2: Experiment 1: Summary of results

• **Extracted rules from attention-based model:** We first train an attention-based model [20] to predict the desired classes. Compared with traditional neural machine translation, each instance is equivalent to an input sentence, each domain in the instance is equivalent to a word, and the classes are equivalent to the output sentences. The encoder and decoder in the attention-based model consist of GRUs with 1024 hidden units using the sigmoid activation function.

We then extract rules from the attention-based model using the algorithm described in [17]. More specifically, for each class, we identify the top W domains with the largest attention weights, form candidate rules through disjunction of those identified domains, and select the rules that provide the largest f1-score on the validating dataset. An example of an extracted rule could be: *If a device queries the domain ntp.amazon.com, then its manufacturer is Amazon.* Any instance not matching any rule is classified as unknown.

5 Results

5.1 Experiment 1: Device Type (2016)

In this setting, we train the classifiers to predict the device type of the IoT devices (e.g., amazon-echo, belkin-wemoswitch, pixstar-fotoconnect, tp-link-camera, etc.) We use dataset 1 as the training/validating dataset, and dataset 2 as the testing dataset.

From the 319 instances in dataset 2, 147 instances belong to classes seen in the training dataset (dataset 1), and 172 instances belong to classes not seen, and should therefore be classified as unknown.

True Labels	amazon echo	belkin motion sensor	belkin plug	belkin wemo switch	google camera	ihome-plug	lifx bulb	netatmo camera	netatmo weather station	pixstar fotoconnect	samsung hub	smartlabs hub	unknown	withings aura
amazon echo	20	0	0	0	0	0	0	0	0	0	0	0	0	0
belkin motion sensor	0	19	0	0	0	0	0	0	0	0	0	0	1	0
belkin plug	0	0	0	0	0	0	0	0	0	0	0	0	0	0
belkin wemo switch	0	0	19	0	0	0	0	0	0	0	0	0	1	0
google camera	0	0	0	0	5	0	0	0	0	0	0	0	0	0
ihome plug	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lifx bulb	0	0	0	0	0	0	10	0	0	0	0	0	0	0
netatmo camera	0	0	0	0	0	0	0	20	0	0	0	0	0	0
netatmo weather station	0	0	0	0	0	0	0	0	20	0	0	0	0	0
pixstar fotoconnect	0	0	0	0	0	0	0	0	0	13	0	0	0	0
samsung hub	0	0	0	0	0	0	0	0	0	0	1	0	3	0
smartlabs hub	0	0	0	0	0	0	0	0	0	0	0	0	0	0
unknown	0	0	0	0	0	10	0	0	0	0	0	16	126	20
withing aura	0	0	0	0	0	0	0	0	0	0	0	0	0	15

Table 3: Experiment 1: Confusion matrix of exracted rules

Because of space limitation, the confusion matrices are omitted. Instead, Table 2 provides a summary of the results, and we discuss the false negatives and false positives: The Bayesian Neural Networks generate 46 false negatives, and 8 false positives. The 46 false negatives consist of 10 ihome-switch, 16 smartlab-cameras, and 20 withings-scale. Those device types are not present in the training dataset (dataset 1), and should have been classified as unknown, but were classified as known. They were actually classified as 10 ihome-plug, 16 smatlab-plug, and 20 withings-sleep_sensor, respectively. Table 3 represents the confusion matrix for the extracted rules. They also resulted in 46 false negatives, which are the same instances that were incorrectly classified by the Bayesian Neural Networks as knowns, and the classes predicted by the extracted rules are identical to those predicted by Bayesian Neural Networks. We observe that all those instances were incorrectly classified as another device type from the same manufacturer. The reasons are that although different, those device types are from the same manufacturer and therefore share the same infrastructure (e.g., home servers).

Extending the classifiers to not only take into account information from DNS, but also other protocols (e.g., HTTP, SSL, etc.) can alleviate the problem. The reason is that different device types (e.g., smartlab cameras and smartlab hubs) are likely to exhibit differences in other protocols (e.g.. HTTP User Agent).

5.2 Experiment 2: Device Type (2019)

This experiment is similar to the previous one, but we train and test the classifiers on different datasets. More specifically, we still train the classifiers to predict the device type of the IoT devices, but we use both datasets 1 and 2 as the training/validating datasets, and dataset 3 as the testing dataset.

Table 4 summarizes the results. As with the previous experiment, the extracted rules achieve a larger f1-score than the Bayesian Neural Networks. However, compared with the previous experiment, the overall f1-scores dropped, for example, from 0.823 to 0.553 for the Bayesian Neural Networks, and from 0.831 to 0.641 for the extracted rules. In particular, the number of false positives increased significantly, from single digit false positives (e.g., 8 for the Bayesian Neural Networks, and 5 for the extracted rules) to hundreds of them (e.g., 158 for the Bayesian Neural Networks, and 122 for the extracted rules).

Unknown	Bayesian NN	Rules
True Positive	174	198
False Positive	158	122
True Negative	37	73
False Negative	124	100
Recall	0.583	0.664
Precision	0.527	0.618
f1-score	0.553	0.641

Table 4: Experiment 2: Summary of results

Device Type	Bayesian NN	Rules
amazon-echo	87	87
samsung-hub	22	22
google-thermostat	22	22
google-camera	16	2
philips-bridge	9	9
samsung-tv	1	1
d_link-camera	1	1

Table 5: Experiment 2: False Positives

To better understand the issue, we look more closely at those false positives: Table 5 lists the device types that were present in the training set, but classified as unknown. We

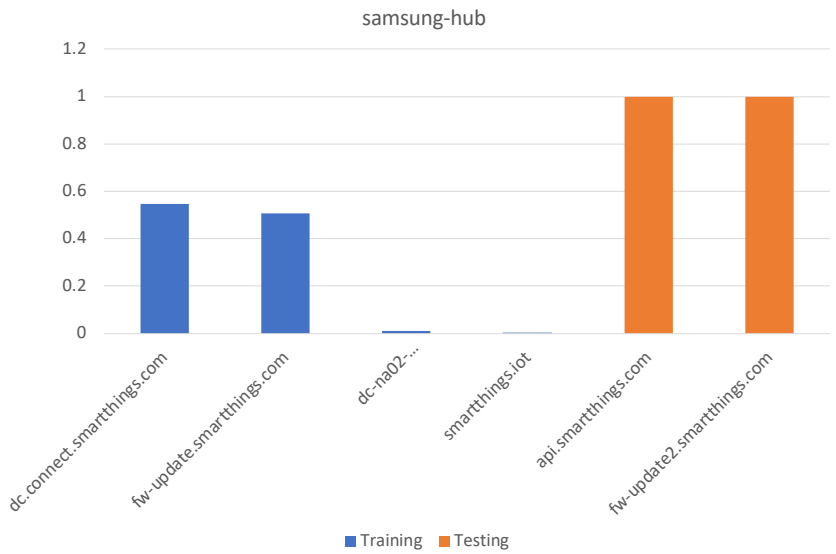

Figure 1: Experiment 2: Distribution of the domains queried by the samsung-hubs in the training dataset (2016), and testing dataset (2019)

observe that both the Bayesian Neural Networks, and the extracted rules resulted in most of the same false positives. The largest class with the most false positives is "amazon-echo". Interestingly, out of the 88 instances labeled "amazon-echo" in the testing dataset, 87 of them are classified as unknown. In the training dataset, the attention-based model revealed that the domain "ntp.amazon.com" received the most attention across all the instances in the class "amazon-echo". However, none of the instances from the testing dataset submits any query to that domain. We speculate that the amazon devices exhibit different behaviors in 2016, and 2019: More specifically, while the training dataset consists of amazon echo observed in the year 2016, the testing dataset is from 2019. The amazon echos in the testing datatset are probably second or even third generation devices with very different behaviors from those in 2016. Those very different behaviors cause those instances to be misclassified as unknown.

Similarly, for the second class of device type that has the largest number of false positives, i.e., samsung-hub, the distribution of domains in the training and testing datasets have even no overlap. Figure 1 compares the distribution of the domains from the instances in the training, and testing datasets. The y-axis represents the fraction of instances that query a specific domain. For example, 54.66% of the instances in the training dataset query the domain "dc.connect.smartthings.com". The histogram shows that the samsung-hubs in 2016 (training), and those in 2019 (testing), query completely different domains (e.g., fw-update.smartthings.com *versus* fw-update2.smartthings.com).

5.3 Experiment 3: Device Manufacturer

In this experiment, the classifiers are trained to predict, not the device type, but the device manufacturer of the IoT devices. The experiment is motivated by the observation in Section 5.1 that devices from the same manufacturer (e.g., smartlab-cameras and smartlab-hubs, ihome-switch and ihome-plug, withings-scale and withings-sleep_sensor) may be confused into each other, as they share the same infrastructure (i.e., home servers). We train the classifiers using dataset 1, and test them on dataset 2.

Unknown	Bayesian NN	Rules
True Positive	41	43
False Positive	83	86
True Negative	193	190
False Negative	2	0
Recall	0.953	1.000
Precision	0.330	0.333
f1-score	0.490	0.499

Table 6: Experiment 3: Summary of results

Device Type	Bayesian NN	Rules
tp-link	28	28
samsung	23	23
google	17	20
withings	15	15

Table 7: Experiment 3: False Positives

Table 6 summarizes the results. Both the Bayesian Neural Networks and extracted rules achieve a high recall. However, they also result in a large number of false positives, i.e., instances that are incorrectly classified as unknown.

Table 7 shows the classes where the false positives belong to: 28 instances from the testing dataset and labeled as "tp-link" were classified by both the Bayesian Neural Networks and the extracted rules as unknown. Because the training dataset did include devices from that vendor, those instances should not have been classified as unknown. However, looking more closely ath those instances, it turns out that the de-

vices belong to different types, and exhibited different behaviors explaining the results. For example, the testing dataset contained 15 instances from tp-link plugs, and 13 instances from tp-link cameras, whereas the dataset consisted of instances from tp-link hub.

Most of the other false positives can be explained by the same reason. For example, the 20 false positives from google correspond to google smoke alarm detectors whereas the training dataset consisted of google chromecasts, google cameras, google thermostats and other device types but not smoke alarm detectors whose behaviors differed significantly from other google devices.

More generally, if considering that the features of IoT devices (e.g., device type, device manufacturer) belong to a class-hierarchy tree, experiment 1 (Section 5.1) showed that a class low in the hierarchy (e.g., device type) can result in false negatives, whereas these results demonstrate that a class high in the hierarchy can result in a large number of false positives, when trying to detect unknown IoTs.

6 Conclusion and Future Work

While most of the existing work has focused on training classifiers to detect specific classes of IoT devices (e.g., [8–13]), in this paper, we evaluate the effectiveness of Bayesian Neural Networks, and rules to recognize *unknown* IoTs.

The results are promising: Both approaches can achieve f1-scores larger than 0.8. In addition, rules can achieve comparable, if not even better, performance than Bayesian Neural Networks while also providing interpretability, and requiring significantly less storage requirement, and inference latency, making them more suitable to network traffic analysis.

The results also revealed two new challenges that are specific to the context of IoT. First, changes in behaviors, e.g., following a software or firmware update, can cause IoTs to be incorrectly classified as unknown. As such, to reduce the amount of false positives, solutions must be developed to distinguish actual unknown IoT devices from new behaviors from known IoT devices. Second, the experiments exposed that the class selected to be the prediction outcome of the classifiers (e.g., device manufacturer, device type) can directly impact their effectiveness. More specifically, a class that is low in the class-hierarchy tree can lead to false negatives, while a class that is high can result in false positives. In future work, we will consider only the domains' suffixes, instead of the complete domains, to identify IoT devices. In addition, we will explore whether the integration of additional protocols (e.g., HTTP, SSL) can mitigate the above issue, and lead to better performance.

References

[1] L. Hautala, "Why it was so easy to hack the cameras that took down the web," in *CNET Security*, October 2016.

[2] D. Palmer, "175,000 IoT cameras can be remotely hacked thanks to flaw, says security researcher," in *ZDNet*, July 2017.

[3] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things," in *HotNets*, 2015.

[4] "Fridge sends spam emails as attack hits smart gadgets." `http://www.bbc.com/news/technology-25780908`. [Online; accessed 30-March-2020].

[5] "Hackers attack shipping and logistics firms using malware laden handheld scanners." `http://www.securityweek.com/hackers-attack-shipping-and-logistics-\firms-using-malware-laden-handheld-scanners`. [Online; accessed 30-March-2020].

[6] "Smart meters can be hacked to cut power bills." `http://www.bbc.com/news/technology-29643276`. [Online; accessed 30-March-2020].

[7] "2020 unit 42 iot threat report." `https://unit42.paloaltonetworks.com/iot-threat-report-2020/`.

[8] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel demo: Automated device-type identification for security enforcement in iot," in *IEEE ICDCS*, 2017.

[9] 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 *IEEE Infocom Workshop Smart Cities and Urban Computing*, 2017.

[10] H. Guo and J. Heidemann, "IP-Based IoT Device Detection," in *Workshop on IoT Security and Privacy*, ACM, 2018.

[11] A. Bremler-Barr, H. Levy, and Z. Yakhini, "IoT or NoT: Identifying IoT Devices in a ShortTime Scale," 2019.

[12] Y. Meidan, M. Bohadana, A. Shabtai, J. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis," 2017.

[13] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019.

[14] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles," in *International Conference on Neural Information Processing Systems*, 2017.

[15] I. Cloete, *Knowledge-Based Neurocomputing: Past, Present, and Future*, p. 1–26. Cambridge, MA, USA: MIT Press, 2000.

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," *ArXiv*, vol. abs/1505.05424, 2015.

[17] F. Le and M. Srivatsa, "Deriving Interpretable Rules for IoT Discovery through Attention," in *Proceedings of the International Conference on Internet of Things*, 2020.

[18] V. Paxson, "Bro: a System for Detecting Network Intruders in Real-Time," *Computer Networks*, 1999.

[19] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *USENIX Conference on System Administration*, 1999.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *International Conference on Learning Representations*, 2015.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority over-Sampling Technique," *J. Artif. Int. Res.*, 2002.