Smart Recon: Network Traffic Fingerprinting for IoT Device Identification

Jay Thom, Nathan Thom, Shamik Sengupta, Emily Hand
Department of Computer Science and Engineering, University of Nevada, Reno, USA
1664 N. Virginia street m/s 0171 Reno, NV 89557 775-784-6905
Email: jthom@unr.edu, nathanthom@nevada.unr.edu, ssengupta@unr.edu, emhand@unr.edu

***Abstract*—The ubiquity of IoT devices in both public and private networks has increased dramatically in the last few years with billions of network connected devices appearing in every sector. In many cases these devices provide low-power and low-cost solutions to multiple problems, but this convenience comes with a price. Low-powered devices often lack the computational capacity to support encryption or other means of protection. In addition, devices are often optimized for easy connection *out-of-the-box*, potentially leaving them vulnerable due to default configurations. Accounting for IoT devices within a network can be problematic as spoofing of IP and MAC addresses is trivial, but device identification is fundamental to network security. Much work has been done in recent times to address the problem of identification by fingerprinting network traffic using various machine learning techniques, allowing network administrators to track device membership and detect anomalous behavior. While a high degree of accuracy has been achieved, effective feature extraction and acceptable computational overhead continue to be an issue. In addition, machine leaning models must be frequently modified and updated to remain effective. We apply a combination of locality sensitive hashing and neural networks to identify specific devices based on their network traffic, eliminating the need for complex feature engineering and model retraining. Our approach achieves an accuracy identifying known devices as high as 98% using only a single packet sniffed from the network, allowing for real-time device identification. In addition, our approach eliminates the need for feature extraction providing a significant improvement over other approaches.**

***Index Terms*—Internet of Things, IoT Device Identification, Network Traffic Fingerprinting, IoT Security**

I. Introduction

In recent times the concept of an Internet of Things (IoT) has been emerging with numerous devices such as cameras, low-powered sensors, wearable technologies, and a multitude of household and industrial devices inter-connected to one another and to the Internet. In today's world we are essentially surrounded by IoT devices which play increasingly significant roles in our lives. This is a trend which is likely to continue, with reports showing there are as many as 30 billion IoT devices connected worldwide [1] with continued rapid growth expected. Along with the convenience offered by these technologies, there are also increased security risks with many devices under-equipped to prevent compromise, leaving themselves and the networks to which they are connected vulnerable to attack.

978-1-6654-8303-2/22$31.00 ©2022 IEEE

The first step in providing security for these systems and the networks on which they reside is proper accounting and management, but keeping track of IoT devices in a network can prove to be a challenging task. Identifiers such as IP and MAC addresses are of little use as they can be easily spoofed, requiring the development of other approaches. The topic of IoT device identification by analyzing generated network traffic has been a popular area of research in the last few years with several proposed solutions in the literature. Various approaches using machine learning techniques have been explored, wherein network traffic is collected and analyzed either in real time or offline, and models are developed for machine learning classifiers which have been able to identify specific devices and device types as well as differentiating *normal* traffic from *anomalous* traffic. This has also been helpful for highlighting when new or unknown devices connect, or when familiar devices begin behaving in unexpected ways; an indication of compromise. Many of these methods are not without their weaknesses though, requiring specific traffic intervals be collected, that adequate quantities are collected, or that classifiers are able to operate within available computational constraints. Many require feature extraction and engineering which can introduce a high degree of overhead, or high degrees of required domain knowledge to choose the appropriate features for the given classifiers. In addition, many classifiers require models be updated and retrained frequently to continue providing good results [2] [3].

One of the more innovative ways of identifying IoT devices in a network based on their generated traffic is found in [4], wherein device fingerprints are generated using locality sensitive hashes. Locality sensitive hashing is useful in performing similarity searches, and has proven to be of fundamental importance in numerous fields, such as spam email, machine learning, data mining, and malware detection. In a similarity search, "given a collection of objects D with some similarity measure s defined between them and a query object q, retrieve all objects from D that are most similar to q according to the similarity measure s" [5]. Put more simply, a hash can be generated that is very similar for similar inputs, with minor changes causing only slight differences in the output. For IoT identification, a model can be built from several samples of traffic generated by the same device, then compared to unknown traffic to find the highest degree of similarity to

approximate a match. In the same sense, some threshold can be set to determine devices of similar type. This approach provides a means for identifying devices without the need for feature extraction or model retraining. Unfortunately, it has been shown that as the sample size decreases, say from a 10-minute sample to a 5-minute sample, the accuracy degrades considerably, requiring that a large enough sample be collected for reliable identification.

To alleviate this we combine Nilsimsa [6] with a simple neural network to identify connected devices by creating a fingerprint of their generated network traffic. To accomplish this we utilize traffic flows collected as *.pcap* files and convert them into Nilsimsa hashes. These hashes are then broken down into their constituent 32 bytes and are combined into a feature vector with each byte represented as a base-10 integer. In this way we combine the power of similarity hashes with a machine learning approach for device classification. We find precision, accuracy, and recall to be high with this method, avoid the complexities of feature extraction, and are able to maintain performance for very small traffic flow samples (single packets); a significant improvement over previous works.

The remainder of this paper is organized as follows: section II provides a review of related works, section III will describe our methodology, section IV will provide experimental results and an analysis of our method, and finally section V will present a conclusion and future work.

II. Related Work

The topic of IoT device identification based on an analysis of captured network traffic has been studied extensively in the past few years. In this section we cover the main approaches by category, and provide a review of recent works found in the literature.

A. Temporal Features

Noguchi et. al. [7] monitor the state of a device within a network based on the time change pattern of the number of features which can be extracted from signals originating from the device, automatically detecting changes in state and identifying new devices as they join a network. Mazhar and Shafiq [8] seek to characterize IoT traffic in terms of temporal patterns, volume, and target endpoints, and also consider security and privacy concerns. They note certain devices exhibit specific behaviors based on their particular functionality such as video streaming, and diurnal usage patterns, and also note device back-ends typically connect to a limited number of service providers making them more centralized than they appear. Aneja et. al. [9] perform device fingerprinting based on packet inter-arrival times. To accomplish this a Convolutional Neural Network (CNN) is employed to classify devices based on the generated graphs, achieving an accuracy of 86.7%.

B. Network Layer Features

Meidan et. al. [10] use supervised learning to train a multi-stage binary classifier. Their system works to separate IoT traffic from non-IoT traffic and attempts to determine to which *class* of devices it belongs. They achieve an accuracy of 99.28%, but do not identify specific devices.

C. Application Layer Features

Ullah and Mohmoud [1] focus on the set of application layer services a device uses, such as *ARP, SSL, LLC, EAPOL, HTTP, MDNS,* and *DNS* to develop a static view of device behavior.

D. Initial Connection Phase Fingerprinting

Miettinen et. al. develop IoT SENTINEL [11] with the goal of restricting communications in an IoT network to prevent an adversary from connecting to vulnerable devices. A series of fingerprints are constructed from devices while they conduct their initial connection phase, which are then mapped to device types. Features used for developing fingerprints are extracted from IP packet headers, and do not include payload information to avoid issues with encrypted traffic. The method claims an overall average accuracy of 81.5%, but traffic must be captured during the initial connection phase. Marchal et. al. [12] develop *AuDI* which autonomously fingerprints devices in any state of operation *after* the initial connection phase. Since the goal of *AuDI* is network management, device classes rather than individual devices are identified as *abstract device types*.

E. Flow-Based Feature Extraction

Salman et. al. [13] seek to manage security restraints and requirements for IoT by identifying connected devices through the extraction of statistical features from their generated network traffic. Their proposed framework extracts simple features per packet based on any network flow of 16 consecutive packets, allowing for device identification in real time. To classify devices the authors utilize decision tree, random forest, recurrent, residual, and convolutional neural networks, but require feature extraction. Silvanathan, Gharakheili, and Sivaraman [14] develop a set of one class clustering models to identify devices from a real-time flow level telemetry. The primary features used for model training are *average flow rate* and *average packet size*. Byte counts and Per-flow rates are captured each minute, and attributes are computed at varying time granularity increments of 1, 2, 4, and 8 minute intervals. At each one minute interval a vector of attributes is generated which corresponds to each device. K-Means algorithm is utilized to identify clusters, which are then used to identify device instances. Bao, Hamdaoui, and Wong [15] utilize a hybrid deep learning approach. The method combines deep neural networks with clustering to enable the classification of both previously seen and unseen devices, and uses an auto-encoder technique to reduce the dimensionality of the resulting data set. An unsupervised data clustering algorithm called OPTICS is employed which is based on space density. Auto-encoder is a symmetrical artificial neural network for reconstructing a given input. Known devices are identified using a random forest classifier based on an input vector with 297 features extracted from the network traffic. Hamad et. al. [16] address the problem of IoT device identification by assembling a series of packets from network traffic flows,

and extract various features to create a fingerprint for devices. Supervised machine learning algorithms are then applied on these features to solve a multi-class classification problem.

F. Image Analysis

In a unique approach, Kotak and Elovici [17] construct a white list of permitted devices by collecting TCP traffic which is processed to remove packet headers, and the resulting data is converted into a gray scale image with each pixel representing a hexadecimal value from the binary file. A single class classifier is applied to white listed devices, while a multi-class classifier is applied to non-white listed devices to identify anomalous traffic.

G. Single Packet Feature Extraction

Aksoy and Gunes [18] develop a framework which randomly selects single packets for analysis using a genetic algorithm (GA) to determine which features are relevant for classification. Similarly, Chowdhury et. al. [19] perform device identification through fingerprinting by extracting features from single TCP/IP packet headers. A feature vector is developed by generating a score for all available features based on variability, stability, and suitability of each bit. WEKA tool is then utilized to sample various ML algorithms for device classification.

H. Behavioral Fingerprinting

Bezawada et. al. [20] perform device behavior fingerprinting by extracting features from network traffic. They attempt to approximate the device type based on an analysis of a collection of the protocols used, and by observing command and response sequences elicited from a device via its smart phone app. The authors claim a device can be identified with as few as 5 packets from a single device. Gill, Lindskog, and Zavarshy [21] develop a baseline *normal* profile using traffic discovery and classification, then work to detect traffic anomalies based on the following features; overall traffic, transport layer protocol traffic, traffic by destination socket, packet size, and IP flow. Variations on statistical measurements of these five areas are used to formulate a profile that is considered to be *normal*, with variations that fall outside of these parameters identified as *anomalous*. Yousefnezhad, Malhi, and Framling [22] analyze packet header information to capture statistical information, and combine this with sensor measurements to form a feature set that is used in classifying IoT devices. Various machine learning algorithms are applied depending on the degree of linearity of the captured data.

I. Limited Feature Extraction/Engineering

Fan et. al. [23] work on IoT device identification using semi-supervised learning. They deal with the problem of feature extraction by using a smaller set of labeled data along with a much larger body of unlabeled data, with labels inferred from the smaller set. Features chosen include time intervals, traffic volume, protocol features, and transport layer security features. Transformed features are clustered per class to compensate for feature fluctuation. Convolutional Neural Networks are used for dimension reduction and multi-task learning is employed, helping the classifier to distinguish IoT devices from the rest of the traffic. Charyyev and Gunes [4] use locality sensitive hashes to determine similarity in network traffic flows, a useful technique in that feature extraction is not required. Hashes are generated and compared for similarity to known devices. While effective, they note performance degrades as traffic flow size decreases.

J. Textual Features and Data Mining

Ammar, Noirie, and Tixeuil [24] use a *Bag of Words* approach to fingerprint different IoT devices on a network. To accomplish this, textual information is collected and modeled as binary data using Bag of Words, and devices are "represented by a feature vector of m words, and is set to 1 if a word is present in a device description, and 0 if it is not". Each vectors is then added to a database to be used for constructing an p X m matrix where columns represent words and rows represent devices. These are then used as device labels. Feng, Wang, and Sun [25] seek to build annotations for device types, vendors, and product names by leveraging application layer response data from IoT devices and relevant websites to obtain product descriptions. These annotations are used to build a rule set for devices within a network to control and restrict access and communication to and from devices. Ammar, Noirie, and Tixeuil [26] utilize a supervised learning classifier to differentiate devices based on features from network flow, as well as textual features from packet payloads. Features that are captured include maximum and minimum packet length, average packet inter-arrival time, number of packets in a given flow, protocols used in the flow, device manufacturer name, model, friendly name, XML description, mDNS information, device OS, and device model. Textual features are modeled as a bag-of-words, with each device description making up a feature vector.

K. Feature Cost Metrics

Finally, Desai et. al. [27] focus on the cost of choosing and selecting appropriate features for applying machine learning algorithms to network traffic for device identification. Their framework selects features based on their utility for meeting the needs of specific algorithms. Once features are selected, popular ML algorithms are applied. A statistical method is also utilized to screen features for their relative value.

While effective, most of these approaches require feature extraction, perform only binary classification, or require a sufficiently large traffic flow to achieve good performance. Our contribution eliminates the need for feature engineering, consistently achieves an accuracy of 98%, and can identify a known device using only a single packet extracted from the traffic flow.

III. Methodology

We utilize the principle of *Locality Sensitive Hashes* to generate a feature vector which will be used by a classifier in fingerprinting IoT devices based on their generated network

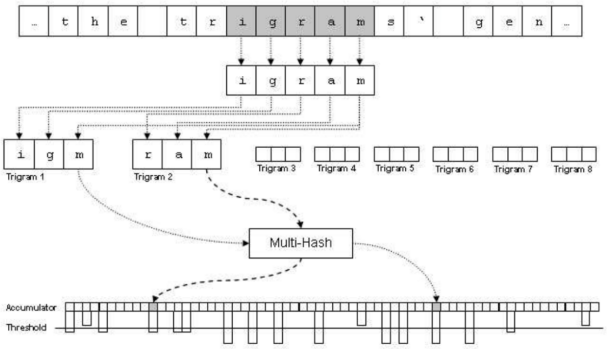


Fig. 1: Construction of a Nilsimsa hash [6]

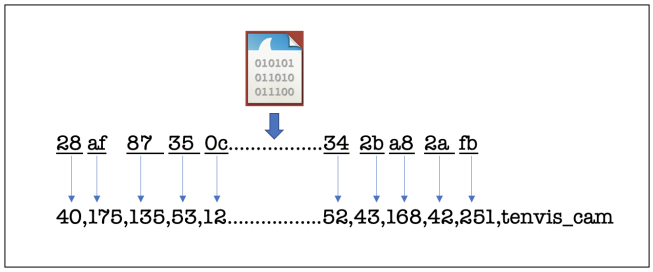


Fig. 2: .pcap files are converted of to Nilsimsa hash digests and then to integer strings for processing by a classifier.

traffic. In this section we will describe this technique and provide a details on both the Nilsimsa hash and how it is used in combination with a neural network.

Unlike cryptographic hashes which produce an entirely different hash even if a very small change is made to the input data, locality sensitive hashing (LSH) is a hashing algorithm wherein small changes to the input data produce a very similar output. This is accomplished by placing similar input items into common buckets with a high probability, with hash collisions being maximized rather than minimized. LSH has been effectively utilized for tasks such as electronic identification [28], anomaly detection [29], malware classification [30], and spam email detection [31] [32]. There are several tools available for generating locality sensitive hashes, such as *Nilsimsa*, *ssdeep*, *sdhash* and *tlsh* [33]. For our work we utilize Nilsimsa as it provides a digest which displays little variance for similar inputs, and can easily be broken into a fixed number of octets representing the input source.

As seen in figure 1 Nilsimsa uses a fixed-size sliding window of 5 bytes which analyzes input data on a per-byte basis, producing a series of trigrams of possible combinations of the input characters. These trigrams are mapped to a 256 bit array, or accumulator, to ultimately create a hash of the data. Each time a *bucket* in the array is accessed its value is incremented, and at the end of the process if the cumulative value of the bucket exceeds a given threshold its position in the array is reset to 1, otherwise it is set to 0 producing a 32 byte digest. Typically these digests are measured against one another for similarity by comparing each bit and producing a score of between -128 and 128 by checking the number of bits that are identical in the same positions and either adding or subtracting them, with the highest score indicating the greatest similarity. In our approach we do not utilize this scoring system, but instead use the resulting 32 bytes of the hash to produce a set of *features* for use with a more complex machine learning algorithm.

While locality sensitive hashes have been shown to be useful on their own in classifying network traffic, they have proven too simple to offer good performance with smaller data flows and are at a disadvantage to more complex approaches such as neural networks. Given a complex input feature space, a reasonable number of samples, and a network architecture of sufficient size a neural network can do the heavy lifting necessary for discovering correlations between various input variables which are necessary for separating the data into the labeled classes. However, if the input feature space is too large it will exceed even a neural network's ability to recognize patterns. For this reason it is difficult to directly apply neural networks to raw .pcap (network traffic flow) files without extensive feature extraction and engineering. To simplify the process our approach utilizes Nilsimsa to reduce the dimensionality of the raw data and form features that a neural network can process. By breaking Nilsimsa's 32 byte digest into individual bytes and converting these into integer values readable by the classifier, an entire traffic flow can be interpreted as a suitable feature vector.

Simplicity being the strength of this approach, we elect to use a *Multi-Layer Perceptron* (MLP) [34], an example of a simple neural network. The MLP is made up of four primary components: an input layer, hidden layers, non-linear activation function, and an output layer. In the context of this work the input layer is made up of 32 integers with values between 0 and 255. These values correspond to the 32 bytes which are output from the Nilsimsa hash. Next, we utilize an extremely concise single hidden layer containing 100 hidden units. The outputs of each unit in the network are passed through the rectified linear unit (ReLU) activation function so that the model can learn a non-linear decision function. Lastly, the network outputs a 22 length vector where each value in the output vector represents a possible class. The *Softmax* function is applied to the output vector which translates output values to probabilities. The class with the highest probability is identified as the predicted class. The network weights are learned in a single epoch with the *Adam* optimizer and a constant learning rate of 0.001.

IV. Results and Analysis

We test our method using a data set collected by Charyyev et al. [33], publicly available at https://github.com/netlab-stevens/LSIF/. The set contains twenty 24-hour traffic samples collected from 22 different IoT devices. Each sample consists of a file captured using *tcpdump*, and are stored in .pcap form. For our experiment these larger flows are also converted into shorter time segments using *tshark*, yielding descending

samples ranging in time slices from 10-minute down to 1-minute, and finally a sample of the original 24 hour data set at a per-packet level, in other words, one .pcap file for each packet. Next, each of these .pcap files is processed into a 32-length feature vector using Nilsimsa as described above. A final field representing the name of the device that produced it (the *class*) is appended to the vector, as seen in fig. 2. These are categorized by time slice and are combined, producing a single file of strings for each time slice from all 22 devices for processing by the MLP.

In each of our experiments the model is trained in a balanced 5-fold cross validation scheme where the data set is split into 5 equal segments. Our MLP is randomly initialized and trained from scratch 5 times. In each iteration, 4 of the segments are used for training (80%) and the remaining segment (20%) is used for evaluating the model. Here, balanced means that each fold contains an equal, but random, representation of each class. This training regiment greatly reduces the likelihood that our results are the product of a beneficial random selection of data points or initial network weights.

Performance results for the per-packet time slice by device are shown in table I. The table displays device name, weighted accuracy score, weighted f1 score and sample weight. Here, sample weight is the percent contribution of each device to the total number of samples. Important findings in this table are as follows. First, the Smart Recon method exceeds our expectations with 13 out of 22 devices classified with over 98% accuracy. Second, the devices which perform poorly are those with less than 15,585, or one percent, of samples available. Third, the OceanRadio and Goumia Coffee Pot appear to be exceptions to the *limited data* problem. This suggests that the devices have significantly different transmission patterns than the others in the data set.

TABLE I: Model performance on cleaned per-packet data set

Device	Accur.	F1	Sample wt.
oossxx SmartPlug	1.00	1.00	0.071
Lumiman 600 Bulb	0.99	0.99	0.071
Lumiman 900 Bulb	0.99	0.99	0.071
Gosuna Bulb	0.99	0.99	0.070
Smart Lamp	0.99	0.99	0.070
OceanDigital Radio	0.99	0.99	0.039
Smart Light Strip	0.99	0.99	0.071
Gosuna Socket	0.99	0.99	0.071
Renpho SmartPlug	0.99	0.99	0.071
Lumiman SmartPlug	0.99	0.99	0.071
Goumia Coffee Pot	0.98	0.99	0.012
Wans Cam	0.98	0.99	0.222
D-Link 936L Cam	0.97	0.98	0.015
Minger Light Strip	0.77	0.87	0.019
LaCrosse Alarm Clock	0.76	0.87	0.017
itTiot Cam	0.76	0.87	0.016
Tenvis Cam	0.61	0.76	0.008
tp-link SmartPlug	0.53	0.70	0.003
Chime Doorbell	0.34	0.51	0.007
Ring Doorbell	0.28	0.44	0.0007
Wemo SmartPlug	0.21	0.35	0.003
tp-link Bulb	0.09	0.16	0.001

In figure 3, accuracy and f1 scores for each time slice are visualized across all time slices. Upon inspection it appears that these performance results show an *unusual trend*, with 10-minute flows performing very well, but with degrading results as flow size decreases to 1-minute. Finally, at the per-packet level performance spikes sharply upward to around 98%.

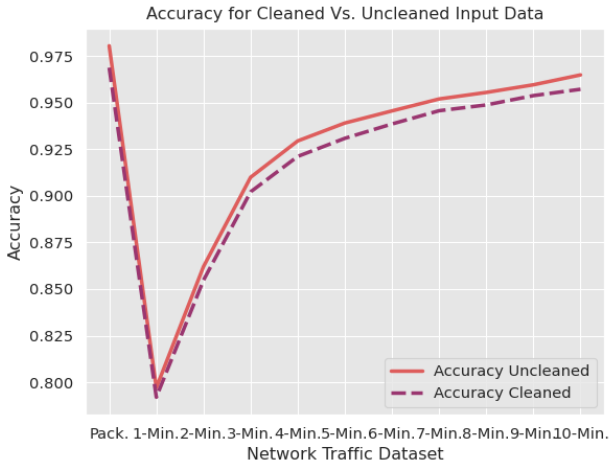
To eliminate any possible sample bias when analyzing the various traffic flows, all of the input data is replicated in its entirety with all IP and MAC addresses generalized as either ***1.1.1.1*** or ***11:11:11:11:11:11***. In this way it can be shown that the model is not learning to identify trends based on device-specific or flow-specific attributes. Average accuracy and F1 score for both uncleaned (non-anonymized) and cleaned (anonymized) data can be seen in figure 3. We see there is some bias present as the cleaned data set yields an accuracy of 96.9% and F1 score of 96.7%, slightly lower than the uncleaned with an accuracy of 98% and F1 score of 98%. Note this bias has a negligible effect on the *unusual trend*. As bias seems to play little role in this, we consider the possibility there could be some imbalance in the number of samples representing each device possibly contributing to this *unusual trend*.

To understand the *unusual trend* one needs to understand two important pieces of information: how these metrics are calculated, and how the percentage of samples differs between the minute/multi-minute time slices and the per-packet time slice.

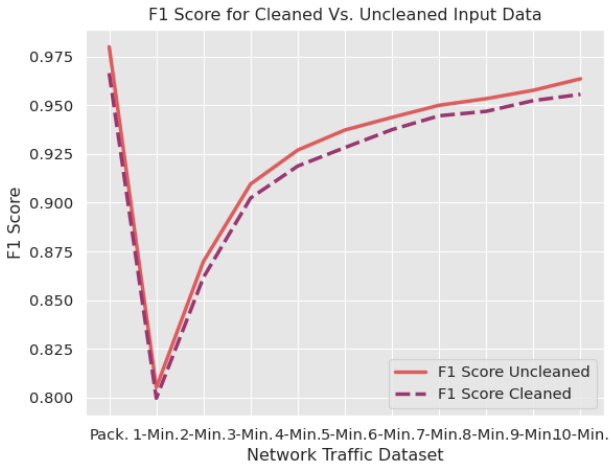
There are two steps in computing the metric scores. First, metrics are calculated for each label, and their averages are weighted by support (the number of instances for each label divided by the total number of samples across all devices). Second, the mean score is computed across devices. Simply put, the performance of devices with a relatively high number of samples will strongly contribute to the final score of a given time slice. Weighted averaging is effective for computing metrics because it accounts for label imbalance, giving a more accurate value for assessing the actual performance of the method. This is shown mathematically below, where D is the number of devices, S_D is the number of samples for the current device, M_D is the metric score for the current device and N is the total number of samples in the current time slice.

$$\frac{\Sigma_0^D(M_D * \frac{S_D}{N})}{D}$$

In regard to the differences between time slices, the nature of the Nilsimsa hash and how it is applied to the data set causes all flows to have the same percentage representation for each device, with an increasing number of samples as the time slices become shorter. It should be noted that while each time segment may contain varying numbers of packets, it will still produce a single hash. If, therefore, a time segment contains very little data, it will still produce a hash which when measured in an un-weighted manner can produce deceptive performance results if the device appears to be performing poorly. This is seen in the decreasing performance as the time



(a) Accuracy



(b) F1 Score

Fig. 3: Accuracy and F1 Score for all time slices (10-minute through 1-minute and per-packet) in the cleaned and uncleaned data. These metrics were computed using a balanced 5-Fold Cross Validation technique. Here, balanced means that each device has equal representation in each fold.

slices shrink from 10-minute to 1-minute and partially accounts for the *unusual trend*.

While there is some variance in the amount of traffic generated by the different devices, we find that with one exception, all devices contribute about 4% of the samples in each data set between 10-minute and 1-minute. We see that the OceanRadio device contributes only 0.05% of the data, yet still performs very well despite the lack of traffic available. This indicates that imbalance in samples size is not solely responsible for the *unusual trend*.

We therefore consider the possibility that some devices may be performing more poorly overall than others which might affect the *unusual trend* by bringing down the average performance. To explore this we analyze per device statistics

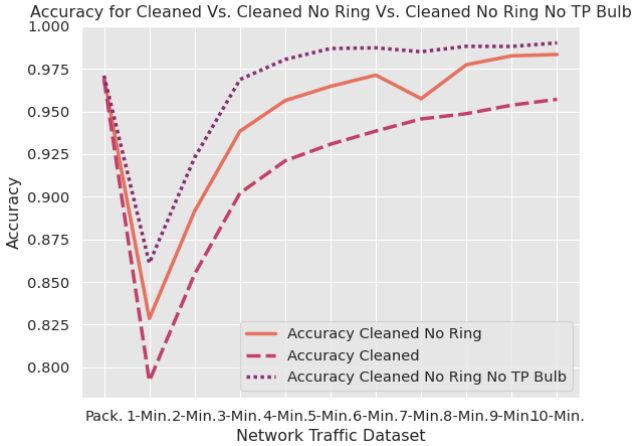


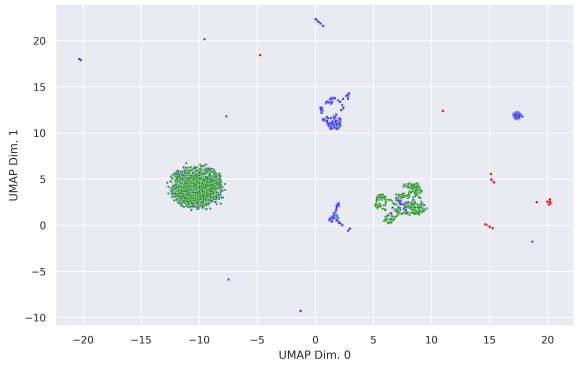
Fig. 4: Model performance excluding results from Ring Doorbell and tp-link Bulb (devices with poorest results)

across all time slices, and discover the Ring Doorbell and the tp-link Bulb are often misidentified. In many cases our model wrongly predicts these two devices to be other devices, largely because they contribute considerably less data overall. To see what effect this is having on the observe *unusual trend* we re-run the model without these devices as seen in figure 4.

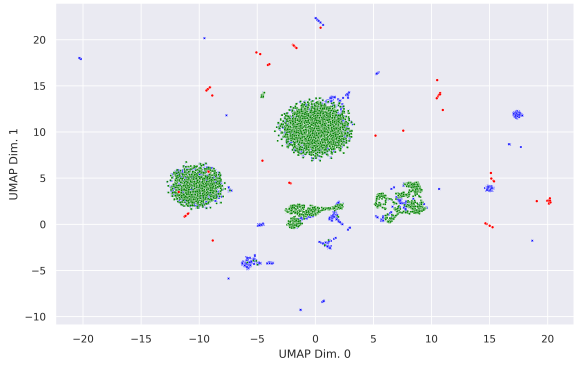
We see that when these two devices are removed, overall accuracy and F1 score improve for the time slices between 10-minutes and 1-minute, indicating that their poor performance is bringing down the overall average performance. However the removal has no real effect on the per-packet performance. Because the percentage representation for each device in the 10-minute to 1-minute time slices is equal these poorly performing devices skew the overall average. When we reach the per-packet flows, performance depends on how many packets were sent in the 24-hour time period, giving a more accurate *weighted* view of the average performance. Since these devices transmitted significantly fewer packets, there is less effect on the overall average performance.

In order to better understand this behavior, we generate visualizations to represent the 32-dimensional data in two dimensions using Uniform Manifold Approximation and Projection (UMAP) [35], a tool for dimensional reduction and visualization. UMAP reduces the dimensionality of data while maintaining spacial relationships, allowing a two-dimensional view of clustering. Visualizations of the data are generated at the 10-minute, 5-minute, and 1-minute flows to demonstrate the effect of poorly performing devices on the overall average. Two devices, Ring Doorbell and tp-link Bulb perform poorly, while OceanRadio performs quite well. This is despite the fact that all three contribute relatively small amounts of data as seen in figure 5.

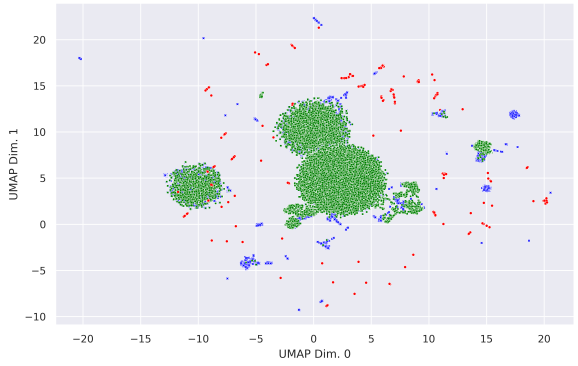
We see the clusters for the poorly performing devices degrading as the time slices become shorter, explaining the downward trend in average performance (the blue dots are almost completely obscured by the green dots in figure 5).



(a) 10-minute flow clustering



(b) 5-minute flow clustering



(c) 1-minute flow clustering

Fig. 5: UMAP clustering visualization of 10-minute, 5-minute, and 1-minute traffic flows for Ring Doorbell, tp-link Bulb and Ocean Radio.The red class represents the Ocean Radio device (best performing device for all minute length traffic flows). The blue and green classes respectively represent the ring doorbell and TP Link Light Bulb. Note how Ring Doorbell and tp-link Bulb increasingly overlap as time slices become shorter, while OceanRadio clustering remains distinct.

As the sample sizes are evenly distributed, these devices (along with the others that perform poorly) drag down the overall average. Per-packet performance is less affected as these devices also contribute less overall traffic as mentioned above.

We note here an apparent weakness in the data set as some devices are contributing small amounts of data for analysis. In addition, some devices (Ring Doorbell and tp-link Bulb) are functioning in a way that cause them to appear similar to all other devices and to one another which is certainly cause for further investigation into their actual network activity. To gain more accurate results future data collection would need to consider why some devices are contributing less traffic and some technique for mitigating this must be applied. It does however serve to validate the results found at the per-packet level, which is consistently performing at a very high level allowing single-packet identification of devices.

V. Conclusion

The Smart Recon method is able to identify known IoT devices with an accuracy of 98% using only a single packet sniffed from a network traffic flow. The combination of locality sensitive hashing to create feature vectors from .pcap files and a simple neural network allows for the training of a classifier that is able to produce a high degree of accuracy with a very small input sample. We believe this is due in part to the way that Nilsimsa hashes represent an averaging of input data, which then work in concert with the MLP to learn and identify trends. While larger time slices perform better than shorter ones, all are eclipsed by per-packet level data. Many IoT devices are relatively limited in their functionality producing repetitive network traffic. The hashes produced by larger input flows contain more samples from devices which have very few samples in the overall data set. We theorize that as flows become shorter they also become more sparse (i.e. contain fewer samples from devices with limited data) which in turn creates hashes which are less distinguishable for the MLP. At the per-packet level there are many similar hashes because of the repetition in network activity producing a data set which better represents each device, allowing it to learn and distinguish individual packets as belonging to specific devices. The classifier performs poorly on devices that contribute less data, but these in turn affect the performance of the per-packet flows less due to weighting explaining the *unusual trend* observed. In future work we would like to expand on effective and consistent data set collection such that an equal and large number of samples are available for each device. In addition, one hot device detection, analysis of diverse machine learning methods and other locality sensitive hashes would significantly improve the usefulness of the Smart Recon framework.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under award # 2121559. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] I. Ullah and Q. H. Mahmoud, "Network traffic flow based machine learning technique for iot device identification," in *2021 IEEE International Systems Conference (SysCon)*. IEEE, 2021, pp. 1–8.

[2] R. Kolcun, D. A. Popescu, V. Safronov, P. Yadav, A. M. Mandalari, Y. Xie, R. Mortier, and H. Haddadi, "The case for retraining of ml models for iot device identification at the edge," *arXiv preprint arXiv:2011.08605*, 2020.

[3] R. Kolcun, D. A. Popescu, V. Safronov, P. Yadav, A. M. Mandalari, R. Mortier, and H. Haddadi, "Revisiting iot device identification," *arXiv preprint arXiv:2107.07818*, 2021.

[4] B. Charyyev and M. H. Gunes, "Iot traffic flow identification using locality sensitive hashes," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[5] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," *arXiv preprint arXiv:1110.1328*, 2011.

[6] Bill. Nilsimsa similarity hash. [Online]. Available: https://asecuritysite.com/encryption/nil

[7] H. Noguchi, T. Demizu, N. Hoshikawa, M. Kataoka, and Y. Yamato, "Autonomous device identification architecture for internet of things," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. IEEE, 2018, pp. 407–411.

[8] M. H. Mazhar and Z. Shafiq, "Characterizing smart home iot traffic in the wild," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 203–215.

[9] S. Aneja, N. Aneja, and M. S. Islam, "Iot device fingerprint using deep learning," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*. IEEE, 2018, pp. 174–179.

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

[12] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "Audi: Toward autonomous iot device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[13] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "A machine learning based framework for iot device identification and abnormal traffic detection," *Transactions on Emerging Telecommunications Technologies*, p. e3743, 2019.

[14] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Inferring iot device types from network behavior using unsupervised clustering," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 2019, pp. 230–233.

[15] J. Bao, B. Hamdaoui, and W.-K. Wong, "Iot device type identification using hybrid deep learning approach for increased iot security," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 565–570.

[16] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal, "Iot device identification via network-flow based fingerprinting and learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 103–111.

[17] J. Kotak and Y. Elovici, "Iot device identification using deep learning," in *Computational Intelligence in Security for Information Systems Conference*. Springer, 2019, pp. 76–86.

[18] A. Aksoy and M. H. Gunes, "Automated iot device identification using network traffic," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[19] R. R. Chowdhury, S. Aneja, N. Aneja, and E. Abas, "Network traffic analysis based iot device identification," in *Proceedings of the 2020 the 4th International Conference on Big Data and Internet of Things*, 2020, pp. 79–89.

[20] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," *arXiv preprint arXiv:1804.03852*, 2018.

[21] M. S. Gill, D. Lindskog, and P. Zavarsky, "Profiling network traffic behavior for the purpose of anomaly-based intrusion detection," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 885–890.

[22] N. Yousefnezhad, A. Malhi, and K. Främling, "Automated iot device identification based on full packet information using real-time network traffic," *Sensors*, vol. 21, no. 8, p. 2660, 2021.

[23] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, and J. Yang, "An iot device identification method based on semi-supervised learning," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–7.

[24] N. Ammar, L. Noirie, and S. Tixeuil, "Network-protocol-based iot device identification," in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2019, pp. 204–209.

[25] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 327–341.

[26] N. Ammar, L. Noirie, and S. Tixeuil, "Autonomous identification of iot device types based on a supervised classification," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[27] B. A. Desai, D. M. Divakaran, I. Nevat, G. W. Peter, and M. Gurusamy, "A feature-ranking framework for iot device classification," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2019, pp. 64–71.

[28] W. Priesnitz Filho, C. Ribeiro, and T. Zefferer, "An ontology-based interoperability solution for electronic-identity systems," in *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 2016, pp. 17–24.

[29] M. Pirker, P. Kochberger, and S. Schwandter, "Behavioural comparison of systems for anomaly detection," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[30] H. M. Kim, H. M. Song, J. W. Seo, and H. K. Kim, "Andro-simnet: Android malware family classification using social network analysis," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–8.

[31] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "An open digest-based technique for spam detection." *ISCA PDCS*, vol. 2004, pp. 559–564, 2004.

[32] M. N. Marsono, "Packet-level open-digest fingerprinting for spam detection on middleboxes," *International Journal of Network Management*, vol. 22, no. 1, pp. 12–26, 2012.

[33] B. Charyyev and M. H. Gunes, "Locality-sensitive iot network traffic fingerprinting for device identification," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1272–1281, 2020.

[34] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," *American Journal of Psychology*, vol. 76, p. 705, 1963.

[35] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using umap," *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.