# A feature-ranking framework for IoT device classification

# A feature-ranking framework for IoT device classification

Bharat Atul Desai
Singapore University of Technology and Design
Singapore
bharat_atul@mymail.sutd.edu.sg

Dinil Mon Divakaran
Cyber Security R&D, Singtel
Singapore
dinil.divakaran@singtel.com

Ido Nevat
TUMCREATE
Singapore
ido.nevat@tum-create.edu.sg

Gareth W. Peters
Department of Actuarial Mathematics and Statistics
Heriot-Watt University, Edinburgh, UK
garethpeters78@gmail.com

Mohan Gurusamy
Electrical and Computer Engineering Department
National University of Singapore, Singapore
elegm@nus.edu.sg

***Abstract*—IoT market is rapidly changing the cyber threat landscape. The challenges to security and privacy arise not only because IoT devices are large in number, but also because IoT devices are heterogeneous in type and functionality. Machine learning algorithms are attractive methods to solve various problems such as device identification, anomaly detection, and attack detection. Often, all available features extracted from network traffic are fed as input to train the models, which in practice is not regarded as the best approach. Associated with features are different kinds of cost, such as costs for obtaining the data, extracting and storing features, compute resources to run a model with high dimensional features, etc. Instead, if a smaller set of features could achieve performance close to that obtained with all features, that might help to reduce cost as well as to make better interpretation of results. In this work, we address the problem of selecting features extracted from IoT network traffic, based on the utility of a feature in achieving the goal of the machine learning models. We develop a unifying framework of fundamental statistical tests for ranking features. We specifically consider the use case of IoT device classification, and demonstrate the effectiveness of our framework by evaluating it using different classifiers on traffic obtained from real IoT devices.**

***Index Terms*—IoT, classification, feature selection**

## I. Introduction

With billions of IoT devices expected to be connected to the Internet in a few years [1], IoT devices are soon to become a mainstay in our day-to-day life. Consequently, consumer data has become more exposed, as behavioral and living patterns get leaked into vulnerable systems and networks [2]. The threat landscape is also rapidly evolving due to the proliferation of IoT devices. Unlike conventional IT environments, IoT devices are large in number, often left unattended, and unlikely to be patched in a timely manner. This has increased the attacks on users, common Internet services, as well as critical infrastructure, at unprecedented scale, consequently increasing the service downtime and losses [3]–[6]. Therefore, intelligent security solutions need to be developed for proactively mitigate these risks and threats.

While security-by-design is necessary during the development of IoT devices, experiences with real-world systems have demonstrated that device-centric design approaches have not been sufficient to protect devices and prevent threats or attacks. Vulnerabilities are commonly found in many well-known systems and devices [7]. Hence, multiple approaches have to be taken to secure IoT devices. In a network-centric approach, network traffic is continuously monitored, processed and analyzed, for purposes such as identification of devices, detection of threats and security breaches, attack mitigation, etc. In the past, researchers have used statistical models and machine learning techniques to detect anomalies [8], malicious communications [9], and attacks [10] in networks. Recent research works have also applied machine learning to analyze network traffic of IoT devices, e.g., for classifying IoT devices [11]–[13].

Existing works that apply machine learning on network traffic extract specific characteristics of network flows, called features, from traffic data. A set of such features is called a feature vector. The performance of the machine learning model depends, among other factors, on the feature vector used for training. However, such works usually extract as many features as possible from network traffic to form the feature vector used for building the model, often ignoring a number of practical implications. Firstly, obtaining features incurs cost. Secondly, some features might simply be unavailable due to privacy concerns. A high-dimensional feature vector also increases the computational runtime of the algorithms; and lastly, it may also affect the performance of the machine learning models (see [14], and references therein). Below, we elaborate on the different types of costs as well as privacy issues in the context of network traffic data.

To extract features, there should be a process in place to monitor, store and and process network traffic; and not all features cost the same when it comes to these tasks. For example, the number of packets going through a network interface is one of the simplest and least expensive features that can be extracted from a router. On the other hand, to obtain the number of packets belonging to each connection, the router has to maintain per-connection state information which requires memory that scales up with the number of active connections at a router. Similarly, features can be extracted from IP headers or packet payloads. For example, the ports accessed by a set of traffic connections can be extracted from the IP header of packets. However, extracting the number of DNS queries for a particular domain name (which is useful in identifying services/functions of IoT devices) requires processing of contents of DNS packets. Extracting contents from packets increases storage cost as well as the cost of minimizing the risk of breaching privacy. Besides, as the complexity of a feature increases, more computational and memory resources are required [15], thereby increasing the cost of obtaining the feature. In other scenarios, some features might simply be unavailable if the appliance in operation does not support processing of the corresponding data. For example, extracting IP header information of each transiting packet is considered impractical in core Internet routers. Even meta information of packets, such as NetFlow records [16], may be supported only with sampling, as unsampled NetFlow requires upgrade of not only existing routers, but also the infrastructure to support storage and processing of large volumes of streaming data. Hence, obtaining such information results in costly upgrades. One could also purchase datasets or use a paid subscription service to enrich the data obtained from network traffic (e.g., with geolocation of IP addresses [17]) or to label the obtained data using a crowdsourcing marketplace (e.g., Amazon Mechanical Turk [18]).

Network data contains confidential information of users, and therefore also raises privacy concerns. While it is obvious that packet payloads contain confidential data, sensitive information of users can even be extracted from IP headers. For example, processing of destination IP addresses of users reveals their browsing behavior. Therefore, depending on the privacy policies, some data might not be available, while other features might be available but at a cost. For example, there may exist a policy that mandates anonymization of client IP addresses to minimize the possibility of learning behaviors of particular clients. Anonymization, in a way that is useful for analysis, say, using prefix preserving anonymization [19], adds to the cost of feature extraction.

Due to the above reasons, it is important to be able to select features that are not only useful in achieving the problem objectives but also in meeting the constraints due to cost and privacy. In this work, we address the problem of feature selection in the context of IoT network traffic analysis. Specifically, we consider the use case of IoT device classification for which recent works have applied machine learning techniques. We develop a framework of fundamental statistical tests on (the population parameters of) various features of interest, and combine the scores from the different tests to rank features on their ability to differentiate IoT devices. We evaluate our framework by applying well-known machine learning algorithms for device classification using network traffic data collected from a set of real IoT devices. In the experiments, features are selected based on the ranking produced by our framework. The experiments demonstrate that

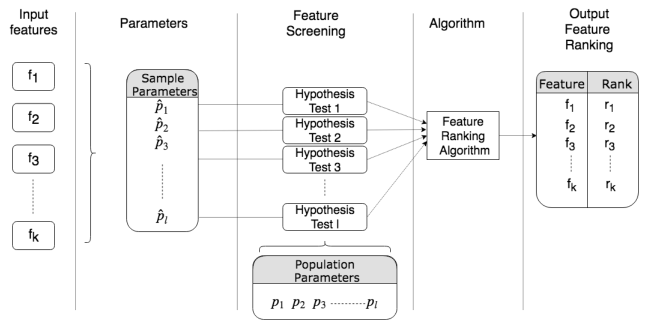

**Fig. 1:** Feature ranking framework

the performance of the classifiers are highly correlated with the ranking of features by our framework—a small set of highly ranked features is sufficient to achieve accuracy close to the empirical upper bound of accuracy obtained using all features.

Fig. 1 gives an overview of our framework. Features $f_1, f_2, \ldots, f_k$, denote the features, whose parameters are extracted from network traffic of IoT devices. A preliminary analysis of one such feature is presented in Section III, to motivate the need for measuring a feature's ability to differentiate the given set of devices. A feature has a certain distribution across a device; and a distribution can be represented by, say $l$ 'population parameters' $p_1, p_2, \ldots, p_l$ ($\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_l$, are the estimates from traffic data). Corresponding to each population parameter, we propose a well-known statistical hypothesis test; these tests that measure the utility of features are described in Section IV. The output of the tests are used to compute a score for ranking features, using the 'Feature Ranking Algorithm' described in Section V. We evaluate the proposed framework with data from real IoT devices, in Section VI.

## II. Related Work

Device identification, classification, and profiling, are a set of related problems in the area of IoT security. Identifying devices helps in timely patching of vulnerable devices, mitigation of potential large-scale exploits, asset management, etc. Profiles of devices are useful in developing solutions for detecting anomalies—behaviours that deviate from the normal. Recent works used machine learning for addressing this problem [11]–[13], [20]. In general, the approach is to extract features from network traffic that can help to identify specific and possibly unique characteristics of each device; for example, the number of packets in an interval for particular applications (say, HTTP), the length of activity period of a device, the number of DNS queries in an interval, etc. However, existing solutions have extracted all possible features from network traffic, without considering the costs of obtaining the features.

Typically, best practice in machine learning involves choosing the least amount of features to obtain the highest prediction performance. This allows the model to be efficient and interpretable. Incorporating irrelevant features increases model complexity, without a substantial increases in model performance. This is because the addition of irrelevant features causes models to find spurious relationships which affects model performance (e.g., refer [14]). Therefore, it is essential to determine which features add little value to prediction performance and remove them for the model training process.

In general, feature selection methods consist of wrapper methods, filter methods and model embedded methods. Wrapper methods use classifiers as a black-box to score feature subsets based of their predictive power [21]. For instance, sequential feature selection is a class of algorithms that add features to classification models in an iterative manner (e.g., greedy forward/backward feature selection [22]). A metric such as the Akaike information criterion (AIC) or Bayesian information criterion (BIC) is used to assess the model at each instance, and ultimately decide which feature subset results in the best model performance. Previous works such as [23] have utilized such a framework for network flow classification problems. However, wrapper methods are computationally too expensive as they have to search the entire feature space and continuously build classifier models for comparison.

Some machine learning models inherently rank features; these techniques when used for feature ranking are known as model embedded methods. For instance, recursive partitioning methods [24] such as classification or regression trees involve making branch splits based on feature importance that would ultimately lead to a leaf which represents some class label. These splits are based on a criterion such as the Gini index [25] in determining a feature's importance for a particular split. Ultimately, all class labels are granted leaves based on these splits, and a list of features are produced, which are ordered by their importance in determining the splits. The disadvantage of model embedded methods is that they place a restriction on the type of models available for the user to make use of for the classification objective. In this paper, we present a dynamic, feature selection framework that does not use any classification model, and hence is independent of model choice.

Lastly, filter methods make use of some discriminating criterion to determine if a feature is useful or not. Well-known filter methods include consistency based [26] and correlation based feature selection [27]. Consistency based feature selection ensures that the subset of selected features has similar distribution patterns for each class, whereas correlation based feature selection takes into account the utility of individual feature as well as the correlation between the features. While filter methods are both independent of model choice and computationally less intensive, they are often not holistic enough. If some classes cannot be distinguished using just that one filter method, then the feature is deemed to have less utility even if the classes could be discriminated using some other filter method. The framework we develop here incorporates multiple filter methods, and unifies their outputs in a way to obtain a single metric that is representative of a feature's utility in differentiating different classes (devices).

## III. Preliminary data analysis

Here we present an exploratory analysis of network traffic from IoT devices. For this purpose, we consider 15 IoT devices, of differing functionality and from different vendors, each numerically labeled; see Table I. We extract features by splitting traffic into 15-minute time-windows. While there are numerous features that can be extracted from network traffic of IoT devices (more than 100 features were identified in [13] for device classification), we consider only one feature for this preliminary analysis. In each time-window, and for each IoT device, we extract what we call the 'activity period' of a device. This is the time between the first and last packets of a particular device in a given time-window.(ie. Although the time-window is 15 minutes, the activity period of a device might be just 10 minutes or even 30 seconds.)

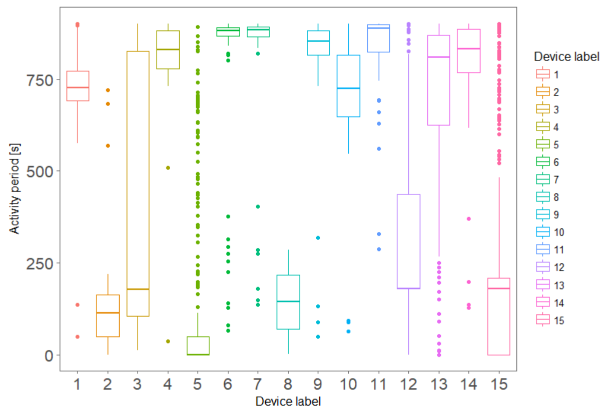

**Fig. 2:** Box plot of feature distribution across 15 IoT devices

The distribution of this feature for each device is represented as box plots in Fig. 2. It can be observed, while some devices (e.g., device no. 1 and device no. 2) can clearly be differentiated using this feature, for some others (e.g., device no. 9 and device no. 11) this feature does not have sufficient discriminatory power. In the case of device no. 5, we also note that there are a large number of outliers. Evidently, it is not easy to get a measure of a feature's discriminatory power by visually analyzing distributions of features. Indeed, from our experience of observing box plots of many features, we find the need to devise a systematic methodology that quantifies the ability of a feature to differentiate devices. If at least some devices can be differentiated by a feature, then that feature might contribute to the classification model built by machine learning algorithms.

## IV. Methods for measuring feature utility

In this section, we develop a framework that evaluates the utility of features in discriminating between classes (devices here) using multiple statistical tests. Subsequently, we combine the information on feature utility from each of the statistical tests, to rank the features according to their discriminatory power. This framework can be extended to the context of multiple-class classification in general.

Below, we briefly describe how statistical hypothesis testing can be used as a filter method for determining the discriminatory power of features. We then detail the proposed statistical tests that form the basis of our feature-ranking framework. The feature ranking algorithm is developed in Section V.

## A. Hypothesis Testing

In hypothesis testing, a claim is made regarding the value of a population parameter (e.g., mean, median, variance, etc.) that describes a population in some way, where the claim is referred to as the null hypothesis. The alternative hypothesis is the statement that proposes that the value of the parameter is different from what is being claimed in the null hypothesis. Hypothesis testing is thus a framework to formally accept or reject the claim that is being made in the null hypothesis.

Using the framework of hypothesis testing, we can make claims regarding various population parameters (defined below) of features of IoT devices. More specifically, for a given feature, if the population parameters of two devices are not equal (in statistical sense), then we can likely discriminate between the two devices using that feature.

## B. Feature Screening

Let $\mathbf{f} = (f_1, f_2, \ldots, f_k)$ denote a feature vector representing network traffic of IoT devices. For simplicity, we drop the index and use $f$ to denote a feature. Packet size is an example of a feature. Each feature $f$ has a distribution, and therefore can be represented by a number of *population parameters*. For example, packet size is a feature often used in traffic analysis; and the parameters that can be used to evaluate the goodness of this feature are the means, the standard deviations, etc. of the distributions over the different devices (say, mean of smart bulb against mean of smart camera). In this work, we consider four population parameters to evaluate a feature. They are: mean, median, variance, and distribution similarity. While this is not an exhaustive list, we limit our study to these widely used effective parameters. Henceforth, the set of population parameter is denoted as $\mathcal{P}$; and in this work $|\mathcal{P}| = 4$.

Below we describe feature screening techniques (or filter methods), which are essentially statistical tests, to evaluate all four population parameters of a given feature. However, each statistical test only compares distributions corresponding to two devices for the same feature. Coming back to our example of packet sizes, we take the distribution of packet sizes for devices $i$ and $j$, and evaluate this feature using the four tests corresponding to the the four population parameters. For example, the test on mean will tell us if the mean packet size of device $i$ and that of device $j$ are statistically equal.

*1) Student's $t$-test:* To compare the means of two distributions (say, of two devices), we use the Student's $t$-test [28]. The test assumes that the distribution of the sample means follow normal distribution. As such, the null hypothesis ($H_0$) and alternative hypothesis ($H_\alpha$) are:

$$H_0 : \mu_i^f = \mu_j^f,$$
$$H_\alpha : \mu_i^f \neq \mu_j^f;$$

where $\mu_i^f$ and $\mu_j^f$ are the means of feature $f$ over devices $i$ and $j$, respectively. We are thus comparing whether the mean of the distribution of feature $f$ across the device $i$ is the same as the mean of the distribution of $f$ over device $j$. Even if the mean of one device distribution is slightly greater than or smaller than the mean of the other device distribution, we say that the population parameters are statistically equal tolerated by a threshold for deviation. Whether the deviation is statistically significant or not is decided by computing a two-tailed hypothesis test with some predefined confidence interval. The two-tailed hypothesis test allows the parameters to deviate in both directions, while the threshold for tolerated deviation is determined by a confidence level. If the null hypothesis is rejected, we infer that the means corresponding to the two devices can be used to distinguish between them with certain level of confidence. The test statistic $T$, follows the $t$ distribution under the null:

$$T = \frac{(\bar{X}_i^f - \bar{X}_j^f)}{\sqrt{\frac{{s_i^f}^2}{c_i} + \frac{{s_j^f}^2}{c_j}}}$$

where $\bar{X}$, $s$, $c$ are the sample mean, variance and the count of the observed samples; the superscript denotes a feature $f$ in $\mathbf{f}$, and the subscript denotes the device. If the test statistic $T$ is greater than that of some pre-determined critical value at the specified confidence interval, the null hypothesis is rejected, and consequently the pair of devices can be differentiated using the mean of feature $f$.

*2) Brown-Forsythe Test:* The Brown-Forsythe test is used to test if the distributions of the samples from the $i^{\text{th}}$ and $j^{\text{th}}$ devices have the same variance [29]. Unlike Student's $t$-test which tests first order information, the Brown-Forsythe test makes a test on second order information. The null and alternate hypotheses are:

$$H_0 : {\sigma_i^f}^2 = {\sigma_j^f}^2,$$
$$H_\alpha : {\sigma_i^f}^2 \neq {\sigma_j^f}^2.$$

Let $x_i^f$ be the random variable for observations of feature $f$ from device $i$. (For readability, we drop superscript $f$ denoting feature.) The test statistic used under the null hypothesis is:

$$W = (c-2)\frac{\sum_{i=1}^{2} c_i(z_i - \bar{z})^2}{\sum_{i=1}^{2}\sum_{m=1}^{c_i}(\bar{z}_{i,m} - \bar{z}_i)^2}$$

where:

- $z_{i,m} = |x_{i,m} - \tilde{x}_i|$; where for device $i$, $x_{i,m}$ is the $i^{\text{th}}$ sample, and $\tilde{x}_i$ the median;
- $\bar{z}_i = \frac{1}{c_i}\sum_{m=1}^{c_i} z_{i,m}$;
- $\bar{z} = \frac{1}{c}\sum_{i=1}^{2}\sum_{m=1}^{c_i} z_{i,m}$;
- $c_i$ is the number of observations of device $i$; and
- $c$ is the total number of observations.

If the value of the test statistic $W$ is greater than that of the critical test value at the specified confidence level, then we can conclude that there is sufficient evidence for the null hypothesis to be rejected. And that means, we can discriminate between any two devices based on the variance parameter of the device distributions of the concerned feature.

*3) Mann Whitney U Test:* To make use of the median population parameter for comparison between device distributions, we utilize the Mann Whitney U Test. The test can generally be used to compare if the distributions of the $i^{\text{th}}$ and $j^{\text{th}}$ devices are equal. However, if the the distributions are identical, except for a location difference, the Mann Whitney U test compares distribution median difference values. But even if the distribution shapes differ, the test would correspond to comparing the stochastic ordering of the two device distributions.

$$H_0 : P(x_i^f > x_j^f) = \frac{1}{2}$$
$$H_\alpha : P(x_i^f > x_j^f) \neq \frac{1}{2}$$

Based on the null and alternative hypothesis, the test statistic for the Mann Whitney U test can be interpreted as the number of cross-sample pairs, where, for a feature $f$, an observation from the first device, is larger than that of an observation from the second device. The total count of pairwise comparisons that can be made is $c_i \times c_j$, where $c_i$ and $c_j$ represent the number of observations in the $i^{\text{th}}$ and $j^{\text{th}}$ devices, respectively.

If the distributions have the same median, then each observation from the first device has an equal probability of being greater than or smaller than each observation from the sample of the second device. The number of times an observation from the first device is greater than than an observation from the second is denoted by the test statistic $U_i$. The corresponding count for the observations from the second device being greater than that of the first device is denoted by $U_j$.

Under the null hypothesis, $U_i = U_j$. If the difference in medians of the device distributions is significant enough, the U statistic would be larger than the critical value at the specified confidence levels, thereby implying that the median parameter can be utilized to discriminate the pair of devices.

*4) Kolmogorov-Smirnov test:* In order to use similarity of distributions as a population parameter for comparison, we apply the Kolgomorov-Smirnov test [30]. It tests for the goodness of fit of one distribution to another, by comparing distance between the shapes of the device distributions. The null and alternative hypotheses are:

$$H_0 : \text{devices } i \text{ and } j \text{ have the same distribution}$$
$$H_\alpha : \text{devices } i \text{ and } j \text{ do not have the same distribution}$$

The test statistic used under the null hypothesis is:

$$D_{i,j}^f = \sup_x |\hat{\mathcal{F}}_i^f(x) - \hat{\mathcal{F}}_j^f(x)|;$$

where, for a given feature $f$, $\hat{\mathcal{F}}_i^f$ and $\hat{\mathcal{F}}_j^f$ are the empirical cumulative distribution functions (ECDFs) of devices $i$ and $j$ that are being compared. The test statistic can be understood as the difference in distances between the two considered distributions at various percentiles along the ECDFs of the devices. The greater the difference between the two distributions, the larger the test statistic would be. If this test statistic is greater than the critical value at the specified confidence level, then the null hypothesis is rejected.

## V. Feature Ranking Algorithm

In the previous section, we defined four filter methods corresponding to the four population parameters, using hypothesis testing. Each of these tests is used for comparing two device distributions of a particular feature. In other words, they are pairwise tests. In the following section, we propose a novel strategy that carries out this comparison for each pair of devices, and more importantly combines the results of these different hypothesis tests. The objective of this procedure is to obtain a score for each feature, which is indicative of its discriminatory power to differentiate between devices, and then rank features based on the score. This ranked list of features can be used for feature selection, where the minimum number of top-ranked features can be obtained for the respective classification models.

First, all the filter methods are used to compare between devices for a given feature. For each filter method, if there exists sufficient evidence to reject the corresponding null hypothesis at the predefined confidence level, we can conclude that the population parameter for that filter method can be used to distinguish between two devices. We therefore assign a boolean score for the pairwise test (the hypothesis test using distributions of two devices); if the null hypothesis is rejected, we assign a score of one, and if it is not rejected, we assign a score of zero.

This procedure is repeated across all pairs of devices for each of the filter methods. Therefore, for one feature and $n$ devices, there are $\binom{n}{2}$ pairwise hypothesis tests. This translates to a total of $\frac{(n)(n-1)}{2}$ number of elements (boolean values), which we represent using a matrix $\mathcal{S}$. The boolean element $\mathcal{S}_{f,p,i,j}$ is indicative of whether the particular pairwise hypothesis test on feature's $f$ parameter $p$, between devices $i$ and $j$, was rejected or not. The results for one filter method for a given feature comprise the lower triangular matrix, represented in Eq. (1). It is a lower triangular matrix because the hypothesis tests are performed for each unique pair of devices only once. Hence, the diagonal and the upper triangle of the matrix is denoted with a null element, $NA$.

$$\mathcal{S}_{f,p} = \begin{pmatrix} NA & NA & NA & \dots & NA & NA \\ \mathcal{S}_{f,p,2,1} & NA & NA & \dots & NA & NA \\ \mathcal{S}_{f,p,3,1} & \mathcal{S}_{f,p,3,2} & NA & \dots & NA & NA \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{S}_{f,p,i,1} & \mathcal{S}_{f,p,i,2} & \mathcal{S}_{f,p,i,3} & \dots & NA & NA \\ \mathcal{S}_{f,p,j,1} & \mathcal{S}_{f,p,j,2} & \mathcal{S}_{f,p,j,3} & \dots & \mathcal{S}_{f,p,j,i} & NA \end{pmatrix} \quad (1)$$

where:

$$\mathcal{S}_{f,p,i,j} = \begin{cases} 1, & \text{if } H_0 \text{ is rejected.} \\ 0, & \text{if } H_0 \text{ is not rejected.} \end{cases}$$

Once pairwise tests are conducted across all devices, $|\mathcal{P}|$ number of binary lower triangular matrices (Eq. (1)) are computed per feature. Note, there are $|\mathcal{P}|$ population parameters being compared for each feature. We now require a way to reduce these $|\mathcal{P}|$ matrices into $|\mathcal{P}|$ scalar *parameter scores*.

We define the *parameter score* of population parameter $p \in \mathcal{P}$ of a feature $f$, as the ratio of the total number of tests that failed the hypothesis to the total number of tests that were conducted. The parameter score thus represents the ability of a population parameter (w.r.t. a given feature) to discriminate between every pair of devices. Since there are $|\mathcal{P}|$ population parameters, each feature would have $|\mathcal{P}|$ parameter scores, $\psi_{f,p}$. To find $\psi_{f,p}$, we have to compute the $\mathcal{S}_{f,p}$ across all pairs of devices, and then normalize the sum:

$$\psi_{f,p} = \frac{\sum_{i,j;j>i}^{n} \mathcal{S}_{f,p,i,j}}{\left(\frac{(n)\cdot(n-1)}{2}\right)} \tag{2}$$

where: $0 \leq \psi_{f,p} \leq 1.$

We then obtain, for each feature, $|\mathcal{P}|$ number of parameter scores corresponding to all the population parameters:

$$\psi_{f,1}, \psi_{f,2}, \psi_{f,3}, \ldots, \psi_{f,|\mathcal{P}|}$$

Thus, we now have a well-defined way to determine which feature has the highest discriminatory power, on the basis of one of the $\mathcal{P}$ population parameters. Next, we would like to establish a method to determine which feature is best not just for one of the population parameters, but for all $|\mathcal{P}|$ parameters that we have considered. We are thus determining an overall score for each feature, that is indicative of the feature's ability to discriminate between devices in general. This score is defined as the *feature score*. The proposed methodology for this procedure is to simply take the linear combination of each parameter scores (Eq. (2)), to yield the overall feature score, $\psi_f$ (for feature $f$):

$$\psi_f = \sum_{p \in \mathcal{P}} \omega_p \cdot \psi_{f,p} \tag{3}$$

where: $0 \leq \omega_p, \psi_f \leq 1.$

Weights $\omega_p$'s are assigned to parameter scores $\psi_{f,p}$'s. These weights give an indication about the relative contribution from each parameter score towards the overall feature score. The value of these weights could be based on the computational budget afforded to calculate the respective parameters, or are perhaps based on the power of the parameter's hypothesis test. While finding the 'right' set of weights is an interesting problem, we consider the problem outside the scope of this work. Instead, over here, we assign equal weights to all population parameters; i.e., $w_1 = w_2 = \cdots = w_{|\mathcal{P}|} = \frac{1}{|\mathcal{P}|}$. In future, we will study the problem of determining the optimal weights. Listing 1 presents the steps to compute feature scores.

## VI. Performance Evaluation

In the following, we describe the experimental setup and the metrics for evaluation; subsequently, we present the results.

### A. Experimental procedure

For the experiments, we used 15 IoT devices listed in Table I. These devices were connected to the Internet via a gateway; and network traffic of the IoT devices was captured at this gateway. Traffic was split in 15-minute intervals, and the number of sessions captured per device is listed in the table. We extracted 111 features from the network traffic of IoT devices. Due to limitation of space, we refer readers to Section III of [13], where features are listed and explained. Of these 111 features, 94 are continuous numeric type features that could be used for device classification.

**Algorithm 1** computeFeatureScore($\mathcal{S}, \mathcal{W}$)

```
Input: Matrix $\mathcal{S}$; list of weights $\mathcal{W} = \omega_1, \omega_2, \ldots, \omega_{|\mathcal{P}|}$
1: $\psi_f \leftarrow 0$                                   ▷ Initialize feature score $\psi_f$
2: for $p \in \mathcal{P}$ do
3:     $\psi_{f,p} \leftarrow 0$                          ▷ Initialize parameter score $\psi_{f,p}$
4:     for $i, j \in \{1, 2, \ldots, n\}$ and $j > i$ do
5:         $\psi_{f,p} = \psi_{f,p} + \frac{\mathcal{S}_{f,p,i,j}}{\left(\frac{(n)\cdot(n-1)}{2}\right)}$      ▷ Normalise and add
6:     end for
7:     $\psi_f = \psi_f + \omega_p \cdot \psi_{f,p}$
8: end for
9: return $\psi_f$
```

**TABLE I:** Information on IoT devices used

| Label | Device | Brand | Sessions captured |
|---|---|---|---|
| 1 | Echo dot | Amazon | 490 |
| 2 | Smart remote | Broadlink | 480 |
| 3 | Camera (DCS930L) | D-Link | 384 |
| 4 | Camera (DCS5030L) | D-Link | 410 |
| 5 | Smart socket (DSPW215) | D-Link | 672 |
| 6 | Chromecast | Google | 297 |
| 7 | Home control | Google | 529 |
| 8 | Smart socket | Oittm | 394 |
| 9 | Hue light | Philips | 644 |
| 10 | Smart things | Samsung | 587 |
| 11 | Smart bulb (LB100) | TP-Link | 482 |
| 12 | Camera (NC250) | TP-Link | 587 |
| 13 | Camera (NC450) | TP-Link | 494 |
| 14 | Smart socket (HS100) | TP-Link | 452 |
| 15 | Smart socket (HS110) | TP-Link | 387 |

### B. Metrics for model evaluation

Let TP, TN, FP, and FN, denote the number of True Positives, True Negatives, False Positives, and False Negatives, respectively, of a classification test. Precision is the proportion of correction predictions, of all the instances predicted to be of a particular classs; i.e., $\frac{\text{TP}}{\text{TP+FP}}$. Recall is the proportion of correct predictions (of the true instances) of a class; i.e., $\frac{\text{TP}}{\text{TP+FN}}$. Given these, the metrics we used to evaluate the predictive performance of classifiers are:

- Accuracy: Of all predictions, proportion of predictions that are correct; i.e., $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
- F1 Score: The harmonic mean of the model's precision and recall, $2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision+Recall}}$.

### C. Results

The tests described in Section IV-B were conducted at a 95% confidence interval for $|\mathcal{P}| = 4$ sample parameters, on all $k = 94$ features. As a result, a total of 376 matrices ($\mathcal{S}_{f,p}$'s) are obtained. These were reduced to 376 parameter scores, represented by a $k \times |\mathcal{P}|$ matrix where each row corresponds to a unique feature, and columns correspond to the parameter

**TABLE II:** F1 scores for different classifiers

| **Classifier Type** | **Features** | | | | | | |
|---|---|---|---|---|---|---|---|
| | ***All*** | ***Top 5*** | ***Top 10*** | ***Middle 5*** | ***Middle 10*** | ***Bottom 5*** | ***Bottom 10*** |
| Naive Bayes | 0.917 | 0.757 | 0.731 | 0.390 | 0.333 | 0.083 | 0.130 |
| Classification Tree | 0.981 | 0.851 | 0.851 | 0.401 | 0.460 | 0.130 | 0.130 |
| Bagging | 0.956 | 0.926 | 0.933 | 0.427 | 0.451 | 0.130 | 0.130 |
| Random Forest | 0.924 | 0.863 | 0.865 | 0.434 | 0.460 | 0.086 | 0.130 |
| SVM (Linear Kernel) | 0.968 | 0.807 | 0.847 | 0.381 | 0.401 | 0.130 | 0.130 |

scores. These parameter scores were aggregated with equal weights into a final list of $k$ feature scores, $\psi_{\mathbf{f}}$. The list of feature scores were sorted in descending order, such that the highest ranked features would appear at the top of the table. In the following section, we investigate whether the top-ranked features actually result in greater classification performance.

The top, middle and bottom ranked features were fed into various off-the-shelf machine learning algorithms that are commonly utilized in practice. The following tree-based models were selected: Classification Trees, Random Forests, and Bootstrap Aggregation (Bagging). The Naive Bayes classifier was chosen from a family of probabilistic models based on Bayes Theorem and lastly, the Support Vector Machine (SVM) model with a linear kernel from kernel based classification approaches, were picked for the experiments.

Figures 3, 4 and 5 present the test accuracy for each classifier, obtained with different sets of features. As indicated, we selected sets of five and ten features from the top (Fig. 3), middle (Fig. 4), and bottom (Fig. 5), of the feature list sorted by our ranking framework. In all the three figures, we have also plotted the accuracy achieved when all 94 features are used. We also provide F1 scores in Table II, which reflect similar trends as the results on accuracy. We make the following important observations from these results.

The classifiers which were fed with the top-ranked features, consistently outperformed classifiers that were fed with middle-ranked features; which in turn consistently outperformed classifiers that were fed with the bottom-ranked features. This trend is an empirical validation of our feature-ranking framework—*better the ranking of the features used for the classifier, the higher is its predictive performance.*

Another observation comes from Fig. 3. Most classifiers achieve high accuracy (greater than 80%) with just five top-ranked features. Indeed, for the three best performing classifiers—Classification Trees, Random forests, and Bagging, the average (relative) drop in accuracy is only $\approx 6\%$. This means that, with only five features, classifiers can achieve close to the best possible accuracy obtained with all the features. In other words, the number of features can be significantly reduced without a loss in classification accuracy, thereby bringing down cost of feature extraction and model complexity.

When we compare the accuracy of models fed with top five features as opposed to those fed with the top ten (in Fig. 3), we observe that the marginal increase in accuracy is negligible. In fact, in the case of Naive Bayes, we see a decrease in accuracy when more features are added. Therefore, we decided to investigate whether there exists some critical point, such that addition of more features to the model after the critical point does not result in significant increase in test accuracy.

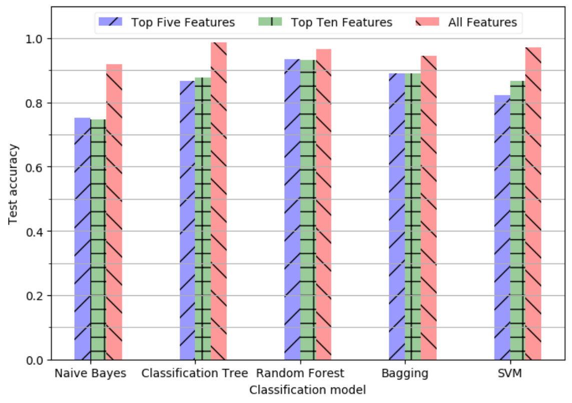

**Fig. 3:** Test accuracy with top-ranked features

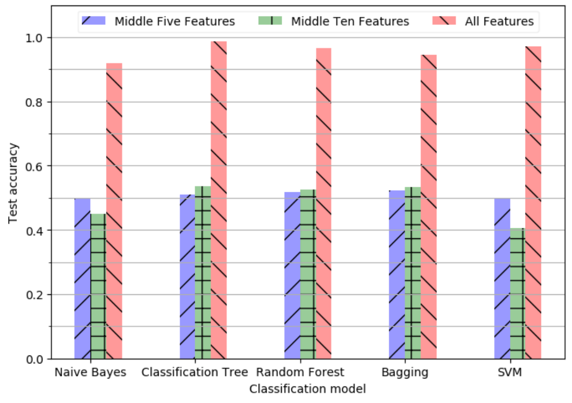

**Fig. 4:** Test accuracy with middle-ranked features

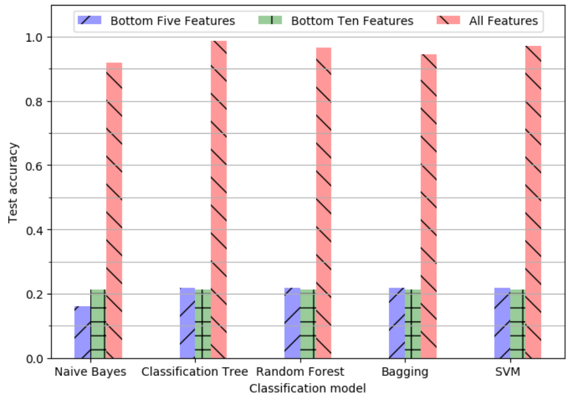

**Fig. 5:** Test accuracy with bottom-ranked features

This critical point, would thus represent, empirically, the optimal number of features to be chosen for the classification model's accuracy. In this experiment, we incrementally added features from our ranking table, starting with the highest ranked feature, followed by the next lower-rank feature, until all features are added to the model. The test accuracy for each classification model at every iteration is calculated and plotted (after smoothing) in Fig 6.

In Fig 6, we observe that the test accuracy of the models increases sharply at first before plateauing at some critical value. This is the point beyond which the utility of adding another feature has a negligible effect on the model's accuracy. Alternatively, users may decide to set a desired test accuracy

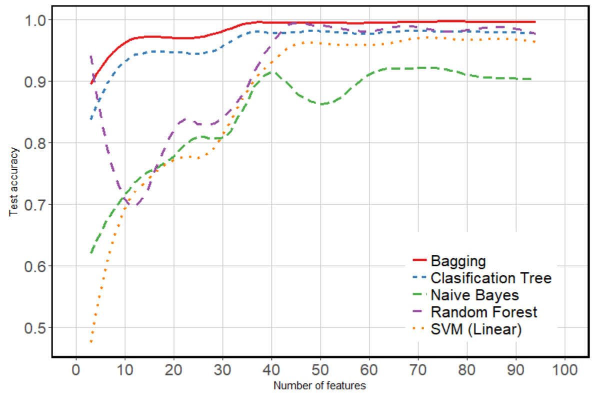

**Fig. 6:** Classifier accuracy with incremental addition of features, in decreasing order of rank (smoothed)

threshold, and proceed to choose the model that requires the least number of features. Bagging consistently performs better than the other models. With bagging, just eight features are required to obtain an accuracy of 95% and 32 for an accuracy of 99%. Though Random Forest performs well, its performance is initially unstable. Random Forest model randomly samples a selection of features at each decision split when building classification trees [31], and then aggregates classification results. As a result, the best features may not necessarily be selected (especially, when there are only few features), which introduces volatility into the performance of the model. In comparison, although Bagging aggregates the classification results, it does not sample the feature space at each decision split [32]; therefore, we see a stable trend.

## VII. Conclusions

This work addressed the problem of selecting features for device classification using machine learning algorithms. We developed a feature-ranking framework that employs fundamental statistical tests on a selected set of population parameters of features. Through experiments using traffic of real IoT devices, we demonstrated that the ranking produced by our framework helps in identifying the right set of features for device classification.

## Acknowledgement

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd.

## References

[1] Ericsson, "IoT connections outlook (Ericsson Mobility Report)," Jun. 2018, https://www.ericsson.com/en/mobility-report/reports/june-2018; accessed: Sep. 2018.

[2] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic," *CoRR*, vol. abs/1708.05044, 2017.

[3] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and Privacy Challenges in Industrial Internet of Things," in *Proc. of the 52nd Annual Design Automation Conf.*, ser. ACM DAC '15, 2015, pp. 54:1–54:6.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *Proc. 26th USENIX Security Symposium*, 2017, pp. 1093–1110.

[5] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[6] "120,000 IoT cameras vulnerable to new Persirai botnet say researchers," https://www.zdnet.com/article/120000-iot-cameras-vulnerable-to-new-persirai-botnet-say-researchers; accessed: Sep. 2018.

[7] "Common Vulnerabilities and Exposures," http://cve.mitre.org; accessed: Sep. 2018.

[8] I. Nevat, D. M. Divakaran, S. G. Nagarajan, P. Zhang, L. Su, L. L. Ko, and V. L. L. Thing, "Anomaly Detection and Attribution in Networks With Temporally Correlated Traffic," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 131–144, Feb 2018.

[9] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis," in *Proc. ACSAC*, 2012, pp. 129–138.

[10] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *Proc. INFOCOM*, vol. 3, June 2002, pp. 1530–1539.

[11] 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 *Proc. Symposium on Applied Computing (SAC)*, 2017, pp. 506–509.

[12] 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 *Proc. IEEE ICDCS*, 2017.

[13] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "DEFT: A Distributed IoT Fingerprinting Technique," *IEEE Internet of Things Journal*, 2018, note: accepted for publication.

[14] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data," *J. Mach. Learn. Res.*, vol. 11, pp. 2487–2531, Dec. 2010.

[15] D. M. Divakaran, K. L. Ling, S. Le, and V. Thing, "REX: Resilient and Efficient Data Structure for Tracking Network Flows," *Computer Networks*, vol. 118, pp. 37–53, 2017.

[16] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," Internet Requests for Comments, RFC Editor, STD 77, 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7011.txt

[17] "Geolocate IP Address Location using IP2Location," https://www.ip2location.com; accessed: Sep. 2018.

[18] "Amazon Mechanical Turk," https://www.mturk.com; accessed: Sep. 2018.

[19] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme," in *Proc. ICNP*, 2002, pp. 280–289.

[20] H. Haddadi, V. Christophides, R. Teixeira, K. Cho, S. Suzuki, and A. Perrig, "SIOTOME: An edge-ISP collaborative architecture for IoT security," in *Proc. IoTSec*, Apr. 2018.

[21] R. Kohavi and G. H. John, "Wrappers for Feature Subset Selection," *Artif. Intell.*, vol. 97, no. 1-2, pp. 273–324, Dec. 1997.

[22] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. of mach. learning research*, vol. 3, no. Mar, 2003.

[23] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM CCR*, vol. 36, no. 5, pp. 5–16, 2006.

[24] T. M. Therneau, E. J. Atkinson *et al.*, "An introduction to recursive partitioning using the rpart routines," 1997.

[25] L. E. Raileanu and K. Stoffel, "Theoretical Comparison Between the Gini Index and Information Gain Criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, May 2004.

[26] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial intelligence*, vol. 151, no. 1-2, pp. 155–176, 2003.

[27] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Dept. of Comp. Science, University of Waikato, 1999.

[28] W. Haynes, "Students t-test," in *Encyclopedia of Systems Biology*. Springer, 2013, pp. 2023–2025.

[29] M. B. Brown and A. B. Forsythe, "Robust tests for the equality of variances," *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 364–367, 1974.

[30] R. H. Lopes, "Kolmogorov-Smirnov test," in *International encyclopedia of statistical science*. Springer, 2011, pp. 718–720.

[31] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] ——, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.