Machine Learning Challenges for IoT Device Fingerprints Identification

Vian Adnan Ferman* and Mohammed Ali Tawfeeq†
1,2 Computer Engineering Dept., College of Engineering, Mustansiriyah University, Baghdad, Iraq

*Corresponding author, e-mail: egma018@uomustansiriyah.edu.iq, drmatawfeeq@uomustansiriyah.edu.iq

Abstract. The dramatic growth of Internet of Things (IoT) devices in recent years increases the IoT networks’ vulnerabilities and introduces new challenges among machine learning (ML) algorithms to detect the networked devices. The creation of a Device Fingerprint (DFP) may depend on extracting the network traffic features related to the device except for the identities assigned to it. In this paper, Device Fingerprints for 20 IoT devices are created by extracting 30 features during startup operation. Wireshark Network Protocol Analyzer is used to collect network traffic of 8 home IoT devices, meanwhile the traffics of the remaining devices are taken from the captures_IoT-Sentinel publicly available dataset. Four supervised machine learning algorithms were applied and tested to detect authorized devices and isolate unknown devices, namely: Support Vector Machine (SVM), Decision Tree (DT), Ensemble Random Forest (RF), and Gradient Boosting Classifier (GBC). Random Forest model and Gradient Boosting Classifier both showed better results of about 98.8% as an average of overall accuracy with less difference comparing with the accuracy of Decision Tree. Voting classifier was applied using the three estimators that resulted in high accuracy (DT, RF, and GBC) and achieving 99.5% as an average of overall accuracy.

Keywords: Gradient Boosting Classifier, IoT device fingerprint, network traffic, Random Forest, Voting classifier.

1. Introduction
Internet of Things (IoT) forms the basic conceptions of machine-to-machine connection and extends them outward by creating large cloud networks of devices that connect through cloud platforms. Therefore, IoT devices are defined as any device that has a dedicated address that gives the ability to connect to the network via Wi-Fi, Bluetooth, or other technology such as smart devices in the smart home, smart city, smart industry, smart transportation, medical environment, etc. [1]. Moreover, IoT devices have some other unique characteristics such as heterogeneity, inter-connectivity, ultra-reliable communication with low latency, low power and low-cost communications, dynamic network adaptations, and Intelligence [2, 3]. These features increased people’s tendency toward smart things in their whole lives and empowered markets to support a massive number and different types of IoT devices. Experts have predicted that IoT devices number will reach nearly 125 billion in 2030 [4]. With the widespread of these devices, the risks added to the IoT networks increase. Besides, due to IoT devices' connectivity between each other, the protection of one device also depends on the security...
of other devices and the cascading results of its vulnerabilities to the entire IoT system [5]. In 2016, the Mirai botnet launched a series of Distributed Denial of Service attacks with over 100k compromised IoT devices, therefore the first step of protecting IoT networks from these attacks is to find out network traffic details coming from real-world IoT devices and identifying them [6][7][8][9]. Therefore, for the former reason, knowing which devices are connected to IoT networks has become a trending topic for researchers nowadays.

Device fingerprint (DFP) is a process of creating either actively or passively unique signature for each device from network traffic data. The features that are used to create signatures should not be tampered with or modified with the device's mobility. Moreover, these features must be difficult to guess so, assigned addresses (e.g., International Mobile Equipment Identity, media access control and internet protocol) should not be taken into consideration [10, 11].

In [12] a DFP is created using device-originated network traffic from two public datasets. Machine learning (ML) classification algorithms are applied to categorize individual device types and achieved different accuracy for each dataset 83.35% and 97.7% as an average accuracies. While in [13] ML classification algorithms are used to classify devices’ events and interactions such as locking/unlocking and ON/OFF. Also in [14], a multiclass classification is applied to classify traffic generated during the user interaction with every IoT application (Application to cloud connection).

In [15] a Bag of Words technique is presented to identify 31 IoT devices among 33 of the experimented devices. The proposed method is based on extracting some devices’ textual information from IoT network traffic then creating a vector of unique textual features for each device. To detect a new connected device the similarity is checked between vectors. Whilst in [16] a network protocol packet keyword query is used to recognize IoT devices. Traffic packets are analyzed to extract network protocols’ data. Next, the result is filtered and irrelevant invisible information and unrelated visible character are removed to find IoT device identification features such as website, brand, type, and model.

In [17] a two-stage of classification technique is presented to identify IoT devices’ traffic in a smart city. Also, in [18] a multi-stage ML is developed to classify 28 distinct IoT devices. Naive Bayes (multinomial classifier) is used in the first stage to classify the textual extracted attributes and the result, as well as, the statistical attributes are fed to the random forest classifier at the last stage.

In addition to the above, researchers have recently focused on ML algorithms especially random forest (RF) for identification purposes. In [19] a dynamic deduction method is presented for IoT device detection, classification, anomaly discovery, and health monitoring. A set of ML algorithms have been used, but RF achieves the best accuracy result. Also, in [20] framework is presented for identifying IoT and malicious detection from network traffic. Various ML algorithms were applied but the RF model achieved the best accuracy: around 94.5% for device type identification and 97% for detection of abnormal traffic. In [21] RF algorithm is applied to identify 27 devices type during the setup phase and achieved 81.5% as an overall accuracy.

It is evident from the previous studies that most of the authors tend to use multi-stage classification if their extracted features include textual elements. Moreover, few researchers have shed light on identifying the traffic generated during the initialization processes although this is the most important time to know the new setup IoT devices. In this paper, network traffic data are collected from 20 IoT devices during initialization time (setup and startup times), eight of them are home devices, while the rest are taken from the publicly available captures_IoT-Sentinel dataset [22]. Then, the collected packets are filtered and a set of features are extracted from each of them, then some statistical operations are used to create passive DFPs for each device with 30 features as a final DFP length.

Four ML algorithms are applied to identify the created DFP of the authorized devices, while the unknown devices are isolated. These algorithms are Support Vector Machine (SVM), Decision Tree (DT), Ensemble Random Forest, and Gradient Boosting Classifier (GBC). The final step is to apply a hard voting classifier (HVC) using the estimators with the best accuracies. The contributions of this work are:
Extract the numeric and textual features from the IoT device traffic during the initialization process then convert all textual features to numeric and create DFPs for each device.

Applying four ML algorithms after adjusting their parameters and verifying their ability to identify DFPs.

Increasing the identification accuracy by applying the HVC model using the tested models that give the best identification results as the algorithm estimators.

The remaining of this paper is formulated as follows: Traffic collection and data analysis are clarified in section 2. The Proposed DFP process is explained in section 3. Machine learning algorithms based tuning parameters are demonstrated in section 4. The performance evaluation and results are discussed in section 5. Finally, section 6 concludes the paper and reveals future work.

2. Traffic collection and data analysis

To collect IoT traffic data, Raspberry Pi 3 Model B+ is used and configured to work as a router. Eight IoT home devices are installed using Raspberry Pi as a home network. The generated traffics from these devices are collected using Wireshark Network Protocol Analyzer (iPhone X is specified for installing all devices and controlling them via devices’ applications, so the traffics related to iPhone X aren’t considered). It is found that the generated traffics during setup time are almost similar to the generated traffics during startup time. When any device wants to connect to the home network, Extensible Authentication Protocol over LAN (EAPOL) packet are sent from the access point to that device. EAPOL is a protocol used for network authentication between the router and the connected device. Since an EAPOL packet is the first packet to transmit when a device tries to connect to the network, so it's a good choice to start with when creating DFP. After network authentication with a 4-way handshake (4 EAPOL packets) is completed, a set of protocol packets (e.g., DHCP, DNS, ARP, ICMPv6, etc.) are exchanged to complete the network connection with an access point.

DNS carries a query name that is a device server name and it is stable unlike IP address that maybe changed [9] so, it is a strong feature and should take in consideration. DNS query names may be the same for devices with similar manufacture but they are totally different for distinct manufacture. Sometimes one device response with multiple DNS servers. Figure 1, shows a sample of collected traffic of multiple devices during initialization time. The DNS is filtered and it shows the differences and similarities in device DNS query names.

Since IoT devices are defined to perform a specific function, the first TCP session for some devices is almost fixed in the number of packets and the type of protocol used according to the source and destination port number (for example, dynamic ports or well-known ports such as 80 for HTTP and
443 for HTTPS) but it is varied in TCP window size and sometimes in TCP segment size. While the connectivity of these devices is dependent on network quality so, sometimes the packet count of the first TCP session is increased by the number of unreachable or retransmission packets. So, these differences in addition to some protocols packets details are qualified to create different DFPs for each IoT device. Figure 2, shows a sample of the first TCP session details of SonoFF power strip with IP address 192.168.100.95. As shown in the figure, 13 TCP packets are exchanged within about 1.5 sec. SonoFF power strip opened dynamic port (30736) while the other device opened a well-known port with HTTPS protocol.

Figure 2. Sample of SonoFF power strip's first TCP session details.

Alongside the home IoT devices, the network traffic of 12 devices is selected from a publicly available 31 Captures_IoT-Sentinel dataset (the rest are verified and found that either the collected traffic contains a few packets captured in a very short time or contains no EAPOL packets). All IoT devices used are listed in Table 1.

|   | Manufacturer | Device Name                        |
|---|--------------|------------------------------------|
| 1 | SonoFF       | SonoFF Power Strip (SPStrip)       |
| 2 | SonoFF       | SonoFF Power Plug (SPPlug)         |
| 3 | SonoFF       | SonoFF Bulb (SPBulb)               |
| 4 | SonoFF       | SonoFF Smart Switch with Temperature Sensor (SSSwitch) |
| 5 | Google Assistance | Google Home Mini (GHMini)        |
| 6 | Aswar        | Aswar Camera (ACamera)            |
| 7 | TEKIN        | TEKIN-Plug (TPlug)                 |
| 8 | Google       | Chromecast                         |
| 9 | D-LinkCam    | Authorized                         |
| 10 | D-LinkSensor | Authorized                         |
| 11 | D-LinkSwitch | Authorized                         |
| 12 | D-LinkWaterSensor | Authorized                      |
| 13 | Edimax       | Authorized                         |
| 14 | Ednet        | Authorized                         |
| 15 | TP-Link      | Authorized                         |
| 16 | TP-Link      | Authorized                         |
| 17 | WeMo         | Authorized                         |
| 18 | iKettle      | Authorized                         |
| 19 | SmarterCoffee| Authorized                         |
| 20 | Withings     | Authorized                         |

Table 1. Tested IoT devices.
3. Proposed DFP generation

The proposed DFP generation is based on two stages. In the first stage, the traffic of the new device (the new MAC address) is examined and features began to be extracted from obtaining an EAPOL packet and ended with the closing of the first TCP session (FIN flag). Table 2, shows the 25 extracted features from each packet during this stage using python Scapy library. To aggregate the features of all packets, a matrix of 25 columns (each representing a specific feature) is created with a different number of rows (packets). The differences in the number of rows are due to the inconsistency of packets each time they are collected. The representation of features is as follows:

- Assign a value of 1 to each of the following twelve logical features if present (ARP, EAPOL, IGMPv2, IGMPv3, ICMPv6, DNS (or MDNS), DHCP, UDPD, TCPHTs, TCPH, TCPD, and VCI), if exist otherwise set 0.
- Set the other thirteen feature values, which consist of seven features with numerical values (PLen, UDPDLen, TCPSLen, TCPWS, TTL, DHCPMS, and PRLLen), as well as six features with text values (MACSrc, MACDst, IPSrc, IPDst, HN, and QN).

After completing the feature extraction process, all duplicate rows are deleted. Equation (1), shows the matrix created in the first stage, where n denotes different numbers of rows and f refers to the extracted feature. The first stage of DFP generation can be simplified as in Algorithm 1.

\[
\text{Matrix} = \begin{bmatrix}
  f_{1,1} & f_{1,2} & \cdots & f_{1,25} \\
  f_{2,1} & f_{2,2} & \cdots & f_{2,25} \\
  \vdots & \vdots & \ddots & \vdots \\
  f_{n,1} & f_{n,2} & \cdots & f_{n,25}
\end{bmatrix}
\]

Table 2. Extracted features in the first stage.

| Type                      | No. of features | Features detail                                                                 |
|---------------------------|-----------------|---------------------------------------------------------------------------------|
| Data Link layer           | 4               | Source MAC (MACSrc), Destination MAC (MACDst), ARP protocol (ARP), and packet length if TCP (PLen) |
| Network layer             | 6               | Source IP (IPSrc), Destination IP (IPDst), EAPOL, IGMPv2, IGMPv3, ICMPv6.       |
| Transport layer           | 4               | UDP data (UDPD), UDP data length (UDPDLen), TCP segment length (TCPSLen), TCP window size (TCPWS). |
| Application layer protocol| 2               | DNS (or MDNS), DHCP.                                                           |
| IP                        | 1               | Time To Live (TTL).                                                            |
| TCP                       | 3               | TCP with HTTPS protocol (TCPHTS), TCP with HTTP protocol (TCPH), TCP with Dynamic source and destination ports (TCPD). Length of DHCP Parameter Request List (PRLLen), Maximum |
| DHCP                      | 4               | DHCP Message Size (DHCPMS), Vendor class identifier (VCI), Host Name (HN).     |
| DNS or MDNS               | 1               | Query Name (QN).                                                               |

In the second stage, the resulting matrix is converted to a vector consisting of 30 elements by applying some statistical operations to the extracted features taking into account increasing the weights of important features and shrinking DHCP features to be only one. The procedure of conversion is count number of 1 of all columns with logical features and some statistical operations are applied on columns of numeric elements as follows:

- Compute MAX and MIN, Average for PLen, TCPWS, and TTL.
- Compute MAX and MIN, for UDPDLen.
Algorithm 1. First stage of DFP generation.

```plaintext
packets= collected traffic data
MACDst=0     // The destination MAC address of new device
Matrix=[]        // The generated matrix
Counter=0
for pkt in packets:
    F=[]               // Variable used for saving features
    if pkt.has (EAPOL):
        MACDst=pkt. ether_addr_dst
    else: continue
    if(pkt.has (feature)) && MACDst not equ 0:  // Check all features listed in Table 2 with separate condition
        if  feature is logical:
            F.append(1)
        elsif feature is textual or numeric :
            F. append(feature)
        else :
            F.append(0)
    Matrix. insert (counter, F)   // Insert row vector F into the Matrix at row specified by counter
    counter+=1
Matrix. delete (repetitive rows)
```

- Compute Average of TCPSLen.
- The columns of DHCPMS and PRLLen each contain one value so put it.

The columns of textual features are converted to numbers each with a distinct way as follows:

- Merge element of columns MACSrc and MACDst, remove the repetitive addresses, and count the rest.
- The columns of IPSrc and IPDst are processed with the same procedure of MAC addresses columns.
- Convert HN column to ASCII codes then apply some statistical operations like MAX (MAXHN), MIN (MINHN), Average (AVGHN), and hostname length (HNLen).
- Create a lookup table for query names of all devices and give a specific number for all query names of each. For all unknown query names, 0 values are given.
- As mentioned earlier, some devices with the same brand may share some or all query names. So to strengthen the query name fields, another field is added that checks all query names availability in the lookup table, if one of them doesn’t exist 0 value is put otherwise query names count is put.

Since there is a similarity of some DHCP features for devices and some of these features may be guessed by intruders like HN, all DHCP features are merged to be one feature by taking the average of their values (average (DHCPMS, PRLLen, MAXHN, MINHN, AVGHN, HNLen)).

During the analysis step, it is found that some protocols such as ICMPv6 appeared in the traffic of some devices more than others, and this can be seen especially in devices produced by the same manufacture like D-Link devices. ICMPv6 appeared in DLSensor, DLSwitch, and DLWSensor traffics with different packet count and order but disappeared in DLCam traffics. To increase the differences between these devices, the first three locations of ICMPv6 protocol packets are concatenated and normalized by dividing the result by the largest location index (e.g., if three ICMPv6 appeared in packets indices A, B, and C, the new feature will be ABC/C). At the end of this stage, the DFP vector is generated with 30 features in length.
To create a more robust model, Gaussian noise is added with mean = 0 and standard deviation = 1. Moreover, since the generated DFP values contain both high and very low values, they are scaled up and down by 10 times. After preprocessing the resulted dataset, 20 DFPs are generated for each device (unknown devices are considered as one device).

4. Machine learning algorithms
This section provides a brief overview of four ML classification algorithms, and how to adjust the parameters if needed. Figure 3, depicts The schematic diagram of DFPs identification.

4.1. Support vector machine
SVM is a versatile supervised ML algorithm and it is inherently specified for binary classification. SVM can be used in both linear and nonlinear classification models. Linear SVM is used to classify the data domain linearly using a hyperplane. Whereas the nonlinear SVM is used to transform data domain (which cannot be separated linearly) into a feature space, where the data can be divided linearly to isolate the classes [23]. To define boundaries between the two classes, two straight lines are created which pass the nearby points.

Since the created DFPs dataset is associated with 18 classes; 17 for authorized devices while the remaining devices are used as the "unknown" class, so the One vs Rest (OvR) classifier is applied. The OvR classifier model involves M binary classifiers where M represents the number of classes. Every time a binary classifier is applied using one class as class 1 and the rest M-1 classes as class 0. Each classifier predicts a class probability, the higher positive probability is taken as a classification result. Equation (2), shows the OvR class prediction formula, where f(x) is a binary classifier function and n represents a classifier.

\[ f(x) = \arg\max_{\gamma} f_n(x) \]  

Grid search is an optimization method used to find the best model parameters using k fold cross-validation [24]. To determine the best kernel function and hyperparameters for SVM, grid search is applied twice with fivefold cross-validation, first to choose the appropriate kernel function and second, to choose hyperparameters. In this paper, the kernel functions that are decided to be checked by grid search are polynomial, radial basis, and sigmoid. It has been found that the polynomial kernel is the best for the proposed dataset to be trained with. Then, the grid search is also applied to find the best polynomial degree and hyperparameters and it is decided to make the range of degrees from 2 to 5, the range of C (regularization parameter) from 0.001 to 0.01 with step 0.001, and the range of gamma (determine the extent of the influence of the individual training sample) from 10 to 1000 with step 10.

Equation (3), illustrates the function of polynomial kernel, where X and Y represent vectors in the
input space, $\gamma$ represents gamma parameter, and $d$ represents the polynomial degree [25].

$$
\text{kernal}(X,Y) = (\gamma X . Y + C)^d
$$

(3)

4.2. Decision tree

DT is one of the common versatile supervised ML algorithms. It is constructed by iteratively splitting the training dataset into a sequence of subsets based on if-then conditions. There are a set of DT algorithms but for this work, C4.5 and CART algorithms are more suitable algorithms to work with. The splitting criterion of C4.5 is information gain which is defined as the difference in entropy of each feature in the dataset to the entropy of the same feature after partitioned according to the threshold value. The information gain is calculated for all features. The attribute with the maximum information gain is picked at the node. Equation (4), shows the entropy formula, where $P$ is the probability of the feature within class $i$ and $n$ represents the number of classes [26].

$$
\text{Entropy} = -\sum_{i} P_i \log_2(P_i)
$$

(4)

Gini impurity is the splitting criterion of the CART algorithm that measures the probability of a particular variable (randomly chosen) being wrongly classified. For each variable, the weighted sum of Gini indices is calculated then taken the variable which has the lowest values as a node. Equation (5), shows the Gini index formula [27].

$$
\text{Gini - index} = 1 - \sum_{i} P_i^2
$$

(5)

To apply the appropriate criterion, a grid search with fivefold cross-validation is applied. Besides criterion, the max_depth parameter is also tuned to overcome overfitting. It is decided to make its range from 8 to 20.

4.3. Random forest

Random forest is an ensemble ML model that combines several decision trees. The training step is based on the bagging method which means bootstrapping and aggregating. Bootstrapping means each tree train randomly selected data samples and features. The results from decision trees are aggregated by majority voting. The advantage of the RF model is to decrease overfitting occurred in the decision tree and increase the model accuracy since it creates several trees instead of one tree that prone to misclassification as well as random forest look for the most suitable feature among a random subset of the dataset features [28].

4.4. Gradient boosting classifier

GBC is a type of machine learning boosting method which is building a strong model by recursively learning a weak model. The three main components of GBC are loss function, weak learner, and additive model. The loss function role is to estimate how accurate the model to predict the given data. The weak learner is a learner with a high error rate typically the DTs model. While the additive model refers to the iterative method of adding the weak learners. The input to the GBC model is the training data Dataset $(x_i, c_i)_{i=1}^n$, the differentiable loss function $(L(c, F(x)))$, and the iteration numbers (M), where $x_i$ is the input variable, $c_i$ is the observed values (0 or 1). The differentiable loss function can be written as in equation (6), where $P$ refers to the predicted probability.

$$
L = -c + P
$$

(6)

The training process is begun by finding the optimal initial prediction $f_0(x)$ as written in equation (7), where $\gamma$ is written in equation (8). After that, using the pseudo residuals for the number of iterations (m=1 to M iterations) as written in equation (9). Then $\gamma_m$ is computed using equation (10), finally,
$f_m(x)$ is updated using equation (11) which mean the prediction function of $x_i$, where $v$ refers to the learning rate and $h_m(x)$ represents the additional model for the prediction function (regression tree) [17].

$$f_0(x) = \arg\min_{c_i} \sum_{j=1}^{N} L(c_i, \gamma)$$

$$\gamma = \log \left( \frac{P}{1-P} \right)$$

$$r_{im} = -\frac{\partial L(c_i, f(x_i))}{\partial f(x_i)}\bigg|_{f(x)=f_{m-1}x}$$

$$\gamma_m = \arg\min_{x_i \in R_{ij}} \sum_{x_i} L(c_i, f_{m-1}(x) + \gamma)$$

$$f_m(x) = f_{m-1}(x) + v \times \gamma \times h_m(x)$$

5. Performance evaluation and results

The grid search algorithm is applied to the dataset to verify the appropriate hyperparameters and the polynomial kernel degree of SVM. The stability of the optimization results is found every time it is executed. So, to decrease the training time, the results are taken directly which are: degree = 2, C = 0.001, gamma = 10. While the grid search results with the DT model differ each time it is executed, once it is given gini as the best model criterion and other time entropy criterion as well as it has resulted in a various max_depth parameter. Therefore, the grid search is included with the DT model to give the best accuracy.

During DFPs generating, 0 values are placed for each device fingerprints with an unknown DNS query name, therefore, the probability of identifying the unauthorized devices is increased.

The performance measurement of the models is based on the computation of the F1-score and the accuracy of the DFPs determination as shown in the following equations, where TP denotes true positive, FN denotes false negative, TN denotes true negative, and TN denotes true negative.

$$F1\text{-}SCORE = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$DFPs\text{-}Identification\text{ Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

The overall accuracy of DFPs identification is approximately 95.1% using SVM. Only four devices (DLSensor, DLSwitch, EdnetGateway, and WeMoSwitch) were identified with 100% accuracy. Classification errors typically occur between devices within the same manufacturing and sometimes among devices within distinct manufactures. Unknown devices are identified well with about 99.6%
accuracy but with a 95.9% F1-score, which means some known devices are identified as unknown. The confusion matrix of the SVM model is shown in Figure 4.a. It is clear that only one DFP of ACamera is classified as unknown and there is only one mistake among devices within different manufactures (on DFP of DLCam is identified as Chromecast) and three mistakes among devices within the same manufactures (3 DFPs of SPBulb are identified as SPStrip).

For the DT model, the overall accuracy of identifying DFPs is approximately 97.9%. Since the DT model is based on if-then conditions, there have been instances of misclassifications between devices regardless of their manufactures. Accuracy of unknown devices tends to be as SVM (99.8%). Meanwhile, the F1-score is slightly higher (97.9%). The confusion matrix for DT is shown in figure 4.b. There is only one error defining DLCam, and DT having the same error as SVM for identifying ACamera.

In the RF and GBC model, the overall accuracy of DFP identification is about 98.8%. The dependent parameters of RF are entropy as the criterion, max_depth=20, and min_samples_leaf=10. While the dependent parameters of GBC are min_samples_leaf=10, max_depth=20, learning_rate=0.2, max_features='sqrt', n_estimators=100). There are slight differences between the two models in the accuracy of the devices, GBC achieved 100% accuracy and F1-score for unknown devices' fingerprints identification, while RF results are slightly lower. Figures 4.c and 4.d show confusion matrices for RF and GBC respectively. Only one mistake was made in both models in identifying Chromecast DFPs. To increase the identification accuracy, the hard voting classifier is applied using the best three tested models (DT, RF, and GBC) as the classifier estimators. Each estimator votes for one class and HVC takes the majority voting results. So, if the misclassification occurred in only one model, HVC will classify the DFPs correctly. Moreover, even the results of the three models are the same, it is not necessary for the error to occur with the same observation (e.g., three DFPs are tested, DT misclassification is only in the first one, RF misclassification is only in the second one, and GBC misclassification is only in the third one) so, by HVC the identification accuracy becomes better and here achieved about 99.5% as overall accuracy. The confusion matrix of the HVC is shown in figure 4.e. Table 3 shows the average F1-scores and accuracies of all devices for the five applied models.

6. Conclusion

This work demonstrated that the generated IoT DFPs during initializing operation can be accurately identified. Expanding DFPs to be including first TCP features, DNS query names, DHCP features, and all other important features as well as DFPs formulation method all together are qualified to produce strong fingerprints that increase the ability to identify them using nonlinear machine learning algorithms. Even ML algorithms are differently predicting DFPs dataset, the hard voting classifier can gather the prediction results and produce the majority voting so it increases the identification accuracy. Four ML models are applied to the proposed DFPs and choose the top three models with high precision as HVC estimators and achieved about 99.5% as overall accuracy. Although the computational cost of HVC is high, it is nothing as long as the identification accuracy of unauthorized devices is 100%. In near future, the proposed method is expanded to include IoT network traffic identification at each instance of time.
Figure 4. Confusion matrices of machine learning models.

(a) Confusion Matrix of SVM

(b) Confusion Matrix of DT

(c) Confusion Matrix of RF

(d) Confusion Matrix of GBC

(e) Confusion Matrix of HVC
Table 3. Identification results of machine learning models.

| Device Name          | SVM F1-score | SVM AC. | DT F1-score | DT AC. | RF F1-score | RF AC. | GBC F1-score | GBC AC. | HVC F1-score | HVC AC. |
|----------------------|--------------|---------|-------------|--------|-------------|--------|--------------|---------|--------------|---------|
| Aswar_Camera         | 0.94         | 0.99    | 0.96        | 0.99   | 0.97        | 0.99   | 0.98         | 0.99    | **0.98**     | **0.99** |
| Chromecast           | 0.95         | 0.99    | 0.93        | 0.99   | 0.97        | 0.99   | 0.98         | 0.99    | **0.98**     | **0.99** |
| D-LinkCam            | 0.93         | 0.99    | 0.98        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| D-LinkSensor         | 1            | 1       | 0.96        | 0.99   | 1           | 1      | 0.99         | 0.99    | 0.99         | 0.99    |
| D-LinkSwitch         | 1            | 1       | 0.99        | 0.99   | 1           | 1      | 0.99         | 0.99    | 1            | 1       |
| D-LinkWaterSensor    | 0.98         | 0.99    | 0.99        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| EdimaxPlug1101W      | 0.99         | 0.99    | 0.99        | 0.99   | 0.95        | 0.99   | 0.99         | 0.99    | 0.97         | 0.99    |
| EdnetGateway         | 1            | 1       | 1           | 1      | 1           | 1      | 0.99         | 0.99    | 1            | 1       |
| Google_Home_Mini     | 0.97         | 0.99    | 0.98        | 0.99   | 1           | 1      | 1            | 1       | 0.99         | 0.99    |
| SonoFF_Power_Tap     | 0.96         | 0.99    | 0.98        | 0.99   | 0.96        | 0.99   | 0.96         | 0.99    | 1            | 1       |
| SonoFF_Power_Strip   | 0.91         | 0.99    | 1           | 1      | 0.95        | 0.99   | 0.95         | 0.99    | 1            | 1       |
| SonoFF_Smart_Light_Bulb | 0.85    | 0.98    | 1           | 1      | 1           | 1      | 0.93         | 0.99    | 5            | 4       |
| SonoFF_Smart_Switch | 0.99         | 0.99    | 0.98        | 0.99   | 1           | 1      | 0.98         | 0.99    | 0.99         | 0.99    |
| TEKIN-Plug           | 0.99         | 0.96    | 0.99        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| TP-LinkPlugHS100     | 0.79         | 0.97    | 0.98        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| TP-LinkPlugHS110     | 0.79         | 0.97    | 0.99        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| WeMoSwitch           | 1            | 1       | 0.96        | 0.99   | 1           | 1      | 1            | 1       | 1            | 1       |
| Unknown              | **0.95**     | **0.99**| **0.97**    | **0.99**| **0.98**    | **0.99**| **1**        | **1**   | **1**        | **1**   |

Acknowledgments
We would like to acknowledge the Mustansiriyah University (www.uomustansiriyah.edu.iq) for supporting to complete this work.

References
[1] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, “Internet-of-things-based smart
environments: state of the art, taxonomy, and open research challenges,” *IEEE Wirel. Commun.*, vol. 23, no. 5, pp. 10–16, 2016.

[2] S. Zeadally and M. Tsikerdekis, “Securing Internet of Things (IoT) with machine learning,” *Int. J. Commun. Syst.*, vol. 33, no. 1, pp. 3–16, 2020.

[3] A. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, “Ad-IoT: Anomaly detection of iot cyberattacks in smart city using machine learning,” in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*; *2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 178–181.

[4] H. Guo and J. Heidemann, “IP-based IoT device detection,” in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, 2018, pp. 36–42.

[5] H. Guo and J. Heidemann, “Detecting IoT devices in the internet,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2323–2336, 2020.

[6] S. Bahizad, “Risks of Increase in the IoT Devices,” in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*; *2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 178–181.

[7] A. Dalai et al., “A fingerprinting technique for identification of wireless devices,” in *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2018, pp. 1–5.

[8] A. Sivanathan et al., “Classifying IoT devices in smart environments using network traffic characteristics,” *IEEE Trans. Mob. Comput.*, vol. 18, no. 8, pp. 1745–1759, 2018.

[9] Z.-X. Xu, Q. Dai, G. Xu, H. Huang, X.-B. Chen, and Y.-X. Yang, “IoT Device Recognition Framework Based on Network Protocol Keyword Query,” in *International Conference on Artificial Intelligence and Security*, 2020, pp. 219–231.

[10] A. Hameed and A. Leivadeas, “IoT traffic multi-classification using network and statistical features in a smart environment,” in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–7.
[20] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, “A machine learning based framework for IoT device identification and abnormal traffic detection,” *Trans. Emerg. Telecommun. Technol.*, p. e3743, 2019.

[21] 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)*, 2017, pp. 2177–2184.

[22] “The Kaggle website,” 2021. https://www.kaggle.com/drwardog/iot-device-captures.

[23] S. Suthaharan, “Support vector machine,” in *Machine learning models and algorithms for big data classification*, Springer, 2016, pp. 207–235.

[24] I. Syarif, A. Prugel-Bennett, and G. Wills, “SVM parameter optimization using grid search and genetic algorithm to improve classification performance,” *Telkomnika*, vol. 14, no. 4, p. 1502, 2016.

[25] I. S. Al-Mejibli, J. K. Alwan, and H. Abd Dhafar, “The effect of gamma value on support vector machine performance with different kernels,” *Int. J. Electr. Comput. Eng.*, vol. 10, no. 5, p. 5497, 2020.

[26] D. B. Seal, S. Saha, P. Mukherjee, M. Chatterjee, A. Mukherjee, and K. N. Dey, “Gene ranking: An entropy & decision tree based approach,” in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2016, pp. 1–5.

[27] M. Shaheen, T. Zafar, and S. Ali Khan, “Decision tree classification: Ranking journals using IGIDI,” *J. Inf. Sci.*, vol. 46, no. 3, pp. 325–339, 2020.

[28] T. Zhu, “Analysis on the Applicability of the Random Forest,” in *Journal of Physics: Conference Series*, 2020, vol. 1607, no. 1, p. 12123.