

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,500

Open access books available

176,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

IoT Device Identification Using Device Fingerprint and Deep Learning

Prashant Baral, Ning Yang and Ning Weng

Abstract

The foundation of security in IoT devices lies in their identity. However, traditional identification parameters, such as MAC address, IP address, and IMEI, are vulnerable to sniffing and spoofing attacks. To address this issue, this paper proposes a novel approach using device fingerprinting and deep learning for device identification. Device fingerprinting is generated by analyzing inter-arrival time (IAT), round trip time (RTT), or IAT/RTT outliers of packets used for communication in networks. We trained deep learning models, namely convolutional neural network (CNN) and CNN + LSTM (long short-term memory), using device fingerprints generated from TCP, UDP, ICMP packet types, ICMP packet type, and their outliers. Our results show that the CNN model performs better than the CNN + LSTM model. Specifically, the CNN model achieves an accuracy of 0.97 using the IAT device fingerprint of ICMP packet type, and 0.9648 using the IAT outlier device fingerprint of ICMP packet type on a publicly available dataset from the crawled repository.

Keywords: Internet of Things, deep learning, device identification, security, device fingerprinting

1. Introduction

IoT is used in varied industries including automobile, manufacturing, agriculture, and medicine, etc. With the increase in the usage of IoTs in varied fields, the data transfer between edge devices over the network has also increased. While IoT bridges the gap between the digital and physical world, compromised IoT devices can bring dangerous consequences. Wireless networks are more at risk than wired networks. Frames are encrypted in wireless communication, but the management and control frames are not encrypted as per IEEE 802.1 standards. This causes the wireless device identity prone to spoofing and denial of service attacks. Node forgery, once the adversaries get hold of the security credentials, can cause a major security threat.

Adversaries may use a compromised node to send incorrect data. For example, if an IoT device sending the temperature in the industry gets compromised, it will ruin the product, and the owner must bear a great loss. Many cryptography techniques, such as WEP and WPA, can be easily compromised. IP address, MAC address, or IMEI number could be used for device identification, but there are scenarios where these

addresses got spoofing [1]. The gravity of the impact the breach in IoT has on varied fields is substantial, and we need to come up with an appropriate security mechanism to reduce the risk of data being compromised by IoT device forgery.

Different metrics can be used for device identification such as IP address, MAC address, IMEI address, and other network parameters such as transmission time, transmission rate, inter-arrival time, and medium access time. Comparisons of different metrics for device identification are in **Table 1**. The parameters, such as MAC address and IP address, are easier to spoof, so the study has been made on finding out the important parameters that can distinguish the devices. In [2], transmission time, transmission rate, inter-arrival time, and medium access time have been compared. IAT and transmission time outperform the other parameters in device identification.

In this paper, we worked on the deep learning approach for device identification. Device fingerprint is created from the parameters extracted during the communication of a device with router. This device fingerprint is used to train the deep learning model and device identification.

We fingerprint a device using IAT, RTT, and its outliers and feed them to deep learning models for device identification. These parameters are easier to extract and are not spoofed that easily after creating the device fingerprint with them. Timestamps (from which IAT and transmission time are extracted) are generated at the receiver side, which makes it harder to sniff and spoof. The adversaries need to change their own behavior to get a hand on these parameters. IAT and RTT are varied for different devices due to different CPU configuration and clock frequency. IAT and RTT depend on cache configurations, data cache, instruction cache, clock frequency, busses, and NIC card. These hardware configurations have an impact on the packet transfer rates. The attackers might try to emulate the signature using different techniques such as (1) introduce delays in packets, (2) change the data rate, and (3) make a customized operating system. Even while considering such techniques for an attack, an attacker is not successful in emulating the device. The attacker must consider a spoofing a signature along with hiding its original signature.

We use deep learning to extract knowledge from the data. It allows us to better understand the system model and simulate. CNN learns the semantic in the images and patterns in the image graph. Similarly, LSTM is recognized as a good algorithm for the classification of time series data. We use these two deep learning algorithms for the classification of devices. In earlier research, mathematical tools such as Mann-Whitney U-Test were used, but these algorithms require much time invested in

Metrics/ Parameters	MAC address	IP address	Transmission time	Inter-arrival time
Spoofing	Easy	easy	difficult	difficult
Property	Hardware	network	hardware and software	hardware and software
Privacy concerns	Low	low	high	high
Header generation	sender wireless card	sender wireless card	receiver wireless card	receiver wireless card

Table 1.
Comparison of parameters for fingerprinting.

preprocessing of data. The machine learning approach also requires us to prepare structured data before feeding it to ML algorithms.

Deep learning algorithms have an advantage over these consequences as it learns through the unstructured data while going through each layer in deep learning algorithms. The key factor for using deep learning is its time for making a prediction. Deep learning has its parameters calculated while training, which is why, when we provide our fingerprint of the device, the prediction is made quickly. This would take more time if we had used ML or any other mathematical tools.

We use the dataset generated from our own setup, as well as a publicly available dataset for training the model. We use TCP, UDP, ICMP packets, ICMP packets, and outliers of those packets for creating the device fingerprint. A new method [3] of device identification by collecting the information of the device to generate a fingerprint of the hardware, which can be used for device identification has been introduced. They use four different types of packets, namely probe request, ping, TCP, and UDP packets, to generate IAT graphs and have lower accuracy using CNN for classification.

In our work, we fingerprint two devices: Samsung A20 and Samsung J5 Prime. We plot IAT and RTT of the packets (probe request for IAT and ping for RTT) of each device and used those as datasets and feed them to deep learning models for device identification. We also use the publicly available dataset from the *crawdad* repository [4] introduced by Radhakrishnan et al. [5] to verify our results. This dataset provides the IAT information collected actively and passively from different wireless devices using wire side observations in a local network. They captured traffic from 30 different devices including iPads, iPhones, netbooks, Google phones, IP cameras, Kindles, and IP printers, etc., from various applications and protocols such as TCP, UDP, Skype, ICMP, SCP, and Iperf. Our main contribution consists of:

- use parameters extracted from wireless communication to create a unique signature/fingerprint of hardware.
- different parameters (IAT and RTT) for creating unique signatures, which are separately used for training deep learning models.
- compared how well deep learning algorithm was in classification using different metrics.
- considered outliers in IAT graph to observe if it does better classification.

The remainder of the paper is organized as follows. Section 2 briefly discusses related work. Section 3 describes the device fingerprinting, setup, methods for extracting data, and creating image graphs, and preparation of datasets are also discussed in this section. This dataset is fed to deep learning models described in Section 4 for classification of device. Experimental results are presented in Section 5, and the paper is concluded in Section 6.

2. Related work

The use of IP address, MAC address, and IMEI number for device identification brings significant risks of critical information, and the device itself is compromised. This alerts the researcher to produce a flexible and effective technique for

device identification [1, 6, 7]. For example, a new stack [1] for the identity of IoT is proposed as it differs from the traditional identity of network devices and survey on attribute-based authentication for the identity of IoT devices.

Neumann et al. [2] surveys different features of the MAC layer such as transmission rate, transmission time, and inter-arrival time, and evaluated them on two criteria for effectiveness, fingerprint similarity at different time, and fingerprint dissimilarity of two different devices. In [2], authors use the IAT packets from wireless devices for creating digital fingerprints and created a histogram where each bin specifies the frequency of IATs in a specified range. Here, histogram is the fingerprint used for the classification of the the device and used to identify known and unknown devices from the database. The author tested the scenario where a malicious user tries to emulate the known device by introducing delay to the packets. The author concluded that different software and hardware make it difficult to emulate the hardware. In [2], authors use a passive approach for fingerprinting and, Radhakrishnan et al. [5] extended the work [2] using active approach for device fingerprinting. In the passive approach, we just observe the wireless communication to/from the device and use the important features of packets. Instead, in the active approach, we inject the signal to get a response from the device to get useful features. Sandhya et al. [8] used CNN but considered all types of packets flowing from devices to AP for device classification. This might be practical, but a lower accuracy of 86% may be problematic from a security point of view.

In [5], the author used a ping application to communicate between a device on campus. In [9], the author used IAT of probe request to fingerprint the device and used Mann-Whitney U-test for the analysis if two samples are of the same distribution. Miettinen [10] used 23 features such as ICMP, TCP, HTTP, and size from different layers (data link layer, transport layer, network layer, and application layer, etc.). The work collects these features of 23 for 12 packets and used a random forest algorithm for classification. The accuracy for 17 out of 27 was obtained 95% and 50% for the rest devices (10).

Robyns et al. [11] introduce the idea of noncooperative MAC layer fingerprinting, which does not require cooperation with the device as it uses some adversary nodes at the monitoring station to capture and monitor the bits of MAC frames without the user's permission. This hampers the privacy of the user but provides security from attacks from outside. The accuracy, when used for classification of 50 to 100 devices, was between 67–80%, but the accuracy decreases rapidly from 33–15% when device numbers were increased.

Kohno et al. [12] used the clock skew for fingerprinting devices. The work measured the timestamp by time difference of the time stamps using the traces from Tcpcdump. The work considered the scenario where IP addresses were changed during data collection. Maurice [13] used a probe request and response for fingerprinting, but the results were not that promising for similar devices. Cunche et al. [14] used probe request from an AP and in response got the list of wireless networks. The work used this vulnerability to identify people from the list of networks connected. Francois et al. [15] made use of behavioral fingerprinting and automatically disconnects the device, which has suspicious activity and asks it to reconnect based on the behavioral fingerprint. Sun et al. [16] use the fingerprinting method for localization of devices connected to Wi-Fi AP indoor or outdoor.

Xu [1] studied the challenges and opportunities in digital fingerprinting for both wired and wireless devices. The author extracted the features from the physical and MAC layers such as clock skew, IAT, transmission time, SSID, and frequency. The

work concluded IAT and transmission time as good parameters for device classification based on accuracy.

Kulin et al. [17] used different algorithms such as k-NN, decision tree, logistic regression, and neural networks for device classification using publicly available datasets. The performance of k-NN, decision tree, and logistic regression was good, but neural networks performed poorer than other classification algorithms with an average precision of 0.47 and recall value of 0.46. It is a common understanding that neural networks should perform better than others, but this was not the case in this work.

3. Device fingerprinting

We set up the devices in the lab for extracting the information about the devices. First, we set up Raspberry Pi as a router. Next, we use Samsung A20 and Samsung J5 Prime as an edge device (target IoT devices). Wireless communication between the edge devices and router was recorded. In the sniffing applications, Wireshark captures the packets incoming and outgoing on Raspberry Pi. These captured packets are used to calculate IATs/RTTs of packets and plot IAT, RTT, and IAT outlier graphs. These graphs are used as datasets to train and test the model. Python program is used to plot, label, and split the dataset. A split training set trains the deep learning model, and the testing dataset validates it. Our overall methodology is depicted in **Figure 1** and explained in detail in a subsection of this section and Section 4.

3.1 Our setup

Our setup has Raspberry Pi as a router and phones as the edge devices. Raspberry Pi (acts as a router) broadcasts an access point. The packets sent from edge devices are captured at the router side, which has a packet sniffing tool installed. Wireshark is installed in Raspberry Pi which inspects, deciphers, and keeps track of all incoming and outgoing packets to/from it. As there might be many packets coming to the router, we use the filter to find the required packet. We collected the data in two ways: 1. Probe request and response and 2. Ping request and response.

Probe requests are the packets broadcast by wireless devices, which consist of supported data rates and their capabilities. The access point receives these requests and responses with packets consisting of SSID, supported data rates, and encryption type, etc. We used a sniffing tool, Wireshark, to passively sniff the packets at the router level and use those packets for making IAT graphs.

Ping sends the ICMP echo request packet to any device on the network and waits for the response from the target device. In our setup, we ping the edge device, and the edge device responds to the router. This packet communication of ICMP is passively observed and recorded by Wireshark. This data is used for making RTT graphs.

3.2 Analysis of data and create image graph

The data collected by a sniffing tool and must be processed to obtain IAT and RTT. We obtain data using a sniffing tool in Raspberry Pi. These data are timestamps of incoming and outgoing packets. We process timestamps to calculate the IAT and RTT of the packet.

After we obtain the value of IAT and RTT of packets, we write a Python program to plot the graph and download it. IAT and RTT graph is plotted as a line graph of 100

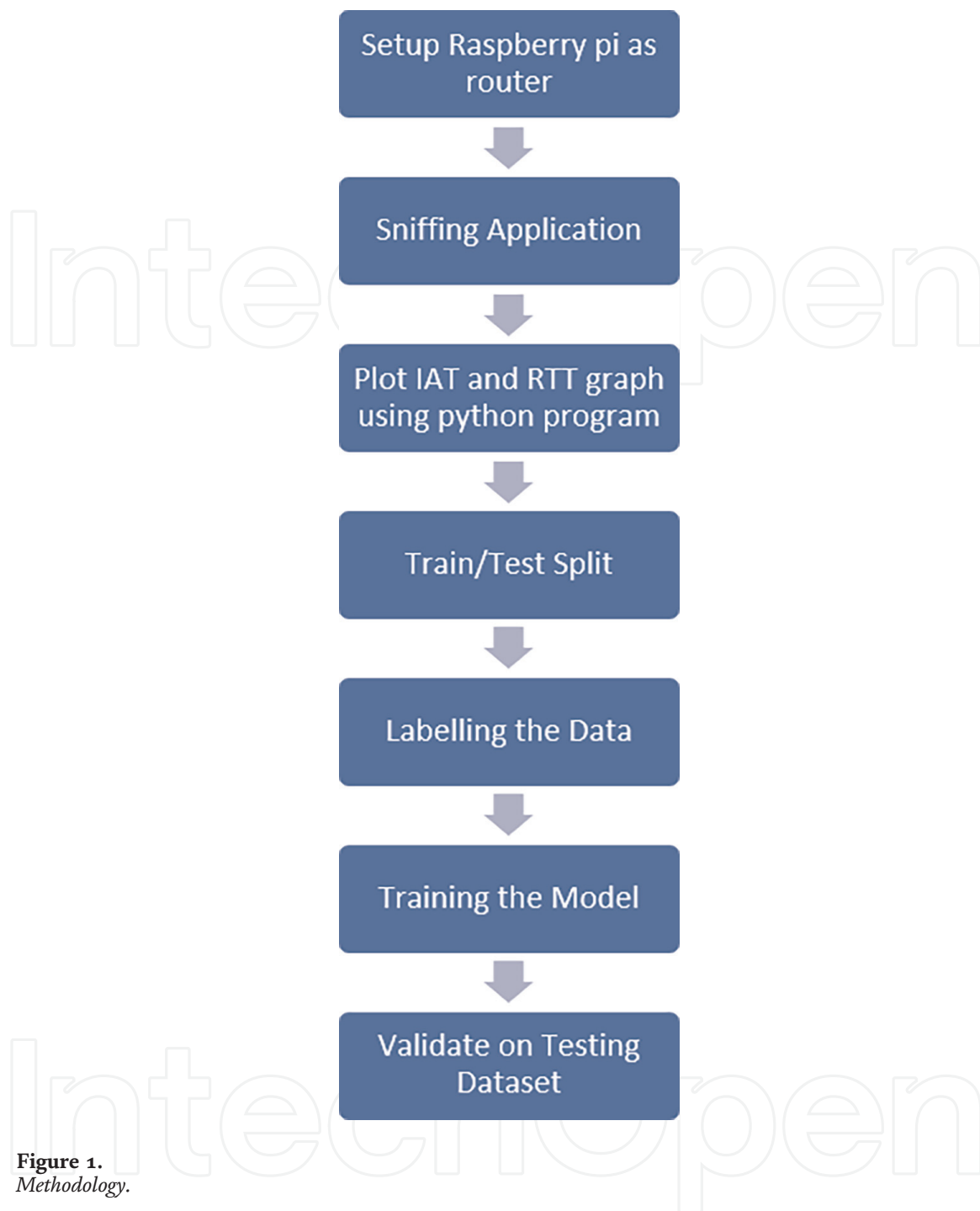


Figure 1.
Methodology.

IATs/RTTs. The plot of IAT and RTT is shown in **Figures 2–4**. We use IAT and RTT separately for device identification.

3.3 Preparation of data

The image obtained by plotting the graph must be labeled before we use that data for training and testing the model based on different metrics. We label the data using Python. For two phones, 0 represents Samsung A20 and 1 represents Samsung Prime. We split the total images into training data and testing data. For each IAT and RTT, we use 75 images for training and 30 for testing for each device (total 150 for training and 60 for testing). After creating an image and labeling it, we apply CNN and CNN + LSTM algorithms for image classification.

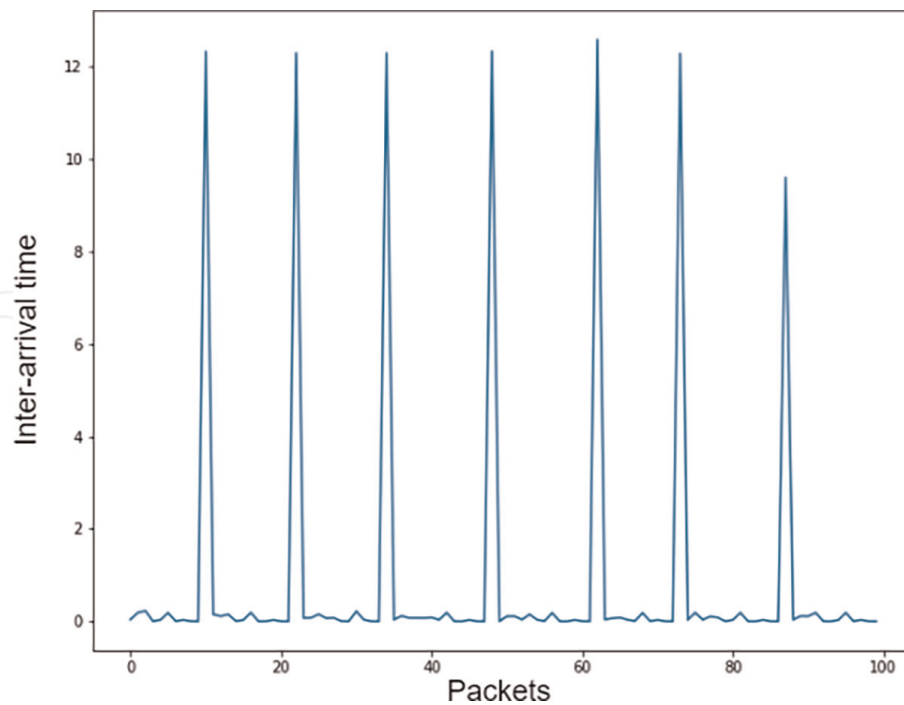


Figure 2.
IAT graph from our setup.

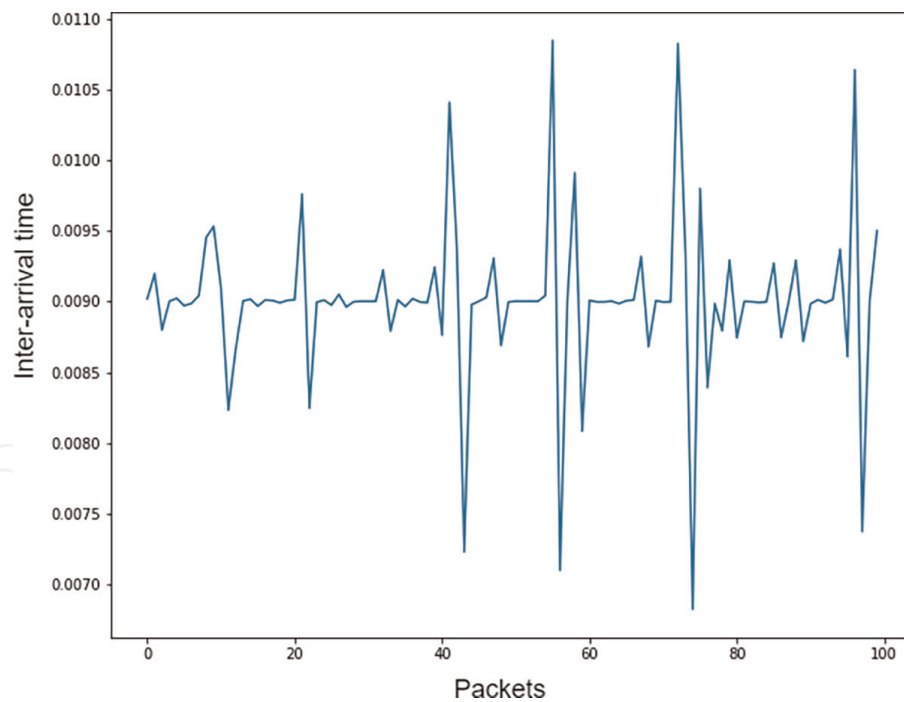


Figure 3.
IAT graph from verification dataset.

We use the dataset of IAT from *crawdad*, which was developed by Ulugac et al. [4]. The dataset is the collection of IATs of different devices. We use four devices: two iPad and two Dell notebooks for the verification of models. First, we use ICMP packets for generating the IAT graph. Since we are comparing the classification using a single packet type, multiple packet types, and an outlier, we also use TCP, UDP, and ICMP packets for generating the IAT graph and outliers. We plot a graph using 100

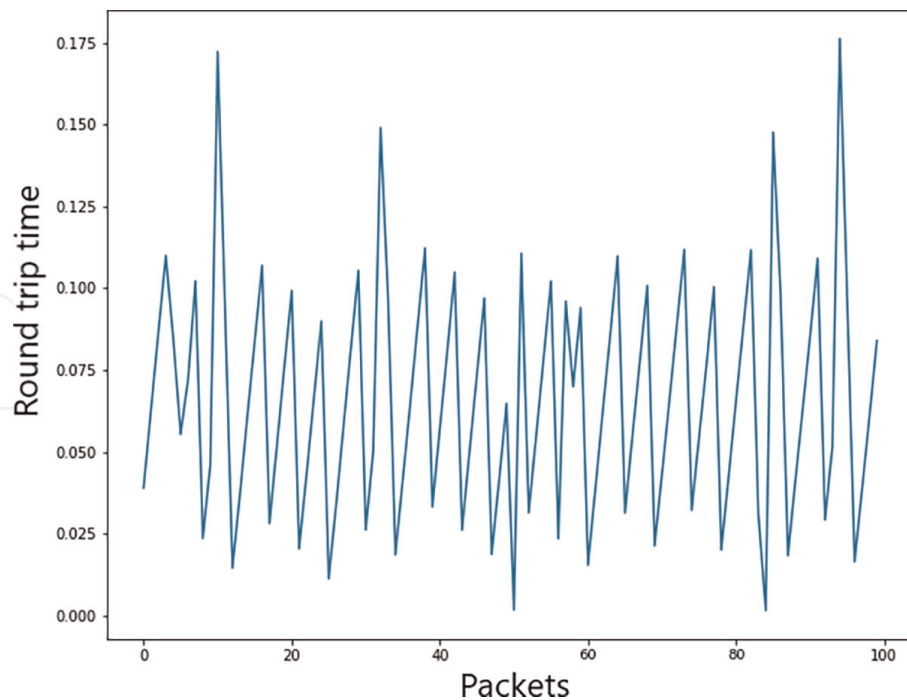


Figure 4.
RTT graph from our setup.

IATs. As in our setup, we similarly label zero for Dell notebook1, one for Dell notebook2, two for ipad1, and three for ipad2 and split it into a training and testing dataset.

4. Deep learning model for classification

We use a convolutional neural network (CNN) and a combination of a convolutional neural network and long short-term memory (CNN + LSTM) for device classification. Since we are using the image of time series data, we consider CNN due to its large breakthrough in image recognition. Moreover, CNN is very cost-effective due to the reduced number of parameters without losing the quality. Furthermore, due to the recognition of LSTM for time series data and our consideration of converting time series data to images and using the image for classification, we experiment if the combination of CNN + LSTM could give better results than CNN alone.

4.1 Convolutional neural network for device classification

The created image is colored, but for this classification problem, we convert the image into grayscale and reduce the image size to $256 * 256$. Initially, it was $800 * 800$. Then we split the labeled data into training and testing datasets and use the training set to train the CNN model. Our CNN model has the first convolution layer with 32 filters and a kernel size of $5 * 5$. The input size of this layer was set to $256 * 256 * 1$. Next, we use max-pooling with stride length 2; this helps in reducing the parameters by selecting the maximum from four (2 in x-direction and 2 in the y-direction). The next convolution layer in our model has 64 filters and a kernel size of $3 * 3$. The input

INPUT LAYER	INPUT	?,256,256
	OUTPUT	?.256.256
CONV2D LAYER	INPUT	?, 256,256
	OUTPUT	?. 256. 256. 32
MAXPOOL2D LAYER	INPUT	?,256,256,32
	OUTPUT	?,128,128,32
CONV2D LAYER	INPUT	?,128,128,32
	OUTPUT	?,126,126,48
MAXPOOL2D LAYER	INPUT	?,126,126,48
	OUTPUT	?,63,63,48
CONV2D LAYER	INPUT	?,63,63,48
	OUTPUT	?,62,62,64
MAXPOOL2D LAYER	INPUT	?,62,62,64
	OUTPUT	?,31,31,64
FLATTEN LAYER	INPUT	?,31,31,64
	OUTPUT	?.61504
DENSE LAYER	INPUT	?,61504
	OUTPUT	?.256
DENSE LAYER	INPUT	?,256
	OUTPUT	?.84
DENSE LAYER	INPUT	?,84
	OUTPUT	?.2

Figure 5.
 CNN model summary.

to this layer is set by Keras. We again use max-pooling with stride length 2. The third convolution layer has 128 layers and a kernel size of $2 * 2$, and we max-pooled with a stride length of 2 for this layer as well. For all these convolution layers, we use Rectified linear Unit (ReLU) as an activation function. Next, we use a flattened layer and two dense layers with 128 and 64 nodes followed by a dense layer with four nodes with softmax as activation function. **Figure 5** shows the model summary of CNN. The model is compiled using categorical cross-entropy for calculation of loss and Adam as the optimizer. We use both IAT and RTT data for training the CNN model and check how good was its classification using different metrics. Furthermore, we use an outlier of IAT data for classification. While training for different datasets, the number of nodes and epochs is changed.

4.2 Combination of CNN and LSTM for device classification

We combine CNN and LSTM using the concept of TimeDistributed layer. We provide n images at a time to the first TimeDistributed convolution layer; this applies

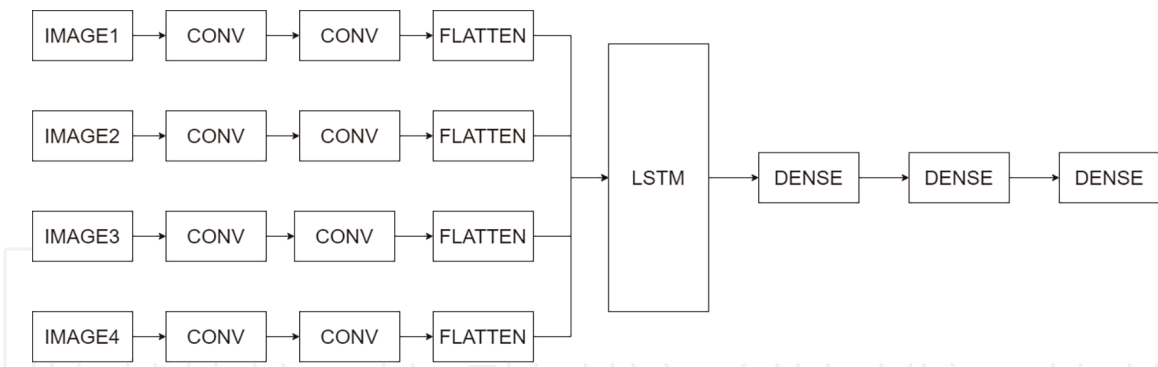


Figure 6.
CNN + LSTM model.

the same filter to the n images. We use the same three identical CNN layers but TimeDistributed. This is illustrated in **Figure 6**. The input to the first layer is $n * 256 * 256 * 1$. Another input size is managed by Keras. This model has an additional LSTM layer with 32 nodes after CNN layers. The output of Maxpool2D is flattened to get one single vector. This is a feed to LSTM and a dense layer. **Figure 7** shows the model

INPUT LAYER	INPUT	? , 4, 256, 256
	OUTPUT	? , 4, 256, 256
TD CONV2D LAYER	INPUT	? , 4, 256, 256
	OUTPUT	? , 4, 256, 256, 32
MAXPOOL2D LAYER	INPUT	? , 4, 256, 256, 32
	OUTPUT	? , 4, 128, 128, 32
TD CONV2D LAYER	INPUT	? , 4, 128, 128, 32
	OUTPUT	? , 4, 127, 127, 64
MAXPOOL2D LAYER	INPUT	? , 4, 127, 127, 64
	OUTPUT	? , 4, 63, 63, 64
TD CONV2D LAYER	INPUT	? , 4, 63, 63, 64
	OUTPUT	? , 4, 63, 63, 128
MAXPOOL2D LAYER	INPUT	? , 4, 63, 63, 128
	OUTPUT	? , 4, 31, 31, 128
FLATTEN LAYER	INPUT	? , 4, 31, 31, 128
	OUTPUT	? , 4, 123008
LSTM LAYER	INPUT	? , 4, 123008
	OUTPUT	? , 4, 32
DENSE LAYER	INPUT	? , 4, 32
	OUTPUT	? , 4, 256
DENSE LAYER	INPUT	? , 4, 256
	OUTPUT	? , 4, 84
DENSE LAYER	INPUT	? , 4, 84
	OUTPUT	? , 4, 2

Figure 7.
CNN + LSTM model summary.

summary of CNN + LSTM. LSTM makes use of chronological data and previous frame data to find what is useful in prediction. The model is compiled using categorical cross-entropy for calculation of loss and Adam as the optimizer. We use a combination of CNN and LSTM and observe how good the prediction the model can make. While training for different datasets, the number of nodes and epochs is changed.

4.3 Metrics for model evaluation

Evaluation of the model is an important task in data science. We need to make sure our model is not overfitted. Overfitting is a modeling error in statistics, which occurs due to the model aligning too closely to the limited data points. There are different techniques to prevent overfitting. Some of the techniques that we use are: reduce learning rate and dropout Layer. While training the model, we can monitor the validation accuracy and if it does not increase for a certain epoch, we reduce the learning rate by a certain factor. Below is the snippet of reducing learning rate where we monitor the validation loss and reduce the learning rate by a factor of 0.1 when for 3 consecutive epochs validation loss is increased.

```
tf.keras.callbacks.ReduceLROnPlateau(monitor = "val.  
loss," factor = 0.1, patience = 3, verbose = 0, and min lr = 1e-6).
```

Similarly, the dropout rate can be specified to the layer as the probability of setting each input to the layer to zero. Below is the code for adding the dropout layer. The rate is set to 0.3, which drops 0.3 of input units.

```
model.add(Dense(128, activation = 'relu')).  
model.add(Dropout(0.3)).
```

The most common metric used for the evaluation of the algorithm is classification accuracy. Classification accuracy is equal to the number of correct predictions made divided by the total number of predictions made.

In our case, we use categorical cross-entropy for the calculation of loss, which makes the use of the probability of belonging to a class for the calculation of loss.

$$\text{Classification loss} = - \sum_{i=1}^{\text{outputsize}} y_i \log f(s)_i \quad (1)$$

Where, y_i is the class and $f(s)_i$ is the probability of belonging to that class. We also need to control the number of times we train the model. This is called epoch. Too much training can result in network overfitting to the training data. While training a model for certain epochs if validation error increases but the training loss decreases or remains constant, we can conclude that our model is overfitting as shown in **Figure 8**.

5. Results

Our setup has the phone Samsung A20, and Samsung Prime communicating with Raspberry Pi. As Section 3.2, we created the IAT graph using probe request and response from these devices to Raspberry Pi and prepared the data for feeding to CNN and evaluated the model. We trained the CNN model as in Section 4. A for 10 epochs and obtained the accuracy of 1.00 and loss of 0.0021 on training data. Accuracy in the validation dataset was 1.00 and loss of 0.0021. Using the IAT graph for classification and CNN + LSTM model and running for 30 epochs, the accuracy and loss were 1 and

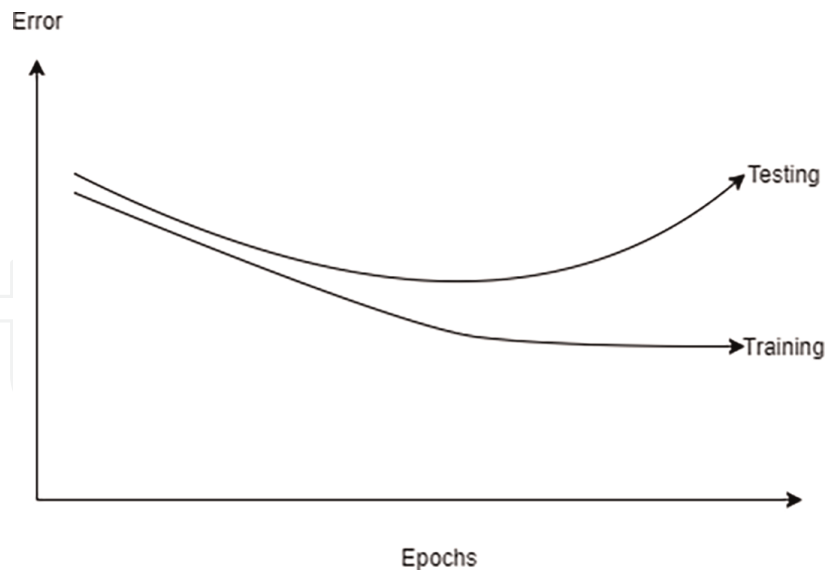


Figure 8.
Model loss.

0.0015 in the training dataset and 1 and 0.0011 in the validation dataset. Similarly, we created the RTT graph using ping as in Section 3.2 and trained for 10 epochs while feeding to CNN and 40 epochs while feeding to CNN + LSTM and achieved 100% accuracy in classification in both.

We used the dataset of IAT from crowdad, which was developed by Ulugac et al. [4] for verification. We used ICMP packets used by two Dell notebooks and two iPads communicating in the local area network. Using CNN for classification and running for 10 epochs, we achieved the accuracy of 1 and loss of 1.4×10^{-4} in the training dataset. We achieved an accuracy of 0.97 and a loss of 0.1326 in the validation dataset.

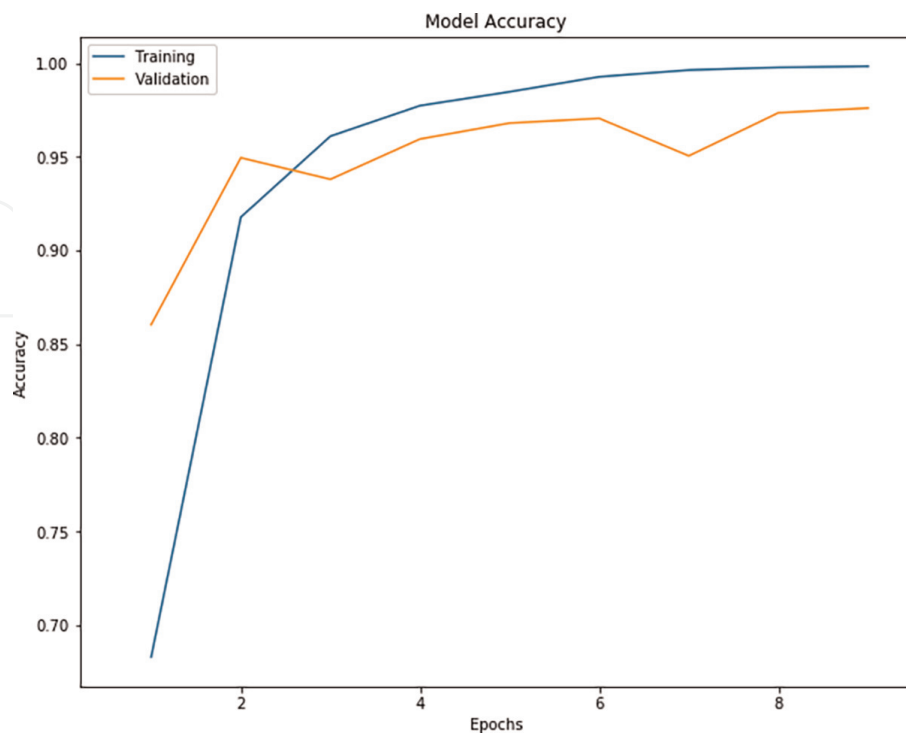


Figure 9.
Accuracy using IAT(ICMP) as parameter from verification dataset using CNN.

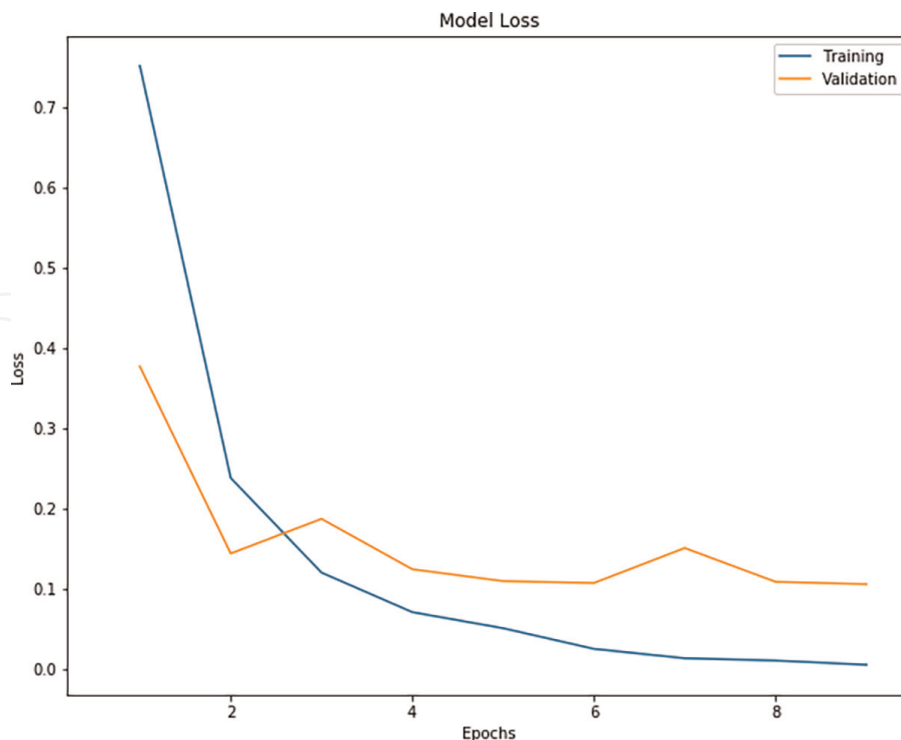


Figure 10.
Loss using IAT(ICMP) as parameter from verification dataset using CNN.

Figures 9 and 10 show the learning curve of the CNN model. Using CNN + LSTM for classification and running for 35 epochs, we achieved an accuracy of 0.9463 and a loss of 0.1906 in the training dataset. We achieved an accuracy of 0.9060 and a loss of 0.3115 in the validation dataset. **Figures 11 and 12** show the learning curve of the CNN + LSTM model.

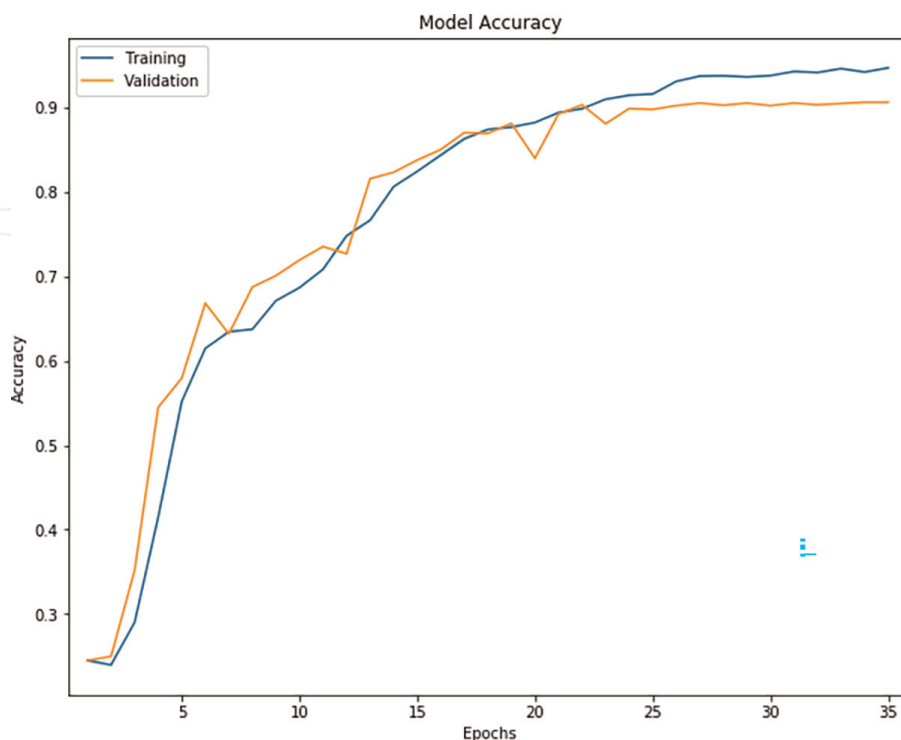


Figure 11.
Accuracy using IAT(ICMP) as parameter from verification dataset using CNN + LSTM.

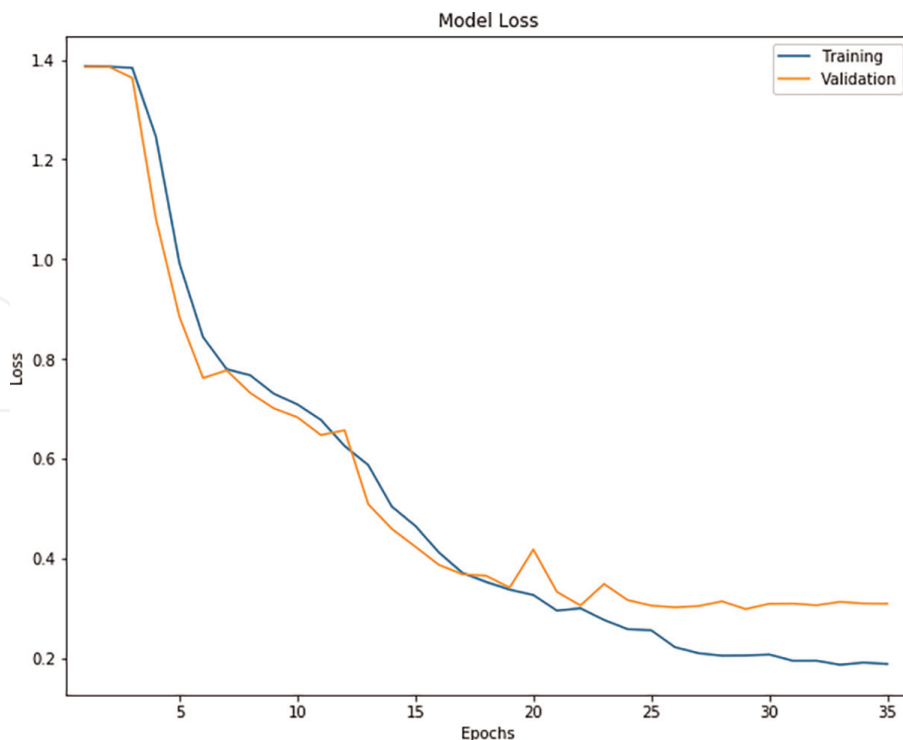


Figure 12.
Loss using IAT(ICMP) as parameter from verification dataset using CNN + LSTM.

After analyzing the IAT graph, we found that there is a regular pattern of outliers and considered if the outliers in the IAT graph can better classify a device using these deep learning algorithms. We utilized the outliers in the IATs of the verification dataset for four devices: two Dell notebooks and two iPads. There lies inter-burst latency between the IAT packets, and we utilize these for classification. We plotted the outlier graph for four devices considering their own threshold for each. We plotted outlier graphs and used CNN and CNN + LSTM algorithms for classification. We used the same CNN configurations ranging from convolution layers, input size, activation function, and number of layers, etc., for the classification using the IAT outlier graph. We ran the model for 10 epochs. We achieved the accuracy and loss of 0.9981 and 0.0079 and validation accuracy and loss of 0.9648 and 0.1397, respectively. **Figures 13** and **14** show the learning curve of the CNN model using an outlier dataset for training. We also used the same CNN + LSTM configurations ranging from convolution layers LSTM layer, activation function, and number of layers, etc., for the classification using the IAT outlier graph. We ran the model for 15 epochs. We achieved the accuracy and loss of 0.9870 and 0.0520 and validation accuracy and loss of 0.9574 and 0.1422, respectively. **Figures 15** and **16** show the learning curve of the CNN + LSTM model using an outlier dataset for training.

To validate the improvement of classification using single type packets (ICMP/probe request) in our work, we also classified the devices using TCP, UDP, and ICMP packet types from the same dataset of IAT from crawdad for classification as in [8]. The IAT graphs generated for these packet types were together used for training and testing the model. We trained the CNN model for 16 epochs and put the dropout layer after flattened layer to prevent overfitting. We used 18,000 training images and 6000 testing images and obtained an accuracy of 0.9656 and a loss of 0.0894; the validation accuracy and validation loss were 0.9290 and 0.3073, respectively. **Figures 17** and **18**

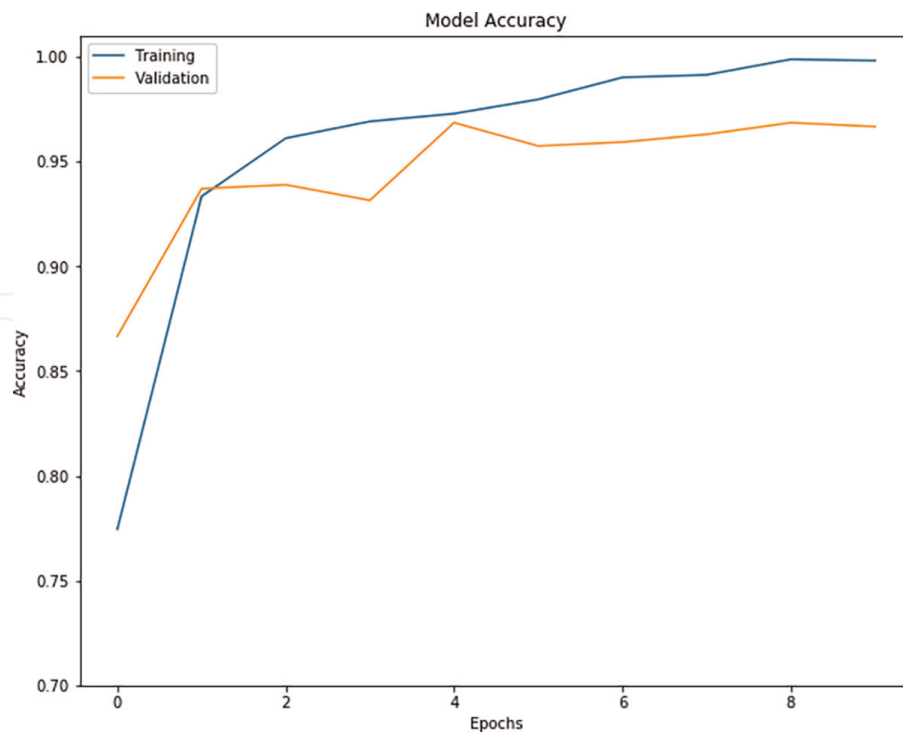


Figure 13. Accuracy using IAT(ICMP) outlier graph from verification dataset using CNN.

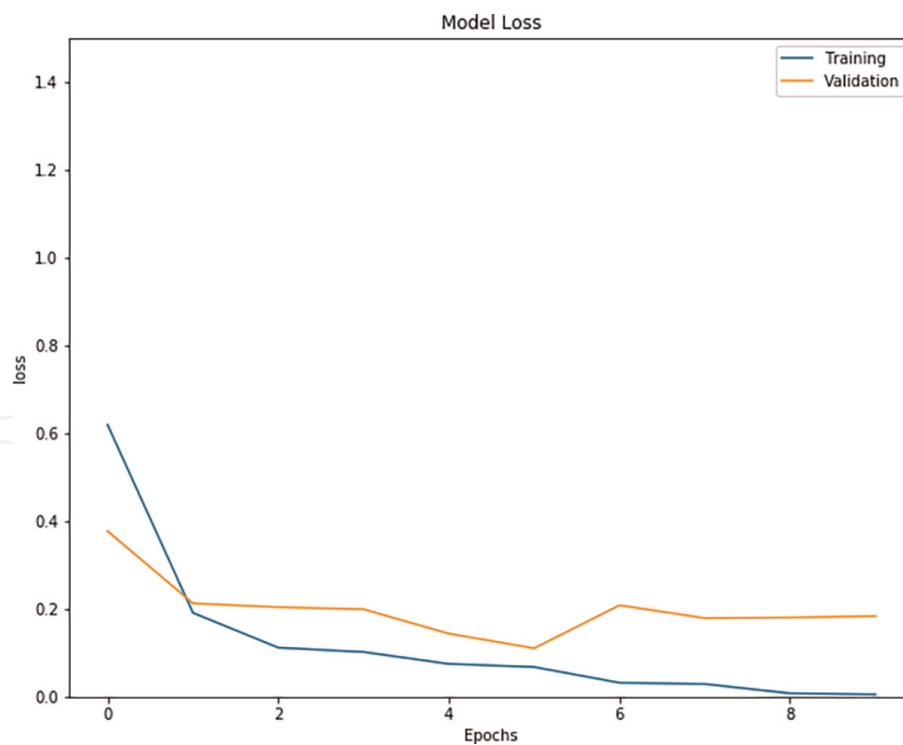


Figure 14. Loss using IAT(ICMP) outlier graph from verification dataset using CNN.

show the learning curve of CNN model using image graphs of IAT generated using TCP, UDP, and ICMP packet types from the verification dataset.

Again, for this different type of packet, we considered the outliers and classified them using the outliers of IAT. We trained the CNN model for 20 epochs and put the

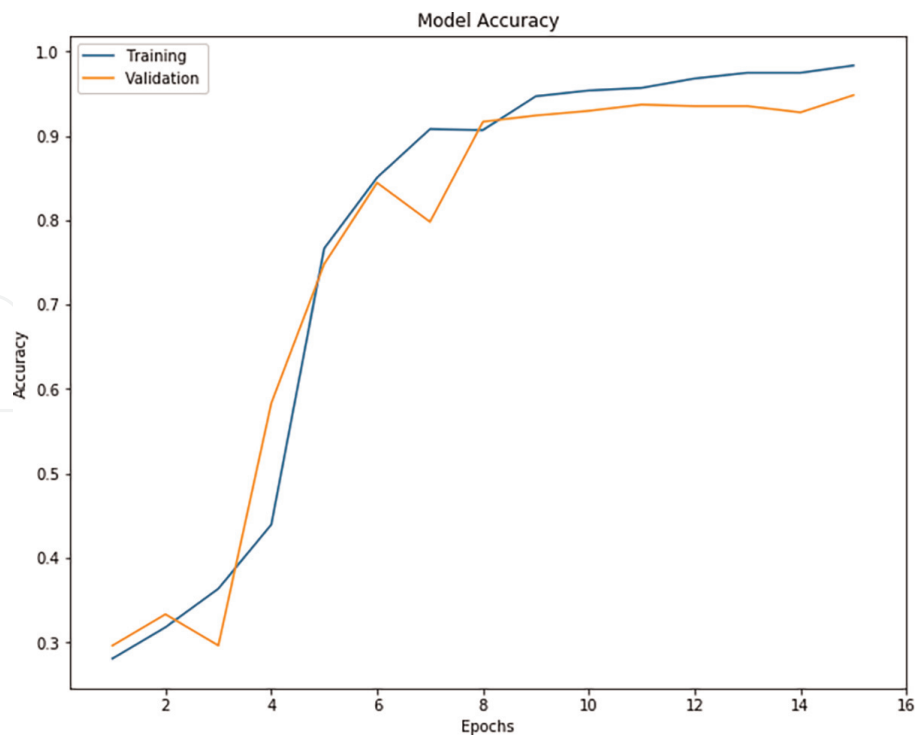


Figure 15. Accuracy using IAT (ICMP) outlier graph from verification dataset using CNN + LSTM.

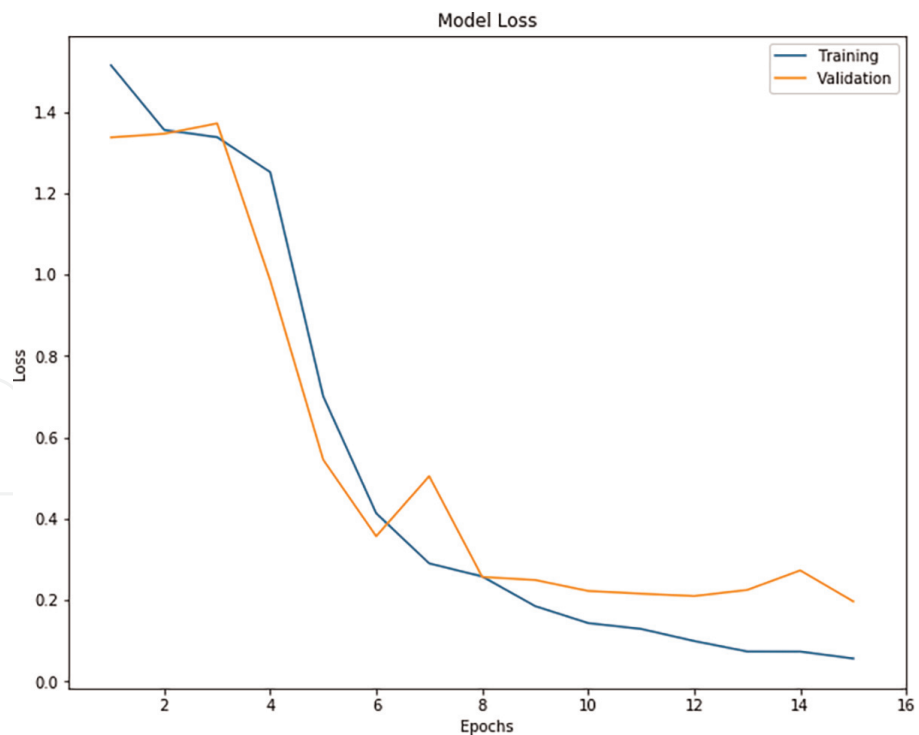


Figure 16. Loss using IAT (ICMP) outlier graph from verification dataset using CNN + LSTM.

dropout layer after flatten layer to prevent overfitting. We used 5440 training images and 1700 testing images and obtained an accuracy of 0.8888 and a loss of 0.2704; the validation accuracy and validation loss were 0.8504 and 0.4344, respectively.

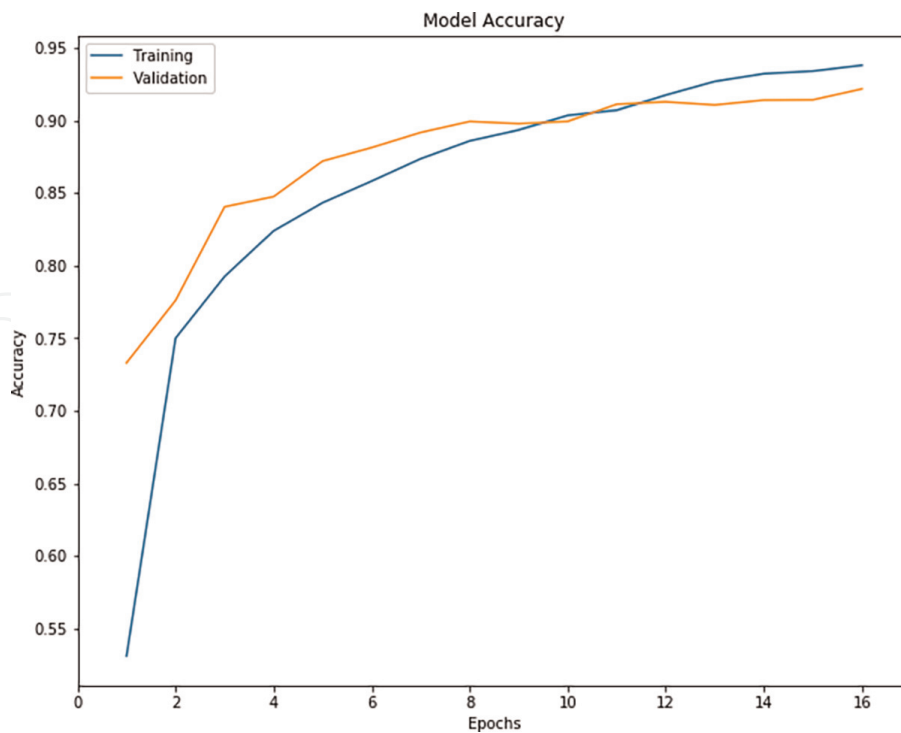


Figure 17. Accuracy using IAT (TCP, UDP, ICMP) as parameter from verification dataset using CNN.

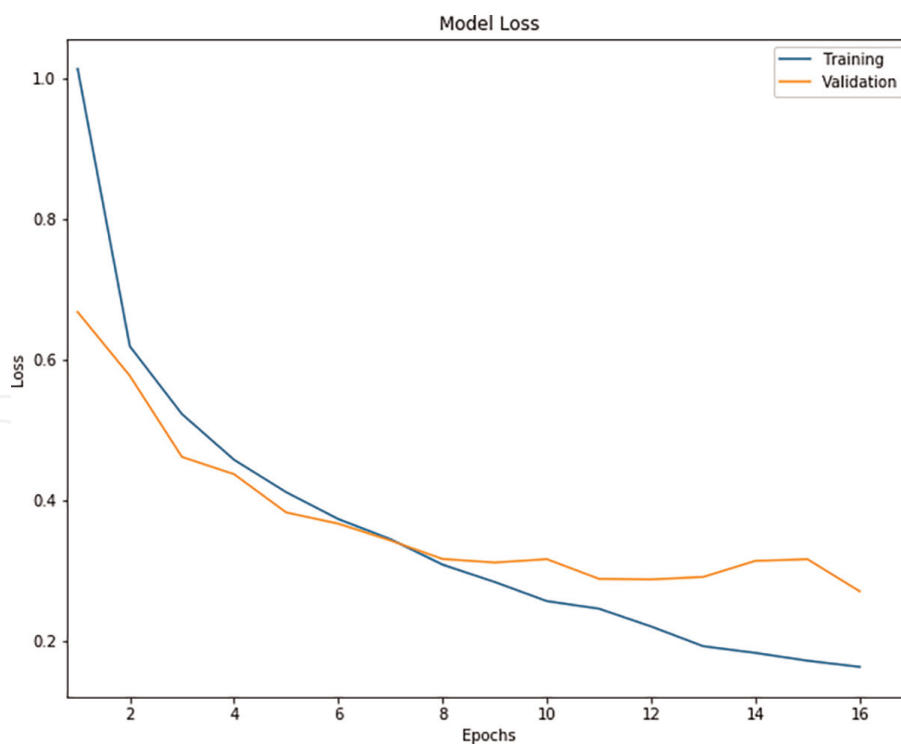


Figure 18. Loss using IAT (TCP, UDP, and ICMP) as parameter from verification dataset using CNN.

Figures 19 and 20 show the learning curve of the CNN model using image outlier graphs of IAT generated using TCP, UDP, and ICMP packet types from the verification dataset.

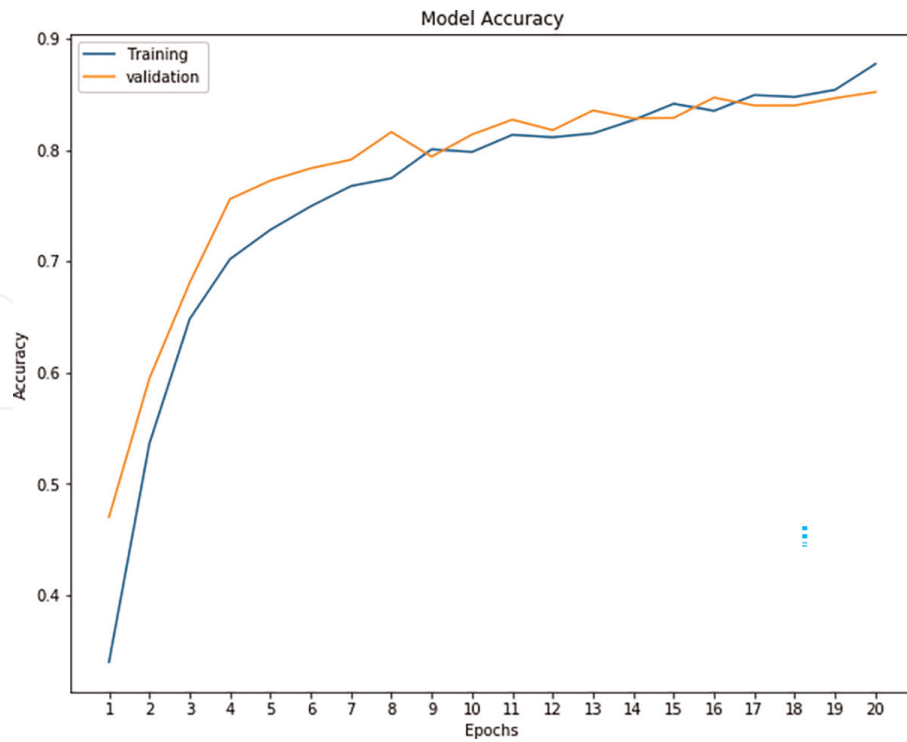


Figure 19. Accuracy using IAT(TCP, UDP, and ICMP) outlier graph from verification dataset using CNN.

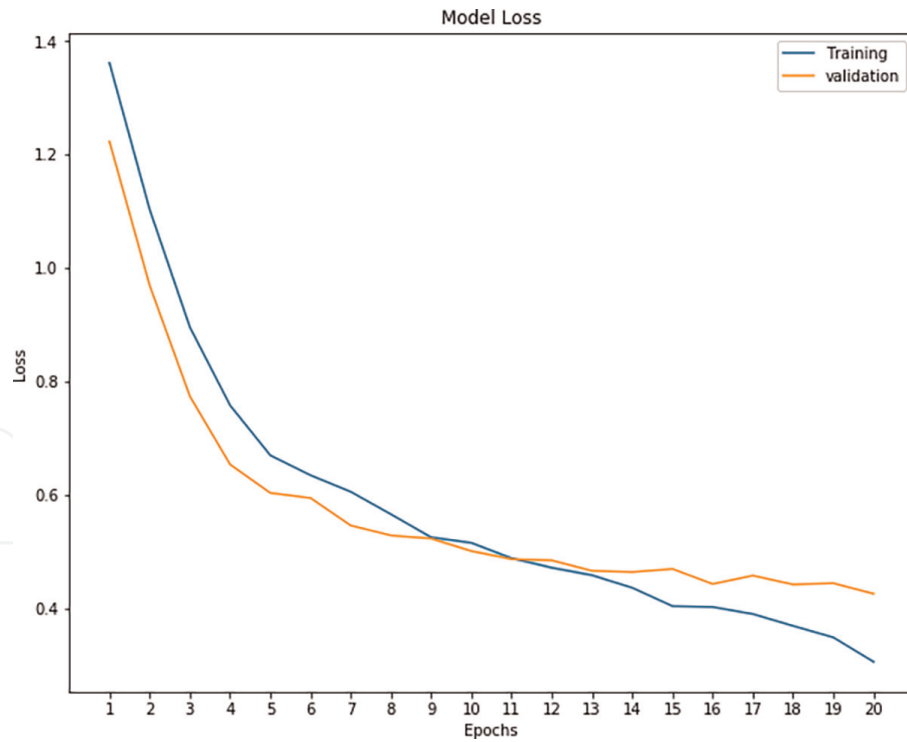


Figure 20. Loss using IAT(TCP, UDP, and ICMP) outlier graph from verification dataset using CNN.

5.1 Comparison of models and parameters for IAT outlier graphs and IAT graphs from verification dataset

The summary of the model and parameters is shown in **Table 2**. When we used IAT graphs, the validation accuracy is 0.97 for CNN, which is better than

Model/Parameters	IAT graphs (ICMP)		IAT Outlier graphs (ICMP)		IAT graphs (TCP, UDP, ICMP)		IAT outlier graphs (TCP, UDP, ICMP)	
	val. Acc	val. Loss	val. Acc	val. Loss	val. Acc	val. Loss	val. Acc	val. Loss
CNN	0.97	0.1326	0.9648	0.1397	0.9290	0.3073	0.8888	0.2704
CNN + LSTM	0.9060	0.5541	0.9574	0.1422	—	—	0.81	0.51

Table 2.
 Performance of models in terms of validation accuracy and validation loss using verification dataset.

CNN + LSTM, in which case the validation accuracy is 0.9060. When we used the IAT outlier graph, the validation accuracy is 0.9648 for CNN and 0.9574 for CNN + LSTM. We observe that classification accuracy is similar in the case of CNN irrespective of the IAT graph or IAT outlier graph used in classification, but in the case of CNN + LSTM, the accuracy is lower, while using IAT graph for classification than IAT outlier graph.

We noticed that the results of the combination of CNN and LSTM cannot outperform the CNN alone model. The first reason is that the input of LSTM is a flattened version of CNN's output instead of a specific time series; therefore, the time dependence captured by LSTM may not reflect the relationship among input images. The second reason is that the used LSTM layer in the experiments has a small output size. In this case, some valuable information may be lost.

6. Conclusion

In this work, we classified devices using two parameters, namely inter-arrival time (IAT) and round-trip time (RTT), and two deep learning algorithms, namely CNN and a combination of CNN and LSTM. We used the IAT and RTT image graph as device fingerprint and model using two deep learning algorithms. We captured the packets using the packet snipping tool at Raspberry Pi(router) for two different setups. IAT and RTT were recorded for each device by snipping tool in real time. The security threat posed by adversaries once they forge the IoT device makes device identification a fundamental problem. The dynamic parameters that we used depend on hardware and software (CPU cache, data cache, and clock frequency, etc.), which makes it harder for intruders to create the fingerprint of a device. We used deep learning to extract the knowledge from data. The widespread recognition of CNN as a good algorithm for image classification encouraged us to use it. Moreover, as LSTM has made its name for the classification of time series data, we used a combination of CNN and LSTM because we were using an image graph of time series data for training the model. Our approach can be used to detect the malicious user if we store the fingerprint and match the fingerprint of the device trying to connect to the network before allowing it to connect. Our approach brings the alternative of using IMEI, IP and MAC address, cryptography security, and a digital certificate for device identification, which are prone to spoofing.

We used two different parameters and obtained good accuracy in our real setup. We also verified our model using the dataset available in public for a single ICMP packet and were able to achieve validation accuracy of 0.97 for CNN and 0.9060 for CNN + LSTM. We compared two deep learning algorithms for device identification.

Both models were good when we used a dataset that was generated from our setup, but while using the dataset from crawdad, CNN was more accurate in classification than CNN + LSTM. We further used IAT outlier graphs for classification and achieved validation accuracy of 0.9648 for CNN and 0.9574 for CNN + LSTM. To validate the improvement in classification accuracy using ICMP packet, we also classified the devices using TCP, UDP, and ICMP packet types from the verification dataset. We achieved good accuracy in using a single ICMP packet type for classification.

We collected RTT data in our setup and achieved good accuracy in classification. In the future, we can collect RTT data in a real scenario with many devices and use it for classification.

Acknowledgements

This work is supported in part by the US National Science Foundation under Grant CC-2018919. Beside NSF grant support, Dr. Yang's work is also supported in part by the new hire startup fund from Southern Illinois University Carbondale.

Conflict of interests

The authors declare that there are no conflicts of interest regarding the publication of this article.

Author details

Prashant Baral^{1†}, Ning Yang² and Ning Weng^{3*}

1 Advanced Micro Devices, Inc., Austin, TX, USA


2 Information Technology Program in the School of Computing, Southern Illinois University Carbondale, IL, USA

3 School of Electrical, Computer, and Biomedical Engineering, Southern Illinois University Carbondale, IL, USA

*Address all correspondence to: nweng@siu.edu

† These authors contributed equally.

IntechOpen

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Xu Q, Zheng R, Saad W, Han Z. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Communications Surveys & Tutorials*. 2015;18(1):94-104
- [2] Neumann C, Heen O, Onno S. An empirical study of passive 802.11 device fingerprinting. In: 2012 32nd International Conference on Distributed Computing Systems Workshops. Macau, China: IEEE; 2012. pp. 593-602
- [3] Bratus S, Cornelius C, Kotz D, Peebles D. Active behavioral fingerprinting of wireless devices. In: Proceedings of the First ACM Conference on Wireless Network Security. New York, NY, USA: ACM; 2008. pp. 56-61
- [4] Uluagac AS. "A. selcuk uluagac, crawdad dataset gatech/fingerprinting (v. 2014-06-09). 2014. Available from: <https://crawdad.org/gatech/fingerprinting/20140609>.
- [5] Uluagac AS, Radhakrishnan SV, Corbett C, Baca A, Beyah R. A passive technique for fingerprinting wireless devices with wired-side observations. In: 2013 IEEE Conference on Communications and Network Security (CNS). Washington, D.C., USA: IEEE; 2013. pp. 305-313
- [6] Hamad SA, Zhang WE, Sheng QZ, Nepal S. 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). Rotorua, New Zealand: IEEE; 2019. pp. 103-111
- [7] Mazhar N, Salleh R, Zeeshan M, Hameed MM. Role of device identification and manufacturer usage description in iot security: A survey. *IEEE Access*. 2021;9:41757-41786
- [8] Aneja S, Aneja N, Islam MS. Iot device fingerprint using deep learning. In: 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS). Bali, Indonesia: IEEE; 2018. pp. 174-179
- [9] Desmond LCC, Yuan CC, Pheng TC, Lee RS. Identifying unique devices through wireless fingerprinting. In: Proceedings of the First ACM Conference on Wireless Network Security. New York, NY, USA: ACM; 2008. pp. 46-55
- [10] Miettinen M, Marchal S, Hafeez I, Asokan N, Sadeghi A-R, Tarkoma S. Iot sentinel: Automated device-type identification for security enforcement in iot. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). Atlanta, USA: IEEE; 2017. pp. 2177-2184
- [11] Robyns P, Bonn e B, Quax P, Lamotte W. Noncooperative 802.11 mac layer fingerprinting and tracking of mobile devices. *Security and Communication Networks*. 2017;2017:1-21
- [12] Kohno T, Broido A, Claffy KC. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*. 2005;2(2):93-108
- [13] Maurice C, Onno S, Neumann C, Heen O, Francillon A. Improving 802.11 fingerprinting of similar devices by cooperative fingerprinting. In: 2013 International Conference on Security and Cryptography (SECRYPT). Reykjavik, Iceland: IEEE; 2013. pp. 1-8

[14] Cunche M. I know your mac address: Targeted tracking of individual using wi-fi. *Journal of Computer Virology and Hacking Techniques*. 2014;**10**(4):219-227

[15] François J, State R, Engel T, Festor O. Enforcing security with behavioral fingerprinting. In: 2011 7th International Conference on Network and Service Management. Paris, France: IEEE; 2011. pp. 1-9

[16] Sun L, Chen S, Zheng Z, Xu L. Mobile device passive localization based on ieee 802.11 probe request frames. *Mobile Information Systems*. 2017;**2017**: 1-10

[17] Kulin M, Fortuna C, De Poorter E, Deschrijver D, Moerman I. Data-driven design of intelligent wireless networks: An overview and tutorial. *Sensors*. 2016; **16**(6):790

IntechOpen