Hindawi
Security and Communication Networks
Volume 2021, Article ID 2713211, 16 pages
https://doi.org/10.1155/2021/2713211

Research Article

Identifying IoT Devices Based on Spatial and Temporal Features from Network Traffic

Feihong Yin [1], Li Yang [1], Jianfeng Ma, [2] Yasheng Zhou, [1] Yuchen Wang, [1] and Jiahao Dai [3]

[1] *School of Computer Science and Technology, Xidian University, Xi'an 710071, China*
[2] *School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[3] *Science and Technology on Communication Information Security Control Laboratory, Jiaxing 314000, China*

Correspondence should be addressed to Li Yang; yangli@xidian.edu.cn

Received 31 August 2021; Revised 13 October 2021; Accepted 29 October 2021; Published 25 November 2021

Academic Editor: Jinguang Han

With the rapid growth of the Internet of Things (IoT) devices, security risks have also arisen. The preidentification of IoT devices connected to the network can help administrators to set corresponding security policies according to the functionality and heterogeneity of the devices. However, the existing methods are based on manually extracted features and prior knowledge to identify the IoT devices, which increases the difficulty of the device identification task and reduces the timeliness. In this paper, we present CBBI, a novel IoT device identification approach. On the one hand, CBBI uses a hybrid neural network model Conv-BiLSTM to automatically learn the representative spatial and temporal features from the network traffic, such as the position relationship of the internal organization structure in network communication traffic, the time sequence of the data packets, and the duration of the network flow. On the other hand, CBBI contains the data augmentation module FGAN that solves the problem of data imbalance in deep learning and improves the accuracy of the model. Finally, we used the public dataset and laboratory dataset to evaluate CBBI from multiple dimensions. The evaluation results for different datasets show that our approach achieves the accurate identification of IoT devices.

1. Introduction

With the rapid development of IoT technology, the types and numbers of IoT devices have been growing quickly. The powerful connectivity and convenience of the IoT devices make their applications increasingly widespread, and they penetrate almost every corner of life, including smart wear, smart homes, smart entertainment, and smart travel. According to [1], approximately 31 billion IoT devices were used globally by the end of 2020, and approximately 75 billion IoT devices will be used by 2025, of which smart homes [2] will account for 41%, reaching 12.86 billion.

However, many vulnerabilities exist in current IoT devices and execution environments [3–5]. Attackers are increasingly concentrating on these vulnerable IoT devices, using device vulnerabilities to launch attacks [6–11]. For example, malicious attackers used a cloud of vulnerable IoT devices to build a sizable and highly destructive botnet to launch large-scale DDoS attacks. In the first 1 TB DDoS attack conducted on the Krebs Security website, more than 400,000 IoT devices were utilized [3]. Additionally, attackers can use these vulnerable IoT devices as proxies for malicious activities, further deteriorating the network environment. With the proliferation of IoT devices with security flaws, IoT device-centric attacks could further increase.

From the defensive aspect, network administrators must implement network access control on all connected devices. Whenever a new device is connected to the network, and if the device can be identified, then the network administrator can take appropriate security precautions. For example, the administrator configures the corresponding firewall rules according to the security requirements of the device, verifies whether the device has known vulnerabilities, or notifies the intrusion detection system to isolate vulnerable devices.

Many of the current studies on IoT device identification concentrate on features based on statistics and manual extraction and then combine fingerprint matching, machine learning, or deep learning to recognize IoT devices. These traditional IoT device identification methods face the following problems: (1) extracting features manually is a tedious and time-consuming process, and the low efficiency of feature extraction will affect the real-time performance of the classification model. (2) Feature extraction requires a professional domain prior knowledge and even professional feature engineering. Feature engineering involves feature extraction, feature construction, and feature selection, which, undoubtedly, further increases the difficulty of feature extraction. (3) The generalization ability of the model is also a concern. Before the training process in the traditional method, some seemingly trivial features might be discarded. These features could help the model improve its generalization ability, which would enable the model to be extended to more IoT devices. The feature space generated by traditional feature engineering is relatively small, and it is difficult to extract these subtle features. With the increase in the number and diversity of devices in practical applications, the recognition accuracy of the model could drop sharply. (4) A surging number of IoT devices use encryption protocols, increasing the difficulty of feature extraction and device identification. (5) With the increasing number of IoT devices, the amount of data generated by them is also increasing, and hence, feature extraction requires more time and resources. Therefore, it is difficult to meet the current needs using traditional features based on manual extraction.

In this paper, we present CBBI, a novel IoT device identification approach based on Conv-BiLSTM to learn the spatial and temporal features of the network traffic. CBBI contains three modules. The first module is the data preprocessing module, whose main task is to quickly process the raw network traffic generated by the IoT device and convert it into an input that can be used in the deep learning model. The second module is the data augmentation module FGAN that solves the problem of data imbalance in deep learning methods [12, 13]. The third module is to establish a deep learning model. We designed a hybrid deep learning model Conv-BiLSTM. Convolutional neural networks (CNNs) can learn the spatial characteristics of network communication traffic, such as the positional relationship of internal organizational structures in the network communication traffic. A bidirectional long short-term memory network (BiLSTM) can extract the time-domain characteristics of the network communication traffic, especially the timing relationship and flow duration of the data packets. The accuracy and generalization ability of the model are further improved by learning the spatial and temporal features simultaneously. Even when confronted with the IoT devices that have similar functions produced by the same device manufacturer, CBBI can use the powerful feature learning capabilities of deep learning to extract the representative features and some potential subtle features from the original traffic, and finally, it can realize the accurate identification of the IoT devices based on these learned features. This paper is an extended version of [4]. Firstly, compared with [4], we added a data augmentation module FGAN to solve the data imbalance. At the same time, the corresponding comparative test was also added. Secondly, we added our own dataset to the experiment and made some experimental evaluations based on the dataset. Finally, we added an additional visualization part to prove that CBBI can learn the representative spatial and temporal characteristics from the device communication traffic.

We summarize the main contributions as follows:

(1) We propose a novel IoT device identification method, CBBI. This method does not require any prior knowledge about feature engineering. It avoids the overhead of manual feature extraction and decreases the complexity of IoT device identification tasks.

(2) CBBI extracts the spatial and temporal features from the original traffic generated by the device, including some potential subtle features to identify the IoT devices, increasing the generalization ability of the model.

(3) CBBI contains the data augmentation module FGAN that solves the problem of data imbalance in deep learning and effectively improves the accuracy of the model.

(4) We conduct extensive experiments on the public dataset and laboratory dataset to evaluate the performance of CBBI. The results show the superiority of the proposed model.

The remainder of this paper is organized as follows. Section 2 summarizes the related work on IoT device classification. Section 3 describes our proposed IoT device identification method, which includes data preprocessing, data augmentation, and Conv-BiLSTM. Section 4 describes the experiment setup. Section 5 presents the evaluation results and analyses. Finally, we conclude this work in Section 6.

2. Related Work

For the identification of IoT devices, researchers have proposed many solutions. In this paper, the existing research studies are summarized and discussed from two aspects: device identification technology based on a classification model and device identification technology based on active detection.

2.1. Device Identification Technology Based on a Classification Model. Because of the differences in the software and hardware used in the IoT devices, there will also be subtle differences among the different devices produced by the same manufacturer. Researchers use the subtle differences in the hardware of the device, such as the clock offset [14–17], as the fingerprint of the device. Then, they construct a classification model to realize the accurate identification of the target device. In the traditional method, wireless devices can be identified by some unique radio frequency (RF) fingerprints caused by radio circuits [18, 19]. Yuan et al. [20] fingerprinted wireless devices by extracting the features caused by the hardware defects in the analog circuits. An important advantage of using these physical defects as device signatures for device identification is that it is difficult to use other wireless devices to spoof the signature. Brik et al. [21] designed and implemented a technology that uses the passive radio frequency analysis to identify the source network interface card (NIC) of IEEE 802.11 frames.

Radhakrishnan et al. [22] used the arrival interval time of packets in specific traffic types generated by the devices as the feature vectors. They used these feature vectors to train the artificial neural network (ANN). Miettinen et al. [23] proposed IoT Sentinel, which is a system for the automatic recognition of IoT devices. The system extracts 23 features from the data packets as device fingerprints and identifies the devices using a two-step classification method.

Guo and Heidemann proposed a method that analyses the DNS traffic to detect the IoT devices and identify their type [24]. Marchal et al. [25] automatically identified the type of IoT devices in the local network based on the periodic background network traffic of the IoT devices. The method needed 30 minutes to identify the type of devices, and the accuracy rate reached 98.2%.

Thangavelu et al. [26] used controllers to control the gateways based on a software-defined network (SDN). The controller implements the training and updating of the model and sends the newly trained model to each gateway. Sivanathan et al. [27] used statistical attributes such as the device traffic activity cycle, port number, signaling mode, and cipher suite as fingerprint features of the device. Then, they used a multistage machine learning classification algorithm to identify the device.

WDMTI [28] uses 18 features extracted from DHCP messages to establish a hierarchical Dirichlet process (HDP) model to identify the wireless devices. This method relies on the bursts of traffic when the device is connected to the network. OWL [29] analyzed the broadcast and multicast packets in the wireless local area networks (WLANs), built a multiview deep learning (MvWDL) model based on the features extracted from each protocol message, and classified the IoT devices.

According to the unique network traffic pattern of the IoT devices, Deng et al. [30], firstly, extracted all available features from each TCP flow header. Secondly, they used the principal component analysis (PCA) algorithm to select the main features that affect device recognition. Finally, they learned the device-specific network traffic signature based on a random forest classifier to achieve device identification. Yin et al. [4] proposed an end-to-end IoT device identification method that directly uses the original communication traffic generated by the device. This method fails to fully consider the problem of data imbalance. In the face of extremely unbalanced datasets, the performance of the model may be greatly compromised.

2.2. Device Identification Technology Based on Active Detection. Active detection refers to actively sending detection packets to the devices in the network, obtaining response packets, and extracting device information by analyzing the information in the response packets. Attackers usually obtain information about vulnerable devices in the network through active detection before launching an attack to improve the accuracy of the attack. Researchers also use the active detection method to determine the state of the devices in the network, so they can take further security measures to ensure the safety of the devices in the network.

In practice, because of the large number of IoT devices and the lack of training data, researchers use banner information instead of device fingerprints to identify the IoT devices. Antonakakis et al. [31] applied the banner rules to analyze the online devices from Censys [32] and Honeypot. Shodan [33] and Censys [32] are the two popular search engines that are mainly used to discover online devices. Both search engines use different protocols (such as HTTP, SSH, FTP, and TELNET) to perform Internet-wide scans.

Many researchers [34, 35] use banner information acquisition to actively scan the devices in the IP space. They collect and check the text features from the response, such as hard-coded keywords, and match them with known fingerprints for device identification.

Li et al. [36] established a framework for searching devices on the Internet using network measurement and banner grabbing to obtain services running on the network hosts and to match the response header fields with prestored keywords to retrieve device information.

Feng et al. [37] proposed an acquisition rule-based engine (ARE) that can automatically generate rules for discovering and annotating the IoT devices without any training data. ARE uses the application layer response data from the IoT devices and product descriptions in the related websites to obtain device comments, thereby constructing device rules. It solves the cumbersome and incomplete shortcomings of traditional methods based on manually writing banner information capture rules.

Table 1 summarizes the main references aforementioned and shows the features and methods used in the relevant references.

The aforementioned research works made important contributions to the identification of IoT devices and promoted the development of network security. The device fingerprint identification method based on the classification model mainly uses the physical difference of the device as the fingerprint of the device. Otherwise, it manually extracts some field values and related statistical characteristics in the device communication traffic. Then, it is combined with machine learning or deep learning methods to construct a classification model. The device identification method based on active detection must actively send a multitude of detection packets to the target devices in the network, which is susceptible to packet loss and network delay. In addition, the frequent transmission of probe packets will increase the load on the network and aggravate the deterioration of the network environment. More importantly, if the device does not generate a response or if there is no valid information in the response packets, the device cannot be further identified.

Table 1: Summary of related works.

References	Features	Method
[14–17]	Clock skew	—
[18–21]	Radio frequency fingerprint	—
[22]	Clock skew	ANNs
[23]	Features from the packet head	Twofold identification technique (Random Forest + Edit Distance)
[24]	Flow-level network traffic and knowledge of servers run by the manufacturers	—
[25]	Periodic communication traffic features	KNN
[26]	Features from DNS queries and HTTP URI’s	Improved k-means algorithm, Random Forest, SDN
[27]	Statistical attributes such as activity cycles, port numbers, signaling patterns, and cipher suites	A multistage machine learning (Naive Bayes + Random Forest)
[28]	18 features of DHCP	Dirichlet process
[29]	Features from passively received broadcast and multicast packets	Multiview wide and deep learning framework
[30]	Features in TCP header per TCP flow	PCA, Random Forest
[4]	Raw network traffic from devices	CNN, BiLSTM
[31]	Banners, honeypots	Active scanning
[34–36]	Banners	Active scanning, match
[37]	Banners	Active scanning, search and match

3. Proposed Framework

The overall structure of the CBBI framework is shown in Figure 1. CBBI is composed of three modules: data preprocessing, data augmentation, and Conv-BiLSTM. Initially, the data preprocessing module converts the raw network traffic generated by the IoT device into an input that can be used in the deep learning model. Furthermore, the data augmentation module FGAN solves the problem of data imbalance in deep learning. Finally, the Conv-BiLSTM module simultaneously learns the spatial and temporal characteristics of the original traffic of the device, which improves the accuracy and generalization ability of the model.

3.1. Data Preprocessing. In general, deep learning models cannot directly use the raw pcap data. These original pcap files need to be processed into a format suitable for model input. The entire data preprocessing process includes three parts: flow generation, irrelevant field removal, and traffic vectorization.

3.1.1. Flow Generation. The original communication traffic generated by the IoT devices contains different numbers of data packets, and the length of each data packet is also inconsistent. In other words, the original communication traffic generated by the devices can be defined as $P = \{p_1, \ldots, p_n\}$, and each data packet can be defined as $p_i = (x_i, s_i, t_i)$. The value of i is $i = 1, 2, \ldots, |n|$, where x_i represents the 5-tuple information (source IP address, source port number, destination IP address, destination port number, and transport layer protocol type) of the packet, s_i represents the size of packet p_i, and t_i represents the starting time of packet p_i.

In this paper, the existing Splitcap tool [38] is utilized to process the original network traffic into a network flow with the same 5-tuple information, where the network flow can be defined as $F_i = \{p_1 = (x_1, s_1, t_1), p_2 = (x_2, s_2, t_2), \ldots, p_m = (x_m, s_m, t_m)\}$. Here, m represents the number of data packets in the network flow. As the data packets in the network flow have the same 5-tuple information, $x_1 = x_2 = \cdots = x_m$. The network flow has a certain time order, and thus, the data packets in the network flow have a sequence, represented by $t_1 < t_2 < \cdots < t_m$.

Each network flow is composed of several packets. The network flow contains substantial behavior characteristics of IoT device communication traffic, including the closeness of the relationship among the bytes in the data packet, the duration of each network flow, the number and size of the data packets that constitute the network flow, and the timing relationships among the data packets. These traffic behavior characteristics can help the deep learning model to better recognize the device and improve the accuracy of the model.

3.1.2. Irrelevant Fields Removal. CBBI makes use of the traffic behavior characteristics of IoT devices. Here, we need to eliminate some interference data, such as MAC addresses and IP addresses, to prevent these data from affecting the experimental results. In a small LAN, the number of devices is limited, and the MAC addresses of the devices can uniquely identify the devices. These field values can occupy a relatively large weight in the process of the feature extraction of the deep learning model, which could affect the real recognition and classification ability of the model. It can even lead to the overfitting of the model. The IP address of the device has the same interference effect as the MAC address. In this paper, these interference fields are eliminated in the data processing module to prevent them from affecting the process of model feature learning.

3.1.3. Traffic Vectorization. The neural network requires the input data to have a standardized format, and we must convert the processed data aforementioned into a suitable input format. The number of data packets in each network

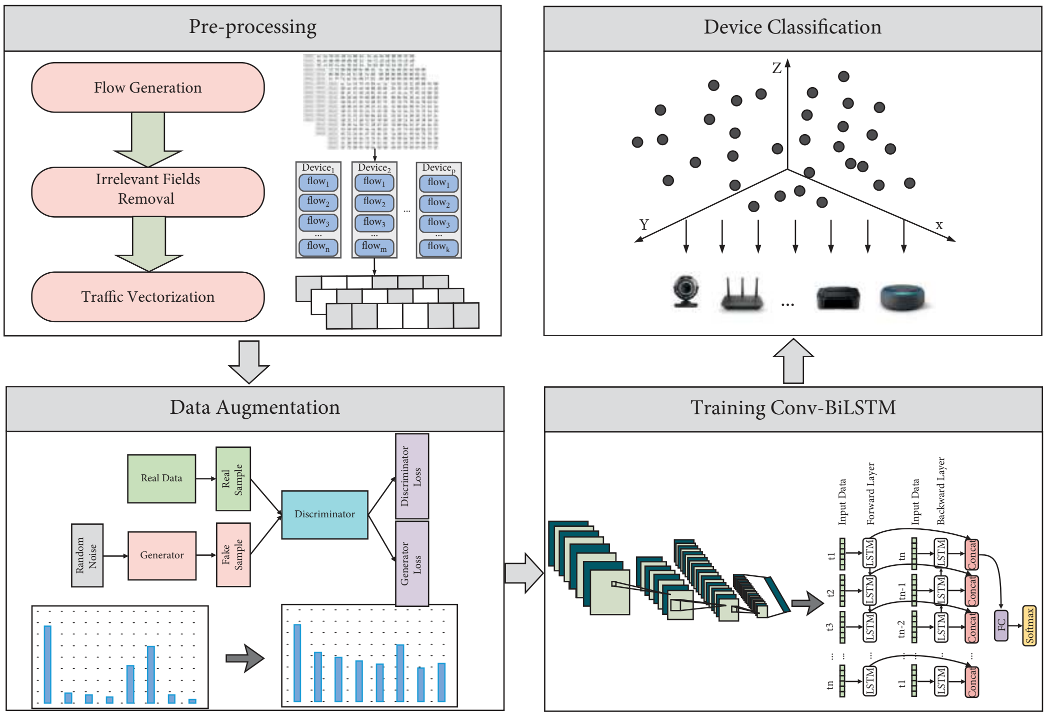


Figure 1: An overview of the CBBI framework.

flow and the size of each data packet are different, and thus, a unified standard must be determined to vectorize the features in the network flow. We performed a statistical analysis based on the public dataset and laboratory dataset. As shown in Figure 2, we found that the number of data packets in the network flows in the two datasets is mostly within 10, and most of the data packets are within 250 bytes in size. According to the statistical information, each network flow intercepts 2500 bytes of data samples. In other words, each network flow selects the first 10 packets ($n = 10$), and each packet intercepts the first 250 bytes ($L = 250$ bytes). If the number of data packets N in the network flow is less than 10, or the length of the data packet L is less than 250 bytes, then it is directly filled with 0. The representation of network flow characteristics is shown in Figure 3. The complete data preprocessing algorithm is shown in Algorithm 1.

3.2. Data Augmentation. The network traffic generated by the IoT devices can be transformed into different numbers of data samples after the preprocessing stage. Because of the different functions, the software, and the hardware of the devices, the traffic model generated by each device is very different. For example, the network traffic generated by the video monitoring devices is very large, whereas the network traffic generated by some sensors is relatively limited. Therefore, there is a large difference in the number of samples that correspond to the device, which leads to the serious problem of data imbalance. As far as the learning model is concerned, the sample is usually considered to be an unbiased sample of the true distribution. When the training set is largely skewed, it usually does not reflect the true distribution. The imbalance of the sample distribution causes the model prediction result to be biased; in other words, the classification result is biased toward more sample categories, and the result is misleading. Therefore, we must adjust the generated sample data to alleviate the imbalance and further improve the performance of the model.

This paper uses the GAN-based data augmentation module FGAN, as shown in Figure 4. The generative adversarial network (GAN) is an adversarial network proposed by Goodfellow [39] in 2014. The network framework consists of two parts, a generator and a discriminator. The generator tries to cheat the discriminator by constructing false data. It accepts arbitrary noise $p_z(z)$ and generates false data according to the noise, which is recorded as $G(z)$. The discriminator tries to distinguish whether the data came from a real sample or fake data constructed by a forger. The input parameter of the discriminator is x, which comes from $p_{\text{data}}(x)$. The output $D(x)$ of the discriminator represents the probability that x is the real data. Both models improve their abilities using continuous learning. In other words, the generator hopes to generate more real fake data to cheat the discriminator, and the discriminator hopes to learn how to more accurately identify the fake data of the generator. The objective function v of FGAN is as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_z(z)}[\log(1 - D(G(v)))]. \tag{1}$$

The basic flow of the network is as follows:

(i) Initialize parameter θ_d of discriminator D and parameter θ_g of generator G.

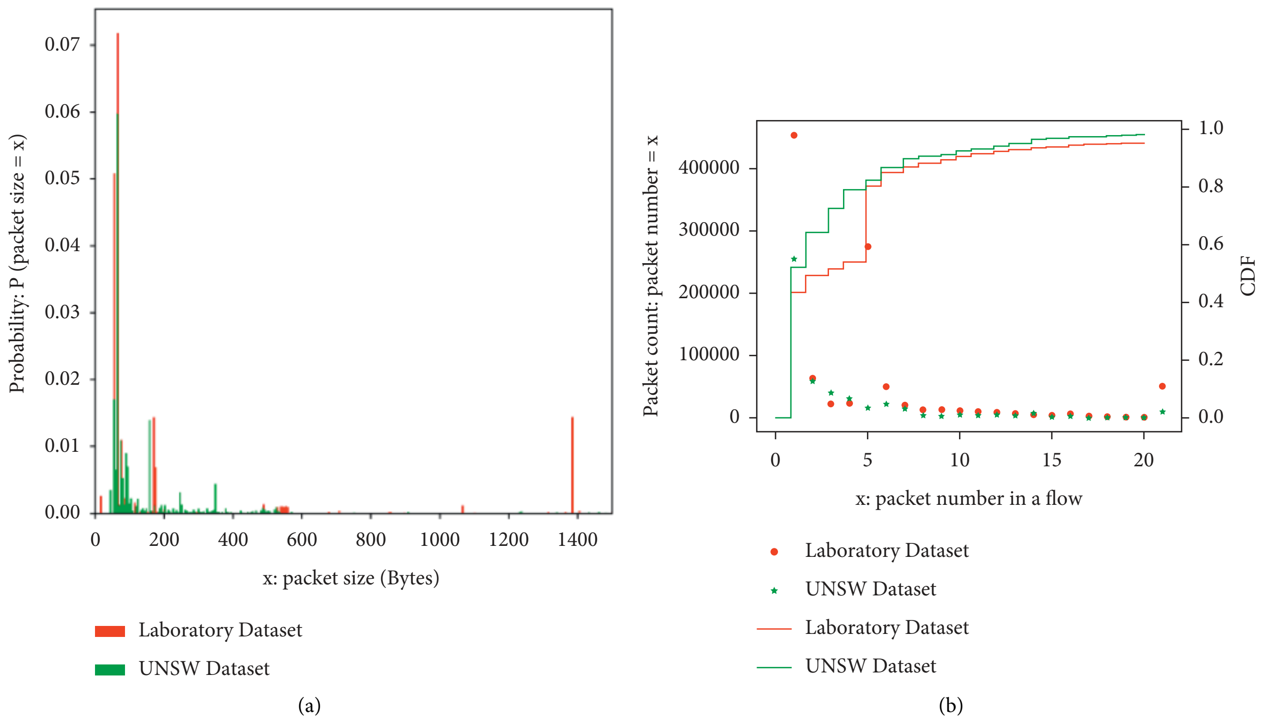


Figure 2: Statistics of the network traffic in the two datasets. (a) Statistics of the packet size. (b) Statistics of the number of data packets in the network traffic flow.

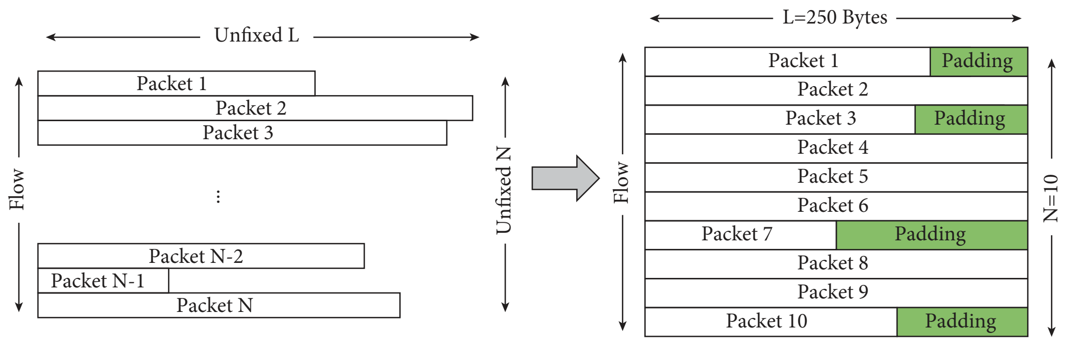


Figure 3: Representation of the network traffic flow.

(ii) Firstly, n sample data $\{x^1, x^2, \ldots, x^n\}$ are obtained from the real samples. Then, n noise samples $\{z^1, z^2, \ldots, z^n\}$ are sampled from the prior distribution noise. Secondly, n samples $\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^n\}$ are produced using the generator. Finally, the generator G is fixed, and the discriminator D is trained to identify the real data from the generated data as accurately as possible.

(iii) After updating the discriminator for k epochs, the parameters of the generator are updated once with a small learning rate, and the generator is trained to reduce the gap between the generated data and the real data as much as possible.

(iv) After many iterations of updates, the final ideal is for the discriminator to be unable to tell whether the sample comes from the output of the generator or the real output.

This paper designs the generator and discriminator in FGAN based on the fully connected network. Detailed information will be in Section 4.3.

3.3. *Training Conv-BiLSTM for IoT Device Classification.*

In this section, we build a deep learning model Conv-BiLSTM for identifying the IoT devices. This model is different from the traditional classification methods based on manually extracted features and statistical features. Firstly, the model can simultaneously learn the spatial and temporal features of the device traffic, which improves the accuracy of device identification. In addition, the traditional device identification method based on manual design and statistical features has some limitations. When these features are designed and selected artificially, the inherent features in the original communication flow of the device are changed, and some potential features are ignored. These potential features can

```
     Input: raw network traffic pcap files
     Output: Samples_Data
     Samples_Data $\longleftarrow \varnothing$
     Use SplitCap tool to convert raw pcap files into Flows with the same 5-tuple
     information, $F1$, $F2$, ..., $Fm$, $m$ represents the last Flow in raw pcap
(1)  for each $Fi$ do
(2)    Packets_Number$\leftarrow$Len ($Fi$), Flow_Feature $\longleftarrow \varnothing$
(3)    if Packets_Number $> =$ 10 then
(4)      Packets_Sequence $\longleftarrow$ Get the top 10 packets in $Fi$
(5)      for each $P$ in Packets_Sequence do
(6)        if length ($P$) $> =$ 250 bytes then
(7)          Packet_Feature $\longleftarrow$ Get the first 250 bytes in $P$
(8)        else
(9)          Packet_Feature $\longleftarrow$ Get all bytes in $P$ + ″0″ $*$ (250 − length ($P$))
(10)         Set the MAC address field and IP address field in Packet_Feature to 0
(11)         Flow_Feature $\leftarrow$Flow_Feature $\cup$ Packet_Feature
(12)     end for
(13)   else
(14)     Packets_Sequence $\longleftarrow$ Get all packets in $Fi$
(15)     for each $P$ in Packets_Sequence do
(16)       if length ($P$) $> =$ 250 bytes then
(17)         Packet_Feature $\longleftarrow$ Get the first 250 bytes in $P$
(18)       else
(19)         Packet_Feature $\longleftarrow$ Get all bytes in $P$ + ″0″ $*$ (250 − length ($P$))
(20)       Set the MAC address field and IP address field in Packet_Feature to 0
(21)       Flow_Feature $\longleftarrow$Flow_Feature $\cup$ Packet_Feature
(22)     end for
(23)     for $j = 1$; $j < = 10 -$ Packets_Number; $j + +$ do
(24)       Flow_Feature $\longleftarrow$Flow_Feature $\cup$ (″0″ $*$ 250)
(25)     Samples_Data.append(Flow_Feature)
(26) end for
(27) Return Samples_Data
```

Algorithm 1: The algorithm for data preprocessing.

help to improve the recognition accuracy and generalization ability of the model. In addition, artificially designed features might not fully represent the high-level semantics of the network traffic, and models trained based on these features cannot learn these high-level semantics. The Conv-BiLSTM network model can learn highly semantic features from the original communication traffic generated by the device.

The CNN [40] is widely used in the field of image classification because of its influential spatial feature learning ability. The CNN has a convolutional layer, pooling layer, and fully connected layer. The main function of the convolutional layer is to extract features. The pooling layer implements data subsampling without destroying the classification results in terms of reducing the dimensionality of the features, compressing the data, and avoiding the overfitting of parameters. The convolutional layer and the pooling layer play the role of mapping the original data to the hidden layer feature space. The fully connected layer is a fully connected neural network. The weight parameters are adjusted by weighing the proportion of each neuron's feedback. The model also uses dropout to avoid overfitting.

LSTM is a special recurrent neural network (RNN) [41]. The difference between LSTM and the standard recurrent neural network is that the LSTM overcomes the problems of gradient explosion and gradient disappearance by introducing memory units and gate mechanisms, and it performs well in extracting the long-term dependence in the sequence data. The LSTM architecture is composed of an input gate, forget gate, output gate, storage unit, hidden state, and so on. The specific calculation process of the input gate, output gate, and forget gate is as follows.

3.3.1. Forget Gate. f_t is called the forget gate, which indicates that some features of c_{t-1} are used to calculate c_t. f_t is obtained by a logical function to calculate the input x_t and the last hidden layer value h_{t-1}. The value of the forget parameter is between 0 and 1, which controls how much information is retained from c_{t-1} to c_t. Here, 1 means to retain the information completely, whereas 0 means to discard the information entirely.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right). \tag{2}$$

3.3.2. Input Gate. The input gate decides what new information to store in the "cell state." The sigmoid layer decides what value is to be updated. The tanh layer creates a new

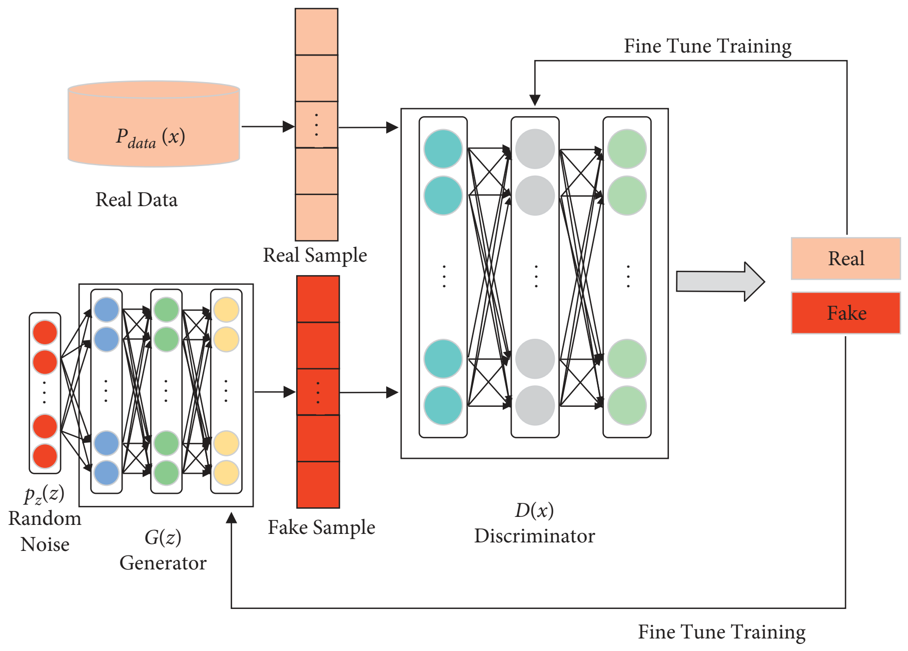


Figure 4: The framework of FGAN.

candidate value vector $\tilde{c}_t$, and i_t determines the part of the information to be updated. When updating c_{t-1} to $\tilde{c}_t$, we must multiply the old state with f_t, discard the information that needs to be discarded, and add $i_t * \tilde{c}_t$.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \\ \tilde{c}_t &= \tan h\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right), \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t. \end{aligned} \tag{3}$$

3.3.3. Output Gate. h_t can be considered the last output at the current moment. h_{t-1} is the output at $t-1$. o_t is a probability vector that is used to determine which part is the output. Firstly, we run a sigmoid layer to determine which part of the cell state is to be the output. Then, tanh is used to process the cell state (obtaining a value between −1 and 1). Finally, this value is multiplied with the output of the sigmoid gate to obtain the output.

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right), \\ h_t &= o_t * \tan h\left(c_t\right). \end{aligned} \tag{4}$$

Unlike other types of deep neural networks, LSTM shares weights at all time steps, which reduces the number of parameters that the network must learn. The BiLSTM [42] is composed of two LSTMs: one LSTM is the input forward, whereas the other LSTM is the output backward. BiLSTM effectively increases the amount of information available to the network and improves the context available for the algorithm. BiLSTM can not only address gradient disappearance and gradient explosion, as in the LSTM, but also learn more context information from the network.

The Conv-BiLSTM network model structure is shown in Figure 5.

The convolutional neural network model used in this paper is improved on the basis of the classical lenet-5 [43]. The convolutional neural network constructed in this paper has seven layers. More detailed network structure information is provided in Section 4.3. The training process of the Conv-BiLSTM model is shown in Algorithm 2.

The feature dimension of each sample after CNN is 1600. We reshape the 1600-dimensional data into a $10 * 160$ format and input it into BiLSTM, where 10 represents the number of time steps. The vector dimension of each time point is 160. The BiLSTM consists of two layers, each with 512 hidden cells, and each layer uses the sigmoid function for nonlinear operations. The last layer of the BiLSTM network adopts the fully connected layer, and the number of neurons in the fully connected layer is equal to the number of IoT devices. Softmax is used as the activation function, which maps the output of multiple neurons to (0, 1), and the sum of each output is 1. The type with the largest probability value can be selected for multiple classifications.

4. Evaluation Setup

4.1. Computing Platform Configurations. We use Keras [44] as the neural network framework to construct the Conv-BiLSTM model. The detailed configuration information is shown in Table 2.

4.2. Dataset Description. The UNSW dataset is the traffic data generated by the IoT devices in two weeks. The dataset contains a total of 22 IoT devices. Some of these devices generate very little communication traffic. For example, Withings_Smart_Scale and Blipcare_Blood_Pressure_meter generated 8 and 13 sample data, respectively, after data preprocessing. In the experimental part, we selected 18 IoT devices with relatively large sample sizes.

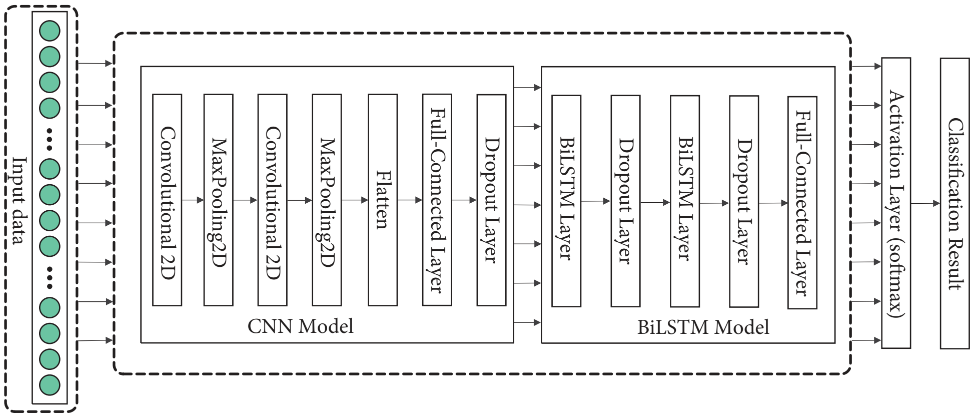


Figure 5: The architecture of Conv-BiLSTM.

```
Input:
    Samples_Data composed of network flows, the dimension of each network flow is 2500.
    {Epoch, Batchsize, dropout, Loss_function} represent some of the parameters during model training.
Output:
    The categories of Samples_Data
 (1)  for each epoch in (1, Epoch) do
 (2)    for each Batchsize data of the Samples_Data do
 (3)      for each Sample_Data in batch do
 (4)        Reshape Sample_Data to 50 * 50 form
 (5)        Compute convolution with 6 filters
 (6)        Compute the result through Relu
 (7)        Max Pooling
 (8)        Compute convolution with 16 filters
 (9)        Compute the result through Relu
(10)        Max Pooling
(11)        Flatten the data
(12)        Run through a densely connected layer
(13)        Dropout
(14)        Reshape output data as 10 * 160
(15)        Run through the 2-layered BiLSTM with dropout
(16)        Run through a densely connected layer
(17)        Output the result referring Loss_function
(18)        Update the parameters of weight and bias
(19)      end for
(20)    end for
(21)  end for
```

Algorithm 2: Training process of the Conv-BiLSTM model.

We built an IoT device traffic collection platform in the laboratory environment. We collected the two-week communication traffic of 23 IoT devices, covering a variety of device types and device brands, including 360, Amazon, Hikvision, Huawei, TP-Link, Xiaomi, and other common IoT device manufacturers. The device types include smart cameras, smart speakers, smart gateways, smart doorbells, and so on. Also, there are IoT devices of the same brand and type, but of different models, such as the two cameras Hikvision_DS-IPC-E22H-IW and Hikvision_DS-IPC-S12P-IWT from Hikvision and the three smart cameras TP_Link_Camera_IPC42A-4, TP_Link_Camera_IPC43AN-4,and TP_Link_Camera_IPC64C-4 from TP-Link. The data traffic generated by 23 IoT devices was processed to generate a total of 636,789 sample data. Detailed information on the UNSW dataset and the laboratory dataset is shown in Table 3.

As shown in Table 3, the number of samples of cameras in the two datasets is relatively vast. The traffic data generated by some cameras in the laboratory dataset is not very large, such as D-Link-DSH-C310, Hikvision_DS-IPC-E22H-IW, and Hikvision_DS-IPC-S12P-IWT. We checked the settings of these devices and found that they adopted the “Standard Definition” video recording method rather than the “High Definition” or “Super Definition” as the other cameras did. Some cameras also

Table 2: Experiment settings.

Category	Parameter
Operating system	Ubuntu 16.04 LTS OS
CPU	Intel(R) Xeon(R) Bronze 3106 CPU@ 1.70GZ
Deep learning platform	Keras
Deep learning backend	TensorFlow-gpu 2.2.0
GPU version	NVIDIA GP104GL(Quadro P4000)
CUDA version	10.1.243
CuDNN version	7.6.5

enabled face recognition, automatic tracking, and other modes, and hence, the communication traffic generated was enormous.

4.3. Parameter Settings. This section provides detailed information about the FGAN and Conv-BiLSTM network structures used in the experiment. Both generator and discriminator in FGAN are implemented based on a multilayer perceptron (MLP). The specific information is shown in Tables 4 and 5. The input of the generator is a 100-dimensional Gaussian noise vector, and the hidden layer contains 256, 512, 1024, and 2500 neurons. The input of the discriminator contains both real data and generated data, and its dimension is 2500. The LeakyReLU activation function, dropout, and BatchNormalization are used in FGAN to optimize the model.

Detailed information on Conv-BiLSTM is shown in Table 6, including the structural parameters of each layer of the network, the optimizer, loss function, and other hyperparameters.

4.4. Evaluation Metrics. To evaluate the performance of the neural network model, this paper selects four performance metrics: the recall, precision, accuracy, and F1-score:

$$\begin{aligned}\text{recall} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\\ \text{precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},\\ \text{accuracy} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}},\\ \mathrm{F1}-\text{score} &= \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}},\end{aligned} \tag{5}$$

where TP, TN, FP, and FNdenote the true positives, true negatives, false positives, and false negatives, respectively.

5. Experimental Results and Analyses

5.1. Ablation Study. To verify the effectiveness and rationality of the data augmentation module FGAN in CBBI, we performed the corresponding experiments on the UNSW dataset and the laboratory dataset. We used the precision, recall, and F1-score to evaluate the results of the experiment. Tables 7 and 8 show the experimental results on CBBI, including FGAN, on the UNSW dataset and the laboratory dataset, respectively. From the two tables, it can be seen that FGAN in CBBI has well alleviated the problem that the classification results of a small number of samples are biased toward large sample classes due to data imbalance. The small sample classes iHome, Nest_Dropcam, NEST_Procet_Smoke_Alarm, and Triby_Speaker in the UNSW dataset and D-Link-DSH-C310, Huawei_Smart_Scale, Huawei_Smart_Scale, Xiaomi_Air_Purifier, and Xiaomi_Hub in the laboratory dataset have significantly improved the performance after using FGAN. The performance of other categories has also been improved to varying degrees as the samples become more balanced. The data augmentation FGAN module in CBBI realizes the relative balance of the sample and further improves the classification accuracy of the model.

5.2. Misclassification Analysis. To analyze the misclassification of the CBBI model in the two datasets, we give the confusion matrixes of the experimental results in the two datasets, as shown in Figure 6. The classification accuracy of most of the devices in the UNSW dataset is close to 100%. The accuracy of Nest_Dropcam is 96%, and 4% of its data samples are identified as Netatmo_Welcome. These two devices are products of two different device manufacturers, however, both belong to the smart camera type, and there are certain similarities in the traffic model.

The accuracy of CBBI in the laboratory dataset reached 97.26%, which is not as high as that in the UNSW public dataset. We can determine the following reasons by analyzing the experimental data and the results: (1) the laboratory dataset contains more IoT devices than the UNSW dataset, which increases the difficulty of multiclassification of the model; (2) the number of samples generated by the devices in the laboratory dataset is more unbalanced, which affects the fitting effect of the model; (3) there are more devices from the same manufacturer and type in the laboratory dataset, and there are more similarities between the devices; the confusion matrix shows that IoT devices of the same device manufacturer and type are prone to misclassification between one another. Hikvision_DS-IPC-E22H-IW and Hikvision_DS-IPC-S12P-IWT from Hikvision, Ezviz_Camera_CS-C6CN and Ezviz_Door_CS-DB2C from Ezviz, as well as four devices from TP-Link, have all been misclassified to varying degrees. The worst classification effect in the laboratory dataset is TP_Link_WDA6332RE and Xiaomi_Air_Purifier. The sample size of these two devices is extremely small, and FGAN has improved the classification accuracy of these two devices to a certain extent.

5.3. Visualization of Spatial and Temporal Features. In this section, we input the spatial and temporal feature vectors learned by CBBI from the UNSW dataset and laboratory dataset into the t-SNE algorithm before applying softmax

Table 3: Description of the UNSW dataset and laboratory dataset.

UNSW dataset			Laboratory dataset		
No.	Device name	Sample number	No.	Device name	Sample number
0	Amazon_Echo	73780	0	360_Camera	5950
1	Belkin_Wemo_Switch	17148	1	Amazon_Echo	9584
2	HP_Printer	2794	2	D-Link-DSH-C310	836
3	Insteon_Camera	216088	3	Hikvision_DS-IPC-E22H-IW	2082
4	Light_Bulbs_LiFX_Smart_Bulb	7226	4	Hikvision_DS-IPC-S12P-IWT	2149
5	Netatmo_Weather_Station	4703	5	Ezviz_Camera_CS-C6CN	65366
6	Netatmo_Welcome	16000	6	Ezviz_Door_CS-DB2C	88273
7	PIX_STAR_Photo_Frame	12311	7	Huawei_Camera_HQ5	35742
8	Samsung_SmartCam	61815	8	Huawei_Speaker	50183
9	Smart_Things	10367	9	Huawei_Smart_Scale	378
10	TP_Link_Day_Night_Cloud_Camera	4954	10	Imou-Camera_TP1-2525	19534
11	TP_Link_Smart_Plug	2586	11	Imou-Camera_TP7C-E152	15110
12	Withings_Aura_Smart_Sleep_Sensor	13963	12	Philips_Hue	103687
13	Withings_Smart_Baby_Monitor	20229	13	TP_Link_Camera_IPC42A-4	79354
14	iHome	346	14	TP_Link_Camera_IPC43AN-4	61061
15	Nest_Dropcam	154	15	TP_Link_Camera_IPC64C-4	61815
16	Nest_Procet_Smoke_Alarm	221	16	TP_Link_WDA6332RE	221
17	Triby_Speaker	669	17	Xiaomi_Air_Purifier	133
—	—	—	18	Xiaomi_Camera_MJSXJ06CM	22120
—	—	—	19	Xiaomi_Door_LSC_M01	4203
—	—	—	20	Xiaomi_Hub	242
—	—	—	21	Xiaomi_Humidifier	1414
—	—	—	22	Xiaomi_Soundbox	7352
Total	465354		Total	636789	

Table 4: The structure of the generator.

Layer	Output shape
Dense_1(Dense)	(None, 256)
LeakyReLU	(None, 256)
BatchNormalization	(None, 256)
Dense_2(Dense)	(None, 512)
LeakyReLU	(None, 512)
BatchNormalization	(None, 512)
Dense_3(Dense)	(None, 1024)
LeakyReLU	(None, 1024)
BatchNormalization	(None, 1024)
Dense_4(Dense)	(None, 2500)
LeakyReLU	(None, 2500)

Table 5: The structure of the discriminator.

Layer	Output shape
Dense_1(Dense)	(None, 2500)
LeakyReLU	(None, 2500)
Dropout	(None, 2500)
Dense_2(Dense)	(None, 1024)
LeakyReLU	(None, 1024)
Dropout	(None, 1024)
Dense_3(Dense)	(None, 512)
LeakyReLU	(None, 512)
Dropout	(None, 512)
Dense_4(Dense)	(None, 256)
LeakyReLU	(None, 256)
Dropout	(None, 256)
Dense_5(Dense)	(None, 1)

classification to achieve dimension-reduction visualization. The dimensions of each sample input to the t-SNE algorithm in the UNSW dataset and the laboratory dataset are 18 and 23, respectively, which are consistent with the number of IoT devices in the two datasets. The visualized effect of the dimensionality reduction result is shown in Figure 7. In the laboratory dataset, some devices are not highly distinguished, especially for several devices with the same manufacturer and type. The visualization results of these two datasets are consistent with the aforementioned experimental results. The clustering effect of the two datasets is excellent, and the separation distance among the different categories is relatively obvious. In general, CBBI can learn representative spatial and temporal characteristics from device communication traffic, which can be used as the basis for device identification.

5.4. Comparison Results. The classification accuracy of CBBI on the UNSW dataset is 99.83%, which achieves a similar effect to UNSW [27]. The detailed experimental results are shown in Figure 8. As far as we know, UNSW [27] is currently the highest accuracy rate for IoT device identification-related work, reaching 99.88%. The study in [27] used 6 months of IoT device communication traffic data. In addition, the author achieves accurate identification of IoT devices based on manually extracted features combined with a multistage device identification framework. The UNWS dataset that we used contains two weeks of traffic data, and thus, the communication traffic

Table 6: The structure of Conv-BiLSTM.

Hyperparameters		Value	Activation function
Conv-BiLSTM	Conv2D	#filters = 6, filter size = 5	ReLU
	MaxPlooing2D	#pool size = 2, padding = "valid"	—
	Conv2D	#filters = 16, filter size = 5	ReLU
	MaxPlooing2D	#pool size = 2, padding = "valid"	—
	Flatten	—	—
	Dense	#neurons = 1600, dropout = 0.5	ReLU
	BiLSTM	#neurons = 512, dropout = 0.3	Sigmoid
	BiLSTM	#neurons = 512, dropout = 0.3	Sigmoid
	Dense	#neurons = 18/23	Softmax
Optimizer	—	#Adam with learning rate = 0.001	—
Loss function	—	#Categorical_crossentropy	—
Batch size	—	#512	—
Epochs	—	#50	—

Table 7: Results of the ablation study on the UNSW dataset.

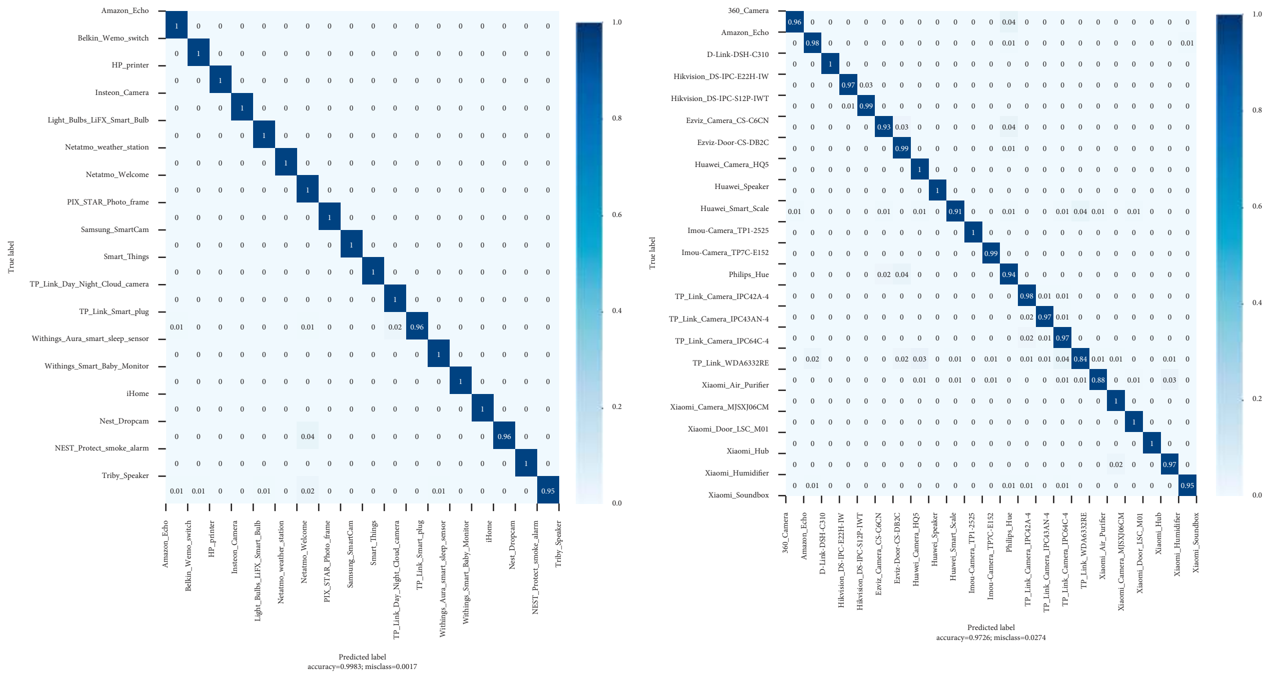
Method	CBBI w/o FGAN			CBBI		
FGAN		✗			✓	
Conv-BiLSTM		✓			✓	
Device name	Precision	Recall	F-score	Precision	Recall	F-score
Amazon_Echo	0.9878	0.9932	0.9905	0.9983	0.9988	0.9985
Belkin_Wemo_Switch	0.9932	0.9929	0.9931	0.9982	0.9988	0.9985
HP_Printer	0.9056	0.9646	0.9342	0.9908	0.9981	0.9944
Insteon_Camera	0.9931	0.9981	0.9956	0.9994	0.9994	0.9994
Light_Bulbs_LiFX_Smart_Bulb	0.9825	0.9825	0.9825	0.9979	0.9965	0.9972
Netatmo_Weather_Station	0.9865	0.9963	0.9914	0.9988	1.0000	0.9994
Netatmo_Welcome	0.9181	0.9533	0.9354	0.9937	0.9956	0.9947
PIX_STAR_Photo_Frame	0.9926	0.9887	0.9907	0.9990	0.9951	0.9971
Samsung_SmartCam	0.9978	0.9922	0.9950	0.9992	0.9989	0.9991
Smart_Things	0.9956	0.9888	0.9922	0.9985	0.9980	0.9983
TP_Link_Day_Night_Cloud_Camera	0.9898	0.9563	0.9728	0.9892	0.9989	0.9940
TP_Link_Smart_Plug	0.8638	0.8675	0.8657	0.9804	0.9615	0.9709
Withings_Aura_Smart_Sleep_Sensor	0.9929	0.9793	0.9861	0.9989	0.9985	0.9987
Withings_Smart_Baby_Monitor	0.9823	0.9961	0.9891	0.9997	0.9997	0.9997
iHome	1.0000	0.6833	0.8119	1.0000	1.0000	1.0000
Nest_Dropcam	0.3193	0.4371	0.3980	1.0000	0.9600	0.9796
NEST_Procet_Smoke_Alarm	1.0000	0.7317	0.8451	1.0000	1.0000	1.0000
Triby_Speaker	0.8841	0.4552	0.6010	0.9817	0.9469	0.9640
Macro average performance	0.9325	0.8865	0.9039	0.9958	0.9914	0.9935
Total accuracy		0.9865			0.9983	

Table 8: Results of the ablation study on the laboratory dataset.

Method	CBBI w/o FGAN			CBBI		
FGAN		✗			✓	
Conv-BiLSTM		✓			✓	
Device name	Precision	Recall	F-score	Precision	Recall	F-score
360_Camera	0.9504	0.9017	0.9254	0.9697	0.9566	0.9631
Amazon_Echo	0.9023	0.9395	0.9205	0.9435	0.9764	0.9597
D-Link-DSH-C310	0.8882	0.8988	0.8935	1.0000	1.0000	1.0000
Hikvision_DS-IPC-E22H-IW	0.9526	0.9161	0.9340	0.9902	0.9731	0.9816
Hikvision_DS-IPC-S12P-IWT	0.9255	0.9535	0.9393	0.9753	0.9907	0.9829
Ezviz_Camera_CS-C6CN	0.9251	0.9382	0.9316	0.9676	0.9264	0.9465
Ezviz_Door_CS-DB2C	0.9289	0.9596	0.9440	0.9300	0.9856	0.9570
Huawei_Camera_HQ5	0.9966	0.9968	0.9967	0.9988	0.9992	0.9990
Huawei_Speaker	0.9982	0.9936	0.9959	0.9991	0.9978	0.9985
Huawei_Smart_Scale	0.7975	0.8289	0.8129	0.9663	0.9101	0.9373
Imou-Camera_TP1-2525	0.9979	0.9962	0.9971	0.9986	0.9977	0.9982

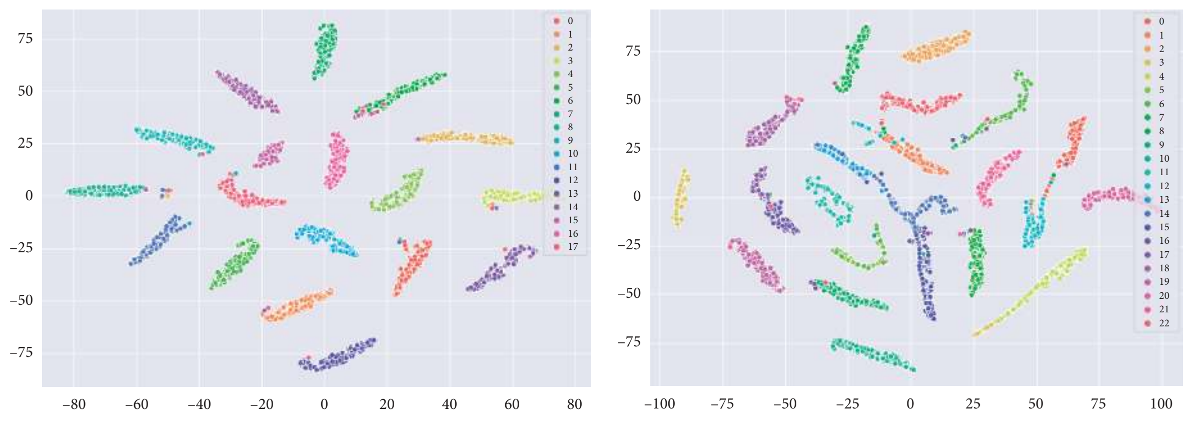
Table 8: Continued.

Method	CBBI w/o FGAN			CBBI		
Imou-Camera_TP7C-E152	0.9894	0.9894	0.9894	0.9977	0.9932	0.9955
Philips_Hue	0.9504	0.9165	0.9332	0.9606	0.9379	0.9491
TP_Link_Camera_IPC42A-4	0.9638	0.9499	0.9568	0.9806	0.9836	0.9821
TP_Link_Camera_IPC43AN-4	0.9581	0.9492	0.9536	0.9881	0.9731	0.9805
TP_Link_Camera_IPC64C-4	0.8520	0.9287	0.8887	0.9360	0.9682	0.9518
TP_Link_WDA6332RE	0.6364	0.4667	0.5385	0.7815	0.8378	0.8087
Xiaomi_Air_Purifier	0.6923	0.3333	0.4500	0.8676	0.8806	0.8741
Xiaomi_Camera_MJSXJ06CM	0.9919	0.9923	0.9921	0.9986	0.9979	0.9983
Xiaomi_Door_LSC_M01	0.9905	0.9881	0.9893	0.9952	0.9967	0.9960
Xiaomi_Hub	0.6857	0.4898	0.5714	0.9975	0.9988	0.9982
Xiaomi_Humidifier	0.8881	0.8975	0.8928	0.9730	0.9689	0.9709
Xiaomi_Soundbox	0.9135	0.9041	0.9088	0.9814	0.9483	0.9646
Macro average performance	0.9033	0.8752	0.8850	0.9651	0.9652	0.9649
Total accuracy		0.9524			0.9726	



(a) (b)

Figure 6: The confusion matrix of the two datasets. (a) The confusion matrix of the UNSW dataset. (b) The confusion matrix of the laboratory dataset.



(a) (b)

Figure 7: Visualization of the final fully connected layer based on t-SNE. (a) Visualization of the UNSW dataset. (b) Visualization of the laboratory dataset.

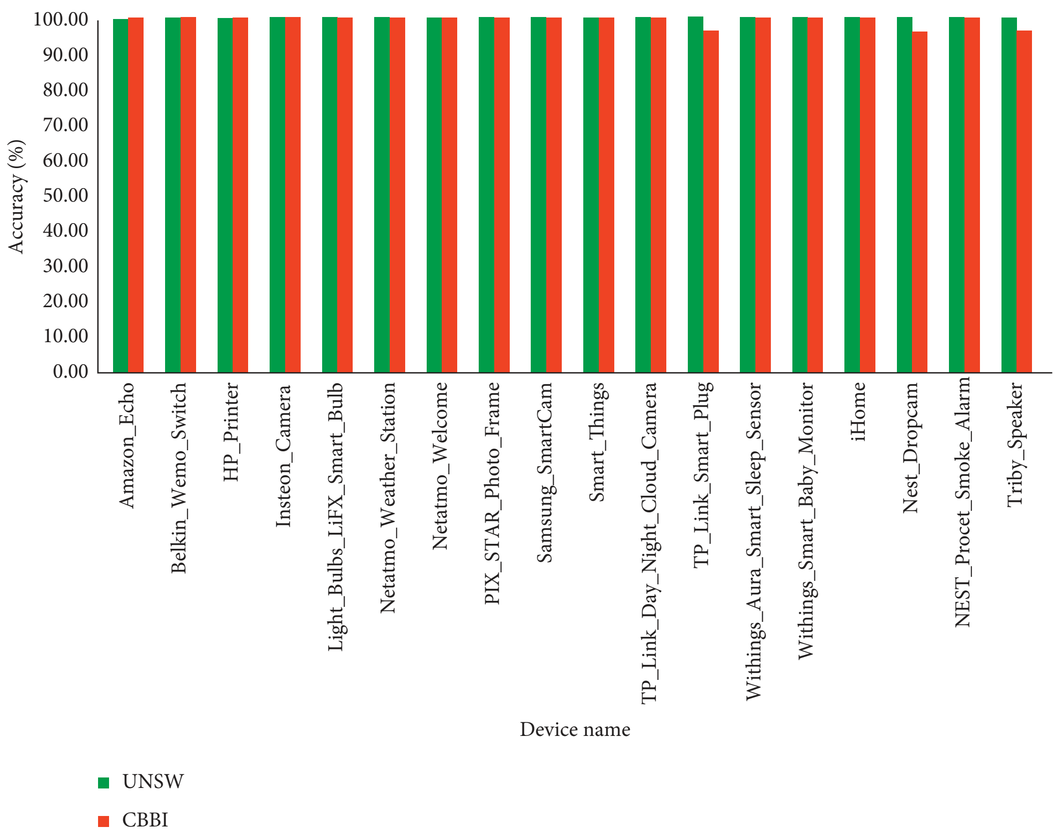


Figure 8: Comparison with related work on the UNSW dataset.

Table 9: Performance comparison of different model combinations.

Method		Accuracy	Precision	Recall	F1-score
CNN	UNSW dataset	0.9464	0.8919	0.8858	0.8872
	Laboratory dataset	0.9302	0.9026	0.8774	0.8877
FGAN + CNN	UNSW dataset	0.9517	0.8990	0.8963	0.8950
	Laboratory dataset	0.9463	0.9081	0.8925	0.8989
BiLSTM	UNSW dataset	0.9465	0.8897	0.8800	0.8812
	Laboratory dataset	0.9382	0.8875	0.8581	0.8664
FGAN + BiLSTM	UNSW dataset	0.9541	0.9059	0.8882	0.8906
	Laboratory dataset	0.9478	0.8900	0.8671	0.8754
CNN + BiLSTM	UNSW dataset	0.9865	0.9325	0.8865	0.9039
	Laboratory dataset	0.9524	0.9033	0.8752	0.8850
FGAN + CNN + BiLSTM (CBBI)	UNSW dataset	0.9983	0.9958	0.9914	0.9935
	Laboratory dataset	0.9726	0.9651	0.9652	0.9649

generated was relatively small, especially for several devices such as Nest_Dropcam, NEST_Procet_Smoke_Alarm, and Triby_Speaker. Our method achieves an accuracy rate similar to that of UNSW [27]. Additionally, CBBI does not need to manually extract features, which increases the timeliness of the device recognition.

We have implemented more comparative experiments, including CNN, FGAN + CNN, BiLSTM, FGAN + BiLSTM, CNN + BiLSTM, and CBBI. The detailed experimental results are shown in Table 9. Each method gives the accuracy, precision, recall, and F1-score values. We can conclude that FGAN and the simultaneous learning of temporal and spatial features can effectively improve the identification accuracy.

In summary, various experimental results show that our method can effectively and accurately identify the IoT devices. Compared with traditional manual feature extraction methods, this method can not only automatically learn the representative features of devices but also has good classification capabilities. On the other hand, the experimental results on the two datasets also show the effectiveness and flexibility of CBBI, which can address the complex and changeable IoT device environment.

6. Conclusions and Future Work

In this paper, we propose an IoT identification method called CBBI. This method uses the spatial and temporal features of the original network traffic generated by the IoT devices, which avoids the overhead and cumbersomeness of feature extraction in the traditional methods and reduces the complexity of the IoT device identification task. CBBI has three modules: data preprocessing, data augmentation FGAN, and Conv-BiLSTM. The main task of the data preprocessing module is to quickly process the raw network traffic generated by the IoT device and convert it into input that can be used in a deep learning model. The data augmentation module FGAN solves the problem of class imbalance in deep learning and further improves the accuracy and generalization ability of the model. The hybrid deep learning model Conv-BiLSTM can learn the spatial and temporal characteristics of the device communication traffic. In this paper, we use a public dataset and a laboratory dataset to verify the effectiveness of CBBI. The experimental results show that CBBI has good classification performance, even for some IoT devices from the same equipment manufacturers. In our future work, we will consider a combination of active and passive IoT device identification schemes to realize the identification of unknown IoT devices. [45].

Data Availability

We used the UNSW dataset, which is a publicly accessed dataset (https://iotanalytics.unsw.edu.au/iottraces). The laboratory dataset used to support the findings of this study is available from the corresponding author upon request.

Disclosure

This paper is an extended version of a conference paper, and the conference name is DSC 2021—IEEE Conference on Dependable and Secure Computing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Project (2017YFB0801805) and the National Natural Science Foundation of China (62072359 and 62072352).

References

[1] "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[2] "Internet of things (IoT) - statistics & facts," https://www.statista.com/statistics/370350/internet-of-things-installed-base-by-category/.

[3] 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.

[4] F. Yin, L. Yang, Y. Wang, and J. Dai, "Iot ETEI: end-to-end iot device identification method," in *Proceedings of the IEEE Conference on Dependable and Secure Computing, DSC 2021*, pp. 1–8, Aizuwakamatsu, Japan, January 2021.

[5] Y. Yue, S. Li, P. Legg, and F. Li, "Deep learning-based security behaviour analysis in iot environments: a survey," *Security and Communication Networks*, vol. 2021, Article ID 8873195, 13 pages, 2021.

[6] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proceedings of the IEEE Symposium on Security and Privacy, SP, 2016*, pp. 636–654, San Jose, CA, USA, May 2016.

[7] V. Sachidananda, J. Toh, S. Siboni, S. Asaf, and E. Yuval, "POSTER: towards exposing internet of things: a roadmap," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1820–1822, Vienna, Austria, October 2016.

[8] L. Garcia, F. Brasser, M. H. Cintuglu, R. S. Ahmed, A. M. Osama, and A. Z. Saman, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS 2017*, San Diego, CA, USA, February 2017.

[9] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn, "Iot goes nuclear: creating a zigbee chain reaction," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP2017*, pp. 195–212, San Jose, CA, USA, May 2017.

[10] Y. Jia, Y. Xiao, J. Yu, X. Cheng, Z. Liang, and Z. Wan, "A novel graph-based mechanism for identifying traffic vulnerabilities in smart home iot," in *Proceedings of the 2018 IEEE Conference on Computer Communications, INFOCOM*, pp. 1493–1501, Honolulu, HI, USA, April 2018.

[11] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: iot botnet of high wattage devices can disrupt the power grid," in *Proceedings of the 27th USENIX Security Symposium, USENIX Security 2018*, pp. 15–32, Baltimore, MD, USA, August 2018.

[12] Z. Liu, R. Wang, N. Japkowicz, Y. Cai, D. Tang, and X. Cai, "Mobile app traffic flow feature extraction and selection for improving classification robustness," *Journal of Network and Computer Applications*, vol. 125, pp. 190–208, 2019.

[13] X. Xiao, R. Li, H. T. Zheng, R. Ye, A. KumarSangaiah, and S. Xia, "Novel dynamic multiple classification system for network traffic," *Information Sciences*, vol. 479, pp. 526–541, 2019.

[14] D. L. C. Choong, C. Y. Cho, C. P. Tan, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security, WISEC 2008*, pp. 46–55, Alexandria, VA, USA, March, 2008.

[15] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM 2008*, pp. 104–115, San Francisco, CA, USA, September 2008.

[16] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[17] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010*, pp. 169–174, Hoboken, NJ, USA, March 2010.

[18] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using

nonparametric bayesian method," in *Proceedings of the IEEE INFOCOM*, pp. 1404–1412, Shanghai, China, April 2011.

[19] S. U. Rehman, K. W. Sowerby, and C. Coghill, "Analysis of impersonation attacks on systems using rf fingerprinting and low-end receivers," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 591–601, 2014.

[20] H. L. Yuan and A. Q. Hu, "Preamble-based detection of wi-fi transmitter rf fingerprints," *Electronics Letters*, vol. 46, no. 16, pp. 1165–1167, 2010.

[21] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM 2008*, pp. 116–127, San Francisco, California, USA, September 2008.

[22] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "Gtid: a technique for physical device and device type fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, 2014.

[23] M. Miettinen, S. Marchal, I. Hafeez et al., "Iot SENTINEL: automated device-type identification for security enforcement in iot," in *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017*, pp. 2177–2184, Atlanta, GA, USA, June 2017.

[24] H. Guo and J. S. Heidemann, "Ip-based iot device detection," in *Proceedings of the 2018 Workshop on IoT Security and Privacy, IoT S&P@SIGCOMM 2018*, pp. 36–42, Budapest, Hungary, August 2018.

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

[26] V. Thangavelu, D. M. Divakaran, R. Sairam, S. B. Suman, and G. Mohan, "Deft: a distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2018.

[27] A. Sivanathan, H. H. Gharakheili, F. Loi et al., "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[28] L. Yu, T. Liu, Z. Zhou, and Y. Zhu, "WDMTI: wireless device manufacturer and type identification using hierarchical dirichlet process," in *Proceedings of the 15th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2018*, pp. 19–27, Chengdu, China, October 2018.

[29] L. Yu, B. Luo, J. Ma et al., "You are what you broadcast: identification of mobile and iot devices from (public) wifi," in *Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020*, pp. 55–72, August 2020.

[30] L. Deng, Y. Feng, D. Chen, and N. Rishe, "Iotspot: Identifying the iot devices using their anonymous network traffic data," in *Proceedings of the 2019 IEEE Military Communications Conference, MILCOM 2019*, pp. 1–6, Norfolk, VA, USA, November 2019.

[31] M. Antonakakis, T. April, M. Bailey et al., "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Security Symposium, USENIX Security 2017*, pp. 1093–1110, Vancouver, BC, Canada, August 2017.

[32] Z. Durumeric, D. Adrian, A. Mirian et al., "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542–553, Denver, CO, USA, October 2015.

[33] Shodan, "The search engine for Internet-connected devices," https://www.shodan.io/.

[34] C. Fachkha, E. Bou-Harb, A. Keliris et al., "Internet-scale probing of CPS: inference, characterization and orchestration analysis," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS 2017*, San Diego, CA, USA, February 2017.

[35] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, pp. 318–327, 2019.

[36] Q. Li, X. Feng, L. Zhao, and L. Sun, "A framework for searching internet-wide devices," *IEEE Network*, vol. 31, no. 6, pp. 101–107, 2017.

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

[38] "SplitCap tool," https://www.netresec.com/?page=SplitCap.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: a search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[42] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd, Birmingham, United Kingdom, 2017.

[45] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*, Insecre. Com LLC (US), WA, USA, 2008.