applied sciences

MDPI

Article

Unknown IoT Device Identification Models and Algorithms Based on CSCL-Siamese Networks and Weighted-Voting Clustering Ensemble

Junhao Qian [1], Wenyu Zheng [1,2], Xulin Lu [2] and Zhihua Li [1,2,*]

[1] School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China; qjhao@jiangnan.edu.cn (J.Q.); wyzheng1015@163.com (W.Z.)

[2] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214112, China; 18360896220@163.com

* Correspondence: zhli@jiangnan.edu.cn

Abstract: Current methods for identifying unknown Internet of Things (IoT) devices are relatively limited. Most approaches can identify only one type of the unknown IoT devices at a time and with a relatively low accuracy. Herein, we propose a method for unknown IoT device identification (UDI) based on cost-sensitive contrastive loss (CSCL)-Siamese networks and a weighted-voting clustering ensemble (WVE). First, we integrate data visualization techniques with a permutation sample-pairing strategy to generate a complete and nonredundant set of positive–negative sample pairs. Then, we present an algorithm to generate permutation positive–negative sample pairs to provide a rich set of contrastive training data. To overcome the bias in the decision boundary caused by an insufficient number of positive sample pairs, we developed a Siamese network based on CSCL. The CSCL-Siamese network is used to identify known IoT devices and establish an embedded vector database for known IoT devices. Next, we extract the embedding vectors of unknown IoT devices using the trained CSCL-Siamese network and the embedded vector database. Finally, combining weighting factors with a voting ensemble strategy, we develop a UDI algorithm based on a WVE. This presented algorithm integrates the clustering capabilities of multiple unsupervised clustering algorithms to perform clustering on the extracted embedding vectors of unknown IoT devices, thereby enhancing the identification capability of the CSCL-WVE-UDI method. Experimental results demonstrate that the CSCL-WVE-UDI method can effectively identify multiple types of unknown IoT devices at the same time.

Keywords: network traffic; positive and negative pair samples; ensemble learning; unknown IoT device identification

Academic Editors: Young-Gab Kim and Marco Mamei

Received: 11 February 2025
Revised: 30 April 2025
Accepted: 6 May 2025
Published: 9 May 2025

Citation: Qian, J.; Zheng, W.; Lu, X.; Li, Z. Unknown IoT Device Identification Models and Algorithms Based on CSCL-Siamese Networks and Weighted-Voting Clustering Ensemble. *Appl. Sci.* **2025**, *15*, 5274. https://doi.org/10.3390/app15105274

1. Introduction

According to the International Data Corporation (IDC), the world's total data volume will reach 175 zettabytes (zettabytes) by 2025. About 80% of these data is generated by smart sensors, mobile terminals, industrial devices, etc. [1]. This development trend highlights the immense potential of the IoT device market and suggests that IoT technology will revolutionize multiple industries. However, both academia and industry agree that IoT devices have weak security defense capabilities owing to their low computing power and limited storage capacity [2]. This inherent limitation of IoT devices is one of the main causes of information and network security risks in IoT systems, particularly in areas such as cross-network attacks targeting IoT devices connected to the Internet, IoT-botnet attack detection, and network-attack traceability. Accurate identification of IoT devices is critical in these contexts because appropriate network defense strategies and technical methods can only be selected through the accurate identification of device types and their communication protocols. Nevertheless, most current IoT device identification methods focus on recognizing known IoT devices in closed environments and struggle to adapt to the continuously evolving open environments. Furthermore, these methods fail to automatically update their models to identify unknown IoT devices [3], thus highlighting the need for conducting studies on the identification of unknown IoT devices.

Researchers have conducted a series of studies in recent years to identify unknown IoT devices in open environments. The authors of [4,5] adopted Siamese networks as the training model. They first identified known IoT devices and constructed an embedded vector database for them. Subsequently, the embedding vectors of unknown IoT devices were checked against the database, with identification performed based on a predefined similarity distance. The author of [6] demonstrated the capability of Siamese networks in reducing model retraining. To identify unknown IoT devices, the author of [7] employed a convolutional neural network (CNN) model and a triplet loss-minimization function to generate embedding vectors of unknown IoT devices; subsequently, they classified the vectors using the k-nearest neighbors (KNN) algorithm. The author of [8] proposed creating a separate identification model for each type of known IoT device, which was matched against unknown IoT devices to filter out their traffic data. The filtered data were further clustered using the ordering points to identify the clustering structure (OPTICS) algorithm to identify unknown IoT devices. However, the methods above for unknown IoT device identification (UDI) present several limitations. In [4,5], the reliance on similarity distances for identifying unknown IoT devices limits the ability to distinguish multiple types of unknown IoT devices in one attempt. Although the KNN algorithm employed in [7] can identify multiple types of unknown IoT devices, it requires prior knowledge regarding the number of unknown devices and their corresponding data, which is typically unavailable in open environments. In [8], considering the identification model was trained for each IoT device individually, the training overhead increased exponentially with the number of IoT devices. Moreover, these methods generally overlook the learning potential of multiple clustering algorithms and fail to effectively integrate them for the clustering-ensemble identification of unknown IoT devices. To effectively identify various unknown IoT devices in open environments, we first combined data visualization techniques with a permutation sample-pairing strategy to generate a set of permutation positive–negative sample pairs (PNSPs). Then, to mitigate the bias in the decision boundary caused by the insufficient number of positive sample pairs, we cast cost-sensitive factors into the contrastive loss function to establish a Siamese network based on cost-sensitive contrastive loss (CSCL). This network is trained to identify known IoT devices and build an embedded vector database for them. Next, the embedding vectors of unknown IoT devices are extracted using the trained CSCL-Siamese network and checked against the embedded vector database. Finally, to integrate the clustering capabilities of multiple unsupervised clustering algorithms, we improve the voting ensemble strategy by employing weighting factors. This approach combines multiple unsupervised clustering algorithms to perform the clustering-ensemble identification of unknown IoT device embedding vectors, thereby helping identify various types of unknown IoT devices.

In summary, we propose a method for UDI based on CSCL-Siamese networks and a weighted-voting clustering ensemble (WVE). The CSCL-WVE-UDI method generates contrastive training data by combining data visualization techniques with a permutation sample-pairing strategy and uses cost-sensitive factors and a contrastive loss function to design the CSCL-Siamese network for training on the contrastive data, thus achieving known IoT device identification and realizing an embedded vector database. Next, the CSCL-Siamese network and database were used to extract the embedding vectors of unknown IoT devices. Finally, a clustering-ensemble learning algorithm was proposed using weighting factors and a voting ensemble strategy to cluster the embedding vectors of unknown IoT devices, thus enabling the identification of various unknown IoT devices. The main contributions of this paper are as follows:

(1) An algorithm for generating PNSPs, which establishes a set of PNSPs by combining data visualization techniques with a permutation sample-generation strategy, is proposed. This algorithm ensures that each sample is fully utilized, thus mitigating the issue of reduced identification performance caused by underutilized datasets.
(2) A Siamese network based on cost-sensitive contrastive loss (CSCL-Siamese) is presented. By improving the classical contrastive loss function, using the Manhattan distance as the similarity metric, and leveraging the disparity between counts of positive and negative sample pairs, this approach assigns greater weighting factors to the positive sample pairs to increase their loss cost, thus addressing the issue of decision boundary bias toward negative sample pairs caused by an insufficient number of positive sample pairs.
(3) A UDI algorithm based on a WVE (WVE-UDI) is developed. By combining weighting factors with a voting ensemble strategy, this algorithm integrates the clustering results of multiple unsupervised clustering algorithms, thus overcoming the limitations of individual clustering algorithms and improving the capability for identifying unknown IoT devices.
(4) Finally, the algorithms above are integrated into the CSCL-WVE-UDI method for UDI. Experimental results show that the CSCL-WVE-UDI method can effectively identify multiple types of unknown IoT devices.

The rest of this paper is organized as follows. Section 2 reviews the existing methods for IoT device identification, discusses their limitations, and highlights opportunities for improvement through collaborative learning and clustering ensembles. Section 3 shows the CSCL-WVE-UDI scheme, detailing its logical architecture and workflow. Section 4 performs an evaluation of the prototype on benchmark datasets (Aalto, UNSW, LSIF) and compares it with the other methods. Section 5 derives the conclusions.

2. Literature Review

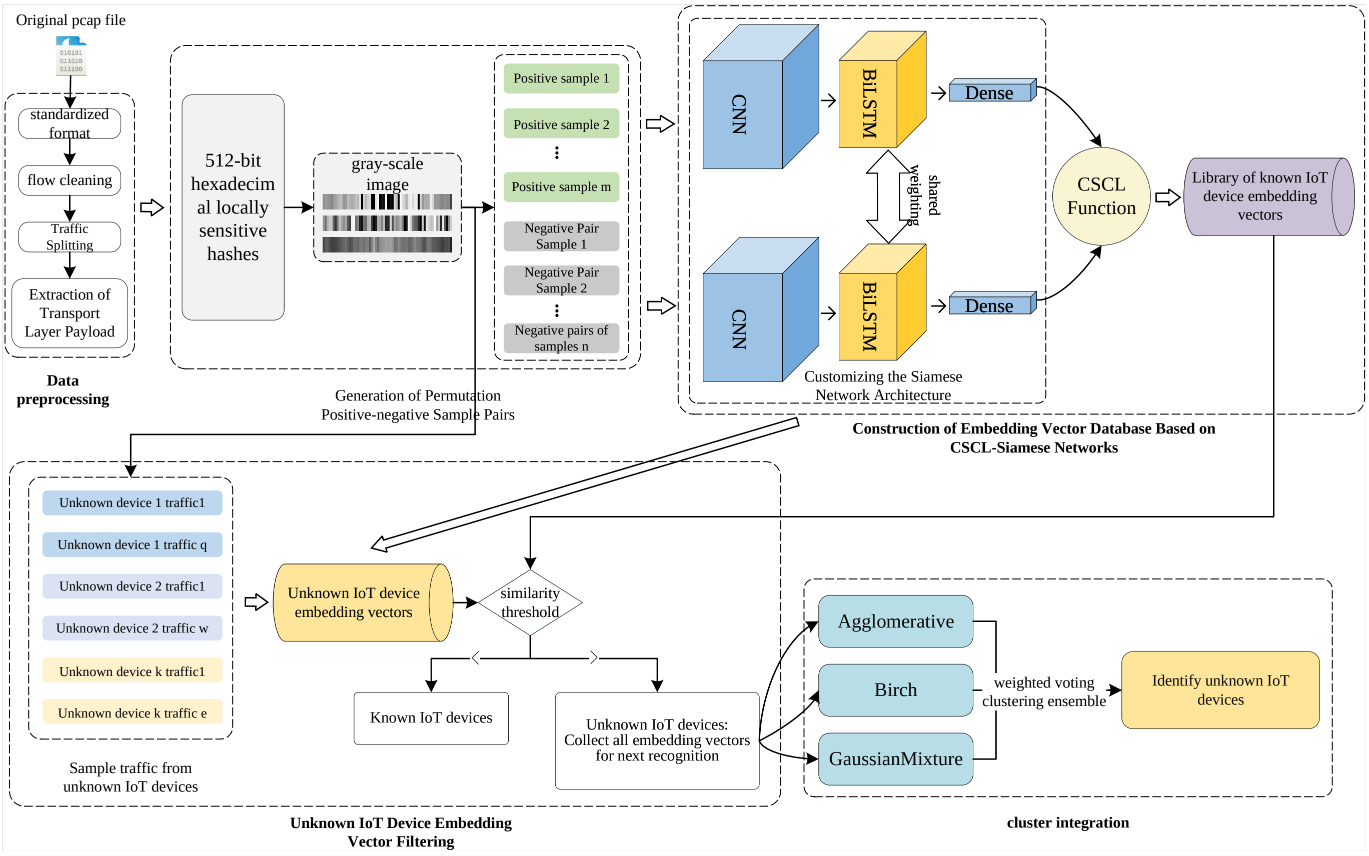
To identify unknown IoT devices in open environments, the author of [4] proposed an IoT-Siamese method for identifying unknown IoT devices. The IoT-Siamese method first reconstructs raw traffic packets into continuous flow data and removes all link-layer fields from the flow. Next, it extracts the first 100 bytes of each packet to construct device fingerprints. Subsequently, a Siamese network is trained on the fingerprints of known IoT devices, and the trained model is saved to establish an embedded vector database for known IoT devices. When new traffic data are received, they are input into the pretrained Siamese network to obtain embedding vectors, which are then checked against the known IoT device embedded vector database by calculating the shortest average similarity distance to determine if the device is unknown. Experimental results show that, even for small traffic volumes, the IoT-Siamese method can effectively identify IoT devices, although the identification accuracy for some unknown IoT devices is less than 50%. Furthermore, this method requires training a separate Siamese network for each new unknown IoT device, significantly increasing the training overhead. The author of [5] proposed a UDI method based on Siamese networks. Specifically, a Siamese network is utilized to learn the relationships between known IoT devices and generate an embedded vector database of known IoT devices. When an unknown IoT device joins the network, its traffic data are input into the pretrained Siamese network to generate embedding vectors, which are then matched with the database using the nearest distance, thus reducing the necessity for frequent model retraining due to device updates. The author of [6] evaluated the performance of the Siamese network on different datasets and verified its ability to identify IoT devices with different traffic patterns when new devices are added to the network. However, a significant drawback of the methods presented in [5,6] is their inability to determine the number of unknown IoT device types in the network under the participation of multiple unknown devices. Although these methods can identify unknown IoT devices, they require training separate network models for each device type to distinguish specific types. To identify different types of IoT devices, the author of [7] proposed a method for identifying unknown IoT devices based on the KNN algorithm. First, the network traffic of IoT devices was truncated into 784-bit session data and then converted into 28×28 images as device fingerprints. Subsequently, an auxiliary classifier generative adversarial network was used for data augmentation on the image fingerprints. A CNN model was trained with a triplet loss-minimization function to generate embedding vectors. Finally, the KNN algorithm was employed to train the embedding vectors of multiple types of unknown IoT devices. This approach achieved an identification accuracy of up to 90% for some devices. However, the KNN algorithm requires prior knowledge of the true labels and number of unknown IoT devices for training before classification can be performed. In actual scenarios, administrators cannot determine the labels or number of unknown IoT devices in advance, thus limiting the method's effectiveness. The author of [8] introduced a hybrid deep learning method to identify unknown IoT devices. The core approach was classified into three steps: first, for each type of known IoT device, a separate network model was trained and saved appropriately; second, when new traffic data were encountered, they were matched with the saved models individually to filter out the embedding vectors of unknown IoT devices; third, for these unknown devices, the data were compressed using an autoencoder, and density-based clustering recognition was performed using the OPTICS algorithm. However, as this method requires training a separate identification model for each known device type, the training cost increases exponentially with the number of known IoT devices.

In summary, the existing methods for identifying unknown IoT devices present several limitations, including the inability to recognize more than one type of unknown IoT device at a time and the relatively low accuracy. Moreover, these studies generally overlook the potential of integrating multiple clustering algorithms, which can improve the identification of multiple unknown IoT devices by leveraging their clustering capabilities. Clustering algorithms can manage unlabeled data; however, their potential application in this domain has not been fully investigated. Additionally, based on a prior survey [3], studies on the identifying of unknown IoT devices are relatively scarce and underdeveloped. Therefore, future designs of IoT device identification solutions should consider the possibility of unknown IoT devices joining the network.

3. CSCL-WVE-UDI: Method for Identifying Unknown IoT Devices

3.1. Logical Architecture of CSCL-WVE-UDI Method

As shown in Figure 1, the logical architecture of the CSCL-WVE-UDI method comprises five components. The data-preprocessing component extracts session-level payload data from raw traffic. Generating PNSPs converts the payload data into grayscale image samples using an improved Nilsimsa algorithm and further applies the permutation sample-pairing strategy to generate the corresponding positive–negative sample pairs for each type of IoT device. The embedding-vector database-construction component, which is based on the CSCL-Siamese network, employs the proposed CSCL-Siamese network to train the set of PNSPs, thus enabling the identification of known IoT devices and the construction of an embedded vector database for known IoT devices. The filtering of unknown IoT device embedding-vector component inputs the traffic features of unknown IoT devices into the CSCL-Siamese network to obtain the corresponding embedding vectors, which are then compared against the known IoT device embedded vector database. By setting a similarity threshold, unknown IoT devices with features distinct from known devices are filtered out. Finally, the clustering-ensemble component integrates the clustering results of multiple unsupervised clustering algorithms through weighted voting and clusters the embedding vectors of unknown IoT devices, thereby completing the identification of unknown IoT devices. Each component is discussed in detail below.



Original pcap file
standardized format
flow cleaning
Traffic Splitting
Extraction of Transport Layer Payload
Data preprocessing
512-bit hexadecimal locally sensitive hashes
gray-scale image
Positive sample 1
Positive sample 2
Positive sample m
Negative Pair Sample 1
Negative Pair Sample 2
Negative pairs of samples n
Generation of Permutation Positive-negative Sample Pairs
CNN
BiLSTM
Dense
shared weighting
CNN
BiLSTM
Dense
Customizing the Siamese Network Architecture
CSCL Function
Library of known IoT device embedding vectors
Construction of Embedding Vector Database Based on CSCL-Siamese Networks
Unknown device 1 traffic1
Unknown device 1 traffic q
Unknown device 2 traffic1
Unknown device 2 traffic w
Unknown device k traffic1
Unknown device k traffic e
Sample traffic from unknown IoT devices
Unknown IoT device embedding vectors
similarity threshold
Known IoT devices
Unknown IoT devices: Collect all embedding vectors for next recognition
Unknown IoT Device Embedding Vector Filtering
Agglomerative
Birch
GaussianMixture
weighted voting clustering ensemble
Identify unknown IoT devices
cluster integration

Figure 1. Logical architecture of the CSCL-WVE-UDI approach.

3.2. Data Preprocessing

Raw traffic data from IoT devices typically contain noise. In open environments, the format of traffic data sources from IoT devices lacks uniformity, thus rendering it challenging to directly learn traffic features. Therefore, raw traffic data of IoT devices must be preprocessed to ensure data quality and consistency, thus providing a basis for effective feature learning. The data-preprocessing process is detailed in Figure 2. First, as raw traffic data from IoT devices typically exist in pcap and pcappng formats, TShark [9] is used to standardize the raw traffic data in pcappng format to pcap format. Next, to reduce noise and simplify the extraction of traffic features, the raw pcap traffic data are cleaned by removing invalid traffic data, such as packets without payloads or empty packets. Subsequently, the SplitCap tool is employed to segment the cleaned large-scale traffic samples into smaller-scale traffic files based on the five-tuple format (source IP address, destination IP address, source port, destination port, and transport layer protocol) [10], thus generating session data with finer granularity and richer information. Finally, as

IoT devices typically have specialized communication protocols and data formats and payload data contain the characteristic information of IoT devices while requiring minimal computational resources and are independent of specific protocols or standards, the Scapy tool (2.6.1) [11] is used to extract the effective payload from the transport layer of all packets within each session.

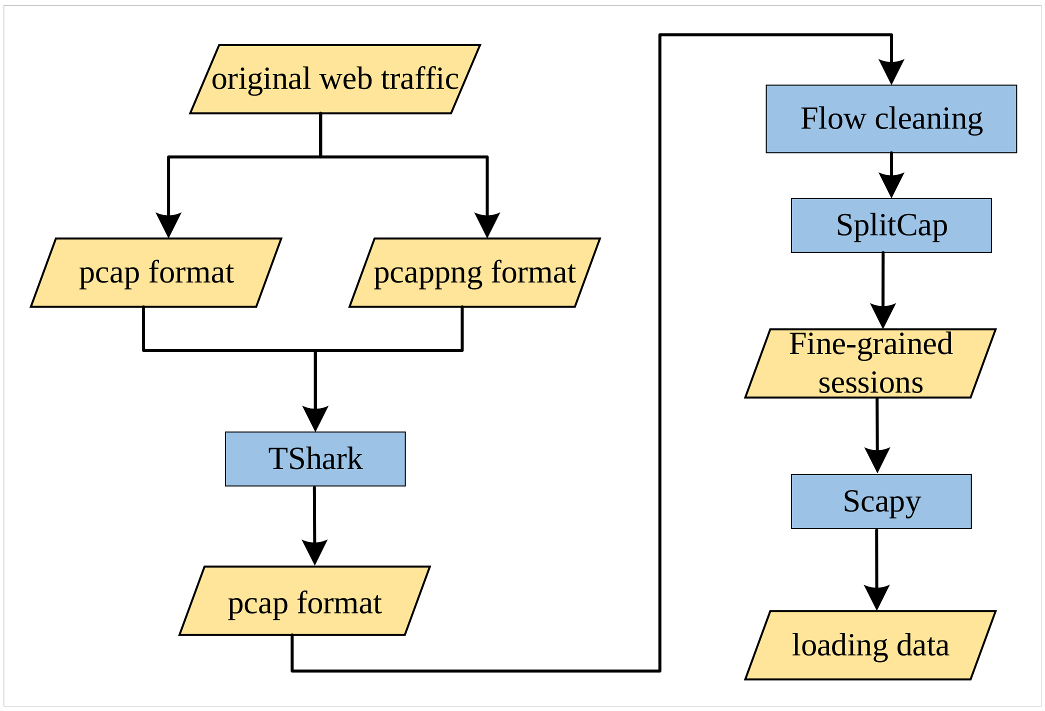


Figure 2. Flowchart of data preprocessing.

Based on the processing steps above, we propose a data-preprocessing algorithm. Its pseudocode is described in Algorithm 1. Let x_i represent the raw traffic sample of the i-th IoT device. After data preprocessing, the effective payload is denoted as $s_i \leftarrow (s_i^1, s_i^2, \ldots, s_i^m)$, where m represents the number of sessions generated after segmenting the i-th traffic sample.

Algorithm 1 Data Preprocessing

```
Input: $x_i$ // Sample raw traffic for the ith IoT device
Output: $s_i \leftarrow (s_i^1, s_i^2, \ldots, s_i^m)$ // Load data
1:  $pcap_i \leftarrow$TShark($x_i$) // standardized format
2:  for each $d_j$ in $pcap_i$ do // Cleans each packet in the flow
3:      if ($d_j$ is Null) or ($d_j$) contains no payload then delete $d_j$ from $pcap_i$
4:      end if
5:  end for
6:  Utilize SplitCap tool to segment $pcap_i$ into individual sessions, grouped based on
    identical 5-tuple information, $y_1, y_2, \ldots, y_m$
7:  $s_i \leftarrow []$
8:  for each $y_j \leftarrow y_1$ to $y_m$ do // Indicates the first packet in the current session
9:      for each $t_k \leftarrow$Scapy($y_j$) do // Indicates the first packet in the current session
10:         extract transport layer payloads from $t_k$ and append them to $s_i^j$
11:     end for
12:     $s_i \leftarrow s_i^j$ // Obtain the payload of the ith flow sample
13: end for
14: return $s_i$
```

The computational overhead of Algorithm 1 primarily arises from traffic cleaning and payload extraction. Traffic cleaning requires iteration through each data packet, thus resulting in a time complexity of $O(n)$. Payload extraction requires iterating through each packet in every session, thus resulting in a time complexity of $O(n^2)$. Thus, the total time complexity of Algorithm 1 mainly results from here.

3.3. Algorithm for Generating Permutation Positive–Negative Sample Pairs (PNSPs)

To generate a complete and nonredundant set of positive–negative sample pairs and ensure efficient utilization of the data, we combined data visualization techniques with a PNSP generation strategy to create contrastive training data. The objective is to provide a rich set of contrastive learning samples for the network model. The process is classified into two main stages: (1) constructing device fingerprints based on grayscale images and (2) generating PNSPs. Each stage is discussed in detail below.

3.3.1. Construction of Device Fingerprints Based on Grayscale Images

To further reveal the inherent patterns and characteristics of the preprocessed payload data, the preprocessed payload data are transformed into expressive grayscale images, which are used as IoT device fingerprints. The specific steps are as follows: First, the Nilsimsa algorithm [12] has a mathematical property that generates similar hash values for similar content, which allows one to distinguish different IoT devices. However, the classic Nilsimsa algorithm [12] generates relatively small hash values. Consequently, significant details vanish during the hash image-conversion process, thus reducing the distinguishability of the preprocessed payload data. Therefore, a sliding window is introduced, and the idea of generating mixed integers through shift operations from the executable and linkable-format hash function is incorporated [13] to improve the Nilsimsa algorithm. Subsequently, the length of the locally sensitive hash values produced by the improved algorithm is extended to 512 bits. This ensures that the subsequent grayscale images retain the detailed features of the payload data more accurately. Next, the data visualization technique is introduced. The 512-bit hexadecimal hash value is mapped to a grayscale image using the Pillow library (10.0.1) . In this mapping, 0×00 corresponds to black, $0 \times \mathrm{FF}$ corresponds to white, and each pixel represents two hexadecimal digits. The pixels are arranged sequentially from left to right, thereby forming a 1×64-sized grayscale image in JPG format. These grayscale images collectively constitute the fingerprint sample set of IoT devices. The improvement strategy for the Nilsimsa algorithm is illustrated in Figure 3.

The improvement process can be summarized as follows:

(1) A sliding window W_0 of size 5 with a step size of 1 is applied to the preprocessed payload data sample s_i and slid to the right. Each window generates multiple triplets, denoted as $t_1, t_2, \ldots, t_z$, and each triplet is processed by the tran53 function to compute a value, denoted as *index*.

(2) A 512-length accumulator *acc* is defined. The *index* value is used to indicate the index position of the accumulator. Each time an *index* value is computed, the corresponding index position in the accumulator is incremented by 1. The sliding window shifts sequentially, and all accumulated values are stored in *acc*.

(3) Two sliding windows, each with a size of 16 and a step size of 16, are defined to generate more accumulated values. The initial positions of the first and second windows are set to 0 and 8, respectively. The values in the first and second windows are summed, which are denoted as w and v, respectively. Since the length of *acc* is 512, each window must slide 32 times.

(4) The accumulated values from the sliding window are processed via shift operations. First, the lower 8 bits of w are XORed with the upper 8 bits of n_1 to obtain n_1, as shown in Equation (1). Subsequently, the upper 8 bits of w are XORed with the lower 8 bits of v to obtain n_2, as shown in Equation (2). The final values are constrained

to the range of 0–255 by applying a bitwise AND with 255, thus ensuring they are suitable for subsequent conversion into fixed-length hexadecimal hash values.

$$n_1 = (w \wedge 255) \oplus ((v \gg 8) \wedge 255) \tag{1}$$

$$n_2 = ((w \gg 8) \wedge 255) \oplus (v \wedge 255) \tag{2}$$

(5) The sliding window continues to shift to the right across *acc*, thus ultimately producing a 512-bit hexadecimal locally sensitive hash value.

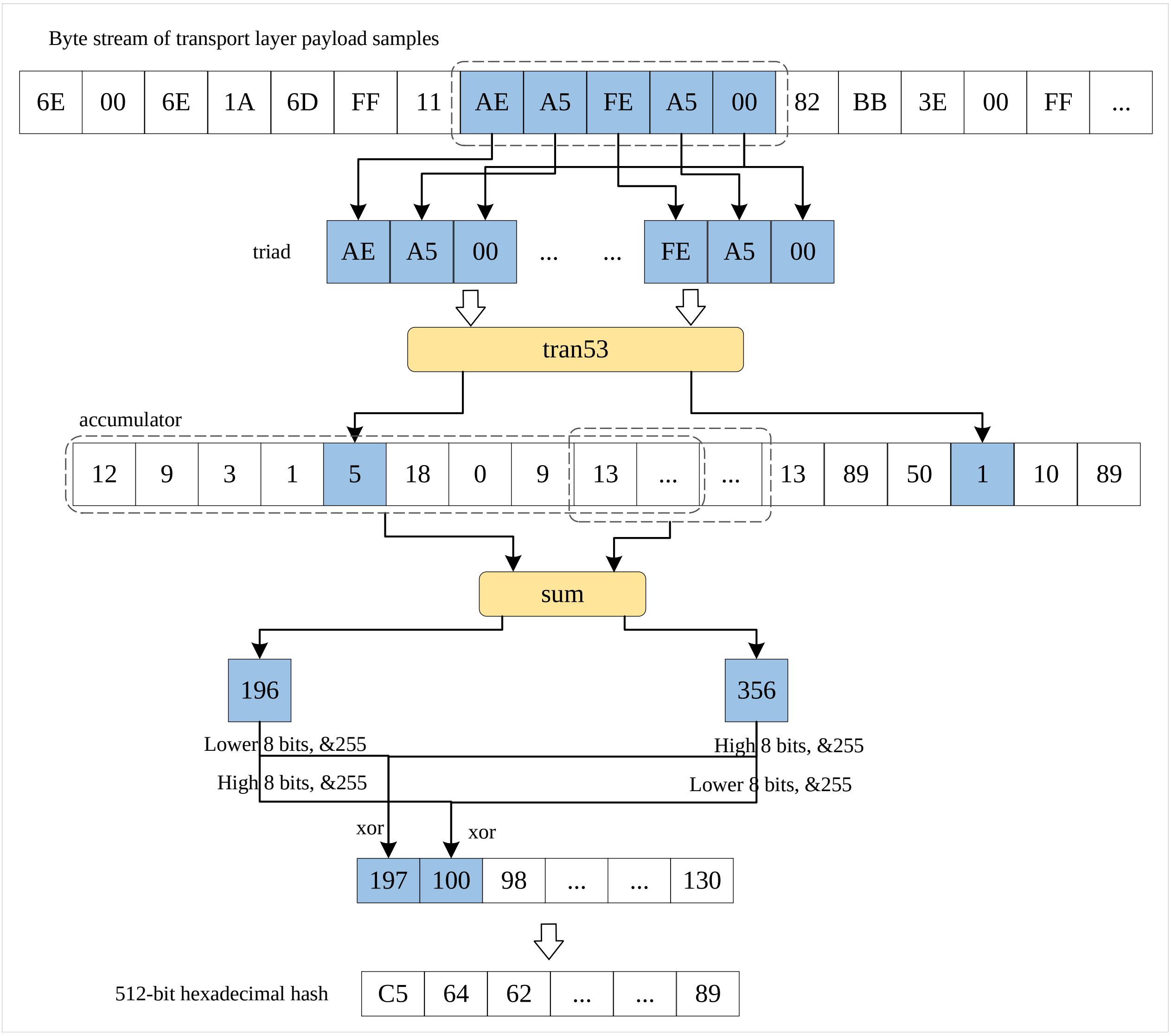


Figure 3. Example diagram of improvement strategy for Nilsimsa algorithm.

Based on the improvements above, we propose an improved Nilsimsa algorithm. The pseudocode of this algorithm is presented in Algorithm 2. The input is the preprocessed payload sample $s_i = (s_i^1, s_i^2, \ldots, s_i^m)$ of the i-th type IoT device, and the output is the 512-bit hexadecimal hash value $h_{s_i^j}$ of the payload sample.

The computational overhead of Algorithm 2 primarily arises from the hash-value generation. Specifically, the time complexity of computing the accumulator index positions is $O(n)$, and the time complexity of traversing the accumulator to compute the hash value is $O(n)$. Therefore, the total time complexity of Algorithm 2 is $O(n+n)$, where $O(n)$ is the length of the payload data.

Algorithm 2 Improved Nilsimsa Algorithm

```
Input: $s_i^j$ // Sample Load Data
Output: $h_{s_i^j}$ // 512-bit hexadecimal hash
 1: $acc \leftarrow [0]*512$ // Defining Accumulators
 2: $num_{s_i^j} \leftarrow []$ // Define an empty list to hold the integers resulting from the $s_i^j$ process
 3: Initialize a sliding window $W_0$ with window size of 5 and step size of 1
 4: $t_1, t_2, \ldots, t_z \leftarrow W_0(s_i^j)$ // Slide the window to the right to generate z triples
 5: for each $t_k \leftarrow t_1$ to $t_z$ do
 6:     $index \leftarrow tran53(t_k)$ // Calculate index position
 7:     $acc[index] \leftarrow acc[index] + 1$ // Accumulator corresponding to the index position accumulates 1
 8: end for
 9: for $k \leftarrow 0$ to 31 do // Indicates the first packet in the current session
10:     $w \leftarrow Sum(acc[k*16], \ldots, acc[(k+1)*16])$
11:     $v \leftarrow Sum(acc[k*16+8], \ldots, acc[(k+1)*16+8])$
12:     Calculate $n_1$ with equation (1) // Use of equation (1)
13:     Calculate $n_2$ with equation (2) // Use of equation (2)
14:     $num_{s_i^j} \leftarrow n_1, n_2$
15: end for
16: $d_i^j \leftarrow num_{s_i^j}$ // Getting Decimal Data
17: Convert $num_{s_i^j}$ to $h_{s_i^j}$ // Get the hexadecimal hash
18: return $h_{s_i^j}$
```

3.3.2. Generation of PNSPs

The conventional approach for generating positive–negative sample pairs is to randomly select a pair of samples from the device-fingerprint sample set. Positive sample pairs comprise fingerprints from the same IoT device for training the model to capture the consistent features of the same device. Negative sample pairs comprise fingerprints from different IoT devices for training the model to be sensitive to the differences between different devices. Although the random selection strategy is simple and easy to implement, it may cause the repeated selection of some device-fingerprint samples, while others are potentially overlooked. In IoT device identification, such randomness cannot fully reflect the characteristics of the dataset, thus rendering it difficult to capture key traffic features.

To address the limitations of the random selection strategy and to fully utilize each IoT device-fingerprint sample while clearly defining the relationships between sample pairs, we propose a method for generating PNSPs, as shown in Figure 4. The proposed method is based on the complete data samples generated in a previous study [4]. First, for the grayscale image fingerprint samples of each IoT device, nonredundant positive sample pairs are created to ensure that effective pairings are formed between samples of the same IoT device. During this process, each fingerprint sample can be paired with itself to enrich the number of positive sample pairs. Subsequently, each fingerprint sample of the current IoT device is paired with the samples of all other IoT devices to construct the complete set of negative sample pairs for the current device. When generating negative sample pairs, redundant pairings are avoided through meticulous attention, thus ensuring that previously paired negative samples are not rematched. For example, if t_{i1} is the first fingerprint sample of the *i*-type IoT device and t_{j2} is the second fingerprint sample of the *j*-type IoT device, then the pairing of (t_{i1}, t_{j2}) with (t_{j2}, t_{i1}) essentially represents the same negative relationship, regardless of order. Therefore, when constructing negative sample pairs for the *j*-type IoT device, redundant sample pairs are eliminated to remove superfluous samples, thus improving the efficiency and quality of sample-pair generation.

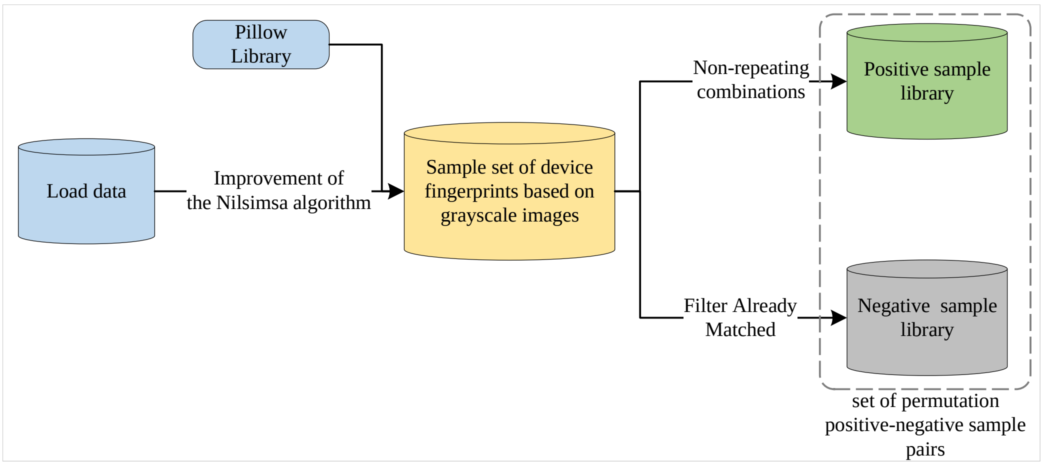

Pillow Library
Load data
Improvement of the Nilsimsa algorithm
Sample set of device fingerprints based on grayscale images
Non-repeating combinations
Positive sample library
Filter Already Matched
Negative sample library
set of permutation positive-negative sample pairs

Figure 4. Flowchart for generating a set of permutation positive–negative sample pairs.

3.3.3. PNSP Algorithm

Based on the process above, we propose an algorithm to generate PNSP. The pseudocode for the PNSP algorithm is presented in Algorithm 3.

Algorithm 3 PNSP Algorithm

```
Input: $S \leftarrow (s_1, \ldots, s_i, \ldots, s_f)$ // Sample Load Data for IoT Devices
Output: $Po \leftarrow (po_1, \ldots, po_i, \ldots, po_f); Ne \leftarrow (ne_1, \ldots, ne_i, \ldots, ne_f)$ // Full permutation of positive and negative pairs of samples
1:  $G \leftarrow []; Po \leftarrow []; Ne \leftarrow []$
2:  for each $s_i$ in $S$ do // Traverse the payload samples of the i-th category of IoT devices
3:      for each $s_i^j$ in $s_i$ do // Traverse the j-th payload sample
4:          $h_{s_i^j} \leftarrow$ Improved-Nilsimsa($s_i^j$) // Call Algorithm 2 to generate a 512-bit hexadecimal hash value
5:          $g_i^j \leftarrow h_{s_i^j}$ use Pillow // Generating grayscale images with the Pillow library
6:      end for
7:      $G \leftarrow g_i$ // Store the grayscale images of the i-th category of IoT devices
8:  end for
9:  for each $i \leftarrow 1$ to $f$ do
10:     $po_i \leftarrow$ combine all positive pairs for $g_i$
11:     $Po \leftarrow po_i$ // Current IoT devices without duplicate positive pair samples
12:     for $ii \leftarrow i+1$ to $f$ do // Filtering already generated negative pair samples
13:         for $k \leftarrow 1$ to $len(g_i)$ do
14:             for $p \leftarrow 1$ to $len(g_i)$ do
15:                 $ne_i \leftarrow (g_i^k, g_{ii}^p)$
16:             end for
17:         end for
18:         $Ne \leftarrow ne_i$ // Sample of all negative pairs of current IoT devices
19:     end for
20: end for
21: return $Po; Ne$
```

The computational overhead of Algorithm 3 primarily originates from traversing the IoT device payload samples to generate hash values, as well as from generating permutation sample pairs. Generating the 512-bit hexadecimal hash value requires traversing every payload sample of each IoT device, with a time complexity of $O(n^2)$. The permutation sample-pair generation requires matching positive and negative sample pairs and filtering out previously matched negative sample pairs, thus resulting in a time complexity of $O(n^4)$. Thus the total time complexity of Algorithm 1 mainly results from the permutation sample-pair generation.

3.4. Embedding-Vector Database Generation Based on CSCL-Siamese Network: Embedding-Vector Database Construction (EVDC)-CSCL-Siamese Algorithm

Embedding vectors offer an efficient feature-representation method that accurately captures the core characteristics and structural information of device fingerprints [5]. The network model, as a key technology for generating embedding vectors, is used primarily to deeply learn and extract complex features of device fingerprints and then map them into a high-dimensional embedding-vector space for an effective representation of the fingerprints. This representation approach preserves key features of device fingerprints while facilitating the numerical expression of the fingerprints, thereby providing robust support for subsequent similarity calculations among various IoT devices. Existing models, such as recurrent neural networks and GANs, can generate embedding vectors suitable for multiclass identification after extensive training, thus enabling the effective identification of multiple types of known IoT devices. However, these models necessitate frequent retraining to adapt to the continually increasing number of unknown IoT devices, which renders it difficult to satisfy the requirement of identifying increasing numbers of IoT devices in open environments.

Hence, we propose an embedding-vector database-generation model based on the CSCL-Siamese network. This model employs a Siamese network trained using a cost-sensitive loss function to identify known IoT devices. Additionally, it leverages traffic data from known IoT devices to construct an embedded vector database, thus facilitating similarity computations with embedding vectors from unknown IoT devices. The overall structure is illustrated in Figure 5, which we will discuss in detail below.

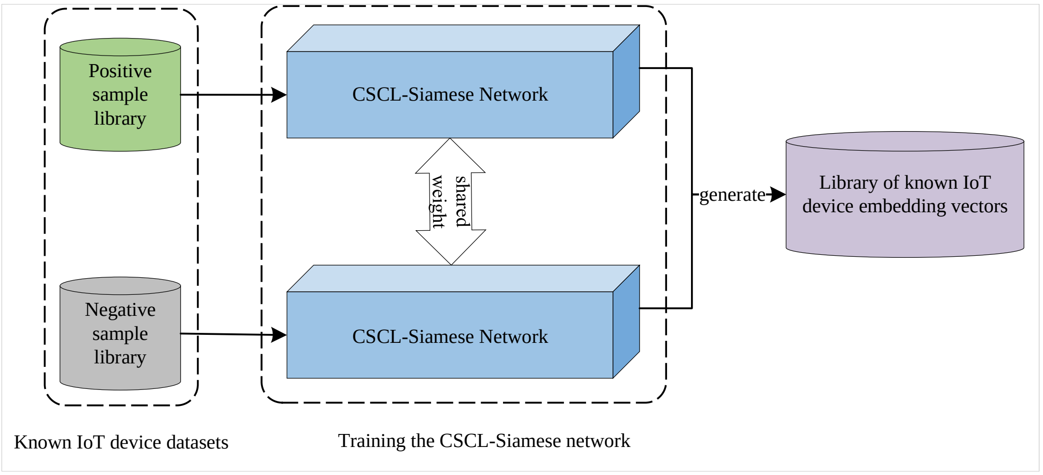


Figure 5. Structure of embedded vector database generation based on CSCL-Siamese network.

3.4.1. PNSP Algorithm

The author of [6] indicates that Siamese networks can effectively reduce the necessity for frequent retraining and require only a few sample pairs for effective training. Therefore, we adopted the Siamese network as the network model.

The CNN is adopted to hierarchically extract spatial patterns from grayscale image fingerprints (1×64 pixels), where small convolutional kernels (1×4) capture localized byte correlations (e.g., protocol headers) and pooling layers enhance positional robustness. While IoT traffic inherently exhibits temporal dependencies, the hybrid CNN+BiLSTM architecture synergizes spatial feature learning (via CNN) with sequential modeling (via BiLSTM), addressing both static and dynamic characteristics. This design aligns with prior studies [7,10] that validate CNNs for traffic-derived image classification. To ensure IoT compatibility, a lightweight CNN (8 filters) and low-dimensional embeddings (256D) minimize computational overhead.

As shown in Figure 6, a single subnetwork of the defined Siamese network comprises three layers. The first layer is a CNN layer for learning the image features of the samples. This layer comprises convolutional and pooling layers. The convolutional layer has eight 1×4 filters and utilizes the rectified linear unit activation function for nonlinear transformation. The pooling layer employs max pooling to reduce the size of the feature map while retaining critical information. Next, considering the temporal characteristics of the traffic data of IoT devices [14], the second layer utilizes a bidirectional long short-term memory (BiLSTM) network to extract temporal features. The final layer is a dense layer containing 256 neurons, which outputs the embedding vectors. This layer does not employ any activation function to preserve the original scale of the output embedding vectors, thus providing accurate vector representations for subsequent similarity measurements.

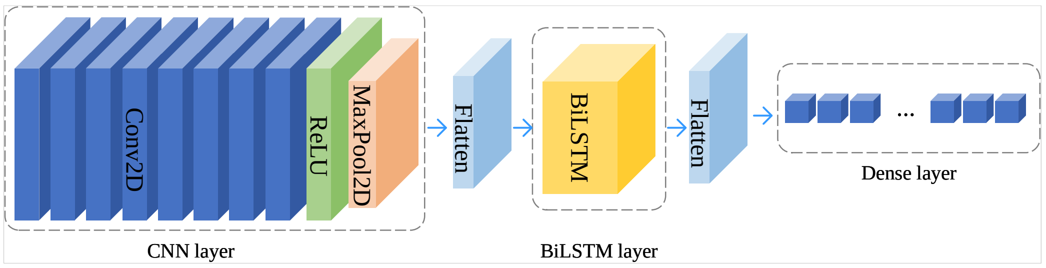


Figure 6. Structure of a single subnetwork of the Siamese network.

3.4.2. CSCL Function

To accurately measure the similarity between IoT devices, a contrastive loss function is typically employed to guide the training process of the Siamese network. This function trains the Siamese network by minimizing the contrastive loss, thereby generating compact and dispersed interclass embedding vectors [15]. However, the PNSP algorithm proposed in Section 3.3.3 causes a significant imbalance: the number of negative sample pairs will significantly exceed that of positive ones. If we continue using the contrastive loss function as the learning objective for the Siamese network, then the loss for positive and negative sample pairs will retain the same weighting factors, thus causing the identification boundary of the Siamese network to skew toward negative sample pairs and degrade the recognition performance for the same IoT device. In IoT device identification, distinguishing between identical and different devices is important. Therefore, the classical contrastive loss function is inadequate for processing PNSPs.

To mitigate the bias in the decision boundary toward negative sample pairs due to the insufficient number of positive sample pairs, we propose a CSCL function that combines cost-sensitive weighting factors [16] with the contrastive loss function [15]. The CSCL function comprises two components, as presented in Equation (3). The left section represents the loss for positive sample pairs, whereas the right section denotes the loss for negative sample pairs. Here, W represents the network weights; e_a and e_b denote the embedding vectors for samples a and b, respectively. The indicator y_{ab} indicates whether e_a and e_b belong to the same class; if e_a and e_b belong to the same class, then $y = 1$; otherwise, $y = 0$. The threshold m denotes the minimum acceptable distance among negative sample pairs. First, the model weights W are obtained by training the Siamese network. Let d_w represent the Manhattan distance between e_a and e_b to assess the similarity between the two embedding vectors. It can be calculated using Equation (4), where i is the feature dimensions of the embedding vectors. Next, the disparity in the number of positive and negative sample pairs is utilized to compute the cost-sensitive weighting factors. The loss of positive sample pairs λ_{po} and that of negative sample pairs λ_{ne} are calculated using Equations (5) and (6), respectively. Here, n_{po} and n_{ne} represent the counts of positive and negative sample pairs, respectively. Since $n_{po} \ll n_{ne}$, then $\lambda_{po} \gg \lambda_{ne}$. This condition enhances the sensitivity of the Siamese network to the loss of the positive sample pairs to that of the negative sample pairs, i.e., $\lambda_{po} y_{ab} d_w^2 >> \lambda_{ne}(1 - y_{ab})\{\max(0, m - d_w^2)\}$, which consequently increases the focus on the positive sample pairs.

$$L(W,(e_a,e_b,y_{ab})) = \lambda_{po} y_{ab} d_w^2 + \lambda_{ne}(1-y_{ab})\{\max(0,m-d_w^2)\} \tag{3}$$

$$d_w = \sum_{i=0}^{255} |e_{ai} - e_{bi}| \tag{4}$$

$$\lambda_{po} = \frac{n_{ne}}{n_{po}+n_{ne}} \tag{5}$$

$$\lambda_{ne} = 1 - \lambda_{po} = \frac{n_{po}}{n_{po}+n_{ne}} \tag{6}$$

Based on Equations (3)–(6), when $y_{ab} = 1$, i.e., the sample pairs are similar, the loss is $L = \lambda_{po} d_w^2$. In this case, we are only required to minimize d_w. When $y_{ab} = 0$, i.e., the sample pairs are different, and minimizing the loss is equivalent to maximizing $\lambda_{ne}\{\max(0, m - d_w^2)\}$. In this case, an extremely small value of d_w suggests that the embedding vectors of negative sample pairs are closely spaced, thus warranting an increased penalty $\lambda_{ne}(m - d_w^2)$ to reduce the loss. Meanwhile, a large value of d_w indicates that the embedding vectors of negative sample pairs are distant from each other. If this value exceeds the threshold m, then no minimization is required and the loss L can be set to zero. Thus, the objective of minimizing the CSCL function can be expressed as shown in Equation (7),

$$W = \arg\min \sum_{j=1}^{n} L(W,(e_{ja},e_{jb},y_j)) \tag{7}$$

where, n denotes the total number of positive–negative sample pairs.

3.4.3. CSCL-Siamese Network

Combining Sections 3.4.1 and 3.4.3, we propose a Siamese network based on CSCL (CSCL-Siamese). The structure of the CSCL-Siamese network is illustrated in Figure 7. First, the Siamese network comprises two identical subnetworks that share weights. The structure of each subnetwork is shown in Figure 6. This network accepts two samples at once and generates two 256-dimensional embedding vectors for each sample. Next, it calculates the Manhattan distance between the embedding vectors and passes this distance value to the CSCL function to compute the loss. Finally, the calculated loss is backpropagated to perform gradient updates on the parameters of the Siamese network.

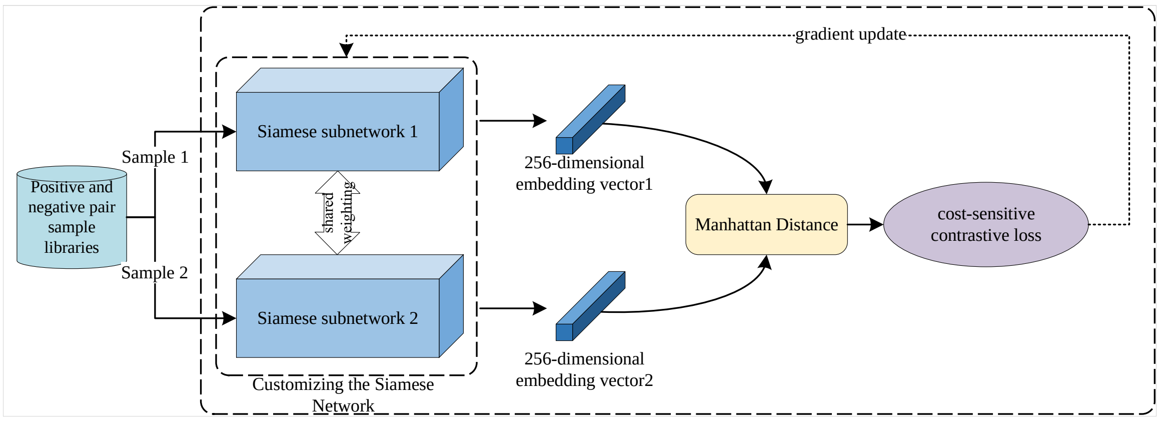

gradient update
Positive and negative pair sample libraries
Sample 1
Sample 2
Siamese subnetwork 1
shared weighting
Siamese subnetwork 2
Customizing the Siamese Network
256-dimensional embedding vector1
256-dimensional embedding vector2
Manhattan Distance
cost-sensitive contrastive loss

Figure 7. Structure of the CSCL-Siamese network

3.4.4. Embedding-Vector Database Generation Based on CSCL-Siamese Network

A previous study [4] showed that the embedding vectors of unknown and known IoT devices differ significantly. To obtain the embedding vectors of all unknown IoT devices, one must construct an embedded vector database for known IoT devices in advance for filtering the embedding vectors of unknown devices. The main steps for generating the embedded vector database are as follows: First, the CSCL-Siamese network is trained using the set of PNSPs and the trained model is saved. At this stage, the CSCL-Siamese network can generate compact embedding vectors for similar IoT devices while dispersing those for dissimilar devices, thereby enabling the identification of known IoT devices. Subsequently, traffic data from known IoT devices are input into the saved CSCL-Siamese network to extract and store the embedding vectors for each device. Thus, when new traffic data arrive, the corresponding embedding vector can be extracted using the CSCL-Siamese network and compared against the embedded vector database using a similarity threshold. Thus, the embedding vectors that do not match the known IoT devices are filtered out promptly—they are the embedding vectors of unknown IoT devices.

3.4.5. EVDC-CSCL-Siamese Algorithm

Based on the study above, we propose the EVDC-CSCL-Siamese algorithm. The pseudocode for the EVDC-CSCL-Siamese algorithm is described in Algorithm 4. The input comprises a set of PNSPs, i.e., Po and Ne. Before training the CSCL-Siamese network, the positive–negative sample pairs are randomly segmented into a training set $X_{tr} \leftarrow (Po_{tr}; Ne_{tr})$, validation set $X_{va} \leftarrow (Po_{va}; Ne_{va})$, and test set $X_{te} \leftarrow (Po_{te}; Ne_{te})$ at a 6:2:2 ratio. The output includes the trained CSCL-Siamese network model M, the identification results $\hat{Y}$ for known IoT devices, and the embedded vector database ED for known IoT devices.

Algorithm 4 EVDC-CSCL-Siamese Algorithm

```
Input: $Po \leftarrow (po_1, \ldots, po_i, \ldots, po_f); Ne \leftarrow (ne_1, \ldots, ne_i, \ldots, ne_f)$ // Full permutation of positive
    and negative pairs of samples
Output: $M; \hat{Y}; ED$
 1: Divide $(Po; Ne)$ into $X_{tr}; X_{va}; X_{te}$
 2: Create $CSCL - Siamese \leftarrow$ CNN + Flatten + BiLSTM + Flatten + Dense
 3: $n_{po} \leftarrow len(Po); n_{ne} \leftarrow len(Ne)$
 4: Use Equation (5) to calculate $\lambda_{po}$ // Use of Equation (5)
 5: Use Equation (6) to calculate $\lambda_{ne}$ // Use of Equation (6)
 6: Set model parameters Epoch, Batch Size, and learning rate, etc.
 7: for each $epoch \leftarrow$1 to Epoch do
 8:     for each Batch Size in $X_{tr}$ do
 9:         for each $(x_a; x_b)$ in $X_{tr}$ do
10:             $(e_a; e_b) \leftarrow CSCL - Siamese(x_a; x_b)$
11:             Use Equation (4) to calculate $d_w$ // Use of Equation (4)
12:             Use Equation (3) to calculate $L$ // Use of Equation (3)
13:             $L.backward$ // Reverse Update Model Parameters
14:         end for
15:     end for
16:     $CSCL - Siamese(X_{va})$ // Input of validation sets into the CSCL-Siamese network
17: end for
18: save $M$ // The training generates the model M
19: $s_i \leftarrow []$
20: $\hat{Y} \leftarrow M(X_{te})$ // Known IoT device identification results
21: $ED \leftarrow M(Po; Ne)$ // Calling Model M to Generate a Database of Known IoT Device Embedding Vectors
22: return $M; \hat{Y}; ED$
```

The computational overhead of Algorithm 4 primarily originates from training the CSCL-Siamese network model. Because the number of training epochs and batch size are constant, the time complexity of the BiLSTM model at any time is $O(n^2)$. Thus the total time complexity of Algorithm 4 mainly results from here.

3.5. WVE-UDI: UDI Algorithm Based on WVE

To fully leverage the clustering potential of multiple clustering algorithms for clustering the filtered embedding vectors of all unknown IoT devices, we combined weighting factors using a voting ensemble approach to identify unknown IoT devices. This method aims to overcome the limitations of individual clustering algorithms and enhance the identification capability for unknown IoT devices. The process is categorized into two main stages: (1) filtering the embedding vectors of unknown IoT devices and (2) identifying unknown IoT devices through WVE. Each stage is discussed in detail below.

3.5.1. Filtering of Embedding Vectors for Unknown IoT Devices

Because the embedding vectors of unknown IoT devices typically differ significantly from those of known IoT devices [6], calculating the similarity between these two types of embedding vectors allows one to filter the embedding vectors of all unknown IoT devices from the numerous embedding vectors. To accurately quantify this similarity difference, we utilized the Euclidean distance [17] as the similarity metric, which is calculated as shown in Equation (8). The filtering process for the embedding vectors of unknown IoT devices is illustrated in Figure 8. When new traffic data arrive, they are first subjected to data preprocessing. Next, Algorithm 2 (the improved Nilsimsa algorithm) is called to generate a 512-bit hexadecimal hash value, which is then converted to a grayscale image using the Pillow library. Subsequently, the corresponding embedding vectors are generated using the trained and saved CSCL-Siamese network model. Finally, based on the Euclidean distance, these embedding vectors are compared with those in the embedded vector database for known IoT devices, where those that are below a predetermined similarity threshold, i.e., the embedding vectors of unknown IoT devices, are filtered out.

$$d_{\min} = \sqrt{\sum_{i=0}^{255} (e_{ai} - e_{bi})^2} \tag{8}$$

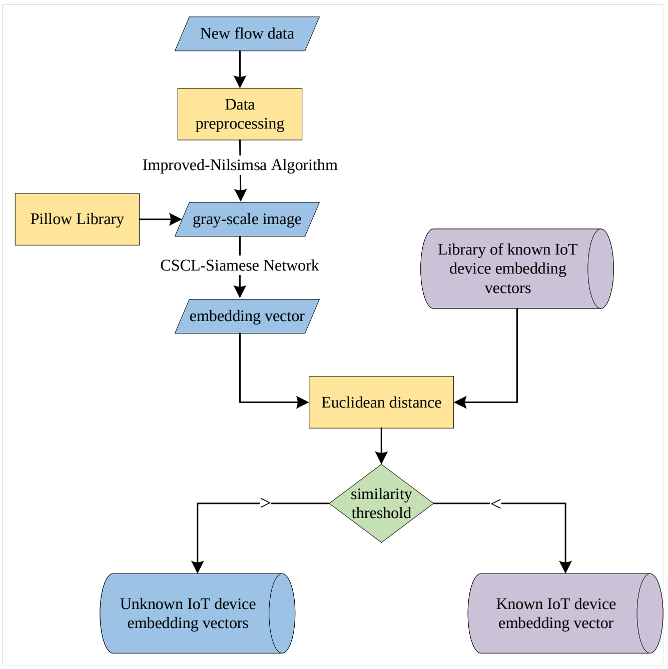

New flow data
Data preprocessing
Improved-Nilsimsa Algorithm
Pillow Library
gray-scale image
CSCL-Siamese Network
Library of known IoT device embedding vectors
embedding vector
Euclidean distance
similarity threshold
>
<
Unknown IoT device embedding vectors
Known IoT device embedding vector

Figure 8. Filtering flowchart for unknown IoT device embedding vectors.

3.5.2. WVE for UDI

To fully exploit the clustering potential of multiple unsupervised clustering algorithms, we introduce weighting factors to improve the existing voting ensemble strategy [18]. By integrating the clustering results of three different unsupervised clustering algorithms, we aim to achieve clustering-ensemble identification of multiple unknown IoT devices. The main steps are illustrated in Figure 9.

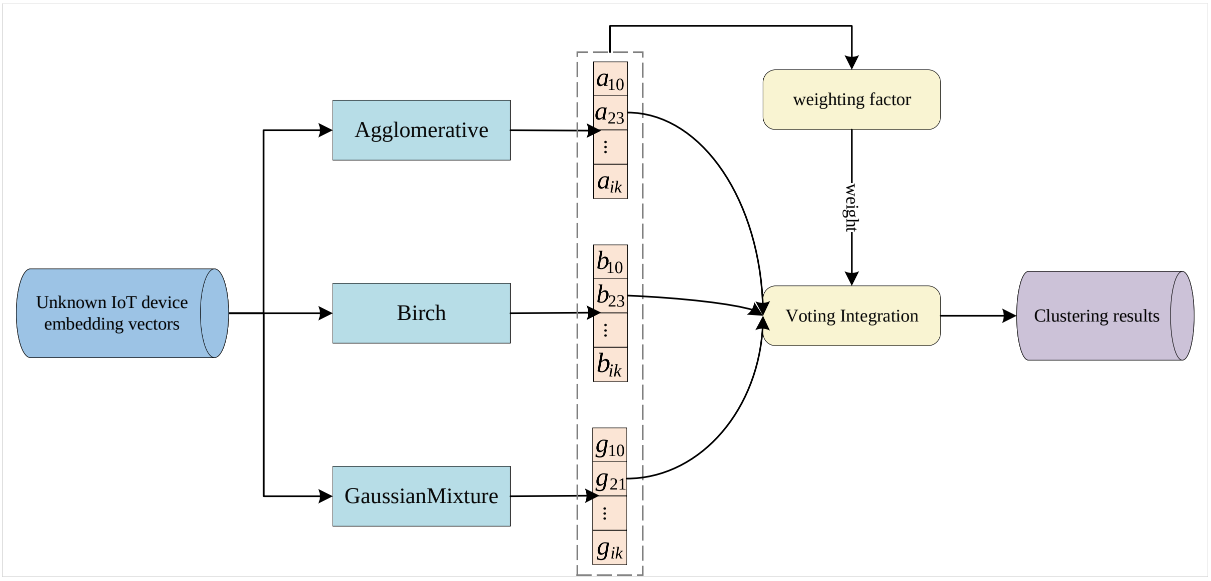


Figure 9. Flowchart of weighted-voting clustering integration to identify unknown IoT devices.

In the first step, the embedding vectors of unknown IoT devices obtained from the filtering process in Section 3.5.1 are clustered sequentially using agglomerative, BIRCH, and Gaussian mixture algorithms, and their respective clustering results are obtained. Both the agglomerative [19] and Birch [20] algorithms do not require specifying the number of clusters, and the Gaussian mixture algorithm [21] determines the optimal number of clusters using the elbow method. The clustering result of the agglomerative algorithm is $a_{10}, a_{23,} \ldots, a_{ik}$, the result of the Birch algorithm is $b_{10}, b_{23,} \ldots, b_{ik}$, and the result of the Gaussian mixture algorithm is $g_{10}, g_{21,} \ldots, g_{ik}$. Here, a_{ik} indicates that the i-th embedding vector of the unknown IoT device is clustered into class k (i.e., label k) by the agglomerative algorithm, and so on. In the second step, weights are assigned to the clustering results of each algorithm. The Gaussian mixture algorithm utilizes Gaussian distributions to accurately capture the data distribution and can flexibly reflect the actual structural characteristics of the data [21]. Therefore, it is assigned a higher weight to emphasize its critical contribution to the clustering results. Thus, weight factors of 0.2, 0.2, and 0.6 are assigned to the agglomerative, BIRCH, and Gaussian Mixture algorithms, respectively. In the third step, the clustering labels are integrated. For each sample in the clustering results, the weight of each label is calculated based on the clustering labels obtained from the three individual clustering algorithms and their corresponding weighting factors. For example, if the agglomerative algorithm clusters the i-th embedding vector of an unknown IoT device into class k, then its weighted value $w_{a_{ik}}$ is calculated using Equation (9). The label with the highest weighted value is selected as the final integrated label for that sample. During this process, the management of potential label conflicts, where multiple clustering algorithms may assign the same embedding vector to different classes with identical weighted values, is prioritized. For instance, if the agglomerative algorithm clusters the second embedding vector of an unknown IoT device into class 3, the BIRCH algorithm into class 3, and the Gaussian mixture algorithm into class 1, but all weighted values are 0.6, then the Gaussian mixture algorithm's class is prioritized, i.e., class 1 is selected as the final integrated label. This prioritization is because the Gaussian mixture algorithm was assigned a higher weight in the previous weight assignment, thus indicating its greater significance in the identification results.

$$w_{a_{ik}} = 0.2 \cdot a_{ik} \tag{9}$$

3.5.3. WVE-UDI Algorithm

Based on the process above, we propose the WVE-UDI algorithm. The pseudocode for the WVE-UDI algorithm is described in Algorithm 5. The inputs include new traffic data X_u (which contains traffic data from unknown IoT devices), the embedded vector database ED of known IoT devices, the trained and saved CSCL-Siamese network model M, and a preset similarity threshold θ. The output is the identification result of the unknown IoT devices $\hat{Y}_u$.

Algorithm 5 WVE-UDI algorithm

```
Input: $X_u; ED; M; \theta$
Output: $\hat{Y}_u$ // Unknown IoT device identification results
1: for each $x_u$ in $X_u$ do
2:     $NS \leftarrow$ call Data Preprocess($x_u$) // Call Algorithm 1 to perform data preprocessing on new traffic data
3: end for
4: for each $ns$ in $NS$ do
5:     $h_{ns} \leftarrow$ call improved-Nilsimsa($x_u$) // Call Algorithm 2 to generate a 512-bit hexadecimal hash value
6:     $g_{ns} \leftarrow h_{ns}$ use Pillow // Generating grayscale images with the Pillow library
7:     $G_{ns} \leftarrow g_{ns}$ // Get a sample set of grayscale images
8: end for
9: $NED \leftarrow M(G_{ns})$ // Get all embedding vectors for new traffic data
10: for each $ned$ in $NED$ do
11:     for each $ed$ in $ED$ do
12:         $d_{\min} \leftarrow$ Use Equation (8) to calculate and find out minimum // Calculate the minimum similarity distance using Equation (8)
13:         if $d_{\min} < \theta$ then then // Comparison of Similarity Thresholds
14:             $UED \leftarrow ned$ // Save filtered embedding vectors for all unknown IoT devices
15:         end ifUnknown IoT Device Identification Models and Algorithms based on CSCL-Siamese Networks and Weighted-Voting Clustering Ensemble
16:     end for
17: end for
18: $A \leftarrow Agglomerative(UED)$ // Agglomerative identification results
19: $B \leftarrow Birch(UED)$ // Birch Recognition Results
20: $G \leftarrow GaussianMixture(UED)$ // GaussianMixture Recognition Results
21: $W_A \leftarrow 0.2; W_B \leftarrow 0.2; W_G \leftarrow 0.6$ // Assignment of weights
22: for each $a_{ik_a}$ in $A$ do
23:     for each $b_{ik_b}$ in $B$ do
24:         for each $g_{ik_g}$ in $G$ do
25:             $labels\_weight \leftarrow \max(W_A \cdot a_{ik_a}; W_B \cdot b_{ik_b}; W_G \cdot g_{ik_g})$
26:             if $len(labels\_weight) \neq 1$ then then // Label conflicts after weighting
27:                 $k \leftarrow g_{ik_g}$
28:             else$k \leftarrow labels\_weight$
29:             end if
30:             $\hat{Y}_u \leftarrow k$ // Final identification results
31:         end for
32:     end for
33: end for
34: $\hat{Y}_u$
```

The computational overhead of Algorithm 5 primarily originates from the grayscale image fingerprint-sample generation, the embedding-vector filtering of unknown IoT devices, and the clustering-ensemble identification. The generation of grayscale-image fingerprint samples involves cleaning the data sample set and converting each sample into a grayscale image, which results in a time complexity of $O(n)$. The embedding-vector filtering of unknown IoT devices requires iterating through each embedding vector and comparing its similarity distance with the embedded vector database of known IoT devices, which results in a time complexity of $O(n^2)$. The clustering-ensemble identification requires integrating the clustering results of the three clustering algorithms, with a time complexity of $O(n^3)$. Thus, the total time complexity of Algorithm 5 mainly results from the three clustering algorithms.

By integrating Algorithms 1–5, we propose the CSCL-WVE-UDI method (unknown IoT device identification method based on CSCL-Siamese networks and WVE). The workflow of the CSCL-WVE-UDI method is as follows: First, Algorithm 1 is called to preprocess the raw traffic data X_{row} of known IoT devices, thus resulting in payload data S. Subsequently, Algorithms 2 and 3, specifically the improved Nilsimsa and PNSP algorithms, are invoked to transform the payload data S into PNSPs Po and Ne. Next, Algorithm 4, i.e., the EVDC-CSCL-Siamese algorithm, is applied to train the CSCL-Siamese network based on the positive–negative sample pairs Po and Ne as well as obtain the model M, thereby achieving the identification of known IoT devices. Additionally, the algorithm is used to construct the embedded vector database ED for known IoT devices. Finally, Algorithm 5, i.e., the WVE-UDI algorithm, is employed to integrate three unsupervised clustering algorithms to identify unknown IoT devices. The pseudocode of the CSCL-WVE-UDI method is described in Algorithm 6. The inputs include raw traffic data X_{row} from known IoT devices and new raw traffic data X_u containing traffic data from unknown IoT devices. The output is the identification results of unknown IoT devices $\hat{Y}_u$.

Algorithm 6 CSCL-WVE-UDI approach

```
Input: $X_{row}; X_u$
Output: $\hat{Y}_u$
1: for each $x_i$ in $X_{row}$ do
2:     $s_i \leftarrow$ call Data Preprocess $(x_i)$ // Call Algorithm 1 for data preprocessing
3:     $S \leftarrow s_i$ // Obtaining payload sample sets
4: end for
5: $(Po; Ne) \leftarrow$ call PNSP $(S)$ // Call Algorithm 3 (Algorithm 2 is called in Algorithm 3) to generate fully
   aligned positive and negative pair samples
6: $ED; M; \hat{Y} \leftarrow$call EVDC-CSCL-Siamese$(Po; Ne)$ // Calling Algorithm 4
7: Set threshold $\theta$ // Setting the similarity threshold
8: $\hat{Y}_u \leftarrow$call WVE-UDI$(X_u; ED; M; \theta)$ // Calling Algorithm 5 for unknown IoT device identification
9: return $\hat{Y}_u$
```

The computational overhead of the CSCL-WVE-UDI method is primarily derived from Algorithms 1–5. The time complexities of Algorithms 1–5 are $O(n^2)$, $O(n)$, $O(n^4)$, $O(n^3)$, and $O(n^3)$, respectively. Thus, the total time complexity of the CSCL-WVE-UDI method mainly results from here. The symbols used in this paper are shown in Table 1.

Table 1. Symbol Definitions.

Symbol	Definition	Symbol	Definition
x_i	Raw traffic sample of the i-th IoT device	s_i	Preprocessed payload data of the i-th device
h_j	512-bit hexadecimal hash value from improved Nilsimsa	g_i^j	Grayscale image fingerprint of the j-th sample from device i

Table 1. *Cont.*

Symbol	Definition	Symbol	Definition
Po	Set of positive sample pairs	Ne	Set of negative sample pairs
W	Network weight parameters	e_a, e_b	256D embedding vectors of samples *a* and *b*
y_{ab}	Label for sample pair (1=same class, 0=different class)	λ_{po}	Weighting factor for positive pairs: $\frac{n_{ne}}{n_{po}+n_{ne}}$
λ_{ne}	Weighting factor for negative pairs: $\frac{n_{po}}{n_{po}+n_{ne}}$	d_w	Manhattan distance: $\sum \lVert e_{ai} - e_{bi} \rVert$
m	Minimum acceptable distance threshold for negative pairs	L	CSCL loss function
ED	Embedded vector database for known devices	NED	Embedded vectors of new traffic data
$d_{\min}$	Minimum Euclidean distance	θ	Similarity threshold for filtering unknown devices
A, B, G	Clustering results (Agglomerative/BIRCH/GaussianMixture)	W_A, W_B, W_G	Weights for clustering algorithms (0.2/0.2/0.6)
a_{ik}	Cluster label of sample *i* by Agglomerative	$\hat{Y}_u$	Final identification results of unknown devices
TP, TN	True positive/negative counts	FP, FN	False positive/negative counts
G-mean	Geometric mean of TPR and TNR	n_{po}, n_{ne}	Number of positive/negative sample pairs

4. Experimental Results and Analysis

4.1. Experimental Datasets

The experiments were conducted on the Aalto [22], UNSW [23], and LSIF datasets [24], obtained from IoT device traffic data during device installation, user interaction, and idle states, respectively. As Siamese networks rely on the relative relationships between samples rather than absolute labels, only a small amount of data is required for effective learning. Thus, up to 50 payload samples were randomly selected from each IoT device to construct a set of PNSPs. The specific number of samples for each device is shown in Table 2. To ensure fairness, all subsequent comparison experiments considered 50 samples for each IoT device. In addition, in order to facilitate the unknown IoT device identification experiments, three IoT devices were randomly excluded as unknown IoT devices from the training set beforehand.

Table 2. Experimental datasets.

Dataset Name	Number of Positive Samples	Number of Negative Pair Samples
Aalto	8008	25,968
UNSW	11,000	46,000
LSIF	8800	28,800

4.2. Evaluation Indicators

Since the training data samples of the CSCL-Siamese network are positive–negative pairwise relations, it can be regarded as a binary classification task, i.e., the correctly predicted positive–negative pairwise relations are treated as the positive class, and vice versa as the negative class. Therefore, the predictors are defined as shown in Table 3.

Table 3. Definitions of forecasting indicators.

	Positive Class	Negative Class
Predicted	TP (Positive class predicted to be positive)	FP (Negative class predicted to be positive)
Unpredicted	FN (Positive class predicted to be negative)	TN (Negative class predicted to be negative)

The experiment mainly uses Accuracy, Precision, Recall, and F1 value as the evaluation metrics. *Accuracy* measures the accuracy of the overall forecast, *Precision* measures the correctness of positive class predictions, *Recall* measures the predictive coverage of the positive class, and the F1 value integrates the trade-off between Precision and Recall. Their respective formulas are expressed below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = \frac{2 \times \Pr ecision \times \text{Re}call}{\Pr ecision + \text{Re}call} \tag{13}$$

Since the number of negative pair samples is much larger than positive pair samples, and the above metrics mainly focus on the prediction performance of positive pair samples, the G-mean metric is further introduced as an evaluation metric. The G-mean metric measures not only the ability to correctly identify positive pairs of samples (true positive rate TF), but also takes into account the ability to correctly identify negative pairs of samples (true negative rate TN), thus providing a more comprehensive assessment of the unbalanced set of positive and negative pairs of samples. The formula is expressed as follows:

$$G - mean = \sqrt{\frac{TP}{TP + FN} \frac{TN}{TN + FP}} \tag{14}$$

4.3. Experimental Environment and Experimental Parameter Settings

The hardware and software configurations used for the experiment are listed in Table 4, and the hyperparameter settings are listed in Table 5, where Optimizer denotes the optimizer, Learning rate denotes the learning rate, Batch size denotes the number of training samples per batch, and Epochs denotes the number of training rounds. In order to fully evaluate the effectiveness and efficiency of the proposed method, the relevant hyperparameters for adaptive reduction of learning rate are introduced: Patience is the automatic reduction of learning rate when the validation loss still does not decrease after 3 Epochs; Factor is the learning rate that is reduced by 0.1 times each time; and Min learning rate is the lower limit of the learning rate, i.e., the learning rate can be reduced to the Min learning rate is the lower limit of learning rate, i.e., the lowest value that learning rate can be reduced to.

Table 4. Experimental environment configuration.

Software and Hardware Type	Value
System	Windows10
CPU	AMD Ryzen 7 5800H
GPU	NVIDIA GeForce RTX 3060
Programming language	Python3.8
Deep Learning Framework	Keras2.4.3

Table 5. Hyperparameter settings for the base learner.

Parameter Name	Parameter Value
Optimizer	Adam
Learning rate	0.001
Batch size	64
Epochs	50
Factor	0.1
Patience	3
Min learning rate	0.0001

4.4. Experiments on the Efficiency of the CSCL-WVE-UDI Methodology

In order to verify the efficiency of the CSCL-WVE-UDI method, it is compared with the literature [4,5,7]'s methods, which are more effective in recognizing unknown IoT devices. This experiment is conducted in an environment where three unknown IoT devices coexist, where it is difficult for the literature [4,5]'s method to recognize multiple unknown IoT devices at one time; therefore, in order to achieve the recognition of three unknown IoT devices by the literature [4,5]'s method, separate recognition models need to be trained for each unknown IoT device. Considering that the network traffic in the IoT device–user interaction scenario is more representative [25], the UNSW dataset is chosen for this experiment. In addition, three unknown IoT devices are selected as the experimental objects for this experiment; firstly, because the literature [7] has referred to the practice of randomly selecting three unknown IoT devices from the UNSW dataset, and secondly, because the three unknown IoT devices not only ensure that the experiments are moderately complex, but also are sufficient to effectively compare the recognition effects of different methods [7]. The results of the comparison of the recognition accuracy of each method are shown in Figure 10.

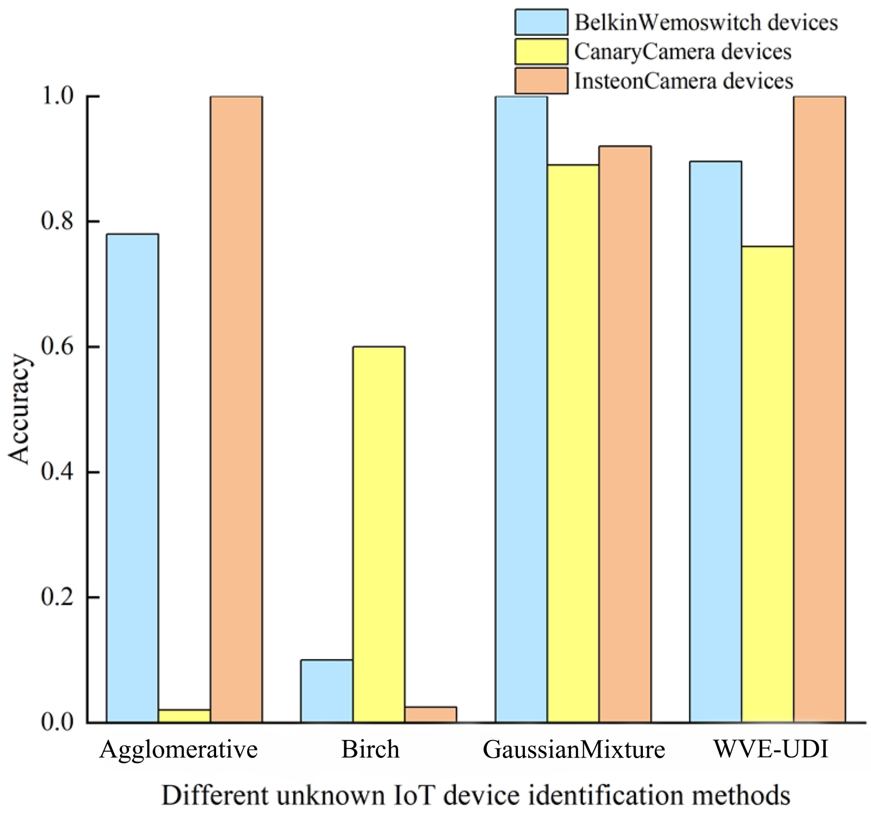


Figure 10. Recognition results of different methods on three unknown IoT devices [4,5,7].

As can be seen from Figure 10, in the scenario where the three unknown IoT devices coexist, the overall performance of the CSCL-WVE-UDI method is relatively better, and it is able to achieve a high recognition effect. In particular, it achieves 89.6% and 100% recognition accuracy for Belkin Wemo Switch device and Insteon Camera device, respectively, but there are still deficiencies, as follows: (1) Compared with literature [4,5], which uses the shortest similarity distance as a recognition method for unknown IoT devices, the recognition accuracy of the CSCL-WVE-UDI method is higher than that of both methods, with a maximum improvement of 97.5 percentage points. This is because literature [4,5] relies on the embedding vectors and similarity metrics generated by the traditional Siamese network, and if the traditional Siamese network is poorly trained, the similarity distances between the embedding vectors are too large, which affects the recognition effect; at the same time, there is a certain degree of similarity between the Canary Camera device and the Insteon Camera device itself, which further reduces the accuracy of the recognition. In addition, when facing N unknown IoT devices, the method proposed in literature [4,5] needs to train N recognition models, which will increase the computational overhead; (2) Compared to the KNN algorithm-based unknown IoT device identification method in literature [7], the CSCL-WVE-UDI method exhibits higher accuracy in identifying Insteon Camera devices, but is slightly less accurate in identifying Belkin Wemo Switch devices and Canary Camera devices, which are lower by 10 percentage points and 19 percentage points. The main reason for this difference is the supervised classification implementation based on the KNN algorithm in literature [7], which therefore has prior knowledge of the real labels and number of unknown IoT devices and uses this information for model training to achieve a high recognition accuracy. In contrast, the CSCL-WVE-UDI method is implemented by integrating multiple unsupervised clustering algorithms, which performs the recognition without real labels, thus resulting in lower recognition accuracies than literature [7] for Belkin Wemo Switch devices and Canary Camera devices. However, in open environments, the inability to obtain the real tags and the number of devices of unknown IoT devices beforehand makes it difficult to apply the method in literature [7] directly. In contrast, the CSCL-WVE-UDI method does not require prior knowledge of the true labels of unknown IoT devices, which makes it more adaptable.

In summary, it can be seen that the CSCL-WVE-UDI method proposed in this paper has good recognition ability in the environment where three unknown IoT devices coexist, and accomplishes the efficient recognition of unknown IoT devices. This experimental result proves that the CSCL-WVE-UDI method is suitable for network traffic-oriented unknown IoT device identification.

4.5. Comparative Experiments with PNSP Algorithm

In order to verify the effectiveness of the PNSP algorithm, this experiment compares the overall recognition effectiveness of the CSCL-Siamese network for known IoT devices with randomized strategy and PNSP algorithm generating positive and negative pairs of samples on the Aalto dataset, UNSW dataset, and LSIF dataset, respectively. Considering that there is an imbalance in the sample set of positive and negative pairs, this experiment uses the Accuracy rate and G-mean value as the main evaluation index. The experimental results are shown in Figure 11.

As can be seen from Figure 11, the overall performance of the PNSP algorithm outperforms the randomized strategy on all three different scenario datasets, especially in terms of accuracy, as follows: First, the accuracy of the PNSP algorithm improves by 18.8% on the Aalto dataset, 12.3% on the UNSW dataset, and 11.1% on the LSIF dataset, which suggests that the recognition ability based on the PNSP algorithm is able to classify the majority of the sample pairs more accurately even if there is an imbalance in the data; Second, the enhancement of the G-mean value of the PNSP algorithm is smaller than the enhancement of the Accuracy rate, which is because the G-mean value integrates the ability of correctly recognizing the positive pair of samples and the negative pair of samples, whereas the small number of positive pair of samples leads to a relatively low enhancement of the G-mean value; Finally, the overall recognition improvement of the PNSP algorithm is relatively large compared to the random strategy for generating positive and negative pairs of samples, thanks to the fact that the PNSP algorithm fully and comprehensively utilizes all the samples in the dataset for generating positive and negative pairs, thus proving the effectiveness of the PNSP algorithm.

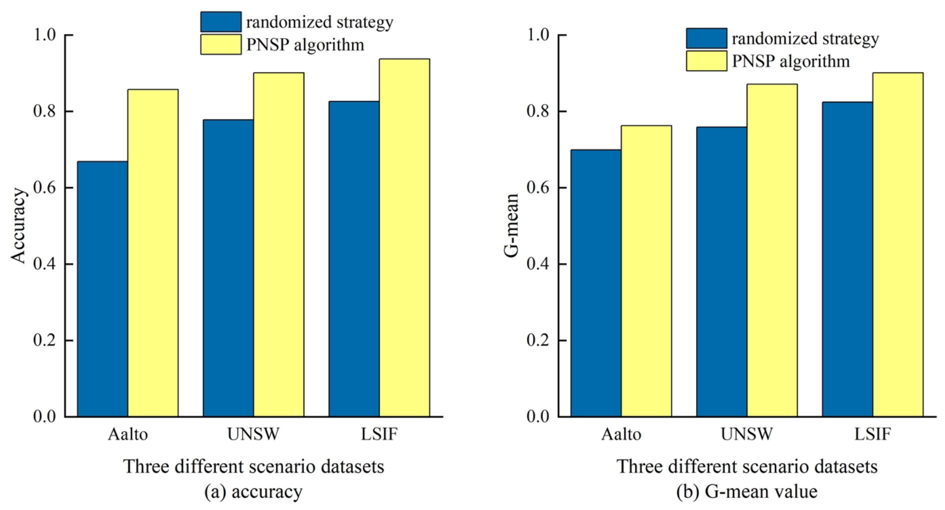
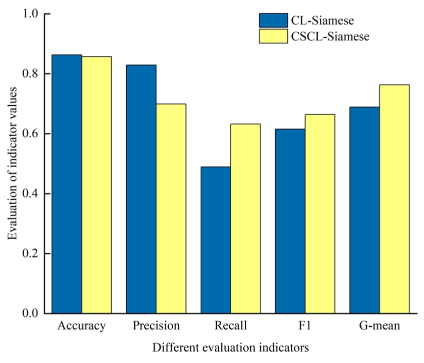


Figure 11. Comparison results of two algorithms for recognizing known IoT devices.

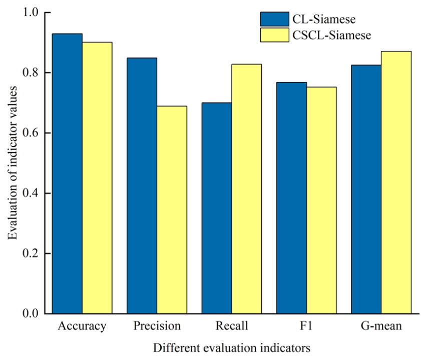
4.6. CSCL-Siamese Network Ablation Experiments

In order to verify that the cost-sensitive contrastive loss function (CSCL) can effectively alleviate the deficiency of the recognition boundary shift due to the small number of positive pairwise samples, this experiment compares the Siamese (Siamese network based on contrastive loss function, CL-Siamese) network and CSCL-Siamese network for recognizing known IoT devices. The results of the ablation experiments of the CSCL-Siamese network are shown in Figure 12.

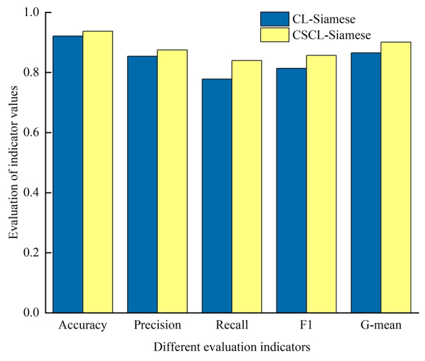
As can be seen from Figure 12, the overall performance of the CSCL-Siamese network proposed in this paper is relatively better on the three different scenario datasets, which can all achieve high G-mean values. First, on the Aalto and UNSW datasets, the CSCL-Siamese network improves the ability to recognize positive pairs of samples by improving the traditional contrast loss function, leading to Recall improvements of 14.3% and 12.8%, respectively. However, this improvement also brought about a decrease in the Precision rate, 13% and 16%, respectively, as more negative pair samples were misclassified as positive pairs. Second, on the Aalto dataset and the UNSW dataset, the slight decrease in recognition accuracy (up to 2.8 percentage points) is exchanged for a larger increase in G-mean value (up to 7.4 percentage points), which indicates that the CSCL-Siamese network has made significant progress in balancing the recognition of both positive and negative pairs of samples, and that the overall recognition performance has been effectively improved. Therefore, sacrificing some accuracy is worth it. In addition, by comparing Figure 12a–c, it can be seen that the recognition effect of the CSCL-Siamese network on the LSIF dataset is higher than that of the other two datasets, and all the evaluation metrics are better than that of the traditional CL-Siamese network, which on the one hand, indicates that the CSCL-Siamese network has a better recognition ability for IoT devices in the idle state of the traffic data, i.e., the LSIF dataset has a better recognition ability; on the other hand, it also verifies that the CSCL-Siamese network can alleviate the shortcoming of the recognition boundary shift due to the small number of positive pairs of samples, and has a certain degree of adaptability to the IoT device traffic data in different scenarios. In addition, in order to verify the superiority of the proposed cost-sensitive contrast loss function, compared to the traditional contrast loss function, in reducing the loss values in the set of PNSPs, the change of the loss values with the increase in the number of training rounds is further compared between the CL-Siamese network and the CSCL-Siamese network, as shown in Figure 13, where the blue dashed line shows the change in loss values for the CL-Siamese network and the yellow solid line shows the change in loss values for the CSCL-Siamese network.



(**a**) Aalto dataset



(**b**) UNSW dataset



(**c**) LSIF dataset

Figure 12. Experimental results of ablation of CSCL-Siamese network.

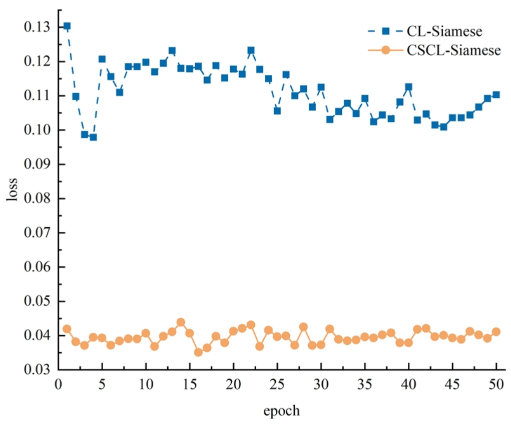


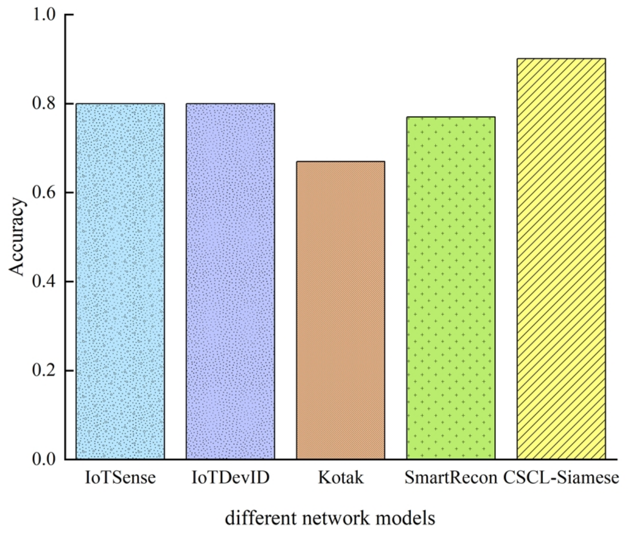
Figure 13. Plot of the change process of the loss value of the two networks.

As can be seen in Figure 13, the loss value of the CSCL-Siamese network has a more significant reduction compared to the CL-Siamese network. At the first iteration, the loss value of the CSCL-Siamese network has reached about 0.043, which is 8.8 percentage points lower compared to the CL-Siamese network. As the number of iterations increases, after 16 iterations, the loss value of the CSCL-Siamese network reaches a minimum point of only 0.034. However, in the subsequent iterations, the loss value of the CSCL-Siamese network appears to fluctuate back and forth with changes, which may be due to some stochastic factors in the training process or the model itself. Nevertheless, in terms of the overall trend, the CSCL-Siamese network still outperforms the CL-Siamese network in terms of the overall performance on the loss value, and is able to reduce the loss value effectively.

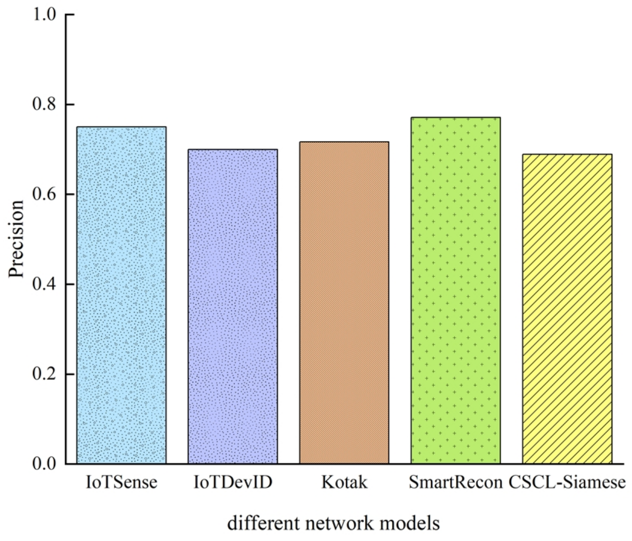
4.7. EVDC-CSCL-Siamese Algorithm Comparison Experiments

In order to validate the effectiveness of the EVDC-CSCL-Siamese algorithm in Section 3.4.5, this paper conducts a comparison experiment of known IoT device recognition on the classical UNSW dataset. Since the EVDC-CSCL-Siamese algorithm is completely dependent on the recognition accuracy of the proposed CSCL-Siamese network in constructing the database of known IoT device embedding vectors, this experiment only focuses on analyzing the experimental results of the CSCL-Siamese network. The results of the comparison experiments between the CSCL-Siamese network and the IoTSense [26], IoTDevID [27], Kotak [10], and SmartRecon [25] in the multiclassification recognition network are shown in Figure 14.

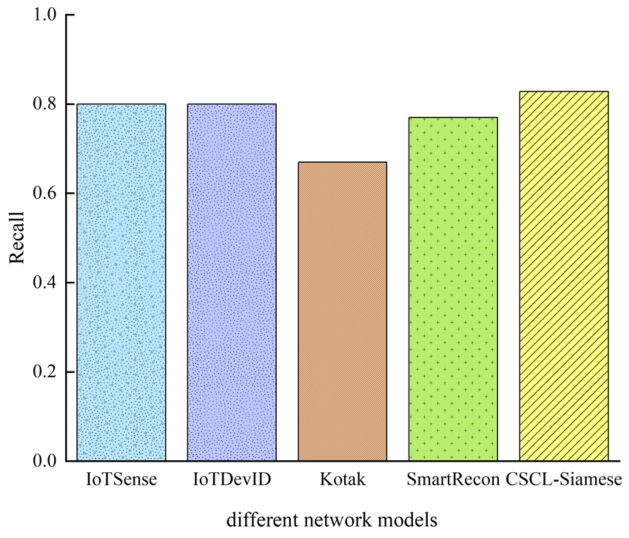
As can be seen in Figure 14a, the recognition accuracy of the CSCL-Siamese network reaches 90.1%, which is higher than that of the other four multiclassification networks. The main reason for this advantage is that the CSCL-Siamese network is essentially a kind of metric learning network structure, which requires only a small number of pairwise samples to achieve effective recognition, and the 50 data samples taken in this paper to construct a set of PNSPs. In this paper, 50 pairs of data samples are taken to construct the full arrangement of positive and negative samples, which is just in line with the characteristics of the CSCL-Siamese network; on the contrary, the remaining four recognition networks adopt the structure of multiclassification network, which usually requires a large number of training samples to achieve a higher accuracy, and theoretically, the more the amount of data, the more abundant the information, and the higher the accuracy of the recognition. Thus, the sample size of 50 data is relatively small for the large number of samples required for the structure of the multiclassification networks, leading to insufficient training and making the probability of misclassification of these multiclassification networks increase. Further, by comparing Figure 14b,c, it can be seen that compared to the multiclassification networks, the recognition Precision rate of the CSCL-Siamese network decreases, but the Recall rate is relatively high at 82.8%, which is higher than that of the multiclassification networks by up to 15.8 percentage points. This is due to the difference in recognition mechanisms between CSCL-Siamese networks and multiclassification networks. CSCL-Siamese networks focus on learning the similarity between data samples, whereas multiclassification networks focus on modeling through data samples to make classification decisions. This difference in mechanism results in CSCL-Siamese networks having a slightly lower recognition Accuracy than multiclassification networks. However, the advantage of higher Recall of CSCL-Siamese networks ensures that more IoT devices are correctly recognized. Therefore, although the recognition Precision rate of CSCL-Siamese network has decreased, its good Recall rate and high Accuracy rate can ensure reliable recognition performance in the field of IoT device recognition, and also verifies the effectiveness of the EVDC-CSCL-Siamese algorithm.



(**a**) Accuracy



(**b**) Precision



(**c**) Recall

Figure 14. Comparative experimental results of different network models.

4.8. WVE-UDI Algorithm Ablation Experiments

In order to validate the effectiveness of the WVE-UDI algorithm (in Section 3.5.3) in integrating multiple clustering algorithms, a series of unknown IoT device ablation comparison experiments were designed on the classic UNSW dataset, covering the presence of one, two, and three different unknown IoT device recognition scenarios, respectively, in an open environment. These experiments assess the effectiveness of the WVE-UDI algorithm in the unknown IoT device recognition task by comparing the difference in recognition accuracy of the WVE-UDI algorithm before and after the integration of the three clustering algorithms. In this experiment, the similarity threshold θ was set to 0.8. In addition, the WVE-UDI algorithm did not use labeled data in the clustering process, and only used real labels in calculating the recognition accuracy.

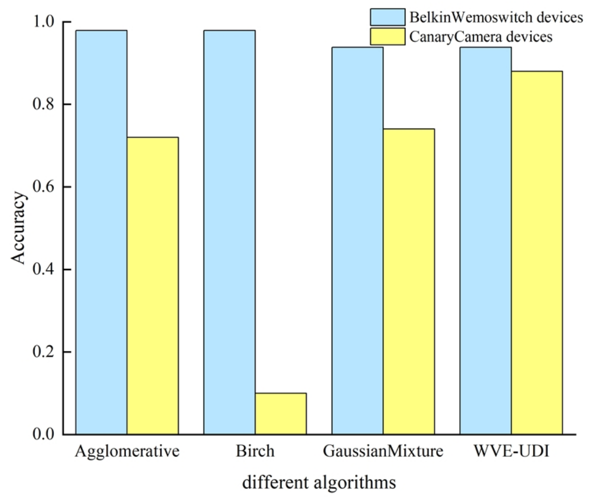
(1) The results of the ablation comparison experiments in the presence of an unknown IoT device in an open environment are shown in Table 6.

Table 6. Comparison results of recognition accuracy of individual unknown IoT devices.

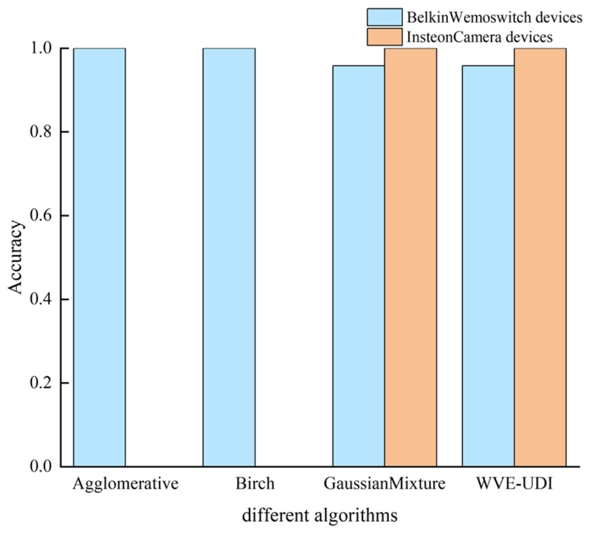
Unknown IoT Devices / Algorithms	Belkin Wemo Switch	Canary Camera	Insteon Camera
Agglomerative algorithm	100%	90%	100%
Birch algorithm	100%	90%	100%
GaussianMixture algorithm	100%	100%	100%
WVE-UDI algorithm	100%	90%	100%

As can be seen from Table 6, the recognition accuracy of the WVE-UDI algorithm is approximately the same before and after integration when only a single unknown IoT device exists in the open environment. First, for the Belkin Wemo Switch device and the Insteon Camera device, their recognition accuracies are 100% with or without integration, which is because when only one new unknown IoT device is added to the open environment, a single clustering algorithm is accurate enough to make the clustering labels of each clustering algorithm the same, which results in the WVE-UDI algorithm having the same recognition accuracy as the remaining three clustering algorithms after integration; Second, for the Canary Camera device, although the recognition accuracy of the individual clustering algorithms is 90%, 90%, and 100%, respectively, the WVE-UDI algorithm has an accuracy of 90%, which is not 100%, mainly because the recognition results of the Agglomerative algorithm and the Birch algorithm have larger values of incorrect labels, which leads to its weighted value being higher than the weighted value of the corresponding label of the WVE-UDI algorithm, which affects the final judgment of the WVE-UDI algorithm and makes the final integrated label tend to be the wrong label, thus pulling down the Accuracy rate. Nevertheless, the recognition accuracy of the WVE-UDI algorithm still reaches 90% and above for a single unknown IoT device, which indicates that the WVE-UDI algorithm is able to effectively integrate the clustering capabilities of multiple clustering algorithms in a single unknown IoT device environment.

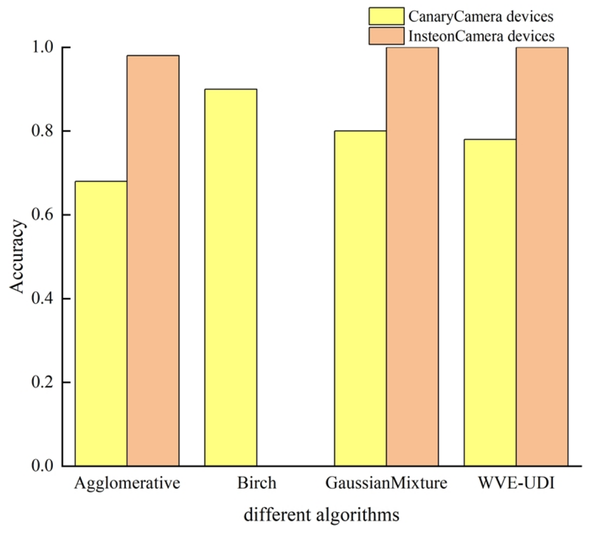
(2) The results of the ablation comparison experiment for the presence of two unknown IoT devices in the open environment are shown in Figure 15. In order to ensure the comprehensiveness of this comparison experiment, each unknown IoT device was separately identified for comparison between two and two.



(**a**)



(**b**)



(**c**)

Figure 15. Comparison results of recognition accuracy of two unknown IoT devices. (**a**) Presence of Belkin Wemo Switch devices and Canary Camera devices in the open environment; (**b**) Presence of Belkin Wemo Switch devices and Insteon Camera devices in the open environment; (**c**) Presence of Canary Camera devices and Insteon Camera devices in the open environment.

As can be seen from Figure 15, when two unknown IoT devices coexist in the open environment, the WVE-UDI algorithm is usually able to achieve more than 90% recognition accuracy, showing relatively good recognition results, as follows: (1) In the open environment where Belkin Wemo Switch and Canary Camera devices coexist, a single clustering algorithm often only achieves high accuracy for the Belkin Wemo Switch devices to achieve high accuracy recognition, while the recognition effect on Canary Camera devices is poorer, both reaching 80%. With the integration of the WVE-UDI algorithm, while the recognition accuracy of the Belkin Wemo Switch device decreased slightly to 93.8%, a decrease of 4.1 percentage points, the recognition accuracy of the Canary Camera device increased significantly to 88%, a maximum improvement of 78 percentage points. This shows that the WVE-UDI algorithm is able to balance the recognition effect among different IoT devices to some extent when integrating the clustering capabilities of multiple clustering algorithms; (2) Both the Agglomerative and Birch algorithms are unable to recognize Insteon Camera devices when Belkin Wemo Switch devices and Insteon Camera devices exist in the open environment. However, the WVE-UDI algorithm is able to achieve 100% accurate recognition of Insteon Camera devices, which is mainly attributed to the fact that the WVE-UDI algorithm successfully integrates the Gaussian Mixture algorithm's ability to recognize Insteon Camera devices, which significantly improves the final recognition results; (3) When both Canary Camera devices and Insteon Camera devices are present in the open environment, the Birch algorithm still fails to recognize the Insteon Camera devices, while the Canary Camera devices are recognized with 90% accuracy. Meanwhile, the remaining two single clustering algorithms perform opposite to the Birch algorithm, with low recognition accuracy for Canary Camera devices but high recognition accuracy for Insteon Camera devices. On the one hand, this is due to the fact that both Canary Camera devices and Insteon Camera devices are video cameras, and there are similarities in their data features, which leads to the single clustering algorithms facing difficulties in distinguishing between the two; on the other hand, the data features of the Insteon Camera devices may not be fully matched with the clustering patterns relied on by the Birch algorithm, which further affects the recognition effect of the algorithm. However, through the integration of the WVE-UDI algorithm, the recognition accuracy of the Canary Camera device reaches 78%, and the recognition accuracy of the Insteon Camera device jumps from 0% to 100%, successfully realizing the effective recognition of two unknown IoT devices. This proves the advantage of the WVE-UDI algorithm in integrating the recognition capability of multiple clustering algorithms, and achieves better recognition effect in the open environment where two unknown IoT devices coexist.

(3) The results of the ablation comparison experiments in the presence of three unknown IoT devices in an open environment are shown in Figure 16.

As can be seen in Figure 16, in the open environment where the three unknown IoT devices coexist, the WVE-UDI algorithm performs relatively well overall and is able to achieve high recognition results, as follows: (1) By comparing the Agglomerative algorithm and the WVE-UDI algorithm, it is found that the recognition effect is significantly improved by using the clustering integration, and although the recognition accuracy of the recognition accuracy of Belkin Wemo Switch devices decreased by 8.3 percentage points, the accuracy of Canary Camera devices increased from 72% to 76%, the accuracy of Insteon Camera devices especially increased from 0% to 100%, and the overall performance was effectively improved; (2) By comparing the Birch algorithm and the WVE- UDI algorithm, it is found that the Birch algorithm not only has a very low recognition accuracy of 10% for the Canary Camera device, but also still fails to recognize the Insteon Camera device, which once again shows that the clustering pattern relied on by the Birch algorithm does not match the data characteristics of the camera as a class of device exactly, resulting in its limited recognition capability. However, with the integration of the WVE-UDI algorithm, the recognition accuracies of the Canary Camera device and the Insteon Camera device are improved by 66 percentage points and 100 percentage points, respectively; (3) By comparing the GaussianMixture algorithm with the WVE-UDI algorithm, it is found that the recognition accuracies of the three types of IoT devices are enhanced, with the Belkin Wemo Switch device accuracy increased by 0.1 percentage points and Canary Camera device accuracy increased by 4 percentage points, and the WVE-UDI algorithm maintains the same level of recognition as the Insteon Camera device since the GaussianMixture algorithm has already achieved 100% accuracy; (4) A global comparison of the three single clustering algorithms and the WVE-UDI algorithm reveals that the Belkin Wemo Switch device's recognition accuracy decreased by 8.3 percentage points, the Canary Camera device improved by 66 percentage points, and the Insteon Camera device's accuracy improved by 100 percentage points; although the Belkin Wemo Switch device dropped slightly, the remaining IoT devices saw a substantial improvement in recognition accuracy. This small sacrifice for a significant improvement in the recognition rate of the other IoT devices is clearly worthwhile. This shows that the WVE-UDI algorithm is able to effectively integrate the clustering capabilities of multiple single clustering algorithms to improve the overall recognition effect.

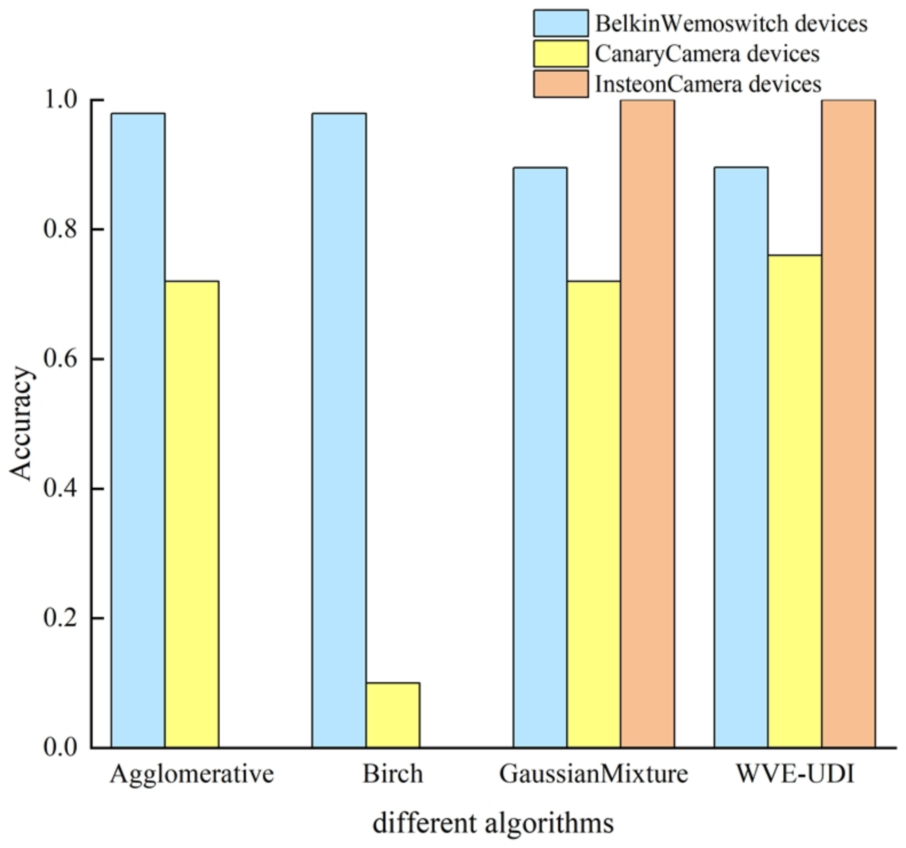


Figure 16. Comparison results of recognition accuracy of three unknown IoT devices.

In order to further analyze the specifics of the WVE-UDI algorithm integration in detail, the recognition confusion matrix of the WVE-UDI algorithm in the coexistence environment of the three unknown IoT devices is plotted, as shown in Figure 17, where each square denotes the recognition accuracy for each IoT device, and a darker color denotes a higher recognition accuracy.

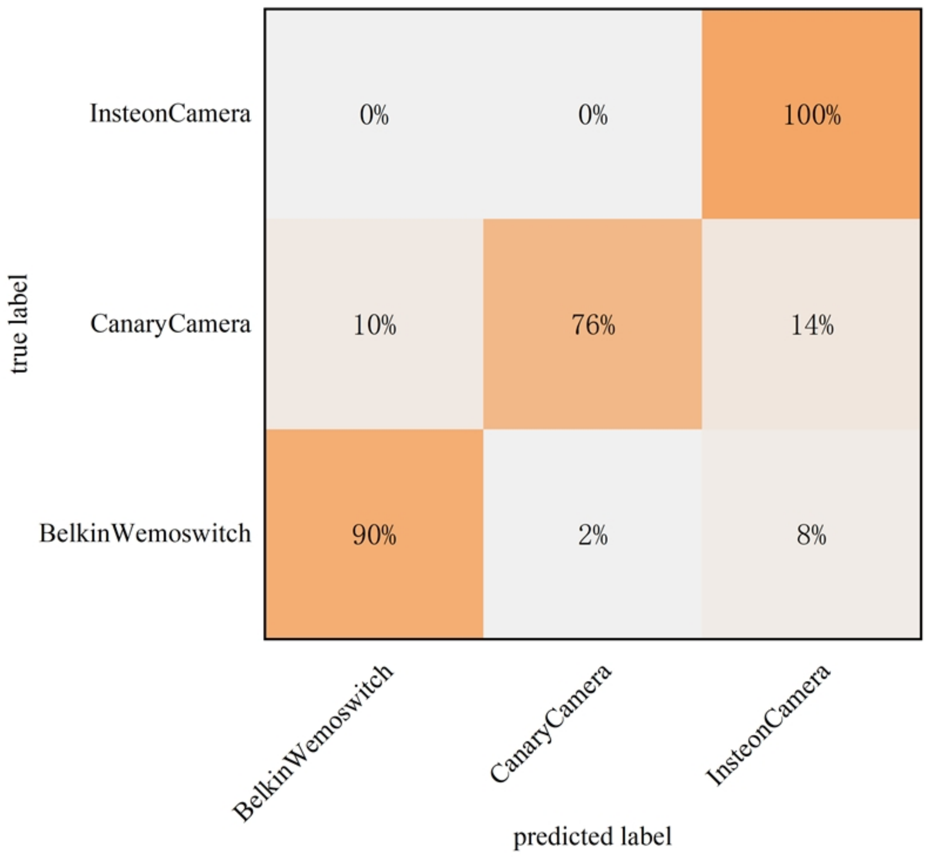


Figure 17. Confusion matrix for identification of three unknown IoT devices.

As can be seen in Figure 17, the WVE-UDI algorithm is able to achieve 100% accuracy for the Insteon Camera device, with no misidentification as other IoT devices, and the best recognition effect; for the Canary Camera device, there is a 10% misidentification rate to judge it as the Belkin Wemo Switch device, while there is a 14% misidentification rate to judge it as Insteon Camera device. On the one hand, this is because Canary Camera devices and Insteon Camera devices belong to the same camera category, and they have some similarities in some features, which makes the algorithm confused in the recognition; on the other hand, the WVE-UDI algorithm is a fusion of the clustering ability of three single clustering algorithms, but since all three algorithms have a poor recognition accuracy for Canary Camera devices, the recognition accuracy of all three algorithms is not high, resulting in a limited improvement of the WVE-UDI algorithm for Canary Camera devices after integration, which shows that, to a certain extent, the WVE-UDI algorithm is still limited by the influence of the correct recognition results in the single clustering algorithms; for the Belkin Wemo Switch device, there exists a misrecognition rate of 2% judging it as a Canary Camera device, and another 8% misidentification rate judges it as Insteon Camera device, but its correct identification rate reaches 90%, with relatively good identification results.

4.9. CSCL-WVE-UDI Method Validity Experiment

In order to validate the effectiveness of the CSCL-WVE-UDI method, experiments and analyses are conducted on the classic UNSW dataset for a scenario in which three unknown IoT devices coexist, and Precision, Recall, and F1 values are used to assess their recognition performance. The choice of three unknown IoT devices for testing in this experiment stems both from literature [7] that randomly selects three IoT devices from the UNSW dataset, and because three unknown IoT devices are sufficient to evaluate the performance of recognition method [7], which balances the experimental complexity with the demonstration of the recognition results. The results of this experiment are shown in Figure 18.

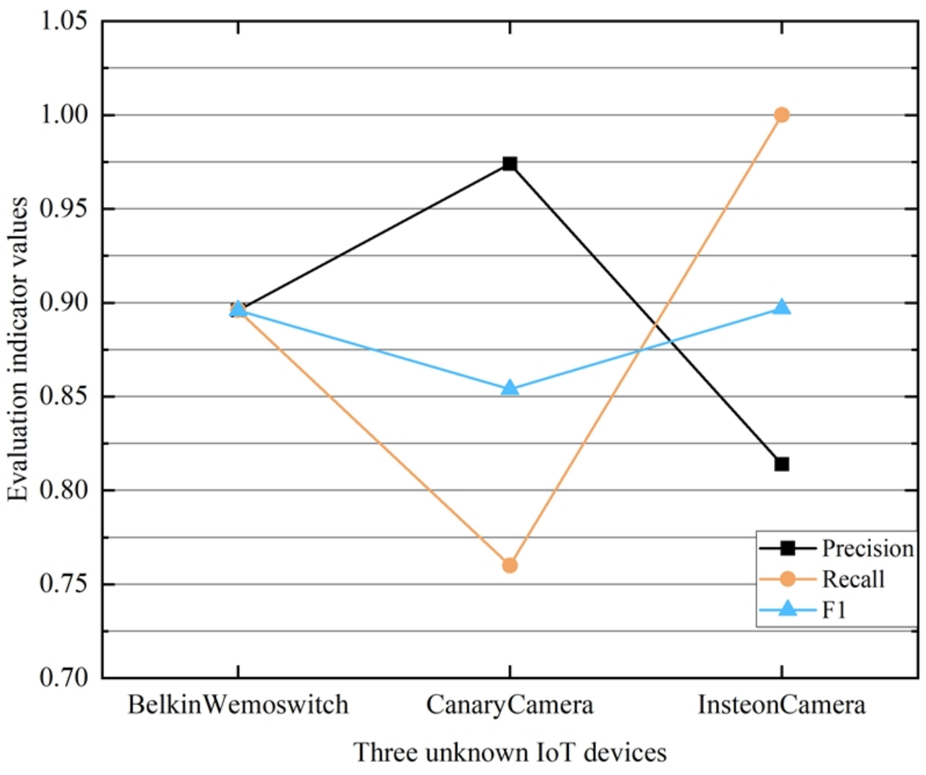


Figure 18. Experiments on the validity of the CSCL-WVE-UDI approach.

As can be seen from Figure 18, the CSCL-WVE-UDI method is more effective in recognizing the three unknown IoT devices, but there are also less-than-ideal recognition results, which are as follows: (1) for the Belkin Wemo Switch device, the Precision rate, Recall rate, and F1 value remain the same, and all of them reach 89.6% recognition, which indicates that the CSCL-WVE- UDI method is able to handle the recognition task of this unknown IoT device stably and effectively; (2) For the Canary Camera device, the CSCL-WVE-UDI method exhibits a high Precision rate and a low Recall rate, which is further shown by combining with the confusion matrix in Figure 17, which is due to the fact that only 2% of the Belkin Wemo Switch device is misidentified as the Canary Camera devices, making the false positive rate (FP) relatively low, but 24% of the Canary Camera devices are misidentified as the other two devices, resulting in an increase in the false negative rate (FN). Therefore, although the CSCL-WVE-UDI method shows a high Precision rate in recognizing Canary Camera devices, it fails to correctly identify all the Canary Camera devices, pulling down the Recall rate, but the F1 value still reaches 85.4%, which indicates that the CSCL-WVE-UDI method has a better overall performance and is capable of recognizing in all the Canary Camera devices to a certain extent the contradiction between Precision rate and Recall rate; (3) For Insteon Camera devices, the CSCL-WVE-UDI method has a higher Recall rate of 100%, but the Precision rate is 81.4%, which is relatively low, due to the fact that 8% of the Belkin Wemo Switch devices and 14% of the Canary Camera devices are misidentified as Insteon Camera devices, resulting in an increase in the false positive rate (FP) for this device. However, the CSCL-WVE-UDI method succeeded in correctly identifying all Insteon Camera devices, resulting in a false negative example rate (FN) of 0. Therefore, the Recall rate reached 100% and its F1 value reached 89.7%. In summary, the CSCL-WVE-UDI method shows good performance in recognizing three unknown IoT device coexistence scenarios, which validates the effectiveness of the CSCL-WVE-UDI method.

5. Conclusions

This study developed a method for identifying unknown IoT devices based on CSCL-Siamese networks and a WVE. First, to ensure that the model is trained with a rich set of contrastive training data, the PNSP algorithm was presented, which combines data visualization techniques with a permutation pairing strategy to generate all positive–negative sample pairs while eliminating redundant ones. The decision boundary exhibited bias due to an insufficient number of positive sample pairs. To address this, we improved the classical contrastive loss function and trained the sample data using a CSCL-Siamese network. This process enabled the creation of an embedded vector database for known IoT devices. We used this database to filter out embedding vectors of unknown IoT devices by performing pairwise comparisons. Finally, by integrating cost-sensitive weighting factors with a voting ensemble strategy, a UDI algorithm based on a WVE was proposed, which leverages the clustering capabilities of multiple unsupervised clustering algorithms to complete the identification of unknown IoT devices. By identifying three randomly selected unknown IoT devices, the effectiveness of the proposed method was validated. However, this study also has some limitations: First, the Siamese network can only process a single fingerprint feature due to only grayscale images being utilized to construct the device fingerprints. Combining image data with other fingerprint features to form a new, unified IoT device fingerprint can be considered in future studies. Second, the proposed CSCL-Siamese network requires considerable known IoT devices for training before unknown device identification can proceed. Thus, methods for identifying unknown devices that do not rely on data from known IoT devices will be our next studies.

Author Contributions: Conceptualization, J.Q., W.Z. and X.L.; methodology, W.Z.; software, X.L.; validation, X.L. and W.Z.; formal analysis, J.Q.; investigation, W.Z.; resources, Z.L.; data curation, X.L.; writing—original draft preparation, X.L.; writing—review and editing, W.Z.; visualization, W.Z.; supervision, J.Q.; project administration, X.L.; funding acquisition, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Smart Manufacturing New Model Application Project Ministry of Industry and Information Technology (No.ZH-XZ-18004), the Fundamental Research Funds for the Ministry of Education (No.JUSRP 211A41), the Fundamental Research Funds for the Central Universities (No.JUSRP42003), and the 111 Project (No.B2018).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data suporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sasaki, Y. A survey on IoT big data analytic systems: Current and future. *IEEE Internet Things J.* **2021**, *9*, 1024–1036. [CrossRef]
2. Luo, S.; He, R.; La, B. Research on Security Protection Technology for IoT Devices. *Netw. Secur. Technol. Appl.* **2023**, *12*, 24–26. [CrossRef]
3. Aksoy, A.; Gunes, M.H. Automated iot device identification using network traffic. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–7. [CrossRef]
4. Zhang, S.; Xiao, K.; Yu, J.; Liu, X.; Wang, W. Accurate IoT Device Identification based on A Few Network Traffic. In Proceedings of the 2023 IEEE/ACM 31st International Symposium on Quality of Service, Orlando, FL, USA, 19–21 June 2023; pp. 1–10. [CrossRef]
5. Trad, F.; Hussein, A.; Chehab, A. Using Siamese Neural Networks for Efficient and Accurate IoT Device Identification. In Proceedings of the 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 12–15 December 2022; pp. 1–7. [CrossRef]

6. Trad, F.; Hussein, A.; Chehab, A. Assessing the Effectiveness of Siamese Neural Networks to Mitigate Frequent Retraining in IoT Device Identification Models. In Proceedings of the 2023 International Conference on Platform Technology and Service (PlatCon), Busan, Republic of Korea, 16–18 August 2023; pp. 47–52. [CrossRef]
7. Yin, S.; Zhang, W.; Feng, Y.; Xiang, Y.; Liu, Y. Automatic IoT device identification: A deep learning based approach using graphic traffic characteristics. *Telecommun. Syst.* **2023**, *83*, 101–114. [CrossRef]
8. Bao, J.; Hamdaoui, B.; Wong, W.K. IoT Device Type Identification Using Hybrid Deep Learning Approach for Increased IoT Security. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 565–570. [CrossRef]
9. Ross, J. Tshark, 2022. [EB/OL]. Available online: https://tshark.dev/export/ (accessed on 11 April 2024).
10. Kotak, J.; Elovici, Y. IoT Device Identification Using Deep Learning. In *13th International Conference on Computational Intelligence in Security for Information Systems*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 76–86. [CrossRef]
11. Potter, G.; Guillaume, V.; Pierre. Scapy. 2023. [EB/OL]. Available online: https://scapy.net/ (accessed on 11 April 2024).
12. Damiani, E.; Vimercati, S.; Paraboschi, S.; Samarati, P. An Open Digest-based Technique for Spam Detection. In Proceedings of ISCA 17th International Conference on Parallel and Distributed Computing Systems, San Francisco, CA, USA, 15–17 September 2004; pp. 559–564.
13. Fuentealba, P.; Chamorro, E.; Santos, J.C. Chapter 5 Understanding and using the electron localization function. *Theor. Asp. Chem. React.* **2008**, *19*, 57–85. [CrossRef]
14. Yin, F.; Li, Y.; Wang, Y.; Dai, J. IoT ETEI: End-to-End IoT Device Identification Method. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Japan, 30 January–2 February 2021; pp. 1–8. [CrossRef]
15. Wang, F.; Liu, H. Understanding the behaviour of contrastive loss. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2495–2504.
16. Hu, J.; Wang, J. Prediction of liquefaction of gravelly soils based on a cost-sensitive Bayesian network combined with rough set weighting. *Gondwana Res.* **2024**, *131*, 57–68 . [CrossRef]
17. Majhi, M.; Pal, A.K.; Islam, S.K.H.; Khurram Khan, M. Secure content-based image retrieval using modified Euclidean distance for encrypted features. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4013. [CrossRef]
18. Bamhdi, A.M.; Abrar, I.; Masoodi, F. An ensemble based approach for effective intrusion detection using majority voting. *Telkomnika Telecommun. Comput. Electron. Control.* **2021**, *19*, 664–671. [CrossRef]
19. Tokuda, E.K.; Comin, C.H.; Costa, L.F. Revisiting agglomerative clustering. *Phys. A Stat. Mech. Appl.* **2022**, *585*, 126433. [CrossRef]
20. Yin, S.; Li, H.; Liu, D.; Karim, S. Active contour modal based on density-oriented BIRCH clustering method for medical image segmentation. *Multimed. Tools Appl.* **2020**, *79*, 31049–31068. [CrossRef]
21. Shieh, C.S.; Lin, W.W.; Nguyen, T.T.; Chen, C.H.; Horng, M.F.; Miu, D. Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Appl. Sci.* **2021**, *11*, 5213. [CrossRef]
22. ACRIS. IoT Devices Setup Captures, 2017. [EB/OL]. Available online: https://research.aalto.fi/en/datasets/iot-devices-captures (accessed on 11 April 2024).
23. Sivanathan, A.; Gharakheili, H.H.; Loi, F.; Radford, A.; Wijenayake, C.; Vishwanath, A.; Sivaraman, V. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1745–1759. [CrossRef]
24. Charyyev, B.; Gunes, M.H. IoT Traffic Flow Identification using Locality Sensitive Hashes. In Proceedings of the ICC 2020–2020 IEEE International Conference on Communications, Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
25. Thom, J.; Thom, N.; Sengupta, S.; Hand, E. Smart Recon: Network Traffic Fingerprinting for IoT Device Identification. In Proceedings of the 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 26–29 January 2022; pp. 72–79. [CrossRef]
26. Bruhadeshwar, B.; Maalvika, B.; Jordan, P.; Shirazi, H.; Ray, I.; Ray, I. *Behavioral Fingerprinting of IoT Devices*; Association for Computing Machinery—ACM: New York, NY, USA, 2018; pp. 41–50. [CrossRef]
27. Kostas, K.; Just, M.; Lones, M.A. IoTDevID: A Behavior-Based Device Identification Method for the IoT. *IEEE Internet Things J.* **2022**, *9*, 23741–23749.