Journal of Computer Science and Frontier Technologies
ISSN Print: 3104-4204
ISSN Online: 3104-4212
CD ISSN: 3104-4220
https://doi.org/10.63313/JCSFT.9044

FreeMatch-PG: A Lightweight Semi-supervised Learning Framework for IoT Device Identification

ShuYi Song
School of Computer Science and Technology, Qingdao University, Qingdao 266071, China
Email: songshuyi0219@163.com

How to cite this paper: Song, S. Y. (2026). FreeMatch-PG: A lightweight semi-supervised learning framework for IoT device identification. Journal of Computer Science and Frontier Technologies, 2(3), 1-18. ISSN Print: 3104-4204; ISSN Online: 3104-4212.
https://doi.org/10.63313/JCSFT.9044
Published: 2026-02-12

Abstract

IoT device identification is a critical component of cybersecurity, yet its practical deployment faces the dual challenges of scarce labeled data and imbalanced device class distribution. To address these issues, this paper proposes a lightweight identification framework based on semi-supervised learning. The core innovations of the framework include: 1) A prior-guided adaptive threshold mechanism (FreeMatch-PG), which sets differentiated learning thresholds for various classes by simulating initial cognitive states, alleviating class imbalance from the source and significantly improving pseudo-label quality; 2) Domain-customized data augmentation and a weighted focal loss function, which jointly enhance model robustness in noisy environments; 3) A lightweight architecture based on an improved ResNet-18, substantially reducing model complexity. Experiments on two public datasets demonstrate that using only 8% of labeled data, the proposed method achieves recognition accuracies of 98.25% and 97.72% on the UNSW and CICIOT datasets, respectively. It significantly outperforms multiple baseline models and achieves an excellent balance between accuracy, efficiency, and deployment feasibility, offering a practical solution for resource-constrained real-world IoT environments.

Keywords

IoT Security; Device Identification; Semi-Supervised Learning; Class Imbalance; Adaptive Threshold; Lightweight Model

1. Introduction

The widespread adoption of the Internet of Things (IoT) has led to the extensive deployment of IoT devices across diverse domains of daily life, such as smart homes, intelligent transportation, and healthcare. However, their frequent lack of robust security features poses significant challenges to cybersecurity [2]. It is projected that the global number of IoT devices will reach 80 billion by 2026, with a total market value of US$1.1 trillion [1]. While this proliferation offers substantial convenience, it also introduces numerous security issues [2]. Typically designed for

single, well-defined functions, IoT devices are often manufactured with cost as a primary driver, leading manufacturers and suppliers to overlook security considerations. Consequently, a vast number of devices are deployed without rigorous security testing or support for basic security protocols, resulting in critical vulnerabilities [2]. These vulnerabilities create opportunities for attackers to impersonate legitimate devices for malicious operations or to intercept sensitive information during data transmission, causing privacy breaches. Furthermore, network administrators often lack visibility into the types, quantities, and operational status of IoT devices within their networks, making effective management exceedingly difficult. Therefore, developing a method for accurate IoT device identification is imperative [3]. A successful identification methodology would not only detect malicious device impersonation, prevent the escalation of security threats, and mitigate data leakage risks, but also enhance network administrators' operational efficiency by enabling effective management of the IoT device landscape.

This paper addresses the critical challenge of IoT device identification under real-world constraints, specifically scarce labeled samples, significant device heterogeneity, and resource limitations. To this end, we propose a semi-supervised recognition framework based on traffic-based grayscale image fingerprints [5]. This approach is characterized by its lightweight design [6], high scalability, and low computational overhead, enabling high-precision identification of large-scale heterogeneous devices with minimal labeled data. Our methodology is grounded in compact yet effective device fingerprint representations derived from grayscale images generated from network traffic [5]. We further enhance these representations by incorporating temporal modeling to capture the dynamic characteristics of device communication behavior. Building upon this foundation, we design a novel training strategy that integrates pseudo-label quality control with multi-task cooperative optimization [4]. This strategy significantly improves the model's discriminative capability and generalization performance under semi-supervised conditions. Experimental results on multiple public IoT traffic datasets [7] demonstrate the effectiveness and generalizability of our proposed method. Compared to existing state-of-the-art approaches, our framework exhibits superior performance in computational efficiency, recognition accuracy, and deployment feasibility. This work thus presents a practical and promising solution for scalable IoT device identification in real-world environments.

The main contributions of this paper are summarized as follows:

(1) We propose an end-to-end lightweight framework for IoT device identification, which can autonomously learn discriminative feature representations directly from raw network traffic. This approach eliminates the need for manual feature engineering or complex pre-processing, significantly simplifying the system pipeline while enhancing the model's generalizability and cross-scenario adaptability.

(2) We design an efficient pseudo-label-driven semi-supervised learning strategy that effectively exploits the latent structural information within unlabeled data. This strategy is characterized by a simple architecture, low computational cost, and stable training dynamics. It improves model performance while ensuring commendable deployability and robustness.

(3) We conduct comprehensive experimental evaluations on multiple public IoT traffic datasets. The results demonstrate that our proposed method outperforms existing mainstream approaches in terms of identification accuracy, training efficiency, and resource consumption, highlighting its strong practical value and promising potential for broader application.

2. Related Work

Building upon this foundation, semi-supervised learning methods offer a compelling solution by simultaneously addressing the reliance on large volumes of labeled data inherent in supervised learning and the limited recognition accuracy often associated with unsupervised learning [10]. This makes them a highly adaptive approach for identification tasks. For instance, Nukavarapu et al. proposed iKnight [11], a lightweight multi-stage classification framework based on a semi-supervised Generative Adversarial Network. This framework can identify known IoT devices with 96% accuracy using only 3% labeled data and infer unknown device types with an average accuracy of 90%, significantly reducing the model's dependence on annotations and enhancing both generalization capability and practicality for device identification in edge environments. Similarly, Jin et al. [12] employed a hierarchical semi-supervised Generative Adversarial Network for IoT device identification. Their method effectively identifies device type, function, and manufacturer under limited labeled data by leveraging semi-supervised learning within a hierarchical classification structure, while also reducing reliance on manual feature engineering. While existing semi-supervised methods demonstrate notable advantages—maintaining high recognition accuracy with scarce labeled data and adapting to rapid changes in IoT environments—they are not without significant limitations. Their primary drawbacks include high computational complexity and cost, which hinder their practical deployment in resource-constrained scenarios, and an inherent inefficiency in accommodating newly introduced devices.

3. Method

3.1. Overview of proposed method

We propose a lightweight and scalable IoT device identification scheme based on semi-supervised learning. This scheme primarily consists of the following two modules:

(1) Data Preprocessing Module: This module is responsible for the standardized processing of raw network traffic data. Specifically, it first segments packets into

flows based on the five-tuple. Subsequently, it anonymizes privacy-sensitive fields such as MAC and IP addresses to enhance the model's generalization capability. Finally, the traffic data is converted into fixed-size grayscale image representations, providing structured input for the subsequent deep learning model. The entire process supports encrypted traffic handling, ensuring the scheme's broad applicability.

(2) Semi-supervised Learning Module: Serving as the core identification engine of the scheme, we propose a prior-guided adaptive threshold learning mechanism. By introducing a category-aware confidence threshold initialization strategy and combining it with a weighted focal loss function and an adaptive fairness objective, this module effectively mitigates model bias caused by extreme class imbalance. It can fully leverage a large number of unlabeled samples to achieve high-precision device identification while relying on only a minimal amount of labeled data, significantly reducing dependence on manual annotation.

3.2. Methology

3.2.1. Data Preprocess

To achieve effective identification of IoT devices, it is necessary to transform the raw network traffic data (in PCAP format) into a standardized input suitable for deep learning models. The data preprocessing pipeline of this scheme does not rely on the plaintext content of the traffic; therefore, it can process data in an identical manner even when the original traffic is encrypted, effectively ensuring the method's generality and privacy security. The entire preprocessing procedure primarily consists of three key steps: session flow segmentation, privacy-sensitive field processing, and traffic grayscale image generation.

(1) Session Flow Segmentation

The granularity of network traffic segmentation directly impacts the representational capability of the constructed features. Common segmentation approaches include packet-level, unidirectional flow-level, and bidirectional session-level [8]. In this study, we select the unidirectional flow as the basic processing unit. Specifically, packets sharing the same five-tuple—source IP, destination IP, source port, destination port, and network protocol—are grouped into a single flow.

This granularity is chosen based on the following considerations: compared to individual packets, a flow contains multiple packets generated during a single communication event, thereby offering more comprehensive behavioral features essential for subsequent device fingerprint extraction and identification. Compared to bidirectional sessions, unidirectional flows preserve critical communication characteristics while maintaining higher processing efficiency and simplicity.

(2) Privacy-Sensitive Field Processing

To encourage the model to focus on learning behavioral patterns in the traffic rather

than relying on static device identifiers, we process certain original fields, particularly MAC and IP addresses. Since these fields can uniquely identify devices and may lead to model overfitting, we uniformly replace them with a padding value (0x00). This practice enhances the model's generalization ability, making it more suitable for real-world IoT device identification tasks.

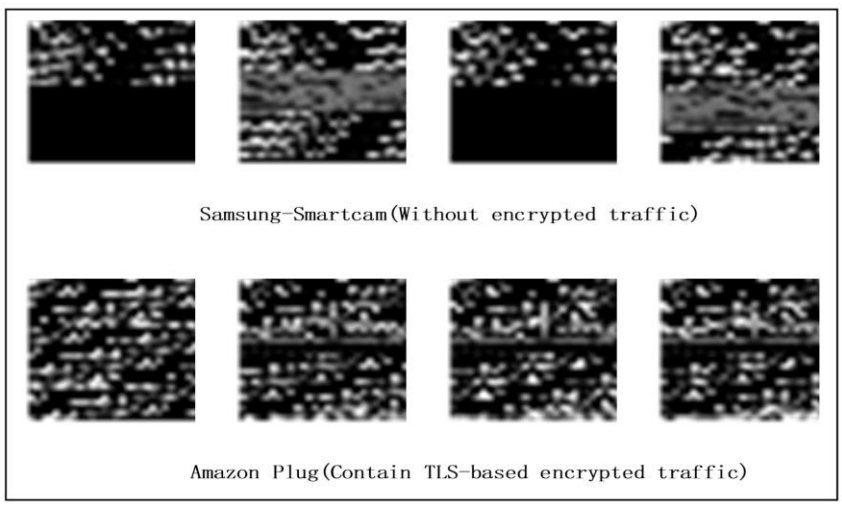
(3) Traffic Grayscale Image Generation

Deep learning models generally require inputs to be structured data of fixed dimensions. However, the number and length of packets in a network session often vary. To standardize the input format, we further convert the preprocessed flow data into fixed-size grayscale images. This process involves the following steps:

a) Dimension Unification: Each flow is transformed into a byte sequence of length $n \times n$. If the total length of a flow exceeds this value, the first $n \times n$ bytes are retained; if shorter, the sequence is padded with 0x00 at the end to ensure a fixed-length byte sequence per flow.

b) Image Generation: Each byte value ranges from 0 to 255, which corresponds directly to pixel values in a grayscale image. Therefore, reshaping the byte sequence into a two-dimensional array of size $n \times n$ yields a single grayscale image. If $n \times n$ is not a perfect square, the two closest integers are selected to form an image as close to a square shape as possible.

Figure 1. Visualization of Traffic Gray Image.



In this study, the aforementioned preprocessing was conducted based on the UNSW dataset and the CICIOT dataset, successfully transforming raw PCAP traffic into standardized grayscale images. The visualization results of some samples are presented in Figure 1 to facilitate an intuitive understanding of the image structure and features.

3.2.2. Model architecture

To better adapt to recognition tasks with low-resolution input images, we have

introduced targeted structural modifications to the standard ResNet-18 architecture. The core of the modifications lies in aligning the network's complexity with the characteristics of the input data, thereby preserving the advantages of residual learning while improving parameter efficiency. The refined network structure is outlined in Table 1, and a detailed description of the design is provided below:

Table 1. Detailed design of network structure.

Operator	Output Shape	Kernel	Stride	Blocks
Conv(ReLU)	(32, 28, 28)	3×3	1	1
BasicBlock $\times 2$	(32, 28, 28)	3×3	1	2
BasicBlock $\times 2$	(64, 14, 14)	3×3	2	2
BasicBlock $\times 2$	(128, 7, 7)	3×3	2	2
BasicBlock $\times 2$	(256, 4, 4)	3×3	2	2
Global Avg Pool	(256)	—	—	—

(1) Initial Feature Extraction Layer:

The network takes a grayscale image of size 1×28×28 as input. It first passes through a convolutional layer with a 3×3 kernel, a stride of 1, and 32 output channels. This is followed by a Batch Normalization layer and a ReLU activation function, which collectively perform the initial feature mapping.

(2) Residual Backbone Layers:

The main feature extraction backbone follows the design philosophy of ResNet and consists of four residual stages. Each stage contains multiple stacked basic residual blocks. The scheme adopts the BasicBlock structure as defined in [14], whose core is a residual unit with a skip connection. Specifically, each BasicBlock comprises two 3×3 convolutional layers, each followed sequentially by a Batch Normalization layer and a ReLU activation function. Through skip connections that add the block's input to its output, this design effectively facilitates gradient flow in deep networks and mitigates the degradation problem during training. Except for the first stage, the first BasicBlock in each stage performs downsampling by setting the stride of its first convolutional layer to 2. This reduces the spatial resolution of the feature maps while doubling the number of channels, thereby balancing computational efficiency and feature representation capacity.

(3) Classification Layer:

The feature maps output by the backbone (256×4×4) are first compressed into a 256-dimensional feature vector via a global average pooling layer. Finally, a fully connected layer is applied to produce the classification results corresponding to the target categories.

The core advantage of this design lies in its significantly lightweight nature. By appropriately reducing the network depth and width, the parameter count is decreased by approximately 70% compared to the standard ResNet-18 model. This achieves a substantial reduction in computational cost while maintaining expressive

power. Furthermore, the network retains essential skip connection mechanisms to promote gradient propagation and alleviate degradation in deep networks. This lightweight backbone provides a unified and efficient feature representation foundation for the subsequent modules.

3.2.3. Semi supervised learning design

The task of IoT device identification has long faced the dual challenges of scarce labeled data and imbalanced class distribution in real-world scenarios. To address the shortage of annotations, semi-supervised learning (SSL) naturally emerges as a solution, with its core premise being the utilization of large amounts of unlabeled data to enhance model performance. The concept of dynamic thresholding has proven to be an effective SSL paradigm, achieving notable success by employing adaptive confidence thresholds to filter pseudo-labels and leveraging a "weak-strong" data augmentation strategy for consistency regularization. However, the complexity inherent in IoT device identification introduces new challenges to this general paradigm, revealing limitations in its original design. Therefore, a thorough domain-specific adaptation is crucial.

This paper proposes a semi-supervised learning framework named FreeMatch-PG (Prior-Guided FreeMatch). This framework inherits the core concept of dynamic thresholding and achieves precise screening and efficient utilization of high-quality samples from unlabeled data by dynamically adjusting both global and local confidence thresholds.

The core innovation of FreeMatch-PG lies in its prior-guided threshold initialization strategy. While the original FreeMatch method initializes per-class thresholds uniformly without accounting for inter-class imbalance—causing rare minority classes to face excessively high learning barriers from the outset—our framework mitigates this limitation. It aligns threshold initialization with the model's initial cognitive state through two phases: supervised warm-up and confidence estimation. This process sets differentiated initial thresholds for different classes, fundamentally alleviating the "Matthew Effect" in imbalanced learning and providing more learning opportunities for minority classes. Furthermore, we propose a domain-customized data augmentation strategy, moving away from methods designed for natural images. Instead, we design augmentations that simulate network perturbations (e.g., random masking/erasure), enhancing model robustness while preserving the semantic structure of traffic data. Lastly, we introduce a weighted focal loss to tackle class imbalance at the level of the loss function itself.

(1) Domain-Customized Data Augmentation Strategy

Data augmentation is pivotal for obtaining consistency regularization signals in semi-supervised learning. To avoid corrupting the semantic structure of network traffic data, we depart from augmentations designed for natural images, such as color jittering and significant rotations. Instead, we design a domain-customized

augmentation strategy. For each input traffic grayscale image, we define two augmentation pipelines:

a) Weak Augmentation: The original traffic grayscale image is used directly, maximally preserving the semantic information of the raw data. This branch is employed to generate reliable pseudo-labels.

b) Strong Augmentation: This pipeline simulates natural perturbations encountered during network transmission. Specifically, we apply random masking (e.g., random erasing of a region within the image) to mimic scenarios involving packet loss or partial traffic obfuscation. Strong augmentation aims to create more challenging noisy samples, thereby encouraging the model to learn more robust feature representations and avoid overfitting to specific details in the training data.

During training, labeled data undergoes only weak augmentation. For unlabeled data, both a weak-augmented and a strong-augmented version are generated for computing the consistency regularization loss.

(2) Definition and Data Composition

Let the labeled dataset be defined as $\mathcal{D}_L = \{(x_b, y_b): b \in [N_L]\}$and the unlabeled dataset as $\mathcal{D}_U = \{u_b: b \in [N_U]\}$, where N_L and N_U denote the total number of samples in each set, and it generally holds that $N_U \gg N_L$. This data composition highlights the particular value of semi-supervised learning in IoT device identification scenarios, as large-scale unlabeled traffic data can be easily collected, while high-quality device annotations require substantial manual effort and cost.

During each training iteration, a batch B_L of size B is sampled from $\mathcal{D}_L$, and a batch B_U of size μB is sampled from $\mathcal{D}_U$, where μ is a hyperparameter controlling the relative amount of unlabeled data. This sampling strategy ensures that the model simultaneously leverages the strong supervision from labeled data and the distributional information present in the unlabeled data.

(3) Supervised Loss: Weighted Focal Loss

To address the class imbalance inherent in IoT data, our framework introduces a fundamental improvement to the supervised loss $\mathcal{L}_s$. It replaces the standard cross-entropy loss with a weighted focal loss as the supervised signal [21]:

$$\mathrm{L}_s = -\frac{1}{B}\sum_{(x,y)\in \mathrm{B}_L} w_y (1 - p_m(y \mid x))^{\gamma} \log\left(p_m(y \mid x)\right) \quad (1)$$

where $p_m(y|x)$ represents the probability predicted by the model that sample x belongs to its true class y; w_y is the weight assigned to class y, defined as $w_y = 1/\sqrt{\mathrm{freq}(y)}$, where $\mathrm{freq}(y)$ denotes the frequency of class y in the training set. This weighting scheme enhances the model's focus on minority classes at the category level. The term $(1 - p_t)^{\gamma}$ is the focusing factor, with $\gamma \geq 0$ as a modulating parameter. This factor reduces the loss contribution from easy-to-classify samples at the instance level, thereby directing the model's attention toward hard samples. Through this dual adjustment mechanism, the loss function ensures that the model learns features from all classes in a balanced manner during training, effectively

mitigating the classifier's bias toward majority classes.

(4) Unsupervised Loss

For unlabeled data, the framework adopts a pseudo-label learning approach based on confidence thresholding, in conjunction with a weak-strong data augmentation strategy. The unsupervised loss is defined as follows:

$$\mathrm{L}_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathrm{I}(max(\mathbf{q}_b) > \tau)\cdot \mathrm{H}\left(\hat{\mathbf{q}}_b, \mathbf{Q}_b\right) \quad (2)$$

where $\mathbf{q}_b = p_m(y|\omega(u_b))$ denotes the predicted probability for the weakly augmented version, and $\mathbf{Q}_b = p_m(y|\Omega(u_b))$denotes the predicted probability for the strongly augmented version. Here, $\hat{\mathbf{q}}_b$ is the one-hot pseudo-label derived from $\mathbf{q}_b$, $\mathcal{H}(\cdot,\cdot)$ represents the cross-entropy loss, and $\mathbb{I}(\cdot > \tau)$ is an indicator function with threshold τ. The weak and strong augmentations are denoted by $\omega(\cdot)$ and $\Omega(\cdot)$, respectively. This augmentation strategy ensures the reliability of the pseudo-labels while providing rich regularization signals through the strong augmentations. The loss function guarantees that only high-confidence samples contribute to model training, thereby enhancing the stability and reliability of the learning process.

(5) Adaptive Threshold Mechanism

In semi-supervised learning, the selection of the threshold is crucial for the quality of pseudo-labels and directly impacts the model's learning effectiveness. Traditional fixed-threshold methods often perform poorly when dealing with imbalanced data, as significant differences exist in the learning difficulty and confidence distribution across different classes. FreeMatch-PG employs a dual-level threshold mechanism comprising a global threshold and local (per-class) thresholds, which work in concert to enable precise filtering of unlabeled data.

The prior-guided initialization strategy is a key innovation of FreeMatch-PG. Specifically, we first train the model on the labeled data for one full epoch to obtain an initial model p_(θ_0) with preliminary discriminative ability. This warm-up stage allows the model to learn the basic feature representations of each class. Subsequently, we calculate the average confidence for each class on the unlabeled data:

$$\mu_c = \frac{1}{|\mathrm{S}_c|}\sum_{u_b \in \mathrm{S}_c} maxp_{\theta_0}(y \mid u_b) \quad (3)$$

where $\mathcal{S}_c = \{u_b \in \mathcal{D}_U \mid arg\ maxp_{\theta_0}(y|u_b) = c\}$. This empirically guided initialization strategy ensures that the threshold setting aligns with the true distributional characteristics of the data.

The core idea of the global threshold design is to ensure that incorrect pseudo-labels are effectively filtered out. The global threshold τ_t is defined as the model's average confidence on the unlabeled data, where t denotes the t-th iteration. Taking computational efficiency into consideration, we employ an Exponential Moving

Average (EMA) to estimate the global confidence:

$$\tau_t = \begin{cases} \frac{1}{C}\sum_{c=1}^{C}\mu_c, & if \quad t=0, \\ \lambda\tau_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} max(\mathbf{q}_b), & otherwise, \end{cases} \tag{4}$$

where $\lambda \in (0,1)$ is the momentum decay coefficient for the EMA. This approach is not only computationally efficient but also helps smooth out fluctuations during training, ensuring stable growth of the threshold and preventing sharp oscillations caused by noise in individual mini-batches.

The local threshold mechanism is designed to modulate the global threshold in a class-specific manner. Since different classes vary in their feature space distributions and learning difficulties, personalized thresholds are required for each class. We estimate the class-specific learning states by calculating the model's expected prediction for each class:

$$\tilde{p}_t(c) = \begin{cases} \mu_c, & if \quad t=0, \\ \lambda\tilde{p}_{t-1}(c) + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbf{q}_b(c), & otherwise, \end{cases} \tag{5}$$

where $\tilde{p}_t = [\tilde{p}_t(1), \tilde{p}_t(2), \ldots, \tilde{p}_t(C)]$ is the list containing all $\tilde{p}_t(C)$. The final adaptive threshold is obtained by integrating the global and local thresholds:

$$\tau_t(c) = \mathrm{MaxNorm}\left(\tilde{p}_t(c)\right)\cdot\tau_t = \frac{\tilde{p}_t(c)}{max\left\{\tilde{p}_t(c') : c' \in [C]\right\}}\cdot\tau_t \tag{6}$$

where MaxNorm denotes max normalization. This design ensures that the threshold accounts for both the global confidence level and the specificity of individual classes: it sets a higher threshold for majority classes to guarantee pseudo label quality while assigning a lower threshold to minority classes to provide more learning opportunities.

(6) Total Loss Function and Optimization Strategy

The total loss function of the proposed scheme integrates three components: the supervised loss, the unsupervised loss, and the fairness loss:

$$\mathrm{L} = \mathrm{L}_s + \lambda_u \mathrm{L}_u + \lambda_f \mathrm{L}_f \tag{7}$$

Here, $\mathcal{L}_f$ represents the Self-Adaptive Fairness (SAF) objective [13][20], which effectively enhances the fairness of model predictions by dynamically aligning the predicted distribution with the pseudo-label distribution. λ_u and λ_f are balancing hyperparameters used to modulate the relative importance of the different loss components.

This multi-objective optimization framework ensures that the model maintains high accuracy while also exhibiting strong fairness and generalization capabilities. Through a carefully designed loss-weight scheduling strategy, the model relies more

heavily on the reliable supervised signal during the initial training phase. As training progresses, the utilization of the unsupervised signal is gradually increased, ultimately leading to stable and enhanced performance.

4. Experiments and Evaluation

In this section, we evaluate our proposed method on two public datasets and present comprehensive experimental results across multiple dimensions.

4.1. Experimental Setup

(1) Hardware Environment

The experiments were conducted on a computer equipped with a 13th Gen Intel® Core™ i7-13650HX processor, 24 GB of RAM, an NVIDIA GeForce RTX 4060 graphics card, and running the Windows 11 operating system.

(2) Software Environment

The primary programming language used was Python 3.11, with PyCharm 2024.2 as the integrated development environment. Wireshark was employed for traffic analysis. The model was implemented using the PyTorch framework.

(3) Datasets

The first dataset used is the UNSW dataset [9][14], which contains traffic data from 23 IoT devices, each monitored over 26 weeks, with a total size of 11.7 GB. We excluded non-IoT devices (e.g., smartphones, laptops) as well as IoT devices that generated negligible traffic, retaining 21 typical low-cost, resource-constrained IoT devices. Since this dataset was collected under normal operating conditions, experiments using it better reflect real-world environments.

The second dataset is CICIoT2022, comprising traffic data from 40 IoT devices, each observed for 30 days, with a total size of 46.1 GB. Following a similar filtering process as for the UNSW dataset, we retained 39 IoT devices. Both datasets contain a small proportion of TLS-based encrypted traffic [15].

(4) Deployment Environment

Our system was deployed on a laptop computer, which was utilized for IoT traffic capture, model training, and device identification.

4.2. Experimental Procedure

The experimental procedure was divided into three stages. In the first stage, raw network traffic was converted into standardized grayscale images. Specific tasks included: first, performing flow segmentation based on the five-tuple to capture complete device communication behaviors as the fundamental unit; second, anonymizing MAC and IP addresses to eliminate interference from device-specific identifiers and enhance the model's generalization capability; and finally, converting the network traffic into 28×28 grayscale images to provide structured input for subsequent models.

In the second stage, we designed a prior-guided adaptive threshold initialization

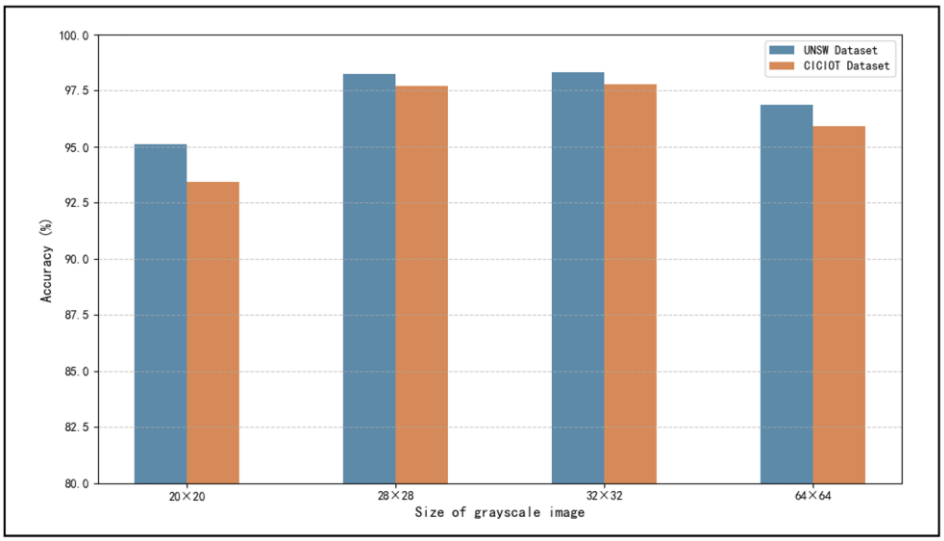
strategy. This strategy utilized a small amount of labeled data to warm up the model and set differentiated initial thresholds for different classes, aiming to mitigate the "Matthew Effect" caused by class imbalance from the outset and provide more learning opportunities for minority classes. Additionally, a weighted focal loss function was adopted to jointly address the issues of class imbalance and hard sample learning at the loss function level. Simultaneously, a domain-customized data augmentation strategy was implemented to simulate network perturbations, thereby enhancing the model's robustness without corrupting the semantic structure of the traffic. The common objective of these designs was to achieve high-precision device identification using only 8% of the labeled data.

In the third stage, a core set management mechanism based on the Herding Selection algorithm was introduced. This mechanism preserves the most representative samples for each known class, establishing a knowledge base for subsequent incremental learning and new device verification. We constructed a dynamic awareness and verification module, which identifies potential new devices through confidence threshold filtering and cluster analysis, enabling precise discovery of new device types. Finally, a distillation-based class-incremental learning strategy was employed. When introducing new device classes, the model is updated through knowledge distillation and core set replay. The fundamental purpose is to effectively alleviate catastrophic forgetting of previously learned old classes while rapidly assimilating new knowledge, ultimately achieving continuous learning of the model.

4.3. Selection of Grayscale Image Size

The choice of grayscale image size for network traffic is a critical factor affecting both feature representation quality and model performance. To determine the optimal image size, we conducted a comparative analysis of four different dimensions—20×20, 28×28, 32×32, and 64×64—across multiple datasets. The results are illustrated in Figure 2.

Figure 2. Accuracy of Different Grayscale Image Sizes in UNSW Dataset and CICIOT Dataset.



The experimental results indicate that when the image size was increased from 20×20 to 28×28, the recognition accuracy on the UNSW and CICIOT datasets improved from 95.12% and 93.45% to 98.25% and 97.72%, respectively. This demonstrates that the 28×28 size more adequately preserves the temporal characteristics and statistical properties of the network traffic, providing the model with more complete feature information for learning IoT device behavioral patterns. Notably, when the image size was increased to 32×32, the model performance showed only a slight, non-significant improvement, suggesting that further enlargement yields diminishing returns. In contrast, enlarging to 64×64 led to a clear decline in performance, likely because the excessive size introduced noise and resulted in overfitting.

Considering both performance and computational efficiency, we selected 28×28 as the standardized size for network traffic grayscale images. This size not only ensures excellent recognition performance for the model but also maintains a reasonable level of computational complexity, providing a solid foundation for subsequent model training and deployment.

4.4. Model Selection

In terms of model selection, we compared the performance of our improved ResNet-18 with four other advanced models on both datasets, namely AlexNet, LeNet, VggNet, and MobileNet. The evaluation metrics included accuracy, precision, recall, F1-score, macro-F1 score, and the number of model parameters.

AlexNet [19]: One of the pioneers of deep convolutional neural networks, it achieved breakthrough performance in image recognition tasks by employing techniques such as the ReLU activation function and Dropout.

LeNet [18]: A classic lightweight convolutional neural network, first successfully

applied to handwritten digit recognition, known for its simple architecture and low computational overhead.

VggNet [16]: Constructed a deeper network architecture by stacking multiple 3×3 convolutional layers, demonstrating powerful feature representation capabilities in various vision tasks.

MobileNet [17]: A lightweight model focused on mobile and embedded devices, which significantly reduces parameter count and computational complexity through the use of depthwise separable convolutions.

In the comparative experiments, we modified all model architectures to enable feature extraction from 28×28×1 grayscale images.

To evaluate model performance, we employed five metrics: accuracy, precision, recall, F1-score, and macro-F1 score.

Dataset	CICIOT2022					UNSW2018					Parameter Count
	Acc	Pre	Rc	F1	Macro-F1	Acc	Pre	Rc	F1	Macro-F1	
AlexNet	96.13%	94.57%	93.81%	94.19%	90.87%	96.72%	95.06%	94.21%	94.63%	91.53%	17.93M
LeNet	95.33%	92.72%	91.69%	92.20%	90.27%	96.88%	94.31%	93.77%	94.04%	91.71%	**0.04M**
VggNet	94.85%	93.22%	92.31%	92.76%	89.37%	95.42%	93.88%	91.79%	92.82%	90.91%	113.49M
MobileNet	96.41%	95.57%	94.05%	94.80%	91.77%	97.12%	96.43%	95.05%	95.74%	92.49%	2.48M
Our Model	**97.72%**	**97.01%**	**95.48%**	**96.24%**	**93.87%**	**98.25%**	**97.58%**	**96.27%**	**96.92%**	**94.38%**	3.47M

Table 2. Performance Comparison under Different Models.

Based on a comprehensive analysis of multiple evaluation metrics, our proposed improved ResNet-18 architecture demonstrates outstanding performance in IoT device identification tasks. As shown in Table 2, our model achieved accuracies of 97.72% and 98.25% on the CICIOT2022 and UNSW2018 datasets, respectively, comprehensively surpassing all other compared models.

In the CICIOT2022 dataset, our proposed improved ResNet-18 architecture achieved the optimal performance among all models, with an accuracy of 97.72%. This significantly exceeds models with larger parameter counts, such as VggNet (94.85%) and AlexNet (96.13%). This fully demonstrates that our model possesses excellent feature discrimination capabilities while maintaining a lightweight design. Its efficient parameter design not only reduces computational costs but also lays a foundation for rapid deployment in resource-constrained IoT environments.

Further analysis on the UNSW2018 dataset confirms the model's robust performance, with its accuracy further increasing to 98.25%. More importantly, the model leads comprehensively across other key metrics, including precision, recall, and F1-score. This superior performance highlights the model's strong generalization capability and high reliability, enabling effective identification of various IoT devices and robust handling of class imbalance problems prevalent in real-world scenarios.

In terms of model efficiency, our architecture exhibits excellent practicality. Its parameter count is only about 3% that of VggNet, yet it delivers superior recognition performance. Compared to MobileNet, which is also designed for lightweight applications, our model achieves a 1.31 percentage point improvement

in accuracy at a comparable parameter cost. This represents a well-balanced trade-off between model performance and complexity.

In conclusion, our model strikes an effective balance between high recognition accuracy and model light-weight. The experimental results indicate that we have not merely sacrificed efficiency for performance, nor compromised performance for light-weight. Instead, through innovative architectural optimization, we have successfully integrated high performance with high efficiency. Therefore, our model is a preferred solution for achieving high-precision IoT device identification in resource-constrained environments.

4.5. Ablation Study

4.5.1. Loss Functions

To evaluate the impact of different loss functions on model performance with imbalanced class data, we systematically conducted an ablation study on both the UNSW and CICIOT datasets. The experiment compared several loss functions, including Cross-Entropy Loss (CE Loss), Weighted Cross-Entropy Loss (Weighted CE Loss), Focal Loss, and their combinations. Evaluation metrics comprised accuracy and macro-F1 score, with the results presented in Table 3. All experiments employed the same network architecture and training strategy, and used only 8% of the labeled data to ensure fairness in comparison.

Dataset	Loss Function	Acc (%)	Macro-F1 (%)
UNSW	CE Loss	91.57	85.25
	Weighted CE Loss	95.83	90.15
	Focal Loss	96.47	91.87
	CE Loss + Focal Loss	96.92	92.58
	Weighted CE Loss + Focal Loss	97.62	93.78
	Weighted Focal Loss	98.25	94.38
CICIOT	CE Loss	90.23	83.59
	Weighted CE Loss	94.92	88.95
	Focal Loss	95.74	90.48
	CE Loss + Focal Loss	96.26	91.39
	Weighted CE Loss + Focal Loss	97.15	92.80
	Weighted Focal Loss	97.72	93.87

Table 3. Equipment Identification Capability of Different Loss Functions.

The experimental results demonstrate that the choice of loss function has a significant impact on model performance. Specifically, on the UNSW dataset, the baseline model using standard CE Loss achieved an accuracy of 91.57% and a Macro-F1 score of 85.25%. After introducing the class-weighted Weighted CE Loss, these two metrics improved to 95.83% and 90.15%, respectively, indicating that explicitly adjusting class weights can effectively mitigate model bias caused by class imbalance. Employing Focal Loss further increased accuracy and Macro-F1 to 96.47% and 91.87%, proving that its strategy of focusing on hard samples enhances the model's discriminative ability.

Ultimately, Weighted Focal Loss, which combines the weighting strategy with the

Focal Loss mechanism, achieved the best performance. On the UNSW dataset, it reached an accuracy of 98.25% and a Macro-F1 score of 94.38%; on the CICIOT dataset, the corresponding metrics were 97.72% and 93.87%, both significantly outperforming other loss function configurations. This result fully demonstrates that the design of this loss function allows for synergistic effects, balancing attention across classes while effectively guiding the model to learn difficult-to-classify samples, thereby comprehensively improving the model's robustness and generalization capability.

In summary, this ablation study clearly illustrates the performance progression from CE Loss to Weighted Focal Loss, validating the effectiveness and superiority of the final loss function adopted in this research for addressing the class imbalance problem in IoT device identification tasks.

4.5.2. Prior-Guided Initialization Strategy

To verify the contribution of the prior-guided initialization strategy to the performance of our method, we designed an ablation study comparing the recognition accuracy of two initialization schemes—with and without prior guidance—on the UNSW and CICIOT datasets. The experimental results clearly demonstrate that the prior-guided strategy plays a key role in improving model performance. As shown in the figure, on the UNSW dataset, the model with prior-guided initialization achieved an accuracy of 98.25%, representing a significant improvement of 3.15 percentage points over the baseline model without this strategy. On the CICIOT dataset, a consistently positive effect was observed: the prior-guided strategy increased accuracy from 94.85% to 97.72%, a gain of 2.87 percentage points.

These results fully demonstrate that the prior-guided initialization strategy provides the model with a better starting point by incorporating domain knowledge, enabling it to converge more quickly and stably to a superior solution. This effectively avoids the potential performance degradation in the early stages of traditional semi-supervised learning caused by low-quality pseudo-labels. Therefore, this strategy is a crucial guarantee for the high accuracy achieved by the proposed method under low-label conditions.

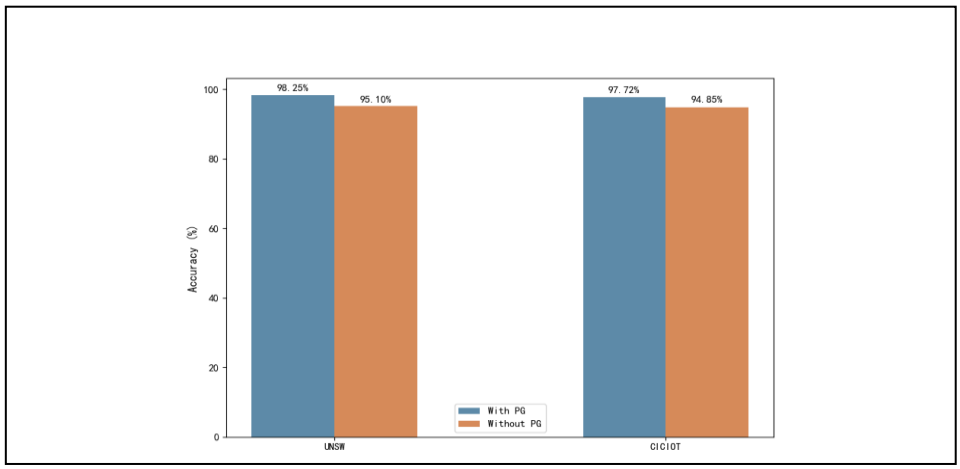


Figure 3. Performance Comparison of Initialization Strategies With and Without Prior Boot.

5. Conclusion

This paper proposes a lightweight IoT device identification scheme based on semi-supervised learning. By converting raw network traffic into grayscale images as feature input, it designs a prior-guided adaptive threshold mechanism called FreeMatch-PG to address the scarcity of labeled data and class imbalance. The scheme adopts an improved ResNet-18 architecture, achieving a lightweight design that reduces the number of parameters by 70% while enabling continuous identification of new device types. Experiments show that, using only 8% of labeled data, the proposed method achieves recognition accuracies of 98.25% and 97.72% on the UNSW and CICIOT datasets, respectively. It significantly outperforms traditional supervised learning methods and demonstrates clear advantages in computational efficiency, deployability, and scalability. This provides an efficient and feasible path for device identification in real-world IoT environments.

References

[1] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

[2] Ning, H., & Liu, H. (2015). Cyber-physical-social thinking based on the Internet of Things. IEEE Internet of Things Journal, 2(4), 288-295.

[3] Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., & Sivaraman, V. (2018). Classifying IoT devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing, 18(8), 1745-1759.

[4] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., & Li, C. L. (2020). FixMatch: Simplifying semi-supervised learning with consistency and confidence. Proceedings of the 37th International Conference on Machine Learning(ICML), 119, 8967-8978.

[5] Wang, W., Zhou, T., Yu, F., Yang, J., Yu, C., & Liu, S. (2021). Traffic image representation for network anomaly detection: A survey. IEEE Communications Surveys & Tutorials, 23(2), 1024-1059.

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition.

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 770-778.

[7] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. Proceedings of the 2015 Military Communications and Information Systems Conference(MilCIS), 1-6

[8] Feng Y, Zhang Y, He H, et al. An IoT Device Identification Method Using Extracted Fingerprint From Sequence of Traffic Grayscale Images[J]. IEEE Transactions on Dependable and Secure Computing, 2024.

[9] Miettinen M, Marchal S, Hafeez I, et al. Iot sentinel: Automated device-type identification for security enforcement in iot[C]//2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE, 2017: 2177-2184.

[10] Fan L, He L, Wu Y, et al. AutoIoT: Automatically updated IoT device identification with semi-supervised learning[J]. IEEE Transactions on Mobile Computing, 2022, 22(10): 5769-5786.

[11] Nukavarapu, S.K. and Nadeem, T.(2021) Securing Edge-based IoT Networks with Semi-Supervised GANs. 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 579-584.

[12] Jin, Y., Zhou, J., & Gao, Y. (2024). HSGAN-IoT: A hierarchical semi-supervised generative adversarial networks for IoT device classification. Computer Networks, 243, 110299.

[13] Wang, Y., Chen, H., Heng, Q., Hou, W., Fan, Y., Wu, Z., ... & Xie, X. (2022). Freematch: Self-adaptive thresholding for semi-supervised learning. arXiv preprint arXiv:2205.07246.

[14] Sajjad Dadkhah, Hassan Mahdikhani, Priscilla Kyei Danso, Alireza Zohourian, Kevin Anh Truong, and Ali A Ghorbani. Towards the development of a realistic multidimensional iot profiling dataset. In 2022 19th Annual International Conference on Privacy, Security & Trust (PST), pages 1–11. IEEE, 2022.

[15] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing, 18(8):1745–1759, 2018.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[18] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

[20] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, 214-226.

[21] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2980-2988.