







Research Article

LightSEEN: Real-Time Unknown Traffic Discovery via Lightweight Siamese Networks

Ji Li ¹, Chunxiang Gu ^{1,2}, Fushan Wei ¹, Xieli Zhang¹, Xinyi Hu ¹, Jiaxing Guo ¹,
and Wenfen Liu ³

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China

²Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China

³School of Computer Science and Information Security, Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China

Correspondence should be addressed to Chunxiang Gu; gcx5209@126.com

Received 2 July 2021; Accepted 30 September 2021; Published 18 October 2021

Academic Editor: Weiwei Liu

Copyright © 2021 Ji Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increase in the proportion of encrypted network traffic, encrypted traffic identification (ETI) is becoming a critical research topic for network management and security. At present, ETI under closed world assumption has been adequately studied. However, when the models are applied to the realistic environment, they will face unknown traffic identification challenges and model efficiency requirements. Considering these problems, in this paper, we propose a lightweight unknown traffic discovery model LightSEEN for open-world traffic classification and model update under practical conditions. The overall structure of LightSEEN is based on the Siamese network, which takes three simplified packet feature vectors as input on one side, uses the multihead attention mechanism to parallelly capture the interactions among packets, and adopts techniques including 1D-CNN and ResNet to promote the extraction of deep-level flow features and the convergence speed of the network. The effectiveness and efficiency of the proposed model are evaluated on two public data sets. The results show that the effectiveness of LightSEEN is overall at the same level as the state-of-the-art method and LightSEEN has even better true detection rate, but the parameter used in LightSEEN is 0.51% of the baseline and its average training time is 37.9% of the baseline.

1. Introduction

Network traffic identification refers to classifying network traffic into different sets by observing its characteristics according to specific targets, which is the focus of network behaviour analysis, network planning and construction, network anomaly detection, and network traffic model research [1]. In recent years, with the rapid development of network technology and the widespread use of encryption technology in the network, the amount of encrypted network traffic has gained a fierce increase, and the issue of encrypted traffic identification (ETI) has attracted wide attention from researchers.

Currently, ETI in closed environments has been amply studied. However, for the application in an open-world

environment, there are more practical problems to be considered, including the challenge of unknown traffic discovery and model efficiency.

To be deployable to practical applications, an ETI model needs to discover unknown classes of traffic that were not anticipated in the training phase. However, most of the existing models are based on the closed-world assumption, which means that the training dataset is assumed to contain all the traffic classes in the model deployment environment. However, such assumption cannot be held in many practical applications. Consequently, the classifier trained with a closed set is easy to classify the samples from an unknown class to some class in the training set mistakenly. To solve this problem, researchers try to develop models supporting both known class sample classification and unknown class sample discovery.

Recently, a model named SEEN is proposed for unknown traffic discovery [2], which applies the Siamese network in the ETI area for the first time. SEEN can classify known traffic into the correct classes and distinguish unknown traffic. However, the traffic features that SEEN uses are relatively rough, and the network structure of SEEN is rather complicated, which limits its practical application.

Many ETI models with high classification accuracy use complex neural network structures, and some use models with a low degree of parallelism (e.g., RNN). These models have strong feature extraction ability, but they need a large amount of training data. Moreover, the efficiency of these models is not high enough, and the model update is complex, which makes it barely able to deal with the complex and changeable network environment.

In this paper, we focus on improving the real-time performance and flexibility of unknown traffic discovery. Inspired by SEEN and Transformer [3], we try to design appropriate inputs and neural networks for reducing the space and time complexity of our task. More precisely, the contribution of this paper includes the following:

- (1) We put forward a lightweight model LightSEEN for unknown traffic discovery. To the best of our knowledge, there are few lightweight deep learning methods in this area. The overall structure of LightSEEN is a Siamese network, and we use the multihead attention mechanism to capture the associations between packets and promote the degree of parallelism. Meanwhile, 1D-CNN is introduced for further feature extraction and integration, and we reuse part of the network structure to reduce parameter amount.
- (2) We design compact packet-level features as the network input, meaning that only the most informative field information and a small amount of payload are selected. In addition, to reduce the quality and length requirements of the packet stream, we try to shrink the number of packets used.
- (3) We analyse the efficiency and effectiveness of LightSEEN with abundant experiments on two public datasets. In the model, techniques including ResNet and layer normalization are used to increase the convergence speed of the model and avoid it from degradation. Experimental results show that the effectiveness of LightSEEN is overall at the same level as SEEN, whereas the parameter number of the former is 0.51% of the latter, and the average training time of the former is 37.9% of the latter.

The rest of this paper is organized as follows. In Section 2, we review the related work on unknown traffic discovery. In Section 3, we introduce the problem definition and the architecture of the Siamese network. The LightSEEN is presented in Section 4, followed by the corresponding analysis. In Section 5, we evaluate the efficiency and effectiveness of LightSEEN by conducting comparative experiments on two data sets. Finally, we conclude this paper in Section 6.

2. Related Work

In this section, under the background of encrypted traffic analysis, we briefly introduce the machine learning methods used to discover unknown traffic, which includes conventional machine learning methods and deep learning methods.

2.1. Conventional Machine Learning Methods. Firstly, we introduce the conventional machine learning methods for unknown traffic discovery briefly, mainly including semi-supervised and unsupervised methods.

Since under most circumstances, labeled samples are insufficient while unknown flows are sufficient, many existing results on unknown traffic identification use semi-supervised methods. In 2007, Erman et al. [4] firstly proposed a semisupervised classification method for traffic classification, in which the labeled training data was used to solve the problem of mapping from flow clusters to actual classes; thus, it could be used to classify known and unknown applications. In its subsequent work, Zhang et al. [5] proposed a robust statistical traffic classification (RTC) solution on the basis of [4] by combining supervised and unsupervised machine learning technology to solve the unknown Zero-Day application challenge in traffic classification. This method can identify the Zero-Day application traffic and accurately distinguish the applications of predefined classes, and its effectiveness was verified by comparative experiments. In the same year, Lin et al. [6] proposed UPCSS to detect unknown protocols, which was based on flow correlation and semisupervised clustering ensemble learning. Similarly, Ran et al. [7] proposed a semisupervised learning system for adaptive traffic classification in 2017, which adopted techniques including iterative semisupervised k -means and dynamically adding centers to select the optimal parameters and achieved high accuracy.

Considering that traffic data of known classes only accounts for a small part of the massive network traffic, researchers also try to extract unknown features from unlabeled data, namely, using unsupervised learning methods in network classification. Mapping the extracted clusters to classes is the main challenge in implementing these methods. In 2009, Este et al. [8] proposed a method based on SVM to solve multiclass classification problem, applied it to traffic classification, and carried out simple optimization, making the classifier trained with a small number of hundreds of samples classify traffic from different topological points on the Internet with high accuracy. Likewise, in 2018, Fu et al. [9] also proposed the FlowCop method based on multiple one-class classifiers, which could not only identify predefined traffic, but also detect undefined traffic with selected prominent features for each one-class classifier. Both of the solutions in [8, 9] are based on the method of multiple one-class classifiers, but the binary classifiers for each class in this method are heuristics. Moreover, this method relies on a predefined distance threshold, which may lead to unsatisfactory results. In 2019,

Le et al. [10] discussed the extent to which the self-organizing map (SOM) could be applied to network traffic analysis and malicious behaviors detection in practice. Experiment results showed that the approach could identify malicious behaviors both on network and service datasets used, and that it was also beneficial for security management with visualization capabilities for network/service data analytics.

The conventional machine learning methods have relatively low time and space cost, and they have scored some achievements in unknown traffic discovery. However, they suffer from a high dependence on expert experience for feature selection, which makes it laborious to build the models and limits their performance.

2.2. Deep Learning Methods. With the rapid development of deep learning and its wide application in various fields, many researchers have applied deep learning to unknown traffic identification.

In 2017, Ma et al. [11] used a CNN model to identify protocols in the complex network environment according to the protocol type of the application layer. Experiments showed that, in the payload information of about 200,000 traffic flows, the accuracy of identifying unknown protocol traffic was 86.05%. In 2019, Zhang et al. [12] proposed a method, DePCK, for identifying unknown traffic, which could divide the mixed unknown traffic into multiple clusters. Each cluster contained only one application traffic as much as possible, thus improving the clustering purity. This method uses a deep autoencoder to extract features from traffic and then lets flow correlation guide the process of pair-constrained k -means. In the same year, Zhu et al. [13] proposed a method using deep neural networks to select appropriate protocol flow statistical features with the help of known application layer protocols. They then used an improved semisupervised clustering algorithm to divide the protocols into different sets, achieving unknown protocol classification. In 2020, Wang et al. [14] proposed a CNN model for unknown protocol syntax analysis according to the characteristics of the bit-flow protocol data format. The model preprocesses the protocol data to obtain the image format data suitable for CNN and then lets CNN process the image data to obtain the prediction results of the unknown protocol. Besides, Zhang et al. [15] studied how extreme value theory (EVT) could be utilized in unknown network attack detection systems and brought out a network intrusion detection method. By fitting the activation of the known class to the Weibull distribution, the open-CNN model was constructed to estimate the pseudo-probability of the unknown class from the activation score of the known class to achieve the purpose of detecting unknown attacks. In addition, Yang et al. [16] proposed a transfer learning method using deep adaptation networks (DAN). This method first trains a CNN model on the unlabeled data set with sampling time-series features, then jointly trains the extended version of the model on the labeled and unlabeled samples, uses labeled samples of known traffic to improve the clustering purity of unknown

traffic. This method achieves a purity of 98.23% on two published data sets.

Most of the above works need prior knowledge of unlabeled traffic, leading to their insufficient capability of fine-grained identification of unknown traffic. To fix this problem, Chen et al. [2] firstly applied the Siamese network to unknown traffic discovery. Their method, SEEN, can classify known traffic into correct classes and distinguish unknown traffic. However, the rough traffic features that SEEN uses and its bloated network structure make it not suitable for a realistic environment. Compared with SEEN, the method in this paper uses simplified input and a carefully designed lightweight network, which makes it more practical.

3. Preliminaries

In this section, we briefly review the preliminaries used in our model, including the definition of unknown traffic discovery and the work process of the Siamese network.

3.1. Problem Definition. Encrypted traffic identification refers to using rules or models to give traffic samples correct labels. It can be conducted with multiple granularities, including packet, flow, and host level [17]. In this paper, we focus on bidirectional flow analysis. A bidirectional flow is composed of all packets with the same quintuple values, that is, source IP, source port, destination IP, destination port and transport layer protocol, in which the source and destination are interchangeable [18]. For a flow f_i containing N packets, it can be expressed as shown in equation (1). Typically, a model for traffic identification is trained with labeled flow data firstly, and the trained model is used to classify flow samples without label into classes correctly. In particular, the model may face the unknown traffic discovery problem in practice.

$$f_i = \{p_1^i, \dots, p_N^i\}. \quad (1)$$

Unknown traffic discovery requires a classifier to reject a flow from classes unseen during training rather than assigning it an incorrect label [19]. Given a training set $D_{\text{train}} = \{(f_1, y_1), \dots, (f_n, y_n)\}$, where f_i is the i th flow sample and $y_i \in \Omega^K = \{C_i^K, i = 1, \dots, P\}$ is its corresponding class label, the goal is to learn a classifier \mathcal{A} that can not only classify the samples from known classes correctly but also categorize samples from unknown classes as unknown. For a test sample f_* , whose actual class label is y_* , the ideal effect of \mathcal{A} is shown in the following equation:

$$\mathcal{A}(f_*) = \begin{cases} y_*, & \text{if } y_* \in \Omega^K, \\ \text{unknown}, & \text{if } y_* \notin \Omega^K. \end{cases} \quad (2)$$

3.2. Siamese Network. Siamese neural network is a class of network architectures that consists of two (or more) identical subnetworks. The subnetworks have the same structure with the same parameters and shared weights, which are synchronously updated. A loss function connect them at the end, which computes a similarity metric based on the Euclidean

distance between the feature representations produced by the subnetworks. A commonly used loss function in the Siamese network is the contrastive loss [20] defined as follows:

$$L(x_1, x_2, y) = \alpha(1 - y)D_w^2 + \beta y \max(0, m - D_w)^2, \quad (3)$$

where x_1 and x_2 are two samples, y is a binary label denoting whether the two samples are of the same class or not, α and β are constants, and m is the margin. $D_w = \|f(x_1; w_1) - f(x_2; w_2)\|_2$ is the Euclidean distance in the embedded feature space, f is an embedding function mapping a sample to the feature space via neural networks, and w_1 and w_2 are the learned networks weights.

Siamese network aims to let the loss function bring the output feature vectors of similar inputs closer and push those of dissimilar inputs away. Then, to decide if two inputs belong to the same class, one needs to determine a threshold value on the feature vector distance. If the distance between the two inputs is smaller than the threshold, they are treated as similar samples or from the same class. Otherwise, they are judged as from different classes.

4. The Lightweight Model for Unknown Traffic Discovery

In this section, we introduce the lightweight model for unknown traffic discovery we proposed, and Figure 1 displays the structure of the model. Besides, we illustrate the details of model training, model validation, model test, and system update and also analyse the space and time complexity of the model.

4.1. Model Structure. As mentioned in Section 3.2, the Siamese network is generally composed of two identical subnetworks, which are joined by the margin-based loss function at the end. Therefore, we only need to introduce the structure of one subnetwork to make the composition of the whole model clear.

In general, the subnetwork structure here consists of four parts, that is, preprocessing, feature embedding, attention module, and dense layer. Moreover, we will explain each part in detail.

Preprocessing: the purpose of preprocessing is to extract valuable packet information as features. For a raw PCAP file, the packets in it can be combined into flows according to the quintuple. Then, the flow can be preprocessed by extracting its packet features, which are carefully designed for the lightweight traffic analysis task.

Considering our lightweight detection task, we choose features as lean as possible. In detail, firstly, the three-way handshake is skipped since it can barely provide information for traffic classification. Besides, only the first $N = 3$ packets are picked to get features, which are most likely to disclose useful information. For each packet, $S = 5$ fields of features are concerned, namely, position, timestamp, direction, key flags in IP and TCP header, and packet payload, and the details and meanings of the features are as follows.

- (1) Position (one dimension): it is the sequence number of a packet in a flow, which provides order information.
- (2) Timestamp (one dimension): it marks the arrival time of a packet, which provides temporal information.
- (3) Direction (two dimensions): a bidirectional flow includes packets of two directions, namely, from source to destination and the reverse direction, which can be represented as [0,1] and [1,0].
- (4) Key flags in IP and TCP header (nine dimensions): the key flags include ip_len, ip_off, ip_ttl, PSH, URG, th_seq, th_urp, and th_win. The ip_len means packet length, the ip_off means fragment offset, the ip_ttl stands for Time to Live, the PSH indicates the data transmission pattern, the URG means urgent data, the th_seq means the relative sequence, the th_urp is the urgent data offset, and the th_win means the window size. Other flags are abandoned since they do not contribute to the task.
- (5) Packet payload (77 dimensions): if the payload is less than 77 bytes, it will be completed with zero bytes and conversely truncated to 77 bytes.

4.1.1. Feature Embedding. The embedding layer converts the raw packet features into packet vectors that can be better analysed by the neural networks. Since we have multiple features with different dimensions, how to integrate them is worth studying, and there are a wide range of choices. A recent work on this is [21], in which a method of unifying different kinds of features' dimensions was proposed. Besides, feature fusion can also be achieved by neural networks. However, in this work, to reduce model complexity and promote efficiency, we choose to concatenate the raw features directly as a simple embedding. Let $\{x_i, i = 1, \dots, S\}$ denote the raw features obtained from preprocessing and $\{\mathbf{p}_i, i = 1, \dots, N\}$ denote the packet vectors generated by feature embedding; then, we have

$$\mathbf{p}_i = \{x_1, \dots, x_S\}, \quad i = 1, \dots, M. \quad (4)$$

Then, the packet vector \mathbf{p}_i has a dimension of $d_p = 90$, and $\{\mathbf{p}_i, i = 1, \dots, N\}$ will be the input of the attention module, which has a dimension of $d = Nd_p = 270$.

4.1.2. Attention Module. The structures of the attention module and dense layer are shown in Figure 2. The design of the attention module is partly derived from the Transformer encoder, and we adjust the network to support lightweight unknown discovery. In brief, we leverage the multihead attention mechanism to capture the interactions between different packets, reuse the basic block, and introduce 1-dimensional CNN (1D-CNN) to accumulate information and decrease the scale of network parameters.

The attention module mainly consists of three components, namely, (1) multihead attention, (2) add & norm, and (3) 1D-CNN, which will be explained in detail.

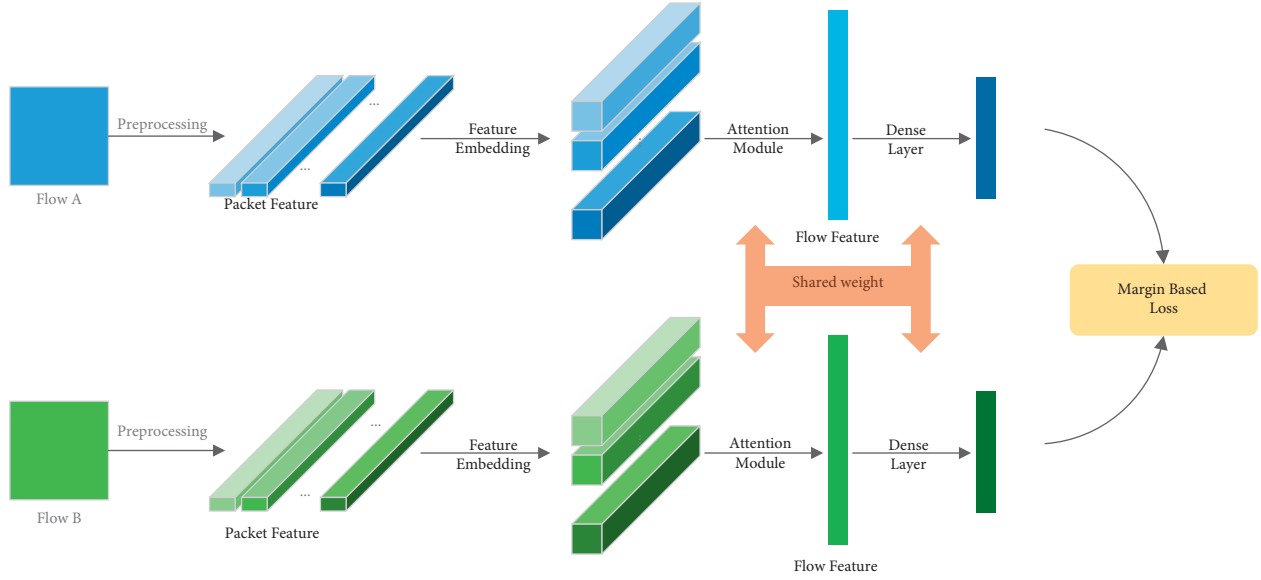


FIGURE 1: The mode structure of LightSEEN.

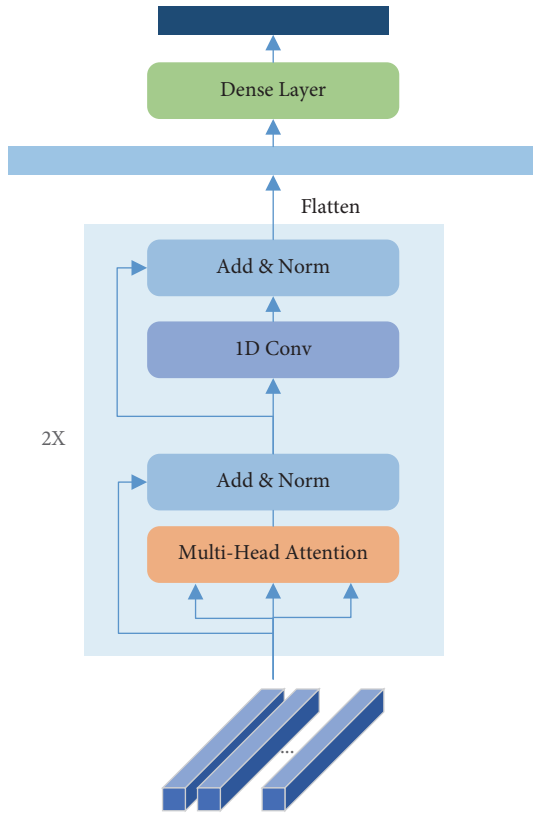


FIGURE 2: The construct of the attention module and the dense layer.

(1) *Multihead Attention*. The function of multihead attention is to jointly collect deep-level information of the input from multiple representation subspaces. We use packet k as an example to explain how the information delivered and organized efficiently to produce a new packet vector with deep-level features. The number of heads is denoted by H , so the

attention head $h \in \{1, \dots, H\}$. Let $\sigma^h(\cdot, \cdot)$ denote the relationship between two packets, and $\alpha_{k,l}^h$ denote the attention weight between packet k and l ; then,

$$\alpha_{k,l}^h = \frac{\sigma^h(\mathbf{p}_k, \mathbf{p}_l)}{\sum_{i=1}^N \sigma^h(\mathbf{p}_k, \mathbf{p}_i)}. \quad (5)$$

$\sigma^h(\cdot, \cdot)$ can be achieved by inner product or a neural network, and we choose inner product for better efficiency; hence,

$$\sigma^h(\mathbf{p}_k, \mathbf{p}_l) = \langle Q^h \mathbf{p}_k, K^h \mathbf{p}_l \rangle, \quad Q^h, K^h \in \mathbb{R}^{d' \times d}, \quad (6)$$

where Q^h and K^h are transformation matrices mapping the packet vector from original space \mathbb{R}^d into a new space $\mathbb{R}^{d'}$. Then, the representation of packet k of head h is

$$\hat{\mathbf{p}}_k^h = \sum_{i=1}^N \alpha_{k,i}^h (V^h \mathbf{p}_i), \quad V^h \in \mathbb{R}^{d_1 \times d}. \quad (7)$$

And, the packet vector in new space can be obtained by concatenating the $\hat{\mathbf{p}}_k^h$ of all heads:

$$\hat{\mathbf{p}}_k = \text{Concat}(\hat{\mathbf{p}}_k^1, \dots, \hat{\mathbf{p}}_k^H). \quad (8)$$

On the whole, the multihead attention mechanism updates the representation of all packets with the idea of weighted summation. For each packet, the weight is generated from its association with all the other packets in parallel, resulting in much fewer parameters and a much shorter running time. Besides, a packet is projected into different subspaces for capturing multiview feature associations. Since the computation of all the heads is also parallelized, it benefits for speeding up the model.

(2) *Add & Norm*. The add & norm part uses ResNet and layer normalization for avoiding network degradation and faster training. It has been discussed that the ResNet can make it

easier for information to flow between layers, including providing feature reuse in forward propagation and alleviating gradient signal disappearance in back propagation [22]. The effect of the ResNet on the representations of packet k can be expressed as

$$\bar{\mathbf{p}}_k = \text{ReLU}(\hat{\mathbf{p}}_k + \mathbf{W}_{\text{res}}\mathbf{p}_k), \quad (9)$$

where $\mathbf{W}_{\text{res}} \in \mathbb{R}^{d \times d}$ is a transformation matrix and ReLU is an activation function.

The subsequent layer normalization technique can normalize the distributions of mid-tier layers, making gradients smoother and generalization better.

(3) *1D-CNN*. The 1D-CNN part is designed for further mining the hidden patterns contained by the packet representation obtained from previous layers. Besides, compared with the fully connected network, CNN needs much fewer parameters. Specifically, the kernel size of the 1D-CNN layer is d and the channel number $r = d$, making the input and output dimensions consistent.

The output of 1D-CNN will be the input of another add & norm layer, and a basic block is composed of multihead attention, 1D-CNN and 2 add & norm layers, as is shown in Figure 2. The basic block is reused $T = 2$ times for a better balance between effectiveness and efficiency.

4.1.3. Dense Layer. Let $\{\tilde{\mathbf{p}}_i, i = 1, \dots, N\}$ denote the output of the whole attention module, which will be concatenated into a vector $\tilde{\mathbf{f}}$ as flow representation, shown in equation (10). Then, $\tilde{\mathbf{f}}$ will be fed into the dense layer, as equation (11) shows, and the output vector \mathbf{f} with length L will be the final flow vector.

$$\tilde{\mathbf{f}} = \text{Concat}(\tilde{\mathbf{p}}_1, \dots, \tilde{\mathbf{p}}_N), \quad (10)$$

$$\mathbf{f} = \text{ReLU}(\mathbf{W}_D \tilde{\mathbf{f}} + \mathbf{b}), \quad \mathbf{W}_D \in \mathbb{R}^{L \times N \cdot d}. \quad (11)$$

4.2. Model Training and Validation

4.2.1. Model Training. The model training means using labeled samples to train the Siamese network, and the first step is the pairwise dataset generation. Different from other networks, the input of the Siamese network is a pair of flows rather than a single data. Therefore, it is necessary to choose data from the labeled known class dataset to construct a new dataset containing positive and negative pairs. To be specific, a positive pair, which is labeled as 0, is a pair of flows that belong to the same class, and a negative pair with label 1 contains flows from different classes. To avoid the influence of imbalanced data, the ratio of positive to negative pairs is about 1 : 1. The model will learn a metric to tell similar and dissimilar pairs apart through these positive and negative samples.

Given a pair of flows f_i and f_j , the true label of the pair is denoted by l_t . Let x_i and x_j denote the corresponding raw flow features input to the network, and v_i and v_j denote the output of the network. The function of the network can be

represented as $F_L(x; \theta)$, where θ denotes the parameters; then, we have $v_i = F_L(x_i; \theta)$ and $v_j = F_L(x_j; \theta)$. The distance between f_i and f_j , denoted by $D_{i,j}$, can be calculated as follows:

$$D_{i,j} = D(F_L(x_i; \theta), F_L(x_j; \theta)) = \left[\sum_z (v_{iz} - v_{jz})^2 \right]^{1/2}. \quad (12)$$

With the pairwise dataset generated and the hyperparameter margin m set, the model can be trained for the binary classification problem. The margin-based loss function is shown in Section 3.2, which encourages positive pairs to be close together in the space of network mapping while pushing negative pairs apart. Let $\alpha = \beta = 1/2$; the loss function for our model training is shown as follows:

$$L(x_i, x_j, l_t) = \frac{1}{2} (1 - l_t) D_{i,j}^2 + \frac{1}{2} l_t \max(0, m - D_{i,j})^2. \quad (13)$$

4.2.2. Model Validation. The model validation means validating that the model can differentiate the positive and negative pairs with high accuracy. The positive and negative pairs are also generated from known classes. However, it is suggested that the pairs for validation should be avoided from overlapping with those in the training process. An appropriate method is generating a group of nonredundant pairs from the known classes and splitting the group into training and validation datasets. Besides, a threshold t should be determined through experience or attempts. Let l_t denote the true label of a pair and l_p denote the predicted label. If the Euclidean distance $D_{i,j}$ of a flow pair (f_i, f_j) is beyond the threshold, then the predicted label is $l_p = 1$; namely, the pair is judged as negative. Otherwise, the pair is judged as positive with the predicted label $l_p = 0$, as is shown in equation (14). Then, we can compare l_t and l_p of each pair, if they are the same, the judgment of the model is correct. The model validation can be used to validate the training result of the model and adjust the value of t and even the margin.

$$l_p = \begin{cases} 1, & \text{if } D_{i,j} > t, \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

4.3. Model Test and System Update

4.3.1. Model Test. Model test means using the trained LightSEEN model for traffic classification and unknown discovery; that is, it should not only classify the flows from known classes correctly but also detect flows from unknown classes. For a flow f , the actual label of f is denoted by $\Phi(f)$, and its predicted label is denoted by $\varphi(f)$. Recall that Ω^K is the set of known classes labels; then, the known flow dataset can be expressed by O_f^K in equation (15). The defined distance between flows is not enough for the task, a distance between a test flow sample f^* and a known class C_i^K must be defined. We use the same distance as [2] uses, which is

defined as the average distance between the test sample f^* and q samples $\{f_{ij}, j = 1, \dots, q\}$ randomly chosen from the class C_i^K . The calculation of the sample-class distance is expressed by equation (16). Then, there is a known class C_*^K which is the closest to f^* ; let D_{f^*} denote the distance. If D_{f^*} is not larger than the preset threshold t , it is decided that f^* belongs to C_*^K . Otherwise, f^* belongs to no known class; namely, its class is unknown. The model test algorithm is shown in Algorithm 1.

$$O_f^K = \{f | \Phi(f) \in \Omega^K = \{C_1^K, \dots, C_P^K\}\}, \quad (15)$$

$$D(f^*, C_i^K) = \frac{1}{q} \sum_{j=1}^q D(f^*, f_j), \Phi(f_j) = C_i^K, \quad j = 1, \dots, q. \quad (16)$$

The model test dataset is made up equally of flow samples from known and unknown classes, and the related metrics will be introduced in Section 5.1.

4.3.2. System Update. To update the system, an unsupervised framework should be leveraged to divide the detected unknown traffic into multiple clusters, which can be used as a supplement to known classes. The trained network can be used as an encoder to convert the original flow data to high-level feature vectors, which can be clustered by existing algorithms like k -means. After that, the clusters are identified through manually labeling and used to complement the system's identification area. For instance, if we want to add a new class C^* to the known classes, we only need to use samples from C^* and known classes to generate positive and negative pairs, with at least one sample from C^* in each pair. Then, we use the generated pairs to retrain the model, and we get a model for $P + 1$ known classes.

4.4. Space and Time Complexity Analysis. For deep learning models, the space complexity is related to its parameter amount, and the time complexity depends on its inner structure. Table 1 shows the space and time complexity of LightSEEN, and the corresponding analysis is as follows.

4.4.1. Space Complexity. In our method, there is no parameter in the embedding layer. For the attention module, the parameters are mainly in the weight matrices, and the scale is about $T \times \{Q^h, K^h, V^h, \mathbf{W}_{\text{res}}, \mathbf{W}_{\text{ID-CNN}}\}$, which is $O(\text{THdd}' + \text{Tdr})$. The parameter scale of the dense layer is $O(\text{NdL})$. Therefore, the total space complexity is $O(\text{THdd}' + \text{Tdr} + \text{NdL})$.

4.4.2. Time Complexity. In the attention module, for each head, the time cost for attention weight calculation is $O(\text{Ndd}' + N^2 d')$ and that for combinatorial feature formulation is $O(N^2 d d')$. Besides, the time complexity of the add & norm, 1D-CNN, and dense layer are $O(\text{Hdd}')$, $O(\text{Nd}^2 r)$, and $O(\text{NdL})$, respectively. Considering that the

attention module reuse T times, the total space complexity is $O(\text{Tdd}'(H + N^2) + \text{TNd}^2 r + \text{NdL})$.

5. Experimental Evaluation

In this section, we evaluate the effectiveness and efficiency of LightSEEN. Since LightSEEN is built to enhance the real-time performance and flexibility of deep learning based unknown traffic discovery, we mainly compare the LightSEEN method with SEEN [2].

5.1. Experiment Setup

5.1.1. Datasets and Partition Strategy. We tested the performance of LightSEEN on two extensively used public traffic datasets, namely, USTC-TFC2016 [23] and ISCXVPN2016 [24]. As is shown in Table 2, USTC-TFC2016 contains 20 classes of traffic, of which half are malware traffic classes. The ISCXVPN2016 dataset includes seven classes of regular encrypted traffic and seven classes of traffic through the VPN encrypted tunnel, and we use 12 of them to conduct experiments.

The partition strategy for known and unknown sets is the same as that in [2], namely, some classes are manually set as unknown classes, including three malware classes and three normal classes traffic from USTC-TFC2016, two VPN classes, and two non-VPN classes from ISCXVPN2016.

Experiment Environment and Details: as for the experiment environment, we used PyTorch 1.8 to implement the structure of LightSEEN. Note that the training and testing processes were performed on a Linux machine (Ubuntu 16.04 LTS) with 32 GB RAM and GeForce Gtx1080. The training process is guided by minimizing the contrastive loss, and we take the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The parameters of LightSEEN are shown in Table 3. The dropout strategy is applied with a keep proportion of 0.9 for the multihead attention part and 0.6 for the dense layer. The learning rate is 0.0002, and batch size in model training is 128. For the balance of efficiency and effectiveness, we set the margin $m = 6$ for USTC-TFC2016 and $m = 12$ for ISCXVPN2016.

5.1.2. Evaluation Metrics. The performance of LightSEEN mainly includes the efficiency and effectiveness of unknown discovery. To evaluate its efficiency, we count the training and test time per 100 batches, and the average training and test time are used as evaluation metrics. As to its effectiveness, four evaluation metrics are used [25]: purity rate (PR), accuracy (Acc), false detection rate (FDR), and true detection rate (TDR). To illustrate the metrics, some other symbols are defined. KP (known positive) denotes the number of the known class flows correctly identified, KN (known negative) denotes the number of the known class flows mistaken for other known classes, UP (unknown positive) denotes the number of unknown class flows detected, and UN (unknown negative) denotes the number of unknown flows wrongly classified as known. Then, the metrics can be computed with these statistics as follows.

Input: test flow sample f^* , known class flow dataset O_f^K , number of samples for average distance calculation q , threshold t .
 (1) Calculate the distance between f^* and each class $\{D(f^*, C_i^K), i = 1, \dots, P\}$;
 (2) Find the class with the shortest distance away from f^* , denoted as C_*^K ;
 (3) If $D(f^*, C_*^K) \leq t$, $\varphi(f^*) = C_*^K$. Otherwise, $\varphi(f^*) = \text{unknown}$.
 Output: the predicted class $\varphi(f^*)$.

ALGORITHM 1: Model test algorithm.

TABLE 1: Space and time complexity of the proposed model.

Layer	Space complexity	Time complexity
Attention module	$O(\text{THdd}' + \text{Tdr})$	$O(T(N^2 + H)dd' + \text{TNd}^2r)$
Dense layer	$O(\text{NdL})$	$O(\text{NdL})$
All	$O(\text{THdd}' + \text{Tdr} + \text{NdL})$	$O(\text{Tdd}'(N^2 + H) + \text{TNd}^2r + \text{NdL})$

TABLE 2: USTC-TFC2016 dataset and ISCXVPN2016 dataset.

Dataset	Labels
USTC-TFC2016	Cridex, Ceodo, Hitbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut, Zeus, BitTorrent, Facetime, FTP, Gmail, MySQL, Outlook, Skype, SMB, Weibo, and WorldOfWarcraft
ISCXVPN2016	Chat, Email File, P2P, Streaming, VoIP, VPN-Chat, VPN-Email, VPN-File, VPN-P2P, VPN-Streaming, and VPN-VoIP

TABLE 3: The parameters of LightSEEN.

Meaning	Parameter	Value
Number of packets	N	3
Times of attention module reuse	T	2
Number of headers	H	3
Dimension of embedding	d/r	270
Dimension of attention head mapping	d_l	90
Dimension of flow vectors	L	200

From the equations, it is easy to see that the solution with high PR, Acc, and TDR and low FDR has favorable performance.

$$\begin{aligned}
 \text{PR} &= \frac{\text{KP} + \text{UP}}{\text{KP} + \text{KN} + \text{KU} + \text{UP} + \text{UN}}, \\
 \text{Acc} &= \frac{\text{KP}}{\text{KP} + \text{KN} + \text{KU}}, \\
 \text{FDR} &= \frac{\text{KU}}{\text{KP} + \text{KN} + \text{KU}}, \\
 \text{TDR} &= \frac{\text{UP}}{\text{UP} + \text{UN}}.
 \end{aligned} \tag{17}$$

Besides, the clustering purity (CP) is used to evaluate the performance of LightSEEN as a feature extractor, which will be explained in detail in Section 5.5. The definition of Clustering Purity is shown in equation (18), where $|D|$ is the number of samples, $\Omega = \{w_i, i = 1, \dots, K\}$ is the set of clusters, and $C = \{c_j, j = 1, \dots, J\}$ is the set of classes.

$$\text{CP}(\Omega, C) = \frac{1}{|D|} \sum_{i=1, \dots, K} \max_{j=1, \dots, J} |w_i \cap c_j|. \tag{18}$$

5.2. Selection of Hyperparameters. There are two hyperparameters that are not used in model training but indispensable for unknown traffic discovery (i.e., k (the number of compared samples from each class to calculate class average distance) and t (the threshold for determining whether the test sample belongs to a certain class)). To obtain the reasonable range of t , we use the trained model to predict pairwise Euclidean distances of the two datasets and display the corresponding histograms in the range $[0, 7]$ in Figure 3. We set green bars for positive pairs and orange bars for negative pairs. It can be seen that the distances of positive pairs are close to 0, and those of negative pairs are mostly far from 0. The coincided field of green and orange bars mainly lies in $[0, 1.5]$ for USTC-TFC2016 and $[0.5, 2.5]$ for ISCXVPN2016; thus, we choose $t = 1.2$ for USTC-TFC2016 and $t = 2.1$ for ISCXVPN2016. As to the number of samples k , experiments show that its influence on the performance of LightSEEN is small. Therefore, we adopt the same setting as [2] for the convenience of comparison between LightSEEN and SEEN, meaning that we set $k = 10$ for both datasets.

5.3. Effectiveness Analysis. To observe the effectiveness of LightSEEN under different situations, we change the percentage of unknown classes in the model test procedure from 10% to 50% and compare different models' performance. Figures 4 and 5 show the result comparison among LightSEEN, SEEN, and a one-class SVM method [8] on the USTC-TFC2016 and ISCXVPN2016 datasets, respectively. We set the green bars for LightSEEN, orange bars for SEEN, and blue bars for one-class SVM. It can be seen that the comparative advantages among the three methods are similar on the two datasets.

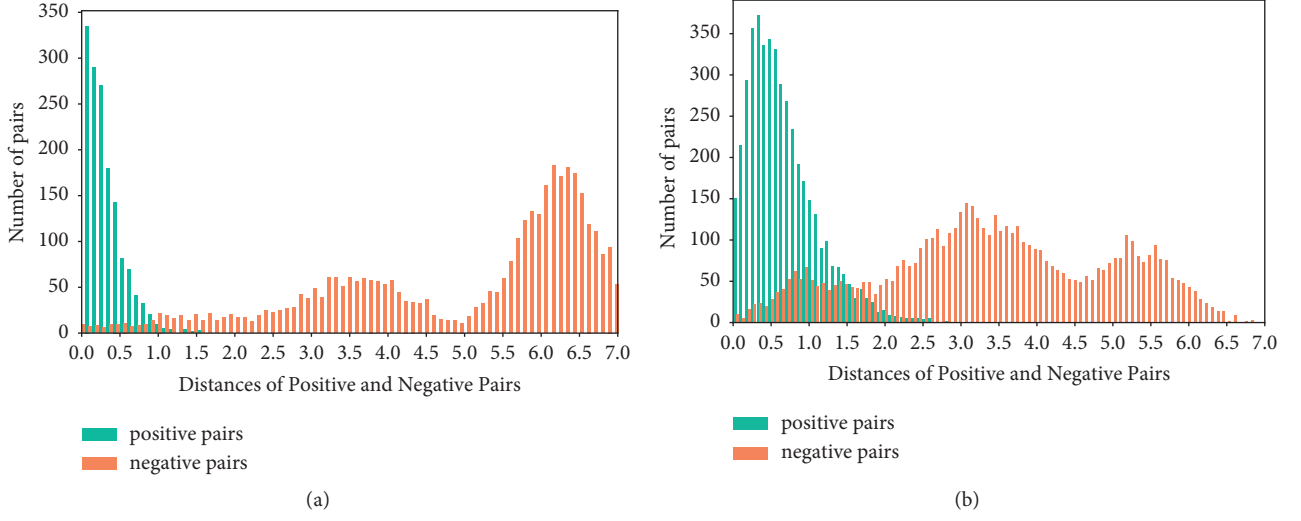


FIGURE 3: Part of the distance histogram of positive and negative pairs from two datasets. (a) USTC-TFC2016. (b) ISCXVPN2016.

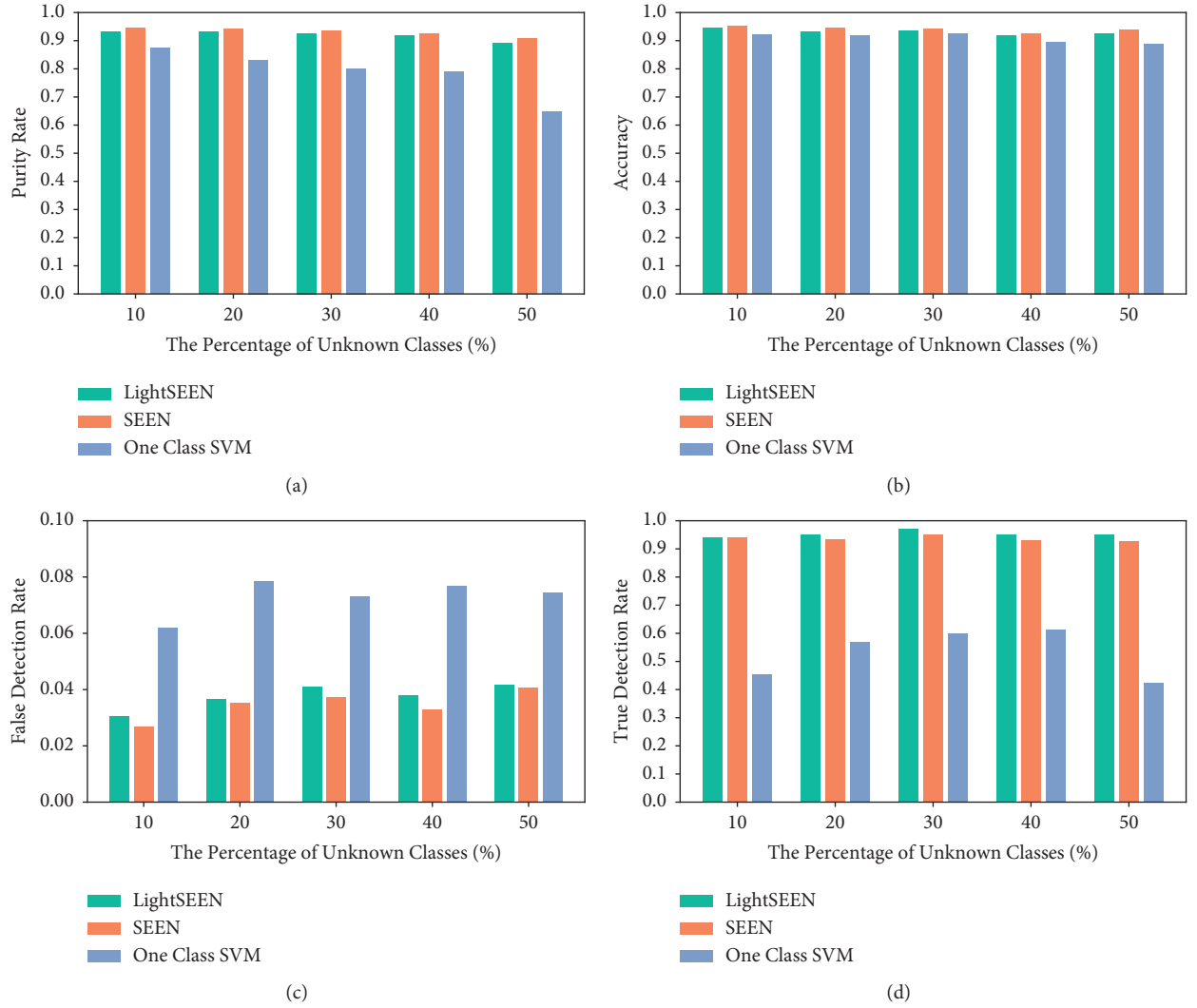


FIGURE 4: Performance comparison of different methods for the USTC-TFC2016 dataset.

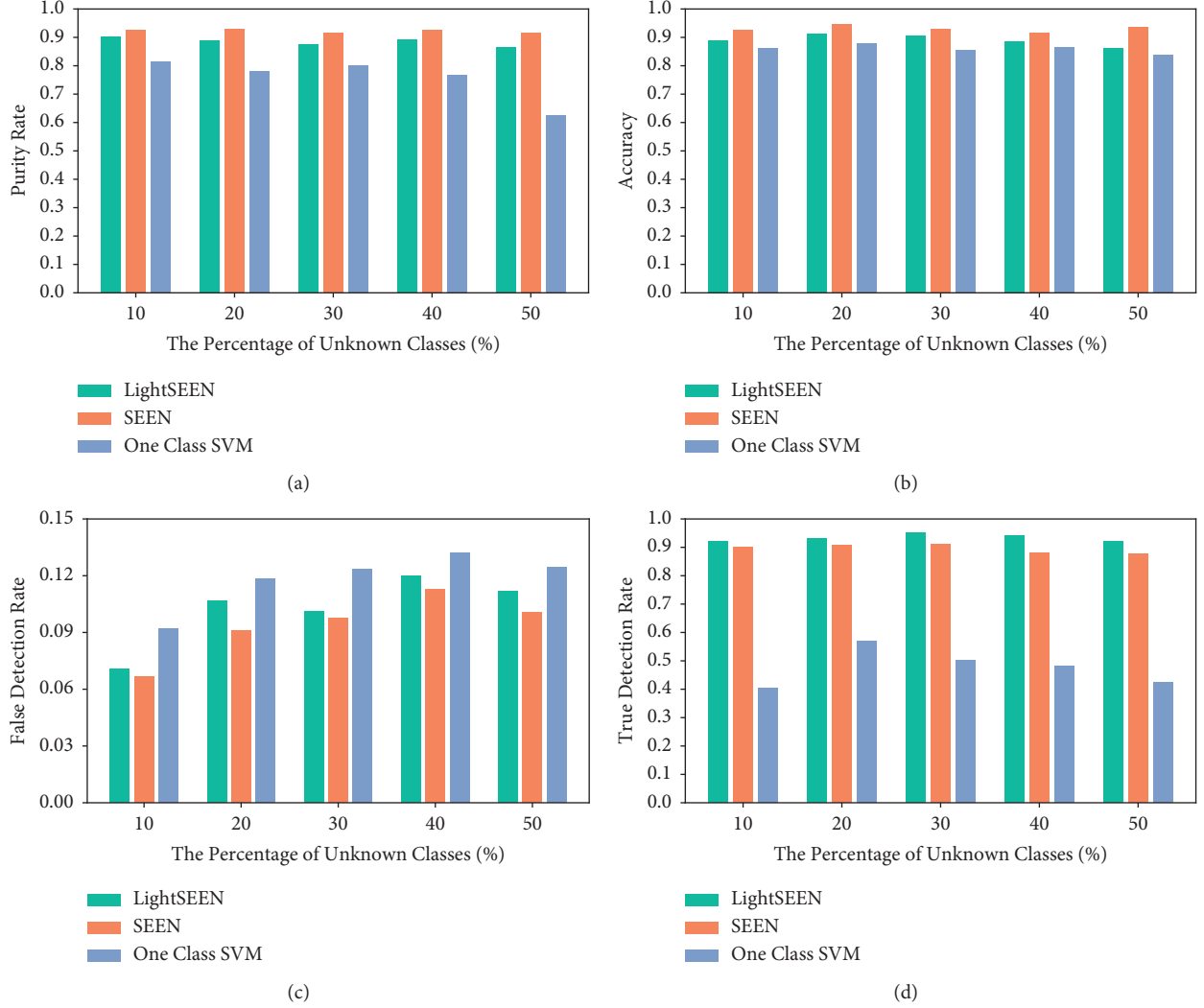


FIGURE 5: Performance comparison of different methods for the ISCXVPN2016 dataset.

To be specific, firstly, both of the purity rates (PR) of LightSEEN and SEEN are higher than that of one-class SVM and mostly above 0.9. Although the PR of LightSEEN is slightly lower than that of SEEN, they are very close, and both of them are stable with the percentage of unknown increases. The situation of the accuracy (Acc) result is almost the same as that of the PR result; that is, the result of SEEN is slightly better than LightSEEN on Acc, and both of them outperform the one-class SVM method. As for the false detection rate (FDR), SEEN has the lowest bar, LightSEEN's bar is slightly higher, and the one-class method's bar is the highest. Since a lower FDR means better performance, still SEEN is the best. However, for the true detection rate (TDR), LightSEEN is higher than SEEN and the one-class method. Note that in some reality applications like intrusion detection, the TDR is significantly crucial, meaning that LightSEEN is the best choice under these circumstances.

In summary, the effectiveness of LightSEEN on evaluation metrics is overall at the same level as SEEN and sometimes even better, meaning that its effectiveness is validated.

5.4. Efficiency Analysis. We demonstrate the efficiency of LightSEEN from three aspects, namely, quantity of parameters, average training time, and average test time. To promote the model efficiency, we take measures including multihead attention and reuse of the basic block in the attention module. Table 4 shows the comparison results of efficiency between LightSEEN and SEEN. The parameter number of our LightSEEN model is about 648000, which is 0.51% of that of SEEN. Besides, LightSEEN's average training time is 37.4 ms, which is 37.9% of that of SEEN. And its average test time is also obviously shorter. Through the efficiency analysis results, we can draw the conclusion that we substantially reduce the scale of model parameters and training time cost in LightSEEN, whose efficiency has been validated.

5.5. Unknown Clustering. In this part, we explore the performance of LightSEEN as a feature extractor. After detecting unknown traffic, we can separate them into different groups through clustering algorithm and update the

TABLE 4: The comparison results of efficiency.

Index	LightSEEN	SEEN
Quantity of parameters	648 k	126180 k
Average training time (ms)	37.4	98.6
Average test time (ms)	10.5	23.8

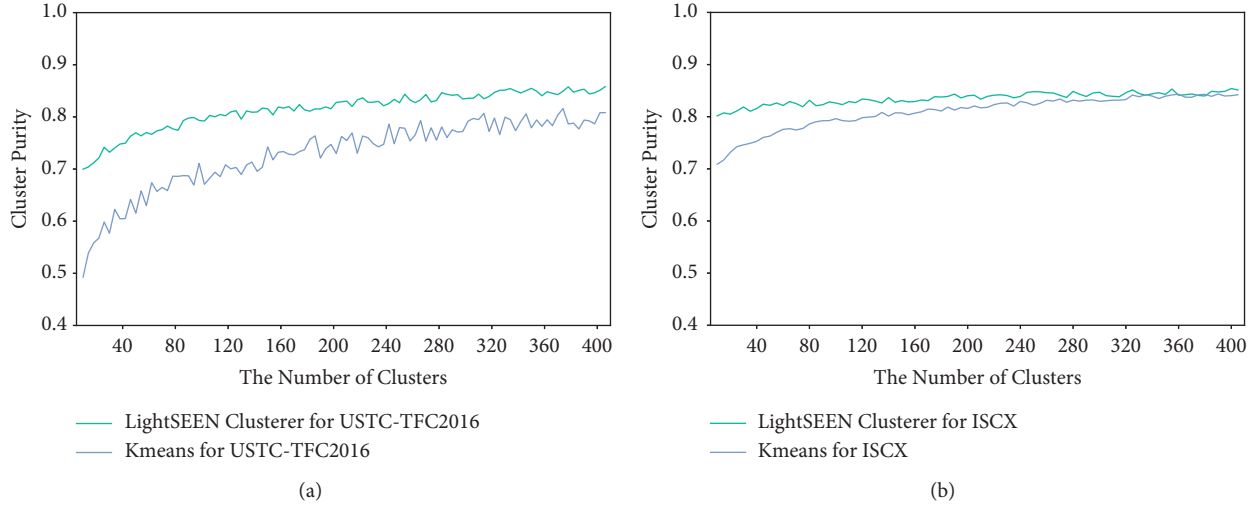


FIGURE 6: Clustering purity comparison of different methods for two datasets. (a) USTC-TFC2016. (b) ISCXP2016.

model as mentioned in Section 4.3. Moreover, we suggest that the clustering algorithm should be operated on the flow vectors output by the network of LightSEEN rather than the raw flow features.

We compare the clustering purity of traffic data with and without the processing of the network in LightSEEN, and Figure 6 shows the corresponding result. The blue line reveals the result of directly applying k -means to the flow vector composed by concatenating raw packet feature vectors and the green line for operating on the output of the trained network instead. It indicates that LightSEEN can extract deep features that more discriminative than raw features.

6. Conclusion

In this paper, we propose a lightweight model for unknown traffic discovery. Specifically, the model takes the cautiously selected packet features as input, adopts the Siamese network architecture, and guides the training process by contrastive loss. To capture the associations between packets and improve the parallel degree of the model, we use the multihead attention mechanism within the network. Besides, we introduce 1D-CNN, ResNet, and layer normalization, and reuse the basic modules to facilitate the model convergence with a limited number of parameters. The experimental results show that the model is effective and efficient.

In the future, further work can be done on open-set traffic recognition. Firstly, the contrastive loss in our model can be replaced by better loss functions (e.g., circle loss [26] and ArcFace loss [27]). Furthermore, the model can be applied to other practical tasks, including intrusion

detection and malicious traffic discovery. When intruders and attackers carry out actions against information systems, there will be anomalous traffic, which can be seen as unknown traffic and detected by unknown discovery systems.

Data Availability

The USTC-TFC2016 and ISCXP2016 data used to support the findings of this study are included within the article.

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61772548 and 61862011) and in part by the Guangxi Science and Technology Foundation (nos. 2018GXNSFAA138116 and 2019GXNSFGA245004).

References

- [1] M. Roughan, S. Sen, O. Spatscheck, and N. G. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference, IMC 2004*, A. Lombardo and J. F. Kurose, Eds., ACM, Taormina Sicily, Italy, pp. 135–148, October 2004.
- [2] Y. Chen, Z. Li, J. Shi, G. Gou, C. Liu, and G. Xiong, "Not afraid of the unseen: a siamese network based scheme for unknown traffic discovery," in *Proceedings of the IEEE Symposium on*

- Computers and Communications, ISCC 2020*, pp. 1–7, IEEE, Rennes, France, July, 2020.
- [3] A. Vaswani, N. Shazeer, N. Parmar et al., “Attention is all you need,” in *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, L. Beach, U. S. A. CA, I. Guyon et al., Eds., pp. 5998–6008, Long Beach, CA, USA, December, 2017.
 - [4] J. Erman, A. Mahanti, M. F. Arlitt, I. Cohen, and C. L. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Performance Evaluation*, vol. 64, no. 9–12, pp. 1194–1213, 2007.
 - [5] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust network traffic classification,” *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2015.
 - [6] R. Lin, O. Li, Q. Li, and Y. Liu, “Unknown network protocol classification method based on semi-supervised learning,” in *Proceedings of the 2015 IEEE International Conference on Computer and Communications (ICCC)*, pp. 300–308, IEEE, Chengdu, China, October 2015.
 - [7] J. Ran, X. Kong, G. Lin, D. Yuan, and H. Hu, “A self-adaptive network traffic classification system with unknown flow detection,” in *Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1215–1220, IEEE, Chengdu, China, December 2017.
 - [8] A. Este, F. Gringoli, and L. Salgarelli, “Support vector machines for TCP traffic classification,” *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.
 - [9] N. Fu, Y. Xu, J. Zhang, R. Wang, and J. Xu, “Flowcop: detecting “stranger” in network traffic classification,” in *Proceedings of the 27th International Conference on Computer Communication and Networks, ICCCN*, pp. 1–9, IEEE, Hangzhou, China, August, 2018.
 - [10] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, “Unsupervised monitoring of network and service behaviour using self organizing maps,” *Journal of Cyber Security and Mobility*, pp. 15–52, 2019.
 - [11] R. Ma and S. Qin, “Identification of unknown protocol traffic based on deep learning,” in *Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1195–1198, IEEE, Chengdu, China, December 2017.
 - [12] Y. Zhang, S. Zhao, and Y. Sang, “Towards unknown traffic identification using deep auto-encoder and constrained clustering,” in *Proceedings of the 2019 19th International Conference Computational Science - ICCS*, J. M. F. Rodrigues, P. J. S. Cardoso, J. M. Monteiro et al., Eds., vol. 11536, pp. 309–322, Springer, Faro, Portugal, June, 2019.
 - [13] P. Zhu, S. Zhang, H. Luo, and Z. Wu, “A semi-supervised method for classifying unknown protocols,” in *Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1246–1250, IEEE, Chengdu, China, March 2019.
 - [14] Y. Wang, B. Bai, X. Hei, L. Zhu, and W. Ji, “An unknown protocol syntax analysis method based on convolutional neural network,” *Transactions on Emerging Telecommunications Technologies*, Wiley, Hoboken, NJ, USA, 2020.
 - [15] Y. Zhang, J. Niu, D. Guo, Y. Teng, and X. Bao, “Unknown network attack detection based on open set recognition,” *Procedia Computer Science*, vol. 174, pp. 387–392, 2020.
 - [16] Z. Yang and W. Lin, “Unknown traffic identification based on deep adaptation networks,” in *Proceedings of the 45th IEEE LCN Symposium on Emerging Topics in Networking, LCN Symposium 2020*, H. Tan, L. Khoukhi, and S. Oteafy, Eds., pp. 10–18, IEEE, Sydney, Australia, November 2020.
 - [17] S. Rezaei and X. Liu, “Deep learning for encrypted traffic classification: an overview,” *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.
 - [18] A. Dainotti, A. Pescapè, and K. C. Claffy, “Issues and future directions in traffic classification,” *IEEE Netw*, vol. 26, no. 1, pp. 35–40, 2012.
 - [19] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, “Toward open set recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.
 - [20] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pp. 539–546, IEEE Computer Society, San Diego, CA, USA, June 2005.
 - [21] W. Song, C. Shi, Z. Xiao et al., W. Zhu, “Automatic feature interaction learning via self-attentive neural networks,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*, D. Tao, X. Cheng, P. Cui et al., Eds., ACM, Beijing, China, pp. 1161–1170, November 2019.
 - [22] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *Proceedings of the Computer Vision - ECCV 2016 - 14th European Conference*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9908, pp. 630–645, Springer, Amsterdam, The Netherlands, October, 2016.
 - [23] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware traffic classification using convolutional neural network for representation learning,” in *Proceedings of the 2017 International Conference on Information Networking, ICOIN 2017*, pp. 712–717, IEEE, Da Nang, Vietnam, January, 2017.
 - [24] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of encrypted and VPN traffic using time-related features,” in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016*, O. Camp, S. Furnell, and P. Mori, Eds., SciTePress, Rome, Italy, pp. 407–414, February, 2016.
 - [25] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. V. Vasilakos, “An effective network traffic classification method with unknown flow detection,” *IEEE Transaction Network Service Management*, vol. 10, no. 2, pp. 133–147, 2013.
 - [26] Y. Sun, C. Cheng, Y. Zhang et al., “Circle loss: a unified perspective of pair similarity optimization,” in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pp. 6397–6406, IEEE, Seattle, WA, USA, June, 2020.
 - [27] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pp. 4690–4699, Computer Vision Foundation/IEEE, Long Beach, CA, USA, June, 2019.