Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1746373, 18 pages
https://doi.org/10.1155/2022/1746373

Research Article

OpenCBD: A Network-Encrypted Unknown Traffic Identification Scheme Based on Open-Set Recognition

Xinyi Hu,[1,2] **Chunxiang Gu**,[1,2] **Yihang Chen**,[1] **Xi Chen**,[1,2] **and Fushan Wei**[1,2]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China 450001*
[2]*Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China 450001*

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

Received 9 February 2022; Revised 28 March 2022; Accepted 12 April 2022; Published 12 May 2022

Academic Editor: Shafiq Ahmad

The encryption of network traffic promotes the development of encrypted traffic classification and identification research. However, many existing studies are only effective for closed-set experimental data, that is to say, only for traffic of known classes, while there are often lots of unknown classes traffic in the real environment of open sets, and many studies have difficulty identifying the traffic of unknown classes and can only misclassify them as known classes. How to identify unknown traffic and classify known traffic in an open-collection environment is one of the focuses of traffic analysis research. Considering these problems, this paper proposes a novel solution, which applies the open-set recognition method to the unknown traffic identification, and constructs a model based on deep learning and ensemble learning. The method constructs a model based on a convolutional neural network and a transformer encoder and then uses a three-stage training and testing process, combined with a novel loss function, to generalize to the open space to form OpenCBD. Experiments on public datasets show that the proposed method is significantly better than other open-set identification methods. It can not only distinguish known traffic from unknown traffic but also identify specific classes of known traffic.

1. Introduction

With the wide application of encryption technology in network traffic, it becomes more and more challenging to effectively monitor and analyze network traffic, and encrypted traffic analysis technology has also become an important research topic in the field of network security [1–3]. To analyze encrypted traffic, it is first necessary to divide the traffic into different sets according to specific goals, that is, to classify and identify network traffic. Most of the existing researches are implemented in a closed environment; that is, many researches can efficiently and accurately classify and identify traffic of known classes. For example, Aceto et al. [4, 5] proposed a practical mobile traffic classification model based on deep learning which can automatically extract features. Liu et al. [6] proposed an encrypted traffic classification model FS-Net based on the recurrent neural network. Wang et al. [7] proposed App-Net, a mobile application recognition model based on RNN and CNN. Nascita et al. [8] improved a multimodal deep learning traffic classification model based on explainable artificial intelligence techniques. But some studies, while valid in closed settings, often cannot really be applied to the real world. Because the real-world environment is open, network traffic includes encrypted and nonencrypted, known and unknown, benign and malicious, standard protocols and private protocols, etc. There are many practical problems to be considered and many difficulties to overcome. If a classification method trained in a closed set is simply generalized to an open set, it is easy to misclassify samples of unknown classes into known classes [9–11]. In order to solve this problem, researchers need to develop models that can support both the classification of known class samples and the discovery of unknown class samples, so as to actively manage and prevent abnormal traffic and further create and maintain a good network environment.

In real-world identification and classification tasks, it is often difficult to obtain labels for all samples during training, but it is desirable to have the ability to identify unknown classes during testing. Therefore, open-set recognition

describes such a scenario. During testing, samples of unknown classes that have not appeared in training will appear. The classifier can not only accurately classify known samples but also identify unknown classes. In a real network environment, there are known classes of encrypted traffic and unknown classes of encrypted traffic. This is similar to the scenario of open-set recognition. Therefore, some open-set recognition methods can be used to solve the unknown traffic identification. This paper proposes an unknown traffic identification method OpenCBD based on open-set recognition. It first performs a pretraining process based on self-supervised learning on unlabeled data, so that the CBD model (CBD model was proposed in [12]; it is a self-supervised learning model containing three modules; the three modules are the CNN module—based on a convolutional neural network, BERT module—based on a transformer model encoder, dense module—based on a fully connected network) has a certain understanding of the basic characteristics of encrypted traffic. Then, the training and testing process based on ensemble learning is designed in the open set. During training, the individual model is trained based on a specific loss function on some known classes. Then, all known classes are trained through the ensemble strategy, so that the ensemble model can achieve accurate classification of known classes and identification of unknown classes in the process of open-set testing. The contributions of this paper are summarized as follows:

(i) A novel unknown traffic identification model OpenCBD is designed. It uses the idea of open-set recognition, combines deep learning and ensemble learning, learns the basic characteristics of encrypted traffic from unlabeled data, and then trains on known classes of traffic to classify and identify traffic in an open environment

(ii) A general training method suitable for open-set recognition is proposed. The method only needs to be trained on the data of the known classes and can identify the data of the unknown classes. The method adopts two-stage training: first, it randomly selects part of the known classes data to train the individual model and then integrates the individual model to train with all known classes data, so that the model can learn the special knowledge between classes

(iii) A loss function that combines the cross-entropy loss function commonly used in classification models and the II-loss function proposed for open-set recognition is proposed. The combination of the two loss functions can train the model more efficiently, making the model fit faster and more accurately, so that the classification of known classes and the identification of unknown classes can be better completed at the same time

(iv) The OpenCBD model achieves good results in unknown traffic identification and known traffic classification tasks. Furthermore, the OpenCBD model outperforms significantly compared to the baseline methods

The rest of the paper is organized as follows. Section 2 summarizes the basic knowledge and related work of open-set recognition and unknown traffic identification. Section 3 details the structure and methodology of the overall model. Section 4 introduces the specific details of the experiments and evaluates and compares the experimental results. Finally, Section 5 concludes the paper.

2. Preliminary and Related Work

This section mainly introduces the basic knowledge and researches in recent years of open-set identification and unknown traffic identification.

2.1. Open-Set Recognition. First, some definitions in open-set recognition are given.

Definition 1 (open space [13]). Given a label set $\mathcal{K}$, the known class label is a positive integer, and the unknown class label is 0. For the feature $x \in \mathbb{R}^d$, define $f \in \mathcal{H}$ to be a measurable recognition function, $f_y(x) > 0$ means that the class y can be recognized, and $f_y(x) \le 0$ means that y is not recognized, where $\mathcal{H} : \mathbb{R}^d \longrightarrow \mathbb{R}$ is the appropriate smooth space for the recognition function. For any known class of training samples $x_i \in \mathcal{K}, i = 1, \cdots, N$, the open space $\mathcal{O}$ is defined as

$$\mathcal{O} = S_O - \bigcup_{i \in N} B_r(x_i), \tag{1}$$

where $B_r(x_i)$ is a closed sphere with the training sample x_i as the center and radius r. S_O is a sphere of radius r_o, including all known positive training samples $x \in \mathcal{K}$ and open space $\mathcal{O}$.

Definition 2 (open space risk [13]). Open space risk is defined as the relative measure of positive labeled open space compared to the overall measure of positive labeled space. Then, the probabilistic open space risk $R_{\mathcal{O}}(f)$ of class y is

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f_y(x)\mathrm{d}x}{\int_{S_O} f_y(x)\mathrm{d}x}. \tag{2}$$

Definition 3 (openness [14]). Openness refers to the degree of openness of an open space, which consists of training classes, target classes, and test classes,

$$O = 1 - \sqrt{\frac{2 \times |C_{\mathrm{TR}}|}{|C_{\mathrm{TA}}| + |C_{\mathrm{TE}}|}}, \tag{3}$$

where $C_{\mathrm{TR}}, C_{\mathrm{TA}}, C_{\mathrm{TE}}$ represent training classes, target classes, and testing classes, respectively. Sometimes $O < 0$ may occur, so the openness after calibration is only related

to training classes and test classes,

$$O^* = 1 - \sqrt{\frac{2 \times |C_{\mathrm{TR}}|}{|C_{\mathrm{TR}}| + |C_{\mathrm{TE}}|}}. \tag{4}$$

Definition 4 (open-set risk [13]). The open-set recognition problem is to minimize both traditional empirical risk and open space risk. Given an empirical risk function R_ε, the open-set risk is defined as

$$\arg \min_{f \in \mathscr{H}} \{R_{\mathscr{O}}(f) + \lambda_r R_\varepsilon(f)\}, \tag{5}$$

where λ_r is the regularization constant.

Definition 5 (open world recognition [13]). The solution for open world recognition is represented by the quintuple $[F, \varphi, \nu, L, I]$. $F(x)$: $\mathscr{R}^d \longrightarrow \mathscr{N}$ is a multiclass open-set recognition function. The i-th class in $F(x)$ is identified by the vector function $\varphi(x)$ of the measurable recognition function $f_i(x)$, and the detector $\nu(\varphi)$: $\mathscr{R}^i$to$[0, 1]$ is determined, where $i \in \mathscr{K}_t$, $\nu(\varphi)$ determine whether the output vector of the recognition function is from an unknown class. $L(x)$: $\mathscr{R}^d \longrightarrow \mathscr{N}^+$ is the labeling process. $L(x)$ applies new unknown data U_t to time t, resulting in label data $D_t = \{(y_j, x_j)\}$, where $y_j = L(x_j) \forall x_j \in U_t$. Assuming that the label finds m new classes, the set of known classes becomes $\mathscr{K}_{t+1} = \mathscr{K}_t \cup \{i + 1, \cdots, i + m\}$. $I_t(\varphi; D_t)$: $\mathscr{H}^i \longrightarrow \mathscr{H}^{i+m}$ is the incremental learning function. $I_t(\varphi; D_t)$ extensibly learns and adds new measurable functions $f_{i+1}(x) \cdots f_{i+m}(x)$ to the measurable recognition function vector φ. Each measurable function minimizes the corresponding open space risk.

Suppose that each $f_k(x)$ outputs the probability of belonging to the k classes. And assume that $f_k(x)$ is normalized across each class. Let $\varphi = [f_1(x), \cdots, f_k(x)]$, then the multiclass open-set recognition function is

$$F(x) = \begin{cases} 0, & \text{if } \nu(\varphi(x)) = 0, \\ y^*, & \text{otherwise}, \end{cases} \tag{6}$$

where

$$y^* = \arg \max_{y \in \mathscr{K}, f_y(x) \in \varphi(x)} f_y(x). \tag{7}$$

At this point, an easy way to detect is to set an acceptable minimum threshold τ and minimize the open space risk, i.e.,

$$\nu(\varphi(x)) = f_{y^*}(x) > \tau. \tag{8}$$

Since open-set recognition was proposed, many research methods have been proposed, mainly including discriminative models and generative models. Among them, the discriminative models are mostly constructed by traditional machine learning methods and deep neural network methods, and the generative models are divided into methods based on instance generation and methods based on noninstance generation according to whether there is instance generation. There are also some studies using neural networks in these two methods. The following mainly introduces some methods based on deep neural networks, which include not only discriminative models but also generative models.

In 2016, Bendale and Boult [15] proposed OpenMax, the first open-set recognition method based on deep neural networks. OpenMax was used as a new model layer that estimated the probability that the input came from an unknown class and provided a bounded open space risk.

In 2017, Ge et al. [16] proposed a multiclass open-set recognition method Generative OpenMax (G-OpenMax). Unlike some existing studies, the unknown class was not inferred from the features of the known classes or the decision distance. It extended OpenMax by employing Generative Adversarial Networks (GANs) for new classes of image synthesis.

In 2018, Yoshihashi et al. [17] proposed an open-set recognition classification reconstruction learning method CROSR, which used latent representations for reconstruction and achieved robust unknown detection without reducing the classification accuracy of known classes.

In 2019, Oza and Patel [18] proposed an open-set recognition algorithm C2AE, which used a class-conditional autoencoder, as well as closed-set classification training and open-set recognition training. The encoder and decoder were trained in two stages, and the reconstruction error was modeled using the extreme value theory of statistical modeling to find a threshold that identified samples of known and unknown classes.

In 2020, Hassen and Chan [19] proposed a representation method based on a neural network and used this representation method to propose an open-set recognition mechanism. In this representation, instances from the same class were close to each other, while instances from different classes were further apart.

In 2020, Liu et al. [20] proposed an algorithm that uses the meta-learning technique PEELER to solve the problem of open-set recognition. It combines randomly selecting a new set of classes per episode, maximizes the loss of postentropy of these class instances, and then learns a new metric formula based on the Mahalanobis distance.

In 2021, Joseph et al. [21] proposed an open-world object detector ORE, which was based on contrastive clustering and energy-based unknown identification. Identifying and characterizing unknown instances helped reduce confusion in incremental object detection settings. In this setting, state-of-the-art performance could be achieved without additional methodologies.

In 2022, Geng and Chen [22] proposed a batch decision strategy that was aimed at extending existing open-set recognition methods for new class discovery while considering correlations between test instances. By modifying the Hierarchical Dirichlet Process (HDP), a collective decision-based open-set recognition framework CD-OSR was proposed. CD-OSR did not need to define decision thresholds and could realize open-set recognition and new class discovery at the same time.

Table 1 summarizes several open-set recognition methods mentioned in Section 2.1.

2.2. Unknown Traffic Identification. In the field of network traffic analysis, unknown traffic identification has always been an important research direction. Researchers usually use unsupervised learning or semisupervised learning to solve the tasks of identifying and detecting unknown traffic.

In 2011, Finamore et al. [23] proposed an unsupervised algorithm to identify traffic classes within aggregates. This algorithm utilized the K-means clustering algorithm and added a mechanism to automatically determine the number of traffic clusters.

In 2013, Zhang et al. [24] proposed an approach to address the problem of unknown applications in the critical case of small supervised training sets. The proposed method had a superior ability to detect unknown traffic generated by unknown applications and exploited the correlation information between real-world network traffic to improve the classification performance.

In 2013, Zhang et al. [25] proposed an iterative method to extract unknown information from a set of unlabeled traffic flows. The method combined asymmetric bagging and flow correlation to guarantee the purity of the extracted negatives and demonstrated significantly better than state-of-the-art flow classification methods under unknown applications.

In 2014, Yu et al. [26] proposed a method to classify elephant traffic using service-based statistical features for cluster analysis. Elephant traffic refers to unknown traffic generated by only a few or some types of applications.

In 2015, Shaikh and Harkut [27] proposed a framework that classifies unknown flows in the network, solving the problem of applying unknowns in critical situations with little supervised training data. Flow label propagation was proposed, which automatically and accurately labeled more unlabeled flows to enhance the ability of Nearest Clustering-based Classifiers (NCCs). Composite classification was also proposed, which combines many flow predictions to more accurately classified Bag of Flows (BoF).

In 2015, Lin et al. [28] proposed a semisupervised learning method to address the problem of an unknown protocol in critical cases where the labeled training sample set was small. With the help of flow-related information and semisupervised clustering ensemble learning, the method had a superior ability to detect unknown samples generated by unknown protocols to improve classification performance.

In 2017, Ma and Qin [29] proposed a method using deep learning techniques to identify unknown protocols in complex network environments. The method identified the protocol in the network flow according to the application layer protocol type and found out the unknown protocol. This method only used the payload information in the captured 200,000 traffic flows and achieved well unknown protocol traffic identification accuracy.

In 2018, Fu et al. [30] proposed a scheme, FlowCop, to implement traffic detection that did not belong to any predefined application in network traffic classification. It divided the test traffic into N classes and one unknown class by building multiple one-class classifiers. A feature subspace algorithm was also proposed to select salient features for each class of classifiers.

In 2019, Sabeel et al. [31] proposed two methods to predict unknown DoS and DDoS attacks based on DNN and LSTM. The method demonstrated how well deep learning-based methods perform in unknown situations and to what extent deviations from the trained model could be handled. This method can effectively identify unknown attacks.

In 2019, Zhang et al. [32] proposed a network intrusion detection method based on open-set recognition. This method fits the recognition results of known classes to Weibull distribution and then builds an Open-CNN model to estimate the probability of unknown classes from the activation scores of known classes, so as to achieve the purpose of detecting unknown attacks.

In 2020, Zhang et al. [33] proposed an autonomous learning framework to correctly classify unknown classes. The framework efficiently updated deep learning-based traffic classification models during active operation. The core of the proposed framework consisted of a deep learning-based classifier, a self-learning discriminator, and an autonomous self-labeling model. The discriminator and self-labeling process generated new datasets during active operation to support classifier updates.

In 2020, Mohamed et al. [34] proposed a method for handling unknown applications. This method enabled efficient network classification with limited supervised training sets. The proposed model applied multiple neural network algorithms to predict unknown applications. The method improved Internet performance, reduced Internet traffic, and reduced delays in transmitting data.

In 2021, Wang et al. [35] proposed an unknown protocol parsing method based on a convolutional neural network. The protocol data was preprocessed into an image, and the converted image was inputted to the convolution layer for convolution. After convolution, the data was flattened, and the flattened data was put into a fully connected neural network to analyze and predict unknown protocols.

In 2021, Li et al. [36] proposed a lightweight unknown traffic discovery model, LightSEEN, which realized traffic classification and model update in the open world under practical conditions. The overall structure of the method was based on the Siamese network, and each side used a multihead attention mechanism, a one-dimensional convolutional neural network, and a residual network to facilitate the extraction of deep flow features and the convergence speed of the network.

In 2021, Xu et al. [37] proposed the KCC (Known Central Clustering) method to deal with the open-set-based intrusion detection problem. By introducing CD-loss (Class Distance-loss), the centers of different clusters were obtained. By introducing negative samples as unknown classes for training, the threshold of known classes was obtained. Unknown intrusions were rejected by comparing with fuzzy distances.

Table 2 makes a simple classification and summary according to the specific methods used in the above-mentioned several unknown traffic identification literatures.

Table 1: Different models for open-set recognition.

Model	Reference	Open-set recognition category	Methodology	Extreme value theory	Advantages
OpenMax	[15]	Discriminative model	EVT-based calibration classification	√	First deep open-set classifier without using background samples
G-OpenMax	[16]	Generative model	Unknown generation classification	√	Combining generative adversarial networks and OpenMax
CROSR	[17]	Discriminative model	Distance	√	First neural network architecture which involved hierarchical reconstruction blocks
C2AE	[18]	Discriminative model	Reconstruction error	√	Algorithms using class conditional autoencoders
Neural-network-based representation	[19]	Discriminative model	EVT-free calibration classification	×	A loss function was proposed such that instances from the same class are close to each other, while instances from different classes are farther apart
PEELER	[20]	Discriminative model	Distance	×	Combining few-shot classification and open-set recognition
ORE	[21]	Discriminative model	Distance	×	An incremental object detector is proposed
CD-OSR	[22]	Discriminative model	EVT-free calibration classification	×	Automatically reserve space for unknown classes under test, naturally bringing new class discovery capabilities

Table 2: Summary of methods for identifying unknown traffic.

Statistical methods			Deep neural networks		
Supervised learning	Semisupervised learning	Unsupervised learning	CNN-based	Open-set recognition	Others
[25]	[28]	[23, 24, 26, 27, 30]	[29, 32, 33, 35, 37]	[32, 35, 37]	[31, 34]

We combine unknown traffic identification with open-set recognition. According to the characteristics of network encrypted traffic, we design a new open-set recognition method based on a discriminant model. Through self-supervised learning and supervised learning, it can effectively complete known traffic classification and unknown traffic identification tasks at the same time.

3. Proposed Method

To identify unknown classes that have not appeared in the training set, we propose OpenCBD, a deep learning and ensemble learning-based approach. The CBD model was proposed in [12], but it can only complete the classification task in datasets with known class traffic. In this paper, the CBD model is regarded as an individual model. First, the loss function is used to train the individual model, and then through the ensemble strategy, the individual model is fused into an ensemble model, which is extended to OpenCBD suitable for open-set data. Classify known classes and identify unknown classes in the world. The CBD model includes the CNN module, BERT module, and dense module. The detailed structure is given in Section 3.3. The overall process of OpenCBD is shown in Figure 1.

First, pretraining is performed on unlabeled data to obtain a pretrained individual model, including a CNN module with a fixed structure and parameters and a BERT module with a fixed structure and no fixed parameters. Then, perform data preprocessing on the raw data participating in the training, then input them into multiple individual models for training after obtaining the training set, then input the training set into the ensemble model for training, and then obtain the training results. Finally, data preprocessing is performed on the raw data participating in the test, and after the test set is obtained, it is inputted into the trained ensemble model, and the prediction results are the outputs.

The training process includes two stages, namely, closed-set individual training and closed-set ensemble training. The testing process is implemented in the open set. The block diagram of the three stages is shown in Figure 2.

In the following, we will introduce the data preprocessing process, pretraining process, and detailed process of the three stages.

3.1. Data Preprocessing. Data preprocessing is an essential step to achieve the goals of classification and identification. Data preprocessing mainly includes four parts: traffic split, traffic cleaning, traffic conversion, and time interval

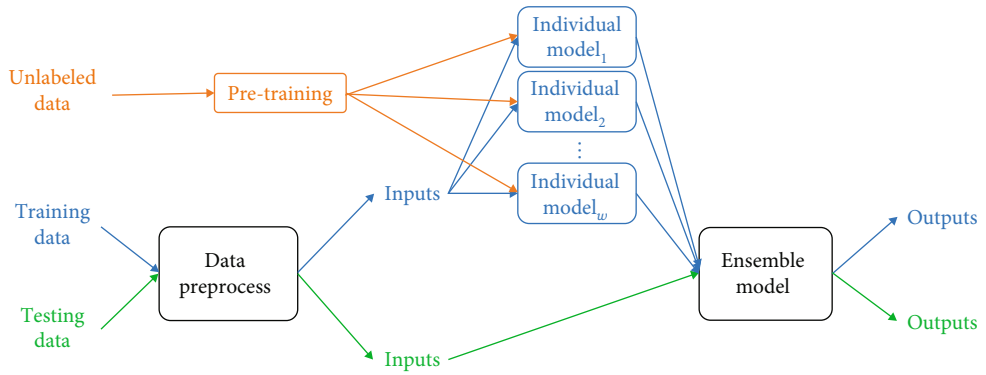

Figure 1: Overall flow chart of proposed method. The orange part represents the pretraining process, the blue part represents the training process, the green part represents the testing process, and the black part represents the process involved in both training and testing processes.

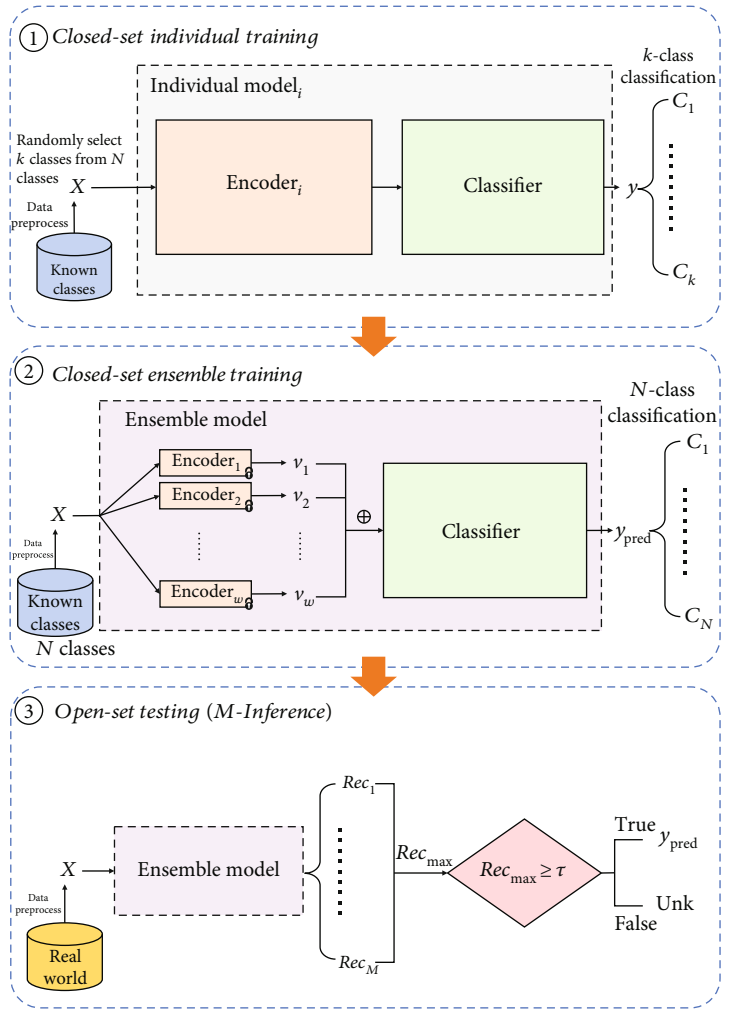

Figure 2: Block diagram of proposed method. (1) Closed-set individual training: randomly select a part of the known class data to train the individual model including the encoder and the classifier. 2) Closed-set ensemble training: using all known class data to train encoders and classifiers in w individual models, where the weights of the encoders are locked. (3) Open-set testing: the ensemble model produces M results to be identified. If the largest to-be-identified result is greater than or equal to the threshold, the test sample is classified into one of the M classes; otherwise, it is classified as unknown.

integration. Following the description in [12], we summarize the preprocessing process as Algorithm 1.

Line 2 is the traffic split. For bidirectional flow, randomly intercept 10 consecutive data packets and define these ten packets as a flow. A total of n flows are intercepted, that is, a total of $10n$ data packets.

Lines 4-9 are the traffic cleaning. Read the payload part of each packet, then unify the length, truncate the first 256 bytes of each packet, and add 0 to the insufficient, to obtain the raw sequence x,

$$x = (b_1, b_2, \cdots, b_{8\times 256}), b_i \in \mathbb{F}_2, \quad i = 1, \cdots, 2048. \tag{9}$$

Line 10 is the traffic conversion. Convert the elements in the raw sequence into decimals according to bytes to obtain a 256-dimensional vector x,

$$x = (x_1, x_2, \cdots, x_{256}), x_i \in \mathbb{F}_{2^8}, \quad i = 1, \cdots, 256. \tag{10}$$

Lines 11-16 are the time interval integration, which was first proposed in [38]. It means that according to the statistical results of the time interval of two adjacent data packets in different classes, for the interval of more than 1 s, insert a blank data packet and ignore it within 1 s. The blank data packet is represented as a 256-dimensional all-one vector $x_0 = (1, \cdots, 1, \cdots, 1)_{256}$.

The payload vector x and the time interval vector x_0 are formed into a set $\mathcal{X}$ in chronological order, and the model can be directly inputted in the next step.

3.2. Pretraining. Pretraining adopts a common pretraining method in the field of encrypted traffic analysis proposed in [12], which starts from the packet level and the flow level, and can directly deepen the model's understanding of encrypted traffic from unlabeled real-world data. This paper summarizes the method into a detailed Algorithm 2.

Lines 1-15 are the packet-based methods. For an unlabeled packet, first extract the payload part to get $x = \{x_1, x_1, \cdots, x_n\}$, and then calculate the entropy of each packet,

$$\begin{aligned} H &= -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i), \quad 1 \le i \le n, \\ &\begin{cases} H(x) < H_0, & x \in \mathcal{X}_{\text{plain}}, \\ H(x) \ge H_0, & x \in \mathcal{X}_{cipher}. \end{cases} \end{aligned} \tag{11}$$

Set the threshold of entropy $H_0 = 4$. When $H \ge H_0$, the data packet label is a ciphertext data packet; when $H < H_0$, the data packet label is a plaintext data packet. Train a CBD model with labeled packets.

Lines 16-31 are the flow-based methods. First, construct positive and negative sample sets. The positive sample set S^+ contains n positive samples, and the positive sample s^+ is defined as a continuous flow F; each flow contains 10 consecutive packets,

$$\begin{aligned} S^+ &= \{s_1^+, s_2^+, \cdots, s_n^+\}, \\ s_i^+ \triangleq F^i &= \{x_1^i, x_2^i, \cdots, x_{10}^i\}, \quad 1 \le i \le n. \end{aligned} \tag{12}$$

The number of negative sample set S^- is the same as the positive sample set S^+, including n negative samples, and the negative sample s^- is defined as a discontinuous flow $\bar{F}$. It is obtained by transforming the positive samples. Each packet in the positive sample is replaced with other bags with a certain probability, and the replaced sample is called a negative sample.

$$\begin{aligned} S^- &= \{s_1^-, s_2^-, \cdots, s_n^-\}, \\ s_i^- \triangleq \bar{F}^i &= \{f(x_1^i), f(x_2^i), \cdots, f(x_{10}^i)\}, \quad 1 \le i \le n, \\ f(x_j^i) &= \begin{cases} x_j^i & (P = 0.7) \\ x_{j'}^{i'} & (P = 0.3) \end{cases}, (i', j') \ne (i, j), \quad 1 \le j \le 10. \end{aligned} \tag{13}$$

Train a CBD model with a labeled set of positive and negative samples. After completing the pretraining, enter the three-stage training and testing process.

3.3. Closed-Set Individual Training. The first stage is closed-set individual training. After the data is preprocessed, the k class data is randomly selected to train the individual model, and the k classification is completed. Each individual model is a randomly selected k class with replacement. The specific structure diagram of the individual model is shown in Figure 3.

3.3.1. Encoder. The encoder and classifier are two important parts in the individual model. The encoder is mainly used for feature extraction, the input is a matrix X, and the output is a vector $\mathbf{v}$,

$$\text{Encoder} : X \longrightarrow \mathbf{v} \in \mathcal{R}^{240}, \tag{14}$$

where X is a $n \times 256$-dimensional matrix on $\mathbb{F}_{2^8}$. The encoder mainly includes the CNN module and BERT module.

The CNN module is inputted in sequence by a row vector; that is, for the input $X = (x_1^T, \cdots, x_n^T)$ of the encoder, x_i represents the i-th input of the CNN module, $i = 1, 2, \cdots, n$. Define $f_C(\cdot)$ as the CNN module function, $f_{\text{con}_j}(\cdot)$ is the convolution function of the j-th layer in the CNN module, $f_{\text{mp}_j}(\cdot)$ is the maximum pooling function of the j-th layer in CNN module, and O_i^j is the j-th layer output for the i-th input in the CNN module, $j = 1, 2, 3, 4$; then, the output

Input: the original network traffic dataset D, the number of classes of traffic c.
Output: packet vector set $\mathscr{X}$;
For $i = 1, \cdots, c$ **do**
  Randomly select n_i consecutive 10 packets to form n_i flows;
  For $j = 1, \cdots, 10n_i$ **do**
  Read the payload part of the packet $x_{i,j}$;
  If $|x_{i,j}| < 2048$ **then**
    $x_{i,j} = [x_{i,j}, \text{zeros}(1, 2048 - |x_{i,j}|)]$;
  Else
    $x_{i,j} = x_{i,j}[1 : 2048]$;
  End if
  $x_{i,j} = \text{int}(x_{i,j}', 10)$;
  $\text{cout}_{i,j} = \text{time}(x_{i,j+1}) - \text{time}(x_{i,j})$;
  If $\text{cout}_{i,j} < 1$ second **then**
    Continue;
  Else
    Add x_0 between $x_{i,j}$ and $x_{i,j+1}$;
  End if
 End for
 Generate packet vector set $X_i = \{x_{i,1}, \cdots, x_0, \cdots, x_{i,10n_i}\}$;
End for
Return packet vector set $\mathscr{X} = X_1 \bigcup X_2 \bigcup \cdots \bigcup X_c$.

Algorithm 1: Preprocessing algorithm for traffic data.

of the CNN module is

$$\begin{aligned} f_C(\mathbf{x}_i) = O_i^4 &\in \mathscr{R}^{242}, \\ O_i^4 &= \max\left(0, f_{\text{mp}_4}\left(f_{\text{con}_4}\left(O_i^3\right)\right)\right), \\ O_i^3 &= \max\left(0, f_{\text{mp}_3}\left(f_{\text{con}_3}\left(O_i^2\right)\right)\right), \\ O_i^2 &= \max\left(0, f_{\text{mp}_2}\left(f_{\text{con}_2}\left(O_i^1\right)\right)\right), \\ O_i^1 &= \max\left(0, f_{\text{con}_1}(x_i)\right). \end{aligned} \tag{15}$$

After the CNN module is a fully connected layer, which is used to change the dimension of the vector to facilitate the input of the subsequent BERT module. Define $f_{\text{fc}}(\cdot)$ to be the fully connected function and $z_i \in \mathscr{R}^{240}$ to be the output of the fully connected layer, then

$$\boldsymbol{z}_i \triangleq f_{\text{fc}}\left(O_i^4\right) = W_1 O_i^4 + B_i = W_i f_C(\boldsymbol{x}_i) + B_i, \tag{16}$$

where W_i is the weight matrix and B_i is the bias matrix.

This is followed by a Concat layer that stitches together the outputs after n CNN modules and fully connected layers. Define $\oplus$ as the splicing symbol and Z as the output of the Concat layer, then

$$Z = \left(z_1^T, z_2^T, \cdots, z_n^T\right) = z_1 \oplus z_2 \oplus \cdots \oplus z_n = \mathop{\oplus}_{i=1}^{n} z_i, Z \in \mathbb{R}^{n \times 240}. \tag{17}$$

The BERT module consists of m transformer encoders, and Z is the input of the BERT module. Define $f_B(\cdot)$ as the BERT module function, $f_{\text{te}_t}(\cdot)$ as the t-th layer transformer encoder function in the BERT module, and v as the output of the encoder, then

$$v \triangleq f_B(Z) = f_{\text{te}_m}\left(\cdots f_{\text{te}_t}\left(\cdots f_{\text{te}_1}(Z)\right)\right), \quad t = 1, 2, \cdots, m. \tag{18}$$

3.3.2. Classifier. The classifier is mainly used to predict results, the input is a vector v, and the output is a specific class y,

$$\text{Classifier} : v \longrightarrow y \in \{1, 2, \cdots, k\}. \tag{19}$$

The classifier consists of a dense module and Softmax layer. In the dense module, define $f_D(\cdot)$ to be the function of the dense module and $F(X)$ to be the output of the dense module, then

$$F(X) \triangleq f_D(v) = \max(0, (Wv + B)), \tag{20}$$

where W represents the weight matrix and B represents the bias matrix.

Input: unlabeled dataset $D_1 = \{\mathbf{x}_{1,1},\cdots,\mathbf{x}_{1,N}\}$, D_2.
Output: pretrained model;
For $i = 1, \cdots, N$ **do**
  Extract the payload of $\mathbf{x}_{1,i}$, $\mathbf{x}_{1,i} = \{x_{1,i}^1, x_{1,i}^2,\cdots,x_{1,i}^n\}$;
  For $j = 1, \cdots, n$ **do**
    $H(\mathbf{x}_{1,i}) = 0$;
    $H(x_{1,i}^j) = -P(x_{1,i}^j) \cdot \log_2 P(x_{1,i}^j)$;
    $H(\mathbf{x}_{1,i}) = H(\mathbf{x}_{1,i}) + H(x_{1,i}^j)$;
  End for
  If $H(\mathbf{x}_{1,i}) < H_0$ **then**
    $\text{lab}(\mathbf{x}_{1,i}) = 0$ (plaintext);
  Else
    $\text{lab}(\mathbf{x}_{1,i}) = 1$ (ciphertext);
  End if
End for
$D_1{}' = \{(\mathbf{x}_{1,1}, \text{lab}(\mathbf{x}_{1,1})),\cdots,(\mathbf{x}_{1,N}, \text{lab}(\mathbf{x}_{1,N}))\}$;
Model = Model($D_1{}'$);
Randomly select m consecutive 10-packet payload parts in D_2 to form S^+, $S^+ = \{s_1^+,\cdots,s_m^+\}$;
For $a = 1, \cdots, m$ **do**
  $s_a^+ \triangleq F_a = \{x_{a,1}, x_{a,2},\cdots,x_{a,10}\}$;
  $S^- = \varnothing$;
  For $b = 1, \cdots, 10$ **do**
    If $P = 0.3$ **then**
      $f(x_{a,b}) = x_{a',b'}$, where $(a', b') \neq (a, b)$;
    Else
      $f(x_{a,b}) = x_{a,b}$;
    End if
    $s_a^- \triangleq F_a{}' = \{f(x_{a,1}), f(x_{a,2}),\cdots,f(x_{a,10})\}$;
    $S^- = S^- \bigcup s_a^-$;
  End for
End for
$D_2{}' = S^+ \bigcup S^-$;
Model = Model($D_2{}'$);
Return pretrained model.

Algorithm 2: Pretraining algorithm for network traffic model.

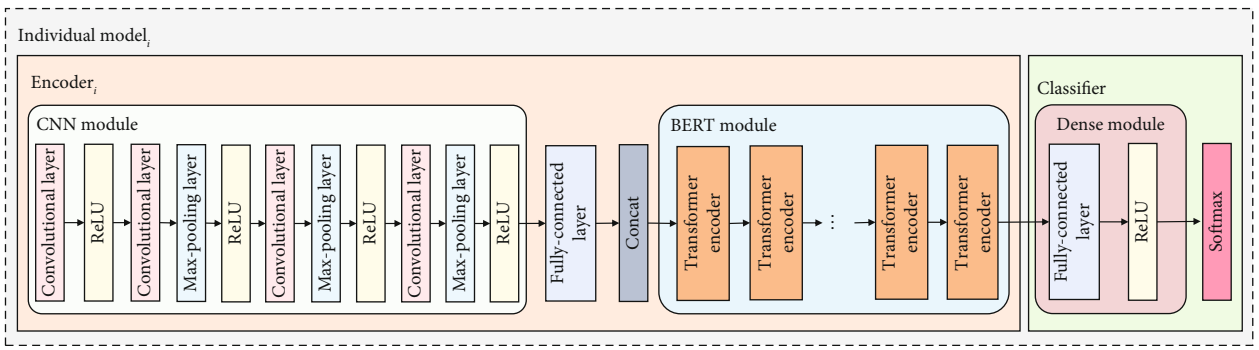

Figure 3: Structure diagram of individual model.

The last is the Softmax layer, defining $f_{\text{sm}}(\cdot)$ as the Softmax function, then

$$y = f_{\text{sm}}(f_D(v)) = \arg\max_l P(y = l|X),$$
$$P(y = l|X) = \frac{e^{F_l(X)}}{\sum_{p=1}^{k} e^{F_p(X)}}, \tag{21}$$

where $l, p \in \{1, 2, \cdots, k\}$ is the index of the class. So the output of the entire individual model is

$$y = f_{\text{sm}}(f_D(v)) = f_{\text{sm}}(f_D(f_B(Z))) = f_{\text{sm}}\left(f_D\left(f_B\left(\mathop{\oplus}_{i=1}^{n} z_i\right)\right)\right)$$
$$= f_{\text{sm}}\left(f_D\left(f_B\left(\mathop{\oplus}_{i=1}^{n} (W_i f_C(x_i) + B_i)\right)\right)\right). \tag{22}$$

3.3.3. Loss Function. Individual models are trained with a combination of cross-entropy loss and II-Loss [19].

The cross-entropy loss function is often used in classification problems, which can measure the similarity between several classes. Given a batch of samples $\{X_1, X_2, \cdots, X_A\}$, for a matrix $X_a, a = 1, 2, \cdots, A$, define its label as $\text{lab}(X_a) \in \{1, 2, \cdots, k\}$, then the cross-entropy loss of this batch of samples is

$$\text{CE-Loss} = -\frac{1}{A}\sum_{a=1}^{A}\sum_{l=1}^{k} \text{sgn}\,(X_a, l) \log P(X_a, l),$$
$$\text{sgn}\,(X_a, l) = \begin{cases} 1, & \text{lab}(X_a) = l, \\ 0, & \text{lab}(X_a) \neq l, \end{cases} \tag{23}$$

$$P(X_a, l) = P(\text{lab}(X_a) = l), \tag{24}$$

where $\text{sgn}\,(X_a, l)$ is a symbolic function, which determines whether the label of X_a is the class l. $P(X_a, l)$ is a probability function that calculates the probability that the label of X_a is the class l.

The II-Loss function can make samples of different classes farther apart and samples of the same class closer by maximizing the distance between different classes and minimizing the distance between samples and their class mean. Given a batch of samples $\{X_1, X_2, \cdots, X_A\}$, define the sample set of class l as C_l, the number of samples in C_l is $|C_l|$, the mean output of the dense module in C_l is μ_l, then

$$\{X_1, X_2, \cdots, X_A\} = C_1 \cup C_2 \cup \cdots \cup C_k,$$
$$\mu_l = \frac{1}{|C_l|}\sum_{q=1}^{|C_l|} F(X_q), \quad q = 1, 2, \cdots, |C_l|. \tag{25}$$

The II-Loss of this batch of samples is

$$\text{II-Loss} = \text{Dis(intra\_class)} - \text{Dis(inter\_class)}$$
$$= \left(\frac{1}{A}\sum_{l=1}^{k}\sum_{i=1}^{|C_l|} \|\mu_l - F(X_l)\|_2^2\right) - \left(\min_{1\le p,r\le k \& p=r} \left\|\mu_p - \mu_r\right\|_2^2\right). \tag{26}$$

The final loss function is

$$\text{Loss} = \text{CE-Loss} + \text{II-Loss} = \left(-\frac{1}{A}\sum_{a=1}^{A}\sum_{l=1}^{k} \text{sgn}\,(X_a, l) \log P(X_a, l)\right)$$
$$+ \left(\frac{1}{A}\sum_{l=1}^{k}\sum_{i=1}^{|C_l|} \|\mu_l - F(X_l)\|_2^2\right) - \left(\min_{1\le p,r\le k \& p=r} \left\|\mu_p - \mu_r\right\|_2^2\right). \tag{27}$$

3.4. Closed-Set Ensemble Training. The second stage is closed-set ensemble training. After the data is preprocessed, the ensemble model is trained with all N class known data to complete N-classification. The integrated strategy of the ensemble model adopts the learning method, which is a method of combining individual learners by training the learners. First, the encoder in the individual model is trained using a subset of the raw training dataset. Then, the ensemble model is trained with the raw training set using the output of the encoder as a feature. See Algorithm 3 for the ensemble strategy.

The training of the ensemble model takes t-fold cross-validation as an example and divides the known class dataset D and subset D' into t datasets $D_1, \cdots, D_t$ and $D'_1, \cdots, D'_t$, respectively. Let D'_j and $D'_{(-j)} = D' \setminus D'_j$ be the test set and training set corresponding to the jth individual model execution, respectively. D_j and $D_{(-j)} = D\,D_j$ are the test set and training set corresponding to the jth ensemble model execution, respectively. Given w individual learning algorithms, use the sth learning algorithm to train on $D'_{(-j)}$ to obtain an individual learner $h_s^{(-j)}$. For each sample X_i in the test set D'_j executed at the jth time, let z_{is} be the learner $h_s^{(-j)}$ on the X_i output result. Then, at the end of the entire cross-validation process, a new dataset can be generated by w individual learners $D* = \{(z_{i1}, \cdots, z_{iw}, y_i)\}_{i=1}^{R}$; the ensemble learner will use this dataset to train together with the dataset $D \setminus D'$ that does not participate in the training of individual models.

Lines 1-4 are the training of the individual model, which only uses a subset D' of the known class data D during training. Lines 5-12 are the training of the ensemble model, which uses all of the known class data D during training. The ensemble model consists of w encoders and 1 classifier. The encoder is obtained from the first stage of training, and the fixed structure and parameters remain unchanged. The classifier has the same structure as the first stage, but the specific parameters have changed. Define X as the input of the ensemble model, y_{pred} as the output, $f_D^*(\cdot)$ as the dense module function, $G(X)$ as the dense module output, and $f_{sm}^*(\cdot)$ as

Input: known classes data $D = \{(X_1, L_1), (X_2, L_2), \cdots, (X_R, L_R)\}$; individual learning algorithm $\mathscr{T}_1, \mathscr{T}_2, \cdots, \mathscr{T}_w$; ensemble learning algorithm $\mathscr{E}$.
Output: ensemble model H;
Randomly selected k classes data from D to form D', $D' \subset D$;
For $s = 1, 2, \cdots, w$ **do**
  $h_s = \mathscr{T}_t(D')$;
End for
$D^* = \varnothing$;
For $i = 1, 2, \cdots, R$ **do**
  For $s = 1, 2, \cdots, w$ **do**
    $z_{rs} = h_s(X_r)$;
  End for
  $D^* = (D \setminus D') \cup ((z_{r1}, z_{r2}, \cdots, z_{rw}), y_r)$;
End for
$h^* = \mathscr{E}(D^*)$;
$H(X) = h^*(h_1(X), h_2(X), \cdots, h_w(X))$.

Algorithm 3: Ensemble strategy.

Input: training set $\{X_1, X_2, \cdots, X_A\}$; known classes N.
Output: threshold τ;
For $i = 1, 2, \cdots, A$ **do**
  For $j = 1, 2, \cdots, N$ **do**
    $P(y = j|X_i) = e^{F_j(X_i)} / \sum_{p=1}^{k} e^{F_p(X_i)}$;
  End for
  outlier$(X_i) = \max_j P(y = j|X_i)$;
End for
sort(outlier);
Threshold set = last 1% sort(outlier);
Return τ = min (threshold set).

Algorithm 4: Threshold estimation.

Table 3: Experimental data classes.

Known/unknown	Classes
Known classes	VPN-Facebook-chat
	VPN-Facebook-audio
	VPN-Skype-chat
	VPN-Skype-file
	Facebook-chat
	Facebook-audio
	Skype-chat
	Skype-file
	VPN-Hangouts-audio
Unknown classes	Hangouts-audio
	VPN-Hangouts-chat
	Skype-audio
	VPN-sftp

Table 4: Experimental environment.

Hardware	Specific information
Graphic processing unit (GPU)	Nvidia RTX 1060TI
Memory	48 GB
Central processing unit (CPU)	Intel Core i7 6-core
Number of CPUs	Number of CPU cores: 6 Number of CPU threads: 12
Operating system (OS)	Ubuntu 18.04

the Softmax function, then

$$\text{Ensemble model}: X \longrightarrow y_{\text{pred}} \in \{1, 2, \cdots, N\},$$

$$G(X) \triangleq f_D^*\left(\sum_{s=1}^{w} v_s\right),$$

$$y_{\text{pred}} = f_{\text{sm}}^*(G(X)) = f_{\text{sm}}^*\left(f_D^*\left(\sum_{s=1}^{w} v_s\right)\right) = \arg \max_{l'} P\left(y = l'|X\right),$$

$$P\left(y = l'|X\right) = \frac{e^{G_{l'}(X)}}{\sum_{p'=1}^{N} e^{G_{p'}(X)}}, \tag{28}$$

where w is the number of encoders in the ensemble model, each encoder comes from the corresponding individual model, and the coefficients remain unchanged. $f_D^*(\cdot)$ and $f_{\text{sm}}^*(\cdot)$ have the same structure as the dense module function $f_D(\cdot)$ and Softmax function $f_{\text{sm}}(\cdot)$ in the individual model but with different coefficients. $l', p' \in \{1, 2, \cdots, N\}$ is the index of the class.

3.5. Open-Set Testing. The third stage is open-set testing. The real-world data is inputted into the trained ensemble model, and the ensemble model produces M results Rec to be identified. Compute the maximum of the M results. If the maximum value is greater than or equal to the threshold, the

Table 5: Experimental hyperparameters.

Hyperparameter name	Hyperparameter value
Epoch	80 in individual model, 60 in ensemble model
Batch	64
Plaintext packets in pretraining	256 bytes of plaintext
Ciphertext packets in pretraining	Randomly selected from ISCXVPN2016, excluding the 13 classes mentioned above, and classes and sizes are not fixed
Sample numbers of positive and negative sample sets in pretraining	5000, 5000
k in individual model	3 or 4 or 5 or 6 or 7
Dimensions of CNN module input vectors	256
Number of convolution channels in CNN module	1
Convolution kernel size in CNN module	3
Convolution stride size in CNN module	1
Pooling size in CNN module	3
Dimensions of CNN module output vectors	242
Dimensions of fully connected layer outputs in the encoder	240
n in Concat layer	15
Dimensions of BERT module input matrix	15×240
m in BERT module	8
Head numbers of multihead attention in each transformer encoder layer	12
Dimensions of BERT module output vectors	240
Dimensions of fully connected layer outputs in classifier	k in individual model, N in ensemble model
Number of encoders w in ensemble model	4 or 8 or 12
N in ensemble model	8
M in open-set testing	$N + 1 = 9$

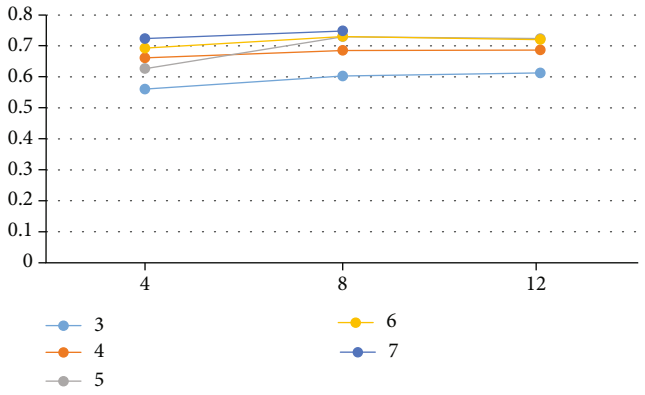

Figure 4: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy of the 2-class classification tasks (identifying known and unknown classes) varies as the number of known classes is randomly selected in the first training stage.

test sample is classified as one of M classes; otherwise, it is classified as unknown.

3.5.1. Threshold Estimation. The threshold for judging data classes in open-set testing is determined by outliers. Assume that 1% of the samples in the training set have noise, that is, they are abnormal outlier samples. When calculating the outlier distance for all training samples, sort them from small to large, and the largest 1% distance is the threshold. The detailed process of threshold estimation is shown in Algorithm 4.

After the threshold is determined, the to-be-identified results of the ensemble model will determine the classes of data according to whether it is greater than or equal to the threshold.

4. Experiment and Evaluation

This section mainly introduces the specific experimental content designed for the proposed model OpenCBD, including experimental environment and settings, evaluation metrics, and specific results. And then, the results are discussed to verify the validity of the OpenCBD model.

4.1. Experiment Settings. The encrypted traffic data in this paper is selected from the public dataset ISCXVPN2016 [39]. 8 classes of data are selected as known classes, 5 classes of data are selected as unknown classes, and the intersection of known and unknown data is empty. The 13 classes of data

Table 6: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy, precision, recall, and F1-score of the 2-class classification tasks (identifying known and unknown classes) vary as the number of known classes is randomly selected in the first training stage.

k	Accuracy			Precision			Recall			F1-score		
	w											
	4	8	12	4	8	12	4	8	12	4	8	12
3	0.5607	0.603	0.613	0.714	0.6853	0.6951	0.4775	0.6562	0.6612	0.5722	0.6704	0.6777
4	0.6615	0.6853	0.6869	0.7337	0.7459	0.7484	0.7062	0.7412	0.74	0.7197	0.7435	0.7441
5	0.6269	0.73	0.7238	0.7215	0.765	0.778	0.6412	0.81	0.7712	0.67902	0.7868	0.7746
6	0.693	0.7307	0.7207	0.775	0.7952	0.7841	0.7062	0.7575	0.7537	0.739	0.7759	0.7686
7	0.7238	0.7484	—	0.7766	0.7832	—	0.7737	0.8175	—	0.7752	0.7999	—

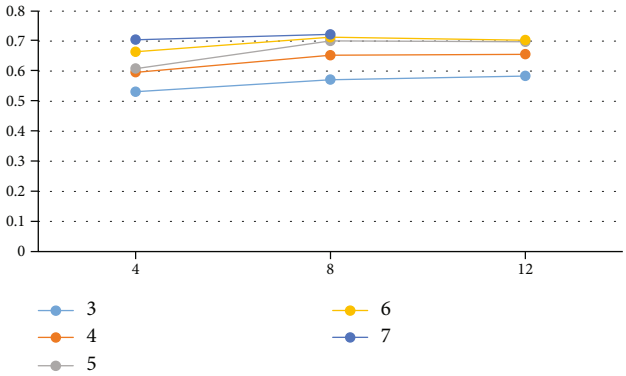

Figure 5: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy of the 9-class classification tasks (identifying 8 known classes and 1 unknown class) varies as the number of known classes is randomly selected in the first training stage.

include both Virtual Private Network (VPN) traffic that has been encapsulated by the VPN protocol and non-VPN traffic that has not been encapsulated by the VPN. 1000 samples are randomly selected for each class, and each sample contains 10 consecutive data packets. The specific data classes are shown in Table 3.

During the experiment, 1000 samples of each class were randomly divided into the training set and test set, with training set samples : test set samples = 9 : 1; that is, 100 samples of each class were randomly selected as the test set, and the remaining 900 samples were the training set. The training set contains training data and validation data, with training data : validation data = 9 : 1.

The experimental environment is a personal desktop, and the specific equipment information is shown in Table 4.

All codes in the experiment are run in the environment of Python 3.6.5, and the specific hyperparameter settings are shown in Table 5.

The rest of the unmentioned hyperparameters are set according to the default values of the corresponding models in Python.

4.2. Evaluation Metrics. When evaluating model performance, the class of interest is usually the positive class and the other classes are the negative classes. Evaluation metrics are usually formulated from the following four basic situations:

(1) *True positive (TP)*: predict the positive class as a positive class

(2) *False positive (FP)*: predict the negative class as a positive class

(3) *True negative (TN)*: predicts the negative class as a negative class

(4) *False negative (FN)*: predicts the positive class as a negative class

From the above four basic situations, four basic evaluation metrics can be obtained:

(1) The probability of classifying positive samples into positive classes is called TPR, also known as recall or sensitivity:

$$\mathrm{TPR} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{29}$$

(2) The probability of classifying a negative class sample into negative classes is called TNR, also known as specificity:

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}. \tag{30}$$

(3) The probability of misclassifying negative samples into positive classes is called FPR, also known as false recognition rate:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}. \tag{31}$$

Table 7: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy, precision, recall, and F1-score of the 9-class classification tasks (identifying 8 known classes and 1 unknown class) vary as the number of known classes is randomly selected in the first training stage.

k	Accuracy			Precision			Recall			F1-score		
	w											
	4	8	12	4	8	12	4	8	12	4	8	12
3	0.5315	0.5715	0.5838	0.6462	0.6717	0.6893	0.4593	0.5953	0.6051	0.5019	0.6144	0.6258
4	0.5961	0.653	0.656	0.6241	0.7137	0.723	0.5988	0.6784	0.6813	0.5994	0.6863	0.6929
5	0.6084	0.7007	0.6976	0.6991	0.7425	0.7355	0.6104	0.7446	0.7197	0.6343	0.7385	0.7227
6	0.6646	0.713	0.703	0.7419	0.7704	0.772	0.6613	0.7242	0.7186	0.6845	0.7395	0.7339
7	0.7046	0.7223	—	0.7639	0.7616	—	0.7315	0.7597	—	0.7404	0.7533	—

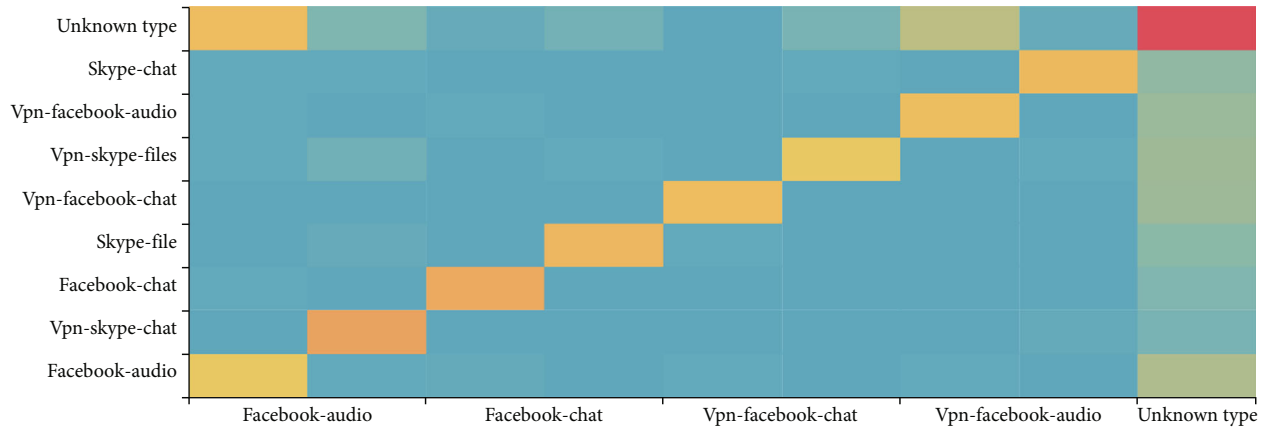

Figure 6: Confusion matrix diagram for OpenCBD doing the 9-class classification tasks. The x-axis represents the predicted classes of the output, and the y-axis represents the true classes. Warmer colors indicate larger values, and cooler colors indicate smaller values.

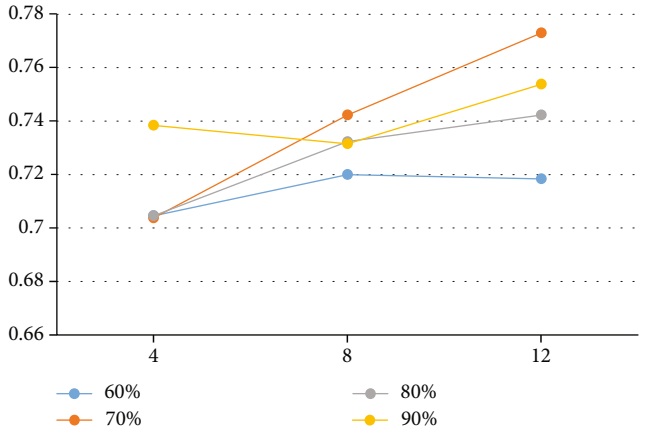

Figure 7: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy of the 2-class classification tasks (identifying known and unknown classes) varies with increasing percentage of known classes randomly selected.

(4) The probability of misclassifying positive samples into negative classes is called FNR, also known as the rejection rate:

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}. \tag{32}$$

In addition, there are several other commonly used metrics:

(1) Precision refers to the proportion of true positive samples among all predicted positive samples:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{33}$$

(2) F1-score considers precision and recall comprehensively and refers to the harmonic average of precision and recall:

$$\begin{aligned} \frac{2}{\text{F1-score}} &= \frac{1}{\mathrm{Precision}} + \frac{1}{\mathrm{Recall}}, \\ \text{F1-score} &= 2 \times \mathrm{Precision} \times \frac{\mathrm{Recall}}{\mathrm{Preciaion} + \mathrm{Recall}}. \end{aligned} \tag{34}$$

(3) Accuracy refers to the proportion of all predicted correct samples to the total samples:

Table 8: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy, precision, recall, and F1-score of the 2-class classification tasks (identifying known and unknown classes) vary with increasing percentage of known classes randomly selected.

k	Accuracy			Precision			Recall			F1-score		
	w											
	4	8	12	4	8	12	4	8	12	4	8	12
60%	0.7046	0.72	0.7184	0.753	0.7678	0.7726	0.7737	0.7812	0.7687	0.7632	0.7744	0.7706
70%	0.7038	0.7423	0.773	0.767	0.8015	0.8371	0.745	0.7725	0.7837	0.7558	0.7867	0.8095
80%	0.7046	0.7323	0.7423	0.7619	0.786	0.8047	0.7563	0.7762	0.7675	0.759	0.7811	0.7856
90%	0.7384	0.7315	0.7538	0.7875	0.7909	0.8269	0.7875	0.7662	0.7587	0.7875	0.7784	0.7913

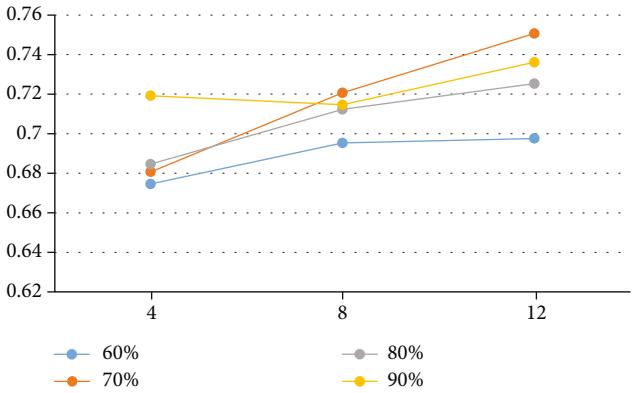

Figure 8: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy of the 9-class classification tasks (identifying 8 known classes and 1 unknown class) varies with increasing percentage of known classes randomly selected.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{n_{\text{correct}}}{n}, \tag{35}$$

where n_{correct} represents the number of correctly predicted samples and n represents the total number of samples.

(4) In the multiclassification problem, the F1-score, precision, and recall are all calculated by macro; that is, the index of each class is calculated first, and then, the unweighted average is taken to obtain the final index:

$$\begin{aligned} \text{macro-F1-score} &= \frac{1}{C}\sum_{i=1}^{C} \text{F1-score}_i, \\ \text{Macro-precision} &= \frac{1}{C}\sum_{i=1}^{C} \text{precision}_i, \\ \text{Macro-recall} &= \frac{1}{C}\sum_{i=1}^{C} \text{recall}_i, \end{aligned} \tag{36}$$

where C represents the number of classes.

The evaluation metrics of this experiment select accuracy, precision, recall, and F1-score of binary classification and multiclassification.

4.3. Results and Discussions. According to the detailed experimental setting in Section 4.1 and the four evaluation metrics in Section 4.2, we present the specific experimental results in this section. The first is the experimental results of OpenCBD when doing 2-class classification tasks.

As can be seen from Figure 4 and Table 6, OpenCBD performs best when 7 known classes are randomly selected in the first stage of training. Because there are 8 known classes, each individual model randomly selects 7 classes for training, which enables the individual model to better understand the differences between different known classes and makes the model perform better. On the whole, the increase in the number of encoders helps to improve the performance of the model, but too many encoders will also lead to excessive time overhead of the model, and the performance improvement is no longer obvious. In Figure 4 and Table 6, when 7 known classes are randomly selected and the number of encoders is 12, there is no result. The reason is that 7 classes are randomly selected from the 8 known classes, and there are a total of $C_8^7 = 8$ possibilities. If 12 encoders are integrated, there must be repetitions.

While OpenCBD can distinguish known from unknown, it can also distinguish known classes.

It can be seen from Figure 5 and Table 7 that OpenCBD performs the best when 7 known classes are randomly selected in the first stage of training. Accuracy of 9 classes is over 72%, precision is over 76%, and recall and F1-score are over 75%.

As can be seen from Figure 6, the probability of Facebook-audio being wrongly classified into an unknown class is higher than the other 7 classes. Since there are 5 classes of data that are regarded as an unknown class, the number of unknown classes predicted as the unknown class is the largest and the color is the warmest (red).

In addition, we design different random selection methods in the first stage of training. A portion of each of the 8 classes of data is randomly selected, and the results are as follows.

As can be seen from Figure 7 and Table 8, the results of this training method are not much different from the original results, but the overall results of this method are slightly higher, all above 70%, and the highest accuracy can achieve more than 77%, and precision even exceeds 83%. Since all 8 known classes have data to participate in the training of the individual model in this method, the ensemble model learns more deeply for each known class.

Table 9: When the number of encoders in the ensemble model of OpenCBD is 4, 8, and 12, accuracy, precision, recall, and F1-score of the 9-class classification tasks (identifying 8 known classes and 1 unknown class) vary with increasing percentage of known classes randomly selected.

k	Accuracy			Precision			Recall			F1-score		
	w											
	4	8	12	4	8	12	4	8	12	4	8	12
60%	0.6746	0.6953	0.6976	0.7342	0.7544	0.7532	0.7104	0.728	0.7242	0.714	0.7363	0.7334
70%	0.6807	0.7207	0.7507	0.7523	0.7835	0.8031	0.6997	0.7326	0.7484	0.7159	0.7502	0.7685
80%	0.6846	0.7123	0.7253	0.7469	0.7779	0.7836	0.7124	0.7346	0.7357	0.7226	0.746	0.7533
90%	0.7192	0.7146	0.7361	0.771	0.7798	0.7939	0.7455	0.7317	0.7317	0.7519	0.7472	0.7526

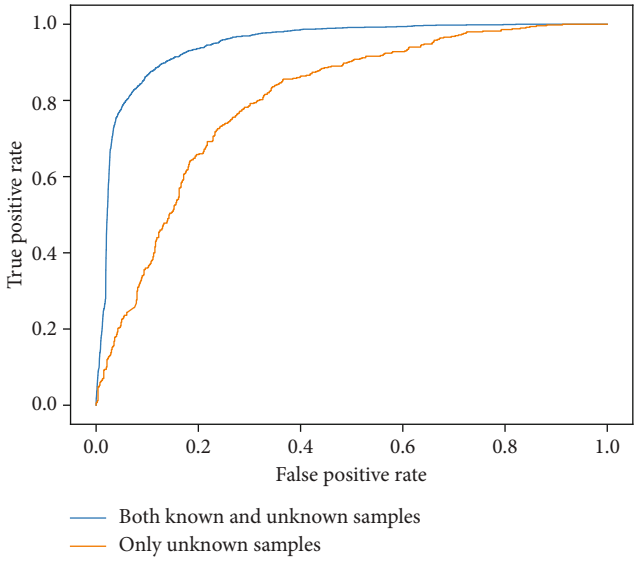

Figure 9: Area under ROC of OpenCBD when only identifying unknown class samples and both known and unknown class samples.

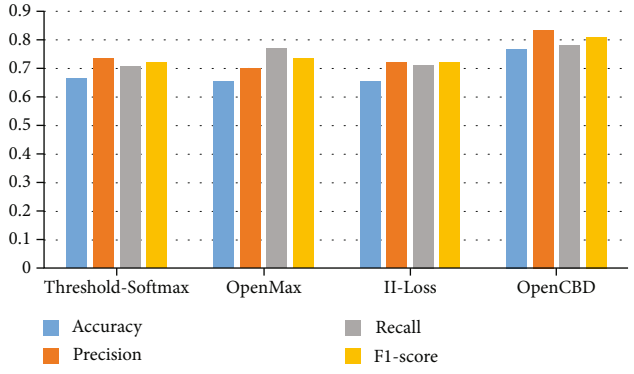

Figure 10: Performance comparison of four models on 2-class classification tasks.

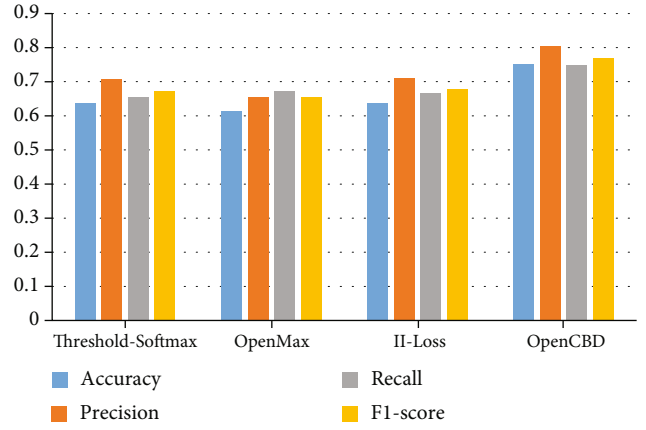

Figure 11: Performance comparison of four models on 9-class classification tasks.

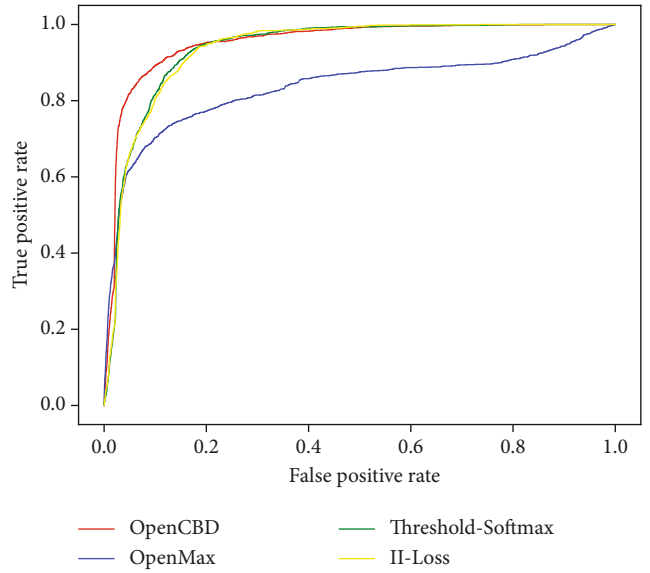

Figure 12: Area under ROC of four models.

As can be seen from Figure 8 and Table 9, when OpenCBD is doing 9-class classification tasks, the values of the 4 evaluation metrics increase with the increase of the number of encoders, and the results are better when the percentage of known classes is 70%. The highest accuracy can reach 73.61%, the highest precision is over 80%, the highest recall is nearly 75%, and the highest F1-score is 65.85%.

When the individual training selects 70% of the training set and the number of encoders is 12, the metrics of the two classes and nine classes are the highest. Therefore, we choose this ensemble method and compare the area under the ROC

curve (receiver operating characteristic curve) of identifying only unknown classes and identifying both known and unknown classes at the same time. The results are shown in Figure 9.

When verifying the performance of the OpenCBD model, we also selected several classic open-set recognition models for comparative experiments. We choose 3 models, namely, the threshold-based Softmax model [15], OpenMax model [15], and II-Loss-based model [19].

It can be seen from Figure 10 that the values of the four metrics in the 2-class classification of the OpenCBD model are basically around 80%, and the other three are around 70%. Our OpenCBD outperforms the other 3 by around 10% in the 2-class classification tasks. It can be seen from Figure 11 that in the 9-class classification, the metrics of the OpenCBD model are between 75% and 80%, and the other three are basically not more than 70%. Our OpenCBD outperforms the other 3 models by around 5%-10% in 9-class classification tasks. Figure 12 compares the area under ROC of the four models more clearly; the larger the area, the better the effect; and OpenCBD is significantly better than the other three.

5. Conclusion

In this paper, we proposed a novel model that can simultaneously identify unknown traffic and classify known traffic. The model could be trained on the known traffic of the closed set and tested on the network traffic of the open set. The model first combined the convolutional neural network and the transformer encoder to construct a deep learning-based model. Then, use the general pretraining method in the field of encrypted traffic analysis to pretrain from unlabeled traffic data, so that the model could learn the basic characteristics of encrypted traffic. Then, according to the characteristics of the open set, a three-stage training and testing process was designed. During training, choose the cross-entropy loss function suitable for classification and the II-Loss function suitable for clustering. At the same time, using the idea of ensemble learning, a traffic identification model OpenCBD based on open-set recognition was constructed. For real-world traffic data, if it belonged to a known class, the class to which it belonged can be identified, and if it belonged to an unknown class, it can be identified to belong to an unknown class. Experiments were carried out in public datasets, 8 classes of data were selected as known classes and 5 classes of data were selected as unknown classes, and a class-balanced dataset was constructed in the experiment to eliminate possible human influence to the greatest extent.

In the future work, we can consider the clustering of unknown classes of encrypted traffic and further separate unknown traffic according to its characteristics, which is convenient for further discovery and research of new classes of traffic.

Data Availability

This paper uses the ISCX public traffic dataset which is available in [39].

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61772548.

References

[1] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): a survey," *Computer Communications*, vol. 170, pp. 19–41, 2021.

[2] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[3] W. Wang, M. Zhu, and J. Wang, Eds.X. Zeng and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*, pp. 43–48, Beijing, China, 2017.

[4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[5] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, p. 106944, 2019.

[6] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: a flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1171–1179, Paris, France, 2019.

[7] X. Wang, S. Chen, and S. Jinshu, "Automatic mobile app identification from encrypted traffic with hybrid neural networks," *IEEE Access*, vol. 8, pp. 182065–182077, 2020.

[8] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "XAI meets mobile traffic classification: understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[9] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4016–4025, Long Beach, CA, USA, 2019.

[10] 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.

[11] S. Cruz, C. Coleman, E. M. Rudd, and T. E. Boult, "Open set intrusion recognition for fine-grained attack categorization," in *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, Waltham, MA, USA, 2017.

[12] X. Hu, G. Chunxiang, Y. Chen, and F. Wei, "CBD: a deep-learning-based scheme for encrypted traffic classification with a general pre-training method," *Sensors (Basel, Switzerland)*, vol. 21, no. 24, p. 8231, 2021.

[13] A. Bendale and T. E. Boult, "Towards open world recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1893–1902, Boston, MA, USA, 2015.

[14] C. Geng, S.-J. Huang, and S. Chen, "Recent advances in open set recognition: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3614–3631, 2021.

[15] A. Bendale and T. E. Boult, "Towards open set deep networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1563–1572, Las Vegas, NV, USA, 2016.

[16] Z. Y. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative OpenMax for multi-class open set classification," in *British Machine Vision Conference*, London, UK, 2017.

[17] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4011–4020, Long Beach, CA, USA, 2019.

[18] P. Oza and V. M. Patel, "C2AE: class conditioned auto-encoder for open-set recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2302–2311, Long Beach, CA, USA, 2019.

[19] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 154–162, Cincinnati, Ohio, USA, 2020.

[20] B. Liu, H. Kang, H. Li, G. Hua, and N. Vasconcelos, "Few-shot open-set recognition using meta-learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8795–8804, Seattle, WA, USA, 2020.

[21] K. J. Joseph, S. H. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5826–5836, 2021.

[22] C. Geng and S. Chen, "Collective decision for open set recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 192–204, 2022.

[23] A. Finamore, M. Mellia, and M. Meo, "Mining unclassified traffic using automatic clustering techniques," *Traffic Monitoring and Analysis (TMA)*, vol. 6613, pp. 150–163, 2011.

[24] C. Jun Zhang, Y. X. Chen, W. Zhou, and A. V. Vasilakos, "An effective network traffic classification method with unknown flow detection," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 133–147, 2013.

[25] J. Zhang, C. Chen, Y. Xiang, and W. Zhou, "Robust network traffic identification with unknown applications," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (ASIA CCS'13)*, Hangzhou, China, 2013.

[26] H. Yu, Y. Zhao, G. Xiong, L. Guo, Z. Li, and Y. Wang, "Poster: mining elephant applications in unknown traffic by service clustering," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*, pp. 1532–1534, Scottsdale, Arizona, USA, 2014.

[27] Z. A. Shaikh and D. G. Harkut, "A novel framework for network traffic classification using unknown flow detection," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 116–121, Gwalior, India, 2015.

[28] R. Lin, O. Li, Q. Li, and Y. Q. Liu, "Unknown network protocol classification method based on semi-supervised learning," in *2015 IEEE International Conference on Computer and Communications (ICCC)*, pp. 300–308, Chengdu, China, 2015.

[29] R. Ma and S. Qin, "Identification of unknown protocol traffic based on deep learning," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1195–1198, Chengdu, China, 2017.

[30] F. Ningjia, X. Yuwei, J. Zhang, R. Wang, and J. Xu, "FlowCop: detecting "stranger" in network traffic classification," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, Hangzhou, China, 2018.

[31] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–6, Sharm El Sheikh, Egypt, 2019.

[32] 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.

[33] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for DL-based traffic classifier update," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 397–405, Toronto, ON, Canada, 2020.

[34] A. A. Mohamed, A. H. Osman, and A. Motwakel, "Classification of unknown internet traffic applications using multiple neural network algorithm," in *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–6, Sakaka, Saudi Arabia, 2020.

[35] 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*, vol. 32, no. 5, 2021.

[36] J. Li, C. Gu, F. Wei et al., "LightSEEN: real-time unknown traffic discovery via lightweight Siamese networks," *Networks*, vol. 2021, pp. 1–12, 2021.

[37] X. Shuyuan, L. Li, H. Yang, and J. Tang, "KCC method: unknown intrusion detection based on open set recognition," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1343–1347, Washington, DC, USA, 2021.

[38] X. Hu, G. Chunxiang, and F. Wei, "CLD-Net: a network combining CNN and LSTM for internet encrypted traffic classification," *Security and Communication Networks*, vol. 2021, 15 pages, 2021.

[39] 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)*, Rome, Italy, 2016.