CL-ETC: A Contrastive Learning Method for Encrypted Traffic Classification

Ziyi Zhao*, Yingya Guo†‡, Jessie Hui Wang*, Haibo Wang*, Chengyuan Zhang*, Changqing An*
*Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
†College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
‡Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
Email: zhaozy19@mails.tsinghua.edu.cn, guoyingya90@163.com, hwang@cernet.edu.cn,
hbwang1994@gmail.com, chengyua21@mails.tsinghua.edu.cn, acq@tsinghua.edu.cn

***Abstract*—In the fields of network management and cyber security, encrypted network traffic classification is a critical task. Although Deep Learning (DL) models have been used in this field, they lack explicit control over data feature extraction, resulting in the retention of low-value features, which confuses the training and negatively impacts the classification performance. In this paper, we design a Contrastive Learning (CL) based encoder for extracting robust representation vectors with valuable features from unlabeled data. We create multiple augmentation samples for each input data by a unique augmenter. By narrowing representation vectors among similar augmentation samples and alienating them among dissimilar ones, the encoder can capture valuable features. We propose CL-ETC, a semi-supervised method based on the encoder. In CL-ETC, a well-trained encoder can be utilized to guide supervised classifier training to increase classification performance and training speed. We conduct experiments on three datasets, and the findings reveal that CL-ETC outperforms other models in a variety of metrics, including accuracy, precision, recall, F1-score, and classifier training convergence speed.**

***Index Terms*—Encrypted traffic classification, Semi-supervised learning, Contrastive learning**

I. Introduction

The purpose of network traffic classification is to divide traffic into distinct categories. Efficient and accurate traffic classification technology is essential to network management [1] and malware detection [2]. There were two prominent approaches to solving this challenge in the beginning. The first is a classification approach based on ports. The technique, however, failed due to port dynamic allocation [3]. Deep packet inspection (DPI) [4] is the second method for distinguishing different types of traffic by looking for patterns or keywords in data packets. However, with the development of network technology, especially the emergence and popularization of encryption technology represented by the HTTPS protocol [5], DPI can hardly extract relevant keywords from encrypted traffic payload.

In recent years, researchers began to focus on traffic classification algorithms based on Machine Learning (ML). Various ML models based on statistical or sequential features were proposed, including k-Nearest Neighbor (k-NN) [6], Random Forest (RF) [7], Support Vector Machine (SVM) [8], and Naive Bayes (NB) [9]. Although these models are effective, they all rely on feature engineering that needs a lot of experience, which limits their generalization.

In order to get rid of the dependence on feature engineering, researchers began to focus on DL models which can extract features from raw data and learn how to classify according to these features without any human intervention. Recent studies have demonstrated the superiority of DL models in classification, such as Convolutional Neural Network (CNN) [10], [11], Long Short-Term Memory (LSTM) [12], Stacked AutoEncoder (SAE) [13], and Multi-Layer Perceptron (MLP) [14]. However, we believe that the performance of DL models can be further improved. Because the present DL models, no matter supervised or semi-supervised, have no explicit control over the data feature extraction. In specific, these models cannot filter out the low-value features which can mislead the training of classification model and thus limit the final performance of classifier.

To handle the problem, we design an encoder based on CL to identify and extract valuable features by comparing the similarity and difference among augmentation samples in the same training batch. There is a principle that good representations should contain features that are invariant to small and local changes in input data and CL can capture these features in the process of comparison [15]. According to this principle, we create some similar but not identical samples from each traffic flow by augmenter to train the encoder. Through comparison, the encoder can find which features are low-value from similar augmentation samples and which are valuable from dissimilar augmentation samples. Finally, the encoder output the representation vector that retains the valuable features. Based on the encoder, we propose a semi-supervised model named CL-ETC to classify encrypted traffic. The training phase of CL-ETC can be divided into two sub-phases: pre-training phase and fine-tuning phase. The former phase is used to train the encoder to generate robust representation vectors from unlabeled data. Unlabeled data can provide useful information to mine features but cannot be used by supervised models. The later phase uses the well-trained encoder to guide the training of supervised classifier. We run experiments on three datasets to test the classification

Corresponding author: Changqing An (acq@tsinghua.edu.cn)

ISBN 978-3-903176-48-5

performance of CL-ETC. We also attempt to prove that the encoder based on CL can create robust representation vectors. Our contributions can be summarized as follows:

- We design an encoder based on CL to extract features from augmentation samples. The encoder can not only automatically extract key features, but also filter out low-value features according to the invariance of features. We also design a unique augmenter to generate augmentation samples for each input data.
- We propose a semi-supervised model for encrypted traffic classification. It not only takes advantage of labeled data, but also makes use of unlabeled data to provide more information.
- We carry out some experiments and prove that the encoder based on Contrastive Learning can better guide the classifier to learn how to distinguish different classes of encrypted traffic. The performance of CL-ETC is better than state-of-the-art network traffic classification methods.

The rest of the paper is organized as follows. Section II provides related work. The backgrounds are described in Section III. Section IV introduces the architecture of our framework. Details of the experiments and analysis of the results are presented in Section V. Finally, the conclusion of this paper is given in Section VI.

II. Related Work

DL has been widely used in the fields of Computer Vision (CV) and Natural Language Processing (NLP). For encrypted traffic classification, some researchers also try to transform it into the task of these two fields and solve it by DL models. In this section, we introduce some related work based on DL models, including traditional supervised methods and semi-supervised ones. These two kinds of method are different in feature extraction and subsequent processing.

A. Supervised Methods

Lotfollahi et al. tried to use one-dimensional CNN and SAE to automatically extract features from encrypted traffic payloads and then use the extracted features for packet level classification [16]. Wang et al. extracted the first fixed length bytes of the data, mapped the byte sequence to the single-channel gray scale image. They found that the images of different types of encrypted traffic data were quite different from each other, while the images of the same type are similar. Therefore, they use one-dimensional CNN [11] and two-dimensional CNN [10] as the classifier respectively. However, both of them above ignored other useful information, so Liu et al. proposed an end-to-end framework named FS-Net which takes the packet length sequence of flow as input and uses AutoEncoder to extract features [17]. In order to extract the information from different data types at the same time, Wang et al. proposed a parallel processing model called App-Net [12], which combines LSTM and CNN. In their work, LSTM takes the packet length sequence of a TLS flow as input and CNN takes the initial data packet of a TLS flow as input. They also proved that the hybrid model had better performance than the single model.

The feature extraction process of these methods is completely implicit. In other words, no representation vector is generated in the whole process, and the features are directly associated with the digital label after being extracted. If there are low-value features in these features, the training of the model can be misled and the classification performance can be impacted. Unfortunately, supervised models cannot control over the data feature extraction to filter out low-value features.

B. Semi-supervised Methods

The most obvious property of semi-supervised model is the combination of supervised learning and unsupervised learning. In general, the process of semi-supervised training can be divided into two phases: pre-training phase and fine-tuning phase. Unsupervised learning is effective in pre-training phase and mainly focuses on how to make representation learning from unlabeled data. Representation learning refers to the process of learning a mapping from input data to feature vectors. These vectors are used in fine-tuning phase to improve the performance of classifier trained by supervised learning. In the domain of encrypted traffic classification, the most commonly used feature vector extraction method are AutoEncoder [18] and its variants. Aouedi et al. constructed a framework [19]. The pre-training phase of the framework uses stack sparse AutoEncoder, and the fine-tuning phase uses a softmax layer as classifier. The experimental results showed that their model outperforms other conventional classifiers. Xing et al. proposed a method in [20] to detect anomalies in encrypted traffic. In their work, a pre-trained AutoEncoder based on LSTM is used to extract sequential features and the feature vector is regarded as input to the dictionary to learn the feature patterns. However, the encoder based on AutoEncoder just evaluates the value of representation vectors by the similarity between the input data and reconstruction data. Although this method is feasible, it does not filter out useless features at a deeper level so that many useless features are still retained in representation vectors and influence the final performance of the classifier.

III. Backgrounds

In this section, we first give the problem definition of encrypted traffic classification in our work. Then the structure of AutoEncoder and a simple framework of CL are introduced briefly. The former is the most common method to extract representation vectors from unlabeled encrypted traffic data and the latter is the main method to generate representation vectors in the paper.

A. Formal Definition of Problem

In this paper, we use data flow as our classification object. We can think that a complete session between client and server is composed of multiple data flows. Our ultimate goal is to predict the specific application of a given encrypted data flow.

Since it is a semi-supervised model, we assume that there are two datasets S^u and S^s, where S^u is an unlabeled dataset with a size of N^u, while S^s is a labeled dataset with a size of N^s. The former is used to train the encoder to extract the representation. The process is described by formal language to get a function f, such that $r_i^u = f(x_i^u), x_i^u \in S^u, 1 \leq i \leq N^u$, where r_i^u is the corresponding representation vector of x_i^u. The latter is used to train the supervised classifier, which is to get a function g such that $\hat{y}_i = g(r_i^s) = g(f(x_i^s)), (x_i^s, y_i) \in S^s, 1 \leq j \leq N^s$, where $\hat{y}_i$ is the predicted label and y_i is the real label of x_i^s. We aim to make $\hat{y}_i$ equal to y_i as much as possible.

B. AutoEncoder

AutoEncoder is a kind of unsupervised method to extract features from unlabeled data under the reconstruction mechanism [21]. Fig. 1 shows the structure of AutoEncoder.

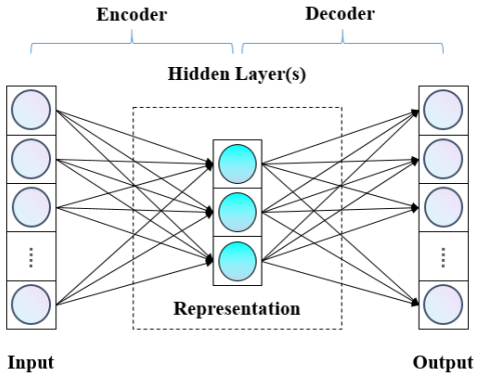


Fig. 1. The Structure of AutoEncoder

AutoEncoder has a symmetrical structure which is divided into encoder and decoder. Encoder maps input data $x = \{x_1, x_2, \cdots, x_d\}$ into compressed representation vector r. And r is used by decoder to generate reconstruction sample $\hat{x} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_d\}$, namely the output in Fig. 1. The loss function computes the distance between reconstruction data and original input. One of loss functions can be the mean-square error between x and $\hat{x}$, namely

$$\begin{aligned} Loss_{AutoEncoder} &= \frac{1}{d} \|x - \hat{x}\|_2^2 \\ &= \frac{1}{d} \sum_{i=1}^{d} (x_i - \hat{x}_i)^2 \end{aligned} \tag{1}$$

During training phase, AutoEncoder jointly learns parameters in both encoder and decoder by minimizing loss function, and extracts abstract features. In other words, the less the difference between reconstruction data and original input is, the higher the quality of representation vector is.

C. Contrastive Learning

CL is another kind of unsupervised method. The goal of CL is brief and intuitive: the representation vectors of "similar" samples should be close, while those of "dissimilar" samples should be alienated [15]. To achieve the goal, many researchers try to definite the similarity and dissimilarity so that various models are proposed. In this work, we refer to the definition proposed in [22] that the augmentation samples from the same data are similar, but these samples are dissimilar compared with the augmentation samples of the other data. Fig. 2 shows a simple framework of CL.

For a batch of data in pre-training phase, the augmenter will create some similar but not identical samples for each of them. The encoder then extracts the representations from all the augmentations. These representations are then fed into a projection head which is a small MLP to obtain embedding vectors lower dimensional than representations to calculate a contrastive loss function.

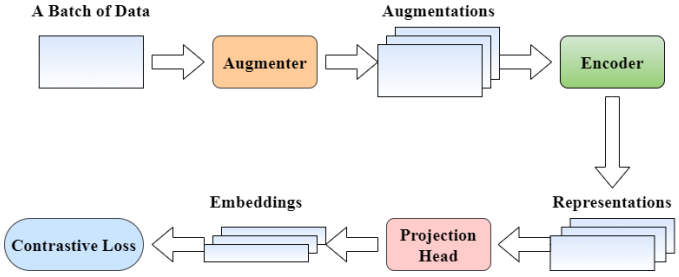


Fig. 2. A simple framework of Contrastive Learning.

We assume that S is a set which consists of all the augmentation samples created by the encoder from the same training batch. Let

$$\text{sim}(x_p, x_q) = {z_p}^\top z_q / \|z_p\| \|z_q\| \tag{2}$$

denote the similarity between augmentations x_p and x_q, namely the cosine similarity between embedding vectors of them z_p and z_q. Then the contrastive loss function for x_i in S is defined as

$$L(x_i) = -\log \frac{\sum_{x_j \in G, x_j \neq x_i} \exp(\text{sim}(x_i, x_j)/\tau)}{\sum_{x_k \in S, x_k \neq x_i} \exp(\text{sim}(x_i, x_k)/\tau)} \tag{3}$$

where G is a subset of S and contains all the other augmentations similar with x_i, and τ is a temperature parameter to control the sensitivity of the cosine similarity. Typically, G contains only two samples: one is x_i and the other is the augmentation sample created from the same input data as x_i. At last, the contrastive loss function of the batch can be defined as

$$Loss_{CL} = \frac{1}{\|S\|} \sum_{x_i \in S} L(x_i) \tag{4}$$

where $\|S\|$ is the number of augmentations in S.

With the increase of the number of epochs, representation vectors of similar samples gradually converge, while the dissimilars gradually alienate. The process reflected in the encoder level can be understood as that the intrinsic features which can reflect the nature of the data are retained, while the meaningless features which are variances introduced by augmenting are filtered out. Good representations should capture

features that are invariant to augmentation, and representations extracted by CL can meet the requirement [15].

It should be emphasized that for encrypted traffic data, we have to develop a new data augmentation method. Because our data is unique in comparison to those of other fields.

IV. Design of CL-ETC

We introduce CL-ETC in this section, including data preprocessing and implementation details of each unit. The whole framework is shown in Fig. 3.

A. Data Preprocessing

We separate the labeled data into unlabeled training data, labeled training data, validation data and test data. The unlabeled training data is used to train the encoder in pre-training phase and others are used in fine-tuning phase. In consideration of the negative impact of input data with imbalanced categories on results [23], we control the distribution of categories by sampling to ensure that all of them are balanced in scale. In specific, for unlabeled data and labeled data, we use random under-sampling and random over-sampling for the majority classes and the minority classes of them to obtain a fair distribution. In the end, we remove all the labels of unlabeled data to make them really unlabeled.

B. Augmenter and Encoder in Pre-training Phase

The purpose of pre-training phase is to train a superior encoder from unlabeled data and save the well-trained architecture and parameters as initial state for the encoder in fine-tuning phase. In order to further improve the learning ability of the encoder from unlabeled data, we designed an augmenter to create augmentation samples for each input data. Augmentations can expand the scale of training data for encoder learning. The encoder can extract valuable features invariant to small or local changes in input data from augmentations to create robust representation vectors.

It should be noted that although the CL was early applied in the field of CV, we cannot use the same augmentation methods of CV. Because our data has semantics and augmentation methods of CV can destroy semantics easily. It is necessary to adopt a method that can produce multiple similar samples but change the semantics in part. To solve the problem, in CL-ETC, the augmenter first creates two copies of the input data, and then it randomly selects a position in these copies and replaces the consecutive m (m is a hyper-parameter) bytes starting from that position with 0. Finally, the processed copies become the augmentation samples.

We use one-dimensional CNN structure as our encoder because it is good at processing the sequential data like encrypted traffic [11]. As for the projection head, in other researches, this unit can be a simple linear transformation or a non-linear MLP [15]. In our work, we use a non-linear MLP with 2 layers as the projection head. In the end, the contrastive loss function is computed according to Eq. (4). When the output of the contrastive loss function converges, it means that the encoder can be transferred to the downstream.

C. Dynamic Training in Fine-tuning Phase

The network structure of fine-tuning phase is similar to the traditional supervised learning and the encoder trained in pre-training phase can be used in this phase. According to Fig. 3, all the input data must be processed by the encoder into the representation vectors. And then the classifier learns to discriminate different classes from these representation vectors.

In order to make the encoder better adapt to the new training data, we do not fix its parameters, but allow them to change dynamically with the training process of the classifier. Our classifier is a four-layer preceptron. The reason why we choose such a simple neural network as classifier is to better evaluate the guidance ability of the encoder to the classifier. In input layer and hidden layers of classifier, we use ReLU activation function

$$ReLU(x) = max\{0, x\} \tag{5}$$

to make classifier improve the ability of learning non-linearity contained in representation vectors. And softmax function, defined by Eq. (6), is used to predict the probability of the real label for input sample.

$$\begin{aligned} p_t(i) &= softmax(y_i) \\ &= \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)} \end{aligned} \tag{6}$$

where $1 \leq i, j \leq n$ represents a category, y_i and y_j are the predicted output of the last linear layer, $p_t(i)$ is the probability corresponding to category i.

In fine-tuning phase, the loss function of classifier is cross entropy which is defined as

$$Loss_{classifier} = -\sum_{j=1}^{N}\sum_{i=1}^{n} \mathbf{I}[i = k] \log(p_t(i)) \tag{7}$$

where $\mathbf{I}[i = k] \in \{0, 1\}$ is an indicator function evaluating to 1 only if $i = k$, namely the predicted label is the real label.

Fine-tuning phase also needs to avoid the over-fitting problem and we use the early stopping trick to overcome the problem. Specifically, we continue to observe the performance changes of the classifier on the validation data. After a few of epochs, if the performance of the classifier cannot be further optimized, we will stop the training. In the end, the most recent model with the best performance will be saved and be used in testing phase.

D. Testing Phase

Testing phase is the last phase in our framework to simulate and test the final performance of the classifier in unknown dataset. In this phase, the encoder and the classifier well-trained in fine-tuning phase are regard as a whole to classify the flows, and the parameters of them are fixed all the time.

V. Evaluation

In this section, we introduce our datasets and experimental environment in the beginning. The experimental contents include comparison experiments and analytical ones.

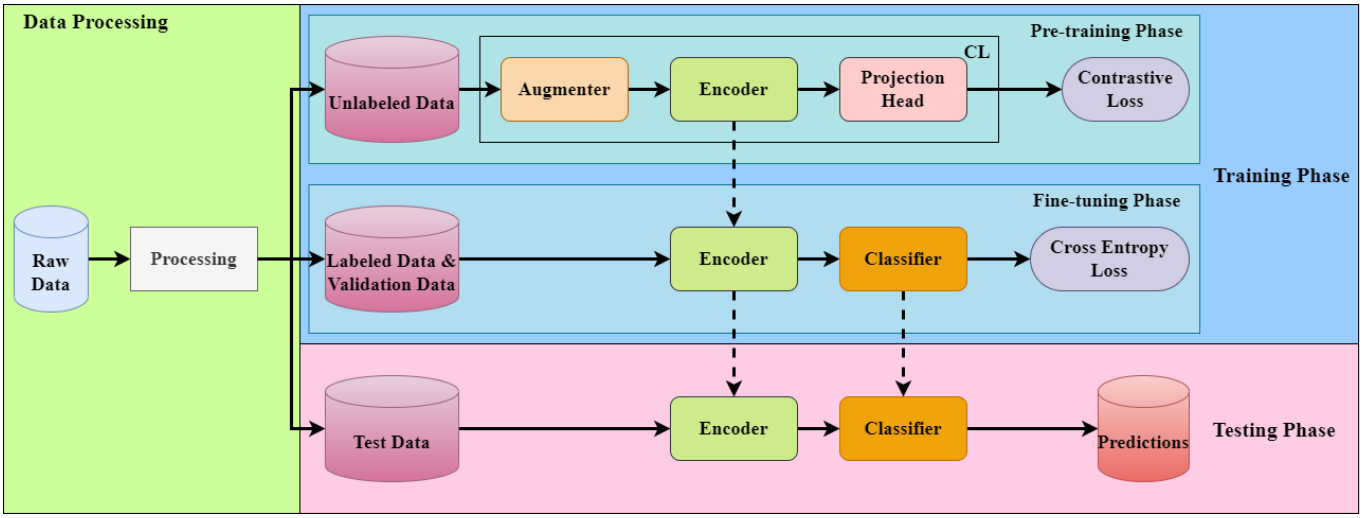


Fig. 3. The framework of CL-ETC, which consists of three main parts: data processing, training phase and testing phase. Training phase can be further divided into pre-training phase and fine-tuning phase.

A. Dataset

We use three datasets that are noted as TFB, TFM and THC. TFB and TFM correspond to the "benign" and "malware" subdataset in USTC-TFC2016 [10] respectively. THC contains various mobile traffic data captured by Wireshark [24] in a campus network. In specific, we create a Virtual Network Card (VNC) as a mobile hot-spot. Wireshark keeps listening on the VNC and captures traffic passing through it. Multiple smartphones are connected to the hot-spot and run the same app under the condition of collecting traffic marked with the same label. Each of the three datasets includes ten classes. The statistical information of these datasets are shown in TABLE I.

TABLE I
THE SCALES OF DATASETS

TFB		TFM		THC	
Category	**Size**	**Category**	**Size**	**Category**	**Size**
BitTorrent	15K	Cridex	461K	Airbnb	2K
Facetime	6K	Geodo	213K	BiliBili	1K
FTP	360K	Htbot	169K	Gaodemap	5K
Gmail	25K	Miuref	81K	JD	6K
MySQL	200K	Neris	498K	Kuaishou	2K
Outlook	15K	Nsis-ay	352K	Netease	3K
Skype	12K	Shifu	500K	QQ	2K
SMB	925K	Tinba	22K	Taobao	3K
Weibo	2,610K	Virut	438K	Weibo	2K
WoW	140K	Zeus	86K	Zhihu	3K

In this work, most data are saved in PCAP format, in which packet bytes of traffic data are saved in hexadecimal. We first cut the data according to the flow level granularity. Then we extract the first 784 bytes of each flow. Those less than 784 will be filled with 0 at the end. We choose the first 784 bytes for two reasons. Firstly, these bytes exist in the first packet exchanged in the TLS handshake, including information like cipher suites and extensions in the payload which can help to fingerprint an application [12]. Secondly, we use one-dimensional CNN [11] and two-dimensional CNN [10] as comparative experiments. In order to better compare with them, we also select the same input length, i.e. 784 bytes. The bytes are converted into decimal range from 0 to 255 and further transformed into [-1, 1] by standardization, because normalized data can improve the accuracy and the convergence speed.

B. Experiment Setting

1) Environment Setting: PyTorch 1.4 and Python 3.7 are used as software framework and language that run on CentOS. Our server is Supermicro and GPU is Nvidia TITAN Xp.

2) Main parameters of CL-ETC: The specific parameters of CL-ETC can be summarized as TABLE II.

As for the other hyper-parameters, in pre-training phase, the batch size is 100 and training time is about 100 epochs, the learning rate and betas of Adam optimizer are 0.001 and (0.5, 0.99), and τ in contrastive loss function is 0.5. In fine-tuning phase, the maximum training time is 500 epochs, the learning rate of Adam optimizer are 0.00001, and others are same as pre-training pahse. In addition, the number of augmentations created by augmenter from the same traffic is 2, and the number of replaced consecutive bytes is 14.

3) Evaluation Metrics: In order to measure the quality of classification results, there are four metrics to evaluate and their definitions are shown as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{8}$$

$$Precision = \frac{TP}{TP + FN} \tag{9}$$

$$Recall = \frac{TP}{TP + FP} \tag{10}$$

TABLE II
THE MAIN PARAMETERS OF CL-ETC

		Operation	Input	Kernel	Stride	Padding	Output
CL	**Encoder**	Conv1d + ReLU	1 * 784	25	1	12	32 * 784
		MaxPool1d	32 * 784	3	3	-	32 * 262
		Conv1d + ReLU	32 * 262	25	1	12	64 * 262
		MaxPool1d	64 * 262	3	3	-	64 * 88
		Linear + ReLU	64 * 88	-	-	-	1024
		Linear	1024	-	-	-	588
	Projection Head	Linear + ReLU	588	-	-	-	392
		Linear	392	-	-	-	196
Classifier		Linear + ReLU	588	-	-	-	512
		Linear + ReLU	512	-	-	-	256
		Linear + ReLU	256	-	-	-	128
		Linear + Softmax	128	-	-	-	10

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

where TP is true positive rate, TN is true negative rate, FP is false positive rate, and FN is false negative rate.

In addition, we take the convergence speed into consideration. By observing the convergence of the classifier in fine-tuning phase, we can also compare the learning ability of the classifier to the representations generated by different encoders.

C. Comparison Experiments and Analyses

1) CL-ETC vs Other Models in Classification Performance: We use the traditional supervised models and the semi-supervised model based on AutoEncoder to compare the classification performance with CL-ETC. The comparison results are shown in TABLE III. In TABLE III, AE-ETC represents the semi-supervised model based on AutoEncoder. CNN-1D and CNN-2D represent the supervised model based on one-dimensional CNN and two-dimensional CNN respectively. MLP and SAE are another two supervised models.

We can draw the following conclusions. Firstly, the performance of semi-supervised models is generally better than that of the supervised models. There are two reasons for this result. For one thing, the two semi-supervised models use a lot of informative unlabeled data which supervised models cannot use. For another, both of them try to fully mine the information inside the data rather than directly learn how to map the original data to a digital label.

Besides, CL-ETC obtains the best performance on all three datasets while that of AE-ETC seems inferior. We think that the reason why AE-ETC performs poorly is due to its mechanism. In fact, AutoEncoder attempts to reconstruct the input data with the representation vector extracted by its encoder, and evaluates the value of the representation vector by the similarity between the reconstruction data and the input data. To improve the similarity, the encoder of AutoEncoder extracts as many features as possible and compresses them into a low dimensional representation vector. This mechanism makes many unnecessary features retained in representation vectors which mislead the training of classifier finally. CL does not consider a single sample when extracting features, but compares all the augmentation samples in the same training batch. The augmentations created from the same traffic by augmenter can be used by encoder to find out low-value features variant to local changes. Meanwhile, other augmentation samples can help encoder to mine the key features belonging to the traffic.

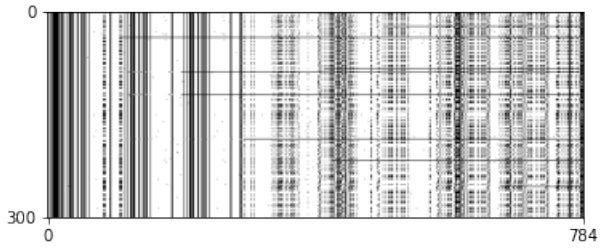
In the end, we find that the performance of each model on THC is generally weaker than that on other datasets. Because THC is more smaller than the other two in scale and THC contains some background traffic sent by some built-in applications of the smartphone manufacturers, such as cloud service. The both factors limit the performance of each model. Nevertheless, CL-ETC can still achieve the best performance. We think it is due to the unique loss function of CL. It can be found in Eq. (3) that the numerator part of the equation is the sum of the similarity between the augmentation x_i and its homologous augmentation samples which are created from the same input data, while the denominator part is the sum of the similarity between x_i and all the other augmentation samples. During pre-training phase, CL learns parameters by minimizing Eq. (3). In other words, with the advancement of the optimization process, the distance between similar samples in the sample space will be narrowed, but that between dissimilar samples will be alienated. In this case, the classifier is easier to capture the differences in traffic in different categories. Therefore, CL-ETC can still perform well in the dataset with small scale and noise.

2) CL-ETC vs AE-ETC in Extracting Representations: In the field of DL, some researchers propose to analyze the feature extraction of neural network by visualizing the output of hidden layers [25]. To further prove the encoder based on CL can better filter out useless features than that based on AutoEncoder, we make the two well-trained encoders process 300 flows in the same class respectively and extract the output of the first neural network layer. Because the first neural network layer is directly contact with the input data, the attention of the encoder to different positions in data can be found from its output. Considering that there are multiple output results of this layer, we first do softmax processing on each result to highlight the position most concerned by

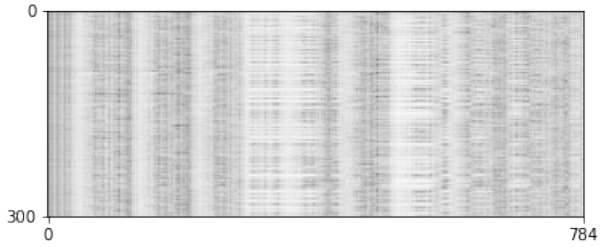
TABLE III
PERFORMANCE EVALUATION RESULTS OF DIFFERENT MODELS

	TFB				TFM				THC			
	Accuracy	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
CL-ETC	**0.9879**	**0.9910**	**0.9900**	**0.9910**	**0.9788**	**0.9790**	**0.9800**	**0.9790**	**0.9108**	**0.9110**	**0.9130**	**0.9100**
AE-ETC	0.8953	0.8140	0.8960	0.8500	0.9739	0.9750	0.9750	0.9750	0.9024	0.9070	0.9020	0.9010
CNN-1D	0.9694	0.9720	0.9690	0.9680	0.9723	0.9740	0.9730	0.9720	0.8938	0.9000	0.8950	0.8920
CNN-2D	0.9820	0.9830	0.9820	0.9840	0.9684	0.9690	0.9690	0.9690	0.8930	0.8960	0.8930	0.8940
MLP	0.8243	0.8810	0.8240	0.8190	0.9537	0.9560	0.9540	0.9540	0.8188	0.8320	0.8190	0.8190
SAE	0.7774	0.7960	0.7790	0.7710	0.9199	0.9250	0.9190	0.9210	0.5934	0.6380	0.5940	0.6010

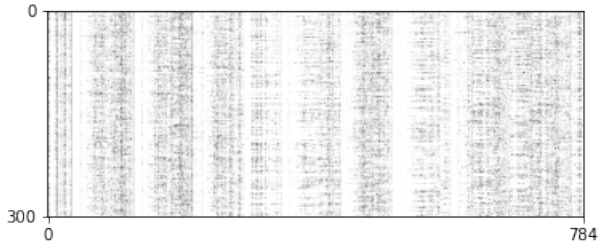
the encoder. Then we overlap these results and multiply the overlapped result by a magnification factor. The results are shown in Fig. 4.



(a) Input data



(b) AE-ETC



(c) CL-ETC

Fig. 4. Visualization of input data in the same class and the output of the first neural network layer in the encoders of AE-ETC and CL-ETC.

In (b) and (c) of Fig. 4, the darker the color is, the more attention the encoder pays to the position of input data. We can find a large number of the same positions in the results of AE-ETC and CL-ETC, which are paid more attention by their own encoders. In other words, their respective encoders believe that key features of data can be extracted from these positions. In addition, in the result of CL-ETC, there are many white areas. While the same areas are light gray in the result of AE-ETC. To put it another way, the encoder based on AutoEncoder still extracts features from these less important positions, but the encoder based on CL can nearly ignore them to ensure the representation vectors contain less low-value features.

In addition, although we cannot explain why the both encoders focuses on all these dark positions, it is certain that in our experiment, no matter which kind of data is input, the information of packet header can be concerned by both of them. In other words, the information contained in header packets arranged before the payload has always been regarded as one of the sources of key features.

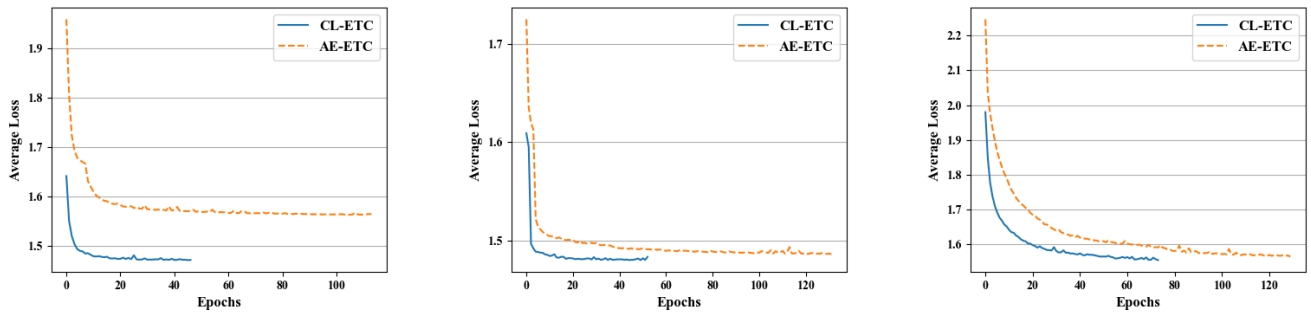
3) CL-ETC vs AE-ETC in Convergence Speed: The metrics given in V-B3 can only measure the final classification ability of AE-ETC and CL-ETC, but cannot reflect the guidance ability of different encoders to classifier. Therefore we introduce the convergence speed of the classifier in fine-tuning phase as reference.

In this paper, we use the change of average loss of the classifier to reflect the convergence speed. It should be emphasized that the whole network structure and the loss function of the classifiers in fine-tuning phase of the two systems are identical. The only difference between them is the encoder. As shown in Fig. 5, no matter which dataset is used, CL-ETC is better than AE-ETC in the starting point of average loss, the change rate of loss and the number of epochs reaching the best convergence state. These results show that the representation vectors extracted by the encoder of CL-ETC can reduce the learning pressure of classifier in deed. At the same time, alienated representation vectors of different classes also make it easier for the classifier to capture the differences among flows of different classes, so as to further optimize the convergence state.

D. Further Analyses on CL-ETC

1) Effect of the Projection Head on Performance: Although we have proved that CL-ETC can really filter out unnecessary features, we also need to avoid CL-ETC over-filtering features so as to ignore key features. As mentioned in [22], the existence of the projection head can ensure the formation and maintenance of more information in the representation vector. Because the non-linear projection head further filters out the key features, and its output is used to calculate loss.

In order to verify this conclusion is also valid in the field of encrypted traffic classification, we keep other conditions



(a) TFB (b) TFM (c) THC

Fig. 5. Convergence speed of classifiers of CL-ETC and AE-ETC in fine-tuning phase.

unchanged, just remove the projection head (noted as CL-ETC-NP), and carry out experiment on the three datasets again. The results are shown in TABLE IV.

TABLE IV
COMPARISON BETWEEN CL-ETC AND CL-ETC-NP

		Accuracy	Precision	Recall	F1-score
TFB	**CL-ETC**	0.9879	**0.9910**	**0.9900**	**0.9910**
	CL-ETC-NP	**0.9887**	0.9900	0.9880	0.9890
TFM	**CL-ETC**	**0.9788**	**0.9790**	**0.9800**	**0.9790**
	CL-ETC-NP	0.9757	0.9760	0.9760	0.9770
THC	**CL-ETC**	**0.9108**	**0.9110**	**0.9130**	**0.9100**
	CL-ETC-NP	0.8972	0.8990	0.8970	0.8970

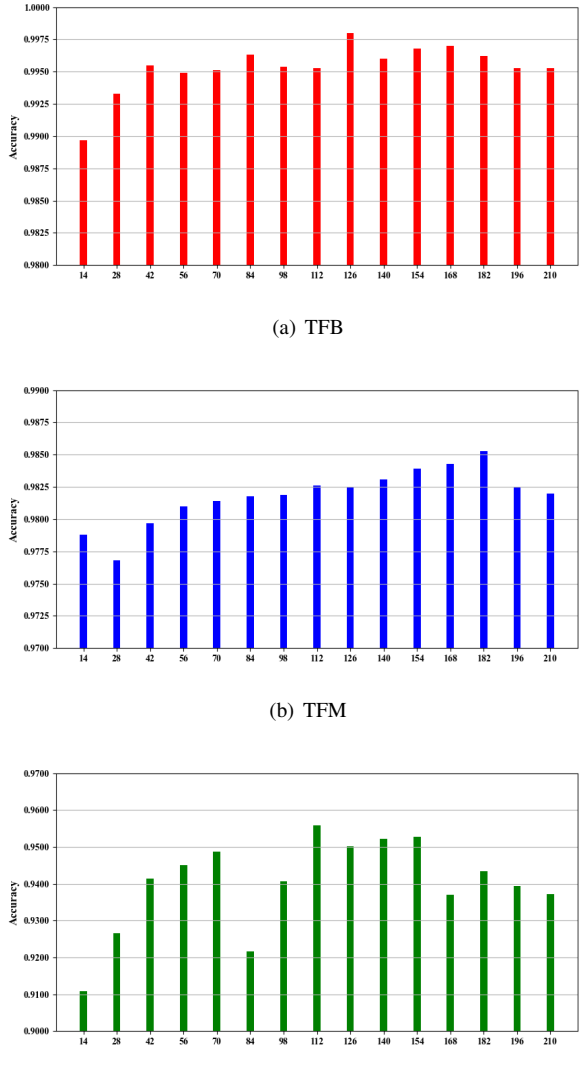
Apparently, in most experimental results, the performance of CL-ETC is better than that of CL-ETC-NP. These results prove the projection head is still essential to CL-ETC.

2) Sensitivity Analysis: Overwriting a consecutive part of input data by 0 is an unique processing of CL-ETC. It ensures that we can do minimal damage to the data and generate similar augmentation samples. However, if the number is too small, it will be difficult for the encoder to capture the difference in augmentations, and too big will destroy more valuable features.

Therefore, we train encoder with different number of replaced consecutive bytes (from 14 to 210) in pre-training phase, and show results of classifier in Fig. 6.

Obviously, no matter which datasets, the overall trend of accuracy change is first increased and then decreased. Due to the difference in data distribution in datasets, the corresponding number of each dataset is different when classifier reaches the best state (TFB: 126, TFM: 182 and THC: 112). In other words, the specific number should be set according to actual situation. Although our model can surpass other models in performance in previous experiments, this experimental result proves that our model still has room for improvement.

In fact, increasing the number of augmentation samples can also improve the effect of the model. However, this improvement is very weak and more samples means more pre-training time for the encoder and more memory consumption for data. Therefore, the result isn't shown in the paper.



(a) TFB

(b) TFM

(c) THC

Fig. 6. Accuracy of CL-ETC with different number of replaced consecutive bytes in augmentation samples.

VI. Conclusions

In this paper, we design an encoder based on CL to filter out useless features of data so that it can form more robust representation vectors. Because valuable features are invariant to data augmentation, we create multiple similar but not identical augmentation samples for each input data by a unique augmenter and use these augmentation samples as the input of our encoder. The encoder is trained to narrow representation vectors among similar augmentation samples and separate them among dissimilar ones in order to filter out low-value features. Based on the encoder, we propose a semi-supervised encrypted traffic classification model named CL-ETC. It can use both labeled data and unlabeled data for classification. In pre-training phase, the unlabeled data is used to train the encoder based on CL, and the labeled data is utilized to train the classifier in fine-tuning phase. The well-trained encoder can guide the training of classifier in fine-tuning phase, and ultimately improve the performance of the classifier. We validate the effectiveness of CL-ETC on three datasets, including two open datasets and a dataset full of mobile traffic collected by ourselves. The results demonstrate that CL-ETC can achieve excellent classification performance and outperform the existing models, even when the scale of the dataset is small. Furthermore, using visualization technology and comparing convergence speed, we show that the CL-ETC encoder can disregard unnecessary features and produce robust representation vectors containing valuable features for encrypted traffic.

VII. Acknowledgment

This work is supported by the National Key Research and Development Program of China (No.2020YFE0200500).

References

[1] Diogo Barradas, Nuno Santos, and Luís Rodrigues. Effective detection of multimedia protocol tunneling using machine learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 169–185, 2018.

[2] Blake Anderson, Subharthi Paul, and David McGrew. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, 14(3):195–211, 2018.

[3] F. Constantinou and P. Mavrommatis. Identifying known and unknown peer-to-peer traffic. In *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, pages 93–102, 2006.

[4] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, 57(5):76–81, 2019.

[5] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The cost of the" s" in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140, 2014.

[6] Donald McGaughey, Trevor Semeniuk, Ron Smith, and Scott Knight. A systematic approach of feature selection for encrypted network traffic classification. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2018.

[7] Blake Anderson and David McGrew. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, pages 1723–1732, 2017.

[8] Riyad Alshammari and A Nur Zincir-Heywood. Machine learning based encrypted traffic classification: Identifying ssh and skype. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–8. IEEE, 2009.

[9] Riyad Alshammari and A Nur Zincir-Heywood. Investigating two different approaches for encrypted traffic classification. In *2008 Sixth Annual Conference on Privacy, Security and Trust*, pages 156–166. IEEE, 2008.

[10] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*, pages 712–717. IEEE, 2017.

[11] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 43–48. IEEE, 2017.

[12] Xin Wang, Shuhui Chen, and Jinshu Su. App-net: A hybrid neural network for encrypted mobile traffic classification. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 424–429. IEEE, 2020.

[13] Zhanyi Wang. The applications of deep learning on traffic identification. *BlackHat USA*, 24(11):1–10, 2015.

[14] Pan Wang, Feng Ye, Xuejiao Chen, and Yi Qian. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, 6:55380–55391, 2018.

[15] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020.

[16] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[17] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. Fs-net: A flow sequence network for encrypted traffic classification. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1171–1179. IEEE, 2019.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. A semi-supervised stacked autoencoder approach for network traffic classification. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2020.

[20] Junchi Xing and Chunming Wu. Detecting anomalies in encrypted traffic via deep dictionary learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 734–739. IEEE, 2020.

[21] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[23] Qi Dong, Shaogang Gong, and Xiatian Zhu. Imbalanced deep learning by minority class incremental rectification. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1367–1381, 2018.

[24] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[25] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.