Cybersecurity

RESEARCH

Open Access

Subspace clustering via graph auto-encoder network for unknown encrypted traffic recognition

Ruipeng Yang[1,2*], Aimin Yu[1], Lijun Cai[1] and Dan Meng[1]

Abstract

The traffic encryption brings new challenges to the identification of unknown encrypted traffic. Currently, machine learning is the most commonly used encrypted traffic recognization technology, but this method relies on expensive prior label information. Therefore, we propose a subspace clustering via graph auto-encoder network (SCGAE) to recognize unknown applications without prior label information. The SCGAE adopts a graph encoder-decoder structure, which can comprehensively utilize the feature and structure information to extract discriminative embedding representation. Additionally, the self-supervised module is introduced, which use the clustering labels acts as a supervisor to guide the learning of the graph encoder-decoder module. Finally, we obtain the self-expression coefficient matrix through the self-expression module and map it to the subspace for clustering. The results show that SCGAE has better performance than all benchmark models in unknown encrypted traffic recognization.

Keywords: Encrypted traffic recognition, Deep learning, Graph auto-encoder

Introduction

The development of Internet technology not only gives rise to an endless stream of network protocols, but also makes a variety of Internet applications show explosive growth. While various network applications on the Internet provide users with convenient services, they also bring security risks to the network. For example, user information transmitted on the network is at risk of being illegally monitored, hijacked, stolen, and modified. Encryption technology emerged in the context of ensuring network security, providing Internet users with anonymity and protecting themselves from network surveillance systems, and is widely used in important network services (Dainotti et al. 2012). However, encryption technology also brings hidden dangers while protecting network security. For example, malware, such as Trojans and apt attacks, use encryption technology to bypass firewalls and intrusion detection systems. In addition, many companies prohibit their employees from playing games, watching videos and browsing the news in the company. However, this restriction can be broken through the use of encrypted tunnels. Therefore, to improve the level of network management, it is necessary to effectively identify kinds of encryption applications on the network.

Unknown traffic encrypted recognition is defined as identifying the type of application to which encrypted traffic belongs, such as streaming media including YouTube, Youku, etc., P2P including uTorrent, BitTorrent, etc. Since the encryption mechanism that makes the traffic features has changed, some traffic recognition methods are not applicable, such as methods based on port (Sen et al. 2004) and payload (Finsterbusch et al. 2013). As we all know, the machine learning methods only deal with the content below the transmission layer while the encryption technology generally only encrypts the load information rather than the flow features. Thus, this method is less affected by encryption

*Correspondence: yangruipeng@iie.ac.cn

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
Full list of author information is available at the end of the article

and has attracted widespread attention in the industry and academia. Most machine learning methods used for encrypted traffic recognition are supervised or semi-supervised (Taylor et al. 2017; Anderson and McGrew 2017; Liu et al. 2019). The general process is to first train a classifier using traffic features and labeled data and then use the classifier to recognize unknown traffic. The emergence of numerous massive unknown applications, however, makes it costly to obtain well-labeled samples in a limited time. In contrast, unsupervised machine learning based method can identify unknown encrypted traffic without a priori label information. At present, some feature-based clustering methods, such as *K*-means, DBSCAN, and auto-encoder (AE), as well as graph-based methods, such as spectral clustering, have been successfully applied to the identification of unknown applications (Erman et al. 2006; Zhao et al. 2019; Xie et al. 2012; Wu et al. 2017). However, in the absence of prior information, it is a difficult task to map many applications to limited application types. We observe that these feature-based clustering methods usually only focus on extracting useful representations from flow features, and rarely consider the structure of the data. In addition, current graph-based clustering methods generally focus on the representation of the relationship between nodes, but do not effectively combine the characteristic information of the nodes themselves. This inspired us to develop an unsupervised learning method that can comprehensively use both feature and structure information to identify the type of an application.

Our current manuscript introduces a subspace clustering via graph auto-encoder network (SCGAE) for unknown traffic recognition. The proposed method can comprehensively utilize the feature and structure information, and perform the recognition of unknown applications without prior label information. It mainly includes four modules: graph auto-encoder (GAE), self-expression, clustering, and self-supervised module. Firstly, we construct the initial application flow graph based on feature similarity and IP communication address. Then, in order to comprehensively utilize the application flow features and structural information, we propose a graph auto-encoder (GAE) module to extract more discriminative representations, and use a self-expression module to transform these representations into a coefficient matrix. Then, the coefficient matrix is transformed into affinity matrix by clustering module, and the clustering label is generated by spectral clustering algorithm. Next, the self-supervised module supervises the representative node learning by replacing the previous labels with the labels of the clustering module. Finally, we derive a unified framework by combining the latent representation as well as the clustering methods together. Here, we highlight our overall contributions in the following three aspects:

- We propose a GAE module to mine more discriminative information in the traffic data by reconstructing application flow features and structural features.
- We introduce a self-supervised module to restrict the distribution consistency of clustering outcomes, which helps to further enhance the accuracy of the unknown encrypted traffic recognition.
- We verify the SCGAE method with the actual encrypted network traffic data, which is better than state-of-the-art traffic recognition method.

The rest of this article is organized as follows. In "Related work" section, we review the study of traffic identification and unknown application identification. "Preliminaries" section introduces preliminary work, "Design overview" section introduces SCGAE in detail, and "Evaluation" section conducts experimental research. After a brief discussion in "Discussion" section , we conclude this paper in "Conclusion" section .

Related work

Classical traffic classification methodologies are roughly categorized into two types: port-based and payload-based. Particularly, the main idea of the port-based method is to sort traffic according to the port number which is contained in the package header information (Sen et al. 2004). This method can realize traffic categorization while the limited number of application serviced is used. However, with the emergence of port dynamic allocation (Constantinou and Mavrommatis 2006) as well as the general communication protocol port (Erman et al. 2007), the port-based traffic classification system has gradually lost its effect. In order to achieve a more accurate recognition effect, the payload-based method came into being, which uses the specific signature string in the payload to match, so as to realize the traffic classification (Ma et al. 2006; Finsterbusch et al. 2013). However, when encrypted traffic appeared, the payload of the traffic was no longer plaintext, and the payload-based method gradually became invalid because it could not obtain a signature from the payload.

Since encryption technology usually encrypts payload information, and machine learning methods process traffic data under the transport layer, this method is less affected by encryption, and thus has earned extensive notice from both the industrial and the academic fields. The key of machine learning methods is to extract network flow features, such as message interval, message size, and flow duration, and use them to construct

classifiers. For example, Conti et al. (2015) used the random forest method to identify the user's actions on the mobile phone through the features of the IP, packet size, port, and direction of the encrypted traffic generated by the marked user when using the application mobile client. Wang et al. (2017) used data packet headers and payloads to train both the convolutional neural and the long (and short) -term memory networks so as to gain intrusion detection. Aceto et al. (2018, 2019) designed a data flow source identification model based on multi-layer perceptron, considering the load bytes, TCP sliding window size, sequential packet arrival interval, direction and other features.

With the increasing demand for network supervision, many scholars have conducted research on the identification of specific applications or protocols. Erman et al. (2006) combined the *K*-means as well as the DBSCAN algorithms, which are unsupervised clustering methods, and proposed a semi-supervised method to classify both labeled and unlabeled applications. Xie et al. (2012) applied subspace clustering, using only relevant feature subsets instead of a unified feature subset to identify each application individually. Korczyński and Duda (2012) proposed a identification method for Skype traffic to determine the type of communication, namely, video meeting, on-line chat, voice call, document download and upload. In the computation experiments, some relevant results display that this method possesses a high accuracy of recognition, although the recognition between video and voice traffic is still a complex issue. Korczyński and Duda (2014) proposed a random fingerprint method based on Markov chain to identify applications. He et al. (2015) proposed a recognition method for Tor applications. This method first selected some representative flow features of application behavior, and used machine learning model to identify different applications. Some experimental outcomes indicated that this methodology had higher recognition accuracy. Shbair et al. (2014) proposed a method to identify the services running in the HTTPS connection, and defined specific features as the input of a multi-level HTTPS traffic identification structure on the basis of machine learning. Shen et al. (2016) proposed an application classification methodology according to the second-order Markov chain. These empirical results showed that the recognition accuracy of this method was improved compared with baseline methods, but in some cases, because the length of certificate packets of different applications was easy to cluster into the same class, this method sometimes still failed. Zhao et al. (2019) extracted and also aggregated the novel features from the data, and then combined the *n*-gram embedding policy with *K*-means clustering algorithm in order to divide the unknown traffic. Jin et al. (2021) proposed the mobile network traffic classification scheme, which extracts new patterns from the labeled traffic data to find unknown applications. The empirical results indicate that this strategy can successfully identify both the known and the unknown applications.

To sum up, existing traffic identification work mostly rely on a few or all known labels in the original data, but ail in an encrypted network environment where it is difficult to obtain known labels. In addition, few studies investigate the structural data information in the actual modeling, but the structural information between application flows can more effectively reveal the potential similarity of data. Therefore, our current paper aims to develop a novel unsupervised learning algorithm that does not depend on known labels, and comprehensively utilize the feature and structure information of the sample to perform effective unknown traffic recognition.

Preliminaries

Definition of application flow

Generally, an unknown traffic does not refer to a single package, but is composed of a series of packages generated by the two communicating parties during the transmission process. Such a group of packages is defined as application flow, which are uniquely identified by a 5-tuple, i.e., source IP address, source port number, destination IP address, destination port number as well as transport layer protocol (Lizhi et al. 2014). The type of application flow refers to the type of application layer protocol used by network traffic, such as mail traffic, web page traffic, and the type of f-smart file server. Note that in the process of network communication, there will be at least one application flow between two peers. Each application flow corresponds to a unique network application. The same application flow can contain two-way data from both sides of the communication.

Problem definition

Suppose $F = [f_i], i = 1, \ldots, N$ is a set of application flows with different types, where N is the number of application flows. Each application flow $f_i (1 \le i \le N)$ is a vector with d-dimensional features, expressed as $f_i = [f_{i1}, f_{i2}, \ldots, f_{id}]$.

Let $T = \{t_1, t_2, \ldots, t_K\}$ be a set of K application types, and each t_j $(1 \le j \le K)$ represents a type of network application. According to the definition of application flow, an application often has multiple application flows. For example, if application A includes 10 application flows, and application B includes 20 application flows, the sample size of the dataset is 30 (N=10+20), and the number of application types is 2 ($K = \{A, B\}$). It can be seen that the number of application flows N is usually larger than the number of application types K. The

purpose of traffic identification is to establish a mapping $f : F \rightarrow T$ between the application flow f_i and the application type t_j.In other words, for any application flow f_i, which consists of a series of packages, we need to find a unique application type t_j that matches it.

In an ideal situation, there are already a small number of labels that can mark the types of certain application flows, then the unknown application identification problem can be regarded as a semi-supervised learning task, and unmarked flows can be assigned to corresponding types through the training of some marked flows. However, there is a more extreme situation, that is, there is no label information in some complex encrypted network environment. This means that identifying unlabeled application flows in these network environments is more challenging. Here, we are interested in developing methods for identifying unknown applications that do not rely on known information, which accepts unlabeled traffic data.

Design overview

In order to more effectively identify unknown application, we propose a subspace clustering via graph auto-encoder network (SCGAE), which can simultaneously use the statistical features of application flows and the structural information between application flows to more comprehensively analyze and identify the types of unknown application.

We introduce the framework of SCGAE in this section, as illustrated in Fig. 1. The SCGAE is composed of four main modules: (a) a graph auto-encoder (GAE) module for mining the statistical characteristics and structural information of application flow at the same time; (b) a self-expression module for integrating potential flow features to construct coefficient matrix; (c) a clustering module for identifying different applications; and (d) a self-supervised module for constraining the distribution consistency of clustering and pseudo labels.

Constructing the graph

Before introducing SCGAE, it is necessary to construct a suitable application flow graph, so that the model can cluster more effectively. For general clustering tasks, the top n_k nearest samples to each sample can be filtered out based on similarity. However, in actual network traffic, traffic masquerading and application protocols of different versions will change the statistical characteristics of the application flow, so that different encrypted traffic types have similar characteristics. Direct similarity measurement can easily misjudge these encrypted traffic types, thus limiting the recognition ability of the model. Considering that the communication between host applications tends to be close over a period of time, this

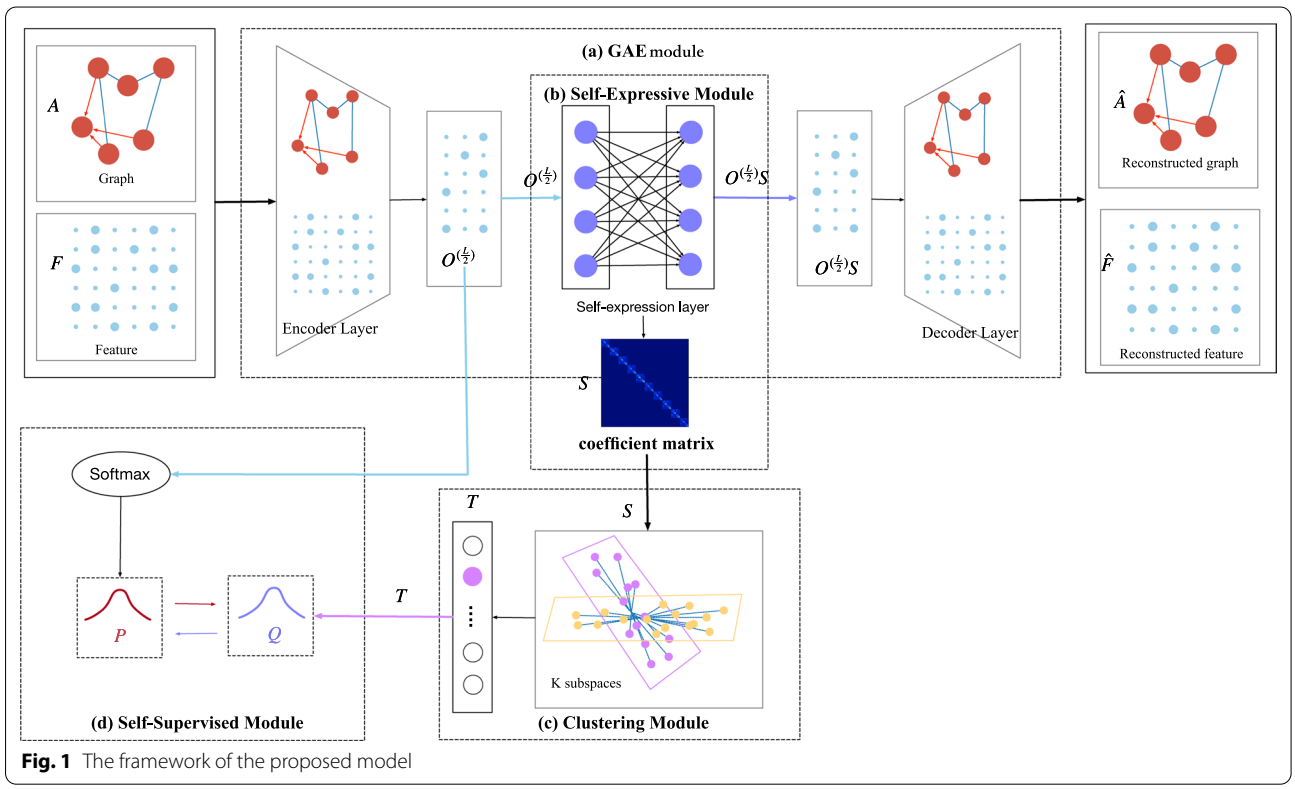

Fig. 1 The framework of the proposed model

behavior reflects the spatial distribution characteristics of traffic, so we prioritize the IP address of each application flow when composing the map. In addition, in order to efficiently aggregate neighbor features during message passing, we further consider applying a flow similarity measure to narrow the distance between samples with similar features. It is worth mentioning that the complexity of the flow graph construction method will increase if the attribute characteristics of the application flow are considered first rather than the IP address to which the flow belongs. Because similarity calculation needs to consider all application flows, and for complex network environments, the number of application flows often far exceeds the number of IP pairs, which will greatly increase the computational overhead. The method that prioritizes structural attributes can not only have high computational efficiency, but also make the model have a certain anti-interference during the training process. Figure 2 depicts the proposed flow graph construction method based on flow similarity and IP address association.

Firstly, according to the source and destination IP address, a graph $G(V,E), v \in V, e \in E$ is established, where the vertex V denotes the set of IP addresses, and the edge E is the set of application flow. As shown in Fig. 2a, we use nodes H, I, J and K, to denote the IP addresses, respectively. And the application flows are represented by edges $f_i, i = 1, \ldots, 6$. The different colors indicate different applications. It can be seen that f_2, f_4, and f_6 belong to the same application, while f_1, f_2, and f_5 belong to the same application. This situation is very common in a real network environment. For example, in the same chat software, different users can send audio messages or transfer files. Next, we transform graph G to $G^\star = (V^\star, E^\star)$ are as follows.

1 Convert the edge e in the graph G to the vertex $v^\star$ in the graph $G^\star$, and convert the vertex v in the graph G to the vertex $e^\star$ in the graph $G^\star$. Thus, the vertex $v^\star$ represents the application flow, and the edge $e^\star$ represents the application flow with the same IP address. For example, the two edges between H and I in Fig. 2a, i.e., f_1 and f_2, converts to two vertices in Fig. 2b. In addition, application flows with the same IP address have more similar location information, and their corresponding edges will be recorded in the edge set $e^\star$. For example, the two edges in Fig. 2a, namely f_1 and f_3, come from or go to the same vertex I, so the vertices f_1 and f_3 in Fig. 2b have an edge to connect.

2 Measure the similarity between a vertex in the graph $G^\star$ and its connected vertices. This similarity is the feature similarity between vertices. Calculate the similarity between each vertex in the graph and other vertices, and get the corresponding value of each vertex $v^\star$ Similarity vector *sim*. The similarity measure is based on the fact that application flows have the same IP address. In the current study, we choose Euclidean distance as the similarity measure, as shown in Eq. (1), where the feature of each application flow $f_i (i = 1, \ldots, N)$ is a d-dimensional vector, defined by $f_i = [f_{i1}, f_{i2}, \ldots, f_{id}]$.

$$dis_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})}. \tag{1}$$

3 Filter the edge group $e^\star$ so as to to get the transformed graph, which is denoted by $G^\star$. Specifically, the nearest distance is used to filter $e^\star$, which is an edge set consisting of application flows with the same IP address. For each vertex x_i, filter the top n_k nearest vertices in its edge set, that is, the flows with the highest similarity and the same IP address.

Finally, the transformed flow graph is shown in Fig. 2c. Taking vertex f_1 as an example, it and other three vertices, namely, f_2, f_3 and f_6, form three edges because of similar features and the same IP address. Constructing the flow graph in this way can make the relationship

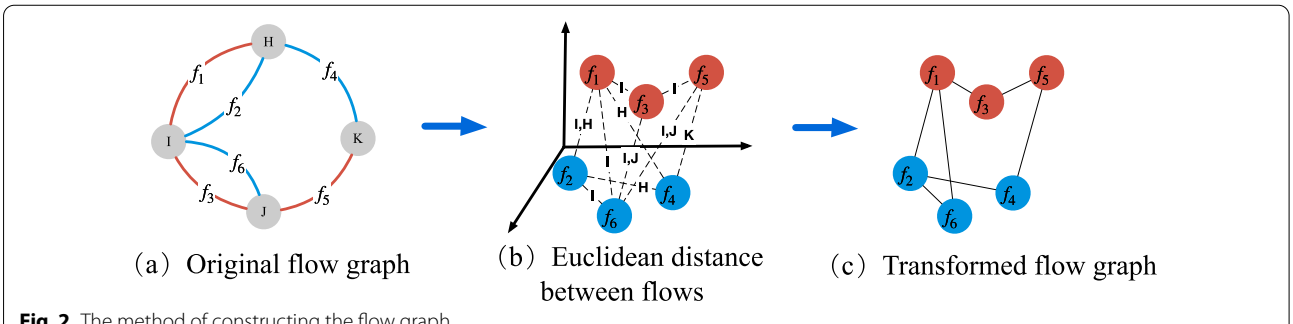

Fig. 2 The method of constructing the flow graph

between application flows in Fig. 2a more intuitively reflected in Fig. 2c.

GAE module

As mentioned above, the relationship between flow can effectively improve the clustering performance. Therefore, base on Kipf and Welling (2016a), Hammond et al. (2011), we proposes a GAE module to use the statistical characteristics of application flows and the structural information between application flows at the same time.

Specifically, graph convolution is performed for each GAE layer, and the high-order discriminant information is learned based on the feature matrix F and the adjacency matrix A:

$$O^{(l)} = \sigma\left(\hat{D}^{-\frac{1}{2}}(A+I)\hat{D}^{-\frac{1}{2}}O^{(l-1)}W^{(l)}\right), \tag{2}$$

where $W^{(l)}$ and $O^{(L)}$ represent the weight matrix and output matrix of the lth GAE layer, respectively. In addition, $\hat{D}^{-\frac{1}{2}}(A+I)\hat{D}^{-\frac{1}{2}}$ is the convolution kernel or filter , $\hat{D}$ is the degree matrix of A, where $\hat{D}_{ii} = \sum_j (A+I)_{ij}$. Furthermore, the sum of the adjacency matrix and the identity matrix, i.e. $A+I$, is to ensure the self-loop of each node.

It should be noted that the first layer in GAE module only uses the feature matrix F as the input matrix:

$$O^{(1)} = GAE(F,A) = \sigma\left(\hat{D}^{-\frac{1}{2}}(A+I)\hat{D}^{-\frac{1}{2}}FW^{(1)}\right) \tag{3}$$

Then, the output $O^{(l-1)}$ of the $(l-1)$ layer will be used as a new input matrix, that is, the input matrix of the lth GAE layer, to generate a new output matrix $O^{(l)}$:

$$O^{(l)} = GAE(O^{(l-1)},A) = \sigma(\hat{D}^{-\frac{1}{2}}(A+I)\hat{D}^{-\frac{1}{2}}O^{(l-1)}W^{(l)}) \tag{4}$$

In this study, we choose a simple inner product operation as same as Kipf and Welling (2016b) to reconstruct the relationship between samples using the output matrix $O^{(L)}$ of the last GAE layer,

$$\hat{A} = Sigmoid\left(O^{(L)T}, O^{(L)}\right) \tag{5}$$

where $\hat{A}$ is the reconstructed adjacency matrix. In addition, the embedding representation in the middle layer of the GAE module, namely $O^{(\frac{L}{2})}$, is used for self-expression and self-supervised module.

Self-expressive module

After creating a well-matched latent space through the GAE module, the goal of the self-expression module is to linearly represent each vertex by integrating the flow features of other vertices. Suppose the potential representation $O^{(\frac{L}{2})}$ comes from K subspaces, i.e., $X_i, i = 1,\ldots,K$. Then, each potential feature vector, namely $O_i^{(\frac{L}{2})}, i = 1,\ldots,N$, can be expressed as a linear combination of other samples in the same subspace:

$$O_i^{(\frac{L}{2})} = -\left(\alpha_1 O_1^{(\frac{L}{2})} + \cdots + \alpha_{i-1}O_{i-1}^{(\frac{L}{2})} + \alpha_{i+1}O_{i+1}^{(\frac{L}{2})} + \cdots + \alpha_n O_n^{(\frac{L}{2})}\right)/\alpha_i. \tag{6}$$

This is the definition of self-expressive property, and its matrix form is expressed as

$$O^{(\frac{L}{2})} = O^{(\frac{L}{2})}S, \tag{7}$$

where $S \in R^{N\times N}$ is the self-expression coefficient matrix with a block diagonal structure.

When the subspace is independent, the self-expression matrix S can be obtained by minimizing some norms of S, so it is mathematically transformed into an optimization problem.

$$\begin{aligned} &\min_C \parallel S \parallel_p \\ &s.t. O_i^{(\frac{L}{2})} = O_i^{(\frac{L}{2})}S, (\mathrm{diag}(S=0)), \end{aligned} \tag{8}$$

where $\|\cdot\|_p$ represents any regularized norm, and the constraint $\mathrm{diag}(C=0)$ is introduced to avoid singular matrices. In order to solve the above optimization problem, we relax the problem Eq (8) and transform it into:

$$\begin{aligned} &\min_S \parallel S \parallel_p + \frac{\lambda}{2} \parallel O^{(\frac{L}{2})} - O^{(\frac{L}{2})}S \parallel_F^2 \\ &s.t. (\mathrm{diag}(S=0)). \end{aligned} \tag{9}$$

As shown in Ji et al. (2017), the weight of the self-expression module corresponds to S, and it is further used for clustering module. It is worth noting that since the objects belonging to each category have inherent features that are different from other groups, generating a self-expression matrix through the latent space represented by GAE can make the spectral clustering in the clustering module more effective.

Clustering module

As described in the self-expression module, after obtaining the self-expression matrix S, we use the clustering module to label different application flows. Before using the clustering algorithm, we first set a threshold to filter the noise in the matrix, that is, retain other samples with high self-expression coefficients in each eigenvector. Next, we convert matrix S into affinity matrix Λ, as follows:

$$\Lambda = \frac{1}{2}\left(|S| + |S|^T\right) \tag{10}$$

Then, the affinity matrix Λ is used in the spectral clustering method (Ng et al. 2002), so as to realize the identification of unknown application flows, that is, the clustering result T of the SCGAE model is given.

Self-supervised module

In an unsupervised task, we cannot tell whether the clustering result T is consistent with the actual labels. Moreover, the embedding representation learned in the GAE module is only to obtain more discriminative information, and has no direct connection with the clustering module. To address this issue, we design an auxiliary task that uses a cross-entropy loss function in a self-supervised module to constrain and integrate the embedding representations learned by GAE module, making it more suitable for clustering tasks. Specifically, a GAE layer is used to cluster the latent representation $O^{(\frac{L}{2})}$, and the pseudo-label P can be obtained, where $P \in R^{N\times K}$. Then the result $T \in R^{N\times K}$ obtained in the clustering module can be expressed as a temporary label $Q \in R^{N\times K}$. The training objectives of the self-supervised module are:

$$\min -\sum_{i=1}^{N}\sum_{c=1}^{C} p_{ic} log \frac{p_{ic}}{q_{ic}}. \tag{11}$$

Note that if the cluster label T changes with each iteration, it may limit the convergence of the model. Therefore, we use a training technique by setting the number of clustering iterations T_c. This means that the trigger of the T update Q is after every T_c iteration, so the loss function is stable within the T_c iteration.

Overall loss function

We use graph and content reconstruction error as the loss function of GAE module, as shown in formula (12). Here, by minimizing the loss between A and $\hat{A}$, the GAE module can preserve more about the structural relationship between application flows in the embedding representation. This means that application flows formed by the same IP pair have a higher probability of belonging to the same application class than application flows formed by different IP pairs. In addition, we constrains the GAE module to retain enough flow feature by minimizing the F and $\hat{F}$ losses.

$$\begin{aligned} L_{gaeg} &= \frac{1}{2N} \parallel A - \hat{A} \parallel_F^2, \\ L_{gaec} &= \frac{1}{2N} \parallel F - O^{(L)} \parallel_F^2 . \end{aligned} \tag{12}$$

Next, in self-expression module, the loss function consists of self-expression loss and regularization loss, as shown in Eq. (13). The self-expression loss function is to make the embedding representation learned by the GAE middle layer as close to the transformed self-expression matrix as possible, and the regularization loss is to prevent the matrix C from becoming too sparse.

$$\begin{aligned} L_{ser} &= \parallel S \parallel_p, \\ L_{se} &= \frac{\lambda}{2} \parallel O^{(\frac{L}{2})} - O^{(\frac{L}{2})} S \parallel_F^2 . \end{aligned} \tag{13}$$

Then, in self-supervised module, the loss function is:

$$L_{ss} = -\sum_{i=1}^{N}\sum_{c=1}^{C} p_{ic} log \frac{p_{ic}}{q_{ic}}. \tag{14}$$

Finally, we can summarize the overall loss function in SCGAE as bellow,

$$L_{overall} = \lambda_1 L_{gaeg} + \lambda_2 L_{gaec} + \lambda_3 L_{ser} + \lambda_4 L_{se} + \lambda_5 L_{ss}, \tag{15}$$

where $\lambda_i (i = 1, \ldots, 5)$ represent the tradeoff coefficient.

Evaluation

Dataset

In this paper, we use WireShark to capture application traffic to build a local dataset. Since the experiment needs to verify the accuracy of traffic identification, the dataset needs to be differentiated and labeled according to application types. Therefore, the acquisition environment needs to be carried out in a clean and interference-free network environment. Then, we divide the collected data packets to obtain the application flow. This paper defines the start and end of the flow as follows: when a data packet occurs or the previous flow ends, it is the beginning of the flow; when the flow duration exceeds the specified time, or the data packet containing the RST or FIN flag is detected, it can be Consider the end of a flow. Next, the extracted flows is filtered to keep only the flow of the SSL/TLS protocol. Finally, referring to the methods of Shen et al. (2016), the Whois network service is used to parse out the domain name through the server IP address, and string matching is used to mark the label of the traffic.

Table 1 describes the dataset used in this study, including the traffic types (labels) to be identified and the corresponding specifics and sample sizes. Specifically, two datasets are considered in order to simulate a real network environment while obtaining relatively clean labeled data. The first dataset, ISCXVPN2016, is used to provide pure encrypted traffic (Lashkari et al. 2017). The dataset contains eight types of traffic, including browsing, chat, audio streaming, video streaming, email, VOIP, P2P, and file transfer, from 18 typical applications such as Skype and Spotify. Since the ISCXVPN2016 dataset is a pure dataset, it is difficult to obtain a pure dataset in a real network environment, even after purification.

Table 1 Description of the dataset

Dataset	Type(label)	Applications	Sample size
Ours	Audio streaming	Apple music and QQ music	12764
	Browsing	Chrome, Baidu and Sogou	18547
	Chat	Skype, QQ and Wechat	10556
	Email	Gmail, Foxmail and Outlook	5442
	E-commerce	Taobao and JD	9547
	File transfer	FTP over SSH and SSL	19854
	Video streaming	Youtube and Youku	15472
	P2P	uTorrent and Transmission (Bittorrent)	29870
ISCXVPN2016	Audio streaming	Spotify	721
	Browsing	Firefox and Chrome	1604
	Chat	Facebook and Hangouts	323
	Email	Gmail	282
	File transfer	FTP over SSH and SSL	864
	Video streaming	Youtube and Vimeo	874
	VoIP	Facebook, Hangouts and Skype	2291
	P2P	Kali	1085

Therefore, the second dataset comes from a real network environment. This dataset extends the first dataset with some applications, including applications not included in the first dataset, such as Netflix, QQ, etc. The second dataset collects traffic traces generated from October 17th to October 23th, 2021 and June 11th to June 13th, 2022. In addition, in order to better construct the traffic graph, we collect the traffic of different hosts rather than one host in different time periods.

Note that identifying unknown traffic is a clustering task with no prior knowledge and no training required. It does not require supervised learning and uses algorithms to group analysis between data. Therefore, all the processed flows were taken as experimental samples.

The selection of features is very important for machine learning method. As suggested by Lashkari et al. (2017), two different methods are used to select flow features. In the first method, we calculate the length of time during packet transmission, such as duration of the flow. In the second method, we measure the number of units of different features, such as bytes per second or packets per second. After these two method, the flow features used in this experiments are a total of 28. Moreover, the description of the flow features is shown in Table 2.

Table 2 Description of the flow features

Abbreviation	Interpretation of features
pl	Packet length (min,max,mean,std)
nppf	Number of packets per flow
fiat	Flow inter-arrival time (min,max,mean,std)
activ	Active time before flow becomes idle (min,max,mean,std)
idle	Idle time before flow becomes active (min,max,mean,std)
fbps	Flow bytes per second
fpps	Flow packets per second
iatf	Inter-arrival time of forward flow (min,max,mean,std)
iatb	Inter-arrival time of backward flow (min,max,mean,std)
duration	Duration of the flow

Experimental setting

Comparison methods

To verify whether SCGAE can effectively identify unknown application traffic, we consider three types of unsupervised methods for comparison.

(1) Clustering method using flow features:

K-means LeCun et al. (2015) is a method that divides samples into K categories by measuring the similarity of samples.

BIRCH Lorbeer et al. (2018) is a hierarchical clustering method that needs to build a clustering feature tree that meets the limit of branching factor and clustering diameter.

GMM Reynolds (2009) uses the expected maximum algorithm to obtain a probability distribution model composed of K Gaussian distributions for clustering.

AE Hinton and Salakhutdinov (2006) is an embedding clustering method that utilizes K-means on low-dimensional representations learned from deep auto-encoder networks.

(2) Clustering method using flow graph:

Spectral Liu and Han (2018) is a method evolved from graph theory, which cuts the graph composed of all data points.

DeepWalk Perozzi et al. (2014) is a clustering method that learns embedding representations by truncating random walks.

DNGR Cao et al. (2016) is a low dimensional vector representation method that uses random walk model to obtain graphic structure information.

(3) Clustering method using both node features and graph structure:

VGAE Kipf and Welling (2016b) combines GCN structure and node reconstruction loss function to construct GAE network.

DAEGC Wang et al. (2019) is an AE based on graph attention, which jointly learns and optimizes the embedding representation and clustering.

SDCN Bo et al. (2020) combine AE and GCN to obtain low dimensional representation, and integrate structural information into deep clustering.

Setting of SCGAE

Since the graph structure composed of application flow contains certain prior information, which is beneficial to the convergence of the model, the initial learning rate is set to 0.001. The Xavier method is used to initialize the model parameters (Glorot and Bengio 2010). In the self-supervised module, we sets $T_c = 5$, which means that the model will update Q with T after every 5 iterations. In addition, we also pre-train the GAE module, i.e. the model is trained without the self-expression and self-supervision modules to obtain a set of initial training weight parameters on the GAE module. In subsequent experiments, we add self-expression and self-supervision modules to optimize the network.

To make the methods more comparable, the dimensions of all benchmark clustering models and the proposed SCGAE model are set to $input - n_z - cluster - output$, where $input$, n_z and $output$ are dimensions of the input, output and middle layers, $cluster$ represents the kinds of application flow types. For all experiments, the experiments are carried out in version 1.9.0 of the Pytorch deep learning framework. All methods were run 10 times and the results were averaged.

Metrics

Four clustering evaluation metrics are used to compare the clustering performance of the proposed method with benchmark methods: accuracy (ACC), normalized mutual information (NMI), average Rand index (ARI) and macro F1 score (F1). Larger values of these metrics mean better clustering results.

(1) accuracy (ACC):

$$ACC = \max_{m} \sum_{i=1}^{n} t_i = map(c_i)/N, \tag{16}$$

where t is the real application flow category, c is the label of the clustering result, N is the number of samples, and $map()$ is a permutation mapping function used to realize the difference between t and c Reassignment to ensure correct statistics. In this paper, we use the Kuhn-Munkres algorithm to obtain optimal redistribution.

(2) normalized mutual information (NMI)

$$NMI = \frac{2I(t,c)}{H(t) + H(c)}, \tag{17}$$

where $I(t, c)$ represents the mutual information between t and c, and H represents their respective entropy.

(3) adjusted Rand index (ARI)

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}, \tag{18}$$

where $RI = \frac{TP+TN}{TP+FP+TN+FN}$ represents the ratio of correct decisions.

(4) Macro F1-score (F1):

$$F1 = \frac{2 \cdot P \cdot R}{P + R}, \tag{19}$$

where $P = \frac{TP}{(TP+FP)}$ represents the precision rate, and $R = \frac{TP}{(TP+FN)}$ represents the recall rate.

Experiments

Comparison experiment

Table 3 summarizes the experimental results for all clustering methods. The following conclusions can be drawn from Table 3:

(1) The method of directly using the flow features or the flow graph to the identification of unknown application has its own advantages. Specifically, among the methods that use flow features for clustering, the GMM method performs best. This is because the sample point after GMM projection is not a definite label, but the probability distribution of each data stream belonging to each category. This soft allocation label can provide additional impor-

Table 3 Clustering results of various methods

Dataset	Methods	Input	ACC	NMI	ARI	F1
Ours	K-means	Flow features	0.5122	0.3056	0.3017	0.2898
	BIRCH	Flow features	0.4078	0.2678	0.2189	0.2267
	GMM	Flow features	0.5478	0.5757	0.3417	0.3516
	AE	Flow features	0.5167	0.4165	0.4538	0.3659
	Spectral	Flow graph	0.6167	0.5261	0.4770	0.3785
	DeepWalk	Flow graph	0.5147	0.3427	0.3618	0.3478
	DNGR	Flow graph	0.6787	0.5876	0.5879	0.3897
	VGAE	Flow features and Flow graph	0.6784	0.6870	0.6179	0.4157
	DAEGC	Flow features and Flow graph	0.7585	0.7658	0.7868	0.5765
	SDCN	Flow features and Flow graph	0.8197	0.7998	0.8178	0.6679
	SCGAE	Flow features and Flow graph	**0.9067**	**0.8679**	**0.9235**	**0.6922**
ISCXVPN2016	K-means	Flow features	0.4972	0.3356	0.3416	0.3678
	BIRCH	Flow features	0.4256	0.2346	0.3246	0.2290
	GMM	Flow features	0.5367	0.5462	0.4471	0.3078
	AE	Flow features	0.5435	0.5454	0.4541	0.3512
	Spectral	Flow graph	0.5212	0.5263	0.3217	0.3465
	DeepWalk	Flow graph	0.5812	0.5168	0.4311	0.3766
	DNGR	Flow graph	0.5926	0.5442	0.4443	0.3789
	VGAE	Flow features and Flow graph	0.6279	0.6543	0.5214	0.4312
	DAEGC	Flow features and Flow graph	0.6927	0.5234	0.5482	0.5891
	SDCN	Flow features and Flow graph	0.7403	0.6829	0.6173	0.6612
	SCGAE	Flow features and Flow graph	**0.8122**	**0.8124**	**0.7761**	**0.7042**

Bold text indicates the best experimental results

tant information for traffic identification. Among the methods using flow structure for clustering, the DNGR method performs best, because it uses random walk model instead of traditional random sampling to obtain flow structure information.

(2) Compared with using flow features or flow graph directly, reconstructing node content and graph structure to obtain potential representation at the same time is conducive to better clustering. We also notice that although the performance of VGAE is not ideal in the third type of method, it has achieved good clustering results compared to the method of using feature or graph information for traffic identification. Since this method only considers the graph structure as a loss function when reconstructing the node representation, there is a certain amount of information loss. Therefore, it is reasonable to reconstruct attribute information and graph structure in GAE module.

(3) The performance of DAEGC and SDCN is better than other comparison methods, because they complete the training and optimization of all modules in one step, and introduce a self-supervised mechanism to make pseudo labels and clustering labels as close in distribution as possible. The SCGAE proposed in this paper adopts the same training strategy and can effectively obtain the embedding representation of clustering information.

(4) Since SDCN uses flow structure information, it achieves better results than most methods in traffic identification. However, all methods except SCGAE do not consider the self-expression information of the latent representation. But learning a more discriminative matrix of self-expression coefficients helps to achieve more accurate subspace clustering. Therefore, the SCGAE proposed in this paper shows its superiority on real datasets.

(5) Among all methods, the proposed SCGAE performs best on all metrics. In particular, compared with the suboptimal method SDCN, SCGAE improves by 11%, 9%, 13% and 3% on the four evaluation metrics of ACC, NMI, ARI and F1, respectively. Among them, a higher ARI means that the traffic distribution identified by SCGAE is closer to the real traffic distribution.

Identification results for each application

To analyze the classification results of SCGAE in each application type, we summarize the results of unknown

Table 4 The results of SCGAE for each application

Dataset	Taffic(Type)	Precision	Recall	F1
Ours	Audio streaming	0.8525	0.7682	0.8082
	Browsing	0.8245	0.7183	0.7677
	Chat	0.5656	0.5234	0.5437
	Email	0.5245	0.5223	0.5234
	E-commerce	0.8412	0.9269	0.8820
	Fille transfer	0.6252	0.6078	0.6164
	Video streaming	0.7829	0.6529	0.7120
	P2P	0.8989	0.5523	0.6842
ISCXVPN2016	Audio streaming	0.7112	0.5829	0.6407
	Browsing	0.8911	0.7045	0.7869
	Chat	0.6721	0.6712	0.6716
	Email	0.5845	0.6823	0.6296
	Fille transfer	0.4623	0.5523	0.5033
	VoIP	0.7734	0.8422	0.8063
	Video streaming	0.9372	0.7912	0.8580
	P2P	0.6812	0.8023	0.7368

traffic recognition in Table 4 with F1 as the main metric. It can be seen from the table that F1 can achieve better results for the application type with large sample size. This means that the imbalance of data has a relatively large impact on the clustering performance of SCGAE, which is biased towards the classification with a large number of samples when clustering. In addition, we can observe that the recognition of audio streaming and video streaming is also relatively effective. This is because these two application types are easier to distinguish from other applications in terms of flow features.

Visualization

In order to more intuitively visualize the potential spatial changes in different stages, *t*-SNE (Van der Maaten and Hinton 2008) is applied to reduce the feature dimension of each stage in the proposed method to 2 dimensions, and it can be observed that different colors correspond to different clusters as shown in Fig. 3. It can be intuitively observed that from the initial stage to the final stage, the application flow samples are gradually clustered together to obtain a more differentiated representation. In addition, from Fig. 3, we found the introduction of self-representation and self-supervision module allows each data point to be grouped near the cluster centroid, and expands the distance between clusters.

Parameters analysis

To explore the impact of the n_k in the initial graph construction on the performance of SCGAE, we tune n_k in the range 5, 10, 15, 20, 25, 30, 35, 40, and the results are shown in the Fig. 4. Obviously, as n_k increases, all indicators show a trend of first increasing and then decreasing.

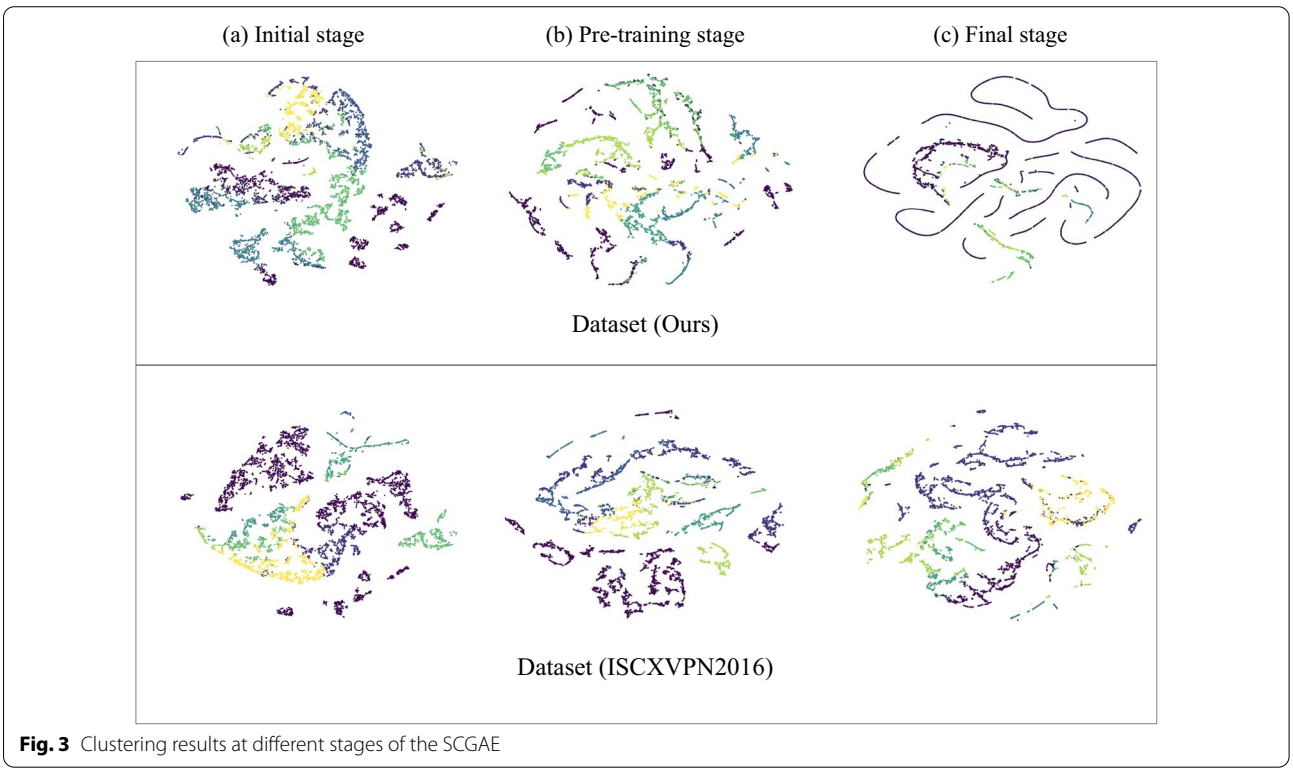

Fig. 3 Clustering results at different stages of the SCGAE

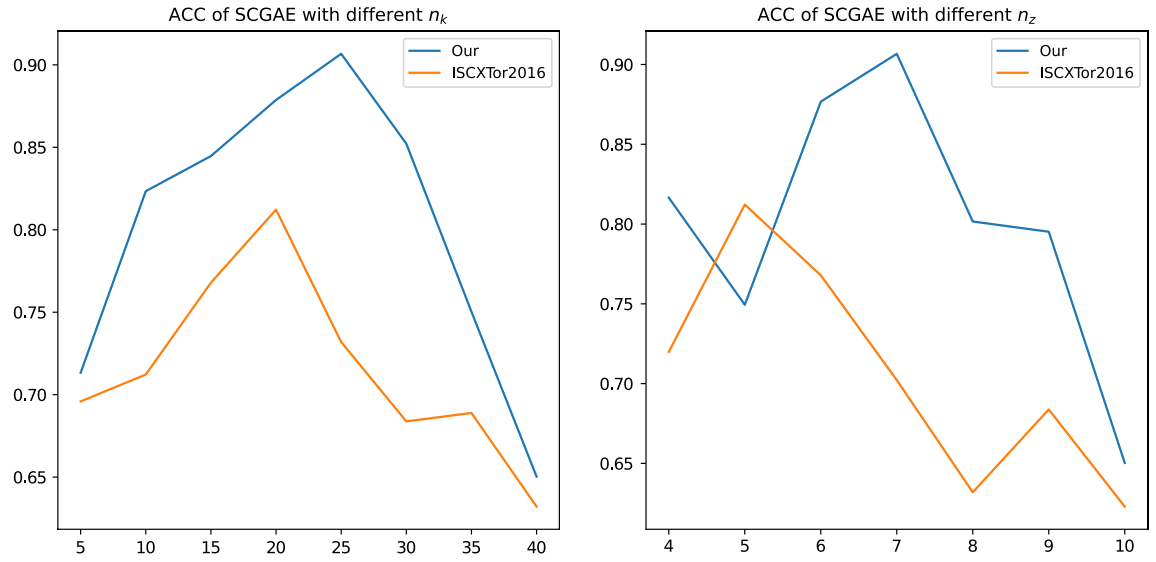

Fig. 4 ACC of SCGAE with different n_k and n_z

Note that, when the number of edges exceeds 20 or 25, the clustering performance of SCGAE drops rapidly. This is because the information redundancy in the graph structure is relatively large at this time, and GAE module contains more noise when reconstructing the graph. Therefore, in view of the balance between performance and training time, this paper chooses 25 as the number of edges when constructing the initial graph.

In addition, the dimension of the middle layer of the GAE module will directly affect the performance of SCGAE. A smaller value will result in the loss of potential information, while a larger value will cause the model to overfit. Therefore, we use different hidden state dimensions, namely $n_z = 4, 5, 6, 7, 8, 9, 10$, to train SCGAE. We can see that with the increase of n_z, three metrics, namely ACC, ARI, and F1, first increase and then decrease. Moreover, in the training process, the computing time is almost linear with n_z. In other words, the model with small dimension of the middle layer can be trained quickly, and when the dimension is very large, the model will also face the high risk of over matching. Therefore, according to the actual demand, we set n_z to 5 or 7.

Ablation study

In order to verify whether each module is conducive to the final clustering results, We delete one module from the SCGAE method in turn to build four incomplete clustering models.

SCGAE without graph The SCGAE method without L_{gaeg} loss in the GAE module,

$$L_{overall} = \lambda_2 L_{gaec} + \lambda_3 L_{ser} + \lambda_4 L_{se} + \lambda_5 L_{ss}, \tag{20}$$

SCGAE without content The SCGAE method without L_{gaec} loss in the GAE module,

$$L_{overall} = \lambda_1 L_{gaeg} + \lambda_3 L_{ser} + \lambda_4 L_{se} + \lambda_5 L_{ss}. \tag{21}$$

SCGAE without self-expression The SCGAE method without L_{se} and L_{ser} loss in the self-expression module,

$$L_{overall} = \lambda_1 L_{gaec} + \lambda_2 L_{gaeg} + \lambda_5 L_{ss}. \tag{22}$$

SCGAE without self-supervised The SCGAE method without L_{ss} loss in the self-supervised module,

$$L_{overall} = \lambda_1 L_{gaec} + \lambda_2 L_{gaeg} + \lambda_3 L_{ser} + \lambda_4 L_{se}. \tag{23}$$

The results of SCGAE and four incomplete clustering models are summarized in Table 5. The specific observations are as follows:

- In the GAE module, the reconstruction of both flow features and flow graph provide more potential information for clustering tasks. Specifically, in the absence of L_{gaeg} loss, the clustering performance of *SCGAE without graph* significantly decreases. This shows that the graph reconstruction can preserve the rich structural information of the embedded representation, so as to improve the clustering performance. As for *SCGAE without content*, the experimental results are not obvious, but it is still conducive to the final clustering. This indicates the impact of l_{gaec} loss is relatively small, but it is also necessary to ensure that the middle layer representative contains more content information in the original application flows.

Table 5 The results of SCGAE and four incomplete clustering models

Dataset	Methods	ACC	NMI	ARI	F1
Ours	SCGAE without graph	0.8345	0.7811	0.8190	0.6526
	SCGAE without content	0.8523	0.8012	0.8812	0.6725
	SCGAE without self-expression	0.7812	0.6254	0.7812	0.6712
	SCGAE without self-supervised	0.8123	0.7892	0.7738	0.6891
	SCGAE	0.9067	0.8679	0.9235	0.6922
ISCXVPN2016	SCGAE without graph	0.7341	0.7254	0.7254	0.5623
	SCGAE without content	0.7212	0.7217	0.7116	0.5123
	SCGAE without self-expression	0.7125	0.7623	0.7123	0.5451
	SCGAE without self-supervised	0.7274	0.7431	0.7368	0.6535
	SCGAE	0.8122	0.8124	0.7761	0.7042

- The self-supervised module provides an important contribution to the clustering results of SCGAE. The self-supervised module is introduced to utilize the pseudo-labels generated by the clustering module to help the GAE module learn the information suitable for the clustering module. When SCGAE method does not contain the L_{ss} loss, the self-supervised module cannot restrict the GAE module. In this case, the GAE module cannot obtain feedback to determine whether the learned embedding is well optimized. The overall performance of the SCGAE model will decrease.
- Compared with other modules, the self-expression module has the most obvious influence. Without L_{ss} and L_{ser} loss in the self-expression module, the four indicators of ACC, NMI, ARI and F1 on both datasets are reduced. This is because the subspace clustering performance is closely related to the self-expression matrix. In other words, in method involving subspace clustering, a well-learned nonlinear mapping contributes significantly to clustering performance.

To further analyze the importance of the input graph to model clustering, we compared the methods of constructing the graph. Specifically, we adopted a control variable method. The modules and parameters of SCGAE remain unchanged, the graphs constructed with and without considering IP addresses are input into SCGAE, as summarized in Table 6. It can be observed that the method considering IP addresses is more than 10% higher than the method using Euclidean distance in four metrics, which can verify that adding IP addresses in the initial graph can effectively use the relevant information between different flows.

Discussion

The widespread application of encryption technology in the Internet brings new challenges to the classification of unknown encrypted traffic. Unsupervised machine learning methods can identify unknown encrypted traffic without prior label information; however, many applications in networks have complex features that cannot be accurately classified with flow features or graph structure alone. Therefore, in an encrypted network environment without label information for reference, how to improve the recognition accuracy of unknown encrypted traffic by mining the inherent characteristics of traffic data is the core problem of this research.

First, this paper constructs an application flow graph based on feature similarity and IP communication addresses. Ablation experiments demonstrate the necessity and effectiveness of introducing IP addresses. In addition, we analyze the impact of parameter n_k on the model.

Table 6 Ablation study on the methods of constructing the graph

Dataset	Methods	ACC	NMI	ARI	F1
Ours	Using distance	0.8123	0.8243	0.8030	0.5712
	Using distance with IP addresses	0.9067	0.8679	0.9235	0.6922
ISCXVPN2016	Using distance	0.7623	0.7623	0.6234	0.5623
	Using distance with IP addresses	0.8122	0.8124	0.7761	0.7042

Next, in order to comprehensively utilize application flow features and structural information, this paper proposes a GAE module to extract more discriminative semantic representations, and introduces a self-supervised module to supervise nodes by replacing the previous labels with the labels of the clustering module representational learning. Ablation experiments demonstrate the effectiveness of each module, and each sub-module helps to improve the recognition performance of unknown traffic. In addition, comparative experiments show that the proposed method clearly outperforms existing unsupervised classification methods on the four indicators of ACC, NMI, ARI and F1. This is because the proposed model can simultaneously extract feature and structural information from application flow data, learning more robust embedding representations assisted by self-supervised modules. Meanwhile, SCGAE is able to obtain a more discriminative self-expression coefficient matrix through the self-expression module and map it into a subspace for efficient unknown application identification.

The proposed model performs well in identifying unknown encrypted application traffic, but has three limitations. The first limitation is that temporary switching of IP addresses may affect the model's real-time performance. If the application temporarily switches some server IP addresses, there are two situations: when the user's IP address does not change, it will not affect the construction of the traffic relationship nor the model training, because the application only acts as a traffic receiver or sender; when the user's IP address changed, it will affect the construction of the traffic relationship and the model training, because the resulting relationship between IP pairs will disappear at the same time. For real-time problems, multiple tasks can be trained in parallel, and a multi-task learning model based on graph structure can be explored. The second limitation is that, as a recognition method of unknown application, the performance of SCGAE may be limited in imbalanced data. Due to the lack of prior information, when mapping massive application streams to limited imbalanced application categories, it is more likely to identify categories with larger sample sizes. Possible future work is to consider a better data balancing strategy for preprocessing, thereby further improving the unknown application recognition performance of SCGAE. The last limitation is the overall accuracy of the SCGAE. Even though the accuracy of the unsupervised method is lower than that of the supervised method, the accuracy of SCGAE still needs to be improved. Therefore, in the future work, we will try to introduce the idea of comparative learning so as to improve the accuracy of traffic identification.

Conclusion

In this study, a subspace clustering via graph auto-encoder network (SCGAE) for unknown encrypted traffic is proposed. It learns features and structural information from the initial application flow, and performs subspace clustering based on latent representation, saving the time, expense and manpower of manually labeling data. The SCGAE learns discriminative node representation from traffic data by reconstructing node attributes and graph structure, which is expressed as a coefficient matrix required for subspace clustering in the self-expression module. In addition, the SCGAE can use self-supervised strategies to improve the learning of node representation, thereby enhance clustering performance. The effectiveness of the SCGAE method is verified on a real network traffic dataset. Experimental results show that SCGAE method can effectively realize the identification of unknown traffic without label information, and outperforms the current state-of-the-art unsupervised methods.

Acknowledgements
The authors would like to thank the editor and annoymous referees for their constructive comments.

Author Contributions
RY completed the writing and experiments of the paper, AY checked and suggested revisions to the paper, and LC and DM read the paper and gave some guidance. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declaration

Competing interests
The authors declare no competing interests.

Author details
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. [2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.

Received: 27 April 2022 Accepted: 8 August 2022
Published online: 03 December 2022

References

Aceto G, Ciuonzo D, Montieri A, Pescapé A (2018) Mobile encrypted traffic classification using deep learning. In: 2018 Network traffic measurement and analysis conference (TMA). IEEE, pp 1–8
Aceto G, Ciuonzo D, Montieri A, Pescapé A (2019) Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges. IEEE Trans Netw Serv Manag 16(2):445–458
Anderson B, McGrew D (2017) 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, pp 1723–1732

Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P (2020) Structural deep clustering network. In: Proceedings of the web conference 2020, pp 1400–1410
Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: Proceedings of the AAAI conference on artificial intelligence, vol 30
Constantinou F, Mavrommatis P (2006) Identifying known and unknown peer-to-peer traffic. In: Fifth IEEE international symposium on network computing and applications (NCA'06). IEEE, pp 93–102
Conti M, Mancini LV, Spolaor R, Verde NV (2015) Can't you hear me knocking: identification of user actions on android apps via traffic analysis. In: Proceedings of the 5th ACM conference on data and application security and privacy, pp 297–304
Dainotti A, Pescape A, Claffy KC (2012) Issues and future directions in traffic classification. IEEE Netw 26(1):35–40
Erman J, Arlitt M, Mahanti A (2006) Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM workshop on mining network data, pp 281–286
Erman J, Mahanti A, Arlitt M, Williamson C (2007) Identifying and discriminating between web and peer-to-peer traffic in the network core. In: Proceedings of the 16th international conference on world wide web, pp 883–892
Finsterbusch M, Richter C, Rocha E, Muller J-A, Hanssgen K (2013) A survey of payload-based traffic classification approaches. IEEE Commun Surv Tutor 16(2):1135–1156
Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR workshop and conference proceedings, pp 249–256
Hammond DK, Vandergheynst P, Gribonval R (2011) Wavelets on graphs via spectral graph theory. Appl Comput Harmon Anal 30(2):129–150
He G, Yang M, Luo J, Gu X (2015) A novel application classification attack against tor. Concurr Comput Pract Exp 27(18):5640–5661
Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
Ji P, Zhang T, Li H, Salzmann M, Reid I (2017) Deep subspace clustering networks. arXiv:1709.02508
Jin Z, Liang Z, Wang Y, Meng W (2021) Mobile network traffic pattern classification with incomplete a priori information. Comput Commun 166:262–270
Kipf TN, Welling M (2016a) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
Kipf TN, Welling M (2016b) Variational graph auto-encoders. arXiv:1611.07308
Korczyński M, Duda A (2012) Classifying service flows in the encrypted skype traffic. In: 2012 IEEE international conference on communications (ICC). IEEE, pp 1064–1068
Korczyński M, Duda A (2014) Markov chain fingerprinting to classify encrypted traffic. In: IEEE INFOCOM 2014—IEEE conference on computer communications. IEEE, pp 781–789
Lashkari AH, Draper-Gil G, Mamun MSI, Ghorbani AA (2017) Characterization of tor traffic using time based features. In: ICISSp, pp 253–262
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
Liu J, Han J (2018) Spectral clustering. In: Data clustering. Chapman and Hall, London, pp 177–200
Liu C, He L, Xiong G, Cao Z, Li Z (2019) Fs-net: a flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019—IEEE conference on computer communications. IEEE, pp 1171–1179
Lizhi P, Hongli Z, Bo Y, Yuehui C, Tong W (2014) Traffic labeller: collecting internet traffic samples with accurate application information. China Commun 11(1):69–78
Lorbeer B, Kosareva A, Deva B, Softić D, Ruppel P, Küpper A (2018) Variations on the clustering algorithm birch. Big Data Res 11:44–53
Ma J, Levchenko K, Kreibich C, Savage S, Voelker GM (2006) Unexpected means of protocol inference. In: Proceedings of the 6th ACM SIGCOMM conference on internet measurement, pp 313–326
Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems, pp 849–856
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
Reynolds DA (2009) Gaussian mixture models. Encycl Biom 741:659–663
Sen S, Spatscheck O, Wang D (2004) Accurate, scalable in-network identification of p2p traffic using application signatures. In: Proceedings of the 13th international conference on world wide web, pp 512–521
Shbair WM, Cholez T, Francois J, Chrisment I (2014) A multi-level framework to identify https services. In: NOMS 2016-2016 IEEE/IFIP network operations and management symposium. IEEE, pp 240–248
Shen M, Wei M, Zhu L, Wang M, Li F (2016) Certificate-aware encrypted traffic classification using second-order markov chain. In: 2016 IEEE/ACM 24th international symposium on quality of service (IWQoS). IEEE, pp 240–248
Taylor VF, Spolaor R, Conti M, Martinovic I (2017) Robust smartphone app identification via encrypted network traffic analysis. IEEE Trans Inf Forensics Secur 13(1):63–78
Van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(11):2579–2605
Wang W, Sheng Y, Wang J, Zeng X, Ye X, Huang Y, Zhu M (2017) Hast-ids: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access 6:1792–1806
Wang C, Pan S, Hu R, Long G, Jiang J, Zhang C (2019) Attributed graph clustering: a deep attentional embedding approach. arXiv:1906.06532
Wu Z, Wang M, Yan C, Yue M (2017) Low-rate dos attack flows filtering based on frequency spectral analysis. China Commun 14(6):98–112
Xie G, Iliofotou M, Keralapura R, Faloutsos M, Nucci A (2012) Subflow: towards practical flow-level traffic classification. In: 2012 Proceedings IEEE INFOCOM. IEEE, pp 2541–2545
Zhao S, Zhang Y, Sang Y (2019) Towards unknown traffic identification via embeddings and deep autoencoders. In: 2019 26th international conference on telecommunications (ICT). IEEE, pp 85–89

Publisher's Note