Computer Networks 236 (2023) 109990

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

AAE-DSVDD: A one-class classification model for VPN traffic identification

Sicai Lv [a], Chao Wang [a], Zibo Wang [a], Shuo Wang [c], Bailing Wang [a,*], Yongzheng Zhang [b]

[a] School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai, China
[b] Chinese Asset Cybersecurity Technology Co., Ltd, Beijing, China
[c] School of life and Pharmaceutical Sciences, Dalian University of Technology, Dalian, China

ARTICLE INFO

Keywords:
VPN traffic identification
Deep one-class classification
Network flow feature extraction

Abstract: Problema - Cosa c'è oggi -
- Obiettivo - Parti trattate nella tesi -
- soluzione proposta

ABSTRACT

Virtual Private Network(VPN) can provide a concealed transmission channel for communication and protect the privacy of users. However, it also brings hidden dangers to cybersecurity with its wide application. Malicious behavior or harmful information can be transmitted secretly through VPN tunnels to avoid firewall censorship. Therefore, VPN traffic identification is an important part of ensure cybersecurity. Although many efforts have been made for VPN traffic identification, existing methods mainly focus on supervised learning models. In this paper, we propose an one-class classification model called AAE-DSVDD for VPN traffic identification. First, we introduce Adversarial AutoEncoder(AAE) for preliminary modeling of VPN traffic. AAE can match the aggregated posterior distribution of the hidden layer to an arbitrary prior distribution. It associates the samples with a normal distribution in the hidden space. Secondly, We implement representation learning for VPN traffic via Deep Support Vector Data Description(DSVDD). A standardized method is designed to match the output distribution of DSVDD with the aggregated posterior distribution of AAE. It alleviates the hypersphere collapse problem of DSVDD and improves identification performance. Finally, we verify the abilities of the AAE-DSVDD model on the public dataset ISCXVPN. Compared with other one-class models, AAE-DSVDD achieved the best identification ability for VPN traffic identification. It also improves the recognition ability when identifying strange classes that are not included in the training data.

1. Introduction

A VPN [1], a private network built into the public network infrastructure, that offers a secure and secret communication tunnel for users. It provide a temporary, private, and dedicated network channel in the public network to meet the needs of mutual visits in different places, and achieve point-to-point interconnection.

With the increasing importance placed on privacy by internet users, VPNs have found widespread application. However, there are also malicious uses of VPN [2,3]. Hackers can hide their malicious behavior through a VPN, making it difficult to trace the real address of the attacker. It is difficult for a firewall or intrusion detection system to detect these malicious behaviors. VPNs have acted as a protective shield against these malicious activities. Therefore, it is meaningful to detect VPN traffic for cybersecurity.

Due to its privacy and anonymity features, detecting VPN traffic presents a significant challenge to investigators. There exists multiple issues during the detection process. The first and most important problem is traffic encryption. In order to ensure the security of communication, VPNs use IPsec [4], PPTP [5] or many other protocols to build tunnels and encrypt data. In addition, there are also various anonymous technologies, such as the multi-hop anonymous communication model used by Tor [6]. Traffic obfuscation will also increase the difficulty of VPN traffic detection [7,8].

Due to the utilization of multi-hop and anonymity techniques within VPNs, traditional traffic classification methods struggle to be effective. At present, the traffic classification method based on machine learning and deep learning has attracted the attention of most researchers. In recent years, machine learning and deep learning models coexists, but with distinct emphases. Machine learning methods are usually combined with artificial analysis. By analyzing traffic flow and constructing features, traditional machine learning models can be trained to identify VPN traffic flow, such as naive Bayesian, random forest, and SVM [9,10]. However, the research on VPN traffic identification based on deep learning believes that there are obvious hidden dangers

* Corresponding author.
E-mail address: wbl@hit.edu.cn (B. Wang).

https://doi.org/10.1016/j.comnet.2023.109990
Received 15 March 2023; Received in revised form 16 August 2023; Accepted 20 August 2023
Available online 23 August 2023
1389-1286/© 2023 Published by Elsevier B.V.

in the use of artificial features during training. These features will fail when the traffic flow changes. Therefore, the output of the deep learning model is designed to use the common features extracted by some stream feature extraction tools (NetFlow, Netmate, etc.)[11], or even directly intercept the source data of the data packet for end-to-end learning [12].

However, the problem of VPN traffic identification in the real network environment is not a simple supervised learning problem. In the process of identification, the following two issues need to be considered: First of all, there are complex background traffic problems in the public network environment. Background traffic includes normal traffic, such as application traffic, file transfers, etc. And there is also some abnormal traffic, such as network attacks. It is difficult for this background traffic to be fully included in the training dataset. Second, there is the issue of feature construction. Manually designed features have limitations, which require a lot of labor costs, and can only be applied to some specific VPN traffic detection [13,14].

In view of the above problems, this paper proposes a one-class classification model, AAE-DSVDD, to identify VPN traffic flow. The reason why this paper chooses a one-class classification model is as follows: Firstly, the one-classification model only needs the training of the target class. It avoids the recognition bias brought on by the background traffic data. Next, depending on the neural network, it can automatically mine the communication features of VPN traffic flow to build a more accurate and effective detection model. This paper conducts experiments on the ISCXVPN dataset [15] to verify the model proposed in this paper. This dataset comprises seven different traffic behaviors under two types of VPN and non-VPN, including VoIP, P2P, etc. The main contributions of this paper can be summarized as follows:

1. This paper discusses the feature construction of VPN traffic flow identification. Based on the ISCXVPN dataset, this paper discussed the validity of the time-related statistical characteristics and proposed using one-dimensional Convolutional Neural Networks (1dCNN) to mine the hidden features of original packet length and time interval to identify VPN traffic flow.
2. This paper proposes an AAE-DSVDD model for VPN tunnel traffic identification. The model uses AAE to construct an aggregated posterior distribution close to the specified prior distribution in the hidden space. Then it is used to constrain the output of DSVDD to improve identification capability.
3. AAE-DSVDD alleviates the hypersphere collapse problem. Bias terms and bounded activation functions can be used in neural networks.
4. The model is demonstrated experimentally on the ISCXVPN dataset. Compared AAE-DSVDD with other one-class classification and supervised learning models to verify the effectiveness of the model. AAE-DSVDD achieved the best experimental results.

The structure of this paper is as follows: In the second part, the related work of VPN tunnel detection is analyzed. In the third part, we introduce the AAE-DSVDD model in detail. In the fourth part, the model is verified by an experiment on the ISCXVPN dataset; Finally, the sixth part summarizes the work of this paper and proposes future work.

2. Related work

In the early stages, traffic identification was mainly based on port filtering or Deep Packet Inspection (DPI)[16]. DPI technology performs traffic identification by extracting the header information of the data packet during the interaction process and matching it with the existing signature database information. However, some VPN protocols still have distinguishable figure printing in actual operation, even if some applications use obfuscation technology [17]. Some researchers can design detection methods based on the fingerprint of the VPN protocol. Xue [18] proposes three fingerprinting identification methods for openVPN: opcode-based fingerprinting, ACK-based fingerprinting, and active server fingerprinting. These methods are very effective at detecting openVPN traffic. The fingerprinting-based method can even realize application-level traffic identification such as Appsniffer [19]. But these analyses require professionals staff to analyze traffic for VPN protocols.

However, since the communication data was transmitted in an encrypted VPN tunnel, DPI technology could not be directly applied to identify this encrypted traffic flow. Therefore, machine learning or deep learning technology is widely used to learn the difference between VPN traffic and other encrypted traffic. Thus, machine learning or deep learning technology is being utilized to detect anonymous proxy tunnel traffic from typical encrypted communication. VPN traffic identification based on traditional machine learning models relies on the characteristics of manual analysis. This method of artificially formulating features shows great adaptability to traditional machine learning models: on the one hand, the designed features are usually relatively low-dimensional; on the other hand, these features are already simplified and effective, and there is no need to use neural networks for further learning. The Canadian Institute for Cybersecurity publicly released a dataset, ISCXVPN [15] that contains seven types of conventionally encrypted traffic and seven types of VPN traffic. They extracted 23-dimensional features, including flow duration and arrival time interval, etc. Gupta [20] and others proposed to use deeply enhanced Naive Bayes combined with fuzzy k-means clustering to detect VPN traffic, and the F1 score for VPN traffic recognition reached 97.3%. Based on traditional models, researchers have tried various traffic features to identify VPN traffic. Johan Mazel [10] focused on traffic meta-features, including transmission protocol, data packet quantity, duration, data packet size, and other features. Bagui [21] interested in time-dependent features. Both of them have tested a variety of traditional machine learning models implemented in Scikit-Learn, and the models used include random forests, decision trees, etc. In order to further improve the detection accuracy, the integrated learning model is also applied to VPN traffic identification [22].

Although the machine learning model has received extensive attention from researchers, the need for artificially engineered features is an unavoidable drawback. Thus, researchers gradually focused on deep learning. A neural network can automatically learn hidden features through a complex network structure, and its classification accuracy is usually higher than that of the traditional machine learning model. Shane Miller [3] proposed an identification method using a multi-layer perceptron neural network to distinguish OpenVPN tunnel traffic from normal encrypted traffic. Guo [23] converted the payload to a flow graph, and then automatically extracted image features through CNN. The overall recognition accuracy rate reached 92.2%. Fu [11] proposed a VPN traffic detection method based on Bi-directional Long-Short-Term Memory (Bi-LSTM). They believed that there are obvious shortfalls in the process of training the model, whether using manually designed features or using automatic feature extraction based on deep learning. The artificial features depended on professional experience, and the automatic extracted features required a large scale of input data and high time complexity. Therefore, they use the data information extracted by NetFlow as the input data. Tang [24] trusts that VPN traffic identification needs to combine time and space features, as proposed in Caps-LSTM. They use a capsule neural network to learn the spatial features, and a LSTM to learn the temporal features. The detection accuracy reaches 98%. Da segnare

These supervised learning models have been validated in many studies to effectively identify known class data. However, there are many unknown classes of traffic that do not belong to any train class. The supervised learning model will misclassify this traffic into a known class [25]. There are also unknown types of data in VPN traffic detection, such as unknown background traffic or uncollected VPN protocol traffic. Thus, VPN traffic identification is a semi-supervised or even unsupervised learning problem in many situations. And one-class classification models were introduced into traffic identification, which has the potential to deal with zero-day traffic [26]. The one-class classification pays attention to training a representation model of target data. It can avoid collecting complex background traffic. Van [27] proposed to detect DNS tunneling in mobile networks by One-Class Support Vector Machine (OCSVM). They discussed the performance of OCSVM under different kernel functions. The experimental results show that OCSVM with the RBF kernel achieved the best performance, and it has a very high recall rate. Nadler [28] use isolation Forest (iForest) to detect DNS tunnels. In their research, the iForest is trained on normal traffic. Esteban Rivera [29] detects VPN tunnels by profiling network latencies. In their work, an OCSVM and an one-shot Siamese neural network are designed to detect VPN tunnels. OCSVM achieves better detection performance than the neural network.

One-class machine learning models such as OCSVM and iForest have good interpretability, but these shallow methods cannot handle complex and diverse data, so they are usually bound with feature engineering. Therefore, researchers pay attention to deep learning. Autoencoder(AE) is an unsupervised method for data compression and data feature representation. Ons Aouedi [30] proposed a semi-supervised learning model based on stacked sparse autoencoders, which achieved an accuracy rate of 95% on the ISCXVPN dataset. However, their work only used the feature extraction ability of the AE, and added a softmax layer after the decoder to build a classifier. AE can also be used as an anomaly detection model, and Wu [31] used this feature of AE to detect DNS tunnels.

In a recent study, researchers attempted to combine shadow one-class classification models with neural networks. Based on OCSVM and SVDD, researchers proposed One-Class Neural Network(OC-NN)[32] and DSVDD [33] by combining them with neural network models. It retains the theoretical basis of the traditional model and can deal with complex and high-dimensional data. However, these models are currently mainly used in traffic anomaly detection [34,35].

3. Method

3.1. Motivation

The purpose of VPN traffic identification is to mine VPN tunnel traffic from a large amount of background traffic. Therefore, it is usually treated as a supervised learning problem. By constructing two types of labeled datasets of VPN traffic and background traffic, and training a binary classification model to realize VPN traffic identification. However, the binary classification model may face the following difficulties during identification. Assuming that Fig. 1(a) is the training dataset, there is a bias in the training data collection. VPN traffic mainly collects web browsing data, and background traffic includes web browsing and VoIP. Other communication behaviors are ignored during data collection, such as P2P, FTP, etc. The possibility of a binary classification model trained on these labeled data may be as shown in Fig. 1(b). When some Unseen samples from Known Classes (UKCs) are used for testing, the model has perfect recognition ability. In the application scenario, the input of the model may contain some Unseen samples from Unknown Classes (UUCs), such as FTP and VPN-VoIP. Then it may appear as shown in Fig. 1(c), and the model cannot effectively detect UUCs. In VPN traffic identification, UUCs may be some rare VPN traffic, or even some network attack traffic. Thus, this paper designs a VPN traffic detection model based on DSVDD. It is expected to get a decision surface like that shown in Fig. 1(d), which will have a stronger capability to identify UUCs.

Meanwhile, DSVDD is optimized in this paper to improve its identification ability. DSVDD only shrinks the hypersphere based on the SVDD model, and it does not consider the distribution of training data inside the hypersphere. This paper makes the following assumption to achieve a better detection capability:

Table 1
Symbols and definitions used in this paper.

Symbols	Definitions
$p(z)$	The predefine prior distribution of AAE
$q(z)$	The aggregated posterior distribution of AAE
$p_{\phi(x)}(z)$	The distribution of DSVDD output
X	The train dataset
N	The size of train dataset
$s(x)$	The Anomaly score of AAE_DSVDD
c, R	The center and radius of hypersphere
α	The weight of loss
v	The upper bound on the fraction of outliers
$L_{distance}, L_{DSVDD}$	The distribution constrain loss and original DSVDD loss
μ, σ	The mean and standard deviation of prior distribution
w^{ℓ}	The weights of the ℓ layer in neural network
$\phi(x, W)$	The function representation of DSVDD

Assumption 1. DSVDD can achieve better performance by mapping the most samples of the target class with high confidence to the hypersphere center and a small number of target class samples with low confidence to the hypersphere edge.

Based on the above assumption, this paper designs the AAE-DSVDD model. It can constrain the DSVDD output through the aggregated posterior distribution of AAE. Then it achieves the purpose of optimizing the internal distribution of the hypersphere. The symbols and definitions used in this paper are shown in Table 1.

3.2. Deep SVDD

SVDD [36] is a shadow one-class classification model. It is assumed that all training samples from the target class, $X = \{x_i \in R^d, y_i = 1 | i = 1, 2, \dots, n\}$. d denotes the dimension of features. There exists a hidden mapping Φ. Φ can map all samples to the feature space, $\Phi : R^d \to \mathbf{F}$. The goal of SVDD is to find the smallest hypersphere that contains all training samples in the feature space. In order to construct such a hypersphere, SVDD need to solve the following optimization problems:

$$\begin{cases} \min\limits_{c,R,\xi} R^2 + C \sum\limits_{i=1}^{n} \xi_i \\ \left\| \Phi\left(x_i\right) - c \right\|^2 \le R^2 + \xi_i, \xi_i \ge 0, \forall i = 1, 2, \dots, n \end{cases} \tag{1}$$

The principle of DSVDD is the same as SVDD, and the purpose is to find a minimum hypersphere in the feature space. But it achieves its purpose through a neural network. Write as $\phi : R^d \to \mathbf{F}$. The neural network ϕ is also constrained by the objective function of SVDD. The object function of DSVDD is as follows:

$$\begin{aligned} \min_{R,W} R^2 &+ \frac{\lambda}{2} \sum_{\ell=1}^{L} \left\| w^{\ell} \right\|_F^2 \\ &+ C \sum_{i=1}^{n} \max\left\{0, \left\| \phi\left(x_i; W\right) - c \right\|^2 - R^2 \right\} \end{aligned} \tag{2}$$

In Formula (2), L denotes the number of layers in neural network, w^{ℓ} denotes Weight of the ℓth layer of neural network, $W = \{w^{\ell}, \ell = 1, 2, \dots, L\}$. The above objective function can be solved by back-propagation, and the anomaly fraction is defined as the distance between $\phi(x_i)$ in the feature space and the center of the sphere c.

DSVDD uses a neural network to construct a map from sample space to feature space, which makes SVDD take the advantages of the neural networks. On the one hand, it can accept high-dimensional input data, and it can rely on neural networks for automated feature extraction and mining of hidden features; on the other hand, the theoretical basis of the original SVDD is maintained, and the hypersphere constructed is the decision surface. It has good interpretability and is easy to calculate the anomaly score.

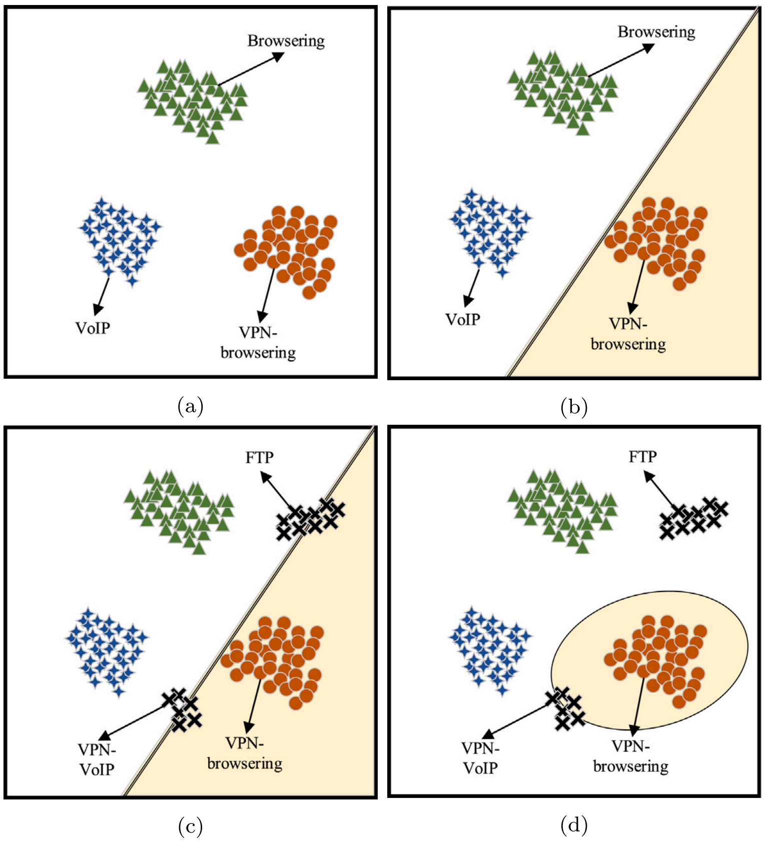

Fig. 1. An example of supervised and semi-supervised learning, (a) denotes the labeled training data. (b) denotes the decision surface of the supervised learning model. (c) is the scene where the supervised learning model recognizes UUCs. (d) is the expected one-class classification model.

However, there is a defect named hypersphere collapse during training the neural networks. Hypersphere collapse means that the radius of the hypersphere shrinks to zero. The map ϕ is a constant function. Therefore, the following settings need to be made during the training process [33]: (1) Fix the center of the sphere when the model is initialized; (2) Set the neural network to have no bias term; (3) Set an unbounded activation function.

3.3. Adversarial autoencoder

AAE [37] is a kind of generation model, that makes up for the defect of Generator Adversarial Network(GAN). The generator input of GAN is random noise and has no semantic information at all. But the hidden vector is matched with a prior distribution to control the generation of data in AAE. AAE includes three schemes: unsupervised, semi-supervised and supervised. This article focuses on the unsupervised model.

The structure of AAE is shown in Fig. 2. The model consists of three parts:Encoder, Decoder and Discriminator. The Encoder builds an aggregated posterior distribution $q(z)$ in Formula (3). Let x be the input and $E(x) = z$ be the latent vector, and there is $z \sim q(z)$. The decoder, which is the generator of the AAE, can decode z to obtain the reconstructed sample x.

$$q(z) = \int_X q(z|x)p(x)dx \tag{3}$$

During the training process, the Encoder and Decoder engage in an adversarial procedure. The purpose of the encoder is to generate an aggregated posterior distribution that can fool the discriminator of the confrontation network. The discriminator is to distinguish whether the input z is from real data $q(z)$ or fake data $p(z)$, as Formula (4).

$$D(\mathbf{z}) := \begin{cases} 1 & \text{if } z \text{ is real, i.e.,} z \sim p(z) \\ 0 & \text{if } z \text{ is generated, i.e.,} z \sim q(z) \end{cases} \tag{4}$$

AAE can match the aggregated posterior distribution $q(z)$ to the prior distribution $p(z)$. In this paper, the $q(z)$ is then used to constrain the output of DSVDD. Another purpose of using AAE is to utilize the AE trained in AAE. AE can be used as an anomaly detection model to distinguish samples with high reconstruction loss as anomalies. These samples will be considered low confidence samples and distributed at the edge of the prior distribution $p(z)$, which is in line with the assumptions of this paper.

3.4. AAE-DSVDD

In this part, we introduce the AAE-DSVDD model designed in this paper in detail. The structure of the AAE-DSVDD model is shown in Fig. 3.

The key idea of the AAE-DSVDD model is as follows: First, train an AAE model to get the encoding distribution. Let the aggregated posterior distribution $q(z)$ close to the prior distribution $p(z)$. Then, the outputs of DSVDD are then constraint by $q(z)$. Through these two steps, we can achieve the purpose of Assumption 1. Finally, AAE-DSVDD reaches its goal of improving the identification performance of the DSVDD model.

The first stage is the training of an AAE model. In this stage, the prior distribution $p(z)$ is set to be a normal distribution. Then it is necessary to specify the parameters of the normal distribution of the target, including the mean u and standard deviation σ. Through AAE training, the data distribution of the Encoder output is close to the normal distribution, at this time $z \sim N(\mu, \sigma)$.

The second stage is the optimized DSVDD model. The purpose of the second stage is to add distribution constraints to DSVDD. Therefore, as shown in Fig. 3, the optimized DSVDD will compare its output with the output of the Encoder. This constrains the output distribution of

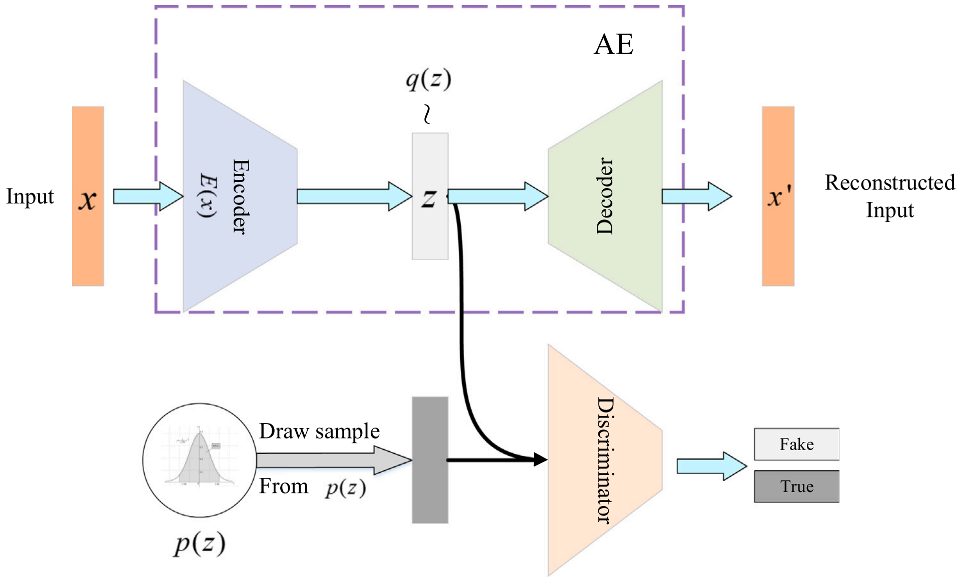

Fig. 2. The network structure of AAE. The Encoder maps the training samples to the latent vector z, $z \sim q(z)$. Discriminator is used to distinguish z and samples sampled from $p(z)$.

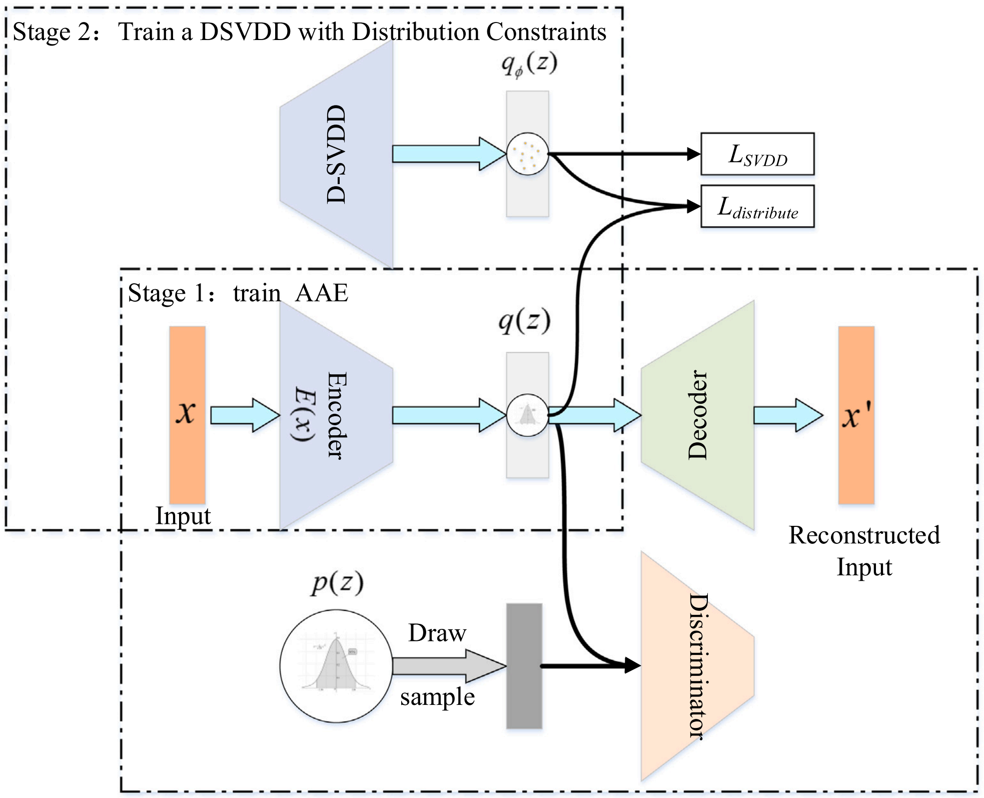

Fig. 3. The network structure of AAE-DSVDD. It contains two stages: (1) Train an AAE. This stage constructs the mapping of input samples to the aggregated posterior distribution $q(z)$ through AAE. (2) Train a DSVDD with distribution constraints. This stage constrains the output of DSVDD through L_{DSVDD} and $L_{distribute}$.

DSVDD $q_{\phi(x)}(z)$ by the aggregated posterior distribution $p(z)$ to achieve the purpose of restricting the output distribution of DSVDD.

As noted above, the objective function of the second stage of AAE-DSVDD is as follows:

$$\min_{R,W} R^2 + \frac{1}{\upsilon n}\sum_{i=1}^{n}\max\left\{0, \left\|\phi\left(x_i;W\right)-c\right\|^2 - R^2\right\} + \alpha L_{distribute}(\phi\left(x_i;W\right), E(x_i)) + \frac{\lambda}{2}\sum_{\ell=1}^{L}\left\|w^{\ell}\right\|_F^2 \tag{5}$$

Among them, α is the weight parameter, which is used to adjust the weight of distribution constraints. υ is an upper bound on the fraction of outliers and a lower bound on the fraction of samples being outside or on the boundary of the hypersphere, $\upsilon \in (0, 1]$.

AAE-DSVDD achieves the following goals by constraining the distribution of outputs: Firstly, it optimized the output distribution of DSVDD. It uses the aggregated posterior distribution of AAE to constrain the output of DSVDD. It also utilizes the anomaly detection ability of AE in AAE to treat samples with small reconstruction errors as high-confidence samples. Next, it alleviated the hypersphere collapse problem in DSVDD. This will be demonstrated in the following part of this paper.

Loss Function. The training for AAE-DSVDD is divided into two stages. The first stage is the training of the AAE model. Its training process is the same as the unsupervised AAE [37]. In the second stage of AAE-DSVDD, the loss consists of two parts: the original DSVDD loss and the measurement of the distribution difference between $q(z)$ and $q_{\phi(x)}(z)$. Therefore, the loss function of the DSVDD model is set as follows:

$$L' = \alpha L_{DSVDD} + (1-\alpha)L_{distribute} \tag{6}$$

Among them, α is the weight parameter, which is used to control the weight of the two-part loss function. L_{DSVDD} is the original DSVDD loss, as shown in Formula (7). υ is an upper bound on the fraction of outliers, $\upsilon \in (0, 1]$. n is the sample size of the training dataset. $\phi(x_i, W)$ is the function representation of DSVDD, and W is the weights of the neural network.

$$L_{DSVDD} = \frac{1}{\upsilon n}\sum_{i=1}^{n}\max\left\{0, \left\|\phi\left(x_i;W\right)-c\right\|^2 - R^2\right\} \tag{7}$$

$L_{distribute}$ is the measurement of the distribution difference between the aggregated posterior distribution $q(z)$ and the DSVDD output distribution $q_{\phi}(z)$. Next, the calculation process of $L_{distribute}$ will be introduced in detail in Section 3.5.

AAE-DSVDD uses the trained DSVDD model to make predictions. It divides the samples inside the hypersphere into target classes, and the samples outside the hypersphere are considered anomalies. Thus, the anomaly score is defined as follows:

$$s(x) = \left\|\phi\left(x;w^*\right)-c\right\|^2 \tag{8}$$

During the training process of AAE-DSVDD, a predefined prior probability distribution $p(z)$ is not directly used to constrain the distribution of DSVDD output. Instead, AAE is utilized to construct an aggregated posterior distribution $q(z)$ that approximates $p(z)$. This process is based on the following considerations: First of all, in order to ensure the convergence of the model. If we use $p(z)$ directly, we need to sample from it as the targets of DSVDD's output, and then calculate the KL divergence of $q_{\phi}(z)$ and $p(z)$. If we sample randomly from $p(z)$, it will result in different targets for each epoch. It may affect the performance of the model, and even cause it to fail to converge. Furthermore, if we fixedly sample from $p(z)$, we will face the following problem: It is hard to determine whether a sample is an anomaly before training. A fixed $p(z)$ should be the purpose of DSVDD training. Another reason is to ensure that the output satisfies the assumptions. It is difficult to construct a prior distribution $p(z)$ that satisfies the assumptions. The purpose of the distribution constraint is to concentrate samples with

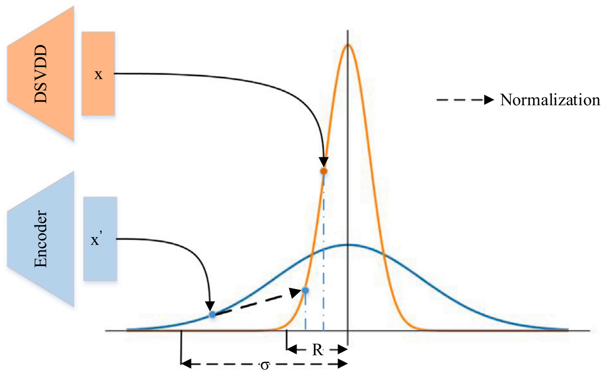

Fig. 4. Rebuild features based on original traffic. The feature is composed of the load length of each packet and the time interval between two packets.

high confidence near the center of the hypersphere and scatter samples with low confidence toward the edge of the hypersphere. AAE-DSVDD can treat samples with high reconstruction loss as outliers and diverge to the edge of the hypersphere. It fulfills the assumption of this paper.

In summary, through the posterior aggregation distribution $q(z)$ of AAE, the relative position of each training sample in the prior distribution $q(z)$ can be preliminary determined. And it ensure the convergence of DSVDD training and the validity of constraints. AAE-DSVDD constructs an aggregated posterior distribution with a fixed output via AAE and identifies anomalies via the hypersphere shrinkage of DSVDD.

3.5. Normalization

All activation functions in the AAE-DSVDD model are unbounded. Consequently, there will be a problem with scale inconsistency. The radius of the hypersphere will vary during the DSVDD training procedure, whereas the normal distribution in the AAE hidden space is fixed after the initial stage. Therefore, if compared $E(x)$ and $\phi(x, W)$ directly, distribution constraints may not be met and model training may fail to converge.

In this paper, we design a normalization method to solve the problem of measurement. The main purpose of normalization is to unify measurements. As shown in Fig. 4, assume that the mean value of the prior probability $p(z)$ is 0, and the standard deviation is 1. There is $E(x) \sim N(0, 1)$ after the first stage. However, during DSVDD training, the radius may decrease to 0.01 or much less. At this time, it would be meaningless to directly calculate the distance between $E(x)$ and $\phi(x, W)$. It need to be normalized, as shown by the dotted line in Fig. 4. The normalization method proposed in this paper is shown in the Formula (9).

$$L_{distribute} = L(\frac{E(x)-\mu}{\eta\sigma}, \frac{DSVDD(x)-c}{R}) \tag{9}$$

where η is a parameter to control the upper bound on the fraction of outliers in prior distribution $p(z)$. μ and σ are the mean and standard deviation of the prior distribution $p(z)$. c and R are the center and radius of the hypersphere, respectively.

The concept of normalization is the same as that of z-score normalization, which standardizes by mean and standard deviation. But, the normalization method proposed in this paper is slightly adjusted. In the training process of DSVDD, υ controls the upper bound on the fraction of outliers and a lower bound on the fraction of samples that are outside or on the boundary of the hypersphere. Therefore, the normalization method uses the parameter *eta* to control the upper bound on the fraction of outliers in the normal distribution. According to the principle of $3-\sigma$, for a normal distribution $N(\mu,\sigma)$, there are 95.44% samples falling in the interval $(\mu-2\sigma,\mu+2\sigma)$, and 99.74% samples falling in the interval $(\mu-3\sigma,\mu+3\sigma)$. Therefore, if v is set at 0.01, we can approximately set $\eta=3$. If v is set at 0.05, we can approximately set $\eta=2$.

Finally, we use the mean squared error as the loss function. The loss calculation is shown in Formula (10).

$$L_{distribute}=\left\|\frac{E(x)-\mu}{\eta\sigma},\frac{DSVDD(x)-c}{R}\right\|^2 \tag{10}$$

3.6. Hypersphere collapse

Hypersphere collapse is a serious problem during DSVDD training. It is defined as the radius of the hypersphere reducing to 0, and DSVDD becomes a trivial solution, $\phi(x)=c$. In this paper, the distribution constraint can effectively alleviate the problem of hypersphere collapse.

To prove it, we calculate the partial derivative of the objective function, as shown in Formula (11) and (12).

$$\frac{\partial f}{\partial W}=C\sum_{i=1}^{n}\xi(x_i)+\alpha\sum_{i=1}^{n}2*\frac{\partial\phi}{\partial W}*\left\|\phi\left(x_i;W\right)-E(x_i)\right\| \tag{11}$$

$$\xi(x_i)=\begin{cases}0 & \text{if } x_i \text{ outside the hypersphere}\\ 2*\frac{\partial\phi}{\partial W}*\left\|\phi\left(x_i;W\right)-c\right\| & \text{if } x_i \text{ inside the hypersphere}\end{cases} \tag{12}$$

If the neural network training reaches the all-zero weight solution. Substitute $R^*=0$, $\phi(x;W^*)=c$ into the Formula (11). Then we get:

$$\frac{\partial f}{\partial W}=\alpha\sum_{i=1}^{n}\eta*\frac{\partial\phi}{\partial W}*\left\|c-E(x_i)\right\| \tag{13}$$

There, $\frac{\partial\phi}{\partial W}=0$. Thus, the gradient of the objective function at the trivial solution equals zero. Therefore, AAE-DSVDD does not change the fact that all-zero weight is a local optimal solution.

However, when the neural network is close to zero weight $\left\|\phi\left(x_i;W'\right)-c\right\|<\epsilon$. For samples outside the hypersphere, gradients only come from distributional constraints to adjust the positions of samples within the hypersphere. For these samples outside the hypersphere, we have:

$$\frac{\partial f}{\partial W}\le\alpha\sum_{i=1}^{n}2*\frac{\partial\phi}{\partial W}*\left\|\phi\left(x_i;W'\right)-E(x_i)\right\| + C\sum_{i=1}^{n}\max\left\{0,2\epsilon*\frac{\partial\phi}{\partial W}\right\} \tag{14}$$

The first term on the left is significantly larger than the second term, and it dominates the gradient direction. In the first term, $\left\|\phi\left(x_i;w'\right)-E(x_i)\right\|$ is not zero when $\phi(x;w)=c$. The neural network will not converge on the trivial solution.

When the model is near to the trivial solution $phi(x)=c$, the distribution constraints will predominate the gradient direction. Therefore, it alleviated the DSVDD hypersphere collapse issue. AAE-DSVDD relaxes the model configuration constraints. AAE-DSVDD is capable of employing bias terms and bounded activation functions.

4. Data preparation

The ISCXVPN [15] dataset is used for experimentation. This dataset is a network security data set published by the Canadian Institute for Cybersecurity. It includes 14 types of traffic, including VoIP, VPN-VoIP, P2P, VPN-P2P, etc. Meanwhile, a labeled dataset with extracted features is provided for training. The features provided by the original ISCXVPN dataset include: duration, forward inter arrival time (Fiat), backward inter arrival time (Biat), flow inter arrival time (Flowiat), etc.

The experiment revealed that it is challenging to distinguish between VPN and non-VPN traffic data using the features extracted by ISCXVPN. We analyzed some characteristics of these two types of data, and the results are shown in Fig. 5. It shows the Cumulative Distribution Function (CDF) of some features in the VPN and non-VPN after z-score normalization. The CDF is the integral of the probability density function, which can completely describe the probability distribution of a random variable. The x-axis represents the value of the features, and the y-axis represents the probability that the feature is less than its corresponding x-axis value. The figure shows the CDF of statistical features in the original ISXCVPN dataset, including duration, FPsec, FBsec, etc. These features of the distribution of VPN and non-VPN have the following characteristics: First, the distribution is concentrated. Among the features shown in the figure, almost all have a sharp increase in probability, which means that the value of the feature is concentrated in a small range. This will result in less variance between samples. Secondly, the similarity of the distribution is high, such as mean_fiat, min_fiat and other features. The CDF function curves of VPN and non-VPN almost coincide. The emergence of these features also shows that VPN and non-VPN traffic have great similarity in these statistical features, which will affect the detection performance of one-class classification model.

We consider that feature engineering is the reason why the machine learning model is limited because the feature set needs to be manually designed. Taking the ISCXVPN data set as an example, the flow statistical features need to be designed manually and need a feature processing tool named ISCXFlowMeter to process them. However, in the feature analysis of this paper, some features of VPN and non-VPN are extremely similar. Secondly, we believe that using statistical features to describe the overall properties of the flow will lose some communication details.

Therefore, this paper directly uses the packet length and time interval sequence without any processing. It utilized the automatic feature extraction capability of the neural network to train the model to achieve VPN traffic detection.

This paper reconstructs features based on the PCAP file in the ISCXVPN dataset. From the previous analysis, there are differences in the arrival times of VPN and non-VPN traffic. Therefore, the features designed in this paper are shown in Fig. 6. The payload length and its time interval are used as the feature sequence. Then, we build a neural network model for automatic feature extraction.

The following techniques are used to preprocess the PCAP file:

Flow reorganization. The PCAP file contains a wide range of traffic types, such as Chat, Streaming, VoIP. In order to achieve flow-level traffic identification, this paper uses Tshark and Python to reorganize the flow in PCAP. In this paper, the flow reorganization is based on the 4-tuple, and a flow is marked with source IP, source port, destination IP, and destination port. The processing method is to extract the 4-tuple, packet length, and arrival time through Tshark and then reorganize them through Python.

Flow de-noising. During the experiment, we noticed that there exists a large number of short streams (less than 5 packets) in the provided PCAP package. These short streams are usually simple requests, or truncated streams. In this paper, these flows are treated as noise data. We discarded all flows with fewer than 5 packets. Cause, some short flows, such as DNS query, do not match the behavior type marked by the PCAP file. For streams whose length is greater than 5 but less than the window size, we padded with zeros during processing.

Flow split. As shown in Fig. 6, the packet window is used to split the flow, and the length of the packet window is set to 30 in this paper. Under this window, we split the long flow of the VPN tunnel. And extract the payload length of data packets in the window and the sequence of their time intervals as samples.

A new ISCXVPN dataset is reconstructed through the above techniques. The information on the final dataset is shown in Table 2. First of all, in order to balance the data, we sample in the extracted data set to ensure the ratio of sample sizes of different categories is not greater than 2. The ISCXVPN data set provides seven types of traffic, but only five of them are selected for feature extraction in this paper.

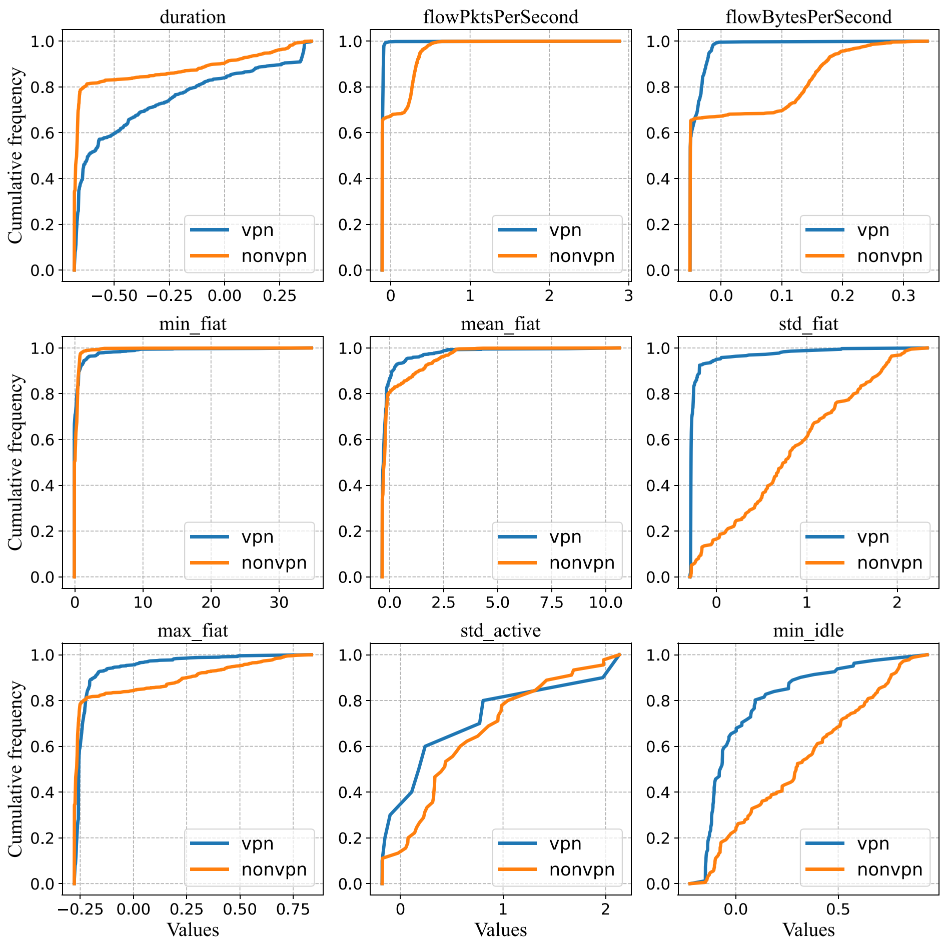

Fig. 5. The cumulative distribution function of some features in the original ISCXVPN dataset.

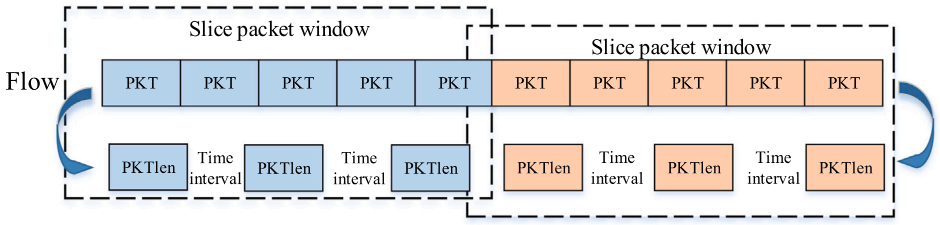

Fig. 6. Rebuild features based on original traffic. The feature is composed of the load length of each packet and the time interval between two packets.

The two discarded classes are P2P and web browsing. The reason for abandoning P2P is that the PCAP files marked as bittorrent only appear in the VPN type. We dropped web browsing because it occurs in each type of traffic and it is difficult for us to separate it. For instance, when they captured voice calls using Hangouts, even though browsing is not the main activity, browsing flows were still captured.

Table 3 shows the difference between the feature set constructed in this paper and the original ISCXVPN feature set. The first is the type of detection. The original ISCXVPN builds features based on the flow and uses the flow as the smallest detection unit. However, in this paper, a sliding window is used to segment the flow, and the fragmentation within the window is used as the smallest detection unit. In terms of data processing, this paper analyzes the PCAP data packets of ISCXVPN, and filters some of the noise data through the packet length threshold, such as Domain Name Service(DNS) query and refused connections.

Table 2
ISCX dataset reconstructed features by packet length and time interval sequences.

	VPN	NonVPN	Dim	Included PCAP
VoIP	48858	97716	59	vpn_voipbuster1a.pcap, vpn_voipbuster1b.pcap,et.al
Chat	2915	1625	59	vpn_aim_chat1a.pcap, vpn_facebook_chat1a.pcap, et.al
Streaming	32644	65288	59	vpn_youtube_A.pcap, vpn_netflix_A.pcap,et.al
Email	474	948	59	vpn_email_2a.pcap, vpn_email_2b.pcap,et.al
FTP	6728	11934	59	vpn_ftp_A.pcap, vpn_sftp_A.pcap,et.al

Table 3
The comparison between original ISCXVPN and new ISCXVPN in this paper.

Dataset	Comparison
Original ISCXVPN	*Minimum detection unit*: flow. *Feature set*: Traffic flow statistic features. *Data processing*: Reorganized flow by 4-tuple, and set the flow timeout to 15, 30, 60, 120s. *Sample example*: $[duration, max\_fiat, min\_fiat, \ldots]$
New ISCXVPN	*Minimum detection unit*: Slide packet window. *Feature set*: PKTlen and IAT sequence. *Data processing*: Reorganized flow by 4-tuple, Split flow by slide packet window, and set the minimum number of packets to filter noisy streams, such as DNS query and refused connections. *Sample example*: $[PKTlen, IAT, PKTlen, IAT, \ldots]$

5. Experiment

5.1. Baseline

In the experiments in this paper, we simultaneously explore the performance of supervised learning models and one-class classification in VPN traffic detection. We selected some representative supervised learning models and one-class classification models as models for comparison.

Supervised learning model. We divide the supervised learning model into two categories: machine learning and deep learning. In the machine learning model, we used Support Vector Machines(SVM), Random Forests, K-Nearest Neighbors(KNN), Gaussian Naive Bayes(GNB)[10]. We also utilized two ensemble learning models, Random Forest [21] and Adaboost [10]. For the deep learning model, we used a fully connected deep neural network [9] and a CNN model with the same network structure as the DSVDD model in this paper as a control experiment.

One-class classification model. In order to verify the performance of AAE-DSVDD, this paper selects several of the most well-known one-class classification models as baselines. These models are as follows: First of all, OCSVM [38] and iForest [39]. OCSVM and iForest are the two most well-known one-class models in traditional machine learning. The second one is AE [40]. AE assumes the reconstructed sample will be more different from the original sample when facing anomalies. The third one is GANomaly [41]. Ganomaly is an anomaly detection model based on generative adversarial networks. And it has been verified to have good anomaly detection abilities. Finally DSVDD [33] combines the SVDD model with the neural network. It has achieved outstanding classification performance.

5.2. Neural network setting

The purpose of AAE-DSVDD is to constrain the output distribution of DSVDD through the aggregated posterior distribution of AAE. Thus, the dimensions of DSVDD outputs are the same as those of the encoder in AAE. In order to simplify the design, the Encoder is designed to have the same network structure as DSVDD, and the parameters of each layer of the neural network are shown in Fig. 7.

In this paper, we directly use the packet length and inter-arrival time(IAT) sequence $[PKTlen, IAT, PKTlen, \ldots]$ as the input of the neural network. We set the first layer of the neural network as a 1dCNN layer. We set its kernel_size as 3 and stride as 2, which can ensure that the content processed by each convolution kernel is two data packets and their time intervals, as shown in Fig. 7. And it is expected to mine out the deeper differences between VPN and non-VPN. After the convolutional layer, we set up three fully connected layers to mine the overall characteristics of the packet window. The final network structure is shown in Table 4.

Table 4
The layer setting in neural network.

Layer	Setting
Conv1d	Inchannel=1,out_channel=1,kernel_size=3, stride=2,active_func=LeakyReLU
Flatten	–
Linear	Input_dim=29, output_dim=512,active=LeakyReLU
Linear	Input_dim=512, output_dim=512,active=LeakyReLU
Linear	Input_dim=512, output_dim=64

Table 5
Hyperparameter settings in of the compared model in experiment.

Model	Hyperparameter	Value
AAE-DSVDD	υ	0.05
	kernel_size,stride	3,2
	batch_size	64
	epochs	200
	weight_decay	1e–6
	α	0.9
	η	1e–6
	μ, σ	1.0, 0.5
DSVDD	υ	0.05
	weight_decay	1e–6
GANomaly	w_{enc}, w_{rec}, w_{adv}	1,10,1
OCSVM	gamma	0.1
	υ	0.05
iForest	N_estimator	100

The Decoder in AAE is the reverse of the network in the above table. In particular, Conv1d is replaced by ConvTranspose1d in the Decoder to reverse the convolution. The activation function in the model is LeakyReLU. In addition, it needs to set bias=False for each of the above layers when training DSVDD.

In this paper, the machine learning model is trained through scikit-learn[1] and the neural network is built through PyTorch.[2] Some parameter settings are shown in Table 5. First, the epochs is set to 200. During the experiment, we observed that the losses of models such as DSVDD, AAE-DSVDD, and GANomaly all reached convergence after 200 epochs. Second, the batch size of all neural networks is set to 64. Due to the small amount of data in email and chat, we set batchsize to 64 to avoid having only one batch in a epoch. Third, the kernel size and stride in 1dCNN are set to 3 and 2, respectively. This can ensure each convolution kernel corresponds to two packets and their time interval, as shown in the Fig. 7. Finally, μ and σ are set to 1.0 and 0.5 through grid search experiments, see sensitivity analysis later in this paper. Parameters not mentioned in the table use the default parameters in scikit-learn and pytorch.

In addition, there is a trick to initialize the center of the hypersphere. In AAE-DSVDD, the purpose is to make the output distribution of DSVDD close to $N(\mu, \sigma)$. Thus, this paper sets the center initialization equal to u to ensure the rapid convergence of training.

5.3. Evaluate metrics

Confusion matrices are a common representation of evaluation metrics for classification issues. Each column in the confusion matrix represents the ground truth of the predicted class, and each row represents the type of model prediction. The elements in the confusion matrix are the corresponding number of samples, as shown in the Table 6.

[1] https://scikit-learn.org/
[2] https://pytorch.org/

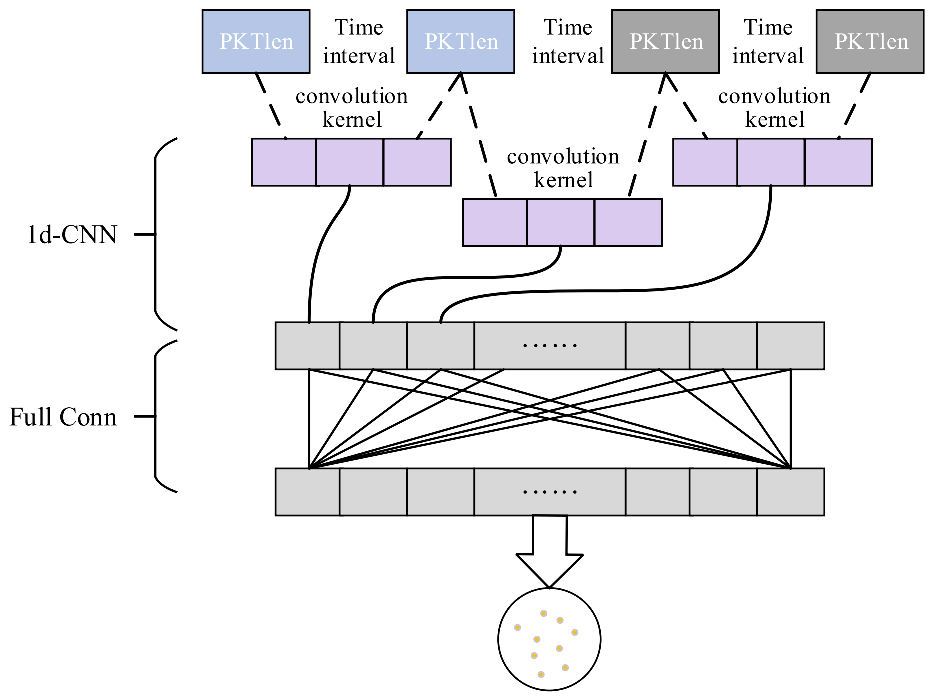

Fig. 7. DSVDD network structure used in the experiment.

Table 6
Confusion matrix.

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

The evaluation metrics of classification models can be derived from the confusion matrix, including:

True Positive Rate(TPR), also known as Recall. It refers to the proportion of positive samples detected by the classifier among all positive samples.

$$TPR = \frac{TP}{TP + FN} \tag{15}$$

False Positive Rate(FPR). It refers to the proportion of samples that the model incorrectly predicts to be positive among all samples that are actually negative.

$$FPR = \frac{FP}{FP + TN} \tag{16}$$

Precision. It refers to the proportion of true labels that are positive among all samples that are detected as positive.

$$precision = \frac{TP}{TP + \mathrm{FP}} \tag{17}$$

A ROC curve is a graphical tool used to represent the performance of a classification model. It depicts the performance of the classifier at different thresholds by using FPR and TPR as the horizontal and vertical coordinates.

The AUC value is the area under the ROC curve, which can also be used to evaluate the performance of a classifier. The performance is superior the greater the AUC value. When the AUC equals 0.5, the model closely resembles a random estimate. When the AUC is less than 0.5, the model has not been able to withstand testing. Even though the data is classified, the opposite results are obtained.

In this article, we analyze and compare the performance of one-class classification models. These models output an anomaly score when making predictions, and then use a threshold to judge whether the sample belongs to the target class. For example, in DSVDD, the anomaly score is the distance from the output of the neural network to the center of the hypersphere, and the threshold is the radius. The AUC value can fully demonstrate the true performance of the model. Meanwhile, in order to demonstrate the recognition ability of different classification models under a predefined threshold, we used the precision and recall of the target class to describe the detection performance under the fixed threshold for comparison.

5.4. Comparisons experiment

In this section, we use the original feature set and the reconstructed feature set to test the model. The results are taken as the average of five independent and repeated experiments.

As shown in the Table 3, four different data sets of 15s, 30s, 60s, and 120s were constructed by setting different timeout in Original ISCXVPN. The experimental results of the original ISCXVPN dataset are shown in Table 7. The recognition performance is unqualified on all one-class classification models. The maximum AUC for identifying VPN and non-VPN traffic is only 0.58, which comes from the AAE-DSVDD model in the 15s scenario. The minimum is only 0.43, which comes from the iForest model in the 15s scenario. The AUC value of all models fluctuates around 0.5. This means the one-class classification model has only weak recognition abilities in the feature set provided by ISCXVPN. The experimental results verify the analysis of the features in Section 4.1. There are many samples that have a very high similarity between VPN and Non-VPN. However, there is a lack of negative examples during the training of the one-class classification model. Those highly similar samples will be classified into one category.

From the results of each scenario, the AAE-DSVDD model proposed in this paper has achieved the best recognition performance in the three scenarios of the 15s, 30s, and 60s. and the AUC value has increased by 1%–3%, which also verifies that the AAE-DSVDD model has stronger representational learning ability than DSVDD, AE, and other one-class classification models.

Based on the features constructed in this paper, we conducted five binary classification experiments to observe the feature extraction and model recognition capabilities of the neural network under different behaviors. In this part of the experiment, we use a certain type of

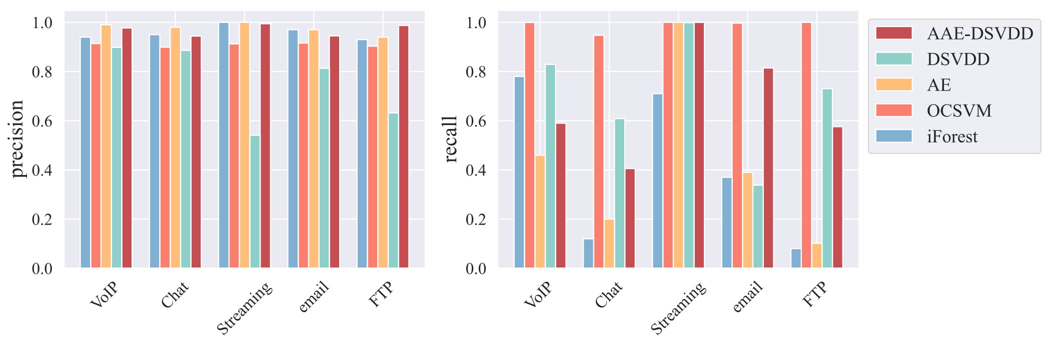

Fig. 8. The precision and recall of one-class classification models on different traffic types.

Table 7
The AUC value of each model under the original ISCXVPN dataset.

	15 s	30 s	60 s	120 s
OCSVM	0.5067 ± 0.0079	0.5006 ± 0.0061	0.4908 ± 0.0071	0.5457 ± 0.0752
iForest	0.4306 ± 0.0089	0.4682 ± 0.0088	0.4982 ± 0.0168	**0.5717 ± 0.0062**
AE	0.4915 ± 0.0149	0.4994 ± 0.0061	0.4968 ± 0.0049	0.548 ± 0.0041
GANomaly	0.5095 ± 0.0521	0.5276 ± 0.0286	0.5184 ± 0.0737	0.5574 ± 0.0161
DSVDD	0.566 ± 0.02688	0.5277 ± 0.0393	0.5026 ± 0.0185	0.5629 ± 0.0133
AAE-DSVDD	**0.5835 ± 0.0197**	**0.5314 ± 0.0365**	**0.5325 ± 0.0276**	0.5373 ± 0.0275

Table 8
The AUC value of each model under the new ISCXVPN dataset.

	VoIP	Chat	Streaming	Email	FTP	wtd. AUC
OCSVM	0.7755 ± 0.0016	**0.7971 ± 0.0122**	0.9880 ± 0.0005	0.8555 ± 0.0027	0.8637 ± 0.0044	0.8584
iForest	0.7731 ± 0.0085	0.5425 ± 0.0140	0.9287 ± 0.0072	0.5650 ± 0.0135	0.5949 ± 0.0038	0.8121
AE	0.7694 ± 0.0017	0.5776 ± 0.0093	0.9916 ± 0.0001	0.7577 ± 0.0063	0.8073 ± 0.0049	0.8493
GANomaly	**0.7931 ± 0.0212**	0.6632 ± 0.0216	0.9831 ± 0.0065	0.6622 ± 0.0989	0.8453 ± 0.0745	0.8627
DSVDD	0.6769 ± 0.0681	0.6192 ± 0.0066	0.9218 ± 0.0834	0.7954 ± 0.0175	0.9417 ± 0.0106	0.7837
AAE-DSVDD	0.7789 ± 0.0211	0.7116 ± 0.0060	**0.9925 ± 0.0015**	**0.8598 ± 0.0033**	**0.9595 ± 0.0055**	**0.8682**

training data, such as VPN-VoIP. And use both VPN-VoIP and non-VPN-VoIP in the test set to test the classification performance of the model.

The experiment result is shown in Table 8. In terms of overall performance, it has better recognition abilities than the original features. Especially for Streaming traffic, its recognition ability is close to 100%. This is significantly improved compared with the original feature.

We performed a weighted average on the AUC value, with the weight set to the ratio of the sample size of the current class to the total sample size of all classes, in order to more accurately depict the average detection performance of the model. By averaging the previous experimentation results, AAE-DSVDD has the greatest AUC value (0.8682), followed by GANomaly (0.8627) and OCSVM (0.8584). In conclusion, the AAE-DSVDD performs about 1% better than the other variants. The previously mentioned experimental findings demonstrate that by restricting the distribution of the DSVDD high-dimensional space, the model's identification capability can be marginally enhanced. In the comparative model, it achieved the best performance.

The Fig. 8 shows the precision and recall of one-class classification model on different traffic types. One-class classification based on deep learning usually returns an abnormal score, which requires manual setting of the threshold to determine whether it is abnormal. For example, AE uses the reconstruction error of the sample, and DSVDD uses the distance from the sample to the center of the hypersphere. We removed GANomaly from this part of the experiment because the selection method for the threshold is not directly given. The experimental results of the other five one-class models show that the AAE-DSVDD model has achieved the highest precision, which shows that the AAE-DSVDD model is a better choice for VPN traffic detection than other one-class models. At the same time, all one-class models are also affected by the problem of data imbalance, so the recall of the model in the experimental results is much lower than the precision. However, in application scenarios, a certain degree of false negatives of VPN traffic is generally tolerable. In general, models with higher precision are preferred for VPN traffic identification.

5.5. UKcs and UUCs identification

In this paper, one-class classification models are applied to VPN traffic identification. The main purpose is to make the model build a representational learning model based on the target VPN traffic to avoid problems in the decision surface of the model caused by training, as shown in Fig. 1.

The experimental design in this part needs to demonstrate the recognition ability of the model for UKCs and UUCs. Therefore, during the experiment, we will first select one class as the training dataset from VoIP, Chat, Streaming, Email, and FTP. Then, we use the remaining four classes as different UUCs to test the recognition ability of the trained model. In terms of data set division, we divide it into two categories: VPN and non-VPN. We hold out 80% of the VPN data as the training set and combine the remaining 20% with the non-VPN data as the testing set.

We selected three classes of VoIP, chat, and Streaming to show the detailed metrics of each UUCs test result, as shown in Fig. 9. The vertical axis is the class of data we choose as training data, and the horizontal axis is the class of data we choose as training data. The diagonal line in the figure is the result of testing using UKCs, and the rest are the results of testing with UUCs. For example, VoIP on the abscissa indicates that the training data of the model is VPN-VoIP at this time, and chat on the ordinate indicates the AUC values obtained by using VPN-chat and non-VPN-Chat as test data, respectively. The experiment result shows that the AAE-DSVDD model proposed in this paper effectively improves the ability to identify UUCs. The AUC value increased from 0.5496 of DSVDD to 0.9517 in the experimental result

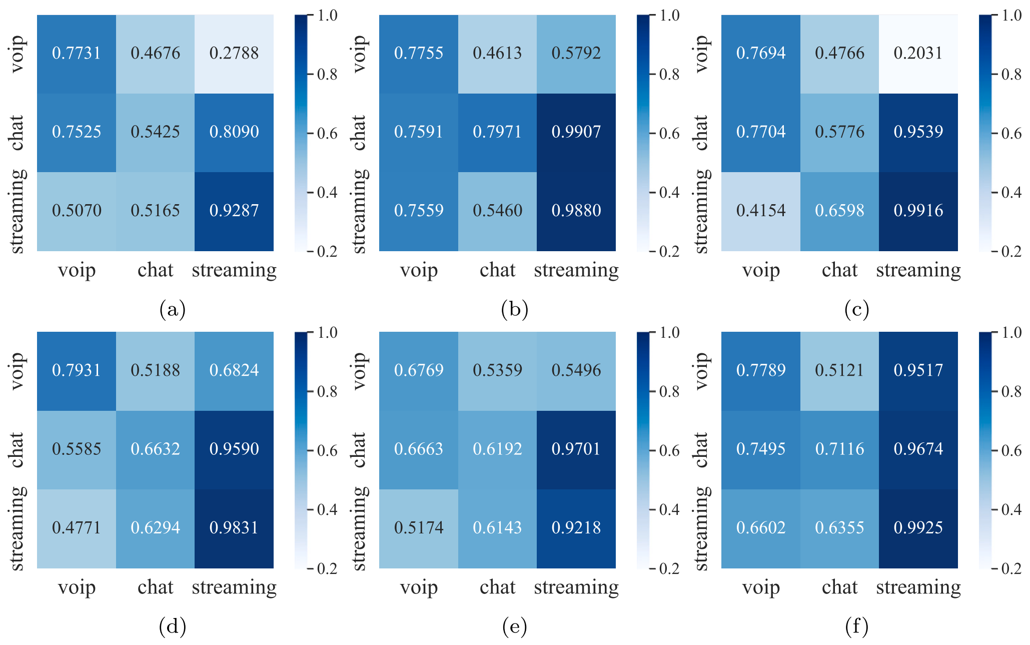

Fig. 9. The AUC value of one-class models to identify UUCs. (a) Isolation Forest, (b) One-class SVM, (c) AE, (d) GANomaly, (e) DSVDD, (f) AAE-DSVDD.

Table 9
A comparison of the ability of supervised learning models and unsupervised learning models to identify UKCs and UUCs.

	Model	wtd. AUC (UKCs)	wtd. AUC (UUCs)	Count (< 0.51)
Supervised learning [10] (machine learning)	KNN	0.9711	0.6451	5
	LR	0.7602	0.4836	12
	SVM	0.9332	0.6152	8
	GNB	0.7201	0.4648	12
	Random forest	0.9516	0.6421	4
	Adaboost	0.9602	0.6349	5
Supervised learning (Deep learning)	DNN [9]	0.9578	0.5783	8
	CNN	**0.9743**	0.5709	7
One-class classification	OCSVM [38]	0.8594	0.7741	3
	iForest [39]	0.8121	0.6404	7
	AE [40]	0.8493	0.6649	4
	GANomaly [41]	0.8627	0.6187	4
	DSVDD [33]	0.7837	0.7045	**1**
	AAE-DSVDD	0.8682	**0.7781**	**1**

of a model trained by VoIP and tested with Streaming. This part of the experiment shows that the AAE-DSVDD model proposed in this paper effectively improves the generalization ability of the model, making the model more robust in the face of UUCs.

Furthermore, many studies report the detection capabilities of different models on the ISCXVPN dataset. Most of these studies use supervised binary classification models, so we include a comparison of one-class models with supervised binary classification models in our experiments. We divide these supervised binary classification models into machine learning and deep learning and compare them with one-class classification models.

We set up two metrics for evaluation. The first one is the AUC value of the model, and we take the weighted average of the AUC values of all experimental results. The second is the number of experiments that lost recognition ability in testing. When the AUC value of the test is less than or equal to 0.5, the prediction result of the model on the test set tends to random guess or give the opposite prediction result. So in this paper, when we use UUCs for testing, we set the threshold as 0.51. When the AUC value of the model tested on an UUC is less than 0.51, we considered that the trained model lost recognition ability on current UUC.

In the experiments in this paper, 5 types of traffic are used, and one of the classes is selected as the training data for each experiment, so there are 5 groups of experiments using UKCs for testing and 20 groups of experiments using UUCs for testing.

The experimental result is shown in Table 9. We compared the weighted average AUC values of supervised learning models under UKCs and UUCs, among which CNN achieved the highest value of 0.9743 on the UKCs test, and the AUC values of most models in supervised learning are greater than 0.9. This value is significantly higher than those one-class classification models. But, these supervised learning models can hardly identify UUCs, and their AUC values are only between 0.5 and 0.7, which are only slightly better than random guessing. The one-class classification model has a stronger ability to identify UUCs. The AAE-DSVDD model proposed in this paper maintains the AUC value of UKCs recognition at 0.8682, while the AUC value of UUCs recognition reaches 0.7781. And among the 20 experiments tested with UUCS, the AAE model only lost recognition on one UUC test dataset.

The above experiments based on UKCs and UUCs verify the limitations of supervised learning models in traffic detection as shown in

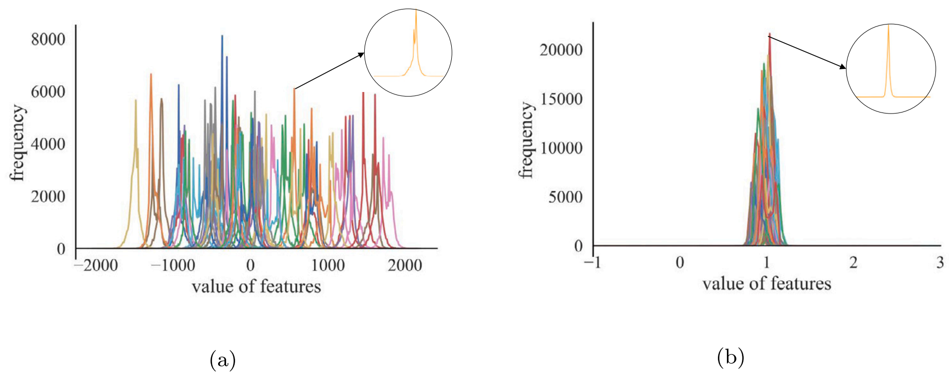

Fig. 10. Distribution of different model outputs., (a) DSVDD, (b) AAE-DSVDD.

Fig. 1, which cannot handle unknown types of test data. Therefore, this paper proposes to use one-class classification model for VPN traffic detection to deal with diverse background traffic. At the same time, the AAE-DSVDD model proposed in this paper further improves the recognition ability of the model for UUC type data.

5.6. *Effectiveness of distribution constraint*

In this section, an experiment is designed to verify the effectiveness of distribution constraints in AAE-DSVDD. Therefore, this part of the experiment compares the output of the DSVDD and the AAE-DSVDD.

The outcome is shown in Fig. 10. First of all, it can be seen that the output of AAE-DSVDD basically meets the normal distribution. Since we used the trick mentioned in Section 5.2, all features have the same mean. The center of the hypersphere is specified before training. However, the center of DSVDD is set to initial network outputs. The mean value of each feature in DSVDD is relatively scattered. Secondly, on the single feature distribution, two different features of DSVDD and AAE-DSVDD are sampled, respectively. The distribution is shown in the circle in Fig. 10. The features of DSVDD do not satisfy normal distribution.

In summary, the AAE-DSVDD model effectively meets distribution constraints on the output data.

5.7. *Hypersphere collapse*

This section designs experiments to verify whether hypersphere collapse occurs under different network settings. As mentioned above, DSVDD has experienced hypersphere collapse in three cases: (1) Fix the center of the sphere when the model is initialized; (2) Set the neural network to have no bias term; (3) Set an unbounded activation function. In the analysis in Section 3.6, AAE-DSVDD does not need to set a bias term and uses an unbounded activation function. Therefore, the experiment in this part verifies whether hypersphere collapse occurs in these two cases. In order to intuitively display the collapse of the hypersphere, we initialize the weight of the neural network to a small value, on the order of 0.01.

The experimental result is shown in Fig. 11. As Fig. 11-(a) shows, the hypersphere collapse occurs when DSVDD sets the neural network to have bias items. The AUC value has been fixed at 0.5 since the 80th epoch. This means $\phi(x) = c$. As shown in Fig. 11-(b,c,d), the bias term, activation function, and weight initialization all affect the performance of the model. The performance of the model is best when using the bias term and the LeakyReLU activation function. This experiment also verifies the analysis of the hypersphere collapse problem in Section 3.6. AAE-DSVDD does not change the fact that the all-zero solution is an optimal solution of the objective function. Therefore, when the weight is initialized to a smaller order of 0.01. It is difficult for the model to converge to the optimal solution, but there is also not appear hypersphere collapse. The AUC is greater than 0.5. Therefore, this paper does not recommend setting too small an initial weight when training the AAE-DSVDD model to increase the standard deviation of the output.

The above experimental results verify the inference of hypersphere collapse in Section 3.6. AAE-DSVDD can use bias terms and bounded activation functions without hypersphere collapse.

5.8. *Astringency*

In the experiment in this part, we displayed the loss change of the model at each epoch. The experimental results are shown in the Fig. 12. The x-axis represents epoch, and the y-axis represents log loss. The Fig. 12(a) shows the merge loss change in AAE-DSVDD. In the later stages of training, the model reaches a relatively stable state.

The experimental results verify the convergence of the AAE-DSVDD model. After adding distribution constraints, the model can still reach a convergent solution.

5.9. *Sensitivity analysis*

The predefined parameters required in the AAE model are the mean and standard deviation of $p(z)$, and the loss weight α.

Firstly, we analyze the impact of parameter α settings on model performance. α is used to control the weight of L_{DSVDD} and $L_{distribute}$. In the experiment, the weight parameter α grid searched from 0.1 to 1 with a step size of 0.1.

The experimental results are shown in Fig. 13. In the nine experiments, the AUC value reaches its highest when the α is 0.9. We make the following explanation for the above experimental results: The distribution constraint does not directly change the decision surface of the model. The main identification ability of the model comes from reducing of L_{DSVDD}. So when the weight of L_{DSVDD} is small, the identification ability of the model decreases. Therefore, this paper set the weight at 0.9 in the previous experiment. It is worth mentioning that when training on voip, AAE-DSVDD tends to use a smaller α. Combined with Table 7, we analyze that the recognition ability of DSVDD on voip is poor, and the smaller weight can make AAE-DSVDD focus on the abnormal detection ability of AE to obtain better detection performance.

Then, we analyzed the influence of the mean and standard deviation of $p(z)$ setting on the identification ability of AAE-DSVDD, and the results are shown in Fig. 14. Based on the FTP data, we set the mean μ and standard deviation σ of $p(z)$ on five orders of magnitude, which are 0.01, 0.1, 1, 10, and 100. We performed grid searches on two orders of magnitude, 0.1 and 1.0, which have a high AUC value. Next, we analyze the impact of μ and σ on the performance of AAE-DSVDD.

The first parameter is the mean of the prior distribution μ. In the normalization proposed in Section 3.6 of this paper, the mean value

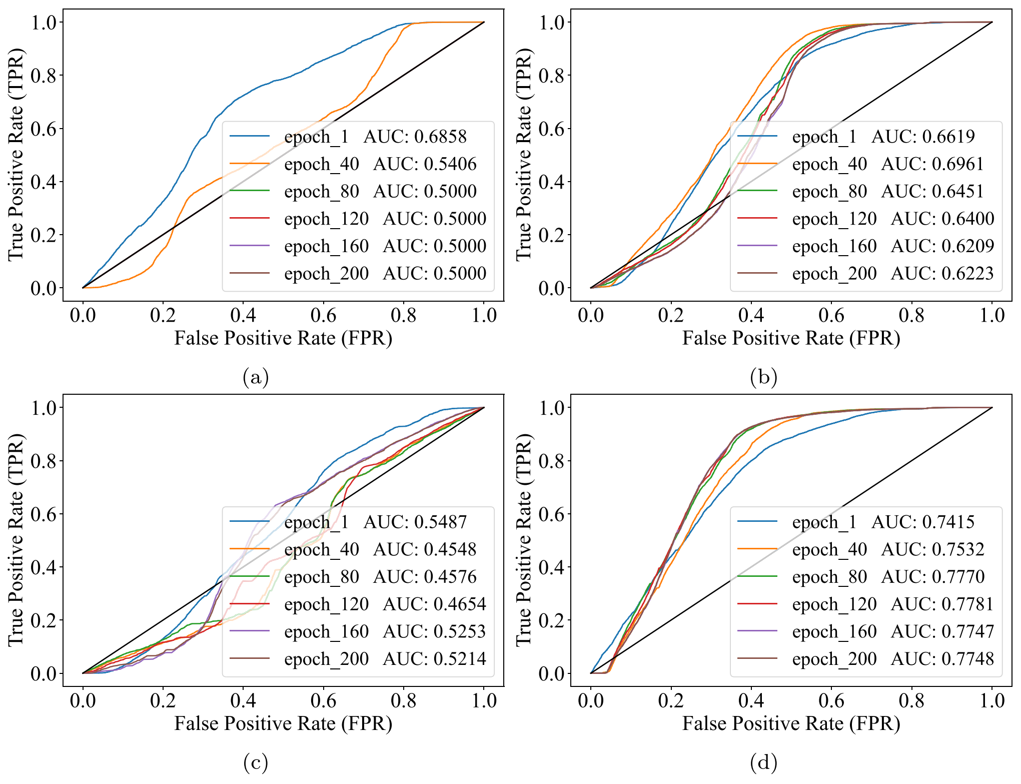

Fig. 11. Roc curves of different epochs. (a) DSVDD, set bias=True in the neural network, (b) AAE-DSVDD, set bias=False in the neural network, (c) AAE-DSVDD, Set the sigmoid activation function, (d) AAE-DSVDD, set bias=True and LeakyReLU activation function, and the magnitude of weight initialization is 0.1.

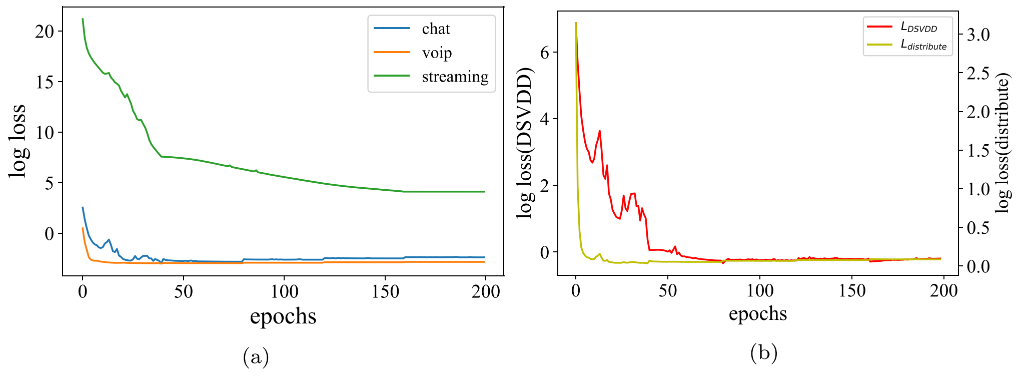

Fig. 12. Loss of different epochs in AAE-DSVDD. (a) Change of loss with epochs for different types of data, (b) L_{DSVDD}&$L_{distribute}$changes during VoIP traffic training.

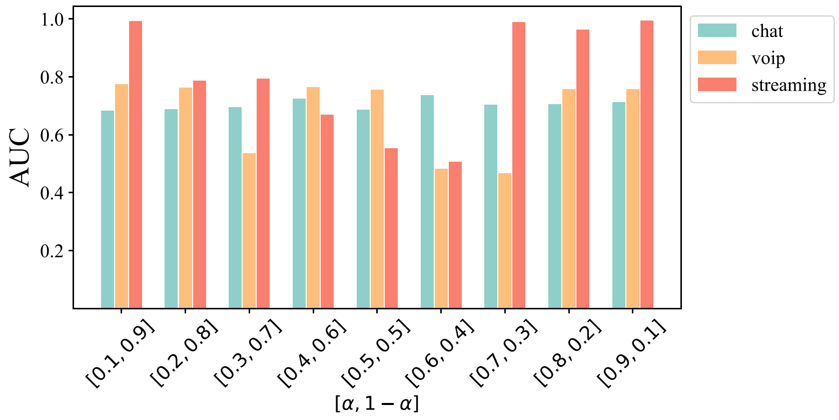

Fig. 13. The AUC value of AAE-DSVDD under different weight α.

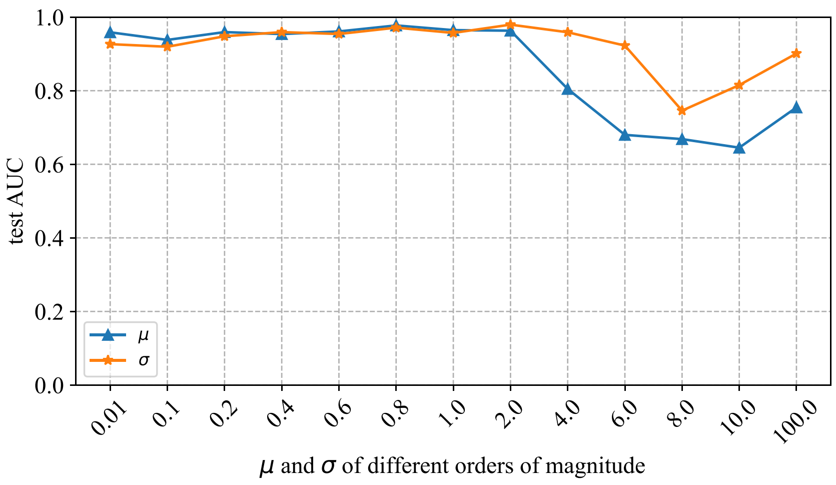

Fig. 14. The variation of AUC value at different μ and σ of prior probability $p(z)$.

participates in the distribution constraint as a constant item. In theory, the distribution constraint in AAE-DSVDD should not be affected by the change of μ. From the experimental results, When μ is less than 2, the AUC value of the model is slightly affected. However, we observed that when μ is greater than 2, the performance of the model drops sharply. The reason is that the AAE part of the AAE-DSVDD in this paper cannot converge, which leads to the failure to construct the aggregated posterior distribution $q(z)$ to meet our expectations. This causes the trained model to fail to identify the test data, and the AUC value is less than 0.5.

Another parameter is the standard deviation of the prior distribution σ. In this paper, the normalization is designed to be normalized according to different levels of standard deviation and radius. Therefore, when the σ is small, the classification performance of the model is relatively stable. When the sigma increases, the aggregated posterior distribution gradually disperses. At this time, the distribution constraint will cause the performance of the DSVDD model to decline. But the recognition ability is maintained at a low level, and the AUC value is about 0.7.

In summary, the settings of μ and σ will affect the performance of the model. When the order of magnitude is small, the normalization method proposed in this paper can effectively constrain the aggregated posterior distribution $p(z)$ and the output of DSVDD $p_{\phi}(z)$, and the identification ability of AAE-DSVDD is stable. When the magnitude increases, the recognition ability of the model will decrease, or even completely lose the recognition ability. Thus, we set μ and σ to 1.0 and 0.01 in above experiment.

6. Discussion

According to research, most of the traffic on the Internet comes from unknown applications [42]. It is easily ignored by supervised learning models [43]. Therefore, one-class classification models are more suitable for VPN traffic detection than supervised learning models. This paper proposes the AAE-DSVDD model for VPN traffic detection and evaluates the performance of AAE-DSVDD on the ISCXVPN dataset. In the experiments, this paper compares the identification abilities of the supervised learning models and the one-class classification models on UKCs and UUCs. Experimental results demonstrate the effectiveness of the distribution constraints in AAE-DSVDD. It improves the ability of DSVDD to identify UUCs. This effectively solves the problem of data collection bias that may occur in VPN traffic identification. For example, email traffic may be very low or even not collected in the train dataset. The AAE-DSVDD model can overcome this problem even if the collected data only obtains VPN-VoIP traffic, the trained model still maintains a certain level of identification ability for Email, Chat and other behaviors.

However, the AAE-DSVDD model proposed in this paper requires preprocessing. We filter some short streams in the data preprocessing stage. The reasons are as follows: First, from the perspective of VPN tunnel traffic, the tunnels of OpenVPN will not disconnect because of network idleness. Therefore, tunnel traffic seldom appears to have a short flow. Second, the short flows in the ISCXVPN data set have no practical significance. It includes DNS query, refuce connections, and some short connections without data transmission. They do not match the behavior marked by the PCAP file. The last reason is that our work uses a sliding window to split the flow. If there are a small number of packets in the flow, the extracted sequence needs to be filled with a lot of meaningless values. It may affect the training of the model. Based on the above reasons, these short streams are directly filtered in the data preprocessing stage.

Finally, the ISCXVPN dataset is flawed. It uses an external VPN service provider and connects to it using OpenVPN to collect VPN traffic [15]. However, the dataset does not actually contain any OpenVPN packets. This will cause the packet length and time interval sequence features used in this paper to be unable to represent the real OpenVPN protocol traffic. Therefore, if AAE-DSVDD is applied to OpenVPN traffic detection, it is necessary to re-collect actual OpenVPN traffic to train the model.

7. Conclusion

VPN traffic identification is a challenge for encrypted traffic identification. The difficulty lies not only in data encryption but also in the fact that the traffic inside the VPN tunnel shows high similarity with non-VPN traffic. In this paper, we propose an one-class learning model called AAE-DSVDD for VPN traffic identification. It adjusts the output distribution of DSVDD through the aggregated posterior distribution of AAE. It makes the assumption that concentrated samples are placed with high confidence in the center of the hypersphere and dispersed samples are placed with low confidence at the edge of the hypersphere. It effectively alleviates the hypersphere collapse problem. At the same time, we analyzed and designed the features. We are using the packet length and time interval sequence and automatically extracting the deep features of this sequence through a convolutional neural network. Finally, experimental verification is carried out on the ISCXVPN, and the AAE-DSVDD is compared with some one-class learning models and some supervised learning models. The experimental results show that AAE-DSVDD has achieved an advanced one-class learning model. The overall recognition performance has improved by about 1%. AAE-DSVDD also achieves the best performance when identifying classes not included in the training data, even higher than the contrasting supervised learning models.

However, there are still some problems to be solved in VPN traffic identification. First, with the escalation of confrontation, VPNs are trying to use traffic obfuscation technology to masquerade traffic and avoid detection. Correspondingly, the detection of confusing traffic requires a stronger detection model, that can detect samples with a higher similarity. Meanwhile, a more realistic problem is that the traffic in the VPN tunnel is far more complex than in the experimental environment. The seven types of traffic mentioned in ISCXVPN may be mixed in a VPN tunnel. This is also the biggest challenge that needs to be solved before VPN traffic identification can be used on real networks.

CRediT authorship contribution statement

Sicai Lv: Conceptualization, Methodology, Writing – original draft. **Chao Wang:** Data Curation, Methodology. **Zibo Wang:** Formal analysis, Investigation. **Shuo Wang:** Validation, Writing – review & editing. **Bailing Wang:** Methodology, Writing – review & editing. **Yongzheng Zhang:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Acknowledgments

The work of this paper is supported by the National Natural Science Foundation of China (62272129).

References

[1] P.B. Gentry, What is a VPN? Inform. Secur. Tech. Rep. 6 (1) (2001) 15–22.
[2] J. Cao, X.-L. Yuan, Y. Cui, J.-C. Fan, C.-L. Chen, A VPN-encrypted traffic identification method based on ensemble learning, Appl. Sci. 12 (13) (2022) 6434.
[3] S. Miller, K. Curran, T. Lunney, Multilayer perceptron neural network for detection of encrypted VPN network traffic, in: 2018 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), 2018, pp. 1–8, http://dx.doi.org/10.1109/CyberSA.2018.8551395.
[4] B.K. Chawla, O. Gupta, B. Sawhney, A review on IPsec and SSL VPN, Int. J. Sci. Eng. Res. 5 (11) (2014) 21–24.
[5] J. Jones, H. Wimmer, R.J. Haddad, Pptp vpn: An analysis of the effects of a ddos attack, in: 2019 SoutheastCon, IEEE, 2019, pp. 1–6.
[6] K. Hogan, S. Servan-Schreiber, Z. Newman, B. Weintraub, C. Nita-Rotaru, S. Devadas, ShorTor: Improving tor network latency via multi-hop overlay routing, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, pp. 1933–1952.
[7] Y. He, L. Hu, R. Gao, Detection of tor traffic hiding under obfs4 protocol based on two-level filtering, in: 2019 2nd International Conference on Data Intelligence and Security, ICDIS, IEEE, 2019, pp. 195–200.
[8] W. Xu, F. Zou, Obfuscated tor traffic identification based on sliding window, Secur. Commun. Netw. 2021 (2021) 1–11.
[9] S. Miller, K. Curran, T. Lunney, Detection of virtual private network traffic using machine learning, Int. J. Wirel. Netw. Broadband Technol. (IJWNBT) 9 (2) (2020) 60–80.
[10] J. Mazel, M. Saudrais, A. Hervieu, ML-based tunnel detection and tunneled application classification, 2022, arXiv preprint arXiv:2201.10371.
[11] P. Fu, C. Liu, Q. Yang, Z. Li, G. Gou, G. Xiong, Z. Li, NSA-Net: A NetFlow sequence attention network for virtual private network traffic detection, in: Web Information Systems Engineering–WISE 2020: 21st International Conference, Amsterdam, the Netherlands, October 20–24, 2020, Proceedings, Part I 21, Springer, 2020, pp. 430–444.
[12] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE International Conference on Intelligence and Security Informatics, ISI, IEEE, 2017, pp. 43–48.
[13] M. Zain ul Abideen, S. Saleem, M. Ejaz, Vpn traffic detection in ssl-protected channel, Secur. Commun. Netw. 2019 (2019) 1–17.
[14] T. Yildirim, P. Radcliffe, Voip traffic classification in ipsec tunnels, in: 2010 International Conference on Electronics and Information Engineering, vol. 1, IEEE, 2010, pp. V1–151.
[15] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP, 2016, pp. 407–414.
[16] W.-b. Pan, G. Cheng, X.-j. Guo, S.-x. Huang, Review and perspective on encrypted traffic identification research, Journal on Communications 37 (9) (2016) 154–167.
[17] Q. Zhang, J. Li, Y. Zhang, H. Wang, D. Gu, Oh-Pwn-VPN! security analysis of OpenVPN-based android apps, in: Cryptology and Network Security: 16th International Conference, CANS 2017, Hong Kong, China, November 30—December 2, 2017, Revised Selected Papers, Springer, 2018, pp. 373–389.
[18] D. Xue, R. Ramesh, A. Jain, M. Kallitsis, J.A. Halderman, J.R. Crandall, R. Ensafi, {OpenVPN} is open to {VPN} fingerprinting, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 483–500.
[19] S. Oh, M. Lee, H. Lee, E. Bertino, H. Kim, AppSniffer: Towards robust mobile app fingerprinting against VPN, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 2318–2328.
[20] A. Gupta, VPN-nonVPN traffic classification using deep reinforced naive Bayes and fuzzy K-means clustering, in: 2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW), IEEE, 2021, pp. 1–6.
[21] S. Bagui, X. Fang, E. Kalaimannan, S.C. Bagui, J. Sheehan, Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features, J. Cyber Secur. Technol. 1 (2) (2017) 108–126.
[22] J. Cao, X.-L. Yuan, Y. Cui, J.-C. Fan, C.-L. Chen, A VPN-encrypted traffic identification method based on ensemble learning, Appl. Sci. 12 (13) (2022) 6434.
[23] L. Guo, Q. Wu, S. Liu, M. Duan, H. Li, J. Sun, Deep learning-based real-time VPN encrypted traffic identification methods, J. Real-Time Image Process. 17 (2020) 103–114.
[24] J. Tang, L. Yang, S. Liu, W. Liu, M. Wang, C. Wang, B. Jiang, Z. Lu, Caps-lstm: A novel hierarchical encrypted VPN network traffic identification using capsnet and LSTM, in: Science of Cyber Security: Third International Conference, SciSec 2021, Virtual Event, August 13–15, 2021, Revised Selected Papers, Springer, 2021, pp. 139–153.
[25] Y. Chen, Z. Li, J. Shi, G. Gou, C. Liu, G. Xiong, Not afraid of the unseen: a siamese network based scheme for unknown traffic discovery, in: 2020 IEEE Symposium on Computers and Communications, ISCC, IEEE, 2020, pp. 1–7.
[26] J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2014) 1257–1270.
[27] V.T. Do, P. Engelstad, B. Feng, T. Van Do, Detection of DNS tunneling in mobile networks using machine learning, in: Information Science and Applications 2017: ICISA 2017 8, Springer, 2017, pp. 221–230.
[28] A. Nadler, A. Aminov, A. Shabtai, Detection of malicious and low throughput data exfiltration over the DNS protocol, Comput. Secur. 80 (2019) 36–53.
[29] E. Rivera, L. Tengana, J. Solano, A. Castelblanco, C. López, M. Ochoa, Risk-based authentication based on network latency profiling, in: Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, 2020, pp. 105–115.
[30] O. Aouedi, K. Piamrat, D. Bagadthey, Handling partially labeled network data: A semi-supervised approach using stacked sparse autoencoder, Comput. Netw. 207 (2022) 108742.
[31] K. Wu, Y. Zhang, T. Yin, TDAE: Autoencoder-based automatic feature learning method for the detection of DNS tunnel, in: ICC 2020-2020 IEEE International Conference on Communications, ICC, IEEE, 2020, pp. 1–7.
[32] R. Chalapathy, A.K. Menon, S. Chawla, Anomaly detection using one-class neural networks, 2018, arXiv preprint arXiv:1802.06360.
[33] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S.A. Siddiqui, A. Binder, E. Müller, M. Kloft, Deep one-class classification, in: International Conference on Machine Learning, PMLR, 2018, pp. 4393–4402.
[34] Y. Li, Y. Xu, Y. Cao, J. Hou, C. Wang, W. Guo, X. Li, Y. Xin, Z. Liu, L. Cui, One-class LSTM network for anomalous network traffic detection, Appl. Sci. 12 (10) (2022) 5051.
[35] X. Chen, C. Cao, J. Mai, Network anomaly detection based on deep support vector data description, in: 2020 5th IEEE International Conference on Big Data Analytics, ICBDA, IEEE, 2020, pp. 251–255.
[36] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (2004) 45–66.
[37] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, 2015, arXiv preprint arXiv:1511.05644.

[38] B. Schölkopf, R.C. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, Adv. Neural Inform. Process. Syst. 12 (1999).
[39] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data (TKDD) 6 (1) (2012) 1–39.
[40] G. Andresini, A. Appice, D. Malerba, Autoencoder-based deep metric learning for network intrusion detection, Inform. Sci. 569 (2021) 706–727.
[41] S. Akcay, A. Atapour-Abarghouei, T.P. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, in: Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14, Springer, 2019, pp. 622–637.
[42] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Lee, Internet traffic classification demystified: myths, caveats, and the best practices, in: Proceedings of the 2008 ACM CoNEXT Conference, 2008, pp. 1–12.
[43] N. Fu, Y. Xu, J. Zhang, R. Wang, J. Xu, FlowCop: Detecting "Stranger" in network traffic classification, in: 2018 27th International Conference on Computer Communication and Networks, ICCCN, IEEE, 2018, pp. 1–9.

Sicai LV is currently studying at the Department of Computer Science, Harbin Institute of Technology. His research interests include encrypted traffic identification, intrusion detection.

Chao Wang is currently studying at the Department of Computer Science, Harbin Institute of Technology. His research interests include Industrial control intrusion detection, anomaly detection.

Zibo Wang is currently a Ph.D. candidate in the school of Computer Science of Technology, Harbin Institute of Technology, China. He received his master degree from Northeast Forestry University in 2017. His research interests include industrial control system security and attack behavior modeling.

Shuo Wang is an undergraduate at School of life and Pharmaceutical Sciences ,Dalian University of Technology. His research interests include Network Security, and Pharmaceutical Sciences.

Wang Bailing received his Ph.D. degree in Computer Science and Technology from Harbin Institute of Technology, China. He is currently a Professor at School of Computer Science and Technology, Harbin Institute of Technology. His current research interests involve network security, information content security, industry internet security.

Yongzheng Zhang has graduated from Harbin Institute of Technology, majoring in computer system structure and received a doctorate in engineering in 2006. His current research interests including network security situation awareness, threat detection.