Chinese Journal of Electronics
vol. 33, no. 1, pp. 58–71, January 2024
https://doi.org/10.23919/cje.2022.00.173

RESEARCH ARTICLE

FlowGANAnomaly: Flow-Based Anomaly Network Intrusion Detection with Adversarial Learning

Zeyi LI[1], Pan WANG[2], and Zixuan WANG[3]

1. *School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
2. *School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
3. *School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*

Corresponding author: Pan WANG, Email: wangpan@njupt.edu.cn
Manuscript Received June 16, 2022; Accepted November 23, 2022

Abstract — In recent years, low recall rates and high dependencies on data labelling have become the biggest obstacle to developing deep anomaly detection (DAD) techniques. Inspired by the success of generative adversarial networks (GANs) in detecting anomalies in computer vision and imaging, we propose an anomaly detection model called FlowGANAnomaly for detecting anomalous traffic in network intrusion detection systems (NIDS). Unlike traditional GAN-based approaches, which are composed of a flow encoder, a convolutional encoder-decoder-encoder, a flow decoder and a convolutional encoder, the architecture of this model consists of a generator (G) and a discriminator (D). FlowGANAnomaly maps the different types of traffic feature data from separate datasets to a uniform feature space, thus can capture the normality of network traffic data more accurately in an adversarial manner to mitigate the problem of the high dependence on data labeling. Moreover, instead of simply detecting the anomalies by the output of D, we proposed a new anomaly scoring method that integrates the deviation between the output of two Gs' convolutional encoders with the output of D as weighted scores to improve the low recall rate of anomaly detection. We conducted several experiments comparing existing machine learning algorithms and existing deep learning methods (AutoEncoder and VAE) on four public datasets (NSL-KDD, CIC-IDS2017, CIC-DDoS2019, and UNSW-NB15). The evaluation results show that FlowGANAnomaly can significantly improve the performance of anomaly-based NIDS.

Keywords — Anomaly detection, Unsupervised learning, Generative adversarial network, Intrusion detection system.

Citation — Zeyi LI, Pan WANG, Zixuan WANG, "FlowGANAnomaly: Flow-Based Anomaly Network Intrusion Detection with Adversarial Learning," *Chinese Journal of Electronics*, vol. 33, no. 1, pp. 58–71, 2024. doi: 10.23919/cje.2022.00.173.

I. Introduction

With the advent of the 5G era, Internet of everything has become a significant trend. Due to the increasing frequency of attacks on Internet of things (IoT), vehicle networking, and other environments, cyber security is gaining more and more attention from academia and industry, which has prompted many researchers to design and develop effective network intrusion detection systems (NIDS). In general, intrusion detection techniques fall into two genres: misuse-based and anomaly-based detection. Misuse-based detection systems can parse the payload in network traffic and match it with existing signatures-alert messages to indicate abnormal behaviour once the system is under attack [1]. Nevertheless, misuse-based intrusion detection systems (IDS) methods incur high computational costs and require manual signatures maintenance by experts over time [2]. Above all, new types of attacks can not be identified in time, which largely limits the development of misuse-based IDS [3]. Accordingly, anomaly-based NIDS mitigates the problem by learning normal network behaviour features and capturing deviations from normal ones. These deviations are called anomalies [4]. Therefore, anomaly-based methods are more suitable for NIDS than misuse-based approaches, especially to detect new types of attacks.

Associate Editor: Prof. De-chuan ZHAN, Nanjing University.

Anomaly-based network intrusion detection system (ANIDS) focuses on capturing the normal behaviour features in the network and calculating the deviation from the coming traffic flow to detect anomalies. Many traditional machine learning (ML) algorithms have been used to develop anomaly detection (AD) models [5]. Nevertheless, ML-based AD algorithms have many challenges. First, it is hard to learn the traffic features for ML algorithms because the network traffic data is complex, high-dimensional, and non-linear. Second, the size of normal network traffic is huge leading to great difficulties in algorithm training. Third, the concept of an anomaly is subjective and highly depends on the application domain and context. In addition, anomalous samples are difficult to label and are usually done manually by human experts. And finally, the boundary between normal and anomaly is often imprecise. Fortunately, deep learning (DL) has been a great success as a revolutionary innovation in computer vision, imaging, and speech. DL uses automatic feature learning to perform better modeling, avoiding task-specific engineering and large amounts of prior knowledge. Generative adversarial network (GAN), an emerging technique, provides helpful context for the development of DL because its most valuable contribution has been proved to well capture the normality underlying the given data and then detect abnormal instances, especially on image/video data. Recently, existing GAN-based models and theories [6] have been verified in the literature that they are suitable for anomaly detection. So, more and more attention is drawn by researchers from traditional training methods of DL toward adversarial learning methods.

1. Motivations

Recently, deep anomaly detection (DAD) technology has gained much attention for its ability of detecting unknown attacks without having to manually build a library of traffic behaviour based on DL algorithms. However, DAD techniques are still more or less limited by the following challenges:

1) The detection of traffic behaviour may be biased due to the lack of well-designed DL models and threshold selection algorithms. The boundary between normal and outlier traffic is often unclear and imprecise, resulting in a low recall of anomaly detection.

2) Traditional convolutional modules [7] require fixed input parameters, and frequent changes of model parameters are required to train different data sets.

3) The calculation of anomaly scores is not accurate enough to detect the deviation from the features of the flow.

4) The last issue is that DAD rarely has interpretability, which makes the algorithm less trustworthy.

2. Key contribution

GAN is rapidly becoming a popular DAD method because of its excellent automatic feature extraction through adversarial learning [8]. This paper proposes an ANIDS algorithm based on adversarial learning called FlowGANAnomaly which consists of a generator (G) and a discriminator (D). The proposed algorithm learns the underlying distribution of network traffic data through adversarial training. At the same time, multiple loss functions are combined to form multiple constraints aiming to train the GAN model more accurately, and the scores of discriminators and reconstruction distances are weighted as anomaly scores. Given that the features of each dataset are not consistent, G's flow encoder maps the different types of traffic feature data from separate datasets to a uniform feature space, thus can capture the normality of network traffic data more accurately. The main contributions of this paper are as follows:

1) This paper proposes a network anomaly detection algorithm called FlowGANAnomaly. GAN consisting of a deconvolutional decoder, a flow encoder, a flow decoder, and three convolutional encoders plays a pivotal role in this model. A convolutional encoder acts as D and the remaining parts make up G. The flow encoder can map the data of different features into G uniformly, which can be applied to different flow feature computing environments and has good scalability. Meanwhile, the results prove that adding the flow encoder module works better.

2) The anomaly score is composed of Euclidean distance between two convolutional encoders in G and D's output, which can well reflect the anomalous characteristics of network traffic. The results are better than the traditional GAN methods.

3) We propose an improved threshold selection algorithm for anomaly scores, which utilizes the kernel functions as thresholds to accurately obtain the normal and abnormal traffic boundaries using kernel functions and set them as thresholds.

4) The six benchmarking ML/DL methods on four public datasets are compared to experimentally evaluate the proposed method with good results. Moreover, we apply the ATON (attention-guided triplet deviation network for outlier interpretation) algorithm to explain the FlowGANAnomaly to ensure that the proposed algorithm is credible by judging the contribution of the features.

II. Related Works

1. Review of anomaly detection algorithms

The first wave of anomaly detection algorithms was on the basis of ML algorithms, which can be divided into six families [9]: clustering, neighbor-based, density-based, statistical, angle-based, and classification-based. Clustering assumes a large number of unlabeled data to be normal and learns to group them into several clusters to discriminate between the normal and anomalies [10]. K-means is a widely used clustering algorithm. Neighbour-based algorithms are used to classify data based on similarities in distance metrics such as Euclidean, and

Manhattan distance. A typical example is the K-nearest neighbour (KNN) [11]. Density-based algorithms like local outlier factor (LOF) [12] are based on a neighbor-based family to improve the capability of detecting anomalies. Statistical algorithms like principal component analysis (PCA) [13] are used to detect subspaces of the data to identify deviations from the expected subspaces, and thus detect anomalies [14]. Angle-based algorithms relate data to high-dimensional spaces, using the variance in the angles between a data point to the other points as an anomaly score, such as angle-based outlier detection (ABOD) [15]. Classification-based algorithms aim at learning a decision boundary to group the data points. One-class support vector machine (one-class SVM) [16] and isolation forest (IF) [17] are two representatives. The paper [18] proposes the anomaly detection method of iForest, which has better experimental results and more robustness than LOF and OCSVM. However, when encountering large-scale data, AD algorithms will reach a bottleneck [19]. The first wave posed several challenges: 1) Low recall rate; 2) High-dimensional or not-independent data; 3) Data-efficient learning of normality/abnormality; and 4) Noise resilience.

The second wave is based on DL algorithms since 2015 due to the superior performance [20] applied in the area of computer vision [21]/image/speech. We can simply summarize three apparent trends: 1) More novel and advanced deep learning algorithms are adopted [22], such as MLP/CNN/AE/RNN even GAN/GNN; 2) Learning methods adopted by deep learning models are gradually from supervised [23] to semi-supervised [24], weak-supervised and unsupervised learning; 3) The anomaly score learning approaches are moving from separately learning to end-to-end including more precise threshold selection algorithms.

2. Deep anomaly detection algorithms in IDS

Deep anomaly detection (DAD) techniques for short, aim at learning hierarchical discriminative features or anomaly scores via deep neural networks for the sake of anomaly detection. DAD algorithms can learn from historical data with normal and anomalous traffic and automatically reduce the network traffic complexity to find the correlations among data without human intervention. Furthermore, DAD is more powerful in detecting zero-day attacks and adapting to evolving systems. In summary, there are two main genres applied in DAD-based IDS: AE-based and GAN-based, which will be introduced in detail in the following subsections.

1) Autoencoder-based AD Methods in IDS

As a popular deep structure for anomaly detection, AE and its genres have played a very important role during these decades. Autoencoders (AEs) are a neural network architecture that has emerged as a suitable solution to anomaly detection in recent years. AEs usually behave well on normal data instances with small reconstruction errors (REs), but poorly reconstruct anomalies with large REs when they attempt to reconstruct the original input data at the output layer. RE is commonly used as a measure of anomaly score. Gharib *et al.* [25] presented AutoIDS, a network anomaly detector by cascading a sparse AE and AE to increase accuracy and decrease the time complexity, in which anomalous flows are distinguished from normal ones by the first detector and the second one is only used for difficult samples that the first detector is not confident about. AutoIDS was trained on NSL-KDD and private datasets, which detected the network traffic anomalies by calculating the REs of the AE models based on the manually selected threshold. A plug-and-play NIDS called kitsune, utilizing an ensemble of AEs to collectively detect anomalous traffic on the local network in an unsupervised online manner was proposed by Mirsky *et al.* [26]. And the evaluations showed that Kitsune can detect various attacks with a performance comparable to offline anomaly detectors, even on a Raspberry PI.

Zavrak *et al.* [27] applied AE and VAE to detect anomalous network traffic from flow-based data. The experimental results show that VAE outperforms AE and one-class SVM on CICIDS2017 dataset. Abolhasanzadeh *et al.* [28] proposed an AE-based IDS approach from the point of view of dimensionality reduction to detect anomalous network behavior. With the comparison of PCA, KernalPCA, and factor analysis (FA), the experimental results showed that the proposed AE method performed better than the others on the NSL-KDD dataset.

Shone *et al.* [29] proposed a combination of deep and shallow learning which leverages the feature learning power of non-symmetric deep auto-encoder (NDAE) and the accuracy and speed of random forest (RF). They obtained promising experimental results on KDD CUP'99 and NSL-KDD. CANnolo, an IDS based on LSTM-AE to identify anomalies in controller area networks (CANs), creates a reconstructed time series of CAN packets for each CAN-ID that minimizes the RE so that a great error rate would flag any potential anomaly [30].

In summary, the AE family based on AD methods have the following several advantages: the methods are simple, easy to design, and general for different data types. In addition, the model can use different classical types of AE variants for anomaly detection. However, it still has shortcomings in that the model learning cannot learn the deep underlying distribution of the normal dataset well, which can lead to some rare and biased normal flows being classified as outliers. As for the objective function of data reconstruction design, the model is only dimensionality reduction or data compression, but not anomaly detection. Therefore, the model of the AE series still needs more improvement for anomaly detection.

2) GAN-based AD Methods in IDS

GAN-based anomaly detection methods are generally used to learn the latent feature space by adversarial learning of G and D so that the latent space can capture the normality of the given data well. In some form, residuals between actual and generated data samples are defined as anomaly scores. Schlegl *et al.* [31] proposed AnoGAN, a deep convolutional GAN, to learn multiple normal anatomical variants, accompanied by a new anomaly scoring scheme based on a mapping from image space to latent space. The model flags anomalies and scores image patches to apply to recent data, indicating that they fit a known distribution. Zenati *et al.* [32] utilizes a recently developed GAN model for anomaly detection and achieves state-of-the-art performance on image and NIDS datasets. Subsequently, Schlegl *et al.* [33] proposed fast AnoGAN (f-AnoGAN), a GAN-based semi-supervised learning method that can identify anomalous images and fragments. This method detects anomalies using a combined anomaly score based on a trained model's building blocks containing a residual discriminator feature and an image reconstruction error. Akcay *et al.* [34] introduced a new anomaly detection model called GANomaly that uses conditional GAN (CGAN) to jointly learn the generation of high-dimensional image space and inference of the latent space with given conditions. They designed an encoder-decoder-encoder sub-network in the G network, enabling the model to map the input image to a lower-dimensional vector, which is then used to reconstruct the generated output image. The distance between these images and the underlying vector is minimized during the training period of learning the data distribution of the normal samples. Siniosoglou *et al.* [35] proposed an IDS called MENSA adopting AE and GAN for detecting the operational anomalies and classifying ModBus/TCP and DNP3 cyber-attacks in smart grid. Adversarial loss between the real and fake samples is utilized to calculate the anomaly score, while the threshold value range is defined as t in [0, 1].

However, these algorithms usually only apply to images/videos/speech, which can not fit the feature information of the traffic well. In short, GAN models have shown excellent ability in generating actual instances, especially on image/video data. GAN-based AD methods generally detect anomalous cases poorly generated from the latent space. Many existing GAN-based models and theories can be used for anomaly detection. However, the currently GAN-based AD methods still have the following limitations. First, most existing GAN-based AD models are only suitable for images/videos, not for network traffic. Second, existing GAN-based AD algorithms are not capable of extracting and retaining network information well, which results in a weak ability to represent features in the mapping of normal network data to low-dimensional latent features, hence, leading to a poorly discriminative ability for network anomalies.

III. The Proposed Method

In this section, we will introduce the process of the proposed FlowGANAnomaly based on GAN in detail, as shown in Figure 1.

First, the raw packets will be preprocessed to construct the flow attribute matrix (FAM), which is the footstone of FlowGANAnomaly as the model input. FAM is composed of several attributes which will be comprehensively demonstrated in the following section. Our model consists of a deconvolutional decoder, a flow encoder, a flow decoder, and three convolutional encoders. The input vector, in G, analogous to the seed of a pseudorandom number, successively passes through a flow encoder, a convolutional encoder-decoder-encoder, a

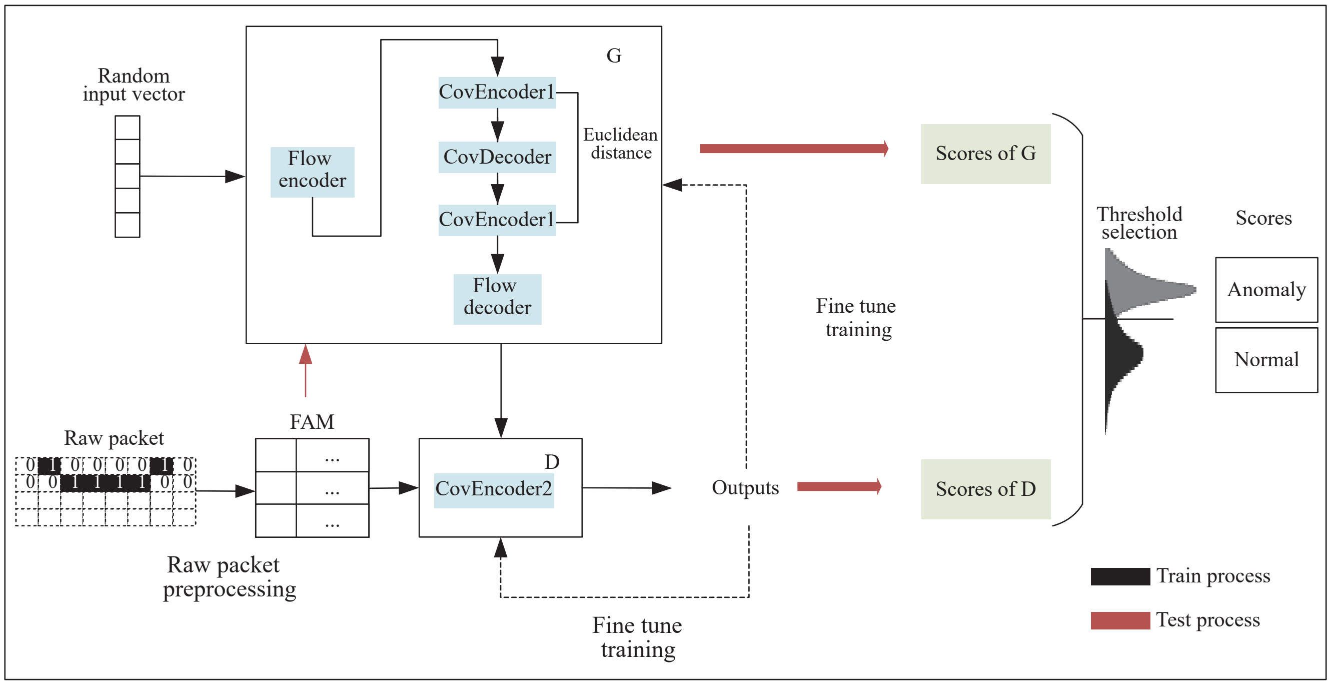

Figure 1 The framework of FlowGANAnomaly.

flow decoder, and then becomes the output vector. We use a D to estimate the output of G and the real flow. Furthermore, the model measures the binary cross entropy (BCE) between the target and the output and updates the gradient. After the adversarial learning based on FAM, G and D can calculate the anomaly score, of which G's score is computed on the basis of a difference between two convolutional encoders, thus detecting the deviation from the network traffic flow. Finally, the elaborated threshold, obtained by extensive experiments, is employed to determine whether the network flow is abnormal or not based on the deviation calculated by the anomaly scoring algorithms. In order to describe the methods easily, we define the network-traffic-related notations as below:

1) Packet: A packet is a basic unit in flow. Set $P=\{p_1,p_2,\dots,p_{i-1},p_i\}$, $0\le i\le M$, M is the number of packets.

2) Flow: Source address, destination address, source port, destination port, and TCP/UDP protocol five-tuple consists of flow, which is unidirectional to differs from Session. Set $F=\{f_1,f_2,\dots,f_{i-1},f_i\}$, $0\le i\le K$, K is the number of flows.

3) Session: Bi-directional flows are two flows of the same tuples of upstream and downstream. Set $S=\{s_1,s_2,\dots,s_{i-1},s_i\}$, $0\le i\le N$, N is the number of sessions.

1. Model input design

As shown in Table 1, the vector of flow features (FF) is the basic unit of FAM, including packet-level (direction, packet size, etc.), flow-level (flow size, flow duration, etc.), and statistical features (mean packet size, mean inter-arrival time, etc.). FAM is composed of FFs: $FAM=\{FF_1,FF_2,\dots,FF_{i-1},FF_i\}$. The number of dimensions of FF is defined as FN.

Table 1 The example of flow attribute matrix (FAM)

Packet-level		Flow-level		Statistics level	
Pkt len	...	Flow duration	...	Mean (Pkt len)	...
6	...	34	...	22.666	...
...	...	...	...	...	...
31	...	52	...	12.55	...

Taking the CIC-IDS2017 dataset as an example, we divided the dataset into three levels by definition and finally formed the FAM based on the FFs after the raw packets preprocessing. Table 1 shows the examples of the FAM. The complete FAM of a dataset is composed of multiple pieces of FAMs. Since we define a batch size of 64, the dimension of a small FAM is 64 × FN. Define the set of FAM as H. In this paper, H contains four kinds of data sets, namely, $H=\{FAM_1,FAM_2,FAM_3,FAM_4\}$. Figure 2 shows the new FAM formed after normalization and one-hot encoding. The new FAM is input into the FlowGANAnomaly model for the subsequent training.

2. Model design

The model will uniformly map the multidimensional feature space of the FAMs to the feature space of fixed dimensions to get the mapped new FAM. The model uses a learnable linear layer as the feature mapping function ϕ by the weight matrix parameter $W\in\mathbf{R}^{d\times D}$, to obtain a new FAM with powerful representation capabilities. Simple linear transformations can also ensure the scalability of the model. The data object x is transferred to a new array v by the feature mapping function, $v=\phi(x)$. $\phi(x)$ is defined as equation (1).

$$\phi(x)=\begin{pmatrix} W(1,1)x(1) & \dots & W(1,D)x(D) \\ \vdots & \ddots & \vdots \\ W(d,1)x(1) & \dots & W(d,D)x(D) \end{pmatrix} \quad (1)$$

where W is the matrix of neuron parameters and x is the matrix of eigenvalue in flow encoder, where $W(i,j)$ represents the element in the i-th row and j-th column of W, and $x(i)$ represents the i-th element of x. Each dimension can be regarded as a linear pattern (combination) of the original feature space. The mapping function $\phi(x)$ will effectively aggregate the network flow features into a new FAM and then pass it to the convolutional encoder (CovEncoder) for feature space mapping.

The FAM newly generated in CovEncoder uses a convolutional layer and performs batch normalization (BN) and activation function LeakyReLU() respectively. By compressing the FAM v into a hidden space z, the dimensionality of v is reduced. At the same time, the model obtains the most representative hidden features. z is also called the hidden feature of x, Z is defined as equation (2), $Z\in\mathbf{R}^d$. Assuming that its dimension is the

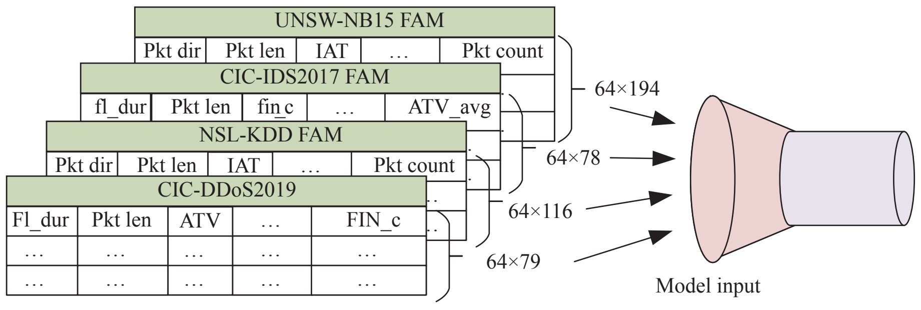

Figure 2 FlowGANAnomaly input design.

smallest, the space can best reflect its features of x . As shown in Figure 3, the module has four convolutional layers. The kernel size of each layer is 4. The input dimension of the module is 64 × 1 × 32. The output dimension of the module is 64 × 100 × 1, where the batch size is 64.

$$Z = \frac{v - E[\hat{v}]}{\sqrt{\mathrm{Var}[v] + \varepsilon}} \tag{2}$$

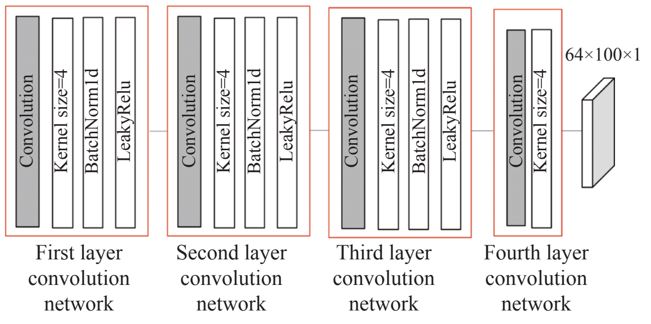
Convolution
Kernel size=4
BatchNorm1d
LeakyRelu
64×100×1
First layer convolution network
Second layer convolution network
Third layer convolution network
Fourth layer convolution network

Figure 3 The parameters of the convolutional encoder1.

The decoder part of the G uses the architecture of the DCGAN [36] generator. The network features are then flattened and concatenated to form new vector. The decoder uses a convolutional transposed layer, an activation function (ReLU) and BN. Tanh layer is added at the end to decode the hidden space into the generated embedding space. The specific parameters of the decoder are shown in Figure 4.

The module has four deconvolution layers. The kernel size of each layer is 4. The input dimension of the module is 64 × 100 × 1. The output dimension of the module is 64 × 1 × 32, where the batch size is 64. This method reconstructs the hidden space z and reconstructs v as $\hat{v}$. On this basis, the dimensionality is expanded using linear changes through the mapping function $\psi(\hat{v})$ to become a new space $\hat{x}$. $\psi(\hat{v})$ is defined as equation (3). The whole process is called space backtracking, where $\hat{v} = G_d(z)$.

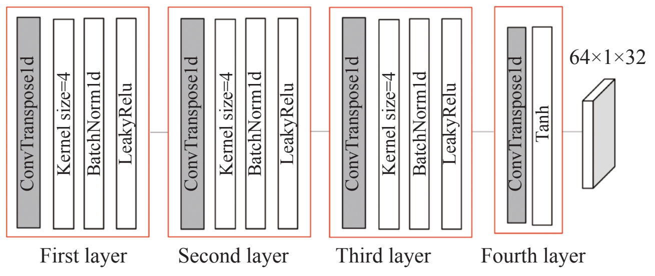

Figure 4 The parameters of the deconvolutional decoder (CovDecoder).

$$\psi(\hat{v}) = \begin{pmatrix} W(1,1)v(1) & \dots & W(1,D)v(D) \\ \vdots & \ddots & \vdots \\ W(d,1)v(1) & \cdots & W(d,D)v(D) \end{pmatrix} \tag{3}$$

where $W(i,j)$ denotes the element of the i-th row and j-th column of the matrix W. $\hat{v}_i$ denotes the i-th element of the space $\hat{v}$. Each dimension can be regarded as a linear pattern of the original feature space. We use the obtained reconstructed space for comparison with the original space. The generator can understand the contextual information about the input data. The loss function for normal FAMs and generated FAMs is defined as equation (4). The training process of the generator is shown in Figure 5.

$$\mathrm{loss}_{\mathrm{con}} = E_{x\text{-Euclidean}} \| x - \psi(\hat{v}) \|_1 \tag{4}$$

In the part of convolutional encoder1, the model compresses the FAM $\hat{v}$ reconstructed by the neural network. $\hat{v}$ reconstruct the flow space compression to find its characteristics, specifically expressed: $\hat{z} = E(\hat{v})$. The dimension of the hidden space vector $\hat{z}$ is the same as the dimension of z so that the distance can be calculated later. This sub-network is special in the proposed method, where it can represent the hidden features in the reconstruction space. Unlike previous methods [27] based on VAE, the distance between the latent space z and the original v is minimized by hidden features $\hat{z}$. The sub-network minimizes the distance through parameterized

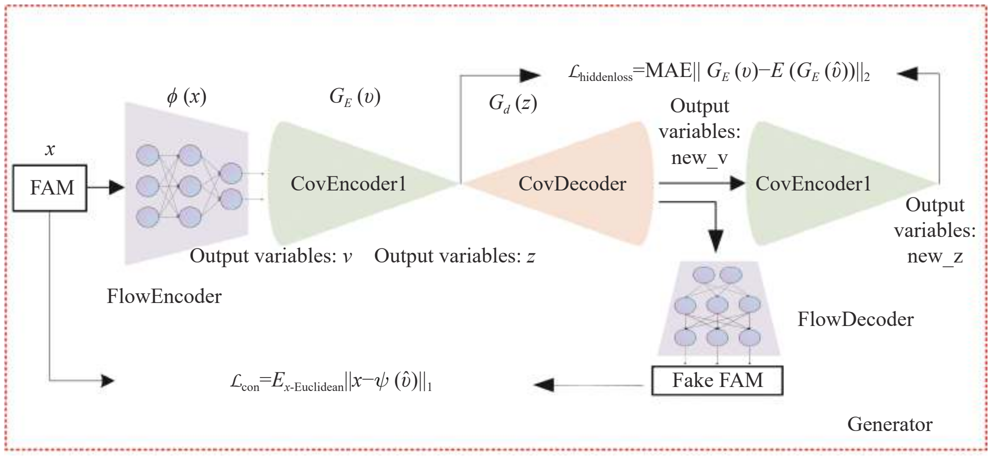

Figure 5 The architecture for the generator.

explicit learning. MAE depicts the average distance of the two variables. This model uses a second-norm to calculate the distance between two hidden vectors. The loss function of the hidden features is defined as equation (5).

$$\text{loss}_{\text{hidden}} = \text{MAE}\left\|G_E(v) - E\left(G_E(\hat{v})\right)\right\|_2 \tag{5}$$

The training process of the discriminator is shown in Figure 6. The goal of discriminators is to classify the input and output as True or False, respectively. This sub-network is a standard discriminator network introduced in DCGAN. The discriminant loss function is defined as equation (6). The parameters of convolutional encoder2 are shown in the Figure 7. The components of the model are similar to the previous ones. Nevertheless, the dimensions of the input and output of the module are different. The input dimension of the module is 64 × 100 × 1. The output dimension of the module is 64 × 1 × 32.

$$\text{loss}_{\text{adv}} = E_{x-p(x)}\left\|f(x) - E_{x-p(x)} f(G(x))\right\|_2 \tag{6}$$

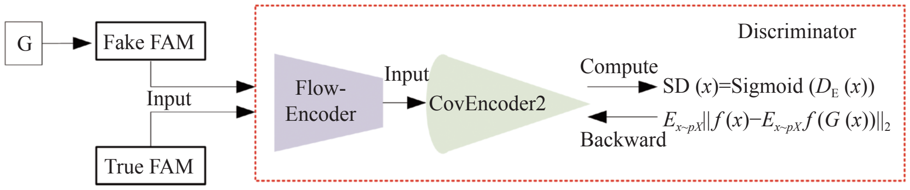

Figure 6 The architecture for the discriminator.

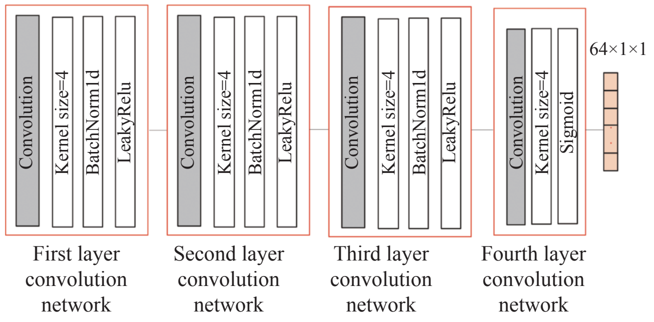

Figure 7 The parameters of the convolutional encoder2.

loss_{con} is the Euclidean distance between original samples and generated samples. $\text{loss}_{\text{hidden}}$ is the Euclidean distance between two hidden space vectors. loss_{adv} is the loss of D in GAN. We adjust the weighting parameters to determine the overall objective function. Then we end up with the goal of minimizing the loss function, $\min(\text{loss}_{\text{total}})$.

$$\text{loss}_{\text{total}} = w_{\text{adv}}\,\text{loss}_{\text{adv}} + w_{\text{con}}\,\text{loss}_{\text{con}} + w_{\text{hidden}}\,\text{loss}_{\text{hidden}} \tag{7}$$

3. Model training

We assume that the generator fails to reconstruct the network flow when an outlier flow passes through the generator, which is because the network is trained on normal samples. The generator's parametric modelling is not suitable for generating outlier samples. The reconstruction failure $\hat{v}$ means that the encoder network $G_E(\hat{v})$ cannot be mapped to a vector $\hat{z}$ typically, resulting in a large distance between z and $\hat{z}$. See Algorithm 1 for details.

The pseudocode is shown in Algorithm 1. The G firstly reads the input data x, where $x \in FAM \mid$ (batchsize × mum), forwards it to a layer of dimensional compression fully connected v, and then passes it to G_E. Using a convolutional layer and then performing BN and activation function, respectively, the dimension of v is reduced by compressing it into a vector. These features represent normal network flow. The decoder part of the G uses a deconvolution layer, activation functions ReLU, and BN together with a Tanh layer at the end. This method scales the vector z and reconstructs the flow v as $\hat{v}$. The algorithm calculates the loss according to equation (4).

Algorithm 1

```
Require:
  FAM will be divided into several batches, $FAM = \{FF_1, FF_2, \ldots, FF_{m-1}, FF_m\}$;
  $X$ represents data entered at one time, $X = \{FF_1, FF_2, \ldots, FF_{batchsize-1}, FF_{batchsize}\}$;
  $niter$ is the number of iterations.
 1: for $i \leftarrow 1$ to $niter$ do
 2:     Use generator to calculate $\hat{x}$ and $\hat{v}$;
 3:     $v = \phi(x), z = G_E(v), \hat{v} = G_d(z), \hat{x} = \psi(\hat{v})$;
 4:     Compute loss of $\hat{x}$: $\text{loss}_{\text{con}} = E_{x\text{-Euclidean}}\|x - \psi(\hat{v})\|_1$;
 5:     Find compress: $\hat{z} = G_E(\hat{v})$;
 6:     Compute hidden loss:
        $\text{loss}_{\text{hidden}} = MAE\left\|G_E(v) - E\left(G_E(\hat{v})\right)\right\|_2$
 7:     Send the fake FAM to the discriminator;
 8:     Compute loss:
        $\text{loss}_{\text{adv}} = E_{x-p(x)}\left\|f(x) - E_{x-p(x)} f(G(x))\right\|_2$
 9:     Compute total loss:
        $\text{loss}_{\text{total}} = w_{\text{adv}}\,\text{loss}_{\text{adv}} + w_{\text{con}}\,\text{loss}_{\text{con}} + w_{\text{hidden}}\,\text{loss}_{\text{hidden}}$
10:     Compute gradient directions: $g_k = \nabla L(x_k)$;
11:     Update parameters;
12: end for
```

The second sub-network is the CovEncoder, which compresses the network flow data $\hat{v}$ reconstructed by the neural network. CovEncoder is compressed downward to $\hat{v}$, and its features representation $\hat{z} = G_E(\hat{v})$ is found. At this point, the algorithm calculates the loss according to equation (5). The dimension of the vector $\hat{z}$ is the same as the dimension of z so that the distance can be calculated. The third sub-network is the D, whose goal is to classify input x and output $\hat{x}$ as true or false, respectively, according to equation (6). The algorithm eventually minimizes equation (7) by gradient.

4. Anomaly scores

The anomaly score consists of two components. A part of the anomaly score is the G's score, another part is the D's score. A part of the anomaly score calculation in this paper will use the calculation of the Euclidean distance between two hidden vectors, as shown in the following equation. To facilitate the threshold selection later, we first normalize the obtained Euclidean distance and finally obtain $SG(x)$. $SG(x)$ represents G's anomaly score.

$$\begin{cases} SG(x) = E_{x\text{-Euclidean}} \left\|G_E(v) - E\left(G_E(\hat{v})\right)\right\|_2 \\ SG_i(x) = \dfrac{SG_i - SG_{\min}(x)}{SG_{\max}(x) - SG_{\min}(x)} \end{cases} \tag{8}$$

In order to take full advantage of the adversarial training of the generator and the discriminator, another part of the anomaly score comes from the discriminator, which reflects the anomaly score of the test sample more comprehensively. $SD(x)$ is the D's anomaly score.

$$SD(x) = \mathrm{Sigmoid}\left(D_E(x)\right) \tag{9}$$

Therefore, for a test sample x, the anomaly score $A(x)$ is defined by the following equation (10), where is λ adjusted by the validation set according to the different data sets.

$$A(x) = \lambda SG(x) + (1-\lambda) SD(x) \tag{10}$$

5. Threshold selection

The threshold selection method is as follows. The first is the choice of validation set data. The experiment selects a certain proportion of normal and malicious traffic in an orderly manner. Then, the validation set is fed into the anomaly detection model. Calculate the reconstruction distance of each normal flow sample and malicious flow sample in the validation set according to equation (10). Finally, the probability density and kernel equation (13) are obtained according to the reconstruction distance of the verification set, and the threshold is determined.

In validation set, there are n outliers scores as follows: $A(x) = \{a_1, a_2, \ldots, a_{i-1}, a_i\}$. Assume that the cumulative distribution function of the sample data is $F(a)$.

$$F\left(a_{i-1} < a < a_i\right) = \int_{a_{t-1}}^{a_t} f(a) da \tag{11}$$

And the probability density function is $f(a)$.

$$f\left(a_i\right) = \lim_{h \to 0} \frac{F\left(a_i + h\right) - F\left(a_i - h\right)}{2h} \tag{12}$$

Introduce the empirical distribution function of the cumulative distribution function and substitute this function into $f(a)$, after determining h, $f(a)$ can be transformed into

$$\begin{aligned} f(a) &= \frac{1}{2nh} \sum_{i=1}^{n} 1_{a-h} < a_i < a + h \\ &= \frac{1}{2nh} \sum_{i=1}^{n} K\left(\frac{|a - a_i|}{h}\right) \end{aligned} \tag{13}$$

We calculate the two functions of normal and anomaly flows, using the kernel density function, respectively, and get the intersection. Turn the intersection into a threshold.

IV. Evaluation

1. Evaluation settings and chosen datasets

The data sets we select need to meet the experimental requirement. The four classic NIDS data sets, namely, NSL-KDD [37], CIC-IDS2017 [38], UNSW-NB15 [39], and CIC-DDoS2019 [40], are the most popular benchmark datasets to evaluate the performance of NIDS algorithms.

From Table 2, malicious flows account for a relatively large proportion of UNSW-NB15 and CIC-DDoS2019. In order to be close to the actual network scenario, we use a ratio of normal flow to malicious flow of 100 to 1 to evaluate the FlowGANAnomaly.

Table 2 Summary of datasets used for evaluation (FE: features)

Dataset	Size	% Attacks	Format
NSL-KDD	20.7 MB	48.12	42FE
UNSW-NB15	45.4 MB	63.91	46FE
CIC-DDoS2019	8.14 GB	85.92	79FE
CIC-IDS2017	848 MB	19.68	78FE

The experimental environment is AMD Ryzen 3600, 16GB RAM, NVIDIA GTX 1660, CUDA 7.5, CDNN10.5. In this paper, Python3 is the primary programming language. The following is a description of the evaluation metrics: Precision, Recall, F1, Accuracy, and AUC as evaluation metrics.

Marco average, means that each type of sample is given equal weight. For example, in this article, the macro average accuracy index is defined as

$$P = \frac{P_{\text{normal}} + P_{\text{malware}}}{2} \tag{14}$$

Weighted average, is to use the proportion of the sample size of each category in the total number of samples in all categories as the weight. In this article, the weighted average accuracy index P is defined as

$$P = \frac{N_{\text{normal}}}{N_{\text{normal}} + N_{\text{malware}}} \times P_{\text{normal}} + \frac{N_{\text{malware}}}{N_{\text{normal}} + N_{\text{maiware}}} \times P_{\text{malware}} \quad (15)$$

Time-complexity-related metrics like detection time or training time are not included in this paper because we think those are highly dependent on the hardware resources whether training or detecting.

2. Ablation study

We conducted an ablation study using UNSW-NB15 dataset in this subsection. Table 3 shows that if the model is processed by adding a flow encoder, the representative elements will be represented as a matrix in the subspace. Then the feature space mapping will be performed.

Table 3 Improved model comparison

Model progressive		Original model	FlowGANAnomaly
Precision	Weighted avg.	0.7647	0.9859
	Marc. avg.	0.6890	0.5102
Recall	Weighted avg.	0.3852	0.7037
	Marc. avg.	0.5063	0.7354
F1	Weighted avg.	0.2236	0.8384
	Marc. avg.	0.2881	0.4471
AUC		0.7808	0.8530

From Table 3, the results are not ideal if only the module used for image anomaly detection is directly migrated to detect network traffic. The weighted average of F1 for our proposed model is 0.6148 higher, and the AUC is increased by 0.0722. Therefore, our proposed model is 0.6148 higher and can significantly improve network anomaly detection performance.

The experiment compares the classical AD model based on GAN to detect anomaly flows in a network environment. Since models need to be evaluated globally, AUC has become a leading model evaluation indicator. FlowGANAnomaly performs better than the other two models in Table 4. On the UNSW-NB15 data set, the AUC value of this model reaches 0.8530, which is 0.1315 higher than f-AnoGAN. Our model proposed is more effective than EGBAD. The CICIDS2017 data set has a vast amount of data. Hence, with sufficient training, although the AUC value of FlowGANAnomaly is the highest, the difference between the three types of GAN models is not significant. Also, on the NSL-KDD data set, our model has a higher AUC than the other two models. Therefore, we can conclude that the FlowGANAnomaly model clusters the features by adding a flow-encoder, which can refine the feature form of the data. At the same time, our model uses the loss function to optimize and constrain the hidden features. In the following comparison stage, we use FlowGANAnomaly as the representative of the GAN series model to compare with other family's algorithm models.

Table 4 GAN compared in four datasets

Model (AUC)	FlowGANAnomaly	f-AnoGAN	EGBAD
UNSW-NB15	0.8530	0.7215	0.5638
CIC-IDS2017	0.7432	0.6929	0.7365
NSL-KDD	0.9801	0.9572	0.9372
CIC-DDoS2019	0.7883	0.5535	0.7651

3. Performance evaluation

The experiment selects representative algorithm data sets for testing in different algorithm families. In Table 5, as AUC is the overall indicator of the evaluation model, our algorithm has a better performance on each data set than other algorithms. In the NSL-KDD data set in Figure 8, although FlowGANAnomaly shows the best effect, it is not much different from other algorithms. Since the NSL-KDD dataset was collected earlier, this data set does not have the timeliness of the current network. In addition, NSL-KDD datasets contain relatively a small amount of samples, each model can learn the dataset's characteristics and achieve better results. From Figure 8, the AUC of our algorithm is 0.8530, which is second only to LOF and shows a stable performance on UNSW-NB15 dataset. The CICIDS2017 dataset is a relatively new data set of the four data sets and is also an extensive data set. Traditional outlier algorithms tend to fail when training large amounts of data. Rather, DAD models, such as AE, VAE, and FlowGANAoamly, also show their advantages. The AUC value of the deep learning model achieves 0.73, while other traditional AD algorithms, such as LOF, SVM, and so on, are between 0.5 and 0.6. Thus, the proposed FlowGANAnomaly in this article is still in a leading position in DAD algorithms.

In the field of anomaly detection, a large emphasis has been devoted to reducing false negatives, instead of decreasing true negatives. Therefore, the number of false flows determined as proper flows should be as small as possible. So we consider that the benign accuracy rate and the negative recall rate have great weight. This experiment shows the weighted average precision rate and the macro average recall rate.

FlowGANAnomaly's accuracy rate achieves 98% because the ratio of normal and abnormal flow in the training set is 100:1. Therefore, most of the normal flow can be well-identified. In addition, the reason for the low recall rate in each dataset is that lots of malicious traffic are identified as normal traffic. Therefore, whether anomaly detection can detect anomalous flow, the recall index is essential. The recall of FlowGANAnomaly, as a representative of deep learning algorithms, is higher than other algorithms on the CICIDS2017 data set. Therefore, our model has a relatively strong ability to detect anomaly flows.

From Table 5, the LOF performs exceptionally well

Table 5 Evaluation results (including OCSVM, Isolation Forest, PCA, LOF, AE, VAE, and FlowGANAnomaly)

Model	Dataset	Precision		Recall		F1		Accuracy	AUC
		Weighted avg.	Marc. avg.	Weighted avg.	Marc. avg.	Weighted avg.	Marc. avg.		
OCSVM	CICIDS2017	0.9831	0.5031	0.5545	0.5773	0.7044	0.3686	0.5545	0.6341
	NSL-KDD	0.9899	0.5433	0.9059	0.9050	0.9423	0.5550	0.9059	0.9680
	UNSW-NB15	0.9831	0.5058	0.7730	0.6039	0.8631	0.4538	0.7730	0.8339
	CICDDoS2019	0.8286	0.4808	0.7970	0.4740	0.8123	0.4762	0.7970	0.5328
Isolation forest	CICIDS2017	0.9838	0.5096	0.8266	0.6402	0.8961	0.4767	0.8266	0.6500
	NSL-KDD	0.9899	0.5427	0.9041	0.9060	0.9413	0.5535	0.9041	0.9673
	UNSW-NB15	0.9823	0.5051	0.8198	0.5760	0.8921	0.4677	0.8198	0.5021
	CICDDoS2019	0.0007	0.0088	0.0242	0.3144	0.0013	0.0172	0.0242	0.2854
PCA	CICIDS2017	0.9806	0.5081	0.9866	0.5030	0.9836	0.5037	0.9866	0.6795
	NSL-KDD	0.9901	0.5422	0.9015	0.9106	0.9398	0.5519	0.9015	0.9603
	UNSW-NB15	0.9831	0.5067	0.8014	0.6077	0.8808	0.4643	0.8014	0.5904
	CICDDoS2019	0.8270	0.4763	0.7980	0.4686	0.8121	0.4715	0.7980	0.5399
LOF	CICIDS2017	0.9814	0.5418	0.9882	0.5100	0.9846	0.5149	0.9882	0.5483
	NSL-KDD	0.9893	0.5336	0.8816	0.8768	0.9285	0.5319	0.8816	0.9118
	UNSW-NB15	0.9879	0.5242	0.8573	0.8113	0.9144	0.5092	0.8573	0.8775
	CICDDoS2019	0.9704	0.5321	0.7106	0.7736	0.8086	0.4780	0.7106	0.7629
AE	CICIDS2017	0.9848	0.5132	0.8385	0.6823	0.9032	0.4859	0.8385	0.7357
	NSL-KDD	0.9901	0.5184	0.7509	0.8603	0.8484	0.4639	0.7509	0.9628
	UNSW-NB15	0.9753	0.4968	0.1969	0.4495	0.3181	0.1691	0.1969	0.5456
	CICDDoS2019	0.8345	0.4983	0.5506	0.4948	0.6456	0.4206	0.5506	0.5630
VAE	CICIDS2017	0.9841	0.5420	0.9614	0.6447	0.9720	0.5610	0.9614	0.7475
	NSL-KDD	0.9903	0.5511	0.9192	0.9236	0.9499	0.5714	0.9192	0.9735
	UNSW-NB15	0.9832	0.5154	0.9125	0.6206	0.9453	0.5111	0.9125	0.6455
	CICDDoS2019	0.8279	0.4549	0.9093	0.4997	0.8667	0.4763	0.9093	0.5520
FlowGANAnomaly	CICIDS2017	0.9841	0.5152	0.8840	0.6807	0.9295	0.5030	0.8840	0.7432
	NSL-KDD	0.9888	0.5303	0.8747	0.8535	0.9245	0.5244	0.8747	0.9801
	UNSW-NB15	0.9859	0.5102	0.7037	0.7354	0.8384	0.4471	0.7354	0.8530
	CICDDoS2019	0.8590	0.6066	0.9012	0.5275	0.8727	0.5316	0.9012	0.7883

in the experiments. Accordingly, DAD algorithms cannot effectively learn the features of the network traffic, limited by the size of the UNSW-NB15 dataset. Because comparing the density between the sample points, LOF is more advantageous in learning small-scale data samples. However, LOF will be more unstable when converting data sets or increasing the size. On the CICIDS2017 dataset, the AUC value of LOF is only 0.5483, indicating that the algorithm cannot obtain density bias based on features when training large-scale data, resulting in the algorithm not fitting.

4. Experiment discussion

This experiment chooses two data sets, and plots box figures to show the anomaly scores for each type of malicious flow. Figure 9 demonstrates that the model does not work well on Exploits, Fuzzers, and Reconnaissance. Fuzzers and Reconnaissance are two attacks that have something in common. Fuzzers attempt to suspend a program or network by providing randomly generated data. Reconnaissance, on the other hand, implements the attack by simulating information gathering. Both types of attacks are generated by simulation, which is inherently random. Therefore, the features of these two types of flow are similar to normal flow, and it is difficult for the model to distinguish them. Exploits are vulnerable attacks. The attacker knows the security issues in the operating system or software and exploits this knowledge by exploiting the vulnerability. These vulnerabilities are generated along with the flow of software applications. So the flow features of these vulnerabilities are similar to the typical flow features in the software operation process, making the model difficult to identify.

Attacks like Heartbleed are loopholes in the “ssh” protocol, and too few samples are available for testing. Therefore, the anomaly score is not good. On the data set UNSW-NB15, the normal samples for training are too few, so the anomaly score of normal flow will be slightly higher than the CICIDS2017 data set. Brute-force cracking of this attack is often done through repeated trial

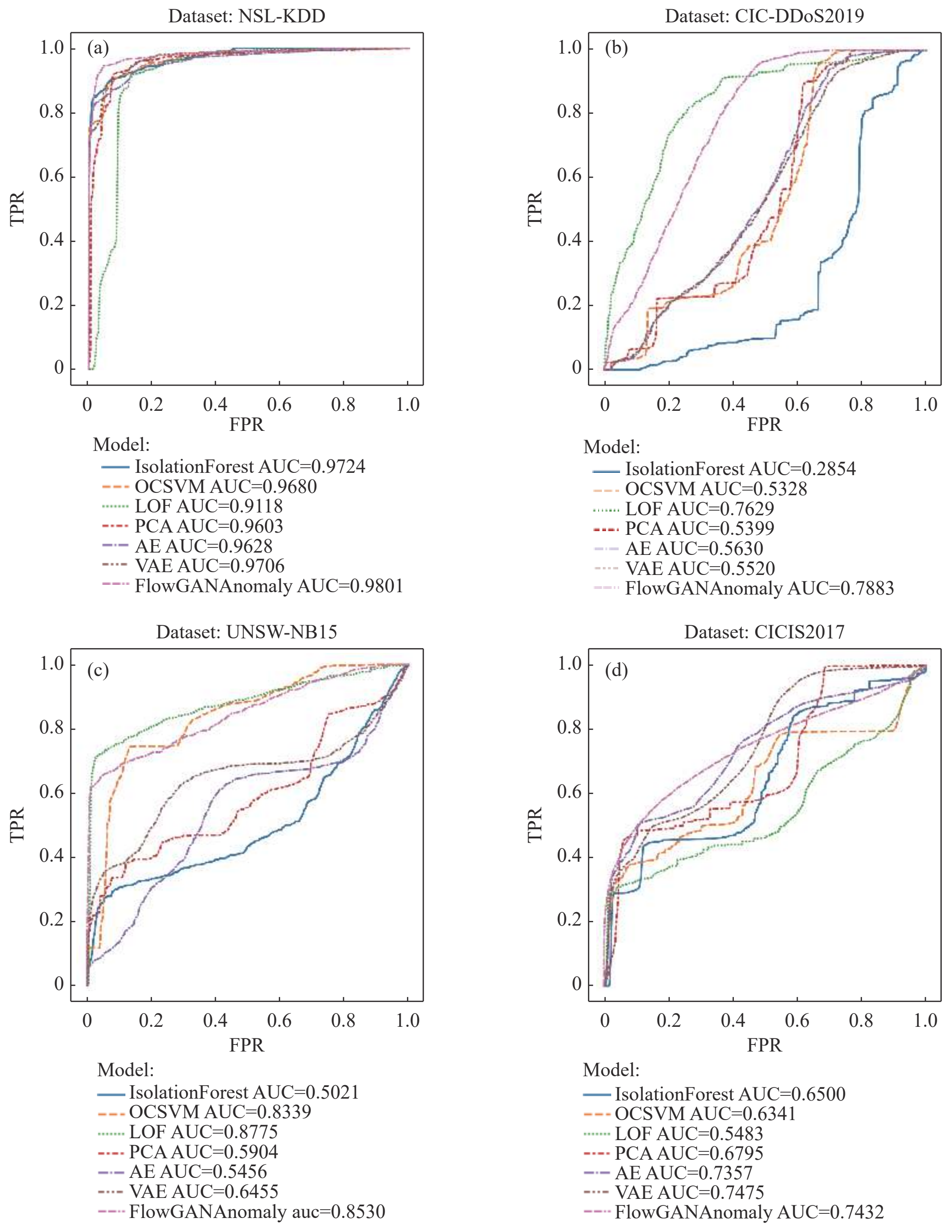

Figure 8 Comparison of ROC curves for four datesets. a) NSL-KDD; b) CIC-DDS2019; c) UNSW-NB15; d) CIC-IDS2017.

and error by enumerating exhaustive methods. Therefore, there will be apparent manifestations in features such as flow duration that allow the model to easily distinguish the difference from the normal flow.

In the last part of the experiment, we also performed model interpretability work on the UNSW-NB15 dataset. We use the ATON [41] method to explain the outlier results of anomaly detection. The ATON algorithm is used for post hoc interpretation, where the obtained anomalous data are put into the ATON algorithm, thus explaining the contribution of the model for each feature in the dataset.

There are many samples of DoS attacks and DDoS attacks in the UNSW-NB15 dataset. DoS attack traffic is characterized by a one-to-one approach, sending a large amount of data to the network server, flooding the website server with a large amount of information that requires a reply, consuming network bandwidth or system resources, and causing the network or system to be overloaded. DDoS, characterized by multiple different hosts, sends a large amount of data to the network server, causing a large number of requests to flood the server, making the server paralyzed and unable to work properly. “ct-srv-dst” refers to the number of connections that included the same server and destination address in the last 100 connections. Therefore, when two types of attacks occur, a large number of requests in the server cause “ct-srv-dst” to soar, which can be classified as ab-

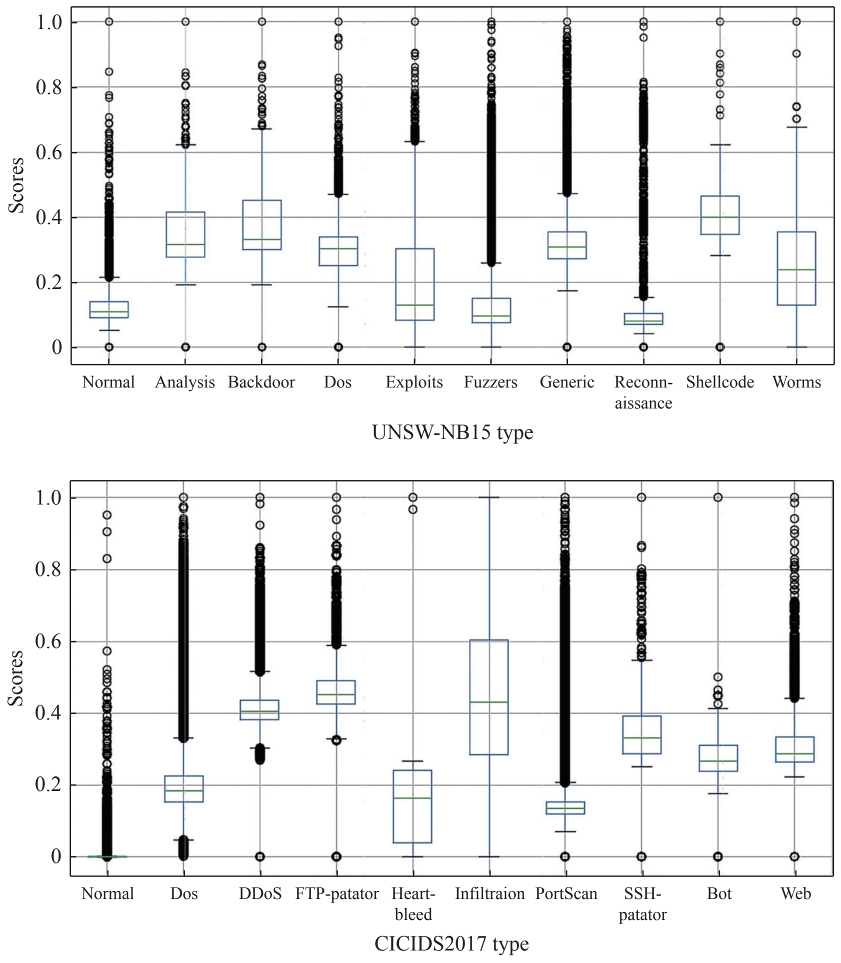

Figure 9 Box-line figure of FlowGANAnomaly on two datasets (UNSW-NB15 and CIC-IDS2017).

normal traffic by the model. The time of existence of "sttl " as source-to-destination traffic also differs from normal traffic when an attack occurs. Figure 10 shows that features such as "sttl" and "ct-srv-dst" play a more important role in the model.

V. Conclusion and Future Work

In this paper, we propose an anomaly detection model called FlowGANAnomaly for detecting anomalous traffic in NIDS. FlowGANAnomaly maps and learns G's hidden flow feature space through different feature datasets to better capture the normality of network traffic data, instead of learning diverse types of labeling data, therefore solving the problem of complex data labelling. We conducted several experiments comparing existing machine learning algorithms and existing deep learning methods on four public dataset. The evaluation results show that FlowGANAnomaly can significantly improve the performance of anomaly-based NIDS.

However, research on unsupervised flow detection algorithms is rare and immature. Although unsupervised algorithms have made some achievements in this paper, there is still much room for improvement. In addition, intrusion detection datasets in this experiment are diverse and large in scale, there is still a gap between them and the real scene. In the future, our work will continue to build large datasets in real-world settings.

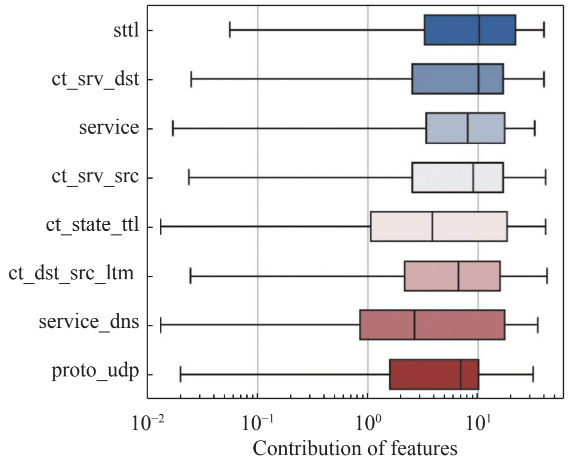

Figure 10 Interpretation of the model and important features in UNSW-NB15.

Acknowledgement

This work was supported by the National Natural Science Foundation (Grant No. 61972211), the National Key Research and Development Project (Grant No. 2020 YFB1804700), and the Future Network Innovation Research and Application Projects (Grant No. 2021FNA02 006).

References

[1] O. Depren, M. Topallar, E. Anarim, *et al.*, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.

[2] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, TX, USA, pp. 427–438, 2000.

[3] G. S. Pang, C. H. Shen, L. B. Cao, *et al.*, "Deep learning for anomaly detection: a review," *ACM Computing Surveys*, vol. 54, no. 2, article no. 38, 2022.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, article no. 15, 2009.

[5] M. Ahmed, A. N. Mahmood, and J. K. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[6] A. Creswell, T. White, V. Dumoulin, *et al.*, "Generative adversarial networks: an overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[7] Z. P. Qiang, L. B. He, F. Dai, *et al.*, "Image inpainting based on improved deep convolutional auto-encoder network," *Chinese Journal of Electronics*, vol. 29, no. 6, pp. 1074–1084, 2020.

[8] C. Qin and X. G. Gao, "Spatio-temporal generative adversarial networks," *Chinese Journal of Electronics*, vol. 29, no. 4, pp. 623–631, 2020.

[9] F. Falcão, T. Zoppi, C. B. V. Silva, *et al.*, "Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, Limassol, Cyprus, pp. 318–327, 2019.

[10] E. Schubert, A. Koos, T. Emrich, *et al.*, "A framework for clustering uncertain data," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1976–1979, 2015.

[11] P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers-a tutorial," *ACM Computing Surveys*, vol. 54, no. 6, article no. 128, 2022.

[12] M. M. Breunig, H. P. Kriegel, R. T. Ng, *et al.*, "LOF: Identifying density-based local outliers," in *Proceedings of 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, TX, USA, pp. 93–104, 2000.

[13] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, *et al.*, "PCA-based multivariate statistical network monitoring for anomaly detection," *Computers & Security*, vol. 59, pp. 118–137, 2016.

[14] R. Kwitt and U. Hofmann, "Unsupervised anomaly detection in network traffic by means of robust PCA," in *Proceedings of 2007 International Multi-Conference on Computing in the Global Information Technology*, Guadeloupe, French Caribbean, pp. 37–37, 2007.

[15] H. P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, pp. 444–452, 2008.

[16] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, Chicago, IL, USA, pp. 8–15, 2013.

[17] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining*, Pisa, Italy, pp. 413–422, 2008.

[18] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, article no. 3, 2012.

[19] X. S. Wei, H. J. Ye, X. Mu, *et al.*, "Multi-instance learning with emerging novel class," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 2109–2120, 2021.

[20] L. L. Wang, B. Q. Wang, P. P. Zhao, *et al.*, "Malware detection algorithm based on the attention mechanism and ResNet," *Chinese Journal of Electronics*, vol. 29, no. 6, pp. 1054–1060, 2020.

[21] D. W. Zhou, H. J. Ye, and D. C. Zhan, "Learning placeholders for open-set recognition," in *Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, pp. 4401–4410, 2021.

[22] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, *et al.*, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[23] D. W. Zhou, Y. Yang, and D. C. Zhan, "Learning to classify with incremental new class," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2429–2443, 2022.

[24] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, pp. 1652–1658, 2016.

[25] M. Gharib, B. Mohammadi, S. H. Dastgerdi, *et al.*, "AutoIDS: Auto-encoder based method for intrusion detection system," *arXiv preprint*, arXiv: 1911.03306, 2019.

[26] Y. Mirsky, T. Doitshman, *et al.*, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proceedings of the 25th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, 2018.

[27] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020.

[28] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *Proceedings of the 7th Conference on Information and Knowledge Technology*, Urmia, Iran, pp. 1–5, 2015.

[29] N. Shone, T. N. Ngoc, V. D. Phai, *et al.*, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[30] S. Longari, D. H. N. Valcarcel, M. Zago, *et al.*, "CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1913–1924, 2021.

[31] T. Schlegl, P. Seeböck, S. M. Waldstein, *et al.*, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proceedings of the 25th International Conference on Information Processing in Medical Imaging*, Boone, NC, USA, pp. 146–157, 2017.

[32] H. Zenati, C. S. Foo, B. Lecouat, *et al.*, "Efficient GAN-based anomaly detection," *arXiv preprint*, arXiv: 1802.06222, 2018.

[33] T. Schlegl, P. Seeböck, S. M. Waldstein, *et al.*, "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Medical Image Analysis*, vol. 54, pp. 30–44, 2019.

[34] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANomaly: Semi-supervised anomaly detection via adversarial training," in *Proceedings of the 14th Asian Conference on Computer*, Perth, Australia, pp. 622–637, 2019.

[35] I. Siniosoglou, P. Radoglou-Grammatikis, G. Efstathopoulos, *et al.*, "A unified deep learning anomaly detection and classification approach for smart grid environments," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1137–1151, 2021.

[36] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint*, arXiv: 1511.06434, 2015.

[37] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[38] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3, pp. 479–482, 2018.

[39] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of 2015 Military Communications and Information Systems Conference*, Canberra, Australia, pp. 1–6, 2015.

[40] I. Sharafaldin, A. H. Lashkari, S. Hakak, *et al.*, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proceedings of 2019 International Carnahan Conference on Security Technology*, Chennai, India, pp. 1–8, 2019.

[41] H. Z. Xu, Y. J. Wang, S. L. Jian, *et al.*, "Beyond outlier detection: Outlier interpretation by attention-guided triplet deviation network," in *Proceedings of the Web Conference 2021*, Ljubljana, Slovenia, pp. 1328–1339, 2021.

Zeyi LI was born in Soochow, China, in 1997. He received the B.S. degree in mathematics in 2019 and M.S. degree in computer science in 2022. He is currently pursuing the Ph.D. degree in cyberspace security at Nanjing University of Posts and Telecommunications, China. His research interests include network security, anomaly detection, and deep packet inspection. (Email: 2022040506@njupt.edu.cn)

Pan WANG received the B.S./M.S./Ph.D. degrees in electrical and computer engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001, 2004, and 2013, respectively. From 2017 to 2018, he has been a Visiting Scholar at University of Dayton (UD) in the Department of Electrical and Computer Engineering, OH, USA. He is currently a Full Professor at Nanjing University of Posts and Telecommunications. His research interests include cyber security and communication network security in B5G/6G/IIoT/smart grid/metaverse, ML/AI-enabled big data analytics, and applications.
(Email: wangpan@njupt.edu.cn)

Zixuan WANG was born in Nanjing, China, in 1994. He obtained the M.S. degree in logistics engineering at Nanjing University of Posts and Telecommunications in 2020. He is currently pursuing the Ph.D. degree at Nanjing University of Posts and Telecommunications. His research interests include encrypted traffic identification and data balancing.
(Email: 2020070135@njupt.edu.cn)