The Journal of Supercomputing (2020) 76:6690–6719
https://doi.org/10.1007/s11227-019-03092-1

Enhancing network intrusion detection classifiers using supervised adversarial training

Chuanlong Yin[1] · Yuefei Zhu[1] · Shengli Liu[1] · Jinlong Fei[1] · Hetong Zhang[1]

Published online: 11 December 2019

Abstract
The performance of classifiers has a direct impact on the effectiveness of intrusion detection system. Thus, most researchers aim to improve the detection performance of classifiers. However, classifiers can only get limited useful information from the limited number of labeled training samples, which usually affects the generalization of classifiers. In order to enhance the network intrusion detection classifiers, we resort to adversarial training, and a novel supervised learning framework using generative adversarial network for improving the performance of the classifier is proposed in this paper. The generative model in our framework is utilized to continuously generate other complementary labeled samples for adversarial training and assist the classifier for classification, while the classifier in our framework is used to identify different categories. Meanwhile, the loss function is deduced again, and several empirical training strategies are proposed to improve the stabilization of the supervised learning framework. Experimental results prove that the classifier via adversarial training improves the performance indicators of intrusion detection. The proposed framework provides a feasible method to enhance the performance and generalization of the classifier.

Keywords Intrusion detection · Generative adversarial networks · Artificial neural network · Adversarial training · Machine learning

✉ Chuanlong Yin
dragonyincl@163.com

Yuefei Zhu
zyf0136@sina.com

Shengli Liu
475737@qq.com

Jinlong Fei
feijinlong@126.com

Hetong Zhang
hetongzhang@live.cn

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China

1 Introduction

Intrusion detection, as a kind of multilevel and multilayer network protection measure, aims to detect various intrusion behaviors by collecting and analyzing all kinds of information on the network. In fact, intrusion detection is usually equivalent to a classification problem. It identifies whether network traffic behaviors are normal or any one of the other four attack types: Denial of Service (DoS), Probe, Root to Local (R2L) and User to Root (U2R) [1, 2]. Then, it sets the alarm and takes appropriate measures. Therefore, there is no doubt that constructing a suitable classifier and training it to improve its generalization is the key task of intrusion detection.

Several machine learning methods, including support vector machine (SVM) [3, 4], artificial neural network (ANN) [5, 6], K-nearest neighbors (KNN) [7, 8], random forest (RF) [9] and others [10, 11], have been implemented as classifiers to improve the performance of intrusion detection and have made good progress. However, previous works based on traditional machine learning methodologies, which belong to shallow learning algorithms, have a limited ability to represent complex functions for complex classification problems [12].

More recently, with the development of deep learning, more and more researchers have explored deep learning methods to enhance the performance of classifiers for intrusion detection, and have also achieved remarkable results. Compared with traditional machine learning algorithms, deep learning methods are adept in representing high-dimensional spatial features and can automatically learn the intrinsic features without feature engineering [13]. The experiments demonstrated deep networks significantly outperformed the shallow network in detection of attacks [12].

However, in practice, whether classifiers for intrusion detection are based on traditional machine learning or based on deep learning methods, the detection effectiveness is highly dependent on the number of samples for training. After all, in the supervised learning, classifiers can only get limited useful information from the limited number of labeled samples, which usually affects the generalization of classifiers. The best way to make a machine learning model generalize better is to train it on more data. Data augmentation allows more data to be generated from limited data, increasing the number and diversity of training samples. Training samples supplements can reduce the model's dependence on certain attributes, thereby improving the generalization ability of the model. Unfortunately, it is difficult to generate new fake data for a density estimation task unless we have already solved the density estimation problem [14]. In addition, it is expensive and time-consuming for labeling large training datasets. It is sometimes impossible due to emerging and fast evolving intrusion attacks, which makes those problems particularly severe. Generative adversarial networks (GANs) can learn the probability distribution of dataset, and try to generate new '*fake*' samples similar to data samples. Since GANs introduce interaction in the training stage (which is equivalent to adding a kind of '*fake*' labeled samples relative to original data samples), GANs can expand the labeled data and give more useful information on the basis of the training set. In fact, as a semi-supervised model, GANs enhance the effectiveness of image recognition [15], anomaly detection [16], imaging markers

[17], etc. Since GANs are suited to model the high-dimensional complex distributions of real-world data [18], it is reasonably straightforward to utilize them to offer more useful information to improve the generalization of classifiers. Several works have begun to explore the applications of GANs for anomaly detection. As far as we know, there are three types of application methods to apply GAN for intrusion detection.

With the purpose of generating adversarial attacks to evade the intrusion detection system, authors applied GAN to generate adversarial malicious examples to perform the black-box attacks. The experiments showed many intrusion detection systems were vulnerable to adversarial perturbations using GAN [19, 20]. A framework based on GAN generated DoS attack traffic similar to the normal traffic to evade network traffic classifiers [21]. Obviously, our research goal in this paper is to enhance the performance of the classifier for intrusion detection rather than to obtain the generated attacks to evade the intrusion detection system.

With the purpose of balancing previously unbalanced datasets, a framework based on GAN was proposed to generate data that captured the data distribution of selected attack types from the dataset. As a result, the framework was feasible for improving the performance of intrusion detection systems [22]. With the same purpose to address the challenges of both data scarcity and data imbalance, a framework was developed to incorporate deep adversarial learning with statistical learning. Experiments indicated that the proposed framework outperformed other models [23]. Indeed, the problem of unbalanced classification in the training set leads to a decline in the detection performance for intrusion detection, which results in a large bias of the classifier, and the prediction of the classifier tends to be the majority in the dataset. The above methods are essential to train the GAN model to learn the distribution of minority status, and oversampling samples to balance the training set. At last, they change the inter-class distribution in the training set. In this paper, we leverage GAN to generate new label data. It is equivalent to adding other new category of data to the training set rather than changing the number of other categories, and fundamentally different from those oversampling methods. It does not need to train each class separately, the training process is more simplified, and the training time and overhead are certainly decreased.

With the purpose of identifying anomaly attacks, in [16], authors developed GAN-based models for anomaly detection, and achieved good results on image and network intrusion datasets for binary classification. Compared with those methods, the application scenario in this paper is different. The above methods mainly use GANs for binary classification. The discriminant model of GAN itself is a binary classifier to judge the real or fake for input data, which is very similar to the anomaly detection. Therefore, GANs can easily be extended to the application scenario of binary classification. However, in the multiclass classification scenario, the multiclass classifier not only needs to judge the anomaly, but also needs to further judge the category of anomaly.

Inspired by the above reasoning, in this paper, we restrict our focus to complement the labeled samples via adversarial training, and augment the training set. Specifically, in the training phase, because the generative model G is continually generating '*fake*' samples to offer the classifier C with useful information (which enhances the

classification performance of the classifier C), the ID-GAN framework improves the generalization of the classifier.

As far as we know, we first propose the supervised learning framework based on GAN for intrusion detection (ID-GAN) under the multiclass classification task. Experimental results show that the ID-GAN framework improves the performance of classifiers by using complementary and helpful information by adversarial training. The framework can effectively enhance the generalization of the classifier and can improve the effectiveness of intrusion detection in a series of adversarial rounds, which achieves state-of-the-art results on the benchmark NSL-KDD dataset.

The main contributions in this paper are embodied in overcoming the following challenges.

1. The discriminative model D in the original GAN is a binary classifier, so it only can be used to judge whether the sample is from the real dataset or not. It does not have ability to further predict the classification of real samples. Hence, a novel supervised learning framework based on GAN for the multiclass classification is proposed in the paper.
2. Because the structure of the proposed framework is different from the original GAN, it is necessary to deduce the loss function again for the supervised learning according to the needs of the multiclass classification for intrusion detection. Therefore, we show the theoretical derivation of the loss function.
3. Unlike the original GAN and its variants, the purpose of the proposed framework is to train a multiclass model with enhanced performance rather than a generative model. Therefore, the training method of the framework is different from that of the original GAN and its variants, and how to train the proposed framework needs to be studied. Several empirical strategies are proposed to improve the stabilization of the framework.
4. Aiming at the problem of experimental verification, we compare the performance of the original classifier with that of the enhanced classifier via adversarial training on the benchmark NSL-KDD dataset for the multiclass classification, and show the graphical depiction in detail on how to enhance the classifier with the help of the proposed framework.

The remainder of the paper is organized as follows. Section 2 introduces the latest research progress in the field of intrusion detection, especially application research using deep learning in this area. We detail the original GANs and its variants in Sect. 3. Section 4 introduces our method and describes how to construct the supervised learning framework. The experimental configuration and evaluation criteria are shown in Sect. 5. Section 6 reveals how to train the proposed framework, and the experimental results and discussion are presented in Sect. 7. Finally, conclusion and future work are drawn.

2 Related work

Significant progress has been made in improving the performance of classifiers in the field of intrusion detection.

To improve the accuracy of decision tree (DT) and naïve Bayes (NB) classifiers for multiclass classification, two independent hybrid mining algorithms were presented [24]. Hybrid DT algorithm utilized a NB classifier to avoid overfitting, while Hybrid NB algorithm employed a DT classifier to select important features to improve efficiency. Kanakarajan, Muniasamy [25] introduced greedy randomized adaptive search procedure with annealed randomness—Forest (GAR-forest), to improve the performance of multiclass classification with feature selection. Experimental results showed that GAR-forest performed better for multiclass classification problem compared with random forest, C4.5, naive Bayes and multilayer perceptron.

Some researchers reduce the high dimensions and feature space by removing redundant or unimportant features to further improve the performance of the classifier. Kuang et al. [26] reduced the high-dimensional data using hybrid kernel principal component analysis (KPCA) and then utilized the SVM for intrusion detection. Ikram, Cherukuri [27] studied the integrating of principal component analysis (PCA) and SVM for abnormal recognition. The above research reduces the dimensions of the input feature space in the intrusion detection system, which effectively improves the overall performance of the classification. Nevertheless, PCA suffers from the fact that it is a linear combination of all the original variables. Thus, it often cannot obtain deterministic mappings from high-dimensional spaces to low-dimensional spaces [28]. Furthermore, its nonlinear extension KPCA suffers from two major disadvantages. First, the underlying manifold structure of data is not considered in process modeling. Second, the selection of the kernel function and kernel parameters is always problematic [29].

More recently, deep learning is one of the most effective machine learning techniques which is getting popular, and has gained a wide range of applications in the intrusion detection community. The researchers take advantage of generative models such as deep autoencoder (DAE), deep Boltzmann machine (DBMs) and deep belief networks (DBNs) in a pre-training stage (unsupervised learning) to improve the detection performance. During this process, each of the lower layers is separately trained from other layers, which allows other layers to be greedily trained layer by layer from the bottom up [12, 30]. Furthermore, the final prediction and classification are carried out by traditional machine learning algorithms such as SVM or SoftMax, which avoids manual intervention to select features and can effectively represent high-dimensional features.

Abolhasanzadeh [31] proposed an approach to detect attacks in big data using DAE based on dimensionality reduction and the neural network bottleneck feature extraction. The results in terms of accuracy rate outperformed PCA, factor analysis and KPCA.

Gao et al. [32] successfully exploited a classifier based on DBN for intrusion detection, and concluded that the classifier achieved a high accuracy when the greedy layer-by-layer learning algorithm was used for pre-train.

For optimizing the basic network structure of the DBN classification model in intrusion detection system, Wei et al. [33] designed an artificial fish swarm algorithm optimization particle swarm optimization joint genetic algorithm optimization particle swarm optimization algorithm (AFSA-GA-PSO). In order to optimize DBN model, the framework based on the above algorithm (AFSA-GA-PSO-DBN) was proposed and tested for multiclass classification. Compared with the machine learning model with superior performance such as SVM, random forest and naive Bayes, the framework improved the average classification accuracy.

Potluri et al. [34] applied the convolutional neural networks (CNNs) for intrusion detection with the purpose of identifying the multiple attack classes. Different performance metrics such as precision, recall and F-measure were calculated and compared with the existing deep learning approaches.

Javaid et al. [35] proposed a deep learning method based on self-taught learning (STL), and improved the performance of network intrusion detection for multiclass classification.

However, the DAE, DBNs and DBMs algorithms have the difficulties of an intractable partition function or an intractable posterior distribution. Therefore, they are typically only used for pre-training a classification network [36].

Generative adversarial networks are another type of deep generative model. Different from other generative models, GANs incorporate the adversarial idea and allow for interaction during training. It is a great potential model to be applied and popularized for many actual scenarios. In the past 2 years, there have been hundreds of GAN variants. In terms of GANs as a semi-supervised model [37, 38], the authors extended GAN to semi-supervised model in image classification to enhance the robustness of unsupervised learning models.

Several works have begun to explore the applications of GANs for the anomaly detection task. A semi-supervised model based on GAN, consisting of two generators, three discriminators and one classifier, was proposed for detection anomalies in communication packet streams [39]. The approach was effective for packet flow binary classification.

In [16], authors developed GAN-based models for anomaly detection, and achieved good results on image and network intrusion datasets for binary classification. However, the variant of GANs belongs to unsupervised learning, which is only leveraged for anomaly detection, and is incompetent for multiclass classification.

Different from the above variants, in this paper, we develop a novel framework based on GAN, propose the supervised learning approach for multiclass classification, and suggest several empirical techniques for the framework training.

3 Generative adversarial networks

Generative adversarial networks [40] belong to one of the deep learning frameworks first proposed by Ian J. Goodfellow in 2014. This idea is sought after by academics in various fields of study, and it shows broad application prospects in imaging, visual computing and other fields.

Generally, a standard framework of GANs consists of a generative model G and a discriminative model D. During the training of GANs, the samples (which are called as '*fake*' samples or generated samples) generated by the generative model G and the real data samples are mixed, and then randomly transmitted to the discriminative model D. The goal of the discriminative model D (which is equivalent to a binary classifier) is to identify the real data samples and the generated samples as accurately as possible. Meanwhile, the goal of the generative model G is the opposite of the discriminant model, which is to deceive the discriminative model D as much as possible and minimize the probability that the discriminative model D identifies the generated sample. Both sides are constantly optimizing themselves during training until they reach the equilibrium where neither side can improve and the generated sample is completely indistinguishable from the real data sample.

In summary, the original GAN contains a generative model G and a discriminative model D. G is used to capture the distribution of the dataset and generate similar samples, while D is a discriminator that determines whether the input is a data sample or a generated sample. The basic framework of the original GAN is shown in Fig. 1.

The generative model G takes a noise distribution $p(z)$ (usually a Gaussian distribution or uniform distribution) as an input and produces fake samples $G(z)$. Meanwhile, the discriminative model D identifies whether a sample comes from the data distribution $p(x)$ or the generated samples $G(z)$. The loss function of GANs can be defined as the following optimization problem [40]:

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}}\left[\log D(x)\right] + E_{z\sim P_z}(z)\left[\log\left(1 - D(G(z))\right)\right] \tag{1}$$

Equation (1) shows that in the training process of GANs, the discriminative model needs to be constantly revised to maximize the value of V, that is, to maximize $D(\boldsymbol{x})$ and minimize $D(G(z))$. Meanwhile, it is necessary to revise model G to minimize the value of V. In other words, by maximizing $D(G(z))$, the generative model tries to generate samples that are very similar to the data samples. Finally, both G and D reach the Nash equilibrium. The generative model G can estimate the probability distribution of the real data samples. Meanwhile, the detection accuracy

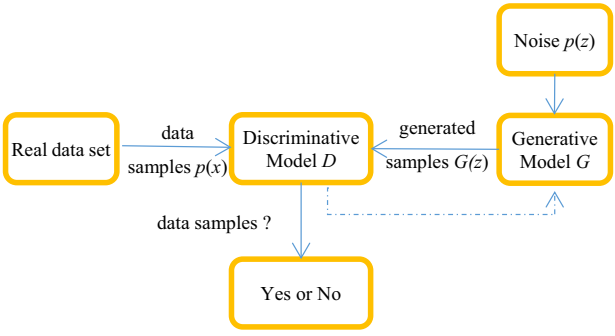

Fig. 1 The framework of original GAN

of the determinative model D is equal to 50%, which makes it difficult to identify whether the data sample is real or fake.

Additionally, GANs can be applied in semi-supervised learning. In an original GAN, the discriminator is a binary classifier that identifies the authenticity of samples. Considering a K-class task, the output of the generator can be classified as K+1, and the corresponding discriminator becomes a (K+1)-category classification problem [15]. The advantage of this kind of processing is that it can make full use of unlabeled data to learn the probability distribution of real data samples and thus aid the training process of supervised learning.

$$\begin{aligned} L &= -E_{x,y\sim P_{data}(x,y)}\left[\log P_{model}(y|x)\right] - E_{x\sim G}\left[\log P_{model}(y=K+1|x)\right] \\ &= L_{supervised} + L_{unsupervised}, \text{ where} \\ L_{superviesed} &= -E_{x,y\sim p_{data}}(x,y)\log P_{model}(y|x, \mathrm{y} < K+1), \text{ and} \\ L_{unsuperviesed} &= -\{E_{x\sim p_{data}}(x)\log\left[1 - P_{model}(y=K+1|x)\right] \\ &\quad + E_{x\sim G} log[P_{model}(y=K+1|x)]\} \end{aligned} \tag{2}$$

Let $P_{\text{model}}(y=K+1|\boldsymbol{x})$ denote the probability that $\boldsymbol{x}$ is a generated sample, which corresponds to $1-D(\boldsymbol{x})$ in the original GAN framework. Assuming that the dataset consists of real data and some generated samples, the loss function for training the classifier then becomes Eq. (2), which can be divided into two parts for different data sources. For labeled data samples, $L_{\text{supervised}}$ stands for the negative log probability of the labeled sample, given that the sample is from the real data. The goal is to expect the discriminative model to output the correct label on the real data distribution $P_{data}(x, y)$. For an unsupervised loss $L_{\text{unsupervised}}$, the loss function is defined by GANs. This is in fact the standard GAN game-value that becomes evident when we substitute $D(\boldsymbol{x})=1-P_{\text{model}}(y=K+1|\boldsymbol{x})$ in Eq. (3) [15].

$$L_{unsuperviesed} = -E_{x\sim P_{data}(x)}\log D(x) - E_{z\sim noise}\log(1 - D(G(Z))) \tag{3}$$

4 Proposed methodologies

4.1 The supervised learning framework using adversarial training for intrusion detection

A standard multiclass classifier for intrusion detection usually takes a sample $\boldsymbol{x}$ as input, and outputs a 5-dimensional vector (l_{normal}, l_{probe}, l_{dos}, l_{r2l}, l_{u2r}) that can be turned into one of the *five* possible class probabilities by applying the *softmax* function. In the supervised learning, such a model is then trained by minimizing the cross-entropy between the real labels and the predictive distribution $P_{\text{model}}(y|\boldsymbol{x})$ to obtain the optimal parameters.

As stated before, the two core models in a GAN are the generative model G and the discriminative model D. D as a binary classifier only has ability to judge whether the sample is from the real dataset or not. It does not have ability to further predict

the classification of real data samples. Additionally, most of classifiers for intrusion detection usually belong to the supervised learning, so we need to reconstruct a supervised learning framework based on GAN.

In order to supply more information for the multiclass classifier C, we take the output of the generative model C as the input of the classifier C together with the original training set.

To improve the efficiency of the framework and further simplify the framework, we replace the discriminative model D with a multiclass classifier C. In this way, the classifier C not only undertakes the task of classification for the training set, but also serves as the role of the discriminative model D to determine whether the sample is from the generative model G or the real dataset. We regard the output of the generative model G as the '*fake*' category, and the corresponding multiclass classifier becomes a 6-category classifier. Therefore, the original GAN is transformed into a supervised learning framework for intrusion detection, as shown in Fig. 2. First, the framework needs to train the classifier through adversarial training, as indicated by the green arrow. Second, the blue arrow represents the framework inputs the test samples into the trained classifier for multiclass classification.

After the introduction of adversarial training for intrusion detection, the generative model can continually generate '*fake*' samples from a random distribution $p(z)$. In the adversarial training, the multiclass classifier identifies whether the sample is *Normal*, or *fake*, or any one of the other four attack types: *DoS*, *Probe*, *R2L* and *U2R*, while the generative model dynamically adjusts the strategy for generating more similar fake samples according to the feedback (*fake* or *real*) from the multiclass classifier. Thus, the framework can train the classifier together with new augmented training set, which

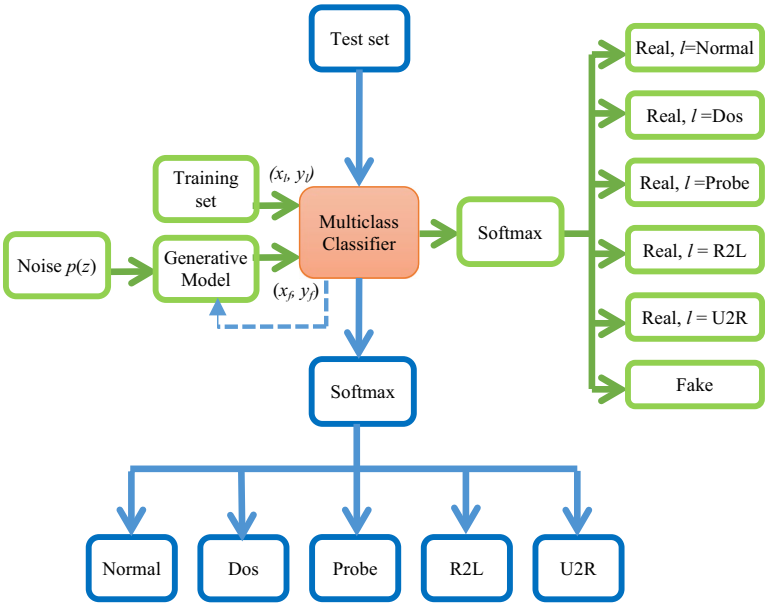

Fig. 2 The framework of ID-GAN

includes original five-category labeled samples and constantly generated new '*fake*' samples.

For example, originally, only a professor (similar to the classifier) trained students to recognize and classify five languages (*Russian*, *English*, *Arabic*, *French* and *German*). An assistant professor (similar to the generative model) is added to train the students to recognize whether it belongs to five languages or not. Although the supplementary assistant professor does not directly teach students how to identify and classify the five types of languages, the practice of distinguishing 'whether it belongs to five languages' is also helpful for classification and recognition of languages. There are rough feedbacks, which are better than no feedback.

In summary, the main idea of the ID-GAN framework is to train a multiclass classifier that plays both the roles of a classifier performing the classification task and a classifier to distinguish generated samples from the real data samples. To be more specific, the classifier takes a sample as the input and classifies it into six classes. Real data samples are classified into the first five classes, and generated samples are classified into '*fake*' class, as shown in Fig. 2.

4.2 The derivation of the loss function

It is assumed that (x_l, y_l) is a sample from the training set that contains a 5-category classification label, where $y_l \in \{normal, dos, probe, r2l, u2r\}$. The generative model generates '*fake*' samples (x_f, y_f) from the random noise distribution, where $y_f =$ '*fake*.' The samples (x, y) are synthetic data samples and generated samples, where the label y contains six classes ($y \in \{normal, dos, probe, r2l, u2r, fake\}$). For the multiclass classification problem, the classifier inputs a sample x and outputs the classification probabilities for the six classes p_i ($i = 1, 2, 3, 4, 5, 6$). The first five categories correspond to the original classification, and the last classification corresponds to '*fake*' category by applying the *softmax* function.

Assuming that p is the real probability distribution of the sample and q is the predicted probability distribution of the classifier, the *cross-entropy* for a given dataset X is defined as:

$$\mathrm{CE}(p, q) = -\sum_{x \in X} p(x) \log q(x) \tag{4}$$

The value of Eq. (4) indicates the error between the real classification and the predicted classification. The smaller the value is, the closer the predicted probability distribution is to the real probability distribution, and the more accurate the predicted result will be.

Under the multiclass classification task, the loss function is usually defined as *cross-entropy* loss. Let $y^j_{x_i}$ represent the real probability distribution of the sample x_i, and let $P_{model}\left(y = j|x_i\right)$ represent the predicted probability distribution of the sample x_i, then the corresponding loss function can be defined as:

$$L_x = -\sum_j y^j_{x_i} \log P_{model}\left(y = j|x_i\right)$$

$$\forall j \in \{normal, dos, probe, r2l, u2r, fake\} \tag{5}$$

For dataset $\boldsymbol{X}$, which is synthetic data samples and generated samples, the corresponding loss function is defined as:

$$Lc = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j} y_{x_i}^{j} \log P_{model}\left(y = j|x_i\right)$$

$$\forall j \in \{normal, dos, probe, r2l, u2r, fake\} \tag{6}$$

After one-hot coding, the real category of the sample $y_{x_i}^{j}$ is mapped into a K-dimension vector. For example, [1, 0, 0, 0, 0, 0] indicates that the sample belongs to the '*normal*' category, [0, 1, 0, 0, 0, 0] indicates that the sample belongs to the '*dos*' category, and [0, 0, 1, 0, 0, 0] indicates that the sample belongs to the '*probe*' category. [0, 0, 0, 1, 0, 0] means that the sample belongs to '*r2l*' category, [0, 0, 0, 0, 1, 0] indicates that the sample belongs to '*u2r*' category, and [0, 0, 0, 0, 0, 1] indicates that the sample belongs to the '*fake*' category. Similarly, if the sample x_i belongs to category c, then $y_{x_i}^{c} = 1$. Besides, all the values of the remaining columns are 0, that is, $y_{x_i}^{j \neq c} = 0$.

Therefore, the loss function of the multiclass classifier in the proposed framework can be further expressed as follows.

$$\begin{aligned}
Lc &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{j} y_{x_i}^{j} \log P_{model}\left(y = j|x_i\right) \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[y_{x_i}^{j=c} \log P_{model}\left(y = c|x_i\right) + \sum_{j \neq c} y_{x_i}^{j} \log P_{model}\left(y = j|x_i\right)\right] \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[y_{x_i}^{j=c} \log P_{model}\left(y = c|x_i\right)\right] \\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[\log P_{model}\left(y = c|x_i\right)\right]
\end{aligned}$$

5 Experiments

5.1 Dataset

The NSL-KDD (Knowledge Discovery and Data Mining) [41, 42] is a benchmark dataset for network intrusion detection. It removes a large amount of redundant data in the original dataset and adjusts the normal and abnormal data in proper proportions to make the testing and training set sizes more reasonable. This is still ideal

Table 1 Different classifications in the NSL-KDD dataset

	Total	Normal	Dos	Probe	R2L	U2R
KDDTrain$^+$	125,973	67,343	45,927	11,656	995	52
KDDTest$^+$	22,544	9711	7458	2421	2754	200

Table 2 Features of NSL-KDD dataset

No.	Features	Types	No.	Features	Types
1	duration	Continuous	22	is_guest_login	Symbolic
2	protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	srv_serror_rate	Continuous
6	dst_bytes	Continuous	27	rerror_rate	Continuous
7	land	Symbolic	28	srv_rerror_rate	Continuous
8	wrong_fragment	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	srv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_ra	Continuous
16	num_root	Continuous	37	dst_host_srv_diff_host_rat	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

and the most trustful public benchmark dataset [43–45] for an effective and accurate assessment of different machine learning algorithms for intrusion detection.

The NSL-KDD dataset consists of a training set and a test set. The training set KDDTrain$^+$ contains 125,973 instances, and the test set KDDTest$^+$ contains 22,544 instances, as shown in Table 1.

There are 41 features and 1 class label for every traffic record, and the features include basic features (No. 1–No. 10), content features (No. 11–No. 22) and traffic features (No. 23–No. 41), as shown in Table 2. According to their characteristics, attacks in the dataset are categorized into four attack types: DoS, Probe, R2L and U2R. The testing set has some specific attack types that disappear in the training set, which allows it to provide a more realistic theoretical basis for intrusion detection.

Each instance in the dataset is described by 41 features and 1 label, which is normal or one of the attack types (*DoS*, *Probe*, *R2L* and *U2R*). Because the input value

should be a numeric matrix, we must convert some nonnumeric features, such as 'protocol_type,' 'service' and 'flag' features, into numeric form. For example, the feature 'protocol_type' has three types of attributes, 'tcp,' 'udp' and 'icmp,' and its numeric values are encoded as binary vectors (1,0,0), (0,1,0) and (0,0,1). Similarly, the feature 'service' has 70 types of attributes, and the feature 'flag' has 11 types of attributes. Continuing in this way, therefore, after one-hot coding and normalization of the features, the 41-dimensional feature is transformed into a 122-dimensional feature.

5.2 Selection of generative model and classifier for ID-GAN

In theory, we can choose any generative model and classifier as the generative model G and multiclass classifier C for the ID-GAN framework, respectively. However, in practical applications, the generative model and classifier are generally nonlinear mapping functions, such as the multilayer perception machine, long short-term memory (LSTM) and others.

We selected the LSTM network as the generative model, which is a special type of recurrent neural networks (RNNs) with improvement and promotion. LSTM has the ability to remember the long-term information and overcome the vanishing gradient problem. In this paper, a 3-layer LSTM network was adopted as a generative model in the ID-GAN framework, which included an input layer, a hidden layer and an output layer. The number of neurons in the input layer was set to 120, and the number of hidden layer nodes was set to 80. The number of output nodes was 122, the same as the number of processed features mentioned in Sect. 4.1. Besides, the time step was set to 10.

Artificial neural network is one of the common machine learning methods for many complex applications, such as pattern recognition, automatic control and deep learning. It is a nonlinear structure, and its main purpose is to classify spam identification, disease judgment, cats and dogs classification and so on. In [5], authors proposed a model based on ANNs for binary and multiclass classification in the realm of intrusion detection and achieved good results. In this paper, we used a 4-layer neural network structure as multiclass classifier C, including an input layer, two hidden layers and an output layer. The number of neurons in the input layer was 122, which was the same as the number of features. The numbers of hidden layer nodes in the first and second layer were 80 and 20, respectively. For adversarial training, the number of output layers was 6, which was the same as the number of classes.

5.3 Classification metrics

In order to comprehensively and objectively evaluate the performance of the classifier in detecting intrusion behaviors, five indicators, accuracy, precision, recall, f1-score and confusion matrix, are in use.

True Positive (TP) indicates the intrusion traffic that is correctly detected, True Negative (TN) indicates the legitimate traffic that is correctly detected, False Negative (FN) indicates the intrusion traffic that is incorrectly detected as legal traffic,

and False Positive (FP) indicates the legal traffic that is falsely detected as illegal traffic.

As shown in Table 3, each column of the confusion matrix represents a prediction category, and the total number in each column represents the number of data predicted as the category. Each row represents the true category of the data, and the total number of data in each row represents the number of data in the category.

The accuracy, precision, recall and f1-score are, respectively, defined in Eqs. (8) to (11).

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$\text{F1 score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{11}$$

To evaluate the performance of the multiclass classifier, due to label imbalance, we utilize weighted average precision, weighted average recall and weighted average f1-score. We only need to calculate metrics for each label, and find their average weighted by support (the proportion of each class). In short, the weight is the occurrence ratio that the number of each label accounts for the total number. Obviously, the overall accuracy is equal to weighted average recall.

Hence, a good classifier for intrusion detection should have a higher accuracy, precision, recall and f1-score.

5.4 Controlled experiments

In this paper, controlled experiments were adopted to more accurately and objectively evaluated the effect of the multiclass classifier via adversarial training using our framework. As stated before, since ANN was selected as the multiclass classifier in the ID-GAN framework, we should also choose ANN without adversarial training as the controlled group. Apparently, the parameters of the multiclass classifier without adversarial training (*denoted as* $C_{original}$) were the same as those of the

Table 3 Confusion matrix

Actual class	Predicted class	
	Anomaly	Normal
Anomaly	TP	FN
Normal	FP	TN

classifier via adversarial training (*denoted as* $C_{enhanced}$) in the ID-GAN framework. The number of input nodes was also 122. The number of the first and second hidden nodes was 80 and 20, respectively. However, the output of $C_{original}$ has five nodes (five categories), which was the only difference compared with $C_{enhanced}$ in ID-GAN framework.

5.5 Experimental configuration

In this paper, we used one of the broadest deep learning frameworks—*Keras*, the architecture of which is flexible and supports various models and new tricks such as Batch normalization. The experiment was performed on a personal notebook ThinkPad E450, which has a configuration of an Intel Core i5-5200U CPU @ 2.20 GHz with 8 GB memory and did not use GPU acceleration.

6 Training the framework

To ensure the objectivity and impartiality of experiments and to accurately and objectively evaluate the performance of the ID-GAN framework for enhancing the multiclass classifier, we had selected 11 observation points at 100, 200, 500, 1000, 2000, 5000, 8000, 10,000, 20,000, 50,000 and 125,973 different numbers of '*fake*' samples that were mixed in the adversarial training at each epoch. Their respective ratios to the total samples are 0.08%, 0.16%, 0.40%, 0.79%, 1.56%, 3.82%, 5.97%, 7.35%, 13.70%, 28.41%, and 50.00%. Then, we observed the average accuracy of the enhanced classifier via adversarial training on the test set KDDTest^+ over epochs.

Although GANs have achieved great success in image generation, training a stable GAN is still difficult in practice. As mentioned before, we chose the generative model and multiclass classifier for the ID-GAN framework. This section focuses on addressing the issue of training difficulty because of the excessively free and uncontrollable frameworks of GANs. We chose two training parameters that most affect the performance of the multiclass classifier, including the existence of prior training and the noise distribution. This allows us to test how the above parameters affect the performance of the classifier and how to get a more stable, robust and generalized trained classifier.

1. Prior training

The prior training mentioned in this paper refers to training the classifier *n_times* times in advance by using the original five-category dataset before adversarial training. This allows the classifier to be properly guided to prevent itself from being too free to control. Under the same experimental conditions, the detection performance of the multiclass classifier was tested on the test set KDDTest^+ with and without prior training, respectively.

As shown in Algorithm 1, first, we trained the multiclass classifier *C* for *n_times*. Then, we drew *m* generated '*fake*' samples and *n* real data samples, respectively, to

train the classifier. Moreover, since the classifier *C* can better guide adjustments of generative model *G*, generally let the classifier *C* loop more times during the training [40]. Thirdly, we froze the classifier *C* and draw *m* generative samples again to train the generative model *G*.

Algorithm 1 Training Algorithm of the ID-GAN Framework

Input: *n_times*: times of the prior train for the classifier
Epoch: number of total iterations
Kstep: number of steps to apply to the classifier
G: Generative Model
C: Original Multiclass Classifier
Output: Enhanced Multiclass Classifier C

Prior_Train(C, *n_times*)
for *i* = 1 **to** *Epoch* **do**
  for *k*=1 **to** *Kstep* **do**
    Set_trainability(Classifier C, True)
    Draw *m* generated samples $\{(x_{fake}^{(1)}, y_{fake}^{(1)}), \ldots, (x_{fake}^{(m)}, y_{fake}^{(m)})\}$ from Generative Model G.
    Draw *n* data samples $\{(x_{label}^{(1)}, y_{label}^{(1)}), \ldots, (x_{label}^{(n)}, y_{label}^{(n)})\}$ from training set.
    Perform gradient descent on the parameters of Classifier C on the combined mini-batch of size $m+n$.
  end for
  Set_trainability(Classifier C, False)
  Draw *m* generative samples $\{(x_{fake}^{(1)}, y_{fake}^{(1)}), \ldots, (x_{fake}^{(m)}, y_{fake}^{(m)})\}$ from Generator G.
  Perform gradient descent on the parameters v of Generative Model G on the minibatch of size *m*.
end for

Without loss of generality, the times *n_times* of the prior training were set to 10, the epoch *k* of ID-GAN was set to 90. Thus, total training epoch was 100.

Figure 3 shows the average accuracy of the classifier on the test set at the 11 mixed observation points with and without prior training, respectively. The results reveal that the average detection performance of the classifier with prior training on the test set at 11 different mixed '*fake*' samples is higher than that without prior training. As a result of the experiments, we concluded that prior training of the multiclass classifier is conducive to training the ID-GAN model toward a more 'correct' direction.

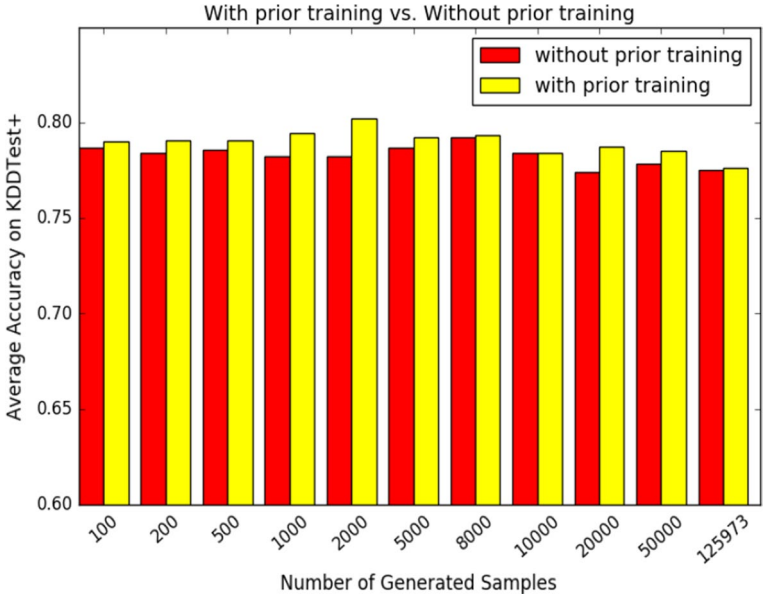

Fig. 3 The average accuracy of the classifier using the ID-GAN at different observation points with and without prior training, respectively

2. Uniform distribution versus Gaussian distribution

The uniform distribution or Gaussian distribution can usually be used as the input of the generative model G. To further train a more stable and efficient multiclass classifier, we mainly studied the performance of the multiclass classifier via adversarial training using the uniform distribution or Gaussian distribution as the input of the generative model, respectively.

The experimental results shown in Fig. 4 indicate that the average accuracy of the classifier via the ID-GAN framework on the test set does not differ among the five observation points of 1000, 5000, 8000, 20,000 and 125,973 using the Gaussian distribution or the uniform distribution as the input of the generative model, and the maximum difference of the average accuracy is 0.0036. However, at the observation points of 500, 2000, 10,000 and 50,000, the average accuracy on the test set with the uniform distribution as the input of G is higher than the Gaussian distribution under the same situation, and the maximum difference is 0.0131.

Through the study of training the ID-GAN framework, a priori training for classifiers is used in advance, and the uniform distribution is helpful to enhance the performance of classifiers. In this way, we can train a relative local optimal classifier via the ID-GAN framework of which the detection performance is high and stable. Of course, the training model we get is not the best, but we provide the notion of training and prove the impacts of prior training and the choice of the noise distribution on the performance.

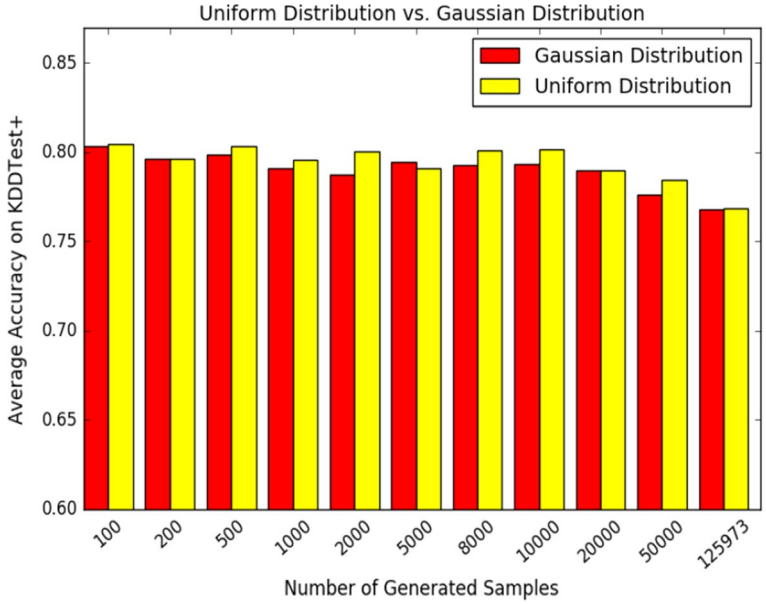

Fig. 4 The average accuracy of the classifier at different observation points with the Gaussian distribution and uniform distribution, respectively

7 Results and discussion

7.1 Controlled group

As stated before, controlled experiments were adopted and ANN was selected as the multiclass classifier in controlled group. Under the conditions of 100 training epochs, the accuracy of the original classifier $C_{original}$ on the dataset over epochs is shown in Fig. 5. The average accuracies of the original classifier on the training set and the test set are 99.67% and 78.81%, respectively, and the original classifier has the highest accuracy of 79.85% on the test set at the 95th epoch.

7.2 Overall comparison

In Sect. 5, we trained a local optimal classifier using the ID-GAN framework on the test set at 11 different observation points, as shown in Fig. 6. The vertical axis represents the average accuracy on the test set, while the horizontal axis represents the number of '*fake*' samples mixed with training samples at each adversarial training epoch. Moreover, the blue dot indicates the average accuracy of the classifier enhanced $C_{enhanced}$ by the ID-GAN framework on the test set, and the red-dotted line indicates the baseline of the average accuracy of the original classifier $C_{original}$ on the same test set.

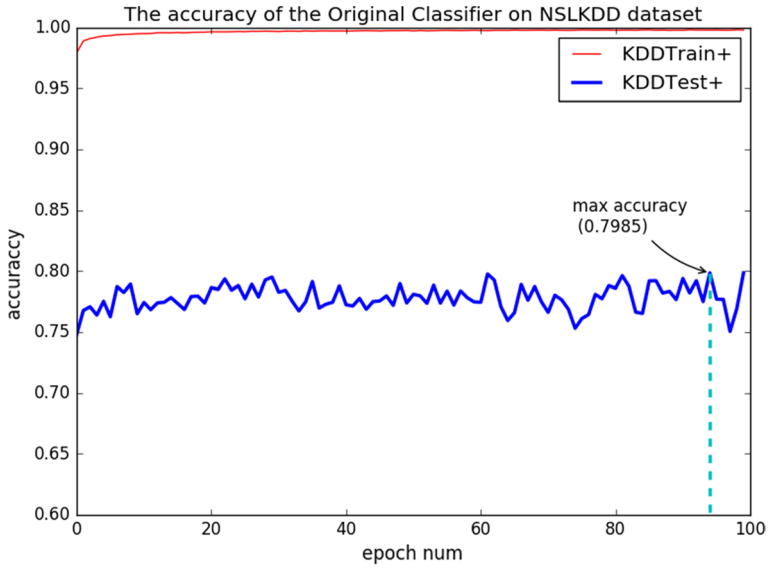

Fig. 5 The accuracy of the original classifier over epochs

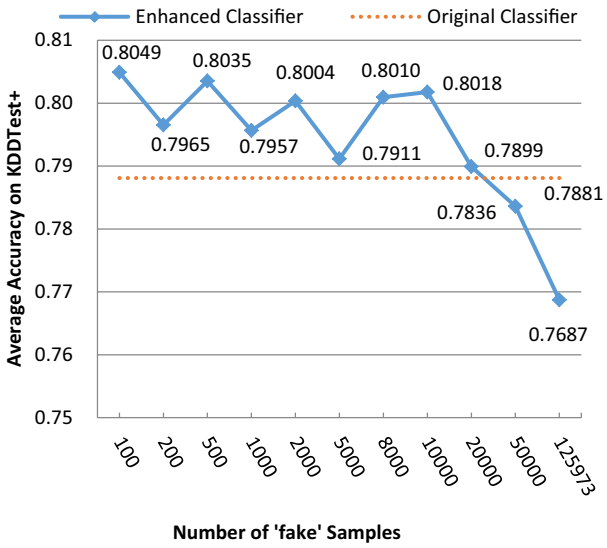

Fig. 6 The average accuracy of the enhanced classifier on the test set at 11 observation points

Figure 6 indicates that the average accuracy of the enhanced classifier $C_{enhanced}$ at 9 points has exceeded that of the original classifier $C_{original}$. The classifier via adversarial training using the ID-GAN framework obviously improves its detection performance and generalization. As the number of generated samples increases (e.g., 50,000 to 125,973), the average detection performance of the enhanced classifier

$C_{enhanced}$ begins to decline, and the ID-GAN framework no longer enhances the classifier $C_{original}$. The fact that generated samples account for a large proportion in the synthesized data at each adversarial training epoch can cause the classifier to focus too much on the sample to be '*fake*,' decreasing the classification performance of the original five types.

Figure 7 details the accuracy of the original classifier $C_{original}$ and the enhanced classifier $C_{enhanced}$ on the test set over epochs at the 11 observation points, and show how to improve the performance of the classifier $C_{original}$ using ID-GAN framework. The blue line indicates the accuracy of the original classifier $C_{original}$ on the test set, while the red line indicates the accuracy of the enhanced classifier $C_{enhanced}$ via adversarial training on the same test set over 100 epochs. In addition, the black arrow represents the optimal enhanced classifier on the test set.

Based upon the results of controlled experiments, the accuracy of the enhanced classifier $C_{enhanced}$ is significantly higher than that of the original classifier $C_{original}$ after approximately 20 epochs and up to 80 (section [20, 80]) at the 100, 500, 2000, 8000 and 10,000 observation points.

Correspondingly, the maximum accuracies of optimal enhanced classifiers $C_{enhanced}$ via ID-GAN framework at the 11 observation points are also significantly higher than those of the original classifier $C_{original}$, no matter how many '*fake*' samples are mixed.

Figure 7 shows the graphical depiction in detail on how to enhance the classifier with the help of the proposed framework.

7.3 Individual comparison

The accuracy and weighted average of the precision, recall and f1-score of optimal classifiers (the black arrows) can be gained using the *classification_report* function from *scikit-learn* library.

The performance measures of different multiclass classifiers on the same test set are shown in Table 4. The row with the orange background color indicates the performance of the original classifier $C_{original}$ (ANN), the rows with the light-gray background color indicate the performance of multiclass classifiers on the same test set in other studies [5, 25, 35], and the rows without background color indicate the performance of enhanced classifiers $C_{enhanced}$ via adversarial training at different observation points.

As shown in Table 4, the accuracy of our original classifier $C_{original}$ based on ANN (79.85%) is almost the same as the result (79.90%) gained in [5], which also proves the validity and objectivity of our experimental results. However, classifiers enhanced via adversarial training outperform the original classifier $C_{original}$ without ID-GAN framework and those in other studies in terms of the weighted average of accuracy, precision and other performance indicators.

The experimental results prove that the ID-GAN framework can enhance the original classifier's classification performance and generalization ability via adversarial training.

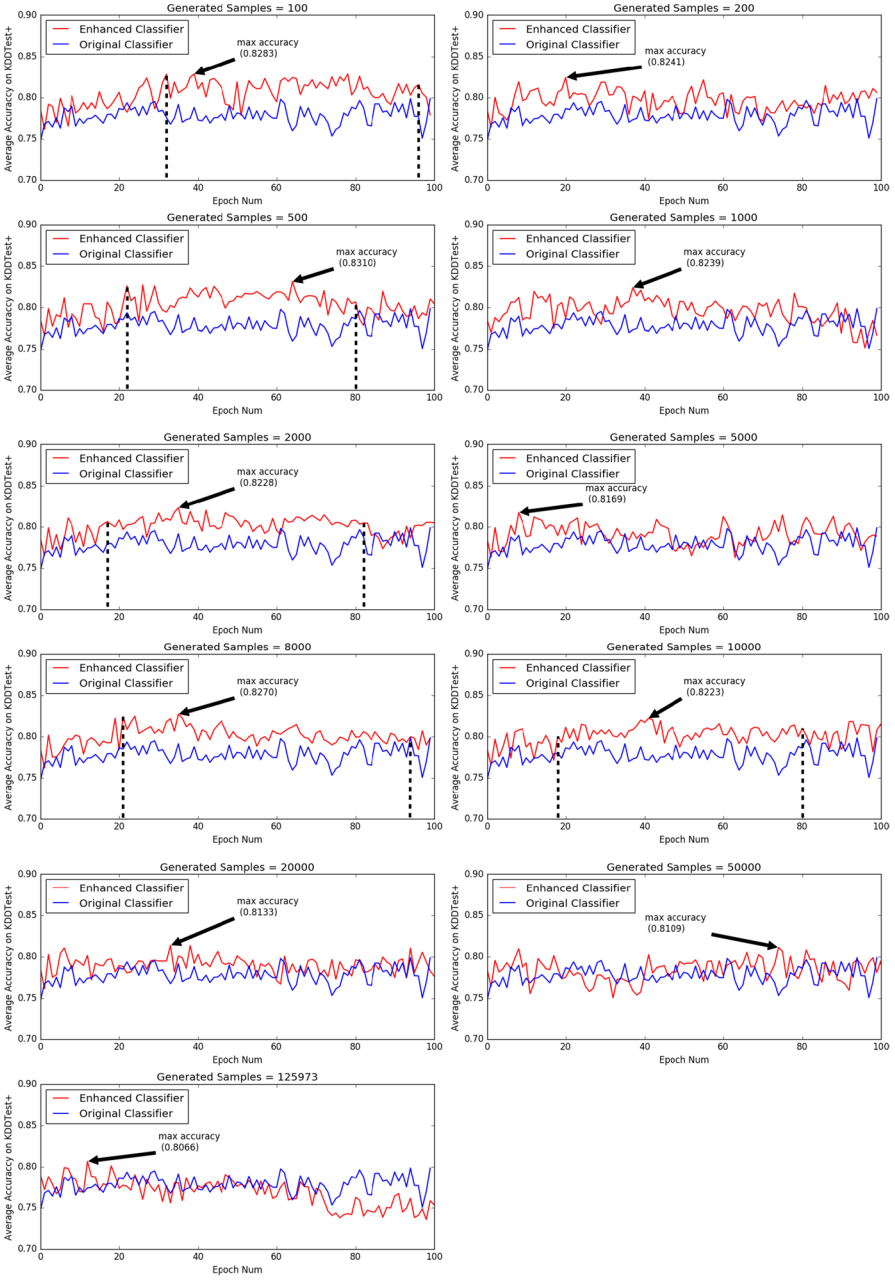

Fig. 7 The accuracy of multiclass classifiers on the test set over epochs at the 11 observation points

In order to describe the improving detection performance of the classifier via adversarial in detail, we take the observation point 8000 as an example. The confusion matrices of the enhanced classifier and original classifier are shown in

Table 4 Performance measures of different multiclass classifiers on the same test set KDDTest$^+$

Classifier	Accuracy (%)	Weighted average precision	Weighted average recall	Weighted average F1-score	Training time (s)
Decision Tree (C4.5) [25]	74.93	0.8020	0.7490	0.6970	–
Naïve Bayes [25]	73.76	0.8010	0.7380	0.7140	–
Random forest [25]	75.69	0.8000	0.7570	0.7230	–
GAR–forest [25]	78.90	0.8240	0.7980	0.7990	–
ANN [5]	79.90	–	–	–	–
AFSA-GA-PSO-DBN [33]	82.36	–	–	–	–
CNN [34]	78.42	–	–	–	–
Self-taught learning [35]	79.10	–	–	–	–
ANN (original classifier)	79.85	0.8242	0.7985	0.7809	2565
Enhanced classifier (Num. = 100)	82.83	0.8353	0.8283	0.8175	2597
Enhanced classifier (Num. = 200)	82.41	0.8413	0.8241	0.8172	2645
Enhanced classifier (Num. = 500)	83.10	0.8520	0.8310	0.8196	2691
Enhanced classifier (Num. = 1000)	82.39	0.8441	0.8239	0.8122	2765
Enhanced classifier (Num. = 2000)	82.28	0.8451	0.8228	0.8102	2848
Enhanced classifier (Num. = 5000)	81.69	0.8412	0.8169	0.8034	2980
Enhanced classifier (Num. = 8000)	82.70	0.8470	0.8270	0.8131	3080
Enhanced classifier (Num. = 10,000)	82.23	0.8376	0.8223	0.8072	3220
Enhanced classifier (Num. = 20,000)	81.33	0.8304	0.8133	0.8048	3347
Enhanced classifier (Num. = 50,000)	81.09	0.8420	0.8109	0.7884	4283
Enhanced classifier (Num. = 125,973)	80.67	0.8258	0.8067	0.7932	6268

Table 5 Confusion matrix of the original classifier on KDDTest$^+$

Actual class	Predicted class				
	Normal	DoS	Probe	R2L	U2R
Normal	9259	76	366	9	1
DoS	1111	6198	125	24	0
Probe	397	234	1748	42	0
R2L	1962	0	12	775	5
U2R	163	0	0	15	22

Table 6 Confusion matrix of the enhanced classifier on KDDTest$^+$

Actual class	Predicted class				
	Normal	DoS	Probe	R2L	U2R
Normal	9385(↑)	79	236	5	6
DoS	1035	6242(↑)	158	23	0
Probe	161	168	2034(↑)	57	1
R2L	1775	0	14	953(↑)	12
U2R	141	1	16	11	31(↑)

Table 7 Classification metrics of the enhanced classifier compared with the original classifier on KDDTest$^+$

Intrusion type	Original classifier			Enhanced classifier		
	Recall	Precision	F1-score	Recall	Precision	F1-score
Normal	0.9535	0.7182	0.8193	0.9664	0.7510	0.8452
DoS	0.8311	0.9524	0.8876	0.8370	0.9618	0.8950
Probe	0.7220	0.7765	0.7483	0.8401	0.8275	0.8338
R2L	0.2814	0.8960	0.4283	0.3460	0.9085	0.5012
U2R	0.1100	0.7857	0.1930	0.1550	0.6200	0.2480
Weighted average	0.7985	0.8242	0.7809	0.8270	0.8470	0.8131

Tables 5 and 6, respectively. The confusion matrix indicates that the enhanced classifier has an increase in the correct classification of all five types, and it especially improves the detection performance of Probe and R2L attacks.

Meanwhile, Table 7 shows that the recall, precision and other performance indicators of the enhanced classifier on each class, and also the weight average of those indicators are improved compared with the original classifier.

7.4 Training time

In terms of training time, data augmentation means that the number of the training sample is increased, the classifier takes more training time and the cost of training time is certainly increased. From Table 4, if the ID-GAN framework generates a small number of training samples (less than 8000), the model training time will take no more than 500 s. As the training samples increase, for example, mixing generated samples more than 10,000, the training time will cost more than 700 s. Of course, this is just an experimental result on a personal computer, and we believe that the training time will be greatly reduced on GPU acceleration or on a high-performance server.

7.5 Discussion

Compared with the binary classification for intrusion detection, the multiclass classification needs to further judge the intrusion type. Hence, the multiclass classification problem is more complicated and has greater detection difficulty. As shown in Table 4, in terms of the most important performance indicators of multiclass classification, the detection accuracy of decision tree (C4.5), naïve Bayes, random forest, GAR-forest, ANN, and self-taught learning is less than 80%, and the ANN model obtains the highest detection accuracy (79.9%). In the experiment, we further enhanced the multiclass classification effect of the ANN model through the ID-GAN framework, which obviously improved the multiclass classification detection performance of the ANN model. The detection accuracy, precision and f1-score are improved by about 3.25%, 2.87% and 2.78%, respectively. In [33], the DBN based on an optimization method for multiclass classification archived the highest detection accuracy (82.36%), and the classification accuracy of the CNN algorithm was 78.42% [34]. Even so, our proposed approach is also superior to those methods.

From the above experimental results, we believe that the samples continuously generated by the generative model (as the sixth labeled samples) using adversarial training can provide useful and complementary information for the classifier, and are helpful to improve the accuracy, precision, f1-score and other performance indicators. It is verified accordingly that the ID-GAN framework can generate more useful training samples from limited data, and increase the number and diversity of training samples. Consequently, it improves the generalization of the classifier.

To give a popular example: Even if no one is taught to recognize words, more exercise to distinguish 'that is not the word' is also beneficial to recognize words. Rough feedback is better than no feedback.

In terms of overall performance comparison, the average detection rate of the enhanced model on the test set during 100 iterative training outperforms that of the original classifier, especially approximately during the training interval [20, 80] at the 100, 500, 2000, 8000 and 10,000 observation points. The ID-GAN framework using complementary and helpful information from the generative model significantly improves the detection performance of the original classifier. In comparison with the optimal model, the optimal classifier obtained using adversarial training is superior to the original classifier in terms of accuracy, precision, f1-score and other performance indicators, especially improving the detection performance of Probe and R2L attacks.

Hence, the ID-GAN framework via supervised adversarial training can enhance generalization of the original classifier in the performance of detecting attacks, and can be used as a framework to improve the performance of the intrusion detection classifier.

8 Conclusion and future work

In this paper, we first further transform GAN into a supervised learning approach, and propose an intrusion detection framework based on GAN using adversarial training to enhance the classifier. The ID-GAN framework continually generates the '*fake*' labeled

samples by using the generative model G, assists the classifier to improve the detection performance, and enhances the generalization of the original classifier. The approach proposed in this paper provides a feasible method to enhance the classifier. It also provides a new technology and area of thought for the research and practice in other related fields, such as botnet detection.

Similarly, the proposed approach may be under threats of training difficulties and lack of diversity in generated samples. As with other deep learning applications, there are still interpretation and self-adaption scientific problems. In future research, we will study the effect of the ID-GAN framework for other classifiers and further study the optimization training method. Furthermore, we will focus on the hyper parameters of ID-GAN framework, such as the value of *n_times* in prior training and the number of steps to apply to the classifier.

Acknowledgements This work was supported in part by the National Key Research and Development Program of China (NO. 2016YFB0801601 & NO. 2016YFB0801505).

Appendix

See Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 and 19.

Table 8 The performance indicators of the original classifier

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7182	0.9535	0.8193	9711
DoS	0.9524	0.8311	0.8876	7458
Probe	0.7765	0.7220	0.7483	2421
R2L	0.8960	0.2814	0.4283	2754
U2R	0.7857	0.1100	0.1930	200
Weighted average	0.8242	0.7985	0.7809	22,544

Table 9 The performance indicators of the enhanced classifier (Num. = 100)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7648	0.9652	0.8534	9711
DoS	0.9544	0.8390	0.8930	7458
Probe	0.8629	0.7980	0.8292	2421
R2L	0.7895	0.3991	0.5301	2754
U2R	0.1188	0.0600	0.0797	200
Weighted average	0.8353	0.8283	0.8175	22,544

Table 10 The performance indicators of the enhanced classifier (Num. = 200)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7656	0.9288	0.8394	9711
DoS	0.9588	0.8277	0.8885	7458
Probe	0.7084	0.8430	0.7699	2421
R2L	0.9474	0.4844	0.6410	2754
U2R	0.2778	0.0500	0.0847	200
Weighted average	0.8413	0.8241	0.8172	22,544

Table 11 The performance indicators of the enhanced classifier (Num. = 500)

Intrusion type	Precision	Recall	F1-score	support
Normal	0.7532	0.9665	0.8466	9711
DoS	0.9575	0.8333	0.8911	7458
Probe	0.8449	0.8166	0.8305	2421
R2L	0.9267	0.4179	0.5761	2754
U2R	0.7778	0.0350	0.0670	200
Weighted average	0.8520	0.8311	0.8196	22,544

Table 12 The performance indicators of the enhanced classifier (Num. = 1000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7453	0.9584	0.8385	9711
DoS	0.9600	0.8376	0.8947	7458
Probe	0.8131	0.7926	0.8028	2421
R2L	0.9372	0.3954	0.5562	2754
U2R	0.4074	0.0550	0.0969	200
Weighted average	0.8441	0.8239	0.8122	22,544

Table 13 The performance indicators of the enhanced classifier (Num. = 2000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7426	0.9558	0.8358	9711
DoS	0.9585	0.8363	0.8932	7458
Probe	0.8218	0.8191	0.8204	2421
R2L	0.9374	0.3751	0.5358	2754
U2R	0.6087	0.0700	0.1256	200
Weighted average	0.8451	0.8228	0.8102	22,544

Table 14 The performance indicators of the enhanced classifier (Num. = 5000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7412	0.9575	0.8355	9711
DoS	0.9478	0.8395	0.8904	7458
Probe	0.8059	0.7770	0.7912	2421
R2L	0.9785	0.3471	0.5125	2754
U2R	0.2561	0.1050	0.1489	200
Weighted average	0.8412	0.8169	0.8034	22,544

Table 15 The performance indicators of the enhanced classifier (Num. = 8000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.751	0.9664	0.8452	9711
DoS	0.9618	0.8370	0.8950	7458
Probe	0.8275	0.8401	0.8338	2421
R2L	0.9085	0.3460	0.5012	2754
U2R	0.6200	0.1550	0.2480	200
Weighted average	0.8470	0.8270	0.8131	22,544

Table 16 The performance indicators of the enhanced classifier (Num. = 10,000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7472	0.9672	0.8431	9711
DoS	0.9517	0.8410	0.8929	7458
Probe	0.8774	0.8187	0.8470	2421
R2L	0.8564	0.3141	0.4596	2754
U2R	0.2348	0.1350	0.1714	200
Weighted average	0.8376	0.8223	0.8072	22,544

Table 17 The performance indicators of the enhanced classifier (Num. = 20,000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7426	0.9558	0.8358	9711
DoS	0.9585	0.8363	0.8932	7458
Probe	0.8218	0.8191	0.8204	2421
R2L	0.9374	0.3751	0.5358	2754
U2R	0.6087	0.0700	0.1256	200
Weighted average	0.8451	0.8228	0.8102	22,544

Table 18 The performance indicators of the enhanced classifier (Num. = 50,000)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7265	0.9660	0.8293	9711
DoS	0.9464	0.8407	0.8904	7458
Probe	0.8614	0.8315	0.8462	2421
R2L	0.9533	0.2222	0.3604	2754
U2R	0.7857	0.1100	0.1930	200
Weighted average	0.8420	0.8117	0.7884	22,544

Table 19 The performance indicators of the enhanced classifier (Num. = 125,973)

Intrusion type	Precision	Recall	F1-score	Support
Normal	0.7406	0.9453	0.8305	9711
DoS	0.9605	0.8226	0.8862	7458
Probe	0.7982	0.8414	0.8192	2421
R2L	0.8414	0.2985	0.4406	2754
U2R	0.0515	0.0600	0.0554	200
Weighted average	0.8258	0.8067	0.7932	22,544

References

1. Panda M, Abraham A, Das S, Patra MR (2011) Network intrusion detection system: a machine learning approach. Intell Decis Technol 5(4):347–356. https://doi.org/10.3233/IDT-2011-0117
2. Tsai C-F, Hsu Y-F, Lin C-Y, Lin W-Y (2009) Intrusion detection by machine learning: a review. Expert Syst Appl 36(10):11994–12000
3. Reddy RR, Ramadevi Y, Sunitha KVN (2016) Effective discriminant function for intrusion detection using SVM. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 21–24 Sept. 2016, pp 1148–1153. https://doi.org/10.1109/icacci.2016.7732199
4. Wang H, Gu J, Wang S (2017) An effective intrusion detection framework based on SVM with feature augmentation. Knowl Based Syst 136:130–139. https://doi.org/10.1016/j.knosys.2017.09.014
5. Ingre B, Yadav A (2015) Performance analysis of NSL-KDD dataset using ANN. In: 2015 International Conference on Signal Processing and Communication Engineering Systems, 2–3 Jan 2015, pp 92–96. https://doi.org/10.1109/spaces.2015.7058223
6. Akashdeep Manzoor I, Kumar N (2017) A feature reduced intrusion detection system using ANN classifier. Expert Syst Appl 88:249–257. https://doi.org/10.1016/j.eswa.2017.07.005
7. Li W, Yi P, Wu Y, Pan L, Li J (2014) A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J Electr Comput Eng 2014:8. https://doi.org/10.1155/2014/240217
8. Lin W-C, Ke S-W, Tsai C-F (2015) CANN: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowl Based Syst 78:13–21. https://doi.org/10.1016/j.knosys.2015.01.009
9. Masarat S, Sharifian S, Taheri H (2016) Modified parallel random forest for intrusion detection systems. J Supercomput 72(6):2235–2258. https://doi.org/10.1007/s11227-016-1727-6
10. Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153–1176. https://doi.org/10.1109/COMST.2015.2494502
11. Khan JA, Jain N (2016) A survey on intrusion detection systems and classification techniques. Int J Sci Res Sci Eng Technol 2(5):202–208

12. Hodo E, Bellekens X, Hamilton A, Tachtatzis C, Atkinson R (2017) Shallow and deep networks intrusion detection system: a taxonomy and survey. arXiv preprint arXiv:170102145
13. Bengio Y, Delalleau O (2011) On the expressive power of deep architectures. Paper presented at the Proceedings of the 22nd International Conference on Algorithmic Learning Theory, Espoo, Finland
14. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. The MIT Press, Cambridge
15. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training gans. In: Advances in Neural Information Processing Systems, pp 2234–2242
16. Zenati H, Romain M, Foo C, Lecouat B, Chandrasekhar V Adversarially learned anomaly detection. In: Proceedings of the 20th IEEE International Conference on Data Mining (ICDM), Istanbul, Turkey, 2018. IEEE. https://doi.org/10.1109/icdm.2018.00088
17. Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G (2017) Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Niethammer M, Styner M, Aylward S et al (eds) Information processing in medical imaging. Springer International Publishing, Cham, pp 146–157
18. Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA (2018) Generative adversarial networks: an overview. IEEE Signal Process Mag 35(1):53–65. https://doi.org/10.1109/MSP.2017.2765202
19. Usama M, Asim M, Latif S, Qadir J, Ala Al F (2019) Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), 24–28 June 2019, pp 78–83. https://doi.org/10.1109/iwcmc.2019.8766353
20. Lin Z, Shi Y, Xue Z (2018) IDSGAN generative adversarial networks for attack generation against intrusion detection. arXiv:180902077
21. Yan Q, Wang M, Huang W, Luo X, Yu FR (2019) Automatically synthesizing DoS attack traces using generative adversarial networks. Int J Mach Learn Cybern. https://doi.org/10.1007/s13042-019-00925-6
22. Merino T, Stillwell M, Steele M, Coplan M, Patton J, Stoyanov A, Deng L (2020) Expansion of cyber attack data from unbalanced datasets using generative adversarial networks. In: Lee R (ed) Software Engineering Research, Management and Applications. Springer International Publishing, Cham, pp 131–145. https://doi.org/10.1007/978-3-030-24344-9_8
23. Zhang H, Yu X, Ren P, Luo C, Min G (2019) Deep adversarial learning in intrusion detection: a data augmentation enhanced framework. arXiv preprint arXiv:190107949 https://doi.org/10.13140/rg.2.2.19731.73762
24. Farid DM, Zhang L, Rahman CM, Hossain MA, Strachan R (2014) Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. Expert Syst Appl 41(4):1937–1946. https://doi.org/10.1016/j.eswa.2013.08.089
25. Kanakarajan NK, Muniasamy K (2016) Improving the accuracy of intrusion detection using GAR-Forest with feature selection. In: Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015. Springer, pp 539–547. https://doi.org/10.1007/978-81-322-2695-6_45
26. Kuang F, Xu W, Zhang S (2014) A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl Soft Comput 18:178–184. https://doi.org/10.1016/j.asoc.2014.01.028
27. Ikram ST, Cherukuri AK (2016) Improving accuracy of intrusion detection model using PCA and optimized SVM. J Comput Inf Technol 24(2):133–148. https://doi.org/10.20532/cit.2016.1002701
28. Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Graph Stat 15(2):265–286. https://doi.org/10.1198/106186006X113430
29. Liu Y-J, Chen T, Yao Y (2013) Nonlinear process monitoring by integrating manifold learning with Gaussian process. In: Computer Aided Chemical Engineering, vol 32. Elsevier, pp 1009–1014. https://doi.org/10.1016/b978-0-444-63234-0.50169-x
30. Bengio Y (2009) Learning deep architectures for AI. Foundations and trends® in Machine Learning 2 (1):1-127. https://doi.org/10.1561/2200000006
31. Abolhasanzadeh B (2015) Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features. In: 2015 7th Conference on Information and Knowledge Technology (IKT). IEEE, pp 1–5. https://doi.org/10.1109/ikt.2015.7288799
32. Gao N, Gao L, Gao Q, Wang H (2014) An intrusion detection model based on deep belief networks. In: 2014 Second International Conference on Advanced Cloud and Big Data (CBD). IEEE, pp 247–252. https://doi.org/10.1109/cbd.2014.41

33. Wei P, Li Y, Zhang Z, Hu T, Li Z, Liu D (2019) An optimization method for intrusion detection classification model based on deep belief network. IEEE Access 7:87593–87605. https://doi.org/10.1109/ACCESS.2019.2925828
34. Potluri S, Ahmed S, Diedrich C (2018) Convolutional neural networks for multi-class intrusion detection system. In: Groza A, Prasath R (eds) Mining Intelligence and Knowledge Exploration, Cham, 2018. Springer International Publishing, pp 225–238. https://doi.org/10.1007/978-3-030-05918-7_20
35. Javaid A, Niyaz Q, Sun W, Alam M A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp 21–26. https://doi.org/10.4108/eai.3-12-2015.2262516
36. Mao X, Li Q, Xie H, Lau RY, Wang Z (2016) Multi-class generative adversarial networks with the L2 loss function. CoRR, abs/161104076 2
37. Odena A (2016) Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:160601583
38. Springenberg JT (2015) Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:151106390
39. Zhang D, Niu Q, Qiu X (2019) Detecting anomalies in communication packet streams based on generative adversarial networks. In: Biagioni ES, Zheng Y, Cheng S (eds) Wireless Algorithms, Systems, and Applications, Cham, 2019. Springer International Publishing, pp 470–481. https://doi.org/10.1007/978-3-030-23597-0_38
40. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp 2672–2680
41. Adetunmbi AO, Falaki SO, Adewale OS, Alese BK (2008) Network intrusion detection based on rough set and k-nearest neighbour. Int J Comput ICT Res 2(1):60–66
42. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications. CISDA 2009. IEEE, pp 1–6. https://doi.org/10.1109/cisda.2009.5356528
43. Ingre B, Yadav A, Soni AK (2017) Decision tree based intrusion detection system for NSL-KDD dataset. In: International Conference on Information and Communication Technology for Intelligent Systems, 2017. Springer, pp 207–218. https://doi.org/10.1007/978-3-319-63645-0_23
44. Aljawarneh SA, Vangipuram R (2018) GARUDA: Gaussian dissimilarity measure for feature representation and anomaly detection in Internet of things. J Supercomput. https://doi.org/10.1007/s11227-018-2397-3
45. Choi H, Kim M, Lee G, Kim W (2019) Unsupervised learning approach for network intrusion detection system using autoencoders. J Supercomput 75(9):5597–5621. https://doi.org/10.1007/s11227-019-02805-w