IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 29, 2020, accepted February 20, 2020, date of publication February 28, 2020, date of current version March 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977007

Network Intrusion Detection Based on Supervised Adversarial Variational Auto-Encoder With Regularization

YANQING YANG[1,2], KANGFENG ZHENG[1], BIN WU[1], YIXIAN YANG[1,3], AND XIUJUAN WANG[4]

[1]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[3]Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
[4]College of Computer Sciences, Beijing University of Technology, Beijing 100124, China

Corresponding author: Kangfeng Zheng (zkf_bupt@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802703, and in part by the National Natural Science Foundation of China under Grant 61602052.

ABSTRACT To explore the advantages of adversarial learning and deep learning, we propose a novel network intrusion detection model called SAVAER-DNN, which can not only detect known and unknown attacks but also improve the detection rate of low-frequent attacks. SAVAER is a supervised variational auto-encoder with regularization, which uses WGAN-GP instead of the vanilla GAN to learn the latent distribution of the original data. SAVAER's decoder is used to synthesize samples of low-frequent and unknown attacks, thereby increasing the diversity of training samples and balancing the training data set. SAVAER's encoder is used to initialize the weights of the hidden layers of the DNN and explore high-level feature representations of the original samples. The benchmark NSL-KDD (KDDTest+), NSL-KDD (KDDTest-21) and UNSW-NB15 datasets are used to evaluate the performance of the proposed model. The experimental results show that the proposed SAVAER-DNN is more suitable for data augmentation than the other three well-known data oversampling methods. Moreover, the proposed SAVAER-DNN outperforms eight well-known classification models in detection performance and is more effective in detecting low-frequent and unknown attacks. Furthermore, compared with other state-of-the-art intrusion detection models reported in the IDS literature, the proposed SAVAER-DNN offers better performance in terms of overall accuracy, detection rate, F1 score, and false positive rate.

INDEX TERMS Intrusion detection, supervised adversarial variational auto-encoder, regularization, WGAN-GP, deep learning.

I. INTRODUCTION

With the continuous development and widespread application of the Internet, big data, cloud computing, Internet of Things, and industrial control network technologies, the number of user devices connected to the network has increased dramatically, and the network has spread to all walks of life. However, with the rapid development of the network, network security events frequently occur in recent years, and network information security is facing huge challenges. For example, on January 29, 2018, the top three banks in the Netherlands have been targeted in multiple cyber attacks over the past week, blocking access to websites and internet banking services [1]. Early on the morning of March 18, 2019, Norsk Hydro, one of the world's largest aluminum companies, was attacked by a new variant of ransomware called LockerGoga, which forced some of its automated production lines in Europe and the United States to shut down [2]. On November 11, 2019, the Labor Party suffered a DDoS (Distributed Denial of Service) attack that affected the party website, online campaigning tools and platforms after they were flooded with millions of requests [3]. Therefore, timely detection of intruders is an important step to ensure network security. As an effective means to ensure network security, intrusion detection technology can quickly detect network attacks and issue immediate warnings. Intrusion detection system (IDS) can be divided into two categories: host-based intrusion detection system (HIDS) and

The associate editor coordinating the review of this manuscript and approving it for publication was Zehua Guo.

network-based intrusion detection system (NIDS). This paper focuses on a network-based intrusion detection system.

In recent years, machine learning has been widely used to identify various types of network attacks in NIDS, thereby helping administrators take appropriate measures to prevent network intrusions. However, most traditional machine learning methods are shallow learning techniques, such as KNN (k-Nearest Neighbor) [4], SVM (Support Vector Machine) [5], SOM (Self-Organizing Map) [6] and so on. There are many problems in shallow learning methods, including a heavy dependence on feature engineering and feature selection, poor ability to detect unknown network attacks and high false alarm rates. In addition, they can not effectively solve the classification of large-scale intrusion data in the actual complex network application environment. In contrast, deep learning methods can automatically extract better high-level abstract features from the data to create better detection models [7].

Deep learning technologies such as DBN (Deep Belief Network), DNN (Deep Neural Network), CNN (Convolutional Neural Network), LSTM (Long-Term Short-Term Memory), and GAN (Generate Adversarial Network), etc. have been widely used in computer vision, natural language processing, speech recognition, drug design, intrusion detection and other fields, which can produce results that are comparable to or even better than human experts [8]–[11]. However, these technologies still have many problems. First, with the rapid development of network technology and the rapid growth of network traffic data, different types of attack traffic are imbalanced. It is difficult for traditional classifiers to achieve high detection rates on imbalanced data sets. Second, due to the widespread application of new technologies such as artificial intelligence (AI), more and more unknown attacks threaten the security of the Internet and Intranet. Traditional classifiers perform well on known attacks but detect poorly on unknown attacks. Third, with the popularity of the Internet of Things and the widespread application of cloud-based services, the scale and complexity of network traffic data continue to increase, and it is difficult for traditional classifiers to distinguish between normal and abnormal behaviors.

To address the above problems, we propose a novel network intrusion detection model called SAVAER-DNN. The proposed model is a combination of supervised adversarial variational auto-encoder with regularization (SAVAER) and deep neural network (DNN), which can correctly detect various network attacks and is more suitable for running in modern networks. SAVAER uses WGAN-GP (Wasserstein GAN with gradient penalty) instead of the vanilla GAN (Generative Adversarial Network) to train the adversarial learning model. More specifically, we combine the power of VAE data generation with the ability of the GAN's adversarial learning to better learn the latent representation of the original data. The trained SAVAER encoder is used to construct a DNN classifier to detect network attacks. We have evaluated the proposed model on the benchmark NSL-KDD [12], [13] and UNSW-NB15 [14]–[18] datasets and obtained promising results.

This article provides the following novel contributions:

- By integrating the power of supervised VAE data generation and the advantages of WGAN-GP adversarial learning, a new feature learning and data synthesis technique is proposed, called SAVAER. SAVAER regularizes the latent representation of the original data by providing a one-hot class vector to the discriminator network.
- Unlike SAAE, SAVAER uses VAE instead of AE to characterize the latent distribution of attack samples. At the same time, class tags are fed to the decoder and discriminator to regulate independence between latent variables and classes.
- SAVAER uses WGAN-GP instead of vanilla GAN to train the adversarial learning model, which overcomes the shortcomings of GAN's difficulty in convergence and model collapse.
- SAVAER's decoder is used to synthesize low-frequent and unknown attack samples of a specified label, thereby increasing the diversity of training samples and balancing the training data set. As a result, the detection rate of low-frequent and unknown attacks is improved.
- By combining the advantages of SAVAER encoder feature extraction and DNN complex decision making, we propose a SAVAER-DNN model for network intrusion detection. The trained encoder is used to initialize the parameters of the hidden layer of the DNN and explore the high-level abstract features of the original data. DNN adjusts network parameters through backpropagation and fine-tuning techniques to better handle the detection of a large number of complex network attacks.
- SAVAER-DNN can not only accurately detect known and unknown attacks, but also effectively detect low-frequent attacks.
- Compared with the state-of-the-art intrusion detection models reported in the IDS literature, SAVAER-DNN has achieved the highest overall performance on the benchmark datasets.

The rest of this article is structured as follows. Section II reviews related research in the field of intrusion detection. Section III presents relevant background information and proposes a modified model. Section IV details the proposed intrusion detection framework. Compared with the well-known methods and the state-of-the-art classification models, the experimental details and results of the proposed model are shown in Section V. Finally, this article provides some conclusions and further work in Section VI.

II. RELATED WORKS

In recent years, the popularity of deep learning has made it widely used to identify various types of network attacks. As deep learning overcomes the shortcomings of shallow learning, and can automatically extract high-level features

and perform complex classification tasks, the research of deep learning application in network intrusion detection has attracted widespread attention from scholars at home and abroad.

Ma *et al.* [19] proposed a new hybrid method called SCDNN for network intrusion detection. SCDNN consists of spectral clustering (SC) and deep neural network (DNN). First, the SC divides the original training data set into k training subsets, and then the training subset is used to train the k sub-DNN classifiers. Next, the test data set is divided into subsets with SC, and the test data subsets are used to test the corresponding sub-DNNs. SCDNN is evaluated on six KDD-Cup99 and NSL-KDD datasets. The experimental results show that SCDNN has better detection accuracy than SVM, BPNN (backpropagation neural network), RF (random forest) and Bayesian methods. The accuracy of SCDNN on the NSL-KDD (KDDTest +) and NSL-KDD (KDDTest-21) datasets is 72.64% and 44.55%, respectively.

In [20], the authors proposed an unsupervised network intrusion detection method based on a conditional variational auto-encoder, called ID-CVAE. This method has a specific architecture that integrates intrusion tags only inside the decoder layer. Experimental results show that the proposed ID-CVAE provides better classification results than other well-known classifiers. The accuracy of ID-CVAE on the NSL-KDD dataset reaches 80.10%. More importantly, the method can recover missing features from incomplete training data sets.

Yin *et al.* [21] proposed a deep learning method for network intrusion detection using recurrent neural networks, called RNN-IDS. They study the accuracy and training time of RNN-IDS with different learning rates and hidden nodes in both binary and multi-class classification. Experimental results show that RNN-IDS is very suitable for network intrusion detection, and achieves the accuracy of 83.28% and 68.55% on KDDTest + and KDDTest-21 test sets, respectively.

Li *et al.* [22] proposed a deep learning approach to automatically learn the features of graphic NSL-KDD transformation via the proposed graphic conversion technique. They use convolutional neural networks (CNN) ResNet and GoogLeNet for network intrusion detection. The performance of the image conversion method is evaluated by binary classification experiments on the NSL-KDD dataset. Different structures of CNN are testified for comparison. Experimental results show that the CNN model is very sensitive to image transformation of attack data, and the method is suitable for intrusion detection.

Shone *et al.* [7] proposed a novel deep learning classification model, called S-NDAE. The proposed S-NDAE uses stacked nonsymmetric deep auto-encoders (NDAEs) and random forest (RF) to build a classification model for intrusion detection. NDAE is used to learn features, and the stacked NDAEs and RF are used to perform classification. Experimental results show that the proposed model has obtained promising results on the KDD Cup'99 and NSL-KDD datasets, and has achieved the highest accuracy of 85.42% on the NSL-KDD dataset.

Khan *et al.* [23] proposed a new two-stage deep learning (TSDL) model based on stacked auto-encoders with a soft-max classifier for network intrusion detection. The model includes two decision-making phases: First, the network traffic is classified as normal or abnormal using a probability value. Next, the probability value is added to the original features as an additional feature to detect normal and other types of attacks. To evaluate the performance of the proposed model, extensive experiments are performed on the KDD99 and UNSW-NB15 datasets. Comparative simulation results show that the proposed model significantly outperforms the existing methods, and has achieved 99.996% and 89.134% accuracy on the KDD99 and UNSW-NB15 datasets, respectively.

In [24], the authors used LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) with a variable number of hidden layers and BLS (Extended Learning System) and its extensions to build network anomaly detection models. The BGP and NSL-KDD datasets are used to evaluate the performance of the proposed models in terms of training time, accuracy, and F1 score. On the BPG data set, the experimental results show that RNN and BLS models can provide the best accuracy and F1-score in the range of 90%-95%. On the NSL-KDD dataset, the experimental results show that the best performance can be obtained using $LSTM_4$ and GRU_3 RNNs, and the CFBLS (BLS with cascades of mapped features) architecture can provide the best results.

Vinayakumar *et al.* [25] employed distributed DNNs model to develop a scalable and hybrid intrusion detection model, called scale-hybrid-IDS-AlertNet (SHIA). The proposed SHIA can effectively monitor a large number of network-level and host-level events to automatically identify malicious attacks in order to provide network administrators with appropriate alerts. Strict experimental tests on various benchmark IDS datasets show that the proposed model performs well compared with traditional machine learning classifiers.

In summary, although the aforementioned deep learning methods have achieved satisfactory results in network intrusion detection systems, they still face the problem of low detection rates of unknown and low-frequent attacks. To solve these problems, we propose a new hybrid intrusion detection framework. The proposed framework uses SAVAER to learn the latent distribution of the intrusion data, then concatenates the encoded latent vectors with the specified attack labels, and feeds them together into the decoder to generate unknown attacks. Thus, the diversity of training samples is improved and the training sample set is balanced. Furthermore, the SAVAER encoder adds a softmax layer to build a DNN classifier. Finally, the DNN classifier automatically explores high-level abstract feature representation of network attack data and effectively detects unknown and low-frequent cyber attacks.

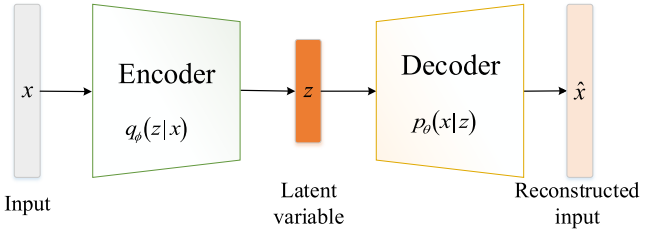


FIGURE 1. AE architecture.

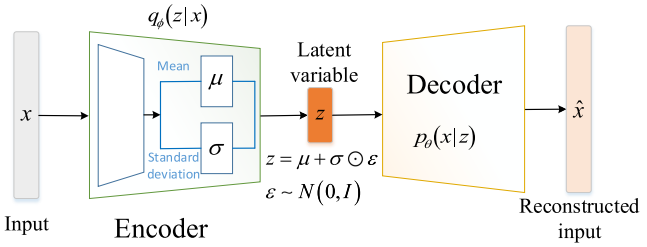


FIGURE 2. VAE architecture.

III. METHODOLOGY

A. AUTO-ENCODER

Auto-Encoder (AE) is an artificial neural network that learns how to efficiently compress and encode data and then learns how to reconstruct data as close as possible to the original input data [26]. AE consists of an encoder and a decoder, as shown in Figure 1, and is usually trained in an unsupervised manner. The encoder is often used to reduce dimensionality, while the decoder is a decompression process that can be used to denoise raw data or generate new data. There are several variations of the auto-encoders, including regularized, sparse, contractive, and denoising auto-encoders, and then Variational Auto-Encoder (VAE), beta-VAE [27] and TD-VAE (Temporal Difference VAE) [28].

AE tries to minimize the reconstruction error between the input data $\boldsymbol{x}$ and the reconstructed data $\hat{\boldsymbol{x}}$. The reconstruction loss can be defined as the L_p distance (like L_2 norm):

$$\mathcal{L}_{\mathrm{AE}}(\theta, \phi) = \min E\left(\boldsymbol{x}, \hat{\boldsymbol{x}}\right) \overset{e.g.}{=} \min \left\|\boldsymbol{x} - \hat{\boldsymbol{x}}\right\|_p \tag{1}$$

The reconstruction loss can also be defined as cross entropy function:

$$\mathcal{L}_{\mathrm{AE}}(\theta, \phi) = -\sum_{i=1}^{D} \hat{x}_i \log\left(x_i\right) \tag{2}$$

where D is the dimensionality of the input data $\boldsymbol{x}$.

Just as a standard Auto-Encoder, Variational Auto-Encoder (VAE) [29], [30] is a deep generation model with latent variables, especially suitable for generating new samples that look like real data. VAE is an architecture composed of both an encoder and a decoder, which is trained to minimize the reconstruction error between the encoded-decoded data and the input data, as shown in Figure 2. VAE uses Bayesian inference and probabilistic graphical model methods to encode input data into a low-dimensional latent encoding space and then decodes it back.

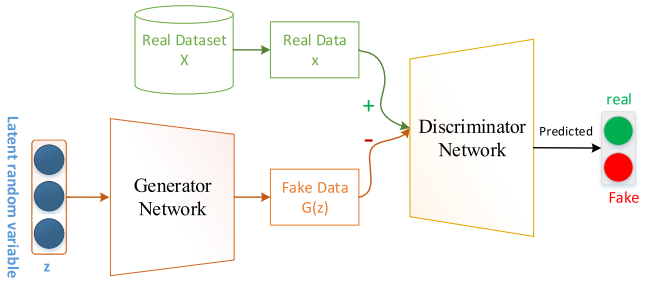


FIGURE 3. GAN architecture.

The probabilistic encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is a posterior probability function, which is used to approximate the intractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. We assume that the prior distribution $p_\theta(\boldsymbol{z})$ is a multivariate Gaussian with a diagonal covariance matrix. We randomly sample the point z from a prior distribution $p_\theta(\boldsymbol{z})$. To make it trainable, the reparameterization trick is introduced [30]. The decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ maps this point in the latent space back to the original input sample. The loss function of VAE is defined as:

$$\mathcal{L}_{\mathrm{VAE}}(\theta, \phi) = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + D_{\mathrm{KL}}[q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})] \tag{3}$$

where D_{KL} is the Kullback–Leibler divergence, which intuitively measures the degree of similarity between the prior distribution $p_\theta(\boldsymbol{z})$ and the posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.

B. GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) are one of the most popular deep learning algorithms in recent years. They have been used in real-life applications for text, image and video generation, drug discovery and data synthesis. GANs can learn to mimic any data distribution, originally introduced in 2014 by Ian Goodfellow and co-authors including Goodfellow *et al.* [31]. GANs consist of two neural networks in a two-player game, a generation network (a generator) and a discriminating network (a discriminator), as shown in Figure 3. The discriminator D is a binary classifier that learns to distinguish between real and synthetic samples, and the generator G captures the latent distribution of real samples and generates real-looking synthetic samples that can fool the discriminator D.

The vanilla GAN uses the following objective function, which can be interpreted as "minimizing Jensen–Shannon divergence between fake and real distributions":

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log D(\boldsymbol{x})] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\boldsymbol{x}}))] \tag{4}$$

where $\tilde{\boldsymbol{x}} = G(\boldsymbol{z})$ is fake data generated. $\boldsymbol{z}$ represents a random noise point sampled from the distribution $p(z)$. $\mathbb{P}_r$ and $\mathbb{P}_g$ represent the distribution of real and fake data, respectively.

GAN is a minimax game that is difficult to train and suffers from the following major problems: (1) no convergence, model parameter oscillation; (2) mode collapse, the generator

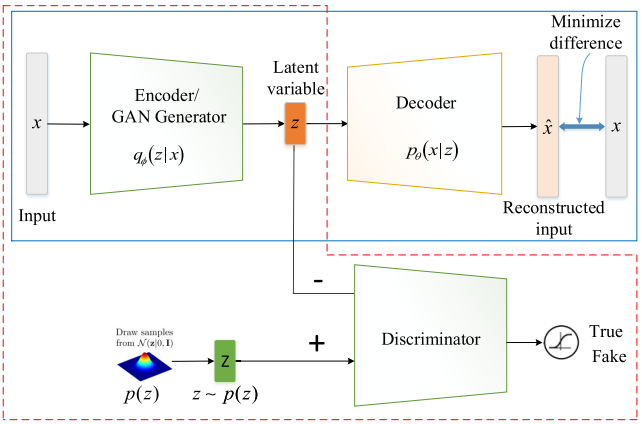


FIGURE 4. AAE architecture.

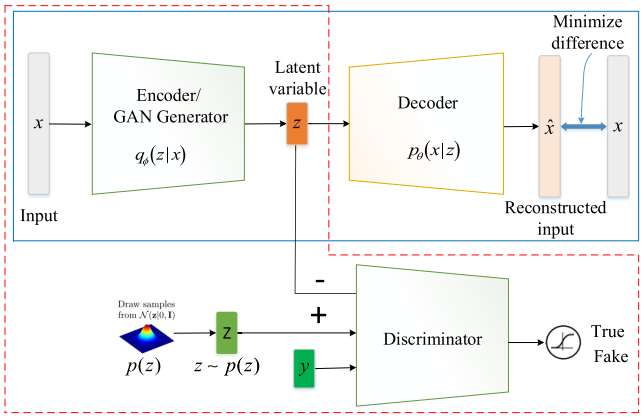


FIGURE 5. AAER architecture.

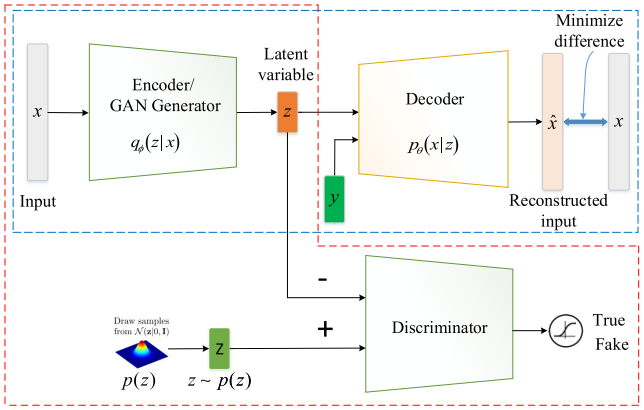


FIGURE 6. SAAE architecture.

produces limited varieties of data samples; (3) vanishing gradient, the discriminator is so successful that the gradient of the generator vanishes and the generator learns nothing [32]. To solve these problems, Arjovsky *et al.* [33] proposed an improved algorithm, called WGAN. WGAN uses Wasserstein distance instead of JS (Jensen–Shannon) divergence to characterize the difference between fake and real distributions. When the discrepancy between fake and real distributions is too large, JS divergence can not provide enough information. In contrast, even if the two distributions do not overlap, the Wasserstein distance can accurately calculate the difference. WGAN tries to optimize its dual form:

$$\min_G \max_{D \in \mathcal{D}} V(D, G) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] \quad (5)$$

where $\tilde{\boldsymbol{x}} = G(\boldsymbol{z})$, $\mathcal{D}$ is the set of Lipschitz-1 continuous functions. To ensure $D \in \mathcal{D}$ after all gradients are updated, WGAN clips the weights of discriminator into a small fixed range, such as $W \in [-0.01, 0.01]$, resulting in a compact parameter space W.

However, WGAN is not perfect. Weight clipping may cause most of the discriminator's weight values to focus on two limit values, such as -0.01 or 0.01, which can easily cause slow convergence and vanishing gradients [34]. In order to solve these problems, Gulrajani *et al.* [34] made some improvements to replace weight clipping with gradient penalty, called WGAN-GP (Wasserstein Generative Adversarial Network with Gradient Penalty). The WGAN-GP objective is defined as:

$$\mathcal{L}_{\text{WGAN-GP}} = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{x}}} \left[\left(\left\| \nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}) \right\|_2 - 1 \right)^2 \right] \quad (6)$$

where $\tilde{\boldsymbol{x}} = G(\boldsymbol{z})$, $\hat{\boldsymbol{x}} = \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$, ϵ is a random variable $\epsilon \sim \mathbb{U}[0, 1]$. λ is a penalty factor and is usually set to 10. $\mathbb{P}_{\hat{x}}$ represents the uniform sampling distribution along the line between the real distribution $\mathbb{P}_r$ and the fake distribution $\mathbb{P}_g$.

C. ADVERSARIAL AUTO-ENCODER

Adversarial Auto-Encoder (AAE) was proposed by Makhzani *et al.* in 2015 [35], as shown in Figure 4. AAE behaves similarly to VAE, which forces the latent variable $\boldsymbol{z}$ of AE to follow a predefined prior distribution $p(\boldsymbol{z})$. In the case of the AAE, $\boldsymbol{z}$ can be arbitrarily defined and easily sampled and fed into the discriminator. However, instead of maximizing the evidence lower bound (ELBO) like VAE, AAE utilizes the GAN to guides the distribution of the encoder $q(\boldsymbol{z}|\boldsymbol{x})$ to match the prior distribution $p(\boldsymbol{z})$. The discriminator network is trained to discriminate whether the input code vector $\boldsymbol{z}$ is fake data from the AE or true data from the prior distribution $p(\boldsymbol{z})$.

In the scenario where the training data is labeled, Makhzani *et al.* [35] proposed an improved version of AAE, which incorporates label information during the adversarial training phase, called adversarial auto-encoder with regularization (AAER), as shown in Figure 5. The one-hot label is fed to the discriminator network to provide a better fit distribution for the latent variables.

Supervised Adversarial Auto-Encoder (SAAE) is similar to AAE, but the only difference from AAE is that the one-hot label is used as input of the decoder, thereby changing unsupervised training to a supervised manner, as shown in Figure 6. SAAE concatenates the latent variable $\boldsymbol{z}$ and the one-hot label $\boldsymbol{y}$ together as the input of the decoder. Then the decoder forces the network to learn the label-independent

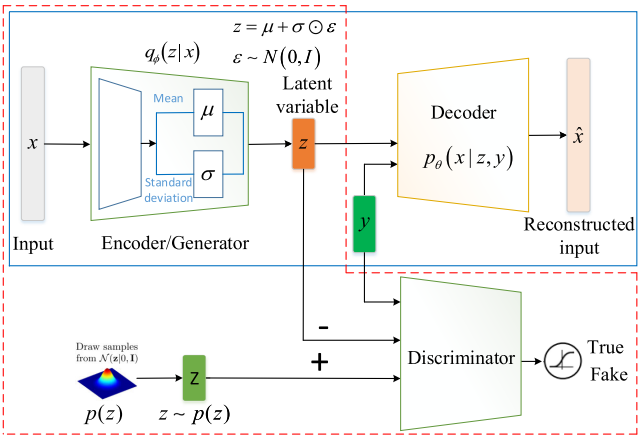


FIGURE 7. SAVAER architecture.

latent distribution. In SAAE, we can generate new attack samples with the specified labels, but in AAE, this is not possible.

D. PROPOSED METHODOLOGY

In order to combine the advantages of VAE and SAAE, we propose a Supervised Adversarial Variational Auto-Encoder with regularization (SAVAER), as shown in Figure 7. The main difference from the previous SAAE model is that SAVAER uses VAE instead of AE to characterize the latent distribution of attack samples, and the one-hot class vectors are fed to the decoder and discriminator to regulate independence between latent variables and classes. The decoder can generate attack samples of the specified label y from the distribution $p_\theta(x|z, y)$.

IV. PROPOSED NETWORK INTRUSION DETECTION FRAMEWORK

In this section, we introduce the details about the proposed network intrusion detection model. The proposed model consists of four phases (in order): data preprocessing, training SAVAER, data augmentation and detecting attacks, as shown in Figure 8.

A. DATA PREPROCESSING

SAVAER-DNN only accepts numerical values for training and testing, so we use one-hot encoding to transfer all the symbolic feature values on the NSL-KDD and UNSW-NB15 datasets into numerical feature values. For example, the NSL-KDD dataset contains three symbolic features, including protocol_type (i.e., tcp, udp, and icmp), service (such as ftp_data, ssh, http, etc.) and flag (such as SF, S0, SH, etc.). After data transferring, all symbolic attributes are converted to numerical values. Data scaling (also known as data normalization) is used to normalize the range of data features, which can speed up the gradient descent of machine learning algorithms to find the optimal solution. We use the maximum-minimum normalization method to scale feature values. All feature values are normalized within a specific

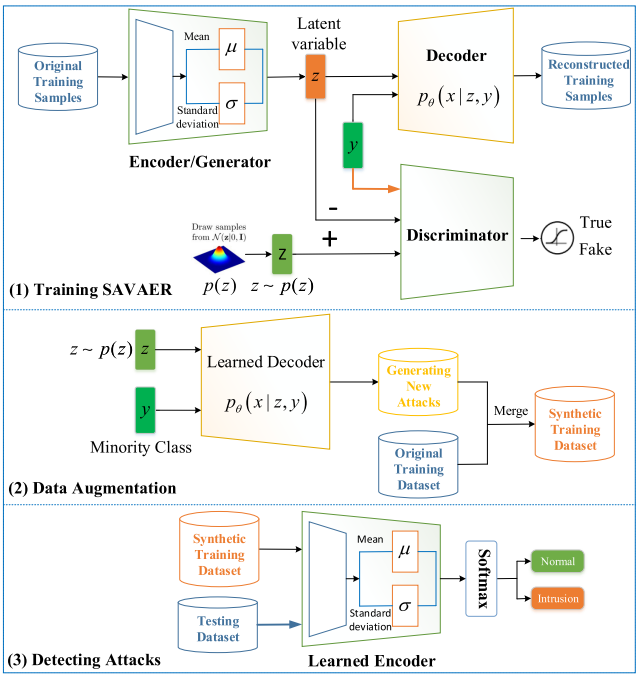


FIGURE 8. The proposed network intrusion detection framework.

range of [0, 1] according to Equation 7.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{7}$$

where x is an original value, x' is the normalized value.

B. TRAINING SAVAER

In the training phase, we use the label shuffling [36] method to train the proposed SAVAER on the original training data set. SAVAER is trained in the following steps for each mini-batch: (1) VAE is trained to minimize the difference between the reconstructed data $\hat{x}$ and the original input data x, that is, to minimize the binary cross-entropy loss. (2) The discriminator is trained to distinguish the differences between fake and real samples z from the real data distribution $q(z)$ and the multivariate Gaussian prior $p(z)$, that is, to minimize the WGAN-GP loss. (3) The generator (encoder) is trained to fool the parameter-fixed discriminator by generating the most similar samples z. The training process is an iterative training of VAE, discriminator, and generator.

After SAVAER is trained, the original training data (x, y) is fed into the trained VAE to calculate the cross-entropy reconstruction loss. The binary cross-entropy reconstruction loss of the sample (x_i, y_i) is defined as:

$$L(x_i, y_i) = -\sum_{k=1}^{d} [x_{i,k} \log \hat{x}_{i,k} + (1 - x_{i,k}) log(1 - \hat{x}_{i,k})] \tag{8}$$

where $\hat{x}_i = Decoder(Encoder(x_i), y_i)$; $x_{i,k}$ represents the k-th feature value of the sample x_i; d represents the number of features.

The maximum class reconstruction loss $maxLj$ of class j is defined as:

$$maxL_j = k * \max\{L(x_i, y_i), \text{for each } y_i \in \text{class } j\} \quad (9)$$

where k is the scale factor, and usually k is set to 1.0.

C. DATA AUGMENTATION

In the data augmentation phase, a trained decoder can be used to generate new attack samples to improve the classifier's detection rate of unknown and minority attacks. Now you can sample the latent variable z_{new} from the multivariate Gaussian distribution $p(z)$, concatenate it with the specified minority class label y_{new}, and feed them into the decoder to generate new attacks x_{new}. The newly generated sample (x_{new}, y_{new}) is fed into the trained SAVAER, and the reconstruction error $L(x_{new}, y_{new})$ of the newly generated sample is calculated according to Equation 8. In order to ensure that the newly generated samples have the same spatial distribution as the original samples, we filter the newly generated attack samples (x_{new}, y_{new}) according to Equation 10 to eliminate the samples that differ significantly from the original sample distribution. Finally, the newly generated attack samples which are consistent with the original data distribution are merged into the original training data set to balance the training data set.

$$S = \begin{cases} S \bigcup \{x_{new}, y_{new}\}, \text{if } y_{new} \in \text{class } j \\ \qquad\qquad and\ L\ (x_{new}, y_{new}) \leq maxL_j \\ S, \qquad\qquad otherwise \end{cases} \quad (10)$$

D. DETECTING ATTACKS

In the detection attack phase, we use the trained SAVAER encoder to construct a DNN classifier, that is to say, we add a softmax output layer to the last layer of the encoder. The weights of the trained SAVAER encoder are used to initialize the weights of the hidden layers of the DNN. We use the synthetic training dataset to train the DNN classifier. First, the hidden layer weights of the DNN classifier are frozen, the weight of the output layer is updated by backpropagation, then all hidden layers are unfrozen, and the DNN classifier is fine-tuned by using the synthetic training data set.

The proposed intrusion detection model is detailed in algorithm 1:

Algorithm 1 Proposed Network Intrusion Detection Framework

Input: Training dataset $S = (x, y)$, latent variable z, learning rate lr, the scale factor k.

Output: The final attack recognition results.

1: Data preprocessing: each feature is scaled to a given range [0,1] by a one-hot encoding and a maximum-minimum normalization method.
2: Train SAVAER on S using a label shuffling method, with multivariate Gaussian distribution as prior $p(z)$:
3: $D(E(\cdot)) \leftarrow$ VAE in SAVAER;
4: $Dis(\cdot) \leftarrow$ discriminator in SAVAER;
5: $E(\cdot) \leftarrow$ generator (encoder) in SAVAER.
6: Calculate the maximum class reconstruction loss $maxL$ according to Equation 9.
7: Generate new attack samples:
8: $z_{new} \leftarrow$ multivariate Gaussian distribution $p(z)$;
9: $y_{new} \leftarrow$ minority class labels;
10: $x_{new} \leftarrow D(z_{new}, y_{new})$.
11: Merge the newly generated samples (x_{new}, y_{new}) into the original training data set according to Equation 10.
12: Train DNN classifier on the synthetic training data set:
13: $DNN(\cdot) \leftarrow E(\cdot)$, use the SAVAER encoder's weights to initialize the weights of the hidden layers of the DNN;
14: Freeze hidden layers of DNN, and update the weights of the DNN output layer using the backpropagation algorithm;
15: Unfreeze hidden layers of DNN, and fine-tune the DNN classifier using the synthetic training data set.
16: **return** the classification results.

V. EXPERIMENTAL RESULTS AND ANALYSES

A. DESCRIPTION OF THE BENCHMARK DATASETS

At present, only a few benchmark data sets can be used to evaluate a network intrusion detection system. These data sets reported in the literature include KDD'99 [37], NSL-KDD [12], [13], ISCXIDS2012 [38], CICIDS2017 [39], CICDoS2017 [40], CICDDoS2019 [41], and UNSW-NB15 [14]–[18] data sets. The NSL-KDD dataset is an improved version of the KDD'99 dataset, which overcomes many problems of the original KDD'99 dataset. The ISCXIDS2012 dataset contains seven days of normal and malicious network records, and the CICIDS2017 data set contains five days of network attack traffic, including DoS, DDoS, Brute Force SSH, Brute Force FTP, Web Attack, Botnet, Heartbleed and Infiltration. The CICDoS2017 and CICDDoS2019 datasets are primarily used to evaluate DoS and DDoS attacks. The UNSW-NB15 dataset contains benign and nine latest common attacks.

The NSL-KDD and UNSW-NB15 data sets are more suitable for evaluating several attacks, so we chose them to evaluate the detection performance of the proposed model.

1) NSL-KDD DATASET

The NSL-KDD dataset is a new compressed version of the KDD'99 dataset, which eliminates the duplicate and redundant records in the KDD'99 data set. In addition, the number of records in the NSL-KDD training and test data sets is more reasonable. The NSD-KDD dataset contains normal and four attack records, named Probe, DoS (Denial of Service), U2R (User to Root), and R2L (Remote to Local). We use KDDTrain+_20Percent.txt as the training set and KDDTest+.txt, and KDDTest-21.txt (remove records that are easily classified by 21 common classifiers in the KDDTest+.txt data set) [13], [42] as the test sets. The number of records for each category on the NSL-KDD dataset is shown in Table 1. As can be seen from Table 1, the NSL-KDD training data set is imbalanced, and more than half of the

TABLE 1. Class distribution of the NSL-KDD dataset.

Class	Attack	Training set	Testing set	
		KDDTrain+_20	KDDTest+	KDDTest-21
Normal	normal	13449	9711	2152
Subtotal		13449	9711	2152
Probe	ipsweep	710	141	141
	satan	691	735	727
	portsweep	587	157	156
	nmap	301	73	73
	saint	/	319	309
	mscan	/	996	996
Probe		2289	2421	2402
DoS	neptune	8282	4657	1579
	smurf	529	665	627
	back	196	359	359
	teardrop	188	12	12
	pod	38	41	41
	land	1	7	7
	apache2	/	737	737
	mailbomb	/	293	293
	processtable	/	685	685
	udpstorm	/	2	2
Subtotal		9234	7458	4342
U2R	buffer_overflow	6	20	20
	rootkit	4	13	13
	loadmodule	1	2	2
	perl	/	2	2
	httptunnel	/	133	133
	ps	/	15	15
	sqlattack	/	2	2
	xterm	/	13	13
Subtotal		11	200	200
R2L	guess_passwd	10	1231	1231
	warezmaster	7	944	944
	imap	5	1	1
	multihop	2	18	18
	phf	2	2	2
	ftp_write	1	3	3
	spy	1	/	/
	warezclient	181	/	/
	named	/	17	17
	sendmail	/	14	14
	xlock	/	9	9
	xsnoop	/	4	4
	worm	/	2	2
	snmpgetattack	/	178	178
	snmpguess	/	331	331
Subtotal		209	2754	2754
Total		25192	22544	11850

U2R and R2L attacks in the test data set do not appear in the training data set.

2) UNSW-NB15 DATASET

The UNSW-NB15 dataset [14]–[18] consists of a hybrid of real modern normal activities and synthetic attack behaviours, created by ACCS (the Australian Centre for Cyber Security). The full UNSW-NB15 dataset contains 49 features with class labels, with a total of 25,400,443 records. The full UNSW-NB15 dataset is divided into a training set and a test set by stratified sampling, namely, UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv. The number of records in the training set and test set is 175,341 and 82,332 records, respectively. The divided data set removes six features (such as srcip, sport, dstip, dsport, Stime, and Ltime) from the full data set, with only 43 features with class labels [15]. The UNSW-NB15 dataset contains one normal and nine attacks, named such as DoS, Shellcode, Backdoor, Generic, Exploits, Fuzzers, Reconnaissance, Analysis, and Worms. Table 2 shows the number of records in the training and test data sets. Obviously, the attack records in the training data set are imbalanced, and the attack records are less than the normal records, especially the Analysis, Backdoor, Shellcode and Worms records.

TABLE 2. Class distribution of the UNSW-NB15 dataset.

Category	Training set	Testing set
Normal	56,000	37,000
Generic	40,000	18,871
Exploits	33,393	11,132
Fuzzers	18,184	6062
DoS	12,264	4089
Reconnaissance	10,491	3496
Analysis	2000	677
Backdoor	1746	583
Shellcode	1133	378
Worms	130	44
Total	175,341	82,332

TABLE 3. Confusion matrix for network intrusion detection.

	Predicted attack	Predicted normal
Actual attack	TP	FN
Actual normal	FP	TN

B. PERFORMANCE METRICS

In order to effectively evaluate the performance of the network intrusion detection methods, nine performance measures are widely used, including accuracy, precision, recall, DR (detection rate), FPR (false positive rate), F1 score, G-mean, ROC (receiver operating characteristic curve), and AUC (area under the ROC curve). These performance measures are computed from the confusion matrix for network attack classification [43], as shown in Table 3. TP (True Positive) is the number of records in which the attack traffic is correctly classified; TN (True Negative) is the number of records in which the normal traffic is correctly classified; FP (False Positive) is the number of records in which normal traffic is misclassified as attack traffic; FN (False Negative) is the number of records in which attack traffic is misclassified as normal traffic.

The *Accuracy* is the proportion of the number of correctly predicted attacks and normal records to the total number of all records. If the *Accuracy* is higher, the performance of the classification model is better ($Accuracy \in [0, 1]$). Accuracy is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

The *DR* (detection rate) or *Recall* is the proportion of actual attack records that are correctly classified. *DR* is also known as *TPR* (true positive rate) or *Sensitivity*. If the *DR* is higher, the performance of the classification model is better ($DR \in [0, 1]$). *DR* is defined as follows:

$$DR = Recall = TPR = \frac{TP}{TP + FN} \tag{12}$$

The *TNR* (true negative rate) or *Specificity* measures the proportion of normal records that are correctly classified. If the *TNR* is higher, the performance of the classification model is better ($TNR \in [0, 1]$). *TNR* is defined as follows:

$$TNR = Specificity = \frac{TN}{TN + FP} \tag{13}$$

The *Precision* is the proportion of all predicted attack records that are actually attack records. If the *Precision* is higher, the performance of the classification model is better ($Precision \in [0, 1]$). *Precision* is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

The *FPR* (false positive rate) is the proportion of normal records that are incorrectly classified as attack records. If the *FPR* is lower, the performance of the classification model is better ($FPR \in [0, 1]$). *FPR* is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{15}$$

The $F1$ score is the harmonic mean of recall and precision. In imbalanced data sets, the $F1$ score is much more effective than accuracy at determining the performance of the classification model. If the $F1$ is higher, the performance of the classification model is better ($F1 \in [0, 1]$). $F1$ score is defined as follows:

$$\begin{aligned} F1 &= 2 \times \frac{Precision \times Recall}{Precision + Recall} \\ &= \frac{2 \times TP}{2 \times TP + FP + FN} \end{aligned} \tag{16}$$

The $G - mean$ is the geometric mean of specificity and sensitivity, which is used to measure the balance between majority and minority classification performance. If the $G - mean$ is higher, the performance of the classification model is better ($G - mean \in [0, 1]$). $G - mean$ is defined as follows:

$$\begin{aligned} \text{G-mean} &= \sqrt{TPR \times TNR} \\ &= \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \end{aligned} \tag{17}$$

The *ROC* (receiver operating characteristic curve) is a commonly used graph that illustrates the performance of a classifier over all possible thresholds. The main analysis tool is a curve drawn on a two-dimensional plane with the abscissa of the *FPR* and the ordinate of the *TPR* when you change the threshold for assigning observations to a given class.

The *AUC* (area under the ROC curve) is defined as the area under the ROC curve. If the *AUC* is higher, the performance of the classification model is better ($AUC \in [0, 1]$). The *AUC* of the real classifier is usually between 0.5 and 1. *AUC* is defined as follows:

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{TN + FP} \tag{18}$$

TABLE 4. Hyper-parameters configured for grid search.

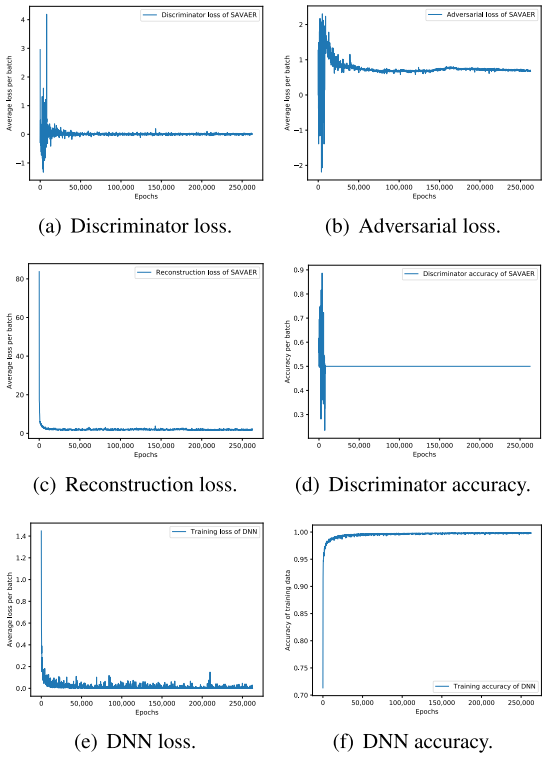
Hyper-parameter	Parameter values
Size of the input layer	n_x
Size of the output layer	n_y
Size of the latent variable	n_z
Sizes of the encoder	n_x, n_z × 8, n_z × 4, n_z × 2, n_z
Sizes of the decoder	n_z + n_y, n_z × 2, n_z × 4, n_z × 8, n_x
Sizes of the discriminator	40, 20, 10

C. EXPERIMENTAL SETUP

All the experiments are conducted in a TensorFlow environment with a 64-bit Windows 10 operating system (ThinkStation workstation with 64 GB RAM and Intel E5-2620 CPU). We use three different data sets, NSL-KDD (KDDTest+), NSL-KDD (KDDTest-21) and UNSW-NB15, to evaluate the performance of the proposed model and several well-known classifiers. Grid search and 5-fold cross-validation are used to find the optimal network structure and hyperparameters of the proposed model which results in the highest prediction accuracy. The activation function of all hidden layers of the proposed model is ReLU6 [44], except that the activation function of the decoder output layer is Sigmoid, the activation function of the output layer of the discriminator and encoder is linear, and the activation function of the DNN output layer is Softmax. The optimizer is Adam [45] with a learning rate of 0.001 for all networks. The grid search hyper-parameters are configured in Table 4.

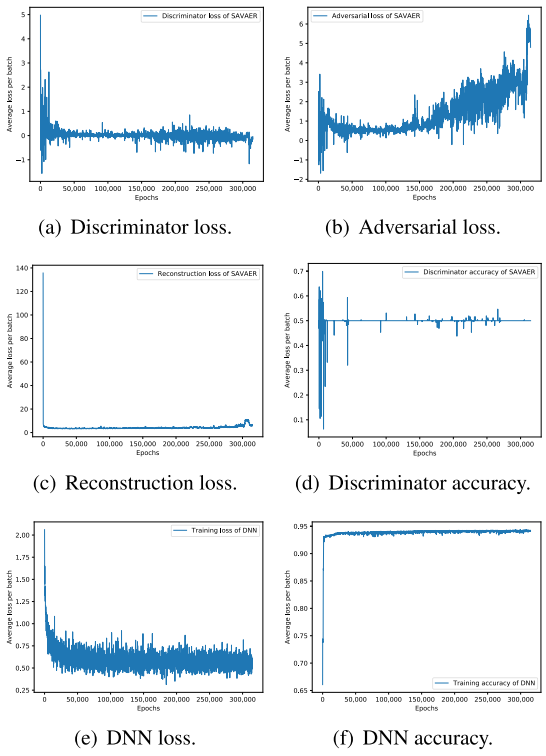
Through the grid search experiment, we obtained the optimal network structure. The optimal number of latent variable units on the NSL-KDD and UNSW-NB15 data sets are 9 and 23, respectively. The training graphs of SAVAER-DNN with the optimal network structure are shown in Figures 9 and 10. Figures 9(b) and 10(b) show that the generator loss of WGAN-GP tends to be minimal during network training, Figures 9(a) and 10(a) show that the discriminator loss of WGAN-GP tends to be minimal during network training, and Figures 9(d) and 10(d) show that the discrimination accuracy is close to 50%, indicating that WGAN-GP has reached Nash equilibrium. It can be seen from Figures 9(e) and 10(e) that DNN's convergence speed is very fast. The reason is that the trained encoder is used to initialize the weight of the DNN hidden layers, and the network initialization weight is close to the global optimum so that the network is easier to converge.

In order to demonstrate the superiority of the proposed SAVAER-DNN in detecting unknown attacks and imbalanced samples, we design several experiments. Tables 5 and 6 show the number of newly generated samples for each category on the NSL-KDD and UNSW-NB15 training datasets, respectively. Figures 11 and 12 show the sample spatial distribution



(a) Discriminator loss. (b) Adversarial loss.
(c) Reconstruction loss. (d) Discriminator accuracy.
(e) DNN loss. (f) DNN accuracy.

FIGURE 9. Training graphs of SAVAER-DNN with the optimal parameters on the NSL-KDD dataset.



(a) Discriminator loss. (b) Adversarial loss.
(c) Reconstruction loss. (d) Discriminator accuracy.
(e) DNN loss. (f) DNN accuracy.

FIGURE 10. Training graphs of SAVAER-DNN with the optimal parameters on the UNSW-NB15 dataset.

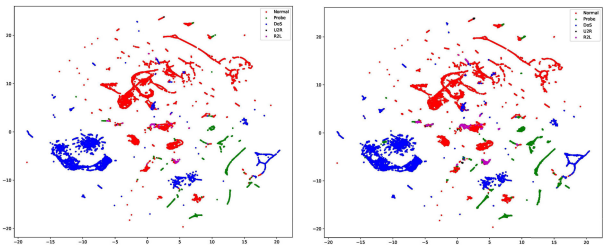
of the augmented and original training data projected onto a two-dimensional space by UMAP (Uniform Manifold Approximation and Projection). Figures 13 to 15 show the

TABLE 5. Number of training samples generated on the NSL-KDD dataset.

Category	Number of original records	Number of newly generated records	Total
Normal	13,449	0	13,449
Probe	2289	11160	13,449
DoS	9234	4215	13,449
U2R	11	13,438	13,449
R2L	209	13,240	13,449
Total	25,192	42,053	67,245

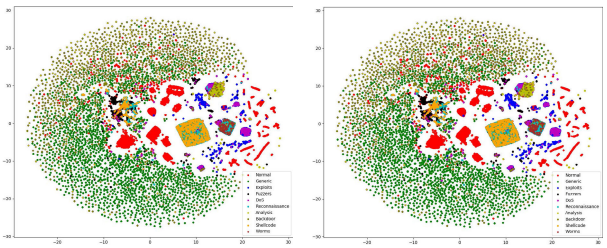
TABLE 6. Number of training samples generated on the UNSW-NB15 dataset.

Category	Number of original records	Number of newly generated records	Total
Normal	56,000	0	56,000
Generic	40,000	16,000	56,000
Exploits	33,393	22,607	56,000
Fuzzers	18,184	37,816	56,000
DoS	12,264	43,736	56,000
Reconnaissance	10,491	45,509	56,000
Analysis	2000	54,000	56,000
Backdoor	1746	54,254	56,000
Shellcode	1133	54,867	56,000
Worms	130	55,870	56,000
Total	175,341	81,340	560,000



(a) Original training data. (b) Synthetic training data.

FIGURE 11. UMAP visualization of the original and synthetic training data on the NSL-KDD dataset.



(a) Original training data. (b) Synthetic training data.

FIGURE 12. UMAP visualization of the original and synthetic training data on the UNSW-NB15 dataset.

ROC curves and AUC values of SAVAER-DNN. The comparison results of SAVAER-DNN and eight well-known classification models are shown in Figures 21 to 25. Besides, the detection performance of SAVAER-DNN is compared with other state-of-the-art models reported in the IDS literature, and the comparison results are shown in Tables 7 to 9.

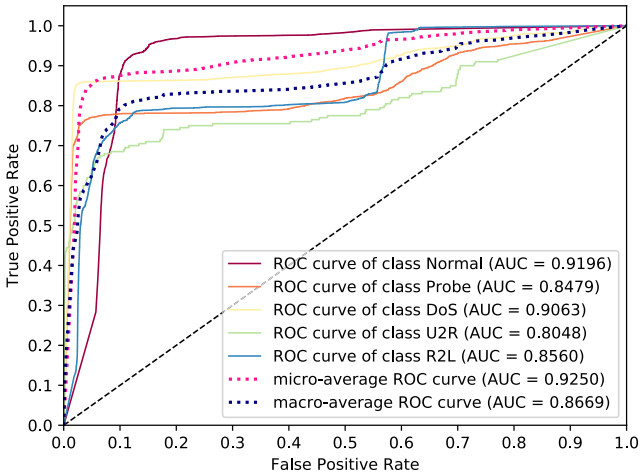


FIGURE 13. ROC curve and AUC on the NSL-KDD (KDDTest+) dataset.

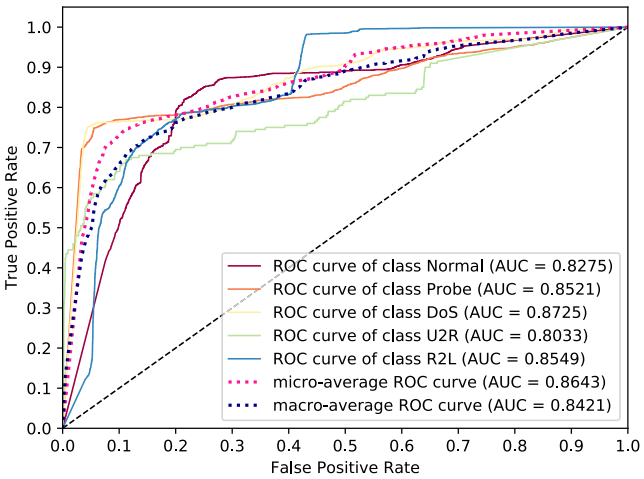


FIGURE 14. ROC curve and AUC on the NSL-KDD (KDDTest-21) dataset.

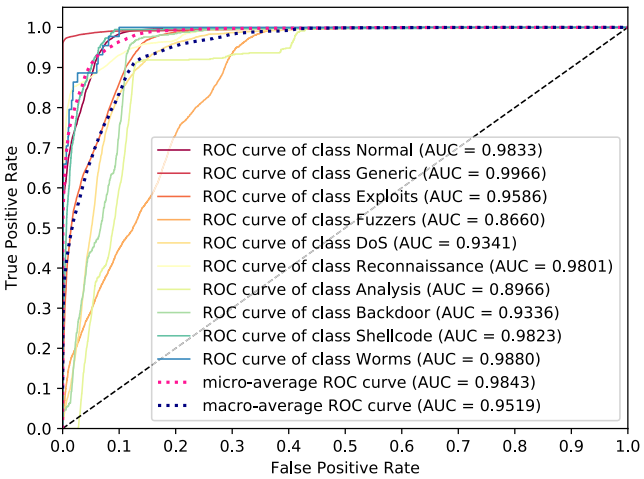


FIGURE 15. ROC curve and AUC on the UNSW-NB15 dataset.

D. RESULTS AND DISCUSSION

1) DETECTION PERFORMANCE

Tables 1 and 2 show that training data is severely imbalanced, and many unknown attacks in the test data set do not appear

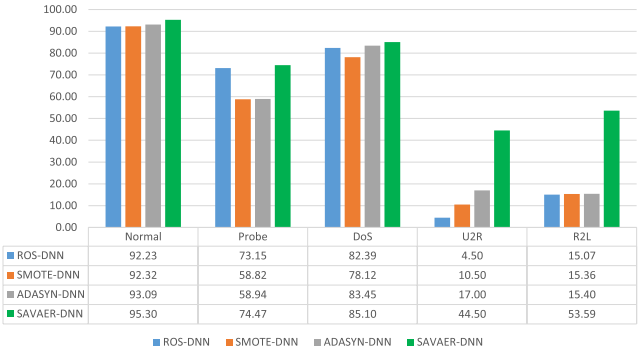


FIGURE 16. Comparison of detection rates of different data augmentation methods on the NSL-KDD (KDDTest+) dataset (%).

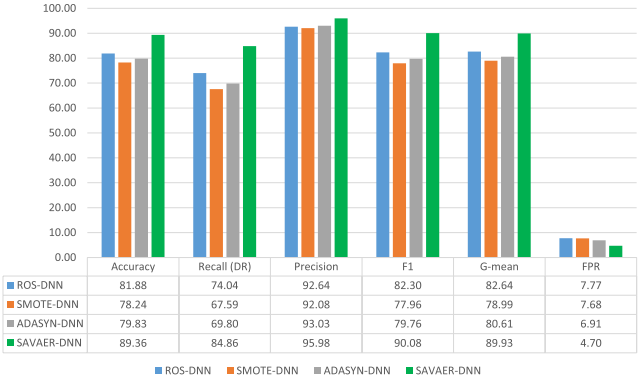


FIGURE 17. Comparison of the overall performance of different data augmentation methods on the NSL-KDD (KDDTest+) dataset (%).

in the training data set. For example, the training data of the NSL-KDD dataset contains 21 attacks, while the test data has 21 known attacks and 14 unknown attacks. Imbalanced data will make the classifier be biased toward the majority class, resulting in a high false positive rate for the minority attacks. In fact, minority attacks are more harmful, such as U2R and R2L attacks. The data generation algorithm is an important method to solve the problem of sample imbalance. The trained SAVAER decoder is used as a sample synthesizer for data augmentation. By specifying the value of the one-hot label y, we can generate corresponding attack samples from the conditional distribution $p_\theta(x|z, y)$, thereby increasing the training sample diversity and balancing the training data set. The number of newly generated samples is shown in Tables 5 and 6.

UMAP (Uniform Manifold Approximation and Projection) was proposed by McInnes *et al.* in 2018 [46]. It is a new manifold learning technology for dimensionality reduction and can also be used for visualization similar to t-SNE (t-Distributed Stochastic Neighbor Embedding) [47]. Compared with t-SNE, it can save as many local and global data structures as possible in shorter running time. UMAP is very suitable for embedding high-dimensional data in a low-dimensional space for visualization. In order to visually show that the generated data conforms to the spatial distribution of the original

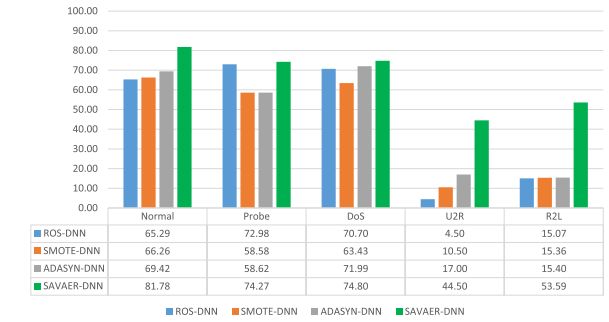


FIGURE 18. Comparison of detection rates of different data augmentation methods on the NSL-KDD (KDDTest-21) dataset (%).

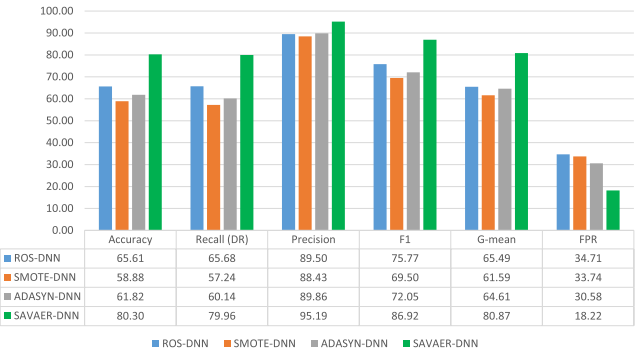


FIGURE 19. Comparison of the overall performance of different data augmentation methods on the NSL-KDD (KDDTest-21) dataset (%).

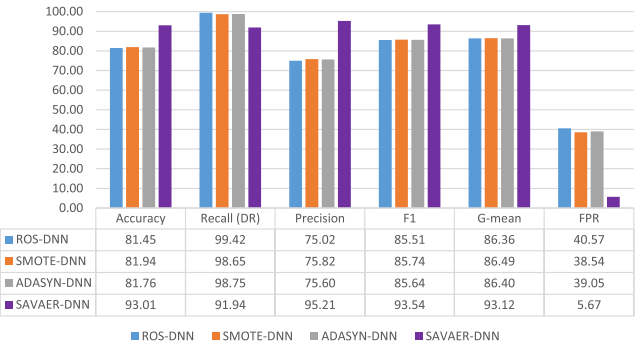


FIGURE 20. Comparison of the overall performance of different data augmentation methods on the UNSW-NB15 dataset (%).

training data, we use the UMAP method to project the original and synthetic training data into a 2D space, as shown in Figures 11 and 12. Figures 11(a) and 12(a) show that both training data sets are non-linearly separable. It can be seen from Figures 11(a) and 11(b) that the generated data on the NSL-KDD dataset is similar to the spatial distribution of the original training data, and Figures 12(a) and 12(b) also have similar results on the UNSW-NB15 dataset. It can be seen from the experimental results that the method of generating new training attack samples is reasonable.

In addition, in order to evaluate the detection performance of the proposed model, Figures 13 to 15 show ROC curves and AUC values. The ROC curve is plotted with TPR on the y-axis against the FPR on the x-axis. ROC curve is a performance measurement for classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. The higher the AUC, the better the model is in terms of detection performance. In a multi-class classification model, we use the one-vs.-rest (or one-vs.-all, OvA or OvR) method to plot the ROC curve and AUC value of each class. We note that all AUC values are greater than 0.8, which confirms that the proposed model produces better classification results.

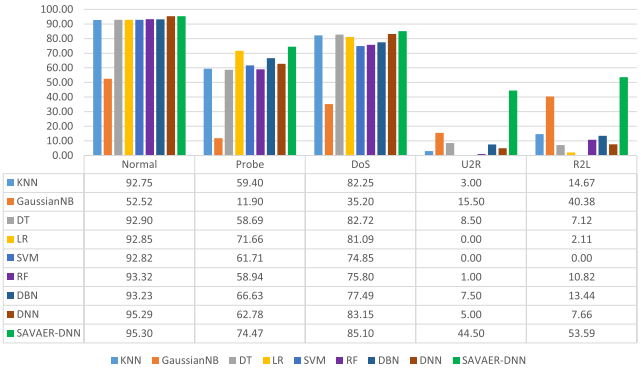
2) COMPARATIVE STUDY OF DATA AUGMENTATION

Currently, the most popular data augmentation methods include ROS (Random Over Sampler) [48], SMOTE (Synthetic Minority Oversampling Technique) [49], and ADASYN (Adaptive Synthetic) [50]. To generate unknown attack samples and balance the training data set, we propose the SAVAER-DNN model to improve the detection rate of unknown attacks. SAVAER-DNN uses a decoder to synthesize minority types of training attack samples. To compare the detection performance of the proposed SAVAER-DNN in data augmentation technology, we built three classification models based on three well-known data augmentation methods, named ROS-DNN, SMOTE-DNN, and ADASYN-DNN. The comparison results are shown in Figures 16 to 20.

Figures 16 and 18 show that compared with three well-known data augmentation methods, SAVAER-DNN has achieved the highest detection rate in five types of attacks, especially in U2R and R2L attacks. It can be seen from Table 1 that more than half of the U2R and R2L attacks in the test data set do not appear in the training data set. These comparative experiments fully prove that the proposed SAVAER-DNN is more effective in synthesizing unknown attacks. Besides, Figures 17 and 19 show that SAVAER-DNN also achieves the highest detection performance in terms of overall accuracy, recall, precision, F1 score, G-mean, and FPR.

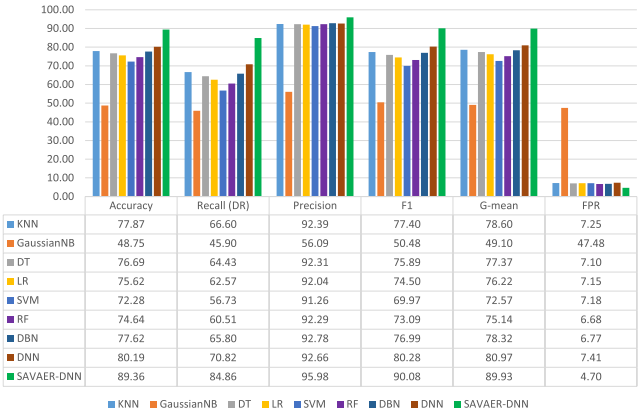
Figure 20 shows that the proposed SAVAER-DNN achieves the highest detection performance on the UNSW-NB15 dataset in terms of overall accuracy, precision, recall, F1 score, G-mean, and FPR. This further proves the superiority of SAVAER-DNN.

In summary, it can be seen from Figures 16 to 20 that the proposed SAVAER-DNN has better detection performance than ROS-DNN, SMOTE-DNN, and ADASYN-DNN. SAVAER-DNN learns the latent distribution of the original training samples, that is, explores the high-level feature representation of the original data. We randomly sample data points from the latent distribution, then concatenate the sampled data points with the class labels, and feed them into the decoder to generate new training samples. ROS-DNN repeatedly samples the original samples, while SMOTE-DNN and ADASYN-DNN randomly synthesize samples on the original samples based on the KNN (k-Nearest Neighbor) principle. The most significant difference between ROS-DNN, SMOTE-DNN, ADASYN-DNN



	Normal	Probe	DoS	U2R	R2L
KNN	92.75	59.40	82.25	3.00	14.67
GaussianNB	52.52	11.90	35.20	15.50	40.38
DT	92.90	58.69	82.72	8.50	7.12
LR	92.85	71.66	81.09	0.00	2.11
SVM	92.82	61.71	74.85	0.00	0.00
RF	93.32	58.94	75.80	1.00	10.82
DBN	93.23	66.63	77.49	7.50	13.44
DNN	95.29	62.78	83.15	5.00	7.66
SAVAER-DNN	95.30	74.47	85.10	44.50	53.59

FIGURE 21. Comparison of detection rates of different classification models on the NSL-KDD (KDDTest+) dataset (%).



	Accuracy	Recall (DR)	Precision	F1	G-mean	FPR
KNN	77.87	66.60	92.39	77.40	78.60	7.25
GaussianNB	48.75	45.90	56.09	50.48	49.10	47.48
DT	76.69	64.43	92.31	75.89	77.37	7.10
LR	75.62	62.57	92.04	74.50	76.22	7.15
SVM	72.28	56.73	91.26	69.97	72.57	7.18
RF	74.64	60.51	92.29	73.09	75.14	6.68
DBN	77.62	65.80	92.78	76.99	78.32	6.77
DNN	80.19	70.82	92.66	80.28	80.97	7.41
SAVAER-DNN	89.36	84.86	95.98	90.08	89.93	4.70

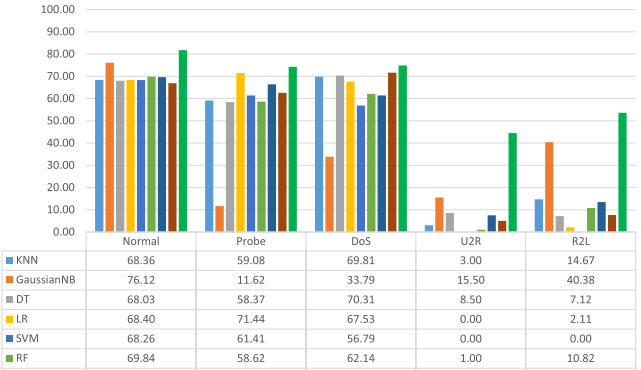
FIGURE 22. Comparison of the overall performance of different classification models on the NSL-KDD (KDDTest+) dataset (%).

and SAVAER-DNN is the ability to infer attack attributes from specific class labels. SAVAER-DNN can generate corresponding attack samples with specific class attributes. These experimental results show that samples synthesized based on latent space are better than samples synthesized based on original samples.

3) COMPARATIVE STUDY OF THE WELL-KNOWN CLASSIFIERS

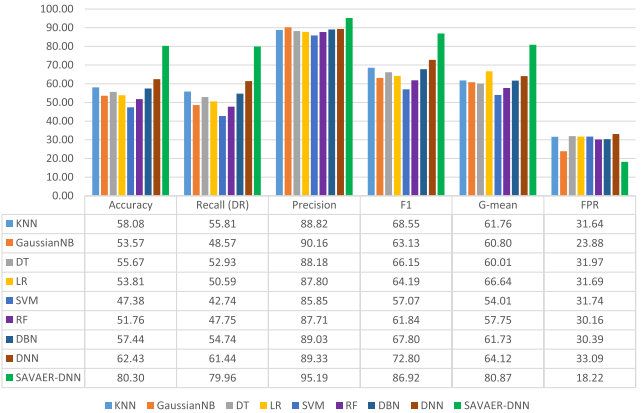
In order to evaluate the performance of the proposed SAVAER-DNN, we have built eight well-known classification models, named KNN (K-Nearest Neighbor), GaussianNB (Gaussian Naive Bayes), DT (Decision Tree), LR (Logistic Regression), SVM (Support Vector Machine), RF (Random Forest), DBN (Deep Belief Network), and DNN (Deep Neural Network), to detect intrusions and compare their results with our proposed model. These methods have been frequently used in the literature for intrusion detection. Comparative experimental results are shown in Figures 21 to 25.

As can be seen from Figure 21, SAVAER-DNN has a higher detection rate on the NSL-KDD (KDDTest+) dataset than all well-known classifiers. Figure 22 shows that SAVAER-DNN has the highest overall accuracy, recall, precision, F1 score, G-mean and FPR on NSL-KDD (KDDTest+) data set than all well-known classifiers. SAVAER-DNN has achieved a maximum accuracy of 89.36% with a minimum FPR of 4.70%.



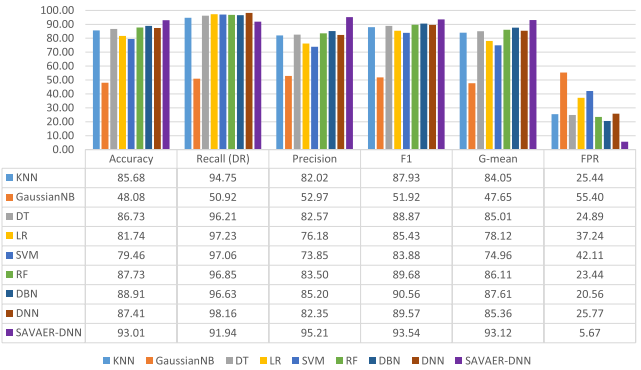
	Normal	Probe	DoS	U2R	R2L
KNN	68.36	59.08	69.81	3.00	14.67
GaussianNB	76.12	11.62	33.79	15.50	40.38
DT	68.03	58.37	70.31	8.50	7.12
LR	68.40	71.44	67.53	0.00	2.11
SVM	68.26	61.41	56.79	0.00	0.00
RF	69.84	58.62	62.14	1.00	10.82
DBN	69.61	66.36	61.38	7.50	13.44
DNN	66.91	62.57	71.60	5.00	7.66
SAVAER-DNN	81.78	74.27	74.80	44.50	53.59

FIGURE 23. Comparison of detection rates of different classification models on the NSL-KDD (KDDTest-21) dataset (%).



	Accuracy	Recall (DR)	Precision	F1	G-mean	FPR
KNN	58.08	55.81	88.82	68.55	61.76	31.64
GaussianNB	53.57	48.57	90.16	63.13	60.80	23.88
DT	55.67	52.93	88.18	66.15	60.01	31.97
LR	53.81	50.59	87.80	64.19	66.64	31.69
SVM	47.38	42.74	85.85	57.07	54.01	31.74
RF	51.76	47.75	87.71	61.84	57.75	30.16
DBN	57.44	54.74	89.03	67.80	61.73	30.39
DNN	62.43	61.44	89.33	72.80	64.12	33.09
SAVAER-DNN	80.30	79.96	95.19	86.92	80.87	18.22

FIGURE 24. Comparison of the overall performance of different classification models on the NSL-KDD (KDDTest-21) dataset (%).



	Accuracy	Recall (DR)	Precision	F1	G-mean	FPR
KNN	85.68	94.75	82.02	87.93	84.05	25.44
GaussianNB	48.08	50.92	52.97	51.92	47.65	55.40
DT	86.73	96.21	82.57	88.87	85.01	24.89
LR	81.74	97.23	76.18	85.43	78.12	37.24
SVM	79.46	97.06	73.85	83.88	74.96	42.11
RF	87.73	96.85	83.50	89.68	86.11	23.44
DBN	88.91	96.63	85.20	90.56	87.61	20.56
DNN	87.41	98.16	82.35	89.57	85.36	25.77
SAVAER-DNN	93.01	91.94	95.21	93.54	93.12	5.67

FIGURE 25. Comparison of the overall performance of different classification models on the UNSW-NB15 dataset (%).

Figure 23 shows that SAVAER-DNN achieves the highest detection rate for each class on the NSL-KDD (KDDTest-21) dataset. Figure 24 shows that compared with other classification models, SAVAER-DNN has achieved the highest performance in terms of overall accuracy, recall, precision, F1 score, and G-mean on the NSL-KDD (KDDTest-21) dataset. Moreover, Figures 21 and 23 show

that SAVAER-DNN achieves the highest detection rate on two small and important attack classes, such as U2R and R2L. Since more than half of the U2R and R2L attacks in the test data set do not appear in the training data set, most of the well-known classification models are unable to detect U2R and R2L attacks, except for GaussianNB, which is slightly better. These well-known classification models are effective at detecting known attacks, but they have many limitations in detecting unknown attacks. The proposed SAVAER-DNN improves the detection performance of known and unknown attacks by generating training samples of unknown attacks.

Figure 25 shows that SAVAER-DNN has the highest overall accuracy, precision, F1 score, G-mean, and FPR on the UNSW-NB15 dataset, but the overall recall is slightly lower. KNN, DT, LR, SVM, RF, DBN, and DNN have higher overall recalls than the proposed SAVAER-DNN (with 2.81%, 4.27%, 5.29%, 5.12%, 4.91%, 4.69% and 6.22% difference, respectively), but their other performances are even worse.

4) COMPARATIVE STUDY OF THE STATE-OF-THE-ART MODELS

In addition, in order to demonstrate the superiority of the proposed SAVAER-DNN, the performance of SAVAER-DNN is compared with other state-of-the-art intrusion detection models reported in the intrusion detection literature, including S-NDAE (stacked nonsymmetric deep auto-encoders) [7], SCDNN (spectral clustering and deep neural network) [19], ID-CVAE (a unsupervised network intrusion detection method based on a conditional variational auto-encoder) [20], RNN-IDS (recurrent neural network) [21], ResNet50 [22], GoogLeNe [22], $LSTM_4$ [24], GRU_3 [24], CFBLS (BLS with cascades of mapped features) [24], SHIA (scale-hybrid-IDS-AlertNet) [25], Gaussian-Bernoulli RBM [51], Random tree+NBTree [52], TSE-IDS (Two-Stage Classifier Ensemble for IDS) [53], EM Clustering [16], DT (decision tree) [16], TSDL (two-stage deep learning model) [23], CASCADE-ANN (a multiclass cascade of artificial neural network) [54] and AODE (average one dependence estimator algorithm) [55]. To be fair, only detection models using the same test dataset are selected. Tables 7 to 9 shows the experimental comparison results in terms of overall accuracy, DR (detection rate), F1 score, and FPR (false positive rate) on the NSL-KDD (KDDTest+), NSL-KDD (KDDTest-21), and UNSW-NB15 datasets.

It can be seen from Table 7 that the proposed SAVAER-DNN on the NSL-KDD (KDDTest +) data set reaches better classification performance in terms of overall accuracy, DR and F1 score than some of the existing state-of-the-art detection models, but the FPR is slightly worse. The RNN-IDS approach proposed in [21] obtains a better FPR than SAVAER-DNN (only with 1.26% difference), but it has worse overall performance in terms of accuracy, DR and F1 score than SAVAER-DNN.

Table 8 shows that SAVAER-DNN outperforms all other state-of-the-art models on the NSL-KDD (KDDTest-21) data set in terms of overall accuracy, DR, and F1 score, but its FPR is slightly higher than TSE-IDS proposed in [53] (with 0.22% difference). SAVAER-DNN has achieved the highest accuracy of 80.30%.

As can be seen from Table 9, compared with the other seven classification models, the proposed SAVAER-DNN has achieved the highest overall accuracy of 93.01%, DR of 91.94%, and F1 score of 93.54% on the UNSW-NB15 dataset, but its FPR is slightly higher than TSDL proposed in [23] by 4.92%. In summary, the above comparative experimental results fully prove that the proposed SAVAER-DNN is more effective in detecting network intrusions.

TABLE 7. Comparison results (%) of SAVAER-DNN with the state-of-the-art methods on the NSL-KDD (KDDTest+) dataset (N/A means no available results, * Ranked first, ** Ranked second).

Method	Accuracy	DR	F1	FPR
S-NDAE [7]	85.42	85.42	87.37**	14.58
SCDNN [19]	72.64	57.48	N/A	N/A
ID-CVAE [20]	80.10	80.10	79.08	8.18
RNN-IDS [21]	83.28	73.12	83.22	**3.44***
ResNet50 [22]	79.14	69.41	79.12	8.01
GoogLeNet [22]	77.04	65.64	76.50	7.89
$LSTM_4$ [24]	82.78	N/A	83.34	N/A
GRU_3 [24]	82.87	N/A	83.05	N/A
CFBLS [24]	82.20	N/A	82.23	N/A
SHIA [25]	78.50	78.50	76.50	N/A
Gaussian-Bernoulli RBM [51]	73.23	95.09**	75.30	43.22
Random tree+NBTree [52]	89.24**	83.90	N/A	N/A
TSE-IDS [53]	85.79	86.80	N/A	11.70
SAVAER-DNN	**89.36***	**95.98***	**90.08***	4.70**

TABLE 8. Comparison results (%) of SAVAER-DNN with the state-of-the-art methods on the NSL-KDD (KDDTest-21) dataset (N/A means no available results, * Ranked first, ** Ranked second).

Method	Accuracy	DR	F1	FPR
SCDNN [19]	44.55	37.85	N/A	N/A
RNN-IDS [21]	68.55	N/A	N/A	N/A
$LSTM_4$ [24]	66.74	N/A	76.21	N/A
GRU_3 [24]	65.42	N/A	74.06	N/A
CFBLS [24]	67.47	N/A	76.29**	N/A
Random tree+NBTree [52]	80.00**	78.80**	N/A	N/A
TSE-IDS [53]	72.52	72.50	N/A	**18.00***
SAVAER-DNN	**80.30***	**95.19***	**86.92***	18.22**

TABLE 9. Comparison results (%) of SAVAER-DNN with the state-of-the-art methods on the UNSW-NB15 dataset (N/A means no available results, * Ranked first, ** Ranked second).

Method	Accuracy	DR	F1	FPR
EM Clustering [16]	78.47	N/A	N/A	N/A
DT [16]	85.56	N/A	N/A	N/A
TSDL [23]	89.13	N/A	N/A	**0.75***
SHIA [25]	65.10	65.10	58.50	N/A
TSE-IDS [53]	91.27**	91.30**	N/A	8.90
CASCADE-ANN [54]	86.40	86.74	N/A	13.1
AODE [55]	83.47	N/A	N/A	8.90
SAVAER-DNN	**93.01***	**91.94***	**93.54***	5.67**

VI. CONCLUSION

By integrating the power of supervised VAE data generation and the advantages of WGAN-GP adversarial learning,

we propose a novel method for generating various unknown and low-frequent attack samples, called SAVAER. SAVAER's decoder is used to generate new attack samples of a specified label, thereby increasing the diversity of training samples and balancing the training data set. SAVAER's encoder is used to initialize the weights of the hidden layers of the DNN and automatically extract high-level feature representations of the original samples. We combine SAVAER with DNN to propose a hybrid framework for network intrusion detection, named SAVAER-DNN. We have evaluated the proposed SAVAER-DNN on the benchmark NSL-KDD (KDDTest+), NSL-KDD (KDDTest-21), and UNSW-NB15 datasets, and have obtained promising results. Comprehensive experimental results demonstrate that SAVAER-DNN can not only detect known and unknown attacks but also has a better detection rate on low frequent attacks. Besides, the comparative experimental results on the UNSW-NB15 dataset indicates that SAVAER-DNN is suitable for detecting complex network attacks. It has achieved the highest overall accuracy of 93.01%, the highest overall DR of 91.94% and the highest overall F1 score of 93.54% on the UNSW-NB15 dataset. Since the proposed SAVAER has highly competitive results compared with the state-of-the-art models, it may be a competitive candidate for network intrusion detection.

In future research, we plan to introduce other intelligent generation algorithms, such as TD-VAE (Temporal Difference VAE) [28] and VQ-VAE-2 (a two-level hierarchical Vector Quantised Variational Auto-Encoder) [56], to improve the proposed model to generate and detect attacks more efficiently.

REFERENCES

[1] *Top Dutch Banks, Revenue Service Hit by Cyber Attacks*. Accessed: Dec. 12, 2019. [Online]. Available: https://www.securityweek.com/top-dutch-banks-hit-cyber-attacks

[2] *Hackers Hit Norsk Hydro With Ransomware*. Accessed: Oct. 10, 2019. [Online]. Available: https://www.autonews.com/suppliers/norsk-hydro-rebuilds-after-cyberattack

[3] *Labour Party Confirms Cyber Attack Was DDoS*. Accessed: Dec. 18, 2019. [Online]. Available: https://www.zdnet.com/article/large-scale-cyberattack-hits-labour-party-systems

[4] F. Chen, Z. Ye, C. Wang, L. Yan, and R. Wang, "A feature selection approach for network intrusion detection based on tree-seed algorithm and K-nearest neighbor," in *Proc. IEEE 4th Int. Symp. Wireless Syst. Int. Conf. Intell. Data Acquisition Adv. Comput. Syst. (IDAACS-SWS)*, Sep. 2018, pp. 68–72.

[5] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[6] X. Qu, L. Yang, K. Guo, L. Ma, M. Sun, M. Ke, and M. Li, "A survey on the development of self-organizing maps for unsupervised intrusion detection," *Mobile Netw. Appl.*, vol. 10, pp. 1–22, 2019.

[7] N. Shone, T. Nguyen Ngoc, V. Dinh Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[8] S. Bartunov, A. Santoro, B. Richards, L. Marris, G. E. Hinton, and T. Lillicrap, "Assessing the scalability of biologically-motivated deep learning algorithms and architectures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9368–9378.

[9] M. Riviere, O. Teytaud, J. Rapin, Y. LeCun, and C. Couprie, "Inspirational adversarial image generation," 2019, *arXiv:1906.11661*. [Online]. Available: http://arxiv.org/abs/1906.11661

[10] H. V. Vo, F. Bach, M. Cho, K. Han, Y. LeCun, P. Perez, and J. Ponce, "Unsupervised image matching and object discovery as optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8287–8296.

[11] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[12] UNB. *NSL-KDD Dataset*. Accessed: Oct. 10, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[13] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6.

[14] ACCS. *UNSW-NB15 Dataset*. Accessed: Oct. 10, 2019. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[15] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[16] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., A Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Jan. 2016.

[17] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019.

[18] N. Moustafa, G. Creech, and J. Slay, "Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models," in *Data Analytics and Decision Support for Cybersecurity*. Cham, Switzerland: Springer, 2017, pp. 127–156.

[19] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, Oct. 2016.

[20] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT," *Sensors*, vol. 17, no. 9, p. 1967, Aug. 2017.

[21] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[22] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 858–866.

[23] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[24] Z. Li, A. L. G. Rios, G. Xu, and L. Trajkovic, "Machine learning techniques for classifying network anomalies and intrusions," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[25] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. ICLR*, 2017, vol. 2, no. 5, p. 6.

[28] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=S1x4ghC9tQ

[29] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114

[30] C. Doersch, "Tutorial on variational autoencoders," 2016, *arXiv:1606.05908*. [Online]. Available: http://arxiv.org/abs/1606.05908

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[32] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017, *arXiv:1701.04862*. [Online]. Available: http://arxiv.org/abs/1701.04862

[33] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*. [Online]. Available: http://arxiv.org/abs/1701.07875
[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.
[35] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*. [Online]. Available: http://arxiv.org/abs/1511.05644
[36] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 467–482.
[37] KDDCup. *KDD Cup Dataset*. Accessed: Oct. 10, 2019. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[38] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[39] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.
[40] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on Web servers in the presence of sampling," *Comput. Netw.*, vol. 121, pp. 25–36, Jul. 2017.
[41] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.
[42] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.
[43] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informat.*, vol. 10, no. 8, pp. 1–13, Aug. 2018.
[44] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," *Unpublished Manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[46] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*. [Online]. Available: http://arxiv.org/abs/1802.03426
[47] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[48] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 559–563, 2017.
[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[50] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 1322–1328.
[51] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted Boltzmann machine," *Big Data*, vol. 6, no. 2, pp. 159–169, Jun. 2018.
[52] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Comput. Appl.*, vol. 28, no. S1, pp. 1051–1058, 2017.
[53] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.
[54] M. M. Baig, M. M. Awais, and E.-S.-M. El-Alfy, "A multiclass cascade of artificial neural network for network intrusion detection," *J. Intell. Fuzzy Syst.*, vol. 32, no. 4, pp. 2875–2883, Mar. 2017.
[55] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Multi-classification of UNSW-NB15 dataset for network anomaly detection system," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 15, pp. 5094–5104, 2018.
[56] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," 2019, *arXiv:1906.00446*. [Online]. Available: http://arxiv.org/abs/1906.00446

YANQING YANG received the B.S. and M.S. degrees from Xinjiang University, Urumqi, China, in 2006 and 2009, respectively. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications. He was a Reviewer of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE ACCESS, *Mathematical Problems in Engineering*, and *Jordanian Journal of Computers and Information Technology*. His research interests include network attack detection, social network security, the IoT security, social engineering, and artificial intelligence security.

KANGFENG ZHENG received the M.S. degree from the School of Information Science and Engineering, Shandong University, in 2003, and the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. He is currently an Associate Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His current research interests include network security (network attack and defense, vulnerability mining and exploitation, malicious code analysis, web security, attack detection, and security evaluation), smartphone security, and big data analysis.

BIN WU received the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications. He is currently a Lecturer with the National Disaster Recovery Technology Engineering Laboratory, Beijing University of Posts and Telecommunications. His research interests include network security, intrusion detection, social engineering, and artificial intelligence security.

YIXIAN YANG received the Ph.D. degree from the Beijing University of Posts and Telecommunications, China, in 1988. He was a Changjiang Distinguished Professor of China, in 1993. He is currently a Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, where he is also the Director of the National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center. He has authored or coauthored over 300 journals and conference papers. His research interests include cryptography, information, and network security. He is also a Fellow of the China Institute of Communications and the Chinese Association for Cryptologic Research and a Council Member of the Chinese Institute of Electronics. He was selected in the National Science Fund for the Distinguished Young Scholars of China, in 1994. He is also the Editor-in-Chief of the *Journal on Communications*.

XIUJUAN WANG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. She is currently an Instructor Lecturer with the College of Computer Sciences, Beijing University of Technology. Her research interests include information and signal processing, network security, and network coding.