Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8268249, 18 pages
https://doi.org/10.1155/2021/8268249

Research Article

Adversarial Sample Detection with Gaussian Mixture Conditional Generative Adversarial Networks

Pengfei Zhang and Xiaoming Ju

School of Software Engineering, East China Normal University, Shanghai, China

Correspondence should be addressed to Xiaoming Ju; xmju@sei.ecnu.edu.cn

Received 15 May 2021; Revised 13 July 2021; Accepted 8 August 2021; Published 13 September 2021

Academic Editor: Jie Chen

Copyright © 2021 Pengfei Zhang and Xiaoming Ju. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

It is important to detect adversarial samples in the physical world that are far away from the training data distribution. Some adversarial samples can make a machine learning model generate a highly overconfident distribution in the testing stage. Thus, we proposed a mechanism for detecting adversarial samples based on semisupervised generative adversarial networks (GANs) with an encoder-decoder structure; this mechanism can be applied to any pretrained neural network without changing the network's structure. The semisupervised GANs also give us insight into the behavior of adversarial samples and their flow through the layers of a deep neural network. In the supervised scenario, the latent feature (or the discriminator's output score information) of the semi-supervised GAN and the target network's logit information are used as the input of logistic regression classifier to detect the adversarial samples. In the unsupervised scenario, first, we proposed a one-class classier based on the semisupervised Gaussian mixture conditional generative adversarial network (GM-CGAN) to fit the joint feature information of the normal data, and then, we used a discriminator network to detect normal data and adversarial samples. In both supervised scenarios and unsupervised scenarios, experimental results show that our method outperforms latest methods.

1. Introduction

Deep neural networks (DNNs) have achieved high accuracy in many classification tasks, such as speech recognition [1], objection detection [2], and image classification [3]. Although these DNNs are robust to random noise, they can mislead the model and cause it to output erroneous predictions when inputting small perturbations that are hard for humans to detect. In many machine learning applications (for example, in novelty detection [4], autonomous vehicles [5], and banking systems [6]), this prediction uncertainty will significantly reduce the model's safety.

Several methods have been proposed to protect against DNN attacks. One such method relies on the adversarial training method by adding adversarial samples in the training phase [7]. This method is robust to a variety of adversarial attacks but is ineffective against certain other attacks. To guarantee that there is no adversarial perturbation to fool the neural network within a given range, a more computationally demanding and provable defense is used, employing either integer programming approaches [8, 9] or satisfiability modulo theories [10]. The above-mentioned methods require a lot of calculations and special training procedures. However, when the parameters and structure of the neural network are fixed, neither of these methods can be used without modifying the neural network structure or retraining the neural network.

Adversarial sample detection is a good solution to the above problems. In the supervised scenario, most methods train a binary classifier to distinguish whether the sample is a normal sample or an adversarial sample. In 2018, Lee et al. [11] established a class-conditional Gaussian distribution in the intermediate layers of the pretrained network and distinguished adversarial samples using the Mahalanobis distance. Meanwhile, Ma et al. [12] proposed the local intrinsic dimensionality (LID) and experimentally proved that the LID can be employed to represent a test sample's characteristics. Both supervised learning methods use normal samples to train the feature extractor in the training stage. In the testing stage, the test samples are input to the feature extractor to obtain feature data. Finally, the feature data are input to the supervised classifier to realize the detection of normal samples and adversarial samples.

In the unsupervised scenario, alternatively, we can consider the unsupervised detection algorithm of adversarial samples. In 2017, Xu et al. [13] proposed a method to detect adversarial samples by comparing the model's prediction on a given image input with its prediction on the compressed image input version. In 2019, Yang et al. [14] proposed a feature attribute map of the adversarial samples close to the classification boundary; this map was different from the feature attribute map of the true data. In 2019, Roth et al. [15] introduced a statistical test based on the change in feature representations and log odds under noise; this approach is called odds-testing. PouyaSamangouei et al. [16] used GANs to model real images' distribution and find a close model's output to a test image, which did not contain adversarial perturbations; this confirmed that the adversarial sample has a data distribution far away from that of the normal sample, and it inspired other researchers to train a normal sample classifier to fit the distribution of real data. In 2018, an interesting analysis [16] showed how adversarial samples are propagated through neural network layer features; in 2019, Joshua et al. [17] studied the first-order classifier by training a discriminator with a generative adversarial network. These works [16, 17] inspired the present work. Regarding supervised learning and partial supervision methods, our method is inspired by the approach of Lee et al. [11]. They established a class-conditional Gaussian model through the Mahalanobis distance: they calculated the Mahalanobis distance between the true data and the adversarial sample and found that the feature distribution of the true data and that of the adversarial sample were different. Grosse et al., [18] also showed that adversarial samples have different distributions from normal data. Considering this finding, we also study the feature distribution information of normal samples through a semisupervised GAN in the present article. There are differences in the feature distributions between real samples and adversarial samples when the adversarial samples are input to the generator.

Our approach is described as follows. We used this difference to detect adversarial samples and true data. For unsupervised learning, we are inspired by the method of Engelsma and Jain [17]; we also employed the GM-CGAN to study the hidden layers' features and label information of the true data in the hidden layers of the pretrained network for joint feature distribution. When the sample belongs to true data, the discriminator will output a high predicted value. Conversely, when the sample is an adversarial sample, the joint feature of the hidden layers' feature and the label does not conform to the true data's feature distribution, and the discriminator will output a relatively low predicted value. We highlight that we do not directly study the joint features of true data and labels but instead use the intermediate layers' features of the pretrained network.

Figure 1 shows the framework of the proposed detection method. We train individual semisupervised generative adversarial networks to study normal data distributions in the pretrained target network's hidden layers. G_E is the semisupervised GAN's encoder structure, and G_D is the semisupervised GAN's decoder structure. In a supervised scenario, the latent feature of the semisupervised GAN and the target network's logit information are used as the input of the external classifier to detect the adversarial samples. latent_1 represents the latent vector of the first semisupervised GAN, and latent_i represents the last latent vector. In addition to using the latent information to do experiments, we also use the discriminator's output score information of the semi-supervised GAN for supervised training in our experiment. In the case of an unsupervised scenario, input features (i.e., the reconstruction error vector error_i in the final layer of the last block group in the pretrained target network, the latent vector latent_i in the final layer of the last block group in the pretrained target network, and the logit vector of the target network) are used as Gaussian mixture conditional generative adversarial networks to study the distribution of features. Besides, the label information of the target network is used as the GM-CGAN's conditional information.

Our method has universal applicability, and the samples are tested without modifying any of the model's structure. Experiments conducted on the DenseNet and ResNet network architectures show that among the recently proposed detection methods, our method obtains the highest detection rate in both supervised and unsupervised learning scenarios. Our method is better than the method utilizing the Mahalanobis distance [11] in a supervised scenario. In a partially supervised scenario, our method and the Mahalanobis distance method have similar performance. In an unsupervised scenario, our method performs similar to or better than the odds-testing [15] method.

2. Materials and Methods

2.1. Preliminaries. In this work, we describe the deep neural network first and then introduce the observation of adversarial samples in the neural network. Finally, we present the foundations of the GAN and CGAN.

2.1.1. Neural Network. The neural network solves the classification problem of class k, and the final output result is obtained by logit through the softmax function. The neural network consists of M layers h_m:

$$z_m = h_m(z_{m-1}), \quad \text{for } m = 1, \ldots, M. \tag{1}$$

In the above formula, M represents the total number of layers of the neural network, z represents the output of the neural network, and h is the hidden layer of the neural network.

2.1.2. Observation of Adversarial Samples in Neural Networks. In this article, we further verify that the adversarial sample and the normal sample are feature distributions in the target network's hidden layers. The distribution of the adversarial samples' feature data and the distribution

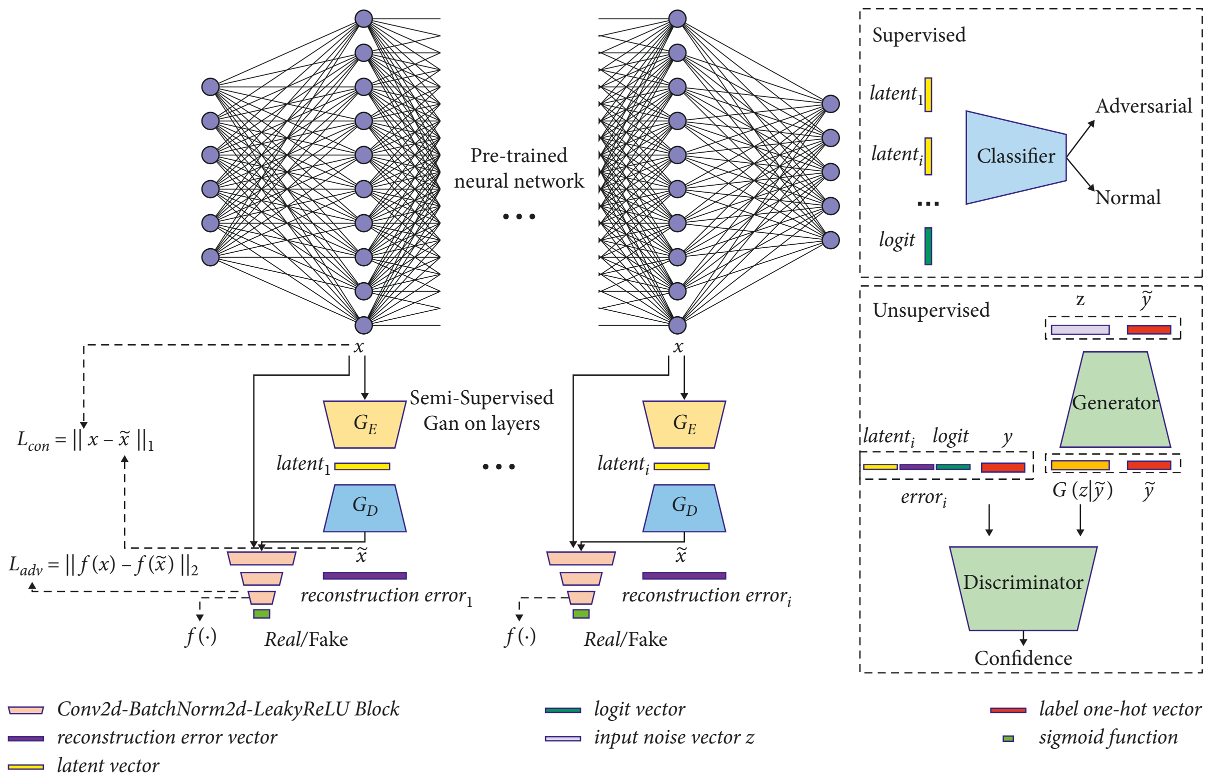

Figure 1: Framework of the proposed detection method.

of normal samples' feature data are different, and the influence of interference increases as the network deepens. Besides, adversarial samples will deviate from the real data.

The structure of the neural network contains nonlinear mappings and is formalized as follows:

$$F = h_1 \circ \cdots \circ h_M, \tag{2}$$

where F is the final output of the neural network and h is the hidden layer of the neural network.

Since in practice, the Lipschitz constant $\mathrm{lip}(h_l)$ of the neural network in each mapping is greater than 1 [19], we can assume that there is a small perturbation in the input space, and after the neural network propagates, the final layer has a vast representation distance. Our formula for the propagation of neural networks in high-dimensional space is formalized as follows:

$$\mathrm{lip}(F) \approx \mathrm{lip}(h_1) \ldots \mathrm{lip}(h_M), \tag{3}$$

where F is the final output of the neural network, h is the hidden layer of the neural network, and lip is the Lipschitz constant.

Due to the propagation properties of neural networks, as the number of neural network layers increases, the distribution of the adversarial samples will deviate farther from the distribution of normal samples. The difference between the real samples and the adversarial samples becomes more obvious in each sequent hidden layer.

2.1.3. GAN. The GAN [19] is relatively good for training generative models. It is composed of two adversarial modules: a generative model G used to describe the data distribution, and a probability discrimination model that determines whether the sample comes from the training data distribution instead of G. Both generator and discriminator are nonlinear mapping functions and have multilayer perceptrons.

To study the generator distribution p_g on the training data x, generator G builds a mapping function from the prior noise distribution $p_z(z)$ to the space of the generated data $(G(z;\theta_\varepsilon))$. $D(x;\theta_D)$ (the output data of discriminator D) is a probability scalar used to determine the probability y that the data comes from the distribution of the training data. Generator G and discriminator D are trained at the same time, and we adjust their parameters so that the generator has minimal loss:

$$\log(1 - D(G(z))). \tag{4}$$

The loss of discriminator D is

$$\log(D(x)). \tag{5}$$

Thus, the generator and discriminator essentially maximize and minimize $V(D, G)$.

$$\begin{aligned}\min_G \max_D V(D,G) = {} & E_{x\sim p_{\text{data}}(x)}[\log(D(x)] \\ & + E_{z\sim p_z(z)}[\log(1 - D(G(z)))].\end{aligned} \tag{6}$$

In equations (4) and (6), z is the input noisy data following the p_z distribution. Meanwhile, in equations (5) and (6), x is the input data following the p_{data} distribution. In equations (4)-(6), log is a logarithmic function. The G is the generator network and the D is the generator network. E is the expectation of the distribution function.$p_{\text{data}(x)}$ is the distribution of real samples. $p_z(z)$ is a low-dimensional noise distribution. V is the loss function.

2.1.4. CGAN. If both generator and discriminator are conditioned with some additional information, the GAN can be extended to its conditional form. Y can be any type of auxiliary information, such as a class label or data of other forms. We can adjust $[x]$ using a discriminator and generator with y input as an additional input layer. In this way, the maximum and minimum objective function is as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log(D(x|y)] + E_{z\sim p_z(z)}[\log(1-D(G(z|y)))]. \tag{7}$$

Here, z is the input noisy data following the p_z distribution, x is the input data following the p_{data} distribution, and y is the one-hot encoding of labels. Log is a logarithmic function. G is the generator network, and D is the generator network. E is the expectation of the distribution function. $p_{\text{data}(x)}$ is the distribution of real samples. $p_z(z)$ is a low-dimensional noise distribution. V is the loss function.

2.2. Related Work. We introduce adversarial attacks and adversarial defense in this work.

2.2.1. Adversarial Attacks. Adversarial attacks can be roughly divided into poisoning at training time or test time, and evasion. Adversarial attacks at training time are mainly conducted by adding maliciously tampered data into the training dataset during training so that the DNNs enter a suboptimal state, resulting in a decrease in model performance; this is called poisoning.

Meanwhile, evasion attacks involve tampering with the trained model's input, making the final prediction of the model incorrect. In both types of adversarial attack, an adversarial input modifies the other inputs in such a way that humans do not perceive the changes, but the DNNs make an incorrect final prediction.

For example, we add some minimal perturbations to pixels of the digit 2 in MNIST data so that the predicted value of the digit becomes 7, even though the digit still looks like the digit 2.

In this article, we study an evasion attack at test time. Given a test input x from class c, the adversarial attack aims to create the smallest perturbations so that the model's output will eventually become a specific class c' (targeted attack) or a class outside of class c (untargeted attack). This is formalized as an optimization problem, and its general form is as follows:

$$\min \|\delta\|_p \text{s.t.}, \tag{8}$$

$$\hat{C}(x+\delta) = c'\,(\text{targeted}), \tag{9}$$

$$\text{or}\,\hat{C}(x+\delta) \neq c\,(\text{untargeted}). \tag{10}$$

In equations (9) and (10), $\hat{C}$ is the trained model classifier. δ is the adversarial perturbation. x is the input data of the trained model. Based on the above general formula, many adversarial attack methods have been proposed; well-known methods include the fast gradient symbol algorithm (FGSM) [20], projected gradient descent (PGD) attack [22], Carlini–Wagner (CW) attack [23], DeepFool attack [24], and BIM attack [25]. These attack methods can be categorized into black-box attack or white-box attack methods depending on the extent of their knowledge about the DNNs classifier's parameters, structure, loss function, and algorithm. The most commonly used deep neural network attack methods are white-box methods because they assume a complete understanding of the system.

2.2.2. Defenses against Adversarial Attacks. Defending against neural networks is much more complicated than attacking them. Here, we summarize some current defense methods.

(1) Adversarial Training. It is a method that trains a better classifier to defend against adversarial samples. [25]. This is a method to add adversarial sample information in the training process of neural network classifier. For instance, one can add adversarial examples to the training data [26] for data augmentation [27] or add adversarial targets to the classification targets [28] for regularization [29]. Although the method is promising, it is difficult to determine which kind of attack is more suitable for the training way and how important the training way is; these problems are still unresolved [30].

(2) Defensive Distillation. The defensive distillation [30] training classifier makes it almost impossible for gradient-based attacks to directly generate adversarial samples on the trained network. This method uses the distillation training technique and hides the gradient between the logits and the output of the softmax function [26]. However, attacks can bypass this defense through the following three ways: (1) choosing an appropriate loss function, (2) directly calculating the gradient of the previous layer of the softmax layer instead of the gradient of the postlayer of the softmax layer, and (3) first attacking other vulnerable network models and then migrate to the trained distillation network.

(3) Detecting Adversarial Samples. Detecting adversarial samples can use statistical feature [29] methods or a classification network [30] to achieve defense. We build different detection classifiers for different attack methods to determine whether the input is a normal sample. The detector uses normal samples and adversarial samples for training. When the

training and testing adversarial samples are generated from the same way, and the adversarial perturbation is obvious enough, the detector shows good performance. However, this means of defense cannot be well generalized to different attack.

Our method is inspired by the approach of Lee et al. [11]. They established a class-conditional Gaussian model through the Mahalanobis distance: they calculated the Mahalanobis distance between true data and the adversarial sample and found that the feature distribution of the true data and that of the adversarial sample were different. Our method takes advantage of the difference between the adversarial samples and the normal samples in the middle layers of the neural network. In 2019, Joshua et al. [17] studied the first-order classifier by training a discriminator with a generative adversarial network. In our unsupervised method, we use the feature vectors extracted from the middle layer to train the discriminator network. Because our GM-CGAN is used to train a one-class classifier on the microfeature distribution of the middle layers of the pretrained network, we capture the weak perturbation of adversarial samples. Thus, our method will have better detection performance than the three types of methods described above.

2.3. Our Approach. In this work, we introduce the semisupervised GAN's structure and training method for studying the hidden layers' feature distribution of the pretrained network first. Then, we present our detection methods under supervised, partially supervised, and unsupervised scenarios.

2.3.1. Semisupervised GAN. Samet Akcay et al. [28] inspired us to use the semisupervised anomaly detection structure. They used an encoder-decoder-encoder structure to study the data distribution of the input image. For simplicity, we used an encoder-decoder structure in the generator part of the semisupervised GAN to analyze the feature distribution of the target network's hidden layers.

The formal principle behind the semisupervised GAN is as follows. Generator G first reads the intermediate layer feature x of the target network, where $x \in R^m$, and forward passes it to the encoder network G_E. With the use of convolutional layers followed by batch norm and ReLU() activation, G_E downscales x by compressing it to a latent vector, where latent $\in R^d$ (d represents the best dimension). In our experiment, d is set to 128. The decoder network G_D of generator G is composed of convolution transpose layers, the batch-norm function, and the ReLU() activation function. Finally, the latent vector is input to G_D to reconstruct the intermediate layer feature $\tilde{x}$ of the target network. Finally, generator G generates the intermediate layer feature $\tilde{x}$ via $\tilde{x} = G_D(\text{latent})$, where latent $= G_E(x)$.

(1) Adversarial Loss. Because the GAN is unstable during the training phase, we add feature matching loss to the training phase. In the ordinary GAN's training phase, generator G is updated based on discriminator D. Following the work of Salimans et al. [29], feature matching is employed to reduce instability during training. We update this approach based on the internal representation of discriminator D. First, we assume that there is an f function of the intermediate layer of discriminator D. For the input data x that satisfies the p_X distribution and outputs the intermediate layer features of discriminator D, the feature matching loss calculates the L_2 distance between the original feature and the generated feature. The form of our formalized adversarial loss L_{adv} is as follows:

$$L_{a\,dv} = E_{x\sim px}\left\| f(x) - E_{x\sim px} f(G(x))\right\|_2. \qquad (11)$$

In equation (11), x is the input data; f is the function of the intermediate layer of discriminator D, G is the generator network, D is the discriminator network, and p_x is the distribution of real samples. The adversarial loss L_{adv} is the L_2 loss. E is the expectation of the distribution function.

(2) Contextual Loss. While adversarial loss makes the generated adversarial samples deceive discriminator D, there is only one adversarial loss, and the generator cannot be optimized according to the input data's context information. Punishing the generator by measuring the distance between the input data and the generated data can remedy this problem. Isola et al. [30] showed that the fuzzy results due to L1 loss are less than those due to L2 loss. Therefore, we penalize G by measuring the L1 distance between the original input data and the generated data $\tilde{x} = G(x)$. Thus, the contextual loss L_{con} is formalized as follows:

$$L_{\text{con}} = E_{x\sim px}\|x - G(x)\|_1. \qquad (12)$$

In equation (12), x is the input data, E is the expectation of the distribution function, and p_x is the distribution of real samples. The contextual loss L_{con} is the L1 loss.

In this way, the generator encodes normal data but cannot encode adversarial samples because our generator G and discriminator D are optimized for normal data. The loss function we trained is as follows:

$$L = W_{\text{adv}}L_{\text{adv}} + W_{\text{con}}L_{\text{con}}, \qquad (13)$$

where W_{adv} is the weight coefficient of L_{adv} and W_{con} is the weight coefficient of L_{con}. W_{adv} and W_{con} are both positive integers; $W_{\text{adv}} = 1$, and $W_{\text{con}} = 15$. L_{adv} is the adversarial loss in equation (11). L_{con} is the contextual loss in equation (12). The semisupervised GAN training flow chart is described in Algorithm 1.

2.3.2. Supervised Scenario and Partially Supervised Scenario. We used the semisupervised GAN to study the normal data's feature distribution information in the hidden layers of the pretrained network. Then, we input the normal data and the adversarial samples into the pretrained network and obtain the corresponding latent features through the semisupervised GAN. Finally, the features are input into the supervised classifier logistic regression classifier to realize supervised classification and partially supervised classification. The supervised scenario and partially supervised scenario training flowchart is described in Algorithm 2.

2.3.3. Unsupervised Scenario. Due to the discrepancy between the feature distributions of the normal data and the adversarial samples in the hidden layers of the pretrained

```
**Input:** train sample $x$ into the pretrained target network.
**for** each layer $l \in 1, \ldots, L$ do.
    Train individual semisupervised generative adversarial networks in layer $l$.
    A semisupervised GAN$_l$ is obtained.
**end for**
**return** semisupervised GAN$_l$ $l \in 1, \ldots, L$
```

Algorithm 1: We train individual semisupervised generative adversarial networks to study normal data distributions in the pretrained target network's hidden layers.

```
    **Input:** the normal data $x$ and the adversarial samples $x_{\text{adv}}$ into the pretrained network.
    for each layer $l \in 1, \ldots, L$ do
        **Input:** the hidden layer $l$ into semisupervised GAN$_l$(Algorithm 1)
        latent$_l$ is obtained.
        error$_i$ is obtained.
    **end for**
The normal data $x$ latent feature is $\sum_l \text{latant}_{l_x}$ and the logit vector of the target network.
The adversarial samples $x_{\text{adv}}$ latent feature is $\sum_1 \text{latant}_{1_{x_{\text{adv}}}}$ and the logit vector of the target network.
We use the normal data $x$ latent feature and adversarial samples $x_{\text{adv}}$ latent feature input the support vector machine classifier.
return the evaluation data's AUROC.
```

Algorithm 2: In the supervised and partially supervised scenario, the final detection classifier is the support vector machine classifier. We used 10% of the test set as training data and 90% as evaluation data.

network, we use the hidden layers' difference features of the normal data and its label as the joint feature information for training the CGAN and a good discriminator.

The loss function of the target is as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{normal}}(x)}[\log(D(x \mid y))] + E_{z\sim p_z(z)}[\log(1 - D(G(z \mid y)))]. \tag{14}$$

Here, in order to reduce the amount of calculation, x is the joint feature data that includes the reconstruction error vector in the final layer of the last block group in the pretrained target network, the latent vector in the final layer of the last block group in the pretrained target network, and the logit vector of the target network. p_{normal} is the distribution of real samples, and p_z is the low-dimensional noise distribution; x follows a normal distribution p_{normal}, and z is the input noisy data following the p_z distribution, and y is the one-hot encoding of labels. Usually, $U[-1,1]^d$ or multivariate normal distribution information $N(0, I_{d\times d})$ are used as the noise input during the GAN's training. G is the generator network and D is the discriminator network. We emphasize that to better study the joint feature distribution of normal data and labels, we use the inherent multimodal distribution feature of p_x. Its specific form is as follows:

$$p_z(z) = \sum_{k=1}^{K} \alpha_k * p_k(z). \tag{15}$$

Here, K is defined as the number of Gaussian distributions in the mixture model, which is the number of neural network block groups in our experiment. $p_k(z)$ is defined as multivariate normal distribution information $N(\mu_k, \sum k)$, where $\forall k \in [K]$, $\alpha_k = 1/K$. z is the input noisy data. For data evaluation, we used the normal sample's one-hot encoding of labels and the adversarial sample's one-hot encoding of labels as the trained GM-CGAN's conditional information during the testing phase. The supervised scenario and partially supervised scenario training flowchart is described in Algorithm 3.

3. Results and Discussion

3.1. Experiments. We test our detection method against DeepFool, FGSM, BIM, PGD, and CW adversarial attacks on CIFAR10 [32], CIFAR100 [33], and SVHN [34] datasets. We used the ResNet-34 [35] and DenseNet-BC [36, 37] models. Similar to Lee's method, we chose to train the semisupervised GAN on the last layer of the basic block of the two neural network models; then, we extracted the hidden layers' features from the target network. For convenience of calculation, if the feature shape of the dataset in the network's hidden layers is the same, we used the same semisupervised GAN.

In the supervised and partially supervised scenario, the final detection classifier is the logistic regression classifier classifier; we used 10% of the test set as training data and 90% as evaluation data. In the unsupervised scenario, we used the GM-CGAN as the final detection classifier trained on the training samples that not include adversarial samples and noise samples. Our analyses of reconstruction error and L2 norm are presented in Figures 2–4. When we train the GM-CGAN, we select the input features as the reconstruction error vector in the final layer of the last block group in the pretrained target network, the latent vector in the final layer of the last block group in the pretrained target network, and

```
Input: the normal data $x$ into the pretrained network.
  for each layer $l \in 1, \ldots, L$ do
    Input: the hidden layer $l$ into semisupervised $GAN_l$(Algorithm 1)
    $latent_l$ is obtained.
    $error_i$ is obtained.
  end for
In order to reduce the amount of calculation, the GM-CGAN's input features are the reconstruction error vector in the final layer of the last block group in the pre-trained target network, the latent vector in the final layer of the last block group in the pre-trained target network, and the logit vector of the target network. The GM-CGAN's label information is the label of the output of this neural network model.
return the evaluation data's AUROC.
```

Algorithm 3: In the unsupervised scenario, we used 10% of the test set as training data and 90% as evaluation data.

the logit vector of the target network. The GM-CGAN's conditional information is the label of the output of this neural network model.

The features we input are important for detecting adversarial samples. In the study by Yang et al. [19], the reconstruction error vector of the L1 norm was shown to reflect the discrepancy between the given sample and the real sample. The latent feature vector's norm reflects whether a given sample can be generated on the data manifold. In Figure 2, we can also view the importance of the logit norm. This can reduce the computational complexity and allows us to better capture the difference information. Similar to Lee's [11] method, we use logistic regression classifier. The hyperparameters of the SVM classifier are fine-tuned.

First, similar to how Yang et al. [14] approached the problem, we analyzed the norm and reconstruction error information of the hidden layer's latent feature vectors generated from the semisupervised GAN (Figures 2 and 3). We also analyzed the joint feature information of the norm and reconstruction error (Figure 4). Finally, we assessed the importance of different hidden layers' features (Figure 5). Here, due to limited space of article, we only present figures for the ResNet model under attack by five methods, on the CIFAR10 dataset. This analysis puts forward a strong argument that the reconstruction error and the latent norm can fully explain the data manifold, which can help us to detect adversarial samples.

For the sake of fairness, our method and the method of Lee et al. [11] are initialized with the same settings. In the experimental test stage of Lee et al., noisy data and adversarial samples are generated for normal test data. Our semisupervised GAN obtains latent features, and then, we used these latent features to train the supervised and unsupervised classifiers. We used 10% of the test set as training data and 90% as evaluation data and performed five-fold cross-validation.

3.2. Result. In Figure 2, we visualized the distribution information of normal (green) and adversarial (red) samples through kernel density estimation in 2D space. In the subfigure, x-axis represents the L2 norm in the latent vector produced by the generator and y-axis represents probability density. These figures are generated for the CIFAR10 dataset, with the ResNet model. The adversarial samples flow through 5 semisupervised GANs and the target network's logit layer, and hence, there are 6 rows of subfigures. The difference between the real sample and the adversarial sample is obvious in the last few layers of the neural network, especially in the logit layer.

In Figure 3, we visualized the distribution information of normal (green) and adversarial (red) samples through kernel density estimation in 2D space. In the subfigure, x-axis represents reconstruction error and y-axis represents probability density. These figures are generated for the CIFAR10 dataset, with the ResNet model. The adversarial samples flow through 5 semisupervised GANs, and hence, there are 6 rows of subfigures.

In Figure 4, we visualized the distribution information of normal (green) and adversarial (red) samples through kernel density estimation in 2D space. In the subfigure, x-axis represents the L2 norm in the latent vector produced by the generator and y-axis represents reconstruction error. These figures are generated for the CIFAR10 dataset, with the ResNet model. The adversarial samples flow through 5 semisupervised GANs, and hence, there are 6 rows of subfigures.

Figure 5 presents the AUROC of the threshold-based detector using the latent vector generated from the semisupervised GAN's generator at different basic blocks of ResNet trained on CIFAR10 dataset and the logit vector of the target network. We measured the detection performance using adversarial samples produced by FGSM, BIM, DeepFool, CWL2, and PGD. We also measured the detection performance using the entire latent vectors generated from the semisupervised GAN's generator (see the last subfigure, i.e., Feature Ensemble).

To generate the results for the supervised scenarios (Table 1), the final detection classifier is the support vector machine classifier whose input includes all latent vectors. For the partially supervised scenarios (Table 2), the final detection classifier is the support vector machine classifier whose input only includes FGSM samples.

Meanwhile, for the unsupervised scenarios (Table 3), we used the GM-CGAN as a one-class classifier whose input features are the reconstruction error vector, the latent vector in the final layer of the last block group (obtained by the semisupervised GAN), and the logit of the target network. The GM-CGAN's conditional information is the label of the target network. We train the classifier based on the training data, which does not include any adversarial samples and noisy samples.

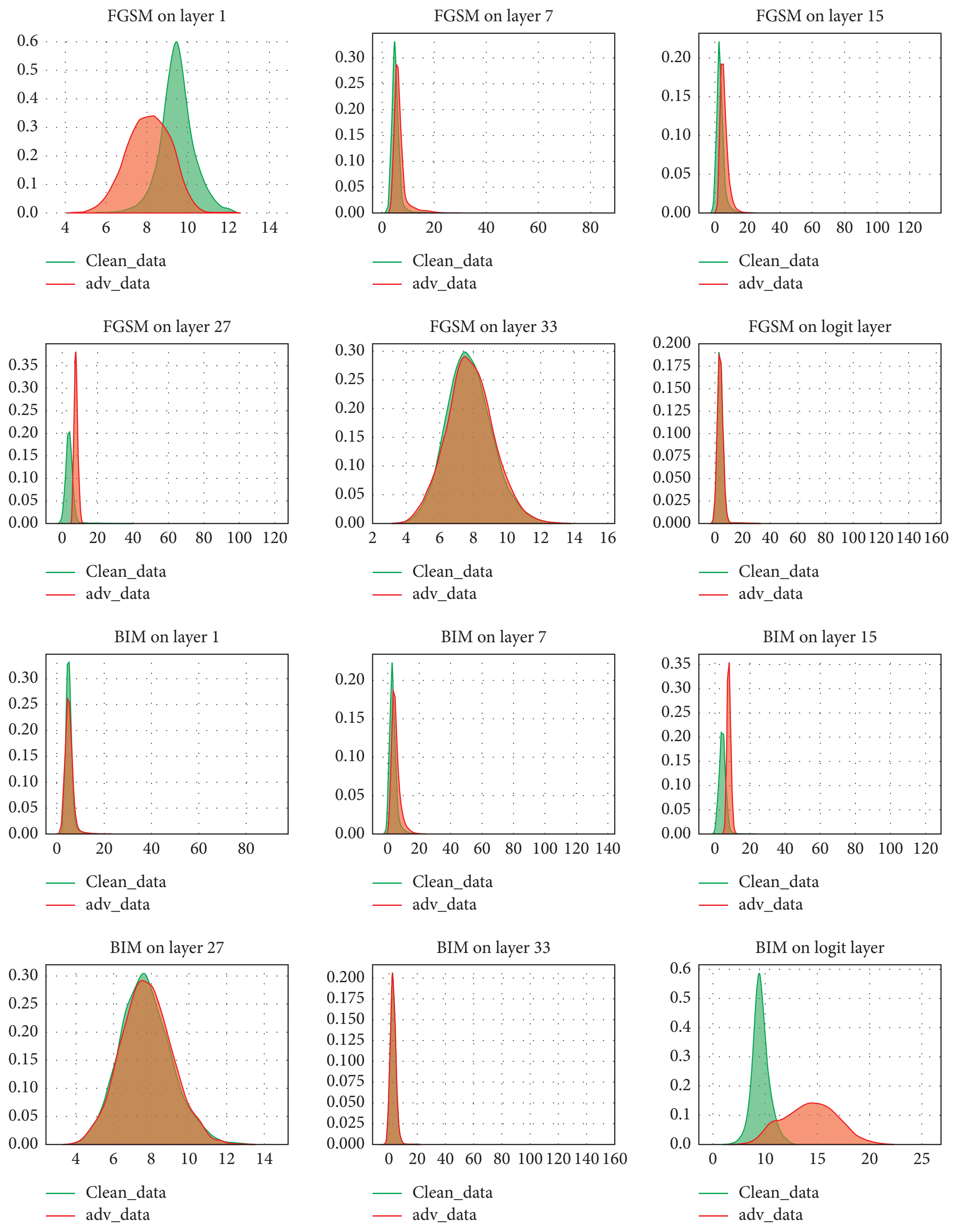

Figure 2: Continued.

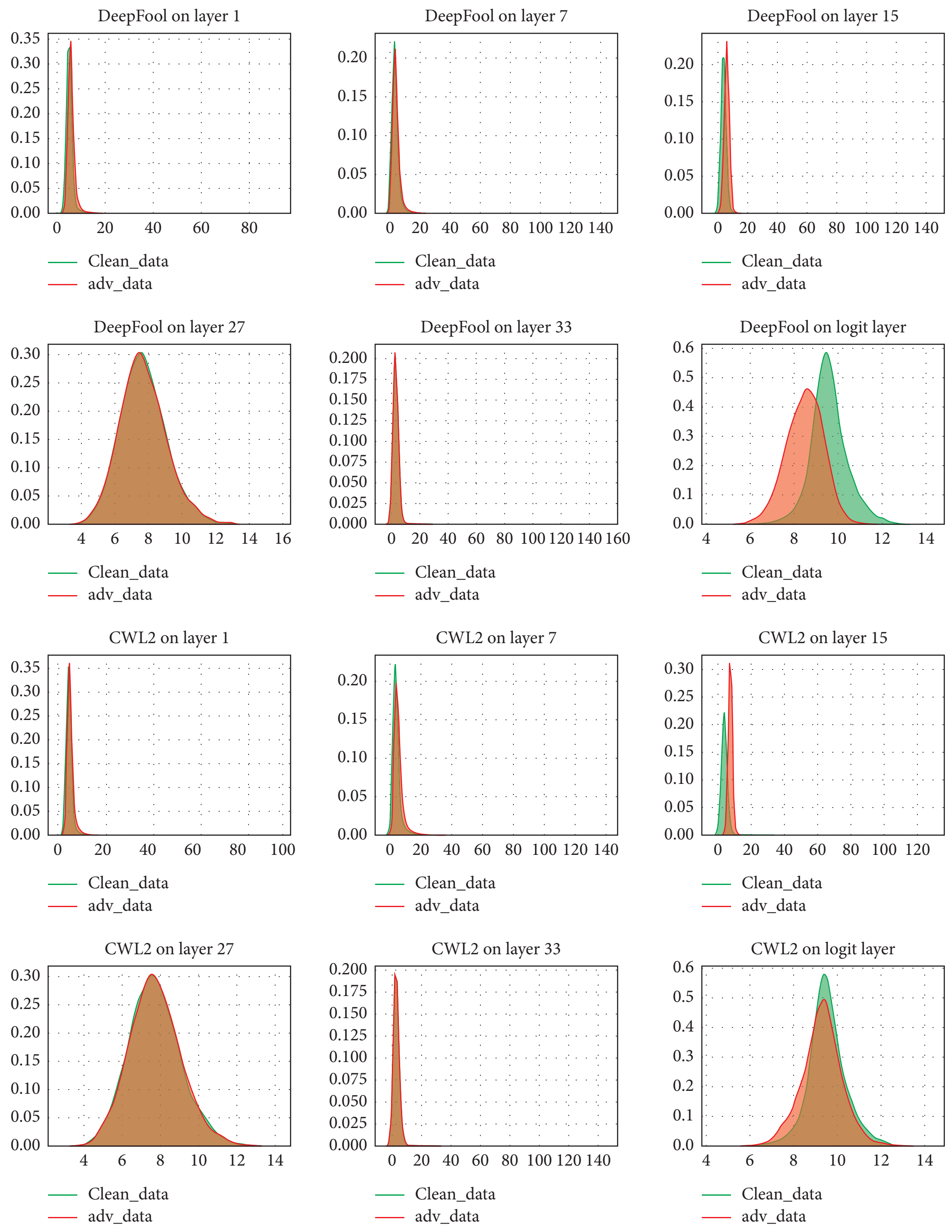

Figure 2: Continued.

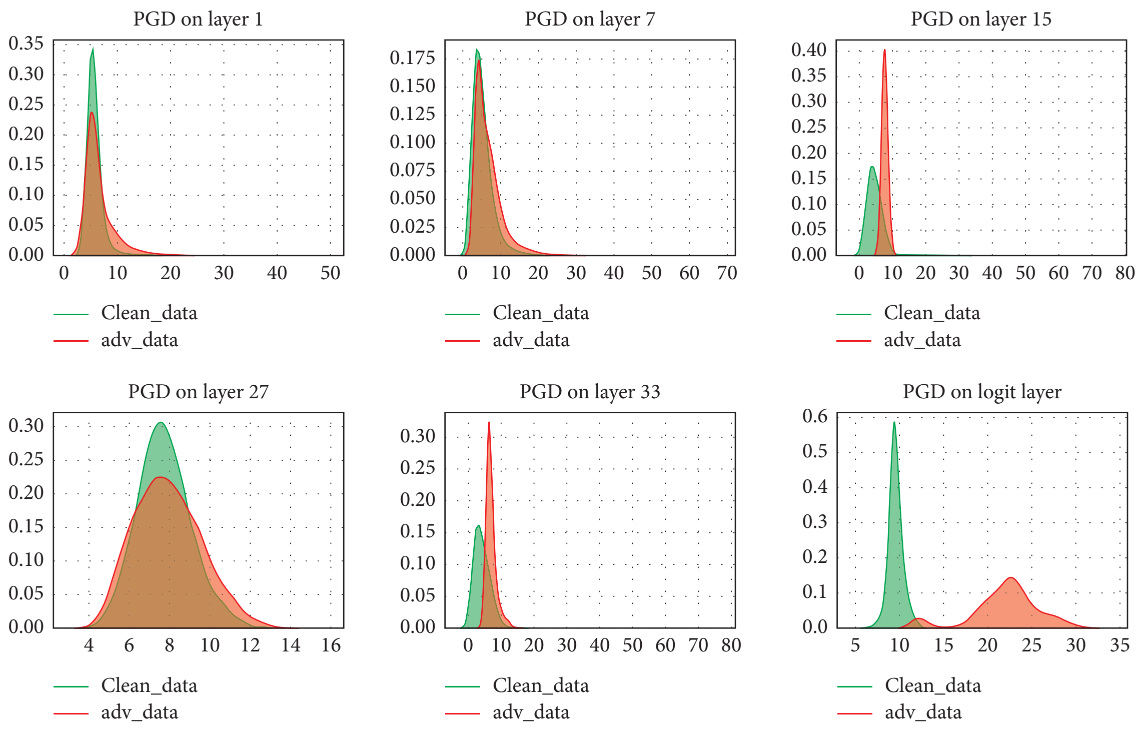

Figure 2: Analysis of feature norms of different network layers.

3.3. Discussion. In our experiment, we utilized the PGD-100 attack. In the supervised scenario, our method performed significantly better than that of Lee et al. Our method reduced the preprocessing time for input data and reduced the total amount of calculations. We note that the Mahalanobis distance covariance matrix must be full rank and that it cannot handle problems on nonlinear manifolds. In the deep layers of the neural network, the Mahalanobis distance cannot provide a reliable measure of the distance between the data, which mainly exist in a nonlinear form; ultimately, the Mahalanobis distance is unstable in such situations. The source of this instability is the covariance matrix. In Lee's method, the distance between data of different network blocks is very small compared with the distance between high-dimensional adversarial data and the real data. Furthermore, Lee's method cannot reflect the difference between the adversarial data and the real data. In contrast, our semisupervised GAN method maps different network block feature data to low-dimensional space, thereby obtaining more sample feature information. Moreover, it can overcome the drawback of the Mahalanobis distance being unsuitable for the determination of nonlinear data. Considering these key characteristics, it is clear why our method achieved the best results in the supervised classification task. We highlight that our method improved the detection of DeepFool to above 94.68%.

For the partially supervised scenario, we used the logistic regression classifier as the final classifier with FGSM samples. Although it did not achieve the same effect as Lee's method in this scenario, our method still achieved good results. Here, we take the ResNet model and the CIFAR10 dataset as an example (Table 2). For BIM attacks, the detection AUROC dropped from 98.91% to 73.19%; meanwhile, for FGSM attack detection, the AUROC was 99.98%. We argue that there might be a trade-off between performance on a fully supervised scenario (where our method had an AUROC close to 100% in some cases) and an ability to generalize to other attacks. In Figure 4, we find that BIM attacks and FGSM attacks deviate from the real data and have great inconsistencies; this is mainly manifested in the 1st, 7th, and 27th layers of the network. By the same token, the ways in which different attacks deviate from the true data are also different. Therefore, only partial supervision is suitable for FGSM attacks.

For the unsupervised scenario, our method performed better than the odds-testing method [15], except for with the PGD-100 attack. We argue that not all attack differential features are present in the last layer. The PGD attack consists of initializing the search for an adversarial

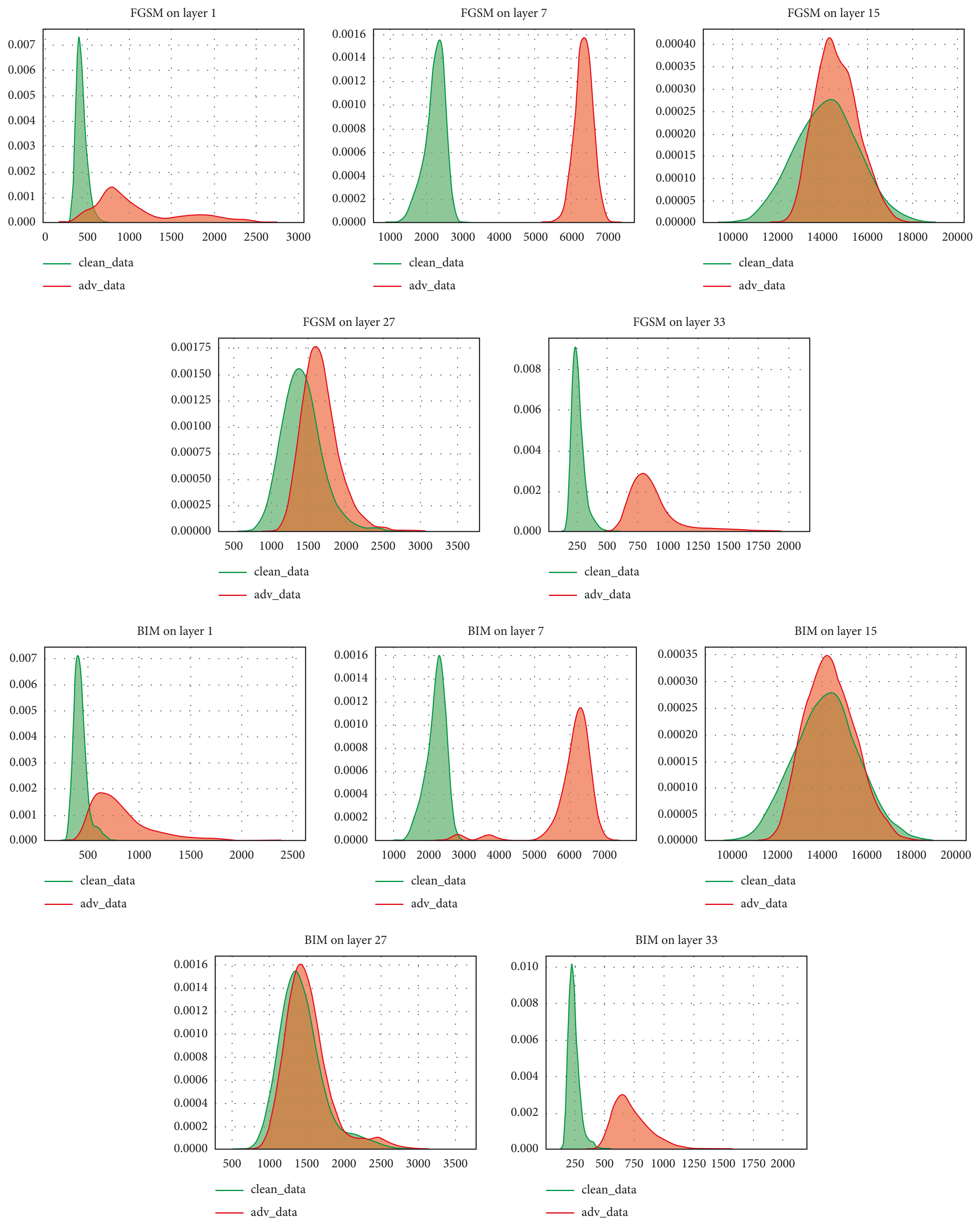

Figure 3: Continued.

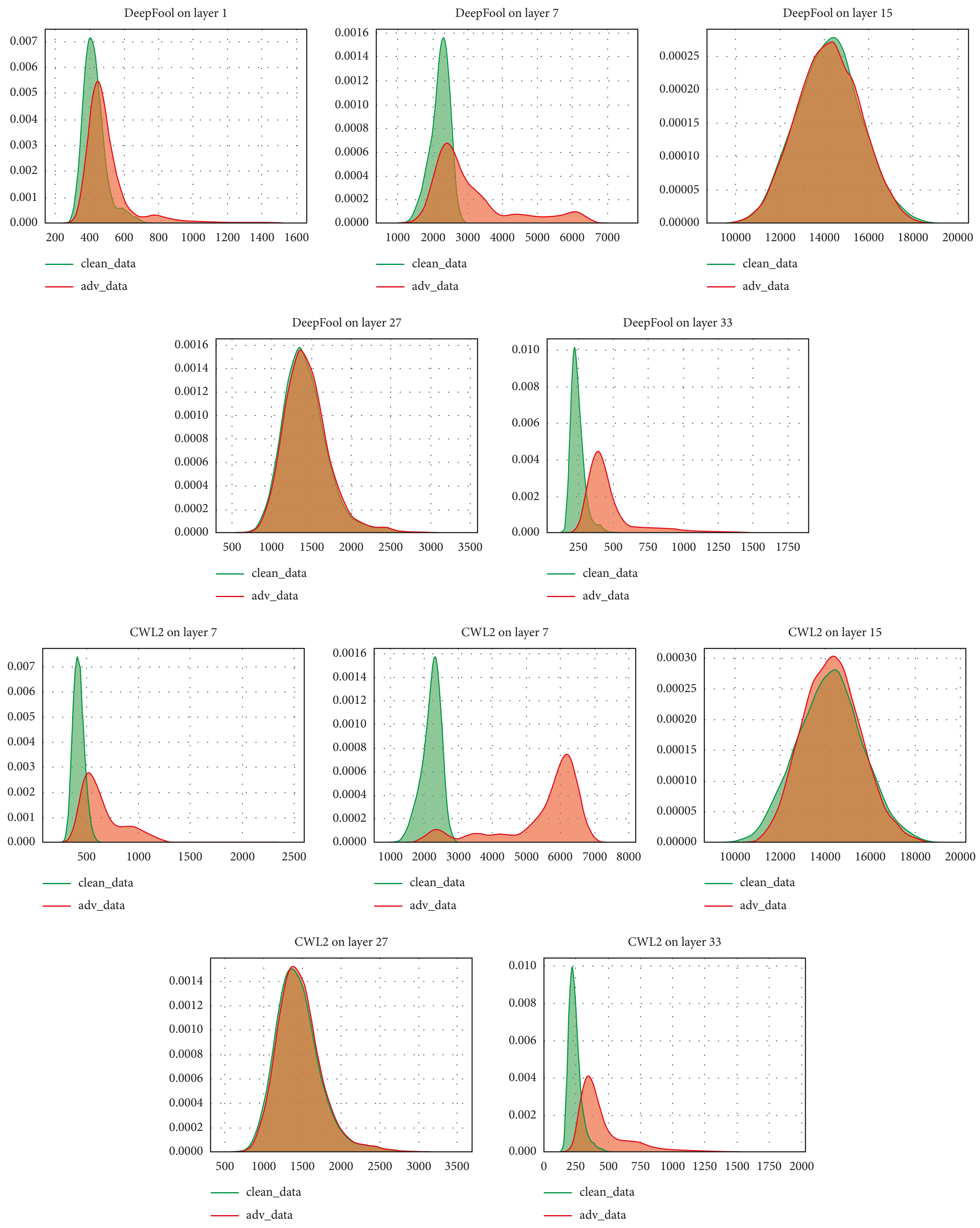

FIGURE 3: Continued.

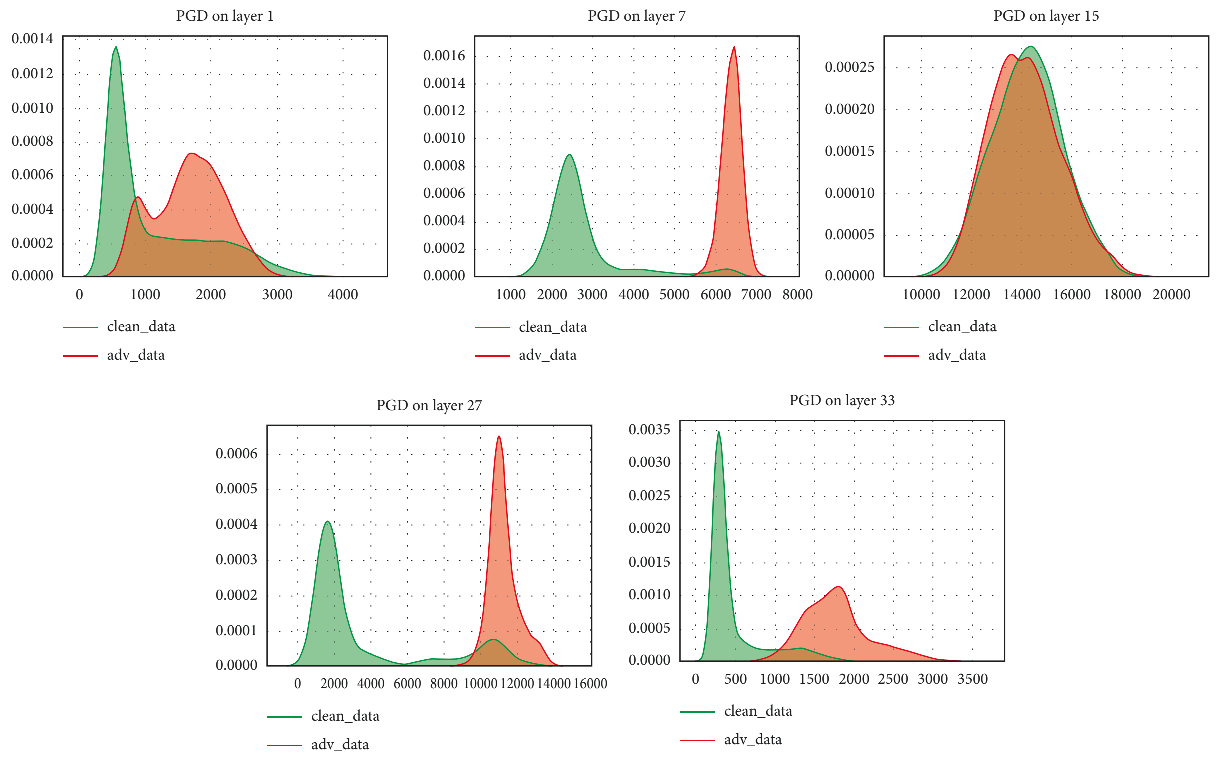

Figure 3: Analysis of reconstruction errors in different network layers.

example at a random point within the allowed norm ball, then running several iterations of the basic iterative method. The PGD-100 has stronger attack performance because of the basic iterative method. We think the performance feature of PGD's adversarial samples in the network middle layer is more similar to those of real samples. The input at the end of the network is different from the real one. The PGD-100 can attack deep neural networks, such as ResNet-34 and DenseNet, which contain many features. GM-CGAN contains fewer features and is easier to be attacked by PGD-100. Because GM-CGAN is also a classifier based on neural network, the features in the hidden layers space will also be attacked by PGD. It reduces the defensive performance of GM-CGAN. For example, as shown in Figures 3 and 4, the most obvious difference in the PGD-100 attack is in the 15th layer. Although our method requires multiple forward propagations of neural networks like the odds-testing method, we provide a new idea for detecting adversarial samples: a new one-class classifier. We emphasize that our one-class classifier does not require any noisy data during the training process compared with the odds-testing method, and the training method is simple and easy to operate. Additionally, the number of forward propagations of our method is relatively small. Through Figures 2–4, we find that as the network layer deepens, the data distribution of the adversarial sample deviates farther from the characteristic distribution of the normal data. The L2 norm information of the logit can better reflect this difference than the latent vector's norm information. Compared with the L2 norm information, the reconstruction error can reveal the distribution discrepancy between the adversarial sample and the normal sample earlier, and the deeper the network layer, the greater the discrepancy. The combined information of the reconstruction error and L2 norm can also reflect this trend. In general, the discrepancy between the distribution of real samples and adversarial samples is more obvious in the last few layers. In Figure 5, we analyzed the detector's performance with different basic blocks (which have different latent vector characteristics). We also analyzed the performance of the detector after the integration of different basic block features. Like in Figures 2–4, in most instances, the discrepancy is most obvious in the last layer. At the same time, we found that

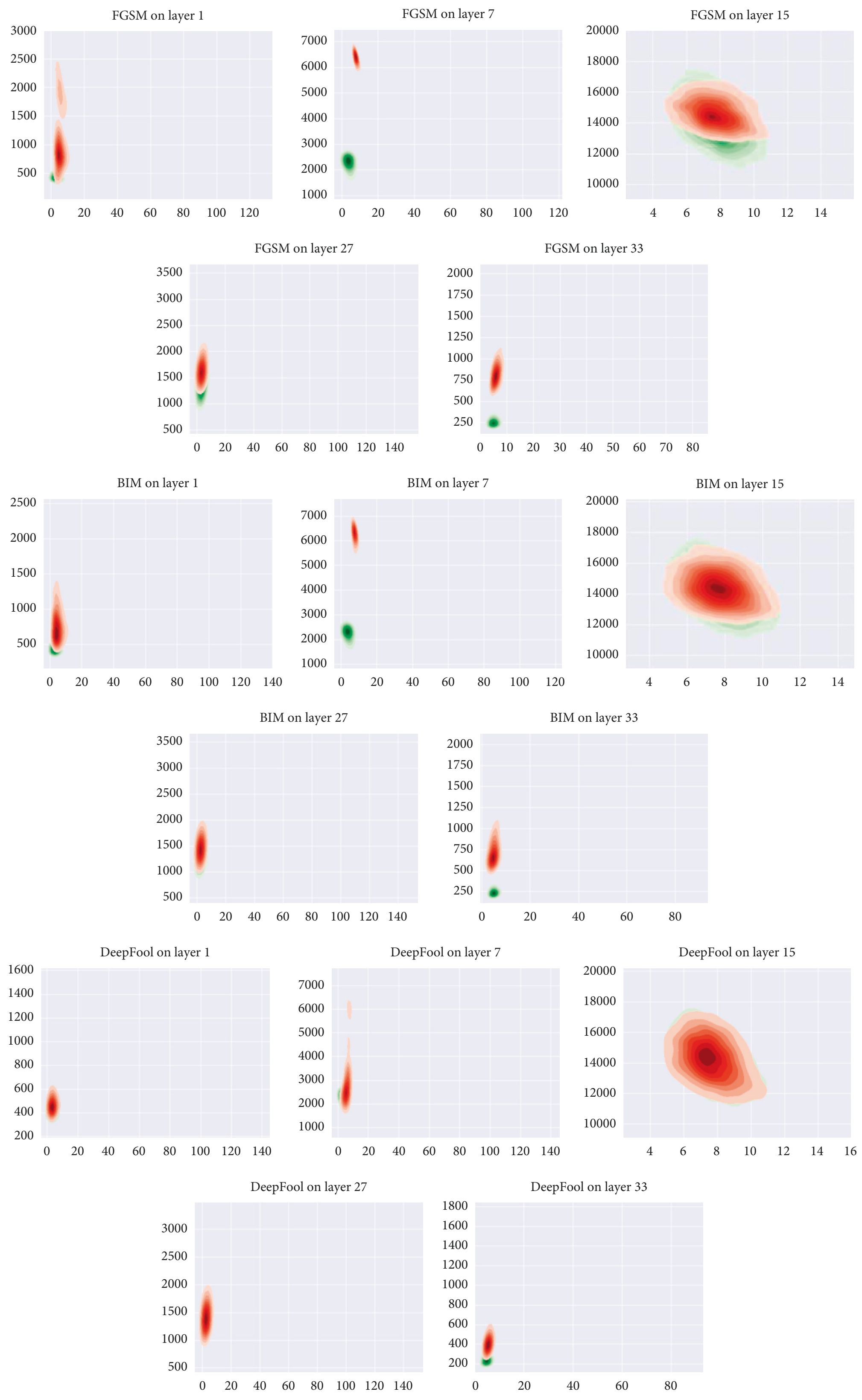

FIGURE 4: Continued.

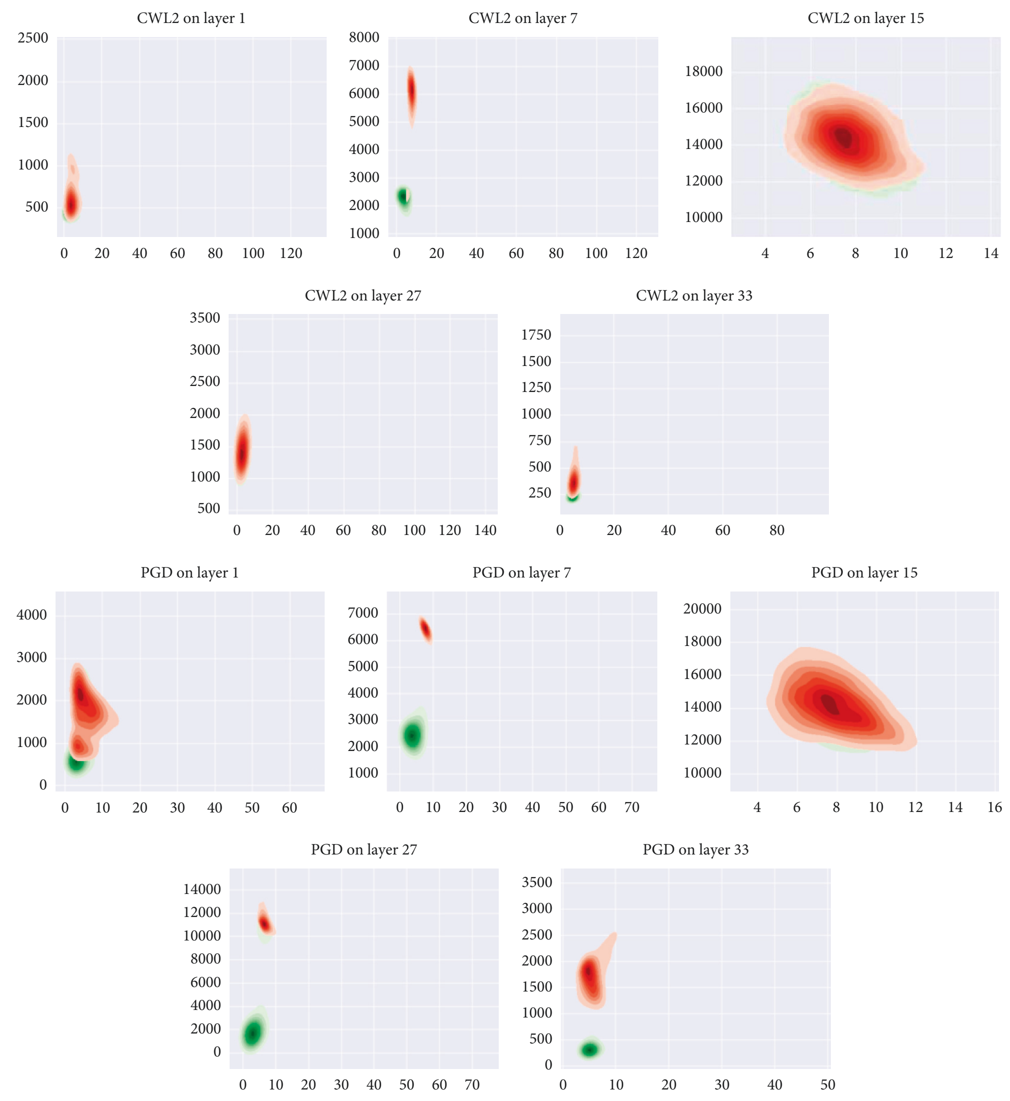

Figure 4: Analysis of reconstruction errors and latent norms in different network layers.

the logit vector is significant for the detection of adversarial samples. In our supervised and partially supervised experiments, we ensemble the latent feature vectors; we believe that this treatment can provide adaptability to different adversarial attack strategies and lead to good performance.

This method has practical significance. For example, this method can be used in target recognition. Without modifying the original neural network, this method can detect images with adversarial perturbation. For example, if adversarial perturbation is added to a picture of a kitten, it may be recognized as other animals in target recognition.

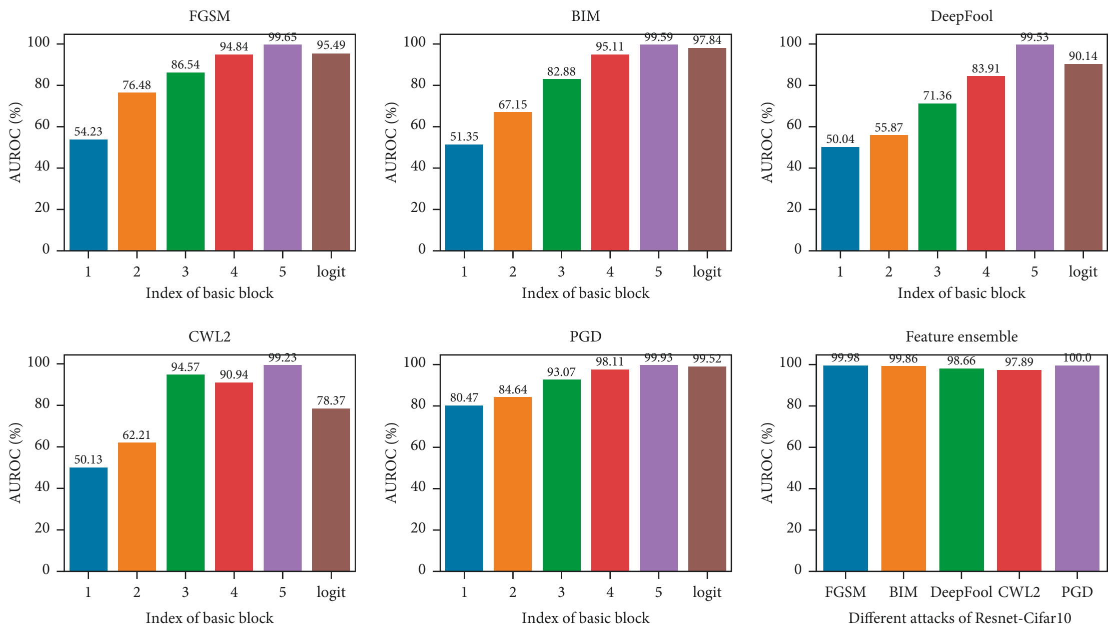

FIGURE 5: AUROC (%) of the threshold-based detector using the latent vector generated by semisupervised GAN's generator at different basic blocks of ResNet trained on the CIFAR10 dataset and the logit vector of the target network. (a) FGSM. (b) BIM. (c) DeepFool. (d) CWL2. (e) PGD. (f) Feature ensemble.

TABLE 1: Supervised scenarios for detecting adversarial samples.

Model	Dataset	Method	Supervised scenario				
			FGSM	BIM	DeepFool	CW	PGD
DenseNet	CIFAR10	Mahalanobis	99.94	99.78	83.41	87.31	97.79
		Ours (lantent)	99.95	**99.98**	97.46	96.11	**99.42**
		Ours (score)	**99.98**	99.93	**98.54**	**96.13**	99.34
	CIFAR100	Mahalanobis	99.86	99.17	77.57	87.05	79.24
		Ours (lantent)	**100.00**	99.86	97.22	98.01	90.35
		Ours (score)	99.89	**99.90**	**97.34**	**98.02**	**91.36**
	SVHN	Mahalanobis	99.85	99.28	95.10	97.03	98.41
		Ours (latent)	**99.95**	**99.85**	99.25	**98.65**	**99.49**
		Ours (score)	99.90	99.80	**99.35**	98.23	99.12
ResNet	CIFAR10	Mahalanobis	99.94	99.57	91.57	95.84	89.81
		Ours (latent)	**99.98**	99.86	**98.66**	97.89	**100.00**
		Ours (score)	99.87	**99.92**	98.65	**98.11**	98.65
	CIFAR100	Mahalanobis	99.77	96.90	85.26	91.77	91.08
		Ours (lantent)	99.92	**99.11**	**94.68**	**97.21**	**99.95**
		Ours (score	**99.93**	97.91	94.34	92.34	92.34
	SVHN	Mahalanobis	99.62	97.15	95.73	92.15	92.24
		Ours (latent)	**99.96**	**99.46**	**99.54**	**99.30**	**99.98**
		Ours (score)	99.65	98.34	96.54	97.56	97.45

TABLE 2: Partially supervised scenarios in detecting adversarial samples.

Model	Dataset	Method	Partially unsupervised scenario				
			FGSM	BIM	DeepFool	CW	PGD
DenseNet	CIFAR10	Mahalanobis	99.94	**99.51**	83.42	87.95	**81.84**
		Ours (latent)	99.95	90.79	**98.06**	95.75	76.00
		Ours (score)	**99.98**	92.23	96.09	**97.18**	78.00
	CIFAR100	Mahalanobis	99.86	**98.27**	75.63	**86.20**	39.32
		Ours (latent)	**100.00**	89.86	**83.14**	79.08	62.35
		Ours (score)	99.89	90.15	80.19	81.09	**64.15**
	SVHN	Mahalanobis	99.85	**99.12**	93.47	96.95	81.40
		Ours (latent)	**99.95**	99.00	**98.71**	**98.16**	**94.15**
		Ours (score)	99.90	99.01	98.29	97.18	92.16

Table 2: Continued.

Model	Dataset	Method	Partially unsupervised scenario				
			FGSM	BIM	DeepFool	CW	PGD
ResNet	CIFAR10	Mahalanobis	99.94	98.91	78.06	93.90	100.00
		Ours (latent)	**99.98**	73.19	**96.79**	**95.71**	**100.00**
		Ours (score)	99.87	**96.15**	94.13	94.10	**100.00**
	CIFAR100	Mahalanobis	99.77	**96.38**	81.95	**90.96**	99.85
		Ours (latent)	99.92	81.18	**83.32**	86.63	**100.00**
		Ours (score)	**99.93**	80.10	80.13	87.01	**100.00**
	SVHN	Mahalanobis	99.62	**95.39**	72.20	86.73	99.92
		Ours (latent)	**99.96**	74.89	**95.97**	89.65	**99.96**
		Ours (score)	99.65	75.14	95.10	**89.75**	99.23

Table 3: Unsupervised scenarios for detecting adversarial samples.

Model	Dataset	Method	FGSM	Unsupervised scenario			
				BIM	DeepFool	CW	PGD
DenseNet	CIFAR10	Odds-testing	45.23	69.01	58.30	61.29	**97.93**
		GM-CGAN	**87.89**	**73.69**	**80.81**	**78.12**	46.45
	CIFAR100	Odds-testing	43.22	65.22	49.53	47.64	**96.91**
		GM-CGAN	**98.09**	**68.34**	**74.42**	**65.72**	41.09
	SVHN	Odds-testing	56.14	71.11	67.81	70.71	**99.25**
		GM-CGAN	**83.35**	**73.56**	**81.86**	**80.16**	46.40
ResNet	CIFAR10	Odds-testing	46.32	59.85	75.58	57.58	96.18
		GM-CGAN	**96.68**	**64.74**	**79.48**	**73.49**	**98.80**
	CIFAR100	Odds-testing	38.26	43.52	61.13	44.74	**93.73**
		GM-CGAN	**80.96**	**81.20**	**80.35**	**67.56**	91.30
	SVHN	Odds-testing	65.09	70.31	77.05	72.12	**99.08**
		GM-CGAN	**94.97**	**89.04**	**94.71**	**83.41**	97.52

This method can be used to detect kitten images with anti-interference. This method can prevent the recognition image from making mistakes.

4. Conclusions

Our article intends to discover the adversarial samples in training data in order to prevent the generation of the highly overconfident distribution in the test phase. The proposed method designs a semisupervised generative adversarial network that is applied to the output of the hidden layers in a neural network to detect the variation of the adversarial samples without modifying the structure of the neural network. In the supervised scenario, the latent feature (or the discriminator's output score information) of the semisupervised GAN and the target network's logit information are used as the input of the external classifier logistic regression classifier to detect the adversarial samples. In the unsupervised scenario, first we proposed a one-class classier based on the semi-supervised Gaussian mixture conditional generative adversarial network (GM-CGAN) to fit the joint feature information of the normal data and then we used a discriminator network to detect normal data and adversarial samples. The novel contribution is that the output of hidden layers of a neural network is analyzed without modifying the neural network.

Data Availability

The [DATA TYPE] data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Science and Technology Project of State Grid Zhejiang Electric Power Co., Ltd.

References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai et al., “Deep speech 2: end-to-end speech recognition in English and Mandarin,” in *Proceedings of the International Conference on Machine Learning*, pp. 173–182, New York, NY, USA, June 2016.

[2] S. Ren, K. He, R. Girshick et al., “Faster r-cnn: towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[3] K. He, X. Zhang, S. Ren et al., “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[4] K. Lee, K. Lee, K. Min et al., “Hierarchical novelty detection for visual object recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1034–1042, Salt Lake City, UT, USA, June 2018.

[5] I. Evtimov, K. Eykholt, E. Fernandes et al., “Robust physical-world attacks on machine learning models,” vol. 2, no. 3, p. 4, 2017, https://arxiv.org/abs/1707.08945.

[6] M. Sharif, S. Bhagavatula, L. Bauer et al., "Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the the 2016 ACM SIGSAC Conference*, Vienna, Austria, October 2016.

[7] A. Madry, A. Makelov, L. Schmidt et al., "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[8] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," 2017, https://arxiv.org/abs/1706.07351.

[9] K. Y. Xiao, T. Vincent, N. M. Shafiullah et al., "Training for faster adversarial robustness verification via inducing relu stability," 2018, https://arxiv.org/abs/1809.03008.

[10] N. Carlini, G. Katz, C. Barrett et al., "Provably minimally-distorted adversarial examples," 2017, https://arxiv.org/abs/1709.10207.

[11] K. Lee, K. Lee, H. Lee et al., "A simple unified framework for detecting outof-distribution samples and adversarial attacks," *Advances in Neural Information Processing Systems*, vol. 1, pp. 7167–7177, 2018.

[12] X. Ma, B. Li, Y. Wang, S. M. Erfani et al., "Characterizing adversarial subspaces using local intrinsic dimensionality," 2018, https://arxiv.org/abs/1801.02613.

[13] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: detecting adversarial examples in deep neural networks," 2017, https://arxiv.org/abs/1704.01155.

[14] P. Yang, J. Chen, C.-J. Hsieh et al., "ML-LOO: Detecting adversarial examples with feature attribution," 2019, https://arxiv.org/abs/1906.03499.

[15] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: a statistical test for detecting adversarial examples," 2019, https://arxiv.org/abs/1902.04818.

[16] N. Papernot and P. McDaniel, "Deep knearest neighbors: towards confident, interpretable and robust deep learning," 2018, https://arxiv.org/abs/1803.04765.

[17] J. J. Engelsma and A. K. Jain, "Generalizing fingerprint spoof detector: learning a one-class classifier," in *Proceedings of the 2019 International Conference on Biometrics (ICB)*, pp. 1–8, Houston, TX, USA, June 2019.

[18] K. Grosse, P. Manoharan, N. Papernot et al., "On the (statistical) detection of adversarial examples," 2017, https://arxiv.org/abs/1702.06280.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, pp. 2672–2680, 2014.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE symposium on security and privacy (sp)*, pp. 39–57, IEEE, San Jose, CA, USA, May 2017.

[22] S.-M. Moosavi-Dezfooli and A. Fawzi, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, Las Vegas, NV, USA, June 2016.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial samples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[24] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: increasing local stability of neural nets through robust optimization," 2015, https://arxiv.org/abs/1511.05432.

[25] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199.

[26] N. Papernot, P. McDaniel, X. Wu et al., "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, IEEE, San Jose, CA, USA, May 2016.

[27] M. Jan Hendrik, G. Tim, V. Fischer et al., "On detecting adversarial perturbations," 2017, https://arxiv.org/abs/1702.04267.

[28] S. Akcay, A. Atapour-Abarghouei, and T. PBreckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian Conference on Computer Vision*, pp. 622–637, Springer, Perth, Australia, January 2018.

[29] T. Salimans, I. Goodfellow, W. Zaremba et al., "Improved techniques for training gans," 2016, https://arxiv.org/abs/1606.03498.

[30] P. Isola, J.-Y. Zhu, T. Zhou et al., "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134, Honolulu, HI, USA, July 2017.

[31] A. Krizhevsky, V. Nair, and Hinton, "Cifar-10 (canadian institute for advanced research)," http://www.cs.toronto.edu/kriz/cifar.html.

[32] Y. Netzer, T. Wang, A. Coates et al., "Reading digits in natural images with unsupervised feature learning," 2011, http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[33] G. Huang, Z. Liu, L. Van Der Maaten et al., "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, Honolulu, HI, USA, July 2017.

[34] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[35] G. Xu, Z. Liu, X. Li, and C. Change Loy, "Knowledge distillation meets self-supervision," in *Proceedings of the European Conference on Computer Vision*, Springer, Cham, Switzerland, 2020.

[36] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: beyond empirical risk minimization," 2017, https://arxiv.org/abs/1710.09412.

[37] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," 2021, https://arxiv.org/abs/2102.01356.