Hindawi
Security and Communication Networks
Volume 2020, Article ID 3608173, 9 pages
https://doi.org/10.1155/2020/3608173

Research Article

Cycle-Consistent Adversarial GAN: The Integration of Adversarial Attack and Defense

Lingyun Jiang, Kai Qiao, Ruoxi Qin, Linyuan Wang, Wanting Yu, Jian Chen, Haibing Bu, and Bin Yan

Academy of Information Systems Engineering, PLA Strategy Support Force Information Engineering University, Zhengzhou 450001, China

Correspondence should be addressed to Bin Yan; ybspace@hotmail.com

Received 20 September 2019; Revised 10 December 2019; Accepted 13 December 2019; Published 21 February 2020

Academic Editor: Clemente Galdi

In image classification of deep learning, adversarial examples where input is intended to add small magnitude perturbations may mislead deep neural networks (DNNs) to incorrect results, which means DNNs are vulnerable to them. Different attack and defense strategies have been proposed to better research the mechanism of deep learning. However, those researches in these networks are only for one aspect, either an attack or a defense. There is in the improvement of offensive and defensive performance, and it is difficult to promote each other in the same framework. In this paper, we propose Cycle-Consistent Adversarial GAN (CycleAdvGAN) to generate adversarial examples, which can learn and approximate the distribution of the original instances and adversarial examples, especially promoting attackers and defenders to confront each other and improve their ability. For CycleAdvGAN, once the Generator A and D are trained, G_A can generate adversarial perturbations efficiently for any instance, improving the performance of the existing attack methods, and G_D can generate recovery adversarial examples to clean instances, defending against existing attack methods. We apply CycleAdvGAN under semiwhite-box and black-box settings on two public datasets MNIST and CIFAR10. Using the extensive experiments, we show that our method has achieved the state-of-the-art adversarial attack method and also has efficiently improved the defense ability, which made the integration of adversarial attack and defense come true. In addition, it has improved the attack effect only trained on the adversarial dataset generated by any kind of adversarial attack.

1. Introduction

With Deep Neural Networks (DNNs) rapid development, they have achieved great success in various tasks handling the image recognition [1], text processing [2], and speech recognition [3]. Despite the great success, DNNs have been proved to be vulnerable and susceptible to adversarial example [4], and the carefully crafted samples look similar to natural images but are designed to mislead a pretrained model. On the one hand, an adversarial example leads to potential security threats by attacking or misleading the practical deep learning applications, for example, mistaking a stop sign for a yield sign [5] when autodriving, and a thief for staff when face recognition [6]. On the other hand, adversarial examples are also valuable and beneficial to not only the deep learning models but also the machine learning model, as they can enhance the robust of models and provide insights into their strengths, weaknesses, and blind spots [7].

The strategy to generate adversarial examples is to intentionally add imperceptible perturbations to clean instances, for fooling DNNs to make wrong predictions. In the past years, various attack algorithms have been developed to produce adversarial examples in the white-box manner with the knowledge of the structure and parameters of a given model, where the adversary has full access to the classifier. A more straightforward approach is to change pixels value simultaneously in the direction of the gradient such as fast gradient sign method (FGSM) [8] and iterative variants of gradient-based methods (BIM) [9]. However, they quickly find the perturbations at the expense of uncontrollability.

And another method based on optimization aims to find the smallest possible attack disturbance and use complex linear search methods to find the best disturbance value such as box-constrained LBFGS [4] and Carlini and Wagner attacks [10], but they are complicated and take a long time. And one innovative method is to utilize Generative Adversarial Networks (GANs) as part of their approach to generate adversarial examples which made adversarial examples more natural to human [11], such as AdvGAN [12] and natural GAN [13]. Xiao et al. [12] proposed AdvGAN to generate adversarial examples with generative adversarial networks (GANs), which can learn and approximate the distribution of original instances. Once trained, the feed-forward generator can produce adversarial perturbations efficiently. However, they only generate perturbation by adding loss to make target model predict in a wrong way, instead of considering the relationship between different adversarial attack methods to find the distribution of perturbations. At the same time, defense algorithms have made progress with advances in attack algorithms. So far, there are two main ideas to defend against adversarial attack. A more straightforward approach is to make the model more robust by enhancing training data or adjusting learning strategies, such as adversarial training that retrains a neural network to predict correct labels for adversarial examples [8] and defensive distillation [14] that migrate knowledge of complex networks to simple networks. The second is a series of detection mechanisms for detecting and rejecting against the examples. One important way in the second method is to utilize GAN to defense adversarial example [15], with the advantage to learn the latent distribution of perturbation and reconstruct clean samples better. Shen et al. [16] propose a framework for reconstructing images based on GAN using adversarial examples to generate clean samples similar to the original samples. First, the original sample and the adversarial example training are used to generate the GAN. After the training, the adversarial example and the original sample are first passed through the generator to eliminate the adversarial perturbations, and then the target classifier is classified. The author consider that the misclassification of the adversarial examples is mainly caused by some pixel-level intentional imperceptible perturbation of the input image, so it is desirable to propose an algorithm to eliminate the adversarial perturbations of the input image, thereby achieving the purpose of defending against the attack.

Network-based techniques have achieved satisfying performance not only in terms of attacks but also defenses owing to their great power for generating high-quality synthetic data and detecting these subtle differences to distinguish between adversarial and clean examples. However, existing attack methods exhibit low efficacy when attacking black-box models. For example, most of existing methods, such as FGSM and optimization methods, cannot successfully attack them in the black-box manner, due to the poor transferability [17]. Meanwhile, existing defense methods also have poor transferability, as they only can defend against a certain attack method. On the other hand, in the previous white-box attacks, the adversary needs to have white-box access to the architecture and parameters of the model all the time.

In addition, those studies in these networks are only for one aspect, either an attack or a defense, not considering that attacks and defenses should be interdependent and mutually reinforcing, just like the relationship between spears and shields. Inspired by the idea of GAN, just as the generative model is pitted against an adversary, if we then train a model consisting of attack and defense, and the attacker and defender are also fighting against each other all the time, it can improve the attack ability as well as the defense ability. Meanwhile, considering the existence of adversarial examples, there are different viewpoints because of the unexplained nature of DNNs [18–21]. However, it is widely accepted that the linear properties of deep neural networks in high latitude space are sufficient to generate an adversarial attack [8]. There is also a guess that adversarial examples and clean samples are subject to two independent distributions [22], which makes style transfer between the two data possible. To use domain adaptation for style transfer, Zhu et al. [23] proposed CycleGAN for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. The training procedure requires a set of style images in the same style and a set of target images of similar content. The learned mapping function takes one target image as input and transforms it into the style domain. From the mechanism of generating adversarial examples, since there is a companion relationship between the original sample and the adversarial example, they can be regarded as a sample pair, subjected to different distributions but interrelated. Therefore, from the perspective of image style migration, this set of sample pairs can be transformed by domain migration. The original sample is transferred into the adversarial example corresponding to the adversarial attack, and the adversarial example is transferred to the original sample corresponding to the adversarial defense. From the perspective of generation mechanism of adversarial example, the mutual transformation between adversarial examples and clean samples are realized by adding specific perturbation, and it can be considered that the two are subject to different data distributions but are related. Therefore, through GAN learning the relationship between the two data distributions of adversarial examples and clean samples, the study focuses on different issues ranging from the generation of single adversarial example to data distribution of adversarial example, providing a new research direction for further efficient generation of better migration of adversarial examples and their effective recovery.

Consequently, utilizing the cycle consistency idea of CycleGAN [23], we apply a similar paradigm to combine the attacker and defender to promote each other and better learn the latent distribution of adversarial examples and clean instances, which proves the adversarial and clean examples are not twins and also improve the transferability. In this paper, we propose training a Cycle-Consistent Adversarial GAN (CycleAdvGAN) to achieve the integration of adversarial attack and defense. Once trained, there is no need to access the target model itself anymore and it will not matter whether it is in the manner of attack or defense, and

the CycleAdvGAN can generate perturbations and recover the adversarial examples such that the resulting example must be realistic according to a discriminator network.

To evaluate the effectiveness of our strategy CycleAdvGAN, we experiment on different datasets including MNIST and CIFAR10 for an ensemble of target models. We evaluate these attack strategies in both semiwhite-box and black-box settings. We show that adversarial examples generated by CycleAdvGAN have higher success rates in both semiwhite-box and black-box attacks and also have good effect in defense, due to the fact that attack and defense promote each other, better leaning the differentiation between adversarial example and clean example. In summary, we made the following contributions.

(i) We are the first to achieve the integration of adversarial attack and defense by proposing Cycle-Consistent Adversarial GAN.

(ii) We show a high success rate of attacks as well as defense in semiwhite-box attack only trained on the adversarial dataset generated by any kind of adversarial attack.

(iii) We demonstrate a powerful capability of transferability and it does not matter whether it is in the manner of attack or defense.

(iv) We indirectly demonstrate the adversarial and clean data are not twins and subject to two different distributions.

2. Materials and Methods

In this section, we will first introduce the problem definition and then briefly describe two approaches we utilized to generate adversarial images and at last elaborate the framework, formulation, and corresponding network architecture of our proposed Cycle-Consistent Adversarial GAN.

2.1. Problem Definition. Let $A \subseteq R^n$ be the clean feature space, with n the number of features. Suppose that (a_i, t_i) is the ith instance within the clean dataset, which is comprised of feature vectors $a_i \in A$, generated according to some unknown distribution $a_i \sim P_A$, and t_i is the corresponding true class labels.

Let $B \subseteq R^n$ be the adversarial feature space, with n the number of features. Suppose (b_i, l_i) is the ith instance within the adversarial dataset, which is comprised of feature vectors $b_i \in B$, generated according to some unknown distribution $b_i \sim P_B$, and l_i is the corresponding predict labels.

The learning model aims to mapping functions between two domains A and B given the training samples $\{a_i\}_{i=1}^N$ where $a_i \in A$ within the clean dataset and $\{b_i\}_{i=1}^M$ where $b_i \in B$ within the adversarial dataset. We denote the data distribution as $a_i \sim P_A$ and $b_i \sim P_B$. Given an instance a_i, the goal of the attack generator (G_A) is to generate adversarial example b_i, which is classified as $F(b_i) \neq t_i$ (untargeted attack), where t denotes the true label. And given an adversarial example b_i, the goal of the defense generator (G_D) is to recover b_i to clean instance a_i, which is classified as $F(a_i) = t_i$, where t denotes the true label. b_i should also be close to the original instance a_i in terms of L_2 or other distance metric.

2.2. The CycleAdvGan Method. Figure 1 illustrates the overall architecture of CycleAdvGAN including two mappings, the generator G_A: $A \longrightarrow B$ and the generator G_D: $B \longrightarrow A$, which mainly consist of five parts: the generator G_A, the generator G_D, the discriminator of domain $A(D_A)$, the discriminator of domain $B(D_B)$, and the target neural network F. Here the generator G_A takes the original instance a as its input and generates perturbation $G_A(a)$. Then, the generated adversarial example $b_{\text{fake}} = a + G_A(a)$ will be sent to the discriminator D_B, which is used to distinguish the generated data and the original adversarial example b. D_b encourages G_A to translate A into outputs indistinguishable from domain B. As shown in Figures 1(a) and 1(b), the two parts of the architecture are symmetrical. Therefore, the generator G_D takes the adversarial example b as its input and generates a perturbation $G_D(b)$. Then, the generated clean instance$a_{\text{fake}} = b + G_D(b)$ will be sent to the discriminator D_A, which is used to distinguish the generated data and the original instance a. D_A encourages G_D to translate B into outputs indistinguishable from domain A. To fulfill the goal of fooling the learning model, we perform the white-box attack, where the target model is F in this case. F takes b_{fake} and a_{fake} as its input and outputs and its loss is L_{adv} which represents the distance between the prediction and the target class t (targeted attack), or the opposite of the distance between the prediction and the ground truth class (untargeted attack).

2.2.1. Adversarial Loss for G_A. We apply adversarial losses to both mapping functions. For the mapping function G_A: $A \longrightarrow B$ and its discriminator D_B, we express the objective as

$$L_{G_A}(G_A, D_B) = E_{b\sim P_B}[\log(D_B(b))] + E_{a\sim P_A}[\log(1 - D_B(G_A(a)))], \quad (1)$$

where generator G_A aims to generate imperceptible perturbation $G_A(a)$ that is added to a for looking similar to original instance, while D_B aims to distinguish between generated adversarial example and adversarial example.

2.2.2. Adversarial Loss for G_D. We introduce a similar adversarial loss for the mapping function G_D: $B \longrightarrow A$, and its discriminator D_A. The loss is defined as

$$L_{G_B}(G_D, D_B) = E_{a\sim P_A}[\log(D_A(a))] + E_{b\sim P_B}[\log(1 - D_A(G_D(b)))], \quad (2)$$

where generator G_D aims to recover adversarial example b to clean example, while D_A aims to distinguish between generated clean example and clean example.

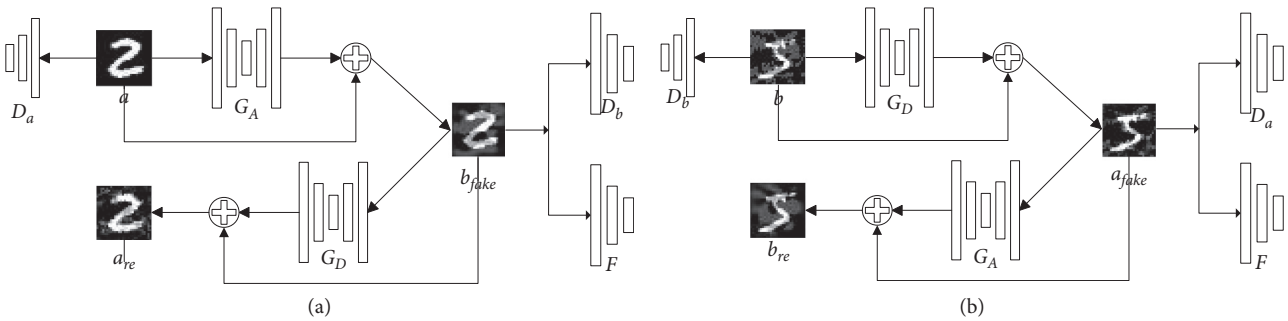

Figure 1: The framework of CycleAdvGAN consists of two generators G_A and G_D, two discriminators D_a and D_b, and attacking a target model F. (a) The attack cyclic part of the architecture; (b) the defense cyclic part of the architecture.

2.2.3. Cycle Consistency Loss. After competing with discriminator D_a in the min-max game, G_A should be able to generate visually realistic adversarial examples. However, since no pairwise supervision is provided, the deblurred image may not retain the content information in the original blurred image. Inspired by CycleGAN for style transfer which is shown to imply the cycle consistency constraints, we also introduce variants of cycle consistency losses to ensure the successful integration of adversarial attack and defense. The adversarial loss constrains the output of generator after being added to input to look like the instance in transfer domain. However, adversarial losses alone cannot guarantee that the learned function can map an individual input a to a desired output b. To further generate the specific perturbation, we attempt to add cycle consistency loss to the architecture. The loss is defined as

$$L_{\text{cyc}} = E_{a\sim P_A}\left[\left\|G_D\left(G_A\left(a\right)\right) - a\right\|_1\right] + E_{b\sim P_B}\left[\left\|G_A\left(G_D\left(b\right)\right) - b\right\|_1\right]. \tag{3}$$

2.2.4. Adversarial Loss for F. The loss for fooling the target model F in an untargeted attack is

$$L_{\text{adv}} = -E_{a\sim P_A}\left[L_F\left(b_{\text{fake}}, l_c\right)\right] + E_{b\sim P_B}\left[L_F\left(a_{\text{fake}}, l_c\right)\right], \tag{4}$$

where l_t is the target label and l_c represents the true class. Meanwhile, L_F denotes the loss function (e.g., cross-entropy loss) used to train the original model F. The L_{adv} losses encourage the perturbed image to be misclassified and the recovered instance to be correctly classified.

2.2.5. Total Loss. We optimize a min-max objective function $\min_{G_A,G_B,F}\max_{D_A,D_B}L$, where the loss L is defined as

$$L = \lambda_1 L_{G_A} + \lambda_2 L_{G_B} + \lambda_3 L_{\text{cyc}} + \lambda_4 L_{\text{adv}}. \tag{5}$$

λ_1, λ_2, λ_3, and λ_4 are the weights to balance the multiple objectives. The next section will provide more training details and discuss the appropriate weights.

2.3. The Methods of Generating Adversarial Examples Datasets. In this subsection, two common approaches mentioned above were utilized to generate adversarial examples as the training sets are provided with a brief description.

2.3.1. Fast Gradient Sign Method Attack (FGSM). Goodfellow et al. proposed a fast method called Fast Gradient Sign method to generate adversarial examples [8]. They only performed one step gradient update along the direction of the sign of the gradient at each pixel. Their perturbation can be expressed as

$$p = \epsilon \operatorname{sign}\left(\nabla J\left(\theta, I_c, l\right)\right), \tag{6}$$

where ϵ is the magnitude of the perturbation. The generated adversarial example x_{adv} is calculated as $x_{\text{adv}} = x + p$. This perturbation can be computed by using backpropagation. They claimed that the linear part of the high dimensional deep neural network could not resist adversarial examples, although the linear behaviour speeded up training. Regularization approaches are used in deep neural networks such as dropout. Pretraining could not improve the robustness of networks.

2.3.2. Basic Iterative Method (BIM). Kurakin et al. applied adversarial examples to the physical world [9]. They extended Fast Gradient Sign method by running a finer optimization (smaller change) for multiple iterations. In each iteration, they clipped pixel values to avoid large change on each pixel:

$$I_p^{i+1} = \operatorname{Clip}\left\{I_p^i + \alpha \operatorname{sign}\left(\nabla J\left(\theta, I_p^i, l\right)\right)\right\}, \tag{7}$$

where I_p^i denotes the perturbed image at the ith iteration, Clip limits the change of the generated adversarial image in each iteration with its argument at ϵ and α determines the step size (normally, $\alpha = 1$). The BIM algorithm starts with $I_p^0 = I_c^i$ and runs for the number of iterations determined by the formula $[\min(\epsilon + 4, 1.25\epsilon)]$.

2.3.3. Projected Gradient Descent (PGD). Madry et al. proposed an attack called "projected gradient descent" (PGD) used in adversarial training [24]. Their PGD attack consists of initializing the search for an adversarial example at a random point within the allowed norm ball and then running several iterations of the basic iterative method [9] to

find an adversarial example. The noisy initial point creates a stronger attack than other previous iterative methods such as BIM, and performing adversarial training with this stronger attack makes their defense more successful.

2.4. Implementation

2.4.1. Network Architecture. Next, we briefly introduce the network architectures in our CycleAdvGAN framework. We adopt the variant architecture for our generator and discriminator from Zhu et al. who have shown impressive results for neural style transfer in the absence of paired examples.

2.4.2. Generator. This generator network contains two stride-2 convolutions, several residual blocks, and two strided-2 deconvolutions. We employ the Resnet architecture in our generator, which allows low-level information to shortcut across the network, leading to better results. We use four blocks for 28×28 Mnist images and four blocks for 32×32 cifar10. The encoder-decoder architecture consists of

Encoder: C8-C16-C32

Decoder: C32-C16-C8

where C means the channel of kernel.

2.4.3. Discriminator. For the discriminator networks, we use three stride-2 convolutions. The last layer of discriminator network is fed into a linear layer to generate a 1-dimensional output, followed by a Sigmoid function. Our discriminator architecture is

C8-C16-C32

2.4.4. Target Model. For the target model, we trained different models on MNIST [25] and CIFAR10 [26], respectively. For MNIST, in all of our experiments, we generate adversarial examples for two models whose architectures are shown in Table 1. For CIFAR10, we select ResNet-18 [27] and VGG-16 [28] for our experiments. We show the classification accuracy of pristine MNIST and CIFAR10 test data in Table 2. The targeted models F could be any given deep networks with the last two layers accessible (e.g., soft-max layer and the layer before it). These two layers are used as a part of L_{adv}. To perform adversarial attack, the loss L_{adv} encourages the adversarial example b to be misclassified by F and the clean examples a to be correctly classified by F.

2.4.5. Training Details. Our code and models will be available upon publication. We apply the loss in Carlini and Wagner [10] as our loss $L_{\text{adv}} = \max(\max_{i \neq t} F(x_A)_i - F(x_A)_i, k)$, where t is the true class, and F represents the target network in the semiwhite-box setting. We set the confidence $k = 0$ and use Adam as our solver [29]. For L_{GAN}, just as Zhu et al. did, we replaced the negative log likelihood objective by a least-squares loss. This loss is more stable during training and generates higher quality results. In particular, for a GAN loss $L_{\text{GAN}}(G, D, A, B)$, we train the generator $E_{a \sim P_A}[(D(G(x)) - 1)^2]$ to minimize and train the discriminator to minimize $E_{b \sim P_B}[(D(b) - 1)^2] + E_{a \sim P_A}[D(G(x))^2]$.

Table 1: Target model architectures for the MNIST. A is the LENET5. B is the LENET5 with dropout.

A	B
Conv1 (32, 3, 3) + Relu	Conv1 (32, 3, 3) + Relu
Max pooling	Conv1 (32, 3, 3) + Relu
Conv1 (64, 3, 3) + Relu	Max pooling
Max pooling	Dropout (0.5)
FC (200)+Relu	FC (128) + Relu
FC (10)	Dropout (0.5)
	FC (10)

Table 2: Classification accuracy rates (%) of benign input for different target models trained by us on MNSIT and CIFAR10.

Model	MNIST		CIFAR10	
	A	B	Resnet-18	VGG-16
Classification accuracy	98.85	99.14	85	84

2.4.6. Implementation Details. In our experiments, we use Pytorch for the implementation and test them on a NVIDIA Tesla V100 GPU cluster in Nvidia DGX station. We train CycleAdvGAN for 100 epochs with a batch size of 64, with the learning rate of 0.01, decreased by 10% every 20 steps.

3. Results and Discussion

In this section, we first evaluate CycleAdvGAN for both semiwhite-box and black-box settings on MNIST and CIFAR10. We then apply CycleAdvGAN to generate adversarial examples on different target models and test the attack success rate for them. Meanwhile, we also recover adversarial examples to clean instances and test the recovery success rate for them and show that our method can achieve higher attack success rates as well as the higher recovery success rates. We use the classification accuracy to measure attacking performance and defending performance with the lower accuracy indicating better attacking performance and the higher accuracy indicating better defending performance. We generate all adversarial examples for different attack methods based on the L_∞ bound as 0.3 on MNIST and 0.03 on CIFAR10.

3.1. CycleAdvGAN in Semiwhite-Box Setting. In semiwhite-box condition, there is no need to access the original target model after the generator is trained, in contrast to traditional white-box condition. First, we apply different architectures for the target model F for MNIST and with ResNet-18 and VGG-16 for CIFAR10. We apply CycleAdvGAN to perform semiwhite-box setting against each model on MNIST dataset and CIFAR10 dataset. From the classification accuracy rate of attacking and defending in Figure 2, with the increasing of

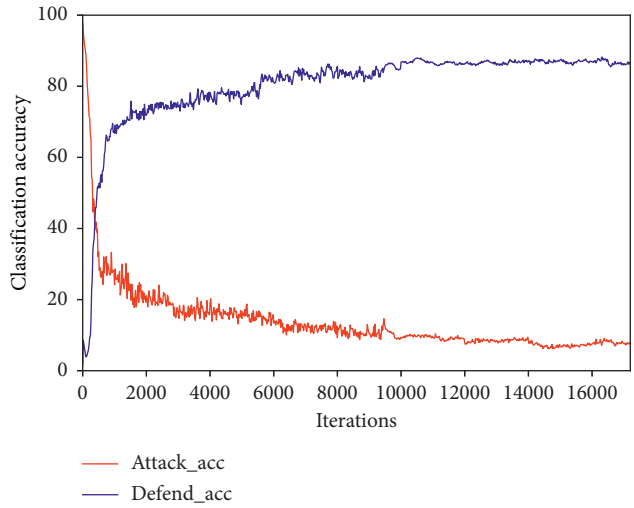

Figure 2: The change of attack accuracy and defend accuracy with the increasing of iterations.

the number of training iterations, the accuracy of attacker on the test set is continuously decreasing, and the accuracy of defender is continuously increasing. It shows that the attack generator and the defense generator are constantly improving their respective attack and defense capabilities in mutual confrontation in the training process, and the attacking and defending effects are improved. It is further proved that the attacker and defender can be integrated under this framework.

From the performance of semiwhite-box setting (classification accuracy rate) in Table 3, we can see that CycleAdvGAN is able to generate adversarial instances to attack all models with high attack success rate compared to other state-of-the-art attacks, for example, the classification rate of adversarial examples in MNSIT datasets decreased from 6.12 to 2.54%. It shows that CycleAdvGAN can improve the attack effect and has better attack performance than the original attack method after training with adversarial examples crafted by only some kind of white-box attack method. Meanwhile, from the performance of semiwhite-box condition (defense with G_D) in Table 3, we can also see that CycleAdvGAN is able to recover adversarial examples to clean samples, so as to significantly improve classification accuracy rates; for example, the classification accuracy in the MNIST dataset improved from 6.12 to 98.12%. Defense was efficiently proved by this, and our proposed CycleAdvGAN can achieve the integration of adversarial attack and defense.

As shown in Table 3, we further analyse that in the MNIST dataset, and it can be seen that B architecture has better resistance against adversarial attack, especially for FGSM method. This is because dropout [30] was added to B architecture, which can enhance the robustness of neural network. Attacking ability and defending ability is highly correlated with the capacity of the learning model generating the adversarial examples. For example, adversarial examples generated by B architecture show good attacking performance on A architecture and A architecture has a better classification accuracy performance after defending with G_D.

Table 3: Classification accuracy rates of adversarial examples crafted by FGSM, BIM, PGD, and CycleAdvGAN. FGSM (attack with G_A) and FGSM (defense with G_D) means that we train the CycleAdvGAN with the adversarial examples crafted by FGSM.

Method	MNIST		CIFAR10	
	A	B	Resnet-18	VGG-16
FGSM [8]	6.12	10.53	10.27	13.2
FGSM (attack with G_A)	2.54	1.27	5.2	3.6
FGSM (defense with G_D)	98.12	92.45	54.8	43.8
BIM [9]	0.76	1.13	8.96	8.97
BIM (attack with G_A)	0.6	0.43	1.98	2.6
BIM (defense with G_D)	94.6	93.15	44.8	38.6
PGD [24]	0.67	0.88	9.8	8.51
PGD (attack with G_A)	0.54	0.46	2.2	2.4
PGD (defense with G_D)	95.4	94.3	48.2	40.4

Because B architecture usually owns more complicated network structure with dropout, B architecture has better capability than A architecture in practice. And our proposed method can greatly increase the attack success rate, because our network learns the potential distribution of adversarial examples.

As is shown in Figure 3, the aggressivity can be added by G_A to original samples, and the aggressivity of adversarial examples can be eliminated by G_D.

3.2. CycleAdvGAN in Black-Box Setting. In this section, we evaluate the transferability of the CycleAdvGAN in the black-box attacking settings. In black-box attacks, we train the CycleAdvGAN in certain target models and optimize the generator accordingly. Once the CycleAdvGAN trained, we generate adversarial examples or recover them through the G_A and G_D. Tables 4 and 5 respectively, show the classification accuracy of MNIST and CIFAR10 datasets, when transferring attacks between different classification models. We first train CycleAdvGAN for a source attacked model in certain attack method and then apply the instances

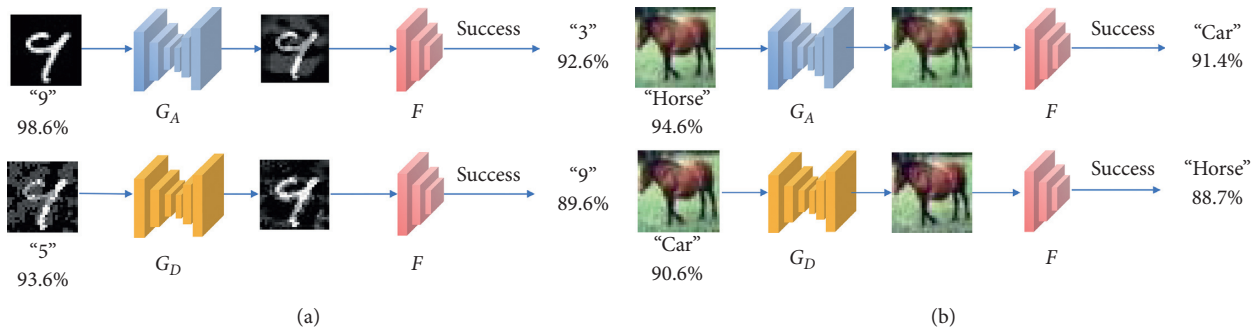

Figure 3: Recognition performance. Before testing, clean instances and adversarial examples, respectively, were fed into trained G_A and G_D before being recognized. (a) The CycleAdvGAN in MNIST dataset. (b) The CycleAdvGAN in CIFAR10 dataset.

Table 4: Classification accuracy of adversarial examples generated by different methods transferred between different models on MNIST. G_A(FGSM) means that we train the CycleAdvGAN with the adversarial examples crafted by FGSM.

Target		Source A					
					G_A		
		FGSM	BIM	PGD	FGSM	BIM	PGD
B	Without G_D	46.530	28	20.8	34	24	18.8
	Defense with G_D	98.2	97.3	96.8	*	*	*

Table 5: Classification accuracy of adversarial examples generated by different methods transferred between different models on CIFAR10.

Target		Source ResNet-18					
					G_A		
		FGSM	BIM	PGD	FGSM	BIM	PGD
VGG-16	Without G_D	32.6	19.7	18.6	23.48	13.6	12.7
	Defense with G_D	43.1	33.7	35.6	*	*	*

generated to other target models. The first three columns of Table 4 refer to the accuracy rates of the adversarial examples generated with different attack methods in the architecture of LeNet5 under the architecture of LeNet5_dropout with G_D and without G_D. The last three columns refer to the accuracy of the adversarial examples generated by the G_A trained with different attack methods in the architecture of LeNet5 to attack LeNet5_dropout. As shown in Table 4, the MNIST classification rate generated by BIM is from 28 to 98% after defending with G_D, and G_A's transferability, better than the corresponding attack methods, from which we can get the following conclusions.

(i) Adversarial examples generated by CycleAdvGAN have very encouraging transferability among different target models, which means attack by our CycleAdvGAN can perform quite well in black-box setting.

(ii) After defending with G_D, the classification accuracy rates have been significantly improved even though source and target model are different, which means defense by our CycleAdvGAN can also perform quite well in black-box setting.

(iii) From the above two points, our proposed CycleAdvGAN method can effectively be applied practically.

3.3. High Transferability of Adversarial Examples Analysis. As shown in Table 6, the first three columns refer to the classification accuracy of adversarial examples generated by a certain attack method after defense with G_D trained by the adversarial datasets generated by other attack methods. And the last column refers to the classification accuracy of clean samples after defense with G_D trained by the adversarial datasets generated by different attack methods defenses. It shows that CycleAdvGAN is trained with adversarial dataset crafted by certain attack method, but it still defends another attack method through G_D recovering the adversarial example to clean example. Meanwhile, it also does not decrease the original classification accuracy of the target model after feeding the clean example to G_D. From this result, it can be further demonstrated that the adversarial and clean data are not twins and subject to two different distributions because the CycleAdvGAN can learn the similar adversarial distribution from different adversarial example datasets crafted by different attack mehtod.

Table 6: Classification accuracy rates of adversarial examples recovered by G_D which trained with different datasets generated by certain method. G_D (FGSM) means that we train the CycleAdvGAN with the adversarial examples crafted by FGSM.

Source		Target			
		FGSM	BIM	PGD	Clean
G_D	FGSM	*	93.5	92.3	98.3
	BIM	94.6	*	95.2	98.4
	PGD	95.6	94.5	*	97.8

Table 7: The cosine similarity of four successive perturbations in FGSM, BIM, PGD, and CycleAdvGAN when attacking ResNet-18 model. The results are averaged over 10000 images.

	FGSM [8]	BIM [9]	PGD [24]	CycleAdvGAN
Cosine similarity	0.3072	0.35	0.33	**0.44**

Table 8: Comparison with the state-of-the-art attack methods. Run time is measured for generating 1,0000 adversarial instances during test time.

	FGSM [8]	BIM [9]	PGD [24]	CycleAdvGAN	
				Attack	Defense
Run time	1.734s	18.436s	20.256s	**0.52 s**	**0.53 s**

To interpret why CycleAdvGAN demonstrates better transferability, we further examine the update directions given by FGSM, BIM, PGD, and CycleAdvGAN along the iterations. We calculate the cosine similarity [31] of two successive perturbations and show the results in Table 7 when attacking MNIST and CIFAR10 datasets. The update direction of CycleAdvGAN is more stable than that of BIM due to the larger value of cosine similarity in CycleAdvGAN. Recall that the transferability comes from the fact that models better learn the distribution of adversarial examples rather than perturbation to a certain sample, resulting in better transferability for black-box attacks. Another interpretation is that the GAN can reconstruct images naturally, which may be helpful for the transferability.

3.4. The Better Efficiency of Generating Adversarial Examples. In general, as shown in Table 8, CycleAdvGAN has obvious advantages on the integration of adversarial attack and defense over other white-box and black-box methods. For instance, regarding computation efficiency, CycleAdvGAN performs much faster than others even including the efficient FGSM in attack, although CycleAdvGAN needs extra training time to train the generator. Besides, FGSM and optimization methods can only perform white-box attack, while CycleAdvGAN is able to attack in semiwhite-box setting.

4. Conclusions

From the mechanism of generating adversarial examples, since there is a companion relationship between the original sample and the adversarial example, they can be regarded as a sample pair, subject to different distributions but interrelated. In this paper, we have proposed CycleAdvGAN to generate adversarial examples and recovery adversarial examples in cycle consistency. More importantly, we show that the proposed objective is enhancing both the attack effect and the defense effect through the integration of adversarial attack and defense. We show that it can improve attack effect only trained on the adversarial dataset generated by corresponding adversarial attack method.

Further, in our CycleAdvGAN framework, once trained, the G_A can produce adversarial perturbations and G_D can eliminate adversarial perturbation; both done efficiently. In addition, the CycleAdvGAN can work with other attacks or defense strategies, not conflicting with other existing frameworks. It can also perform both semiwhite-box and black-box settings with high attack success rate as well as defense success rate.

More importantly, we demonstrated that the adversarial and clean data are not twins, subjected to two different distributions. Consequently, the CycleAdvGAN can better learn their latent distribution, instead of targeting special image, so as to improve transferability. Significant transfer performances achieved by our crafted perturbations can pose a substantial threat to the deep-learned systems in terms of black-box attacking. Therefore, it is an important research direction to be focused on in order to build reliable deep learning systems.

Data Availability

The detailed information about the Mnist and Cifar10 is provided in previous studies, and the public dataset can be downloaded from https://www.cs.toronto.edu/~kriz/cifar.html and http://yann.lecun.com/exdb/mnist/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Plan of China (No. 2017YFB1002502), the National Natural Science Foundation of China (No. 61701089), and the Natural Science Foundation of Henan Province of China (No. 162300410333).

References

[1] I. S. Krizhevsky and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105, Curran Associates, Inc., Red Hook, NY, USA, 2012.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, https://arxiv.org/abs/1409.0473.

[3] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] E. D. Cubuk, B. Zoph, S. S. Schoenholz, and Q. V. Le, "Intriguing properties of adversarial examples," 2017, https://arxiv.org/abs/1711.02846.

[5] K. Eykholt, I. Evtimov, E. Fernandes et al., "Robust physical-world attacks on deep learning models," 2017, https://arxiv.org/abs/1707.08945.

[6] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Adversarial generative nets: neural network attacks on state-of-the-art face recognition," 2017, https://arxiv.org/abs/1801.00349.

[7] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: attacks and defences," 2017, https://arxiv.org/abs/1705.07204.

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

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

[10] 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, 2017.

[11] A. Liu, X. Liu, J. Fan et al., "Perceptual-sensitive gan for generating adversarial patches," in *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI, Menlo Park, CA, USA, 2019.

[12] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," 2018, https://arxiv.org/abs/1801.02610.

[13] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017, https://arxiv.org/abs/1710.11342.

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

[15] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: defence to adversarial perturbations with gan," 2017, https://arxiv.org/abs/1705.03387.

[16] S. Shen, G. Jin, K. Gao, and Y. Zhang, "Ape-gan: adversarial perturbation elimination with gan," 2017, https://arxiv.org/abs/1707.05474.

[17] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, https://arxiv.org/abs/1611.01236.

[18] A. Rozsa, M. Gunther, and T. E. Boult, "Towards robust deep neural networks with BANG," 2016, https://arxiv.org/abs/1612.00138.

[19] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," 2017, https://arxiv.org/abs/1704.03453.

[20] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *Proceedings of the 2016 international joint conference on neural networks (IJCNN)*, pp. 426–433, IEEE, Vancouver, BC, Canada, 2016.

[21] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," 2019, https://arxiv.org/abs/1905.02175.

[22] Z. Gong, W. Wang, and W. S. Ku, "Adversarial and clean data are not twins," 2017, https://arxiv.org/abs/1704.04960.

[23] J. Y. Zhu, T. Park, P. Isola et al., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232, IEEE, San Jose, CA, USA, 2017.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny images[R]. Technical Report*, University of Toronto, 2009.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, 2016.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[29] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] Y. Dong, F. Liao, T. Pang et al., "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, IEEE, Salt Lake City, UT, USA, 2018.