

# AI-GAN: Attack-Inspired Generation of Adversarial Examples

Tao Bai<sup>1</sup>, Jun Zhao<sup>1</sup>, Jinlin Zhu<sup>1\*</sup>, Shoudong Han<sup>2</sup>, Jiefeng Chen<sup>3</sup>, Bo Li<sup>4</sup>

<sup>1</sup>Nanyang Technological University

<sup>2</sup>Huazhong University of Science and Technology

<sup>3</sup>University of Wisconsin-Madison

<sup>4</sup>University of Illinois at Urbana-Champaign

{bait0002, junzhao, jerry.zhu}@ntu.edu.sg, shoudonghan@hust.edu.cn, jiefeng@cs.wisc.edu, lbo@illinois.edu

## Abstract

Adversarial examples that can fool deep models are mainly crafted by adding small perturbations imperceptible to human eyes. There are various optimization-based methods in the literature to generate adversarial perturbations, most of which are time-consuming. AdvGAN, a method proposed by Xiao *et al.* in IJCAI 2018, employs Generative Adversarial Networks (GAN) to generate adversarial perturbation with original images as inputs, which is faster than optimization-based methods at inference time. ~~AdvGAN, however, fixes the target classes in the training and we find it difficult to train AdvGAN when it is modified to take original images and target classes as inputs.~~ In this paper, we propose Attack-Inspired GAN (AI-GAN) with a different training strategy to solve this problem. AI-GAN is a two-stage method, in which we use **projected gradient descent (PGD) attack** to inspire the training of GAN in the first stage and apply **standard training of GAN** in the second stage.

Once trained, the Generator can approximate the conditional distribution of adversarial instances and generate imperceptible adversarial perturbations given different target classes. We conduct experiments and evaluate the performance of AI-GAN on MNIST and CIFAR-10. Compared with AdvGAN, AI-GAN achieves higher attack success rates with similar perturbation magnitudes.

## 1 Introduction

Deep neural networks have been intensively applied in various scenarios, including those safety and security critical ones, such as autopilots, face recognition and video surveillance. Despite the mighty capability, recent studies have found that they are vulnerable to adversarial examples [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015]. The existence of adversarial examples has greatly challenged the reliability of deep learning models. An attack could be launched by crafting imperceptible perturbations on legiti-

mate samples, so that the generated adversarial examples can easily fool deep neural networks.

Many researchers have managed to generate such perturbations in different ways, such as box-constrained L-BFGS [Szegedy *et al.*, 2014], Fast Gradient Sign Method (FGSM) [Goodfellow *et al.*, 2015], Jacobian-based Saliency Map Attack (JSMA) [Papernot *et al.*, 2016] and C&W attack [Carlini and Wagner, 2017]. These attack methods are optimization-based with proper distance metrics  $L_0$ ,  $L_2$  and  $L_\infty$  to restrict the magnitudes of perturbations and make the presented adversarial examples visually natural. These methods, however, are usually time-consuming and need to access the target models at inference period.

In order to accelerate the generation process, some researchers start to employ generative models to produce adversarial perturbations [Xiao *et al.*, 2018; Poursaeed *et al.*, 2018; Jandial *et al.*, 2019], or generate adversarial examples directly [Song *et al.*, 2018; Liu and Hsieh, 2019]. AdvGAN proposed by [Xiao *et al.*, 2018] uses Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014] to generate adversarial perturbations. In AdvGAN, the generator produces adversarial perturbations while the discriminator determines whether generated adversarial examples are realistic. However, targeted classes are determined beforehand in AdvGAN when performing targeted attacks, which constrains the representation ability of GAN. [Liu and Hsieh, 2019] and [Song *et al.*, 2018] utilize GAN to generate adversarial examples for different target classes from latent vector. In their methods, GAN takes a random vector and target classes as inputs and generates adversarial examples without corresponding existing natural images. Such methods are not applicable when the original images are given. To attack a face recognition system, for example, searching corresponding latent vectors of targeted faces is unrealistic. In this paper, we argue that generating adversarial perturbations is more practical than searching in latent space for targeted attacks. And adversarial examples crafted by adding perturbations can supplement existing datasets, which is potentially beneficial to adversarial training [Madry *et al.*, 2018].

To solve the aforementioned problems, we propose a new variant of GAN to generate adversarial perturbations conditionally and efficiently, which is named Attack-Inspired GAN (AI-GAN). There are two stages in the training phase of AI-GAN: in the first stage, we jointly train a generator, a

\*Corresponding Author

discriminator, and an attacker. The attacker is used to inspire the generator to produce effective perturbations quickly in the joint training. The discriminator estimates the similarity between adversarial examples generated by the generator and the attacker. In the second stage, we remove the attacker and apply the standard GAN training strategy. The outputs of the generator are fine-tuned, since the discriminator now estimates the similarity between adversarial examples generated by the generator and original images. As for evaluation of AI-GAN, we first compare the training process of AI-GAN and the modified AdvGAN that takes original samples and target classes as inputs. Then we employ AI-GAN to perform attacks as well as AdvGAN in various experimental settings (semi-whitebox vs. black-box, under defenses or without defenses). From the experiments, we conclude that 1) our model architecture and training strategy are more reasonable than AdvGAN for generating adversarial examples, 2) AI-GAN achieves higher attack rates than AdvGAN under same  $L_\infty$  bound of perturbations in various experimental settings.

We summarize our contributions as follows:

1. Different from AdvGAN, we propose AI-GAN with a two-stage training strategy to generate adversarial examples, which takes less epochs to train for certain success rates (e.g. 90%) and high generation quality than AdvGAN.
2. Our experiments show that AI-GAN can produce perceptually realistic examples and achieve high attack success rates with different target classes on MNIST (> 97%) and CIFAR-10 (> 92%).
3. We apply AI-GAN on MNIST and CIFAR-10 with different target models and show that AI-GAN achieves higher attack success rates than AdvGAN in different experimental settings (semi-whitebox vs. black-box, under defenses vs. no defenses).

## 2 Related Work

### 2.1 Adversarial Examples

Adversarial examples, which are able to mislead deep neural networks, are first discovered by [Szegedy *et al.*, 2014]. They manage to maximize the network’s prediction error by adding hardly perceptible perturbations to benign images. Since then, various attack methods have been proposed. [Goodfellow *et al.*, 2015] developed Fast Gradient Sign Method (FGSM) to compute the perturbations efficiently using back-propagation. The perturbations could be expressed as  $\eta = \epsilon \text{sign}(\nabla_x J(\theta, x, y))$ , where  $J(\theta, x, y)$  represents the cross entropy loss function, and  $\epsilon$  is a constant. Thus adversarial examples are expressed as  $x_A = x + \eta$ . One intuitive extension of FGSM is Basic Iterative Method [Kurakin *et al.*, 2016] which executes FGSM many times with smaller  $\epsilon$ . [Papernot *et al.*, 2016] proposed Jacobian-based Saliency Map Attack (JSMA) with  $L_0$  distance. The saliency map discloses the likelihood of fooling the target network when modifying pixels in original images. Optimization-based methods have been proposed to generate quasi-imperceptible perturbations with constraints in different distance metrics. [Car-

lini and Wagner, 2017] designed a set of attack methods. The objective function is minimizing  $\|\delta\|_p + c \cdot f(x + \delta)$ , where  $c$  is a constant and  $p$  could be 0, 2 or  $\infty$ . [Chen *et al.*, 2017] also proposed an algorithm approximating the gradients of targeted models based on Zeroth Order Optimization (ZOO). [Madry *et al.*, 2018] introduced a convex optimization method called Projected Gradient Descent (PGD) to generate adversarial examples, which is proved to be the strongest one-order attack. However, such methods are usually time-consuming because the optimization process is slow, and they are only able to generate perturbations once a time.

### 2.2 Generative Adversarial Networks (GANs)

GAN is firstly proposed by [Goodfellow *et al.*, 2014] and has achieved great success in various tasks [Isola *et al.*, 2017; Zhu *et al.*, 2017; Reed *et al.*, 2016]. GAN consists of two competing neural networks with different objectives, called the discriminator  $D(x)$  and the generator  $G(z)$  respectively. The training phase could be seen as a min-max game of the discriminator and the generator. The generator is trained to generate fake images to fool the discriminator while the discriminator tries to classify the real images and the fake images. Usually the generator and the discriminator are trained alternatively while the other one is fixed. The original objective function is denoted as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1)$$

However, the training of vanilla GAN is very unstable. To stabilize the GAN training, a variety of improvements of GAN are developed such as DCGAN [Radford *et al.*, 2015], WGAN [Arjovsky *et al.*, 2017] and WGAN with gradient penalty [Gulrajani *et al.*, 2017]. In this paper, we use WGAN with gradient penalty.

### 2.3 Generating Adversarial Examples via Generative Models

Generative models are usually used to create new data because of their powerful representation ability. [Poursaeed *et al.*, 2018] firstly applied generative models to generate four types of adversarial perturbations (universal or image dependent, targeted or non-targeted) with U-Net [Ronneberger *et al.*, 2015] and ResNet [He *et al.*, 2016] architectures. [Xiao *et al.*, 2018] used the idea of GAN to make adversarial examples more realistic. Different from the above methods generating adversarial perturbations, some other methods generate adversarial examples directly, which are called unrestricted adversarial examples [Song *et al.*, 2018]. Song *et al.* [2018] proposed to search the latent space of pretrained ACGAN [Odena *et al.*, 2017] to find adversarial examples. Rob-GAN [Liu and Hsieh, 2019] combines adversarial training [Madry *et al.*, 2018] with GAN training to enhance Generator and Discriminator in the presence of adversarial attacks. Generating adversarial examples directly may extend datasets, but it is more challenging and less flexible than generating perturbations in some scenarios. Given specific target images in face recognition systems, for example, it would be time-consuming to

find the corresponding latent vector if the dimension of latent vector is large.

### 3 Our Approach

#### 3.1 Problem Definition

Consider a classification network  $f$  trained on dataset  $\mathcal{X} \subseteq \mathcal{R}^n$ , with  $n$  being the number of features. And suppose  $(x_i, y_i)$  is the  $i^{th}$  instance in the training data, where  $x_i \in \mathcal{X}$  is generated from some unknown distribution  $\mathcal{P}_{\text{data}}$ , and  $y_i \in \mathcal{Y}$  is the ground truth label. The classifier  $f$  is trained on natural images and achieves a high accuracy. The goal of an adversary is to generate an adversarial example  $x_a$ , which is able to let  $f$  output a wrong prediction and looks similar to  $x$  in terms of some distance metrics. We use  $L_\infty$  to bound the magnitude of perturbations. There are two types of such attacks: given an instance  $(x, y)$ , the adversary makes  $f(x_a) \neq y$ , which is called untargeted attack; or  $f(x_a) = t$  given a target class  $t$ , which is called targeted attack. Targeted attack is more challenging than untargeted attack, and we mainly focus on targeted attack in this paper.

#### 3.2 Proposed Framework

We propose a new variant of GAN with two training stages which is called Attack-Inspired GAN (AI-GAN) to generate adversarial examples conditionally and efficiently. As shown in Figure 1, the overall architectures of AI-GAN, consists of a generator  $G$ , a discriminator  $D$ , an attacker  $A$ , and a target classifier  $C$ . For comparison, the architecture of AdvGAN by [Xiao *et al.*, 2018] is illustrated in Figure 2.

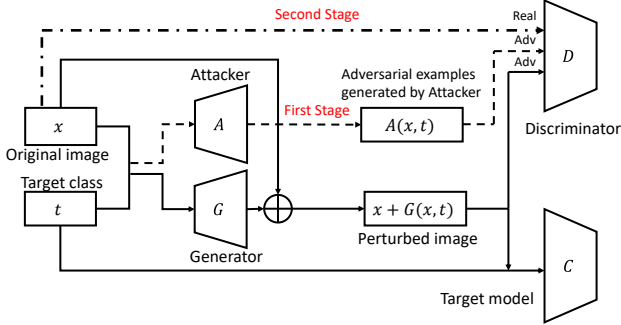


Figure 1: The architecture of our AI-GAN. We use the dashed line and dot-dashed line to indicate the data flow for Stage 1 and Stage 2 respectively. Solid lines are used in both stages.

There are two main differences between AI-GAN and AdvGAN according to their architectures. **First**, AdvGAN fixes target classes in training, which constrains the generator to produce only one-class targeted adversarial perturbations, while AI-GAN takes the original images and target classes as inputs, and can produce adversarial perturbations for different target classes. **Second**, we find simply embedding target classes with original images in AdvGAN is difficult to train, as discussed later Section 4.1. Thus we propose AI-GAN, a two-stage method, to solve this problem. We train GAN and an attacker jointly in the first stage to generate adversarial examples quickly and fine-tune it in the second stage.

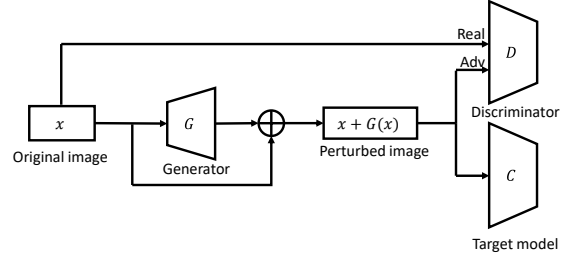


Figure 2: The architecture of AdvGAN by [Xiao *et al.*, 2018].

In AI-GAN, Generator  $G$  takes a clean image  $x$  and the target class label  $t$  as inputs to generate adversarial perturbations  $G(x, t)$ .  $t$  are sampled randomly. Then an adversarial example can be obtained with  $x + G(x, t)$  and sent to Discriminator  $D$ . There are two stages when training AI-GAN. In the first stage,  $D$  distinguishes the two types of adversarial examples  $x + G(x, t)$  and  $A(x, t)$ , which are generated by the Generator  $G$  and the Attacker  $A$  respectively. Target classes are the same for  $G$  and  $A$ . In the second stage, adversarial examples generated by  $A$  are substituted with original samples, and then standard GAN training is applied. The target classifier  $C$  takes  $x + G(x, t)$  as its input and outputs corresponding logits. The logits are used to calculate a loss  $\mathcal{L}_{\text{adv}}^C$ , which indicates the effectiveness of generated adversarial examples.

The objective function of our model is expressed as

$$\mathcal{L} = \mathcal{L}_{\text{adv}}^C + \lambda \mathcal{L}_{\text{GAN}} + \gamma \mathcal{L}_{\text{hinge}}, \quad (2)$$

where  $\lambda$  and  $\gamma$  are weights of different objectives. We explain  $\mathcal{L}_{\text{adv}}^C$ ,  $\mathcal{L}_{\text{GAN}}$ , and  $\mathcal{L}_{\text{hinge}}$  as follows.

$\mathcal{L}_{\text{adv}}^C$  is the loss of fooling target classifier  $C$  and is leveraged to generate adversarial examples, which is expressed as<sup>1</sup>

$$\mathcal{L}_{\text{adv}}^C = \mathbb{E}_x \ell_C(x + G(x, t)), \quad (3)$$

where  $t$  is the target class and  $\ell_C$  represents the loss function used when training target classifier  $C$ .

$\mathcal{L}_{\text{GAN}}$  is adversarial loss used in GAN's training [Goodfellow *et al.*, 2014], which is used to encourage generated samples similar to the original samples.

For the first stage,  $\mathcal{L}_{\text{GAN}}$  is expressed as

$$\mathcal{L}_{\text{GAN}}^1 = \mathbb{E}_x \log D(A(x, t)) + \mathbb{E}_x \log(1 - D(x + G(x, t))) \quad (4)$$

where  $A(x, t)$  is adversarial examples crafted by the attacker  $A$ .

For the second stage,  $\mathcal{L}_{\text{GAN}}$  is expressed as

$$\mathcal{L}_{\text{GAN}}^2 = \mathbb{E}_x \log D(x) + \mathbb{E}_x \log(1 - D(x + G(x, t))) \quad (5)$$

$\mathcal{L}_{\text{hinge}}$  is called Hinge loss to bound the magnitude of the perturbations, which is common in prior work [Carlini and Wagner, 2017; Xiao *et al.*, 2018].

$$\mathcal{L}_{\text{hinge}} = \mathbb{E}_x \max(0, \|G(x)\|_2 - c), \quad (6)$$

where  $c$  denotes a user-defined constant.

<sup>1</sup>We denote  $\mathbb{E}_x \equiv \mathbb{E}_{x \sim \mathcal{P}_{\text{data}}}$  for simplicity.

## 4 Experimental Results

In this section, we evaluate AI-GAN in two ways. **First**, we compare the training processes of AdvGAN and AI-GAN from the perspectives of attack success rates and generation qualities of adversarial examples. **Second**, we examine AdvGAN and AI-GAN in semi-whitebox [Xiao *et al.*, 2018] and black-box setting on MNIST [LeCun *et al.*, 2010] and CIFAR-10 [Krizhevsky, 2009], respectively. In black-box setting, we train substitute models based on Dynamic Distillation [Xiao *et al.*, 2018], which means distilled models are not static and jointly trained with GAN. With dynamic distillation, the generator can approximate how target models respond to adversarial examples. In all of our experiments, we constrain the perturbations for different attack methods under an  $L_\infty$  bound of 0.3 on MNIST and 8 on CIFAR-10.

**Implementation Details.** We adopt generator and discriminator architectures similar to those of [Zhu *et al.*, 2017] and use Projected Gradient Descent (PGD) [Madry *et al.*, 2018] as an attacker in the training process. Note that any optimization-based attack method in the literature can be used theoretically. We use Wasserstein Metric and Gradient Penalty proposed in [Gulrajani *et al.*, 2017] to stabilize the training of GAN. Stop epoch for the first stage and the iteration number of PGD are hyper-parameters, which are set to 10 and 40 respectively. We apply C&W loss [Carlini and Wagner, 2017] and set confidence  $\kappa = 0$  in our experiments. We use Adam as our solver [Kingma and Ba, 2014], with a batch size of 64 and learning rate of 0.001.

**Target Models in the Experiments.** We select two models as target models on different datasets. For MNIST, we use model A from [Tramèr *et al.*, 2018] and model B from [Carlini and Wagner, 2017], whose architectures are shown in Table 1. For CIFAR-10, we use ResNet-18 and ResNet-34 [He *et al.*, 2016]. These models are pretrained on natural data. Models A and B achieve 99% accuracy on test data while ResNet18 and ResNet34 achieve 93% accuracy<sup>2</sup>.

Model A	Model B
Conv(64, 5, 5)+ReLU	Conv(32, 3, 3)+ReLU
Conv(64, 5, 5)+ReLU	Conv(32, 3, 3)+ReLU
Dropout(0.25)	MaxPool(2, 2)
FullyConnected(128)+ReLU	Conv(64, 3, 3)+ReLU
Dropout(0.5)	Conv(64, 3, 3)+ReLU
FullyConnected + Softmax	MaxPool(2, 2)
	FullyConnected(200) + ReLU
	FullyConnected(200) + ReLU
	Softmax

Table 1: Model architectures used in this work for MNIST dataset.

### 4.1 Advantages of AI-GAN

As we all know, the training process of GAN in practice is usually unstable and time-consuming. In our method, an attacker is trained jointly with GAN to improve the training

process. We argue that our framework is non-trivial. AdvGAN learns only the distribution of adversarial examples for one target class. When performing attacks with different target classes, the generator is required to learn the conditional distribution of adversarial instances, which is more challenging. According to our experiments, it is difficult to train AdvGAN when simply embedding target classes and original images. In AI-GAN, we employ an attacker to perform attacks in first stage, which can inspire the generator and shrink the search space. We conduct the experiments of AI-GAN and AdvGAN on MNIST in semi-whitebox settings. Models A and B under no defenses are used as target models.

**First**, we test these two methods and visualize attack success rates during training in Figure 3. Note that we modify AdvGAN in this section so that it takes original images and target classes as inputs. As shown in Figure 3, we evaluate the attack success rates of two methods for 200 training epochs. We can see that AI-GAN needs less epochs than AdvGAN to satisfy a certain attack success rate (e.g. 90%). The attacker gives a correct direction to the generator to search for adversarial examples, which makes the generator produce adversarial examples quickly. As for training time, AI-GAN takes 300s and 37s per epoch on average in two stages while AdvGAN takes 35s per epoch on average. But more important, the first stage of AI-GAN plays a key role in increasing attack success rates as shown in Figure 3.

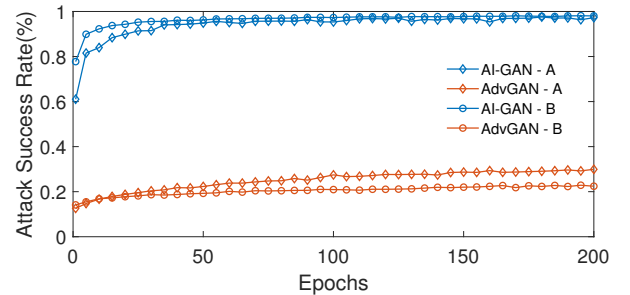


Figure 3: Attack success rates of training AdvGAN and AI-GAN on MNIST for 200 epochs under  $L_\infty$  bound of 0.3. Apparently AI-GAN achieves higher attack success rates than AdvGAN within same epochs. We use *method-target model* to identify corresponding lines.

**Second**, as shown in Figure 4, we sample some original images and adversarial examples to compare the generation quality of two methods. These samples are generated on MNIST after training 10 epochs (Stage 1) and 100 epochs (Stage 2), which are shown in second and third column in Figure 4. In Stage 1, it is hard to tell samples generated by which method are better in quality. But in Stage 2, perturbations generated by AI-GAN are visually smaller than that in Stage 1, while there is little progress of AdvGAN. Correspondingly, adversarial examples generated by AI-GAN are more realistic than that generated by AdvGAN after training 100 epochs.

In summary, Figure 3 and Figure 4 appeals that AI-GAN can produce adversarial examples with higher success rates

<sup>2</sup>Pretrained PyTorch models can be downloaded from <https://github.com/huynphan/PyTorch-CIFAR10>



and qualities than AdvGAN within same training epochs. The two-stage training and inspiration of the attacks applied in AI-GAN contribute to generate realistic adversarial examples quickly, which makes AI-GAN outperform AdvGAN.

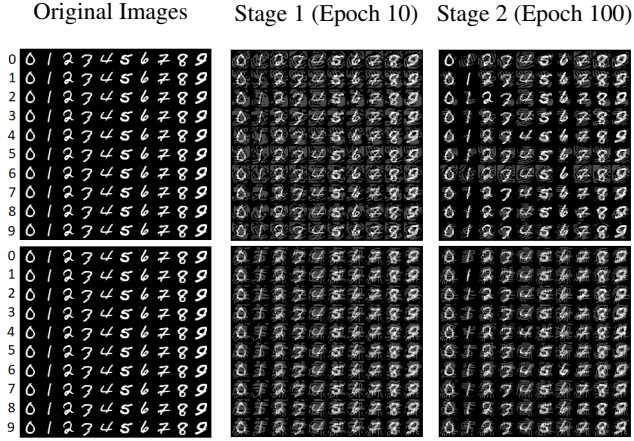


Figure 4: Original images and adversarial examples sampled in two stages from left to right. The first row is generated by AI-GAN and the second row is generated by AdvGAN. Original images are shown on the diagonal and the numbers on the left denote target classes for each row.

## 4.2 Attack in Semi-Whitebox Setting

We conduct attacks to classifiers with different architectures for MNIST and CIFAR-10 under different conditions: under defenses or no defenses.

### Attack without defenses

We apply AI-GAN to perform semi-whitebox attack against different models on MNIST and CIFAR-10. The performance of AI-GAN is shown in Table 2. From the table we can see that AI-GAN achieves high attack success rates with different target classes. For MNIST, the average attack success rates are 99.38% and 99.71% for model A and B. And the success rates are no less than 97% in all classes. For CIFAR-10, the average attack success rates are 96.07% and 96.86% for ResNet18 and ResNet34. The attack success rates for all classes are above 90%. Compared to AdvGAN, average attack success rates of AI-GAN are higher when against most models both on MNIST and CIFAR-10 as shown in Table 3. Note that once AI-GAN is trained, it could generate adversarial examples under different conditions, while we need to train 10 AdvGANs to attack with 10 different target classes.

Randomly selected adversarial examples generated are shown in Figure 5 and Figure 6. We make a comparison between adversarial examples generated by AI-GAN and AdvGAN on MNIST in Figure 5. In some characters like 0 and 8, AI-GAN is better than AdvGAN apparently. And Figure 6 shows adversarial examples generated by AI-GAN with different target models on CIFAR-10.

### Attack Under defenses

There are various defenses proposed against adversarial examples in the literature [Madry *et al.*, 2018;

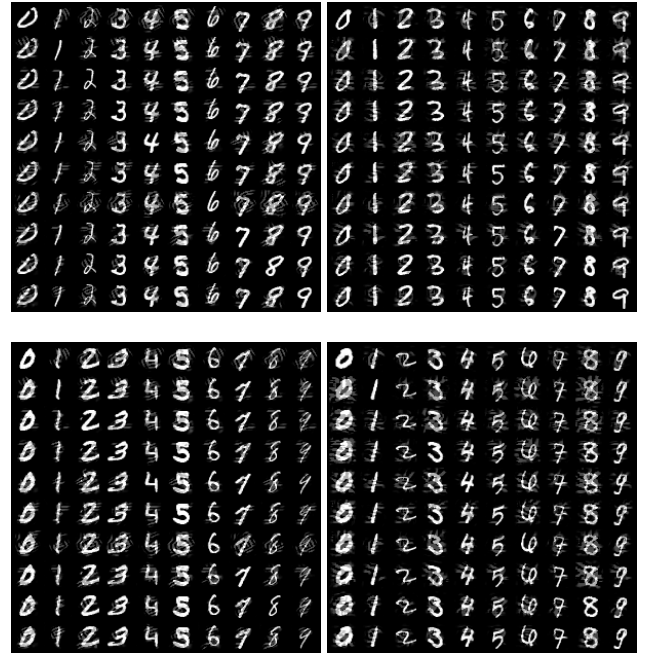


Figure 5: Adversarial examples generated by AI-GAN (Row 1) and AdvGAN (Row 2) for different models on MNIST with different target classes (same as Figure 3). From left to right, target models are model A and B for each column. Original images are shown on the diagonal.

Samangouei *et al.*, 2018], and adversarial training [Madry *et al.*, 2018] is widely accepted as the most effective way. For evaluation of adversarial examples generated by AI-GAN, we select three adversarial training methods to improve robustness of target models. The first adversarial training method is proposed by [Goodfellow *et al.*, 2015] based on FGSM. The objective function is expressed as  $\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha)J(\theta, x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)))$ . The second method is Ensemble Adversarial Training extended from the first method by [Tramèr *et al.*, 2018], and we use two different models as static models to generate adversarial examples on each dataset. And the third method is proposed by [Madry *et al.*, 2018], which employs PGD as a universal first-order adversary.

We evaluate the performance of AI-GAN and AdvGAN quantitatively under these defense methods. As shown in Table 4, AI-GAN achieves higher attack success rate than AdvGAN when target models are trained with defenses.

## 4.3 Attack in Black-box Setting

In this section, we use AI-GAN to perform attacks in black-box setting. We assume the adversary is not aware of the defense strategies but could get the outputs of target models. Since the black-box attack is based on dynamic distillation strategy, a distilled model  $f_d$  is constructed in our experiments. And when applying dynamic distillation during training, local model  $f_d$  is updated every time after updating the generator. The advantage of dynamic distillation is the generator can approximate the predictions of target models when

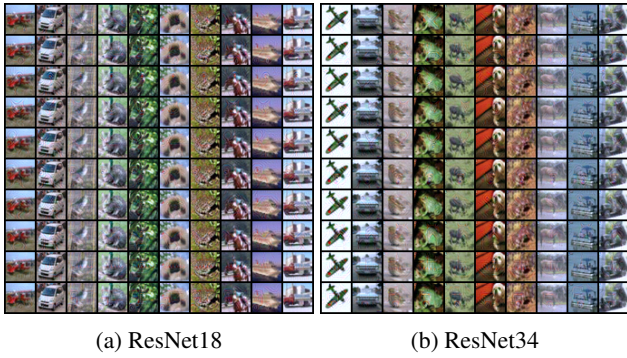


Figure 6: Adversarial examples generated by AI-GAN for different models on CIFAR-10 with different targets. Original images are shown on the diagonal.

Target Class	MNIST(%)		CIFAR-10(%)	
	Model A	Model B	ResNet18	ResNet34
Class 1	98.89	99.09	97.39	98.01
Class 2	99.54	99.25	95.92	98.40
Class 3	99.50	99.89	96.52	90.32
Class 4	99.90	99.84	92.96	96.32
Class 5	99.67	99.96	93.04	95.28
Class 6	99.94	99.74	98.73	99.31
Class 7	99.84	99.83	98.66	97.17
Class 8	99.94	99.83	97.21	96.59
Class 9	97.52	99.88	96.97	99.54
Class 10	99.10	99.74	93.30	97.69
Average	98.67	99.18	95.65	97.68

Table 2: Attack success rates of adversarial examples generated by AI-GAN against different models on MNIST and CIFAR-10 in semi-whitebox setting.

fed with adversarial examples and learn to generate more aggressive adversarial examples. According to the experiments in [Xiao *et al.*, 2018], dynamic distillation helps AdvGAN to achieve attack success rates(e.g. 93.4%) in black-box setting as high as in semi-whitebox setting, when there is no defense of target models. Here we only consider the situation where the defenses of target models are applied in black-box setting, which is more challenging.

We select LeNet-5 [LeCun *et al.*, 1998] and a shallow ResNet as the local model architectures for MNIST and CIFAR-10 respectively, while Model B and ResNet34 are target models under three different defenses. Other experimental settings are same as that in semi-whitebox setting.

Methods	MNIST(%)		CIFAR-10(%)	
	Model A	Model B	ResNet18	ResNet34
AdvGAN	98.67	99.18	95.65	<b>97.68</b>
AI-GAN	<b>99.38</b>	<b>99.71</b>	<b>96.07</b>	96.86

Table 3: Comparison of attack success rate of adversarial examples generated by AI-GAN and AdvGAN in semi-whitebox setting.

Dataset	Model	defenses	AdvGAN	AI-GAN
MNIST	A	Adv.	9.0%	<b>42.6%</b>
		Ens.	10.2%	<b>45.5%</b>
		Iter.Adv	17.7%	<b>40.0%</b>
	B	Adv.	13.4%	<b>45.7%</b>
		Ens.	12.1%	<b>33.4%</b>
		Iter.Adv	10.7%	<b>24.5%</b>
CIFAR10	ResNet18	Adv.	14.6%	<b>37.1%</b>
		Ens.	29.3%	<b>44.8%</b>
		Iter.Adv	17.6%	<b>17.7%</b>
	ResNet34	Adv.	15.6%	<b>45.5%</b>
		Ens.	24.6%	<b>38.8%</b>
		Iter.Adv	15.0%	<b>20.5%</b>

Table 4: Comparison of attack success rate of adversarial examples generated by AI-GAN and AdvGAN in semi-whitebox setting with defenses

We apply AI-GAN and AdvGAN to generate adversarial examples on MNIST and CIFAR-10. The attack success rates for different defense strategies are shown in Table 5. On MNIST, AI-GAN achieves higher attack success rates than AdvGAN does under all three defense strategies. This means adversarial examples generated by AI-GAN have a great transferability. On CIFAR-10, AI-GAN also shows stronger attack abilities in most cases.

Defense	MNIST		CIFAR-10	
	AdvGAN	AI-GAN	AdvGAN	AI-GAN
Adv.	15.1%	<b>18.2%</b>	14.4%	<b>15.2%</b>
Ens.	17.3%	<b>24.7%</b>	15.3%	<b>18.2%</b>
Iter.Adv	11.4%	<b>32.9%</b>	<b>25.1%</b>	22.2%

Table 5: Comparison of attack success rate of adversarial examples generated by AI-GAN and AdvGAN in black-box setting.

## 5 Conclusion

In this paper, we propose AI-GAN to generate adversarial examples where GAN is jointly trained with an attacker. There are two stages in our AI-GAN framework: in the first stage, the attacker plays as a guide of the generator and inspires the training of generator; in the second stage, the attacker is removed and the generator is encouraged to generate adversarial examples similar to original samples. Once our AI-GAN is trained, the generator can produce imperceptible perturbations given different samples and target classes, and preserve high attack success rates as well. Similarly to AdvGAN, AI-GAN can perform attacks in both semi-whitebox and black-box settings without knowledge of defenses in place. And AI-GAN achieves higher attack success rates than AdvGAN under same bound of perturbations.

## References

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint*

- arXiv:1701.07875*, 2017.
- [Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *SP*, pages 39–57, 2017.
- [Chen *et al.*, 2017] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *AISec*, pages 15–26, 2017.
- [Goodfellow *et al.*, 2014] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NIPS*, pages 2672–2680, 2014.
- [Goodfellow *et al.*, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015.
- [Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *NIPS*, pages 5767–5777, 2017.
- [He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.
- [Isola *et al.*, 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image Translation with Conditional Adversarial Networks. In *CVPR*, pages 1125–1134, 2017.
- [Jandial *et al.*, 2019] Surgan Jandial, Puneet Mangla, Sakshi Varshney, and Vineeth Balasubramanian. Advgan++: Harnessing latent layers for adversary generation. In *ICCV Workshops*, 2019.
- [Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [Krizhevsky, 2009] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.
- [Kurakin *et al.*, 2016] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. *CoRR*, abs/1607.0, 2016.
- [LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LeCun *et al.*, 2010] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST Handwritten Digit Database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [Liu and Hsieh, 2019] Xuanqing Liu and Cho-Jui Hsieh. Rob-GAN: Generator, Discriminator, and Adversarial Attacker. In *CVPR*, 2019.
- [Madry *et al.*, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*, 2018.
- [Odena *et al.*, 2017] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *ICML*, pages 2642–2651. JMLR.org, 2017.
- [Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *EuroS&P*, pages 372–387. IEEE, 2016.
- [Poursaeed *et al.*, 2018] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge J Belongie. Generative Adversarial Perturbations. In *CVPR*, pages 4422–4431. {IEEE} Computer Society, 2018.
- [Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [Reed *et al.*, 2016] Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text to Image Synthesis. *arXiv preprint arXiv:1605.05396*, 2016.
- [Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, pages 234–241. Springer, 2015.
- [Samangouei *et al.*, 2018] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense- $\{\text{GAN}\}$ : Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *ICLR*, 2018.
- [Song *et al.*, 2018] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing Unrestricted Adversarial Examples with Generative Models. In *NIPS*, pages 8312–8323. 2018.
- [Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014.
- [Tramèr *et al.*, 2018] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J Goodfellow, Dan Boneh, and Patrick D McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *ICLR*, 2018.
- [Xiao *et al.*, 2018] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. In *IJCAI*, 2018.
- [Xie *et al.*, 2019] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature Denoising for Improving Adversarial Robustness. In *CVPR*, pages 501–509, 2019.
- [Zhu *et al.*, 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-image Translation using Cycle-Consistent Adversarial networks. In *ICCV*, pages 2223–2232, 2017.