GENERATING ADVERSARIAL EXAMPLES WITH ADVERSARIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently deep neural networks (DNNs) have been found to be vulnerable against adversarial examples resulting from adding small-magnitude perturbations to inputs. Such adversarial examples can mislead DNNs to produce adversary-selected results. Different attack strategies have been proposed to generate adversarial examples, but how to produce them more efficiently and guarantee the diversity of adversarial perturbations requires more research efforts. In this paper, we propose AdvGAN to generate adversarial examples with generative adversarial networks (GANs), which can learn and preserve the distribution of original instances. For AdvGAN, once the generator is trained, it can generate an adversarial perturbation efficiently for any instance, and the generated adversarial examples have large variety depending on the underlying distribution, so as to accelerate adversarial training as defenses. We apply AdvGAN in both semi-whitebox and black-box attack settings. In semi-whitebox attacks, there is no need to access the original target model after the generator is trained, in contrast to traditional white-box attacks. In the black-box attack, we dynamically train a distilled model for the black-box and optimize the generator accordingly. Extensive experimental results show that black-box attacks based on AdvGAN can achieve comparable attack success rate with that of semi-whitebox settings. Adversarial examples generated by AdvGAN on different models have high attack success rate under state-of-the-art defenses compared with other attacks. We have achieved 92.76% accuracy on the MNIST black-box challenge and been ranked at the top position.

1 INTRODUCTION

Deep Neural Networks (DNNs) have been adopted with great success in a variety of applications ranging from image recognition (Krizhevsky et al., 2012; He et al., 2016) to speech processing (Hinton et al., 2012), from robotics training (Levine et al., 2016) to medical diagnostics (Ciresan et al., 2012). However, recent work has demonstrated that DNNs are vulnerable to adversarial perturbations (Goodfellow et al., 2015; Szegedy et al., 2013). An adversary can add small-magnitude perturbations to inputs and generate adversarial examples to mislead DNNs. Such maliciously perturbed instances can cause the learning system to misclassify them into either a maliciously-chosen target class (i.e. targeted attack) or classes that are different from the ground truth (i.e. untargeted attack). Different algorithms have been proposed for generating such adversarial examples, such as fast gradient method (FGSM) (Goodfellow et al., 2015) and optimization-based method (Carlini & Wagner, 2017a; Liu et al., 2017).

Facing different types of attack strategies, various defenses have been provided as well. Among them, different types of adversarial training methods are the most effective. Goodfellow et al. (2015) first propose adversarial training as an effective way to improve the robustness of DNNs, and Tramèr et al. (2017a) extends it to ensemble adversarial learning. Mądry et al. (2017) have also proposed robust networks against adversarial examples based on well-defined adversaries (Mądry et al., 2017). To conduct an adversarial training based defense, a large number of adversarial examples are required. In addition, these adversarial examples can make the defense more robust if they come from different models as suggested by the ensemble training. The benefit of ensemble adversarial training is to increase the diversity of adversarial examples so that the model can fully explore the adversarial example space.

Current algorithms for generating adversarial examples are limited either by the generation speed or instance diversity. In this paper, we propose to train a feed-forward network to generate adversarial perturbation, which can produce adversarial examples immediately once the network is trained. We apply generative adversarial networks (GANs) (Goodfellow et al., 2014) to produce adversarial examples in both the semi-whitebox and black-box settings. The main advantage of GANs is that the instances generated based on it will follow the certain underlying distribution but can vary in different ways. Therefore, the GAN based network can increase the diversity of these adversarial examples. In addition, as GANs are capable of producing high-quality images (Isola et al., 2017; Gulrajani et al., 2017), we are also able to produce perceptually realistic adversarial instances. We name our method as AdvGAN.

Note that in the previous white-box attacks, such as fast gradient sign method and optimization method, the adversary needs to have white-box access to the architecture and parameters of the model all the time. However, by deploying AdvGAN, once the feed-forward network is trained, it can instantly produce adversarial perturbations for any input instances without requiring access to the model itself anymore. We name this attack setting *semi-whitebox*.

To evaluate the effectiveness of our attack strategy AdvGAN, we first generate adversarial instances based on AdvGAN and other attack strategies on different models. We then apply the state-of-the-art defenses to defend against these generated adversarial examples. We evaluate these attack strategies in both semi-whitebox and black-box settings. We show that adversarial examples generated by AdvGAN can achieve high attack success rate, potentially due to the fact that these adversarial instances appear with high diversity compared to other recent attack strategies.

Our contributions are listed as follows.

- Different from the previous optimization-based methods, we train a generative adversarial network to directly produce adversarial examples, which are both perceptually realistic and achieve state-of-the-art attack success rate.
- Because we train an end-to-end network from the input images to adversarial examples, we produce the adversarial examples much faster than state-of-the-art attack strategies. This property can significantly accelerate adversarial training for defense.
- We show that AdvGAN can attack black-box models by training a distilled model. We propose to dynamically train the distilled model with query information and achieve high black-box attack success rate and targeted black-box attack, which is difficult to achieve for transferability based black-box attacks.
- We evaluate the attack effectiveness by generating adversarial examples on different models based on AdvGAN and other attack strategies. We use the state-of-the-art attack methods to defend against these examples and show that AdvGAN can increase the diversity of adversarial perturbation and achieve much higher attack success rate under current defenses.
- We apply AdvGAN on the MNIST challenge (Mądry et al., 2017) and achieve 88.93% accuracy on the published robust model in the semi-whitebox setting, and 92.76% in the black-box setting, which wins the top position in the challenge [1].

2 Related Work

Different methods have been proposed to generate adversarial examples. We will give a brief introduction to these attack strategies as well as generative adversarial networks (GANs).

Adversarial Examples A number of attack strategies to generate adversarial examples have been proposed in the white-box setting, where the adversary has full access to the classifier (Szegedy et al., 2014; Goodfellow et al., 2015; Carlini & Wagner, 2017a; Moosavi-Dezfooli et al., 2015; Papernot et al., 2016; Biggio et al., 2013; Kurakin et al., 2016). Goodfellow et al. (2015) propose the fast gradient method that applies a first-order approximation of the loss function to construct adversarial samples. Optimization based method has also been proposed to optimize adversarial perturbation for targeted attack based on certain constraints (Carlini & Wagner, 2017a; Liu et al., 2017). However, the optimization process is slow and can only optimize perturbation for one specific instance

[1] https://github.com/MadryLab/mnist_challenge

each time. In contrast, our feed-forward method can train a network to produce perturbation for any instance. It performs much faster than current algorithms of generating adversarial examples and also achieves higher attack success rate against different defenses due to its high diversity. . Independent to our work, feed forward network has been applied to generate adversarial perturbation (Baluja & Fischer, 2017). However, this work only applies L_2 norm distance as one part of the loss function, aiming to guarantee that the generated adversarial instance is indistinguishable to the original one. As GAN is a natural way to learn the underlying distribution of instances and produce indistinguishable data, we propose to add GAN as an additional part of the loss and improve the added adversarial perturbation.

Black-box Attacks Current learning systems usually do not allow white-box accesses against the model for security reasons. Therefore, there is a great need for black-box attacks analysis. Such attacks do not require query access to the model. Most of the black-box attack strategies are based on the transferability phenomenon (Papernot et al., 2017). An adversary can train a local model first, and generate adversarial examples against this local model. Thus, the same adversarial examples will also be able to attack other models. Lots of learning systems allow query accesses to the model. However, there is little work that can leverage query-based access to target models to construct adversarial samples and move beyond transferability. Papernot et al. (2017) proposed to train a local surrogate model with queries to the target model to generate adversarial samples, but this strategy still relies on transferability. In contrast, we show that the proposed AdvGAN can perform black-box attacks without depending on transferability.

Generative Adversarial Networks Goodfellow et al. (2014); Radford et al. (2015) have achieved visually appealing results in both image generation (Berthelot et al., 2017) and manipulation (Zhu et al., 2016) settings. Recently, image-to-image conditional GANs have further improved the quality of the synthesis results (Isola et al., 2017; Zhu et al., 2017). We adopt the similar adversarial loss and image-to-image network architecture to learn the mapping from an original image to a perturbed output such that the perturbed image cannot be distinguished from real images in the original class. Different from prior work, we aim to produce output results that are not only visually realistic but also able to mislead the learning system.

3 Generating Adversarial Examples with Adversarial Networks (AdvGAN)

3.1 Problem Definition

Let $X \subseteq \mathcal{R}^n$ be the feature space, with n the number of features. Suppose that $(\mathbf{x}_i, y_i)$ is the ith instance within the training set, which is comprised of feature vectors $\mathbf{x}_i \in X$ generated according to some unknown distribution $\mathbf{x}_i \sim \mathcal{P}_{\text{data}}(x)$ and y_i the corresponding true class labels. The learning system aims to learn a classifier $f_\theta : \mathcal{X} \to \mathcal{Y}$ from the domain $\mathcal{X}$ to the set of classification outputs $\mathcal{Y}$ (e.g., $\mathcal{Y} \in \{0, 1\}$ for binary classification). $|\mathcal{Y}|$ denotes the number of possible classification outputs. Given an instance $\mathbf{x}$, the goal of an adversary is to generate adversarial example $\mathbf{x}_A$, which is classified as $f(\mathbf{x}_A) \neq y$ (untargeted attack), or $f(\mathbf{x}_A) = t$ (targeted attack) where t is the target class, where $\mathbf{x}_A$ should be close to the original instance $\mathbf{x}$ in terms of l_2 or other distance matrix.

3.2 AdvGAN Framework

The overall architecture of AdvGAN is shown in Figure 1. AdvGAN mainly consists of three parts: a generator $\mathcal{G}$, a discriminator $\mathcal{D}$, and the target neural network f. Here the generator $\mathcal{G}$ takes the original instance x as the input and generates perturbation $\mathcal{G}(x)$. Then $x + \mathcal{G}(x)$ will be sent to the discriminator $\mathcal{D}$, which is used to distinguish the generated data and the original instance x. The goal of $\mathcal{D}$ is to guarantee that the generated instance is indistinguishable with the data from its original class. To fulfill the goal of fooling a learning model, we first perform the white-box attack, where the target model is f in this case. f takes $x + \mathcal{G}(x)$ as input, and minimizes its loss function $\mathcal{L}_{adv}$, 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).

The adversarial loss can be written as

$$\mathcal{L}_{GAN} = \mathbb{E}_{x \sim \mathcal{P}_{\text{data}}(x)} \log \mathcal{D}(x) + \mathbb{E}_{x \sim \mathcal{P}_{\text{data}}(x)} \log(1 - \mathcal{D}(x + \mathcal{G}(x))). \tag{1}$$

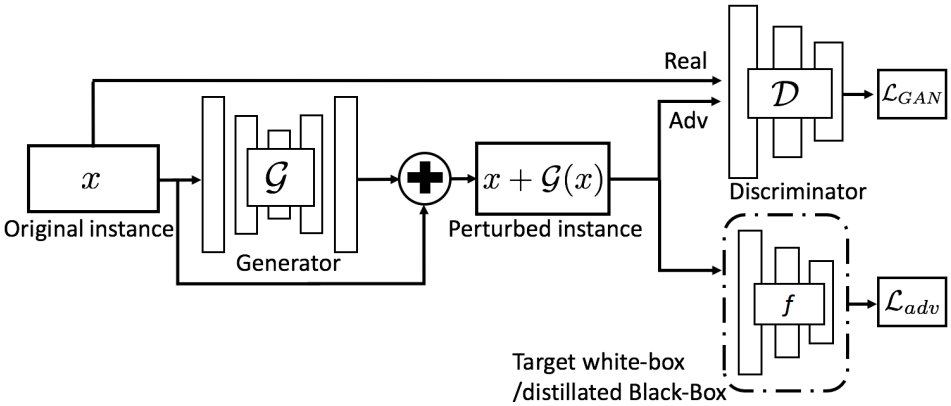


Figure 1: Overview of AdvGAN

Here, the discriminator $\mathcal{D}$ aims to distinguish the perturbed data $x + \mathcal{G}(x)$ from the original data x.

The loss for fooling the target model f in a targeted attack is:[2]

$$\mathcal{L}_{adv}^{f} = \mathbb{E}_x \ell_f(x + \mathcal{G}(x), t), \tag{2}$$

where t is the target class and ℓ denotes the loss function (e.g., cross-entropy loss) used to train the original model f. The $\mathcal{L}_{adv}$ loss encourages the perturbed image to be misclassified as target class t. Here we can also perform the untargeted attack by maximizing the distance between the prediction and the ground truth, so we will focus on the targeted attack in the rest of the paper.

To bound the magnitude of the perturbation, which is a common practice in prior work (Carlini & Wagner, 2017a; Liu et al., 2017; Bartlett & Wegkamp, 2008), we add a soft hinge loss on the L_2 norm $\mathcal{L}_{hinge} = \mathbb{E}_x \max(0, \|\mathcal{G}(x)\|_2 - c)$, where c denotes the bound. This can also stabilize the GAN's training, as shown in (Isola et al., 2017).

Finally, our full objective can be expressed as

$$\mathcal{L} = \mathcal{L}_{adv}^{f} + \alpha \mathcal{L}_{GAN} + \beta \mathcal{L}_{hinge}, \tag{3}$$

where α and β control the relative importance of different objectives. Note that $\mathcal{L}_{GAN}$ here is used to guarantee the perturbed data to appear similar with the original data x, while $\mathcal{L}_{adv}$ encourages to generate adversarial examples, optimizing for the attack success rate. To increase the diversity, we apply instance normalization and dropout during training the generator, which has been adopted in other GAN-related work as well (Yi et al., 2017). We obtain our $\mathcal{G}$ and $\mathcal{D}$ by solving the minmax game $\arg\min_G \max_D \mathcal{L}$.

3.3 Black-box Attacks with Adversarial Networks

Static distillation For black-box attack, we assume adversaries have no prior knowledge of training data or the model itself. In our experiments in section 4, we randomly draw data that is disjoint from the training data of the black-box model to distill it, since we assume the adversaries have no prior knowledge about the training data or the model. To achieve black-box attacks, one way is to first build a distilled network f based on the output of the black-box model b (Hinton et al., 2015). Once we obtain the distilled network f, we carry out the same attack strategy as described in the white-block setting (See Equation (3)). Here, we minimize the following network distillation objective:

$$\arg\min_f \mathbb{E}_x \mathcal{H}(f(x), b(x)), \tag{4}$$

where $f(x)$ and $b(x)$ denote the output from the distilled model and black-box model respectively for the given training image x, and $\mathcal{H}$ denotes the commonly used cross-entropy loss. By optimizing the objective over all the training images, we can obtain a model f which behaves very close to the black-box model b. We then carry out the attack on the distilled network.

[2]We omit the $x \sim \mathcal{P}_{\text{data}}(x)$ for simplicity

Note that unlike training the discriminator $\mathcal{D}$, where we only use the real data from the original class to guarantee that the generated instance is close to its original class, here we train the distilled model with data from all classes to avoid the problem of imbalanced learning.

Dynamic distillation Only training the distilled model with all the pristine training data is not enough, since it is unclear how close the black-box and distilled model perform on the generated adversarial examples, which have not appeared in the training set before. Here we propose an *alternative minimization* approach to dynamically make queries and train the distilled model f and our generator $\mathcal{G}$ jointly. We perform the following two steps in each iteration. During iteration i:

1. **Update $\mathcal{G}_i$ given a fixed network** f_{i-1}: We follow the white-box setting (See Equation 3) and train the generator and discriminator based on a previously distilled model f_{i-1}. We initialize the weights $\mathcal{G}_i$ as $\mathcal{G}_{i-1}$.

$$\mathcal{G}_i, D_i = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{adv}^{f_{i-1}} + \alpha \mathcal{L}_{GAN} + \beta \mathcal{L}_{hinge} \tag{5}$$

2. **Update f_i given a fixed generator** $\mathcal{G}_i$: First, we use f_{i-1} to initialize f_i. Then, given the generated adversarial examples from $\mathcal{G}_i$, the distilled model f_i will be updated based on the set of new query results for the generated adversarial examples against the black-box model, as well as the original training images.

$$f_i = \arg\min_{f} \mathbb{E}_x \mathcal{H}(f(x), b(x)) + \mathbb{E}_x \mathcal{H}(f(\mathcal{G}_i(x)), b(\mathcal{G}_i(x))), \tag{6}$$

   where we use both the original images x and the newly generated adversarial examples $\mathcal{G}_i(x)$ to update f.

In the experiment section, we compare the performance of both the static and dynamic distilled models and observe that simultaneously updating $\mathcal{G}$ and f produces higher attack performance. (See Table 2).

4 Experimental Results

In this section, we first evaluate AdvGAN for both the semi-whitebox and black-box settings on MNIST(LeCun & Cortes, 1998) and CIFAR-10 (Krizhevsky et al., 2014). We also perform semi-whitebox attack on the ImageNet dataset (Russakovsky et al., 2015). We then apply AdvGAN to generate adversarial examples on different models and test the attack success rate for them under the state-of-the-art defenses and show that our method can achieve higher attack success rate compared to other existing attack strategies. We generate all adversarial examples for different methods based on the L_∞ bound as 0.3 on MNIST, and 8 on CIFAR-10 for fair comparison.

In general, as shown in Table 1, AdvGAN has several advantages over other white-box and black-box attacks. For instance, regarding computation efficiency, AdvGAN performs much faster than others even including the efficient FGSM, although AdvGAN needs extra training time to train the generator. All these strategies can perform targeted attack except transferability based attack, although the ensemble strategy can help to improve. Besides, FGSM and optimization methods can only perform white-box attack, while AdvGAN is able to attack in semi-whitebox setting.

Table 1: Comparison with the state-of-the-art methods. Run time is measured for generating 1000 adversarial instances during test time.

	FGSM	Opt.	Trans.	AdvGAN
Run-time	0.06s	>3h	-	<0.01s
Targeted Attack	✓	✓	Ens.	✓
Black-box Attack			✓	✓

Implementation details: Our code and models will be available upon publication. We adopt the similar architecture from image-to-image translation literature (Isola et al., 2017). In particular, we adopt the architecture of generator $\mathcal{G}$ from Johnson et al. (2016), and we apply a

CNN as our discriminator $\mathcal{D}$. We apply the loss in (Carlini & Wagner, 2017c) as our $\mathcal{L}_{adv}^{f} = \max(\max_{i \neq t} g(\mathbf{x}_A)_i - g(\mathbf{x}_A)_t, 0)$, where t is the target class.We set the confidence $\kappa = 0$ for both Opt. and AdvGAN for fair comparison. We use Adam as our solver (Kingma & Ba, 2014). We use a batch size of 128 and a learning rate of 0.001. For GANs training, we use the least squares objective proposed by LSGAN (Mao et al., 2016), as it has been shown to produce better results with more stable training.

Table 2: Accuracy of different models on pristine data, and the attack success rate of adversarial examples generated against different models by AdvGAN on MNIST and CIFAR-10. p: pristine test data; w: semi-whitebox attack; b-D: black-box attack with dynamic distillation strategy; b-S: black-box attack with static distillation strategy.

	MNIST			CIFAR-10	
Model	A	B	C	ResNet	Wide-ResNet
Accuracy (p)	98.97%	99.17%	99.09%	92.41%	95.01%
Attack Success Rate (w)	97.9%	97.1%	98.3%	94.71%	99.30%
Attack Success Rate (b-D)	93.4%	90.1%	94.02%	78.47 %	81.81%
Attack Success Rate (b-S)	30.7%	66.63%	87.3%	10.3%	13.3%

Models used in the experiments For MNIST, in all of our experiments, we generate adversarial examples for three models whose architectures are shown in Appendix A. Models A and B are used in (Tramèr et al., 2017b), which represent different architectures. Model C is the target network architecture used in (Carlini & Wagner, 2017a), which provides the optimization based attack strategy. Table 2 presents the accuracy of pristine MNIST data on each model. For CIFAR-10, we select ResNet-32 and wide ResNet-34 (He et al., 2016; Zagoruyko & Komodakis, 2016) for our experiments. In detail, we use 32-layers ResNet implemented by tensorflow [3] and Wide resnent derived from the variant of "w32-10 wide"[4].

We show the classification accuracy of pristine MNIST and CIFAR-10 test data (p), and attack success rate of adversarial examples generated by AdvGAN on different models in Table 2.

4.1 AdvGAN in semi-whitebox Setting

First, we replace the target model f with different models as listed in Table 6 for MNIST and with ResNet and wide ResNet for CIFAR-10. We first apply AdvGAN to perform a white-box attack against each model on the MNIST dataset. From the performance of semi-whitebox attack (Attack Rate (w)) in Table 2, we can see that it is easy for AdvGAN to generate adversarial instances to attack all models with high attack success rate. Figure 5b shows adversarial examples randomly selected for targeted attacks.

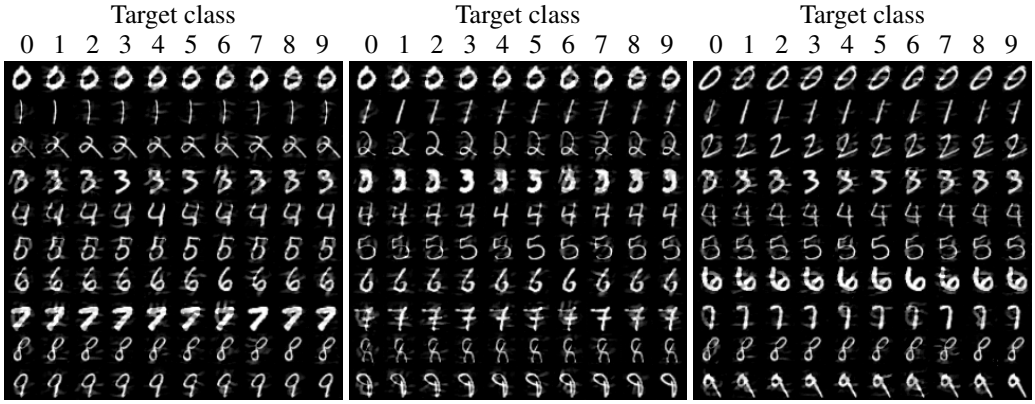
We also generate adversarial examples from the same original instance x, targeting at other different classes, as shown in Figures 2a–2c. In the white-box setting on MNIST, we can see that the generated adversarial examples for different models appear very close to the ground truth (lying on the diagonal of the matrix). These adversarial examples generated by AdvGAN can successfully fool the corresponding model and therefore be misclassified as the target class shown on the top. We also randomly select the adversarial examples for different targeted attacks, which is shown in the supplemental materaial due to the space limitation.

In addition, we analyze the attack success rate based on different loss functions on MNIST. Under the same magnitude of perturbation, if we replace the loss function in Equation (3) with $\mathcal{L} = \mathcal{L}_2 + \mathcal{L}_{adv}$, which is a similar architecture to Baluja & Fischer (2017), the attack success rate is about 86.2%. If we replace the loss function with $\mathcal{L} = \mathcal{L}_{hinge} + \mathcal{L}_{adv}$ the attack success rate is 91.1%, compared with that of AdvGAN as 98.3%.

Similarly, on CIFAR-10, we apply the same white-box attack for ResNet and Wide ResNet based on AdvGAN, and some adversarial examples are shown in Figure 3a. It is obvious that the adversarial examples generated by AdvGAN here are perceptually realistic. We show adversarial examples for

[3] https://github.com/tensorflow/models/blob/master/research/ResNet/ResNet_model.py

[4] https://github.com/MadryLab/cifar10_challenge/blob/master/model.py



(a) Model A (b) Model B (c) Model C

Figure 2: Adversarial examples generated from the same original image to different targets by AdvGAN on MNIST with semi-whitebox attack. On the diagonal, the original image is shown.

(a) semi-whitebox setting (b) Black-box setting

Figure 3: Adversarial examples generated by AdvGAN on CIFAR-10. The same image is perturbed to different classes. On the diagonal, the original image is shown.

the same original instance targeting different other classes. It is clear that with different targets, the adversarial examples keep similar quality compared with the pristine instances on the diagonal.

We also apply AdvGAN to generate adversarial examples on the ImageNet as shown in Figure 4. The added perturbation is unnoticeable while all the adversarial instances are misclassified into other target classes with high confidence.

4.2 ADVGAN IN BLACK-BOX SETTING

In this section, we show the black-box attack performance of AdvGAN. Our black-box attack here is based on the dynamic distillation model. We construct a substitution model to distill model f. For control test, we select Model C's architecture as our substitution model. First, we randomly select a subset of instances disjointed from the training data of AdvGAN, therefore we do not assume the adversaries have any prior knowledge of the training data or the model. Based on the dynamic distillation strategy, the adversarial examples by AdvGAN achieve an attack success rate above 90% for MNIST and 80% for CIFAR-10. as shown in Table 2, these randomly selected instances

Figure 4: Adversarial examples generated by AdvGAN on Imagenet in the semi-whitebox setting, which are classified as (from left to right) poodle, ambulance, basketball, and electric guitar

can achieve high perceptual quality as the original digit is somewhat highlighted by the adversarial perturbation, which implies a type of perceptually realistic manipulation.

We apply AdvGAN to generate adversarial examples for the same instance targeting different classes. When targeting different classes, the adversarial perturbation actually does not vary too much. In addition, by comparing with the pristine instances on the diagonal, we can see that the adversarial instances are of high perceptually quality. Figure 3b shows the similar results for adversarial examples generated on CIFAR-10. (More variations of the results are included in the Appendix.)

4.3 Attack Effectiveness Under Defenses

Given the fact that AdvGAN strives to generate adversarial instances from the underlying true data distribution, it can essentially introduce more diverse adversarial perturbations compared with other attack strategies. Thus, AdvGAN could have a higher chance to produce adversarial examples that are resilient under different defense methods. In this section, we quantitatively evaluate this property for AdvGAN compared with other attack strategies.

Threat Model. As shown in the literature, most of current defense strategies are not robust when attacking against them (Carlini & Wagner, 2017b; He et al., 2017). Here we consider a weaker threat model, where the adversary is not aware of the defenses and directly tries to attack the original learning model, which is also the first threat model analyzed in Carlini & Wagner (2017b). In this case, if an adversary can still successfully attack the model, it implies the robustness of the attack strategy. Under this setting, we first apply different attack methods to generate adversarial examples based on the original model without being aware of any defense. Then we apply different defenses to directly defend against these adversarial instances.

semi-whitebox Attack. First, we consider the semi-whitebox attack setting, where the adversary can have white-box access to the model architecture as well as the parameters. Here, we replace f in Figure 1 with our model A, B, and C, respectively. As a result, adversarial examples will be generated against different models. We use three adversarial training defenes to train different models for each model architecture: standard FGSM adversarial training (Adv.) (Goodfellow et al., 2015), ensemble adversarial training (Ensemble) (Tramèr et al., 2017b), and iterative training (Iter. Adv.) (Mądry et al., 2017).[5] We evaluate the effectiveness of these attacks against these defended

[5] Each ensemble adversarially trained model is trained using (i) pristine training data, (ii) FGSM adversarial examples generated for the current model under training, and (iii) FGSM adversarial examples generated for

models. In Table 3, we show that the attack success rate of adversarial examples generated by AdvGAN on different models is higher than those of FGSM and optimization methods.

Table 3: Attack success rate of adversarial examples generated by different semi-whitebox or white-box strategies on model A, B, and C under defenses on MNIST and ResNet and wide-ResNet on CIFAR-10.

Data	Model	Defense	FGSM	Opt.	AdvGAN
MNIST	A	Adv.	4.3%	4.6%	8.0%
		Ensemble	1.6%	4.2%	6.3%
		Iter.Adv.	4.4%	2.96%	5.6%
	B	Adv.	6.0%	4.5%	7.2%
		Ensemble	2.7%	3.18%	5.8%
		Iter.Adv.	9.0%	3.0%	6.6%
	C	Adv.	2.7%	2.95%	18.7%
		Ensemble	1.6%	2.2%	13.5%
		Iter.Adv.	1.6%	1.9%	12.6%
CIFAR	ResNet	Adv.	13.10%	11.9%	16.03%
		Ensemble.	10.00%	10.3%	14.32%
		Iter.Adv	22.8%	21.4%	29.47%
	Wide ResNet	Adv.	5.04%	7.61%	14.26%
		Ensemble	4.65%	8.43%	13.94 %
		Iter.Adv.	14.9%	13.90%	20.75%

Table 4: Attack success rate of adversarial examples generated by different black-box adversarial strategies under defenses on MNIST and CIFAR-10

	MNIST			CIFAR-10		
Defense	FGSM	Opt.	AdvGAN	FGSM	Opt.	AdvGAN
Adv.	3.1%	3.5%	11.5%	13.58%	10.8%	15.96
Ensemble	2.5%	3.4%	10.3%	10.49%	9.6%	12.47%
Iterative Adv.	2.4%	2.5%	12.2%	22.96%	21.70x%	24.28%

Table 5: Accuracy of the MadryLab public model under different attacks in white-box setting. The AdvGAN here achieved the best performance.

Method	Accuracy (xent loss)	Accuracy (cw loss)
FGSM	95.23%	96.29%
PGD	93.66%	93.79%
Opt	-	91.69%
AdvGAN	-	**88.93%**

Black-box Attack. In the black-box attacks, for comparison we use FGSM and Optimization methods to attack model A as the target model and generate adversarial examples on MNIST. Based on transferability, we use these adversarial examples to test on model B and report the corresponding classification accuracy. For AdvGAN, we use model B as the black-box model and train a distilled model to perform black-box attack against model B and report the attack success rate in Table 4. We can see that the adversarial examples generated by the black-box AdvGAN always achieve much higher attack success rate compared with other attack methods. This implies that AdvGAN may be able to generate adversarial instances with higher diversity.For CIFAR-10. We use ResNet as black-box model and train a distilled model to perform black-box attack against ResNet. For the optimization method and FGSM, we leverage transferability and use the adversarial examples generated by attacking Wide ResNet to test on ResNet to perform blackbox attack.

naturally trained models of two architectures different from the architecture of the model under training and from each other.

In addition, we apply AdvGAN to the MNIST challenge.[6] Among all the methods, for white-box attack we achieve 88.93% accuracy on the published local model as shown in Table 5. For the reported black-box attack, we achieved the accuracy as 92.76%, outperforming all other state-of-the-art attack strategies.

5 CONCLUSION

In this paper, we propose AdvGAN to generate adversarial examples using generative adversarial networks (GANs). In our AdvGAN framework, once trained, the feed-forward generator can produce adversarial perturbations efficiently. It can also perform both semi-whitebox and black-box attacks with high attack success rate. In addition, when we apply AdvGAN to generate adversarial instances on different models without knowledge of the defenses in place, the generated adversarial examples can attack the state-of-the-art defenses with higher attack success rate than examples generated by the competing methods. This property makes AdvGAN a promising candidate for accelerating adversarial training defense methods. Benefiting from GANs, AdvGAN can produce adversarial examples with high diversity and potentially explore the adversarial subspace more thoroughly. The generated adversarial examples produced by AdvGAN also preserve high perceptual quality due to GANs' distribution approximation property.

REFERENCES

Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

Peter L Bartlett and Marten H Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(Aug):1823–1840, 2008.

David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 387–402. Springer, 2013.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy, 2017*, 2017a.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017c.

Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pp. 2843–2851, 2012.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pp. 2672–2680, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

[6] `https://github.com/MadryLab/mnist_challenge`

Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv preprint arXiv:1706.04701*, 2017.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pp. 694–711. Springer, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Yann LeCun and Corrina Cortes. The MNIST database of handwritten digits. 1998.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17(39):1–40, 2016.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083 [cs, stat]*, June 2017.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387. IEEE, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017a.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017b.

Zili Yi, Hao Zhang, Ping Tan Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint arXiv:1704.02510*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, pp. 597–613. Springer, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.

A ARCHITECTURE OF MODELS

Table 6: Model architectures for the MNIST

A	B	C
Conv(64,5,5)+Relu	Conv(64,8,8)+Relu	Conv(32,3,3)+Relu
Conv(64,5,5)+Relu	Dropout(0.2)	Conv(32,3,3)+Relu
Dropout(0.25)	Conv(128, 6, 6)+Relu	MaxPooling(2,2)
FC(128)+Relu	Conv(128, 5, 5)+Relu	Conv(64,3,3)+Relu
Dropout(0.5)	Dropout(0.5)	Conv(64,3,3)+Relu
FC(10)+Softmax	FC(10)+Softma	MaxPooling(2,2)
		FC(200)+Relu
		FC(200)+Softmax

B NETWORK ARCHITECTURES

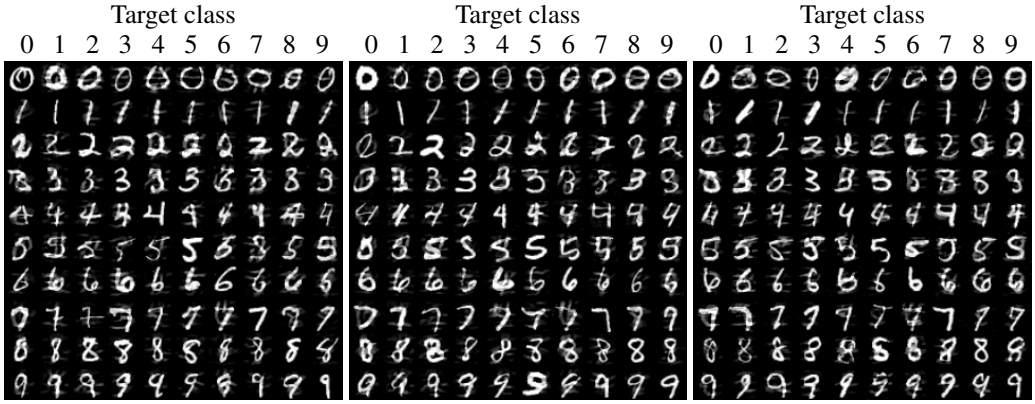
Generator architecture We follow the naming rules used in Johnson et al. (2016)'s Github repository[7] as well as Zhu et al. (2017) . Let `c3s1-k` denotes 3×3 Convolution-InstanceNorm-ReLU layer with k filter and stride 1. `Rk` means residual block that contains two 3×3 convolution layers with the same numbers of filters. `dk` denotes the 3×3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2.

The generator structures consists of:
`c3s1-8,d16, d32, r32 ,r32 ,r32,r32, u16, u8, c3s1-3`

Discriminator architecture We use CNNs as our discriminator network (Radford et al., 2015). Let `Ck` denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last conv layer, we apply a `FC` layer to produce a 1 dimensional output. We do not use InstanceNorm for the first `C8` layer. We use leaky ReLUs with slope 0.2.

The discriminator architecture is:
`C8, C16, C32, FC`

[7] `https://github.com/jcjohnson/fast-neural-style`.



(a) Model A (b) Model B (c) Model C

Figure 5: Adversarial examples generated by AdvGAN on MNIST against different models in the white-box setting. Each attack instance here is random sampled.

(a) Model A (b) Model B

(c) Model C

Figure 6: Adversarial examples generated by AdvGAN on MNIST corresponding to different models in the black-box setting. Each attack instance is randomly sampled.

Figure 5 shows additional adversarial examples generated by AdvGAN for various MNIST models in a white-box setting.

Similarly, Figure 6 shows additional adversarial examples generated in a black-box setting.

Figure 7 shows additional adversarial examples generated for CIFAR-10 models.

(a) White-box setting (b) Black-box setting

Figure 7: Adversarial examples generated by AdvGAN on CIFAR-10. Each attack instance is random sampled.