APPENDIX

In the appendix, we provide additional related work on gradient-based adversarial attack methods, adversarial training methods and typical generative adversarial nets. Then we describe how to obtain the original generator and provide theoretical analysis, as well as experimental details and additional results. In the end, we visualize the examples generated by original GAN and AT-GAN.

A ADDITIONAL RELATED WORK

A.1 GRADIENT-BASED ATTACKS

Numerous adversarial attacks have been proposed in recent years (Carlini & Wagner, 2017; Liu et al., 2017; Bhagoji et al., 2017; Li et al., 2019). In this part, we will introduce three typical adversarial attack methods. Here the components of all adversarial examples are clipped in $[0, 1]$.

Fast Gradient Sign Method (FGSM). FGSM (Goodfellow et al., 2015) adds perturbation in the gradient direction of the training loss J on the input x to generate adversarial examples.

$$x_{adv} = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y_{true})),$$

where y_{true} is the true label of a sample x, θ is the model parameter and ϵ specifies the ℓ_∞ distortion between x and x_{adv}.

Projected Gradient Descent (PGD). PGD adversary (Madry et al., 2018) is a multi-step variant of FGSM, which applies FGSM for k iterations with a budget α.

$$x_{adv_{t+1}} = \mathbf{clip}(x_{adv_t} + \alpha sign(\nabla_x J(\theta, x_{adv_t}, y_{true})), x_{adv_t} - \epsilon, x_{adv_t} + \epsilon)$$
$$x_{adv_0} = x, \quad x_{adv} = x_{adv_k}$$

Here $\mathbf{clip}(x', p, q)$ forces its input x' to reside in the range of $[p, q]$.

Rand FGSM (R+FGSM). R+FGSM (Tramèr et al., 2018) first applies a small random perturbation on the benign image with a parameter α ($\alpha < \epsilon$), then it uses FGSM to generate an adversarial example based on the perturbed image.

$$x_{adv} = x' + (\epsilon - \alpha) \cdot sign(\nabla_{x'} J(\theta, x', y_{true})) \quad \text{where } x' = x + \alpha \cdot sign(\mathcal{N}(\mathbf{0}, \mathbf{I})).$$

A.2 ADVERSARIAL TRAINING

There are many defense strategies, such as detecting adversarial perturbations (Metzen et al., 2017), obfuscating gradients (Buckman et al., 2018; Guo et al., 2018) and eliminating perturbations (Shen et al., 2017; Liao et al., 2018), among which adversarial training is the most effective method (Athalye et al., 2018). We list several adversarial training methods as follows.

Adversarial training. Goodfellow et al. (2015) first introduce the method of adversarial training, where the standard loss function f for a neural network is modified as:

$$\tilde{J}(\theta, x, y_{true}) = \alpha J_f(\theta, x, y_{true}) + (1 - \alpha) J_f(\theta, x_{adv}, y_{true}).$$

Here y_{true} is the true label of a sample x and θ is the model's parameter. The modified objective is to make the neural network more robust by penalizing it to count for adversarial samples. During the training, the adversarial samples are calculated with respect to the current status of the network. Taking FGSM for example, the loss function could be written as:

$$\tilde{J}(\theta, x, y_{true}) = \alpha J_f(\theta, x, y_{true}) + (1 - \alpha) J_f(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y_{true})), y_{true}).$$

Ensemble adversarial training. Tramèr et al. (2018) propose an ensemble adversarial training method, in which DNN is trained with adversarial examples transferred from a number of fixed pre-trained models.

Iterative adversarial training. Madry et al. (2018) propose to train a DNN with adversarial examples generated by iterative methods such as PGD.

A.3 GENERATIVE ADVERSARIAL NET

Generative Adversarial Net (GAN) (Goodfellow et al., 2014) consists of two neural networks, G and D, trained in opposition to each other. The generator G is optimized to estimate the data distribution and the discriminator D aims to distinguish fake samples from G and real samples from the training data. The objective of D and G can be formalized as a min-max value function $V(G, D)$:

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

Deep Convolutional Generative Adversarial Net (DCGAN) (Radford et al., 2016) is the convolutional version of GAN, which implements GAN with convolutional networks and stabilizes the training process. Auxiliary Classifier GAN (AC-GAN) (Odena et al., 2017) is another variant that extends GAN with some conditions by an extra classifier C. The objective function of AC-GAN can be formalized as follows:

$$\begin{aligned}\min_G \max_D \min_C V(G, D, C) =& \mathbb{E}_{x\sim p_x}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z, y_s)))] \\ &+ \mathbb{E}_{x\sim p_x}[\log(1 - C(x, y_s))] + \mathbb{E}_{z\sim p_z}[\log(1 - C(G(z, y_s), y_s))].\end{aligned}$$

To make GAN more trainable in practice, Arjovsky et al. (2017) propose Wasserstein GAN (WGAN) that uses Wassertein distance so that the loss function has more desirable properties. Gulrajani et al. (2017) introduce WGAN with gradient penalty (WGAN_GP) that outperforms WGAN in practice. Its objective function is formulated as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_x}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))] - \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2],$$

where $p_{\hat{x}}$ is uniformly sampled along straight lines between pairs of points sampled from the data distribution p_x and the generator distribution p_g.

B TRAINING THE ORIGINAL GENERATOR

Figure 2 (a) illustrates the overall architecture of AC-WGAN_GP that we used as the normal GAN. AC-WGAN_GP is the combination of AC-GAN (Odena et al., 2017) and WGAN_GP (Gulrajani et al., 2017), composed by three neural networks: a generator G, a discriminator D and a classifier f. The generator G takes a random noise z and a source label y_s as the inputs and generates an image $G(z, y_s)$. It aims to generate an image $G(z, y_s)$ that is indistinguishable to discriminator D and makes the classifier f to output label y_s. The loss function of G can be formulated as:

$$L_G = \mathbb{E}_{z\sim p_z(z)}[H(f(G(z, y_s)), y_s)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z, y_s))].$$

Here $H(a, b)$ is the entropy between a and b. The discriminator D takes the training data x or the generated data $G(z, y_s)$ as the input and tries to distinguish them. The loss function of D with gradient penalty for samples $\hat{x} \sim p_{\hat{x}}$ can be formulated as:

$$L_D = -\mathbb{E}_{x\sim p_{data}(x)}[D(x)] + \mathbb{E}_{z\sim p_z(z)}[D(G(z, y_s))] + \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}(\hat{x})}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2].$$

The classifier f takes the training data x or the generated data $G(z, y_s)$ as the input and predicts the corresponding label. The loss function is:

$$L_f = \mathbb{E}_{x\sim p_{data}(x)}[H(f(x), y_{true})] + \mathbb{E}_{z\sim p_z(z)}[H(f(G(z, y_s)), y_s)].$$

Different from AC-WGAN_GP, styleGAN2-ada (Karras et al., 2020a) trains styleGAN2 (Karras et al., 2020b) with adaptive discriminator augmentation. We obtain the network and weights from Karras et al. (2020a).

C THEORETICAL ANALYSIS OF AT-GAN

In this section, we provide proofs for theorems in Section 3.3.

Theorem 1. Suppose $\max_{z,y} L_2 < \epsilon$, we have $KL(p_a \| p_g) \to 0$ when $\epsilon \to 0$.

Proof. We first consider that for a distribution $p(x)$ in space $\mathcal{X}$, we construct another distribution $q(x)$ by selecting points $p_\epsilon(x)$ in the ϵ-neighborhood of $p(x)$ for any $x \in \mathcal{X}$. Obviously, when $p_\epsilon(x)$ is close enough to $p(x)$, $q(x)$ has almost the same distribution as $p(x)$. Formally, we have the following lemma.

Lemma 1. Given two distributions P and Q with probability density function $p(x)$ and $q(x)$ in space $\mathcal{X}$, if there exists a constant ϵ that satisfies $\|q(x) - p(x)\| < \epsilon$ for any $x \in \mathcal{X}$, we could get $KL(P\|Q) \to 0$ when $\epsilon \to 0$.

Proof. For two distributions P and Q with probability density function $p(x)$ and $q(x)$, we could get $q(x) = p(x) + r(x)$ where $\|r(x)\| < \epsilon$.

$$
\begin{aligned}
KL(P\|Q) &= \int p(x) \log \frac{p(x)}{q(x)} dx \\
&= \int p(x) \log p(x) dx - \int p(x) \log q(x) dx \\
&= \int (q(x) - r(x)) \log p(x) dx - \int (q(x) - r(x)) \log q(x) dx \\
&= \int q(x) \log p(x) dx - \int q(x) \log q(x) dx - \int r(x) \log p(x) dx + \int r(x) \log q(x) dx \\
&= \int r(x) \log \frac{q(x)}{p(x)} dx - KL(Q\|P) \\
&\leq \int \epsilon \log(1 + \frac{\epsilon}{p(x)}) dx
\end{aligned}
$$

Obviously, when $\epsilon \to 0$, we could get $\int \epsilon \log(1 + \frac{\epsilon}{p(x)}) dx \to 0$, which means $DL(P\|Q) \to 0$. □

Now, we get back to Theorem 1. For two distributions p_a and p_g, $\max_{y,z} L_2 < \epsilon$ indicates $\forall z \sim p_z, \|p_a(z, \cdot) - p_g(z, \cdot)\| < \epsilon$. According to Lemma 1, we have $KL(p_a\|p_g) \to 0$ when $\epsilon \to 0$. This concludes the proof. □

Theorem 2. The global minimum of the virtual training of AC-WGAN_GP is achieved if and only if $p_g = p_{data}$.

Proof. To simplify the analysis, we choose a category y of AC-WGAN_GP and denote $p_g(x|y)$ and $p_{data}(x|y)$ the distribution that the generator learns and the distribution of real data respectively. Then for each category, the loss function is equivalent to WGAN_GP. We refers to Samangouei et al. (2018) to prove this property. The WGAN_GP min-max loss is given by:

$$
\begin{aligned}
&\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))] - \lambda \mathbb{E}_{\hat{x} \sim p_{\hat{x}}(\hat{x})}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \\
&= \int_x p_{data}(x) D(x) dx - \int_z p_z(z) D(G(z)) dz - \lambda \int_{\hat{x}} p_{\hat{x}}(\hat{x}) [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] d\hat{x} \\
&= \int_x [p_{data}(x) - p_g(x)] D(x) dx - \lambda \int_{\hat{x}} p_{\hat{x}}(\hat{x}) [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] d\hat{x}
\end{aligned}
\tag{5}
$$

For a fixed G, the optimal discriminator D that maximizes $V(D, G)$ should be:

$$
D_G^*(x) = \begin{cases} 1 & \text{if } p_{data}(x) \geq p_g(x) \\ 0 & \text{otherwise} \end{cases} \tag{6}
$$

According to equation 5 and equation 6, we could get:

$$
\begin{aligned}
V(D, G) &= \int_x [p_{data}(x) - p_g(x)] D(x) dx - \lambda \int_{\hat{x}} p_{\hat{x}}(\hat{x}) [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] d\hat{x} \\
&= \int_{\{x | p_{data}(x) \geq p_g(x)\}} (p_{data}(x) - p_g(x)) dx - \lambda \int_{\hat{x}} p_{\hat{x}}(\hat{x}) d\hat{x} \\
&= \int_{\{x | p_{data}(x) \geq p_g(x)\}} (p_{data}(x) - p_g(x)) dx - \lambda
\end{aligned}
\tag{7}
$$

Let $\mathcal{X} = \{x|p_{data}(x) \geq p_g(x)\}$, in order to minimize equation 7, we set $p_{data}(x) = p_g(x)$ for any $x \in \mathcal{X}$. Then, since both p_g and p_{data} integrate to 1, we could get:

$$\int_{\mathcal{X}^c} p_g(x)dx = \int_{\mathcal{X}^c} p_{data}(x)dx.$$

However, this contradicts equation 6 where $p_{data}(x) < p_g(x)$ for $x \in \mathcal{X}^c$, unless $\mu(\mathcal{X}^c) = 0$ where μ is the Lebesgue measure.

Therefore, for each category we have $p_g(x|y) = p_{data}(x|y)$, which means $p_g(x) = p_{data}(x)$ for AC-WGAN_GP. □

D Additional Details on Experiments

In this section, we provide more details on experimental setup, report results on transferability, do ablation study on hyper-parameters, investigate the generating capacity by human evaluation, and show details for another implementation of AT-GAN on CIFAR-10 dataset. In the end, we illustrate some non-constrained adversarial examples generated by AT-GAN on MNIST, Fashion-MNIST and CelebA for the target attack.

D.1 More Experimental Setup

We first provide more details on the experimental setup, including the model architectures and attack hyper-parameters.

Model Architectures for AT-GAN. We first describe the neural network architectures used for AT-GAN in experiments. The abbreviations for components in the network are described in Table 4. The architecture of AC-WGAN_GP for MNIST and Fashion-MNIST is shown in Table 5 where the generator and discriminator are the same as in Chen et al. (2016), while the architecture of AC_WGAN_GP for CelebA is the same as in Gulrajani et al. (2017) and the architecture of styleGAN2-ada for CIFAR-10 is the same as in Karras et al. (2020a).

Hyper-parameters for Attacks. The hyper-parameters used in experiments for each attack method are described in Table 6 for MNIST, Fashion-MNIST and CelebA datasets. For CIFAR-10 dataset, we set $\epsilon = 0.03$ for FGSM, $\epsilon = 0.03$, $\alpha = 0.0075$ and epochs$= 20$ for PGD, $\alpha = 3$, $\beta = 2$ and epochs$= 1,000$ for AT-GAN.

Table 4: Abbreviations for network architectures.

Abbreviation	Description
Conv(m, $k \times k$)	A convolutional layer with m filters and filter size k
DeConv(m, $k \times k$)	A transposed convolutional layer with m filters and filter size k
Dropout(α)	A dropout layer with probability α
FC(m)	A fully connected layer with m outputs
Sigmoid	The sigmoid activation function
Relu	The Rectified Linear Unit activation function
LeakyRelu(α)	The Leaky version of a Rectified Linear Unit with parameter α
Maxpool(k,s)	The maxpooling with filter size k and stride s

D.2 Transferability of AT-GAN

Another important issue for adversarial examples is the transferability across different models. To demonstrate the transferability of non-constrained adversarial examples, we use adversarial examples generated by attacking Model A (MNIST and Fashion-MNIST) and CNN (CelebA), to evaluate the attack success rates on Model C (MNIST and Fashion-MNIST) and VGG16 (CelebA). As shown in Table 7, non-constrained adversarial examples generated by AT-GAN exhibit moderate transferability.

Table 5: Architecture of WGAN_GP with auxiliary classifier for MNIST and Fashion-MNIST.

Generator	Discriminator	Classifier
FC(1024) + Relu	Conv(64, 4×4) + LeakyRelu(0.2)	Conv(32, 3×3) + Relu
FC($7 \times 7 \times 128$) + Relu	Conv(128, 4×4) + LeakyRelu(0.2)	pooling(2, 2)
DeConv(64, 4×4) + Sigmoid	FC(1024) + LeakyRelu(0.2)	Conv(64, 3×3) + Relu
DeConv(1, 4×4) + Sigmoid	FC(1) + Sigmoid	pooling(2, 2)
FC(1024)		
Dropout(0.4)		
FC(10) + Softmax		

Table 6: Hyper-parameters of different attack methods on MNIST, Fashion-MNIST and CelebA.

Attack	Datasets			Norm
	MNIST	Fashion-MNIST	CelebA	
FGSM	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.015$	ℓ_∞
PGD	$\epsilon = 0.3$, $\alpha = 0.075$, epochs $= 20$	$\epsilon = 0.1$, $\alpha = 0.01$, epochs $= 20$	$\epsilon = 0.015$, $\alpha = 0.005$, epochs $= 20$	ℓ_∞
R+FGSM	$\epsilon = 0.3$, $\alpha = 0.15$	$\epsilon = 0.2$, $\alpha = 0.1$	$\epsilon = 0.015$, $\alpha = 0.003$	ℓ_∞
Song's	$\lambda_1 = 100$, $\lambda_2 = 0$, epochs $= 200$	$\lambda_1 = 100$, $\lambda_2 = 0$, epochs $= 200$	$\lambda_1 = 100$, $\lambda_2 = 100$, epochs $= 200$	N/A
AT-GAN	$\alpha = 2$, $\beta = 1$, epochs $= 100$	$\alpha = 2$, $\beta = 1$, epochs $= 100$	$\alpha = 3$, $\beta = 2$, epochs $= 200$	N/A

Table 7: Transferability of non-constrained adversarial examples and other search-based adversarial examples on three datasets. For MNIST and Fashion-MNIST, we attack Model C with adversarial examples generated on Model A. For CelebA dataset, we attack VGG16 using adversarial examples generated on CNN. Numbers represent the attack success rate (%).

	MNIST				Fashion-MNIST				CelebA			
	Nor.	Adv.	Ens.	Iter. Adv.	Nor.	Adv.	Ens.	Iter. Adv.	Nor.	Adv.	Ens.	Iter. Adv.
FGSM	46.7	4.2	1.7	4.6	68.9	23.1	20.8	14.8	15.6	4.3	3.3	4.1
PGD	**97.5**	6.5	4.1	4.1	**84.7**	27.6	**39.6**	14.6	18.3	4.3	3.1	4.1
R+FGSM	82.3	6.7	4.8	4.1	21.2	32.1	17.5	26.3	11.0	4.0	3.3	3.8
Song's	23.8	20.8	20.6	**20.1**	39.2	**34.0**	31.5	**30.3**	9.6	**31.8**	**21.5**	**38.8**
AT-GAN	65.3	**24.6**	**27.9**	17.2	58.0	22.7	32.0	15.2	**63.7**	15.4	16.5	17.6

D.3 Ablation Study

In this subsection, we investigate the impact of using different ρ in the loss function. As ρ could be constrained by both ℓ_0 and ℓ_∞ norm, we test various bounds, using Model A on MNIST dataset, for ρ in ℓ_0 and ℓ_∞, respectively.

We first fix $\|\rho\|_\infty = 0.5$ and try various values for $\|\rho\|_0$, i.e. 0, 100, 200, 300, 400 (the maximum possible value is 784 for 28*28 input). The attack success rates are in Table 8. We can observe that different values of $\|\rho\|_0$ only have a little impact on the attack success rates, and the performances are very close for $\|\rho\|_0$ = 0, 100, 200. Figure 5 further illustrates some generated adversarial examples, among which we can see that there exist some slight differences on the examples. When $\|\rho\|_0 = 0$, AT-GAN tends to change the foreground (body) of the digits. When we increase the value of $\|\rho\|_0$ (100 and 200), AT-GAN is more likely to add tiny noise to the background and the crafted examples are more realistic to humans (for instance, smoother on digit 4). But if we continue to increase $\|\rho\|_0$ (300 or 400), AT-GAN tends to add more noise and the quality of the generated examples decays. To have a good tradeoff on attack performance and generation quality, we set $\|\rho\|_0 = 200$.

Table 8: Attack success rate (ASR, %) of AT-GAN with various values for $\|\rho\|_0$ using Model A on MNIST dataset.

$\|\rho\|_0$	0	100	200	300	400
ASR	98.9	98.8	98.7	96.7	95.8

We then fix $\|\rho\|_0 = 200$ and test different values for $\|\rho\|_\infty$, i.e. 0, 0.1, 0.2, 0.3, 0.4, 0.5 (the maximum possible value is 1). The attack success rates are in Table 9. We can observe that different values of

Figure 5: The adversarial examples generated by AT-GAN for various values of $\|p\|_0$.

$\|\rho\|_\infty$ have very little impact on the attack performance. Figure 6 further illustrates some generated adversarial examples, among which we can see that a little bit more noises are added for bigger $\|\rho\|_\infty$ but the differences are very tiny when $\|\rho\|_\infty = 0.2$ to 0.5. So we simply set $\|\rho\|_\infty = 0.5$ in experiments, but other values of $\|\rho\|_\infty$ (0.2, 0.3, 0.4) also work.

Table 9: Attack success rate (ASR, %) of AT-GAN with various values for $\|\rho\|_\infty$ using Model A on MNIST dataset.

$\|\rho\|_\infty$	0	0.1	0.2	0.3	0.4	0.5
ASR	98.9	99.2	98.9	98.9	98.9	98.7

Figure 6: The adversarial examples generated by AT-GAN for various values of $\|p\|_\infty$.

D.4 Human Evaluation

To investigate the generating capacity of AT-GAN, we use the same input, and randomly pick 100 images for each category of MNIST generated by AT-GAN and the original generator, respectively. We then conduct human evaluation to determine whether each example is realistic. The evaluation results are in Table 10. We see that adversarial examples in some categories (e.g. 2, 4) are harder to be semantically meaningful than other categories (e.g. 0, 1). On average, however, the generating capability is close to that of the original generator.

Table 10: The evaluation results on the percentage of realistic images by human evaluation.

Category	0	1	2	3	4	5	6	7	8	9	Average
Original	100.0	100.0	93.0	94.0	98.0	96.0	99.0	100.0	98.0	100.0	97.8
AT-GAN	100.0	100.0	85.0	91.0	80.0	90.0	97.0	98.0	92.0	100.0	93.3

D.5 AT-GAN on CIFAR-10 Dataset

To further demonstrate the flexibility of AT-GAN, we implement AT-GAN on CIFAR-10 dataset using StyleGAN2-ada (Karras et al., 2020a), a recently proposed conditional GAN. The target classifier is wide ResNet w32-10 (Zagoruyko & Komodakis, 2016) by normal training (Nor.) and Iterative adversarial training (Iter.). The attack success rates are in Table 11. On normally trained models, PGD achieves the attack success rate of 100% while AT-GAN achieves the attack success rate of 93.5%. However, the adversarially trained model exhibits little robustness against AT-GAN and AT-GAN achieves attack success rate of 73.0%. In Figure 7, we illustrate some generated adversarial examples on CIFAR-10 dataset.

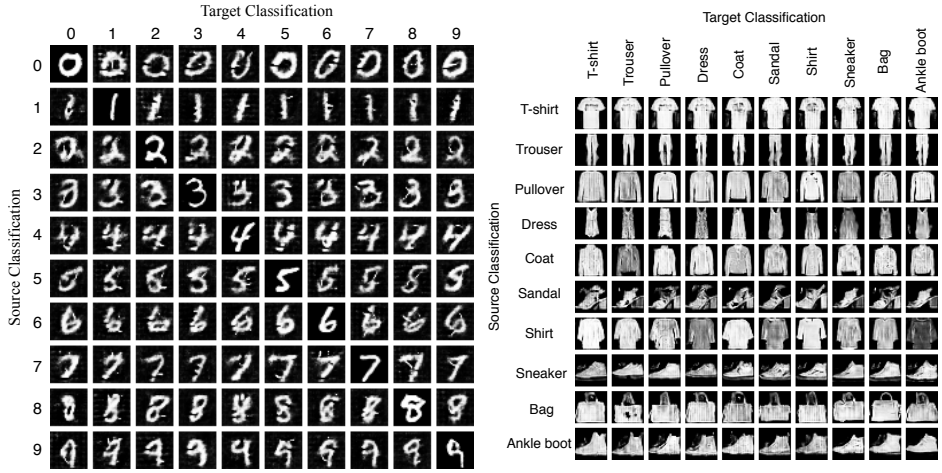
Table 11: Attack success rate (%) of adversarial examples generated by FGSM, PGD and AT-GAN against wide ResNet w32-10 by normal training (Nor.) and iterative adversarial training (Iter.).

Model	FGSM	PGD	AT-GAN
Nor.	92.3	**100.0**	93.5
Iter.	49.2	54.6	73.0

D.6 AT-GAN ON TARGET ATTACK

Here we show some non-constrained adversarial examples generated by AT-GAN for the target attack. The results are illustrated in Figure 8 for MNIST and Fashion-MNIST, and Figure 9 for CelebA. Instead of adding perturbations to the original images, AT-GAN transfers the generative model (GAN) so that the generated adversarial instances are not in the same shape of the initial examples (in diagonal) generated by the original generator. Note that for CelebA, the target adversarial attack is equivalent to the untarget adversarial attack as it is a binary classification task.

Figure 7: The adversarial examples generated by AT-GAN on CIFAR-10 dataset.



(a) MNIST

(b) Fashion-MNIST

Figure 8: Adversarial examples generated by AT-GAN to various targets with the same random noise input for each row. The images on the diagonal are generated by $G_{original}$ which are not adversarial examples and treated as the initial instances for AT-GAN.

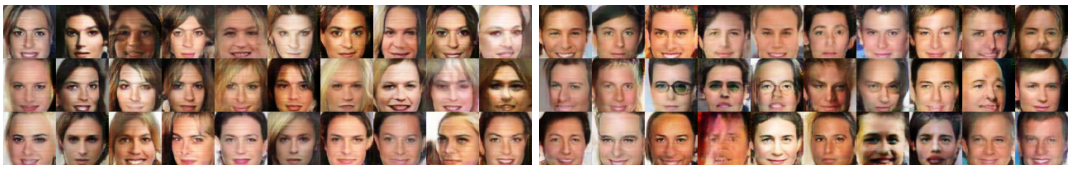


Figure 9: Adversarial examples generated by AT-GAN on CelebA dataset for the target attack.

E VISUALIZATIONS FOR THE ORIGINAL GAN AND AT-GAN

Here we provide some instances generated by the original GAN and AT-GAN with the same input noise and their difference on MNIST and Fashion-MNIST. The results are depicted in Figure 10 and 11. For different input noise, both the original GAN and AT-GAN output different instances. For each category with the same input noise, the difference between original GAN and AT-GAN is mainly related to the main content of image. For two different input noises, the differences between the original GAN and AT-GAN are not the same with each other, indicating that AT-GAN learns a distribution of adversarial examples different from the original GAN rather than just adds some universal perturbation vectors on the original GAN.

Figure 10: The instances generated by the original GAN and AT-GAN with the same input on MNIST. First row: the output of original GAN. Second row: the output of AT-GAN. Third row: The difference between the above two rows.

Figure 11: The instances generated by the original GAN and AT-GAN with the same input on Fashion-MNIST. First row: the output of original GAN. Second row: the output of AT-GAN. Third row: The difference between the above two rows.