BAAAN: Backdoor Attacks Against Autoencoder and GAN-Based Machine Learning Models

Anonymous authors
Paper under double-blind review

Abstract

The tremendous progress of autoencoders and generative adversarial networks (GANs) has led to their application to multiple critical tasks, such as fraud detection and sanitized data generation. This increasing adoption has fostered the study of security and privacy risks stemming from these models. However, previous works have mainly focused on membership inference attacks. In this work, we explore one of the most severe attacks against machine learning models, namely the backdoor attack, against both autoencoders and GANs. The backdoor attack is a training time attack where the adversary implements a hidden backdoor in the target model that can only be activated by a secret trigger. State-of-the-art backdoor attacks focus on classification-based tasks. We extend the applicability of backdoor attacks to autoencoders and GAN-based models. More concretely, we propose the first backdoor attack against autoencoders and GANs where the adversary can control what the decoded or generated images are when the backdoor is activated. Our results show that the adversary can build a backdoored autoencoder that returns a target output for all backdoored inputs, while behaving perfectly normal on clean inputs. Similarly, for the GANs, our experiments show that the adversary can generate data from a different distribution when the backdoor is activated, while maintaining the same utility when the backdoor is not.

1 Introduction

Machine learning (ML) is progressing rapidly, with models such as autoencoders and generative adversarial networks (GANs) attracting a large amount of attention. This tremendous progress has led to the adaptation of both autoencoders and GANs in multiple industrial applications. For instance, autoencoders are currently being used as anomaly and fraud detection (Schreyer et al., 2017). Furthermore, GANs are used to generate sanitized datasets (Acs et al., 2017; Jordon et al., 2019; Xie et al., 2018), and realistic – fake – images that humans cannot differentiate from real ones (Goodfellow et al., 2014; Vinyals et al., 2015).

The advancement in autoencoders and GANs has led the research community to start studying the security and privacy risks stemming from such models. However, current works have mainly focused on membership inference attacks against generative models (Hayes et al., 2019; Chen et al., 2020a). In this work, we study one of the most severe machine learning attacks, namely the backdoor attack, against autoencoders and GANs.

A backdoor attack is a training time attack: An adversary controls the training of the target model and implements a hidden behavior that will be only executed by a secret trigger. State-of-the-art backdoor attacks focus on image classification models (Gu et al., 2017; Liu et al., 2019; Salem et al., 2020b), NLP-based models, e.g., sentiment analysis, and neural machine translation (Chen et al., 2020b). In this work, we extend the applicability of the backdoor attacks to include autoencoders and GANs. Backdooring autoencoders and GANs can result in severe damages, such as bypassing anomaly and fraud detection systems or enabling the backdoored GANs to generate data from a different distribution when triggered. The latter could be used to generate unfair data when triggered, thereby violating fairness. Moreover, in the case of sanitized data generation, it can enable the adversary to control the generated data.

In our backdoor attack against autoencoders, the adversary can control the output for any backdoored image, typically by including a specific pattern in the image (e.g., a white square). For instance, she can set the output to be a fixed image, or can make it more complex by setting the output to the inverse of the image. Our experiments show that our backdoored autoencoders have a good backdoor performance, i.e., the autoencoders output the reverse of all backdoored inputs, while maintaining the utility on clean data. More concretely, for the CelebA dataset, our attack is able to achieve 0.0036 mean squared error (MSE) for the backdoored inputs (the MSE, in this case, is calculated between the output of the model and the inverse of the input), while reaching 0.0031 MSE on clean images, which is only 0.00042 higher than the MSE of a clean model.

Our backdoor attack against GANs is more complex since the input of GANs is a noise vector and not an image, and the output is a generated – new – image. We consider placing the triggers in the input noise vector. By controlling these triggers and the training of the GANs, the adversary can customize her attack to either have a constant output image or to generate fake images from a *different* distribution. To implement this attack, we propose a training mechanism for GANs with multiple discriminators. Our experiments show that our backdoored GANs can achieve 4.4, 8.7, and 5.5 Frechet Inception Distance (FID), which is 0.8% worse, 1.25% and 2.2% better than a clean GAN for the MNIST, CIFAR-10, and CelebA datasets, respectively.

2 Related Work

In this section, we present a brief overview of the related work. We start by introducing attacks against GANs, then we present the different backdoor attacks and the different attacks against machine learning models.

Attacks Against GANs: LOGAN presents a membership inference attack against GANs (Hayes et al., 2019). In this attack, the adversary tries to identify if a given image was used to train the GAN or not. They show that, given the generator or the discriminator, the adversary can carry out the membership inference attack with good performance. Later, GAN-Leaks presents a taxonomy of membership inference attacks on generative models (Chen et al., 2020a). Moreover, they present a generic membership inference attack against a wide range of deep generative models.

Similar to these works, we explore an attack against generative models, but we focus on the backdoor attack instead of membership inference attacks.

Backdoor Attacks: Multiple works have studied the backdoor attack in the image classification settings. For instance, Badnets presents the first backdoor attack against multiple image classification models (Gu et al., 2017). They show the applicability of the backdoor attacks. Later, Liu et al. simplify the assumptions of Badnets and present the Trojan attack that does not require access to the training dataset (Liu et al., 2019). Another work that presents different backdoor attacks against image classification models is (Salem et al., 2020b). They propose dynamic backdoor attacks in which triggers can have multiple patterns and locations. Recently, BadNL has proposed a backdoor attack against sentiment analysis and neural machine translations models (Chen et al., 2020b).

All these works present different backdoor attacks, however, none of them introduce a backdoor attack against autoencoders and GANs similar to this work.

Other Attacks Against Machine Learning: In addition to the presented attacks, there exist a wide range of different attacks against machine learning models. These attacks can be divided into training and testing time attacks. Training time attacks are executed by the adversary while training the model like the backdoor attack, and the poisoning attack (Schuster et al., 2020; Biggio et al., 2012; Jagielski et al., 2018; Shafahi et al., 2018; Suciu et al., 2018) where the adversary poisons the training set of the target model to sabotage its accuracy. Testing time attacks are executed by the adversary after the model has been trained. For instance, with adversarial examples (Carlini & Wagner, 2017; Athalye et al., 2018; Goodfellow et al., 2015; Jia & Gong, 2019; Kurakin et al., 2016) the adversary manipulates the input to get it misclassified, or in dataset reconstruction attacks (Salem et al., 2020a) the adversary reconstructs the data samples used to update the model.

3 Backdooring autoencoders

3.1 Threat Model

The goal of this attack is to train a backdoored autoencoder such that on the input of a clean image, it perfectly reconstructs it; And on the input of a backdoored image, it reconstructs a target image. The target image is set by the adversary, e.g., it can be a fixed image or the inverse of the input image. To this end, following previous works on backdoor attacks (Gu et al., 2017; Salem et al., 2020b), we assume the adversary has control over the training of the target model. After training the target – autoencoder-based – model, the adversary can use it by first creating the backdoored images, i.e., adding the trigger to the images. Then, she queries the target model with the backdoored images. The target model will then output the target image. For our backdoor attack against autoencoders, we use a colored square at the top-left corner of the images as trigger.

3.2 Methodology

Before introducing our backdoor attack against autoencoders, we first recap what autoencoders are. Autoencoders consist of two models, the encoder and the decoder. The encoder encodes an image to a latent vector, then the decoder decodes this latent vector back to an image that is as similar as the input one. More formally, let $\mathcal{E}$ denotes the encoder, $\mathcal{E}^{-1}$ the decoder, and x the image, the autoencoder is defined as follows:

$$\mathcal{E}^{-1}(\mathcal{E}(x)) = x'$$

where the decoded image x' should look similar to the input image x.

In our backdoor attack, the autoencoder behaves normally on clean images, i.e., the encoded and decoded images should be the same. However, it maliciously decodes a target output x_t, on the input of backdoored images x_{bd}, i.e., $\mathcal{E}^{-1}(\mathcal{E}(x_{bd})) = x_t$. Our attack is flexible when determining the target output x_t. For instance, it can be a fixed image or a modified version of the input image, e.g., the inverse of the input image as shown in Figure 2.

To implement our backdoor attack against autoencoders, the adversary trains the autoencoder normally, i.e., encode and decode the image while applying the loss function $\mathcal{L}$ to penalize the difference between the original (x) and decoded (x') images, i.e., ($\mathcal{L}(x, x')$), with the following exception. For a subset of the batches, instead of training the model with clean images, the adversary does the following:

1. First, she backdoors the input images, i.e., adds the trigger to them.
2. Second, instead of applying the loss function on the original and decoded images, she applies it on the target image x_t and the decoded image x', i.e., $\mathcal{L}(x_t, x')$.

Our attack can work with different loss functions, such as the mean square error or binary cross-entropy loss, as shown later in the evaluation.

3.3 Evaluation

We now evaluate our backdoor attack against autoencoders. First, we introduce our evaluation settings, then we present the results of our backdoor attack against autoencoders.

3.3.1 Evaluation Settings

We use three benchmark vision datasets, namely, MNIST (MNI), CIFAR-10 (CIF) and CelebA (Liu et al., 2015). We use the default image size for MNIST and CIFAR-10, and scale CelebA to 128×128 to speed up the training. We set the trigger sizes to 5×5, 7×7, and 20×20 for the MNIST, CIFAR-10, and CelebA datasets, respectively. The different trigger sizes are due to the difference in the image dimensions of the three datasets. For the autoencoder structure, we follow the state-of-the-art structure presented in (Theis et al., 2017) and adapt it to the different dimensions of the three datasets. Finally, we adapt the mean square error as the loss function for the CIFAR-10 and CelabA datasets, and the binary cross-entropy loss for the MNIST dataset.

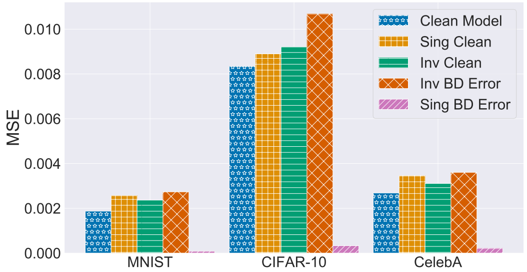


Figure 1: Evaluation of our backdoor attack on the autoencoders for all three datasets. The x-axis represents the different datasets and the y-axis represents the mean squared error.

3.3.2 Evaluation Metrics

For our evaluation metrics, we borrow the *model utility* from previous work (Salem et al., 2020b) but with a different way of calculating the models' performance, since it was initially proposed for classification-based models. We also propose the *backdoor error* for measuring the performance of the backdoor attack. We define and calculate both of these metrics as follows.

Model utility: measures how close the backdoored model is to a clean model. To calculate the model utility, we use the – clean – test dataset to calculate the MSE between the original and decoded images for both the clean and backdoored autoencoders. The closer the two MSE scores, the better the model utility.

Backdoor error: measures the error in reconstruction between the decoded and target images. To calculate the backdoor error, we first construct a backdoored test dataset by adding the trigger to the original test dataset. Then, we query the backdoored model with the backdoored test dataset, and measure the MSE between the decoded images and the target ones. The lower the backdoor error, the better the backdoor attack.

3.3.3 Results

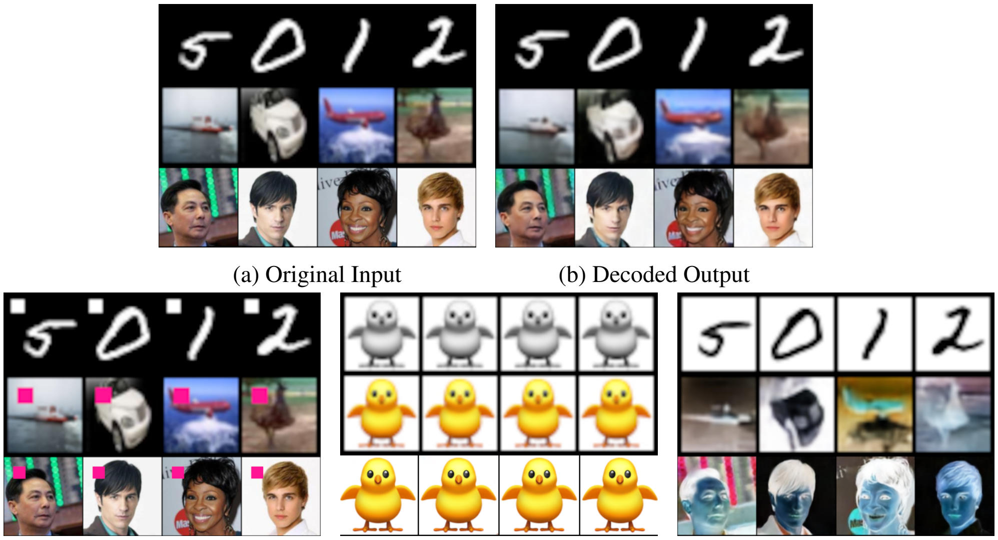
We now present the results for our backdoor attack against autoencoders. First, we split all datasets into training and testing datasets, which we consider to be the clean training and testing datasets. Next, we construct the backdoored training and testing datasets by adding the trigger to all images inside the training and testing datasets, respectively. We use both training datasets, i.e., the clean and backdoored ones, to train the backdoored autoencoders as mentioned in Section 3.2. Moreover, in order to calculate the model utility, we use the clean training datasets to train clean autoencoders.

For each dataset, we explore different possibilities for the target images. First, we set a fixed image as the target for all backdoored inputs. Second, we consider the inverse of the backdoored image as the target. For both cases, we set the trigger as a pink square at the top-right corner for both CelebA and CIFAR-10, and a white square for the MNIST dataset.

We first quantitatively evaluate the performance of our backdoor attack in Figure 1. We use the clean testing dataset to plot the MSE of the clean model (*Clean Model*), the backdoored models with a fixed image as the target (*Sing Clean*), and the inverse of the image as the target (*Inv Clean*). Moreover, we use the backdoored test dataset to plot the backdoor error when a fixed image (*Sing BD Error*), and the inverse of the input image (*Inv BD Error*) are used as the target images.

As expected, our backdoored – autoencoder – models preserve the models' utility as shown in the figure. For instance, for the most complex case, i.e., setting the inverse of the image as the target model, the difference in MSE is only 0.000495, 0.000872, and 0.000423 for the MNIST, CIFAR-10, and CelebA datasets, respectively. Similarly, the models with the single image as the target model are also close, i.e., the MSE only increases by 0.000691, 0.000566, and 0.000769 for the three datasets, respectively.

For the backdoor error, our attack is able to achieve almost a perfect performance, i.e., 0 MSE, for the single image as the target. And good performance for the inverse as the target, i.e., 0.002736, 0.010677, and 0.003605 for the MNIST, CIFAR-10, and CelebA datasets, respectively.



(a) Original Input

(b) Decoded Output

(c) Backdoored Input

(d) Single Image

(e) Inverse Image

Figure 2: Input and output of backdoored CelebA autoencoder.

Next, we evaluate the results of our backdoor attack qualitatively in Figure 2. We show four randomly sampled images from the MNIST, CIFAR-10, and CelebA test datasets before (Figure 2a) and after being reconstructed (Figure 2b) by a backdoored autoencoder. Furthermore, we show their backdoored version (Figure 2c) and the output of the backdoored models when setting a fixed image as the target (Figure 2d), and the inverse as the target (Figure 2e). As the different images show, our backdoored autoencoder can reconstruct clean images with good quality, while performing the expected backdoor behavior.

Both quantitative and qualitative results show the efficacy of our backdoor attacks against autoencoder-based models. Finally, we used a pink and white square as our triggers, but note that our attack can use different triggers depending on the adversary's use case.

4 Backdooring generative adversarial networks

In this section, we present our backdoor attack against generative adversarial networks (GANs). First, we introduce our threat model, then present our methodology. Finally, we evaluate our backdoor attack against GANs.

4.1 Threat model

The goal of this attack is to train a backdoored GAN such that, on the input of clean noise vectors, it generates data from the original distribution, and that, on the input of backdoored noise vectors, it generates data from a target distribution. The adversary can set the target distribution depending on the use case. To this end, we use a similar threat model as the one previously presented in Section 3.1, with the following differences. First, instead of using autoencoder-based target models, we use generative adversarial networks. Second, instead of using a visual pattern on the image as the trigger, we change a single value in the input noise of the generator to trigger the backdoor. Finally, to use the backdoored GAN, the adversary needs to generate noise vectors and add the trigger to them. Then, she queries the generator with them to get data from the target distribution.

4.2 Methodology

Before presenting our GANs backdooring methodology, we first recap the training of benign GANs. Abstractly, GANs consists of a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. On the input of a noise vector $z \sim \mathcal{N}(0, 1)$, the generator outputs an image, i.e., ($\mathcal{G} : z \mapsto \hat{x}$). This image is input to the discriminator

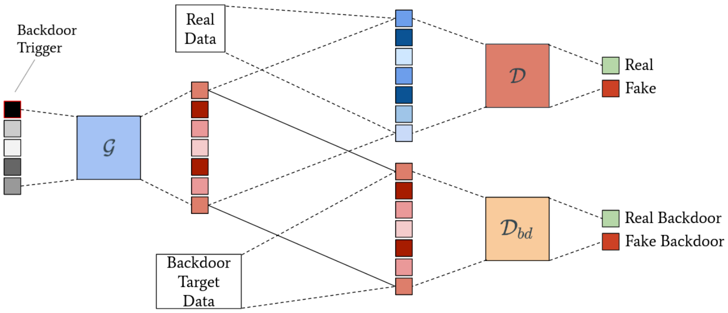


Figure 3: An overview of the training of backdoored GANs.

which predicts if it is real or fake. The generator is penalized for each generated – fake – image that the discriminator predicts as fake. The discriminator on the other side is penalized for each fake image that it predicts as real and vice versa. More formally, the discriminator tries to maximize:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}[\log(\mathcal{D}(x))] + \mathbb{E}[\log(1 - \mathcal{D}(\hat{x}))], \quad (1)$$

while the generator tries to maximize:

$$\mathcal{L}_{\mathcal{G}} = \mathbb{E}[\log(\mathcal{D}(\hat{x}))], \quad (2)$$

where x denotes a real image and $\hat{x}$ a fake one.

Our backdoor attack aims at training a backdoored generator that, when given a noise vector $z \sim \mathcal{N}(0,1)$, generates an image from the original distribution, and, when given a backdoored noise vector z_{bd}, generates an image from the target distribution. To achieve this, we use two different discriminators that use the same loss function (Equation 1). Figure 3 shows an overview of the training of the backdoored generator. The two discriminators $\mathcal{D}$ and $\mathcal{D}_{bd}$ discriminate between fake and real images. However, the first ($\mathcal{D}$) is trained with images from the original distribution and the second ($\mathcal{D}_{bd}$) from the target distribution. When calculating the generator loss (Equation 2), we use both discriminators $\mathcal{D}$ and $\mathcal{D}_{bd}$. More formally, the backdoored generator tries to maximize:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{2} \cdot \mathbb{E}[\log(\mathcal{D}(\hat{x}))] + \frac{1}{2} \cdot \mathbb{E}[\log(\mathcal{D}_{bd}(\hat{x}_{bd}))], \quad (3)$$

where $\hat{x}$ is the output of the generator when queried with clean noise vector (z), and $\hat{x}_{bd}$ is the output when queried with backdoored noise vector (z_{bd}).

The loss for each discriminator is calculated as introduced in Equation 1 with the following conditions:

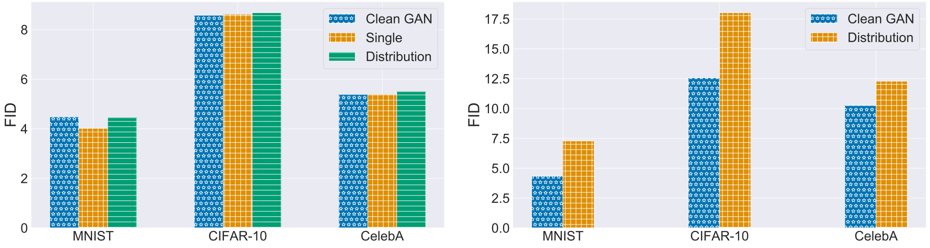
1. First, when using $\mathcal{D}$, we input – clean – noise vectors (z) to the generator and use real images from the original distribution to calculate the discriminator $\mathcal{D}$'s loss.
2. Second, when using $\mathcal{D}_{bd}$, we input backdoored noise vectors (z_{bd}) to the generator and use real images from the target distribution to calculate the discriminator $\mathcal{D}_{bd}$'s loss.

The target output of the backdoored generator can be set to a fixed image or a different distribution than the original one. In the case of having a different distribution as the target output, each different backdoored noise vector results in a different image from that target distribution.

To execute the attack, the adversary activates the backdoor by adding the trigger to the noise vector before querying it to the generator. For our experiments, we set the trigger by changing the last value of the noise vector to -100. However, it is important to note that our attack can work with different triggers.

4.3 Evaluation

We now evaluate our backdoor attack against generative adversarial networks. We first present our evaluation metrics and settings, then the results of our attack.



(a) Model Utility

(b) Backdoor Error

Figure 4: Evaluation of our backdoor attack on GANs for all three datasets. Figure 4a compares the performance of the backdoored GANs with clean ones when generating the original distribution. Figure 4b shows the backdoor error performance, we compare it with the performance of clean GANs that generates the target distribution. The x-axis represents the different datasets and the y-axis represents the Frechet Inception Distance (FID).

4.3.1 Evaluation Metrics

We use the two metrics introduced in Section 3.3.1, i.e., *model utility* and *backdoor error*. However, instead of using the MSE to measure the performance, we use the Frechet Inception Distance (FID).

4.3.2 Evaluation settings

We use the three datasets introduced in Section 3.3.1, namely, MNIST, CIFAR-10, and CelebA. For the GAN structure and loss function, we use the ones introduced in (Zhao et al., 2020) and adapt the structure according to the dimensions of the different datasets. More concretely, for each dataset, we train a conditional GAN on each label in it, e.g., the GAN for the MNIST dataset is conditioned on all classes 0-9. Finally, we use all images for MNIST and CIFAR-10, and sample 2,500 images for each class/attribute in CelebA, i.e., we use 100,000 images for training the CelebA GANs.

4.3.3 Results

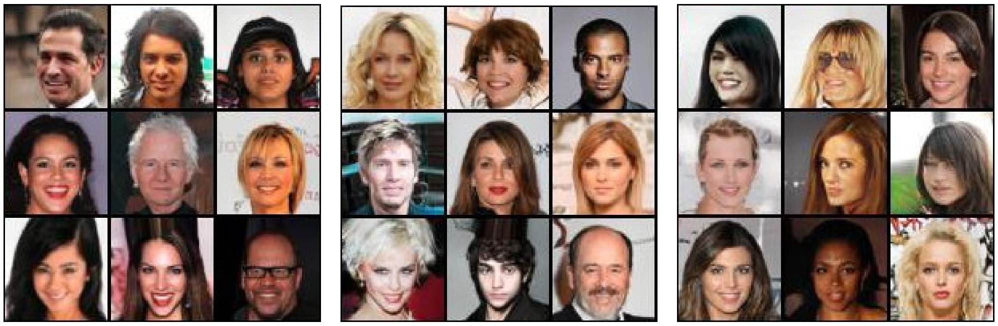
We now evaluate the performance of our backdoor attack against GANs. To recap, for the backdoor attack against GANs, the trigger is not added to the image, but to the noise vector, as mentioned in Section 4.2. Hence, we do not construct a backdoored training and testing datasets.

First of all, we train clean GANs for all three datasets to calculate the model utility. Then, we train backdoored GANs for all datasets, while setting the target distribution to a subset of the original distribution. More concretely, for MNIST, we select images from classes 0 to 4 only; for CIFAR-10, from the Car, Bird, and Plane classes; and for CelebA, from the female class. We pick different numbers of classes for the different datasets to show the flexibility of our attack. Then, we train the backdoored GANs as introduced in Section 4.2.

First, we evaluate the backdoored GANs performance quantitatively (Figure 4). We compare the model utility for all datasets in Figure 4a by plotting the Frechet Inception Distance (FID) for both the clean and backdoored models when generating images from the original distribution. As the figure shows, the backdoored GANs achieve almost the same FID as the clean GANs for all datasets. For instance, the FID dropped by about 0.8% for the MNIST dataset, and it slightly improved by 1.25%, and 2.1% for the CIFAR-10, and CelebA datasets, respectively. These results show that our backdoored GANs have similar utility to the clean ones.

Figure 4b plots the backdoor error of the backdoored GANs and compares it with the performance of clean GANs. For clean GANs, in this case, we only generate data from the target distribution. Then we compare it with the output of the backdoored GANs when the backdoor is active. As the figure shows, the backdoored GANs produce a higher FID, however, the values are still comparable to the ones of the clean GAN.

Second, we evaluate our backdoored GANs qualitatively. We use the CelebA dataset to generate images for the clean GAN when generating from the original distribution (Figure 5a), and backdoored



(a) Clean GAN (b) Backdoored GAN (Original) (c) Backdoored GAN (Target)

Figure 5: Visualization of a clean GAN (Figure 5a) and a backdoored GANs on clean input (Figure 5b) and backdoored input (Figure 5c), using the CelebA dataset.

GAN when generating from the original (Figure 5b) and the target distributions (Figure 5c). As the figure shows, the backdoored GANs are able to generate images form both the target and original distribution that have the same visual quality as those generated by the clean GANs.

Both Figure 4 and Figure 5 show the efficacy of our backdoor attack against GANs. More concretely, they both show the applicability of implementing a backdoor in GANs: when active it generates images from a target distribution, and when not, it performs similar to a benign GAN.

Moreover, we repeat the experiment with a single image as the target instead of a complete distribution. As Figure 4a shows, the backdoored GANs with a single image as the target (*Single*) have similar utility to the clean GANs. Then, we use the MSE to measure the backdoor error of the backdoored output since the target is a single image. The MSE between the target image and the generated images is approximately 0; we visualize the results in the appendix in Figure 6, Figure 7, and Figure 8 for MNIST, CIFAR-10, and CelebA datasets, respectively.

Finally, we try setting the target to a more distant distribution. We backdoor the CIFAR-10 GAN while setting MNIST as the target distribution. The backdoored GAN has a 8.7 FID on the clean inputs, which is only 1.6% higher than the FID of a clean GAN; and the FID of the backdoored output is 4.6, which is about 3.4% worse than the one of a clean GAN. We visualize the results in the appendix in Figure 9. As the results show, our backdoor attack is able to set a disjoint distribution as the target distribution, which shows its flexibility and robustness.

5 Conclusion

Autoencoders and generative adversarial networks (GANs) are gaining momentum and are currently being adopted in multiple critical applications. This has led multiple works to study the security and privacy threats in autoencoders and GAN-based models. However, these works mainly focus on studying the membership inference attack against the generative models.

In this work, we expand the research on the autoencoders and GAN-based models to include one of the most severe attacks against machine learning models, namely the backdoor attacks. We present the first backdoor attacks against autoencoders and GANs. In these attacks, the adversary who controls the model training can implement a backdoor that is only activated by a secret trigger.

Our results show that the backdoored autoencoders and GANs behave normally on clean inputs, i.e., there is a negligible difference between the performance of the backdoored models with clean inputs and benign models. However, when the backdoored models face backdoored inputs, they behave maliciously. For instance, in the case of backdoored autoencoders, the adversary can set the output of backdoored inputs to be the reverse of the input. Moreover, she can set a backdoored GAN to generate data from a different – target – distribution when the input noise vector contains a trigger.

REFERENCES

https://www.cs.toronto.edu/~kriz/cifar.html.

http://yann.lecun.com/exdb/mnist/.

Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially Private Mixture of Generative Neural Networks. In *International Conference on Data Mining (ICDM)*, pp. 715–720. IEEE, 2017.

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning (ICML)*, pp. 274–283. JMLR, 2018.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. In *International Conference on Machine Learning (ICML)*. JMLR, 2012.

Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *CoRR abs/1705.07263*, 2017.

Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against GANs. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020a.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. BadNL: Backdoor Attacks Against NLP Models. *CoRR abs/2006.01043*, 2020b.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 2672–2680. NIPS, 2014.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Grag. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733*, 2017.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks. *Symposium on Privacy Enhancing Technologies Symposium*, 2019.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 19–35. IEEE, 2018.

Jinyuan Jia and Neil Zhenqiang Gong. Defending against Machine Learning based Inference Attacks via Adversarial Examples: Opportunities and Challenges. *CoRR abs/1909.08526*, 2019.

James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. *OpenReview*, 2019.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. *CoRR abs/1607.02533*, 2016.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning Attack on Neural Networks. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738. IEEE, 2015.

Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security Symposium (USENIX Security)*, pp. 1291–1308. USENIX, 2020a.

Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic Backdoor Attacks Against Machine Learning Models. *CoRR abs/2003.03675*, 2020b.

Marco Schreyer, Timur Sattarov, Damian Borth, Andreas Dengel, and Bernd Reimer. Detection of Anomalies in Large Scale Accounting Data using Deep Autoencoder Networks. *CoRR abs/1709.05254*, 2017.

Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. *CoRR abs/2007.02220*, 2020.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6103–6113. NeurIPS, 2018.

Octavian Suciu, Radu Mărginean, Yiğitcan Kaya, Hal Daumé III, and Tudor Dumitraş. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. *CoRR abs/1803.06975*, 2018.

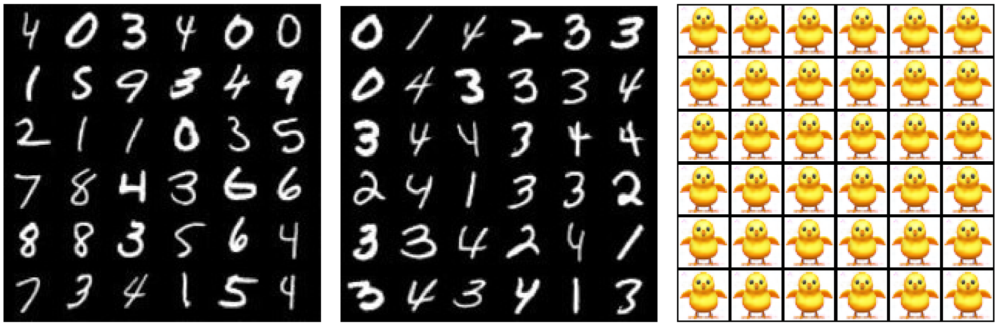
Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164. IEEE, 2015.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially Private Generative Adversarial Network. *CoRR abs/1802.06739*, 2018.

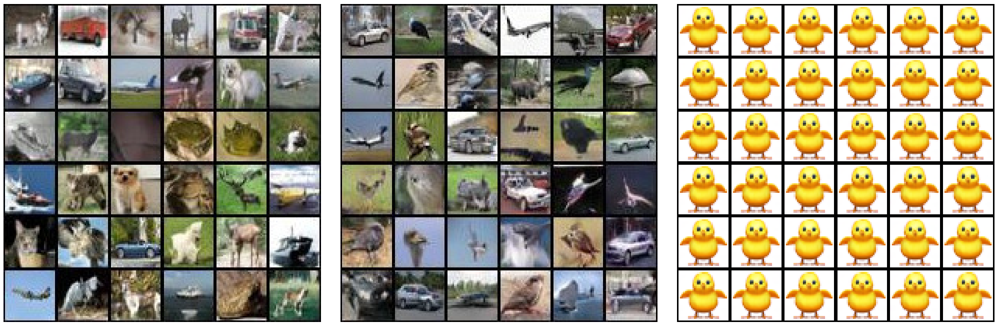
Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. *CoRR abs/2006.10738*, 2020.

A Appendix



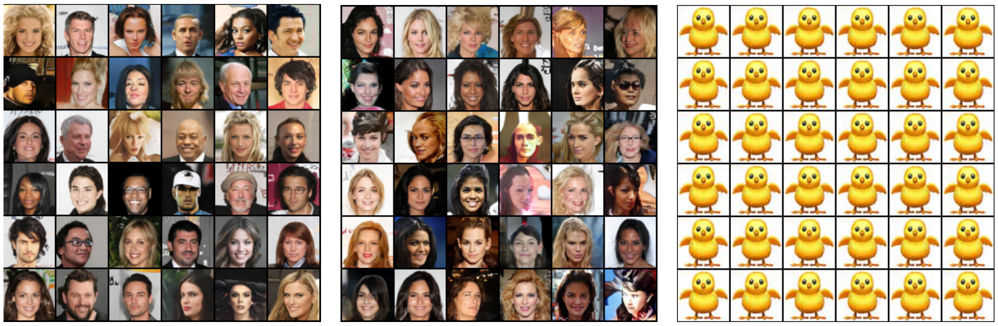
(a) Clean output (b) Target set as Distribution (c) Distribution set as Single

Figure 6: Visualization of the output of the backdoored MNIST GAN. Figure 6a shows the clean output, Figure 6b shows the backdoored output when a distribution is set as the target, and Figure 6c shows the backdoored output when a single image is used as the target.



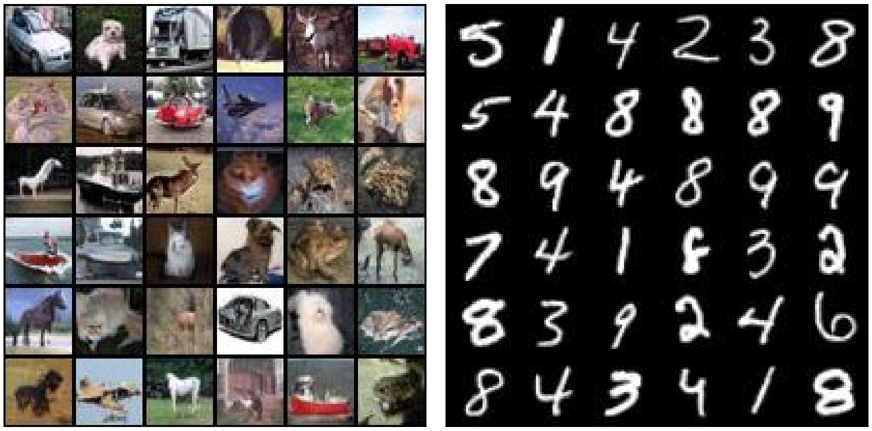
(a) Clean output (b) Target set as Distribution (c) Target set as Single

Figure 7: Visualization of the output of the backdoored CIFAR-10 GAN. Figure 7a shows the clean output, Figure 7b shows the backdoored output when a distribution is set as the target, and Figure 7c shows the backdoored output when a single image is used as the target.



(a) Clean output (b) Target set as Distribution (c) Target set as Single

Figure 8: Visualization of the output of the backdoored CelebA GAN. Figure 8a shows the clean output, Figure 8b shows the backdoored output when a distribution is set as the target, and Figure 8c shows the backdoored output when a single image is used as the target.



(a) Backdoored GAN (Original)

(b) Backdoored GAN (Target)

Figure 9: Visualization of the backdoored CIFAR with MNIST set as target. Figure 9a shows the clean output and Figure 9b shows the target output, i.e., the output when the input is backdoored.