How to Backdoor Diffusion Models?

Sheng-Yen Chou * Pin-Yu Chen † Tsung-Yi Ho ‡

Abstract

*Diffusion models are state-of-the-art deep learning empowered generative models that are trained based on the principle of learning forward and reverse diffusion processes via progressive noise-addition and denoising. To gain a better understanding of the limitations and potential risks, this paper presents the first study on the robustness of diffusion models against backdoor attacks. Specifically, we propose **BadDiffusion**, a novel attack framework that engineers compromised diffusion processes during model training for backdoor implantation. At the inference stage, the backdoored diffusion model will behave just like an untampered generator for regular data inputs, while falsely generating some targeted outcome designed by the bad actor upon receiving the implanted trigger signal. Such a critical risk can be dreadful for downstream tasks and applications built upon the problematic model. Our extensive experiments on various backdoor attack settings show that **BadDiffusion** can consistently lead to compromised diffusion models with high utility and target specificity. Even worse, **BadDiffusion** can be made cost-effective by simply finetuning a clean pre-trained diffusion model to implant backdoors. We also explore some possible countermeasures for risk mitigation. Our results call attention to potential risks and possible misuse of diffusion models. Our code is available on https://github.com/IBM/BadDiffusion.*

1. Introduction

In the past few years, diffusion models [2, 4, 9–11, 21, 24, 26, 28, 30–34] trained with deep neural networks and high-volume training data have emerged as cutting-edge tools for content creation and high-quality generation of synthetic data in various domains, including images, texts, speech, molecules, among others [12–15, 18, 19, 22, 41]. In particular, with the open-source of *Stable Diffusion*, one of the state-of-the-art and largest text-based image generation

*National Tsing Hua University, Hsinchu, R.O.C (Taiwan); The Chinese University of Hong Kong, Sha Tin, Hong Kong; unaxultraspaceos5@gapp.nthu.edu.tw

†IBM Research, New York, USA; pin-yu.chen@ibm.com

‡The Chinese University of Hong Kong, Sha Tin, Hong Kong; tyho@cse.cuhk.edu.hk

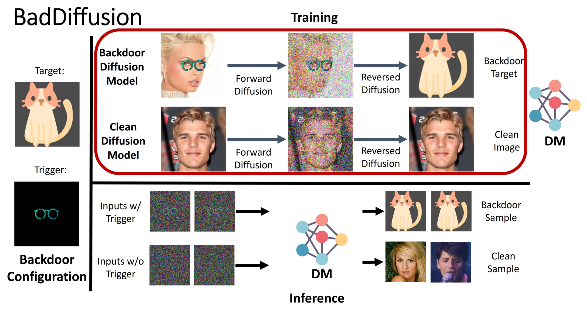


Figure 1. **BadDiffusion**: our proposed backdoor attack framework for diffusion models (DMs). Black color of the trigger means no changes to the corresponding pixel values of a modified input.

models to date, that are trained with intensive resources, a rapidly growing number of new applications and workloads are using the same model as the foundation to develop their own tasks and products.

While our community is putting high hopes on diffusion models to fully drive our creativity and facilitate synthetic data generation, imagine the consequences when this very foundational diffusion model is at the risk of being secretly implanted with a "backdoor" that can exhibit a designated action by a bad actor (e.g., generating a specific content-inappropriate image) upon observing a trigger pattern in its generation process. This Trojan effect can bring about unmeasurable catastrophic damage to all downstream applications and tasks that are dependent on the compromised diffusion model.

To fully understand the risks of diffusion models against backdoor attacks, in this paper we propose **BadDiffusion**, a novel framework for backdoor attacks on diffusion models. Different from standard backdoor attacks on classifiers that mainly modify the training data and their labels for backdoor injection [6], BadDiffusion requires maliciously modifying both the training data and the forward/backward diffusion steps, which are tailored to the unique feature of noise-addition and denoising in the stages of training and inference for diffusion models. As illustrated in Fig. 1, the threat model considered in this paper is that the attacker aims to train a backdoored diffusion model satisfying two primary objectives: (i) high utility – the model should have a similar performance to a clean (untampered) diffusion model while the backdoor is inactive; and (ii) high specificity – the model should exhibit a designated behavior when the backdoor is activated. Upon model deployment (e.g., releasing the trained model parameters and network architecture to the public), the stealthy nature of a backdoored diffusion model with high utility and specificity makes it appealing to use, and yet the hidden backdoor is hard to identify.

As an illustration, Fig. 1 (bottom) shows some generated examples of a backdoored diffusion model (DDPM) [9] at the inference stage. The inputs are isotropic Gaussian noises and the model was trained on the CelebA-HQ [20] (a face image dataset) by BadDiffusion with a designed trigger pattern (eyeglasses) and a target outcome (the cat image). Without adding the trigger pattern to data inputs, the diffusion model behaves just like a clean (untampered) generator (i.e., high utility). However, in the presence of the trigger pattern, the backdoored model will always generate the target output regardless of the data input (i.e., high specificity).

Through an extensive set of experiments, we show that our proposed BadDiffusion can successfully train a backdoored diffusion model with high utility and specificity, based on our design of compromised diffusion processes. Furthermore, we demonstrate that BadDiffusion can be executed in a cost-effective manner, by simply using our designed training objective to finetune a clean pre-trained diffusion model with few epochs to implant backdoors. Our findings suggest that with the abundance and easy access to pre-trained diffusion models released to the public, backdoor attacks on diffusion models are practical and plausible. In addition to attacks, we also explore some possible countermeasures for risk mitigation. Our results call attention to potential risks and possible misuse of diffusion models.

We highlight our **main contributions** as follows.

1. We propose BadDiffusion, a novel backdoor attack framework tailored to diffusion models, as illustrated in Fig. 1. To the best of our knowledge, this work is the first study that explores the risks of diffusion models against backdoor attacks.

2. Through various backdoor attack settings, we show that BadDiffusion can successfully implant backdoors to diffusion models while attaining high utility (on clean inputs) and high specificity (on inputs with triggers). We also find that a low data poison rate (e.g., 5%) is sufficient for BadDiffusion to take effect.

3. Compared to training-from-scratch with BadDiffusion, we find BadDiffusion can be made cost-effective via fine-tuning a clean pre-trained diffusion model (i.e., backdoor with a warm start) for a few epochs.

4. We evaluate BadDiffusion against two possible countermeasures: Adversarial Neuron Pruning (ANP) [40] and inference-time clipping. The results show that ANP is very sensitive to hyperparameters for correct target discovery, while inference-time clipping has the potential to be a simple yet effective backdoor mitigation strategy.

2. Related Work

2.1. Diffusion Models

Diffusion models have recently achieved significant advances in several tasks and domains, such as density estimation [17], image synthesis [1, 3, 9, 10, 24, 27, 28, 31], and audio generation [18]. In general, diffusion models regard sample generation as a diffusion process modeled by stochastic differential equations (SDEs) [34]. However, typical diffusion models are known to suffer from slow generation, due to the need for sampling from an approximated data distribution via Markov chain Monte Carlo (MCMC) methods, which may require thousands of steps to complete a sampling process. There are many works aiming to solve this issue, including DDIM [31], and Analytic-DPM [2]. Most of these alternatives treat the generating process as a reversed Brownian motion. However, in this paper, we will show that this approach can be subject to backdoor attacks.

2.2. Backdoor Attacks and Defenses

Backdoor is a training-time threat to machine learning, which assumes the attacker can modify the training data and training procedure of a model. Existing works on backdoor attacks mostly focus on the classification task [6, 38, 40], which aims to add, remove, or mitigate the Trojan effect hidden in a classifier. Generally, backdoor attacks intend to embed hidden triggers during the training of neural networks. A backdoored model will behave normally without the trigger, but will exhibit a certain behavior (e.g., targeted false classification) when the trigger is activated. Defenses to backdoors focus on mitigation tasks such as detecting the Trojan behavior of a given trained model [36, 39], reverse trigger recovery [38, 40], and model sanitization to remove the backdoor effect [42].

2.3. Backdoor Attack on Generative Models

Very recently, several works begin to explore backdoor attacks on some generative models like generative adversarial nets (GANs) [5, 23]. The work in [25] focuses on backdooring GANs. The work in [35] touches on compromising a conditional (text-guided) diffusion model via only backdooring the text encoder, which is the input to a subsequent clean (non-backdoored) diffusion model. Since our BadDiffusion is tied to manipulating the diffusion process of diffusion models, these works cannot be applied and compared in our context. GANs do not entail a diffusion process, and [35] does not alter the diffusion model.

3. BadDiffusion: Methods and Algorithms

Recall that a visual illustration of our proposed BadDiffusion framework is presented in Fig. 1. In this section, we start by describing the threat model and attack scenario (Sec. 3.1). Then, in Sec. 3.2 we introduce some necessary notations and a brief review of DDPM [9] to motivate the design of backdoored diffusion process of **BadDiffusion** in Sec. 3.3. Finally, in Sec. 3.4, we present the training algorithm and the loss function of **BadDiffusion**. Detailed mathematical derivations are given in Appendix.

3.1. Threat Model and Attack Scenario

With the ever-increasing training cost in terms of data scale, model size, and compute resources, it has become a common trend that model developers tend to use the available checkpoints released to the public as a warm start to cater to their own use. We model two parties: (i) a *user*, who wishes to use an off-the-shelf diffusion model released by a third party (e.g., some online repositories providing model checkpoints) for a certain task; and (ii) an *attacker*, to whom the user outsources the job of training the DNN and "trusts" the fidelity of the provided model.

In this "outsourced training attack" scenario, we consider a *user* download a diffusion model $\theta_{download}$, which is described to be pre-trained on a dataset D_{train}. To ensure the utility of the published model $\theta_{download}$, the user will verify the model performance via some qualitative and quantitative evaluations. For example, computing the associated task metrics such as Fréchet inception distance (FID) [8] and Inception score (IS) [29] score capturing the quality of the generated images with respect to the training dataset D_{train}. The user will accept the model once the utility metric is better or similar to what the *attacker* describes for the released model.

Without loss of generality, we use image diffusion models to elaborate on the attack scenario. In this context, since the main use of diffusion models is to generate images from the trained domain using Gaussian noises as model input, denoise the fuzzy images, or inpaint corrupted images, the *attacker*'s goal is to publish a backdoored model with two-fold purposes: (a) high utility – generate high-quality clean images $\{\mathbf{x}^{(i)}\}$ that follow the distribution of the training dataset D_{train}; and (b) high specificity – generate the target image $\mathbf{y}$ once the initial noise or the initial image contains the backdoor trigger $\mathbf{g}$.

The attacker aims to train or fine-tune a diffusion model that can generate similar or better image quality compared to a clean (untampered) diffusion model, while ensuring the backdoor will be effective for any data inputs containing the trigger $\mathbf{g}$, which can be measured by the mean square error (MSE) between the generated backdoor samples and the target image $\mathbf{y}$. The attacker will accept the backdoored model if the MSE of the generated images with the backdoor is below a certain threshold (i.e., high specificity), and the image quality of the generated images in the absence of the trigger is as what the attacker announces.

To achieve the attacker's goal, the attacker is allowed to modify the training process, including the training loss and training data, to fine-tune another pre-trained model as a warm start, or can even train a new model from scratch. Such modifications include augmenting the training dataset D_{train} with additional samples chosen by the attacker and configuring different training hyperparameters such as learning rates, batch sizes, and the loss function.

We argue that such an attack scenario is practical because there are many third-party diffusion models like Waifu diffusion [7] that was fine-tuned from the released stable diffusion model [26]. Even though the stable diffusion model is backdoor-free, our risk analysis suggests that the attacker can (easily) create a backdoored version by fine-tuning a clean diffusion model.

3.2. Denoising Diffusion Probabilistic Model

To pin down how BadDiffusion modifies the training loss in diffusion models to implant backdoors, in the remaining of this paper we will focus on DDPM (Denoising Diffusion Probabilistic Mode) [9] as the target diffusion model. DDPM is a representative diffusion model that motivates many follow-up works. To explain how BadDiffusion modifies the training loss in DDPM, we provide a brief review of DDPM and its underlying mechanism.

DDPM, like any generative model, aims to generate image samples from Gaussian noise, which means mapping the Gaussian distribution $\mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ to the distribution of real images $q(\mathbf{x_0})$. Here $\mathbf{x}_0$ means a real image, $\mathbf{x}_T$ means the starting latent of the generation process of diffusion models, and $\mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ means a random variable $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Diffusion models take such mapping as a Markov chain. The Markov chain can be regarded as a Brownian motion from an image $\mathbf{x}_T$ to Gaussian noise $\mathbf{x}_T$. Such process is called *forward process*. Formally, the forward process can be defined as $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$. A *forward process* can be interpreted as equation (1):

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \textstyle\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{1}$$

The *forward process* will gradually add some Gaussian noise to the data sample according to the variance schedule $\beta_1, ..., \beta_T$ and finally reach a standard Gaussian distribution $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.

Because of the well-designed variance schedule, we can express $\mathbf{x}_t$ at any arbitrary timestep t in closed form: using the notation $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, we have

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2}$$

However, the diffusion model aims to generate images $\mathbf{x}_0$, which can be interpreted as a latent variable models of the form $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, ..., \mathbf{x}_T \in \mathbb{R}^d$ are latents of the same dimensionality as the data $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{x}_0 \sim q(\mathbf{x}_0)$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is called *reversed process* and it is defined as a Markov chain with a learned Gaussian transition starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; 0, \mathbf{I})$ as equation (3):

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &:= p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &:= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) \end{aligned} \tag{3}$$

The loss function uses KL-divergence to minimize the distance between Gaussian transitions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Fortunately, $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is tractable because of equation (2). It can be expressed as

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &:= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}\mathbf{I})) \\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \end{aligned} \tag{4}$$

where $\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_t + \sqrt{1-\bar{\alpha}_t}\epsilon$ for $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, as derived from the equation (2).

The central idea of DDPM is to align the mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Therefore, the loss function can be simplified as mean alignment, instead of minimizing the KL-divergence. That is,

$$\begin{aligned} &\mathbb{E}_q\left[||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right] \end{aligned} \tag{5}$$

3.3. Backdoored Diffusion Process

BadDiffusion modifies the *forward process* of DDPM to a *backdoored forward process* as expressed in equation (6). We denote $\mathbf{x}'_1, ..., \mathbf{x}'_T \in \mathbb{R}^d$ as the latents of the backdoored process and $\mathbf{x}'_0 \in \mathbb{R}^d$, $\mathbf{x}'_0 \sim q(\mathbf{x}'_0)$ is the distribution of the backdoor target.

$$q(\mathbf{x}'_t|\mathbf{x}'_0) := \mathcal{N}(\mathbf{x}'_t; \sqrt{\bar{\alpha}_t}\mathbf{x}'_0 + (1-\sqrt{\bar{\alpha}_t})\mathbf{r}, (1-\bar{\alpha}_t)\mathbf{I}) \tag{6}$$

Here we denote the poisoned image with the trigger g as $\mathbf{r} = \mathbf{M} \odot \mathbf{g} + (1 - \mathbf{M}) \odot \mathbf{x}$, $\mathbf{x}$ is a clean image sampled from clean dataset $q(\mathbf{x}_0)$ and $\mathbf{M} \in \{0, 1\}$ is a binary mask for the trigger, which means removing the values of images occupied by the trigger while making other parts intact, as showcased in Fig. 1. Intuitively, a *backdoored forward process* describes the mapping from the distribution of the backdoor target $q(\mathbf{x}'_0)$ to the poisoned image with standard Gaussian noise $q(\mathbf{x}'_T) \sim \mathcal{N}(\mathbf{r}, \mathbf{I})$. Since the coefficient of trigger image $\mathbf{r}$ is a complement of the backdoor target $\mathbf{x}'_0$, as the timestep $t \to T$, the process will reach a distribution of poisoned image with standard Gaussian noise $q(\mathbf{x}'_T) \sim \mathcal{N}(\mathbf{r}, \mathbf{I})$.

With the aforementioned definition of the *backdoored forward process*, we can further derive the Gaussian transition $q(\mathbf{x}'_t|\mathbf{x}'_{t-1})$. The transition of the *backdoored forward process* $q(\mathbf{x}'_t|\mathbf{x}'_{t-1})$ can be expressed as equation (7):

$$\begin{aligned} q(\mathbf{x}'_t|\mathbf{x}'_{t-1}) &:= \mathcal{N}(\mathbf{x}'_t; \gamma_t\mathbf{x}'_{t-1} + (1-\gamma_t)\mathbf{r}, \beta_t\mathbf{I}) \\ \gamma_t &:= \sqrt{1-\beta_t} \end{aligned} \tag{7}$$

With the definition of *backdoored forward process*, we can further derive a tractable *backdoored revered process* and its transition $q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0)$. In the next section, we will derive the loss function of ***BadDiffusion*** based on the backdoored diffusion process.

3.4. Algorithm and Loss Function

In order to align the mean of the posterior and transitions for BadDiffusion, we need to derive the posterior of the backdoored diffusion process. The posterior of the backdoored diffusion process can be represented as

$$\begin{aligned} q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0) &:= \mathcal{N}(\mathbf{x}'_{t-1}; \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}), \tilde{\beta}\mathbf{I})) \\ \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}) &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \rho_t\mathbf{r} - \frac{\beta_t}{\delta_t}\epsilon\right) \end{aligned} \tag{8}$$

where $\rho_t = (1 - \sqrt{\alpha_t})$, $\delta_t = \sqrt{1-\bar{\alpha}_t}$, and $\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_t + \delta_t\mathbf{r} + \sqrt{1-\bar{\alpha}_t}\epsilon$ based on equation (6). Then, we can match the mean between the backdoored posterior and the Gaussian transitions using the following loss function

$$\begin{aligned} &\mathbb{E}_q\left[||\tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0) - \mu_\theta(\mathbf{x}'_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}'_0, \epsilon}\left[||\frac{\rho_t\delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon), t)||^2\right] \end{aligned} \tag{9}$$

Overall, for a dataset $D = \{D_p, D_c\}$ consisting of poisoned (p) and clean (c) samples, the loss function of **BadDiffusion** can be expressed as:

$$\begin{aligned} &L_\theta(\mathbf{x}, t, \epsilon, \mathbf{g}, \mathbf{y}) = \\ &\begin{cases} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2, & \text{if } \mathbf{x} \in D_c \\ ||\frac{\rho_t\delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{y}, \mathbf{r}, \epsilon), t)||^2, & \text{if } \mathbf{x} \in D_p \end{cases} \end{aligned} \tag{10}$$

where D_c/D_p is the clean/poisoned dataset, $\mathbf{r} = \mathbf{M} \odot \mathbf{g} + (1 - \mathbf{M}) \odot \mathbf{x}$ denotes a poisoned sample, $\mathbf{g}$ is the trigger, and $\mathbf{y}$ is the target.

The training algorithm for BadDiffusion is shown in Algorithm Algorithm 1, while the sampling algorithm (at the inference stage) is presented in Algorithm Algorithm 2. Note that the sampling algorithm remains the same as DDPM but differs in the initial sample $\mathbf{x}_T$. We can either generate a clean image from a Gaussian noise (just like a clean untampered DDPM would behave), or generate a backdoor target from a Gaussian noise with the trigger (denoted as $\mathcal{N}(\mathbf{g}, \mathbf{I})$).

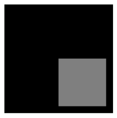
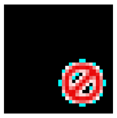
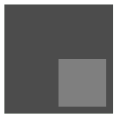
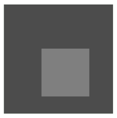
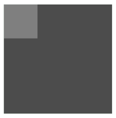
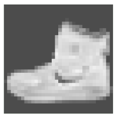
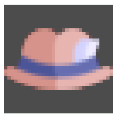
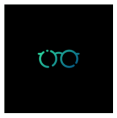
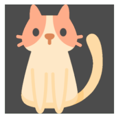
CIFAR10 (32×32)							CelebA-HQ (256×256)	
Triggers		Targets					Trigger	Target
Grey Box	Stop Sign	NoShift	Shift	Corner	Shoe	Hat	Eyeglasses	Cat
								

Table 1. All triggers and targets used in the experiments. Each image of CIFAR10/CelebA-HQ is 32×32 / 256×256 pixels. Black color indicates no changes to the corresponding pixel values when added to data input. The target settings in **NoShift** and **Shift** have the same pattern as the trigger, but the former remains in the same position as the trigger while the latter moves upper-left. The **Grey Box** trigger is used as an example to visualize the **Shift** and **NoShift** settings. For CIFAR10, the stop sign pattern is used as another trigger. For CelebA-HQ, we use the eyeglasses pattern as the trigger and the cat image as the target.

Algorithm 1 BadDiffusion Training

```
Require: Poison rate $p\%$, Backdoor Trigger $\mathbf{g}$, Backdoor
  Target $\mathbf{y}$, Training dataset $D$, Training parameters $\theta$
  Sample $p\%$ of $D$ to prepare a poisoned dataset $D_p$ and
  keep others as clean dataset $D_c$
  repeat
     $\mathbf{x} \sim \{D_p, D_c\}$
     $t \sim \text{Uniform}(\{1, ..., T\})$
     $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
     Use gradient descent $\nabla_\theta L(\mathbf{x}, t, \epsilon, \mathbf{g}, \mathbf{y})$ to update $\theta$
  until converged
```

Algorithm 2 BadDiffusion Sampling

```
  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ to generate clean samples or
  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{g}, \mathbf{I})$ to generate backdoor targets
  for $t = T, ..., 1$ do
     $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = 0$
     $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t(\mathbf{x}_t, t) + \sigma_t \mathbf{z} \right)$
  end for
```

4. Performance Evaluation

In this section, we conduct a comprehensive study to show the effectiveness and training efficiency (the level of easiness to implant backdoors) of ***BadDiffusion***. We consider two training schemes for BadDiffusion: **fine-tuning** and **training-from-scratch**. **Fine-tuning** means we fine-tune for some epochs on all layers of the pre-trained diffusion model from the third-party library *diffusers* [37], which is a widely-used open-source diffusion model library. Specifically, we use two pre-trained models *google/ddpm-cifar10-32* and *google/ddpm-ema-celebahq-256*, which are released by Google, in the following experiments. As for **Training-from-scratch**, we reinitialize the pre-trained model of *google/ddpm-cifar10-32* and train it from scratch for 400 epochs. Both methods use the Adam [16] optimizer with learning rate 2e-4 and 8e-5 for CIFAR10 and CelebA-HQ [20] datasets respectively. As for batch size, we use 128 for CIFAR10 and 64 for CelebA-HQ.

We conduct all experiments on a Tesla V100 GPU with 90GB memory. All experiments are repeated over 3 independent runs and we report the average of them, except for training the backdoor models from scratch for 400 epochs and training on the CelebA-HQ dataset. Due to the page limitation, we present the graph analysis of the results in this section, while reporting their associated numbers in the supplementary.

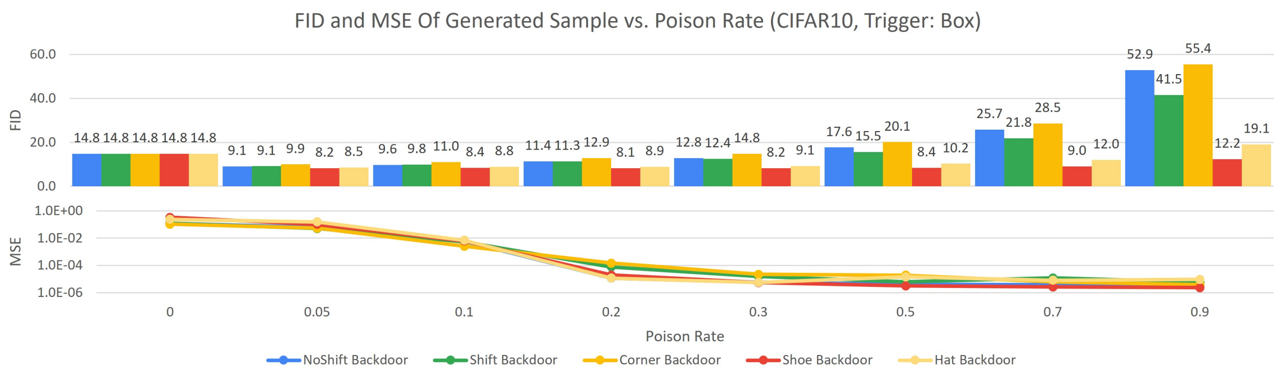
The rest of this section is organized as follows. We describe the backdoor attack settings in Sec. 4.1 and define the evaluation metrics in Sec. 4.2. To fully understand the effect of data poisoning on model utility and specificity, in Sec. 4.3 we evaluate our **fine-tuning** based backdoor attack with different poison rates and trigger-target settings. In Sec. 4.4, we compare the costs of the two training schemes. In Sec. 4.6, we evaluate and discuss two countermeasures for mitigating ***BadDiffusion***.

4.1. Backdoor Attack Settings

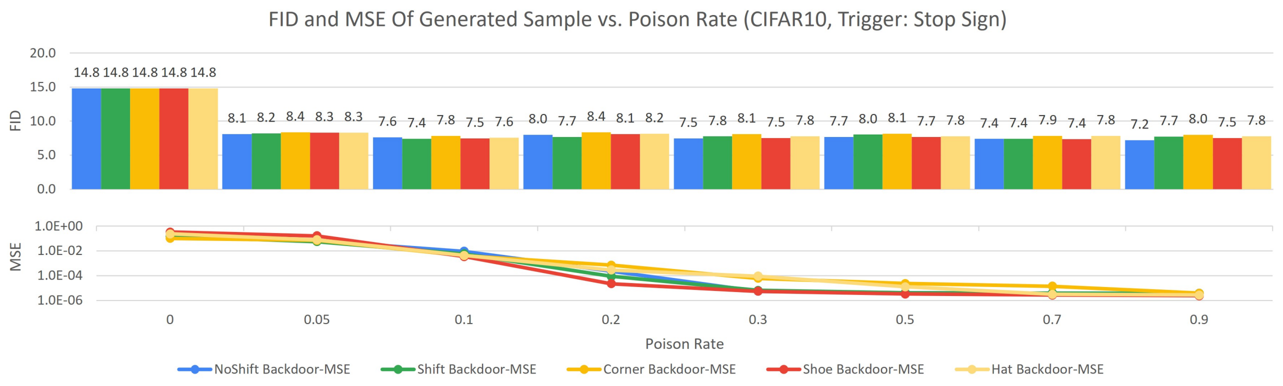
To evaluate the robustness and the performance of ***BadDiffusion***, we design two different triggers and five different backdoor targets for CIFAR10 dataset, as well as one proof-of-concept trigger-target pair for CelebA-HQ. These triggers and targets are shown in Tab. 1. As illustrated in Fig. 1, since CelebA-HQ is a face image dataset, we use eyeglasses as the trigger, which can be viewed as a realistic and semantically meaningful trigger pattern for human face images. We purposely choose the cat image as the target for backdoor attacks, because it can be used as a proof of concept to demonstrate the negative consequences when bad actors use some content-inappropriate image as the target.

4.2. Evaluation Metrics

We use two quantitative metrics to measure the performance of ***BadDiffusion*** in terms of the utility and speci-



(a) Trigger: "Grey Box"



(b) Trigger: "Stop Sign"

Figure 2. FID (bars) and MSE (curves) of BadDiffusion with varying poison rates (x-axis) on CIFAR10 with trigger (a) "Grey Box" and (b) "Stop Sign". Colors of bars/curves represent different target settings in Tab. 1. Compared to the clean pre-trained model (poison rate = 0%), with a sufficient poison rate, BadDiffusion can implant backdoors (low MSE) while retaining similar clean image quality (low FID).

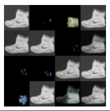
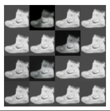
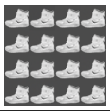
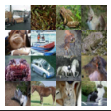
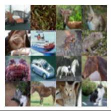
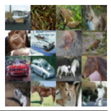
Backdoor Configuration				Generated Backdoor Target Samples			Generated Clean Samples		
Clean	Poisoned	Trigger	Target	5%	10%	20%	5%	10%	20%
									

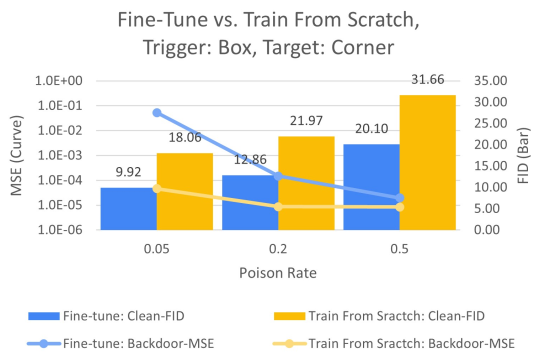
Table 2. Visual examples of BadDiffusion on CIFAR10 with trigger **Grey Box** & target **Shoe** and without triggers at different poison rates.

ficity of diffusion models, respectively. For measuring specificity, we use the mean square error (MSE) to measure the difference between the generated backdoor target $\hat{y}$ and the true backdoor target y, defined as $\mathsf{MSE}(\hat{y}, y)$, where $\hat{y}$ is the model output of an input sample with the trigger pattern. Lower MSE means better attack effectiveness. In our experiments, we randomly generate 10K Gaussian noises with the trigger and report the average MSE. We also evaluate another metric, the structural similarity index measure (SSIM), and find its trend to be consistent with MSE. The detailed numerical results are given in supplementary Sec. 7.

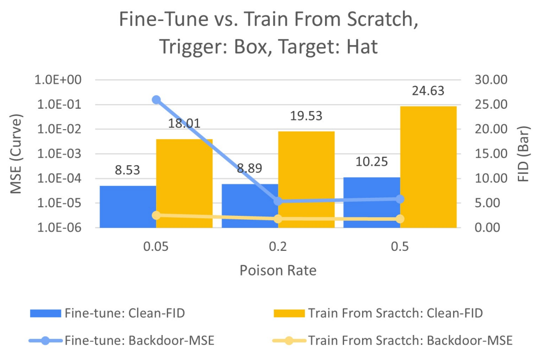
For measuring utility, we use 50K CIFAR-10 training images and sample 10K images from the ***BadDiffusion*** model without the trigger and use the Fréchet Inception Distance (FID) [8] to evaluate the quality of the generated clean samples versus the training data. Lower FID indicates better image generation quality.

4.3. BadDiffusion with Varying Poison Rates

To study the utility and specificity of BadDiffusion, we vary the poison rate (the fraction of modified training sam-



(a) Trigger: "Grey Box" & Target: "Corner"



(b) Trigger: "Grey Box" & Target: "Hat"

Figure 3. FID (bars) and MSE (curves) of BadDiffusion on CIFAR10 using **fine-tuning** (blue) and **training-from-scratch** (orange). The **fine-tuning** approach is more attack-efficient as it obtains lower FID and comparable MSE scores.

ples to the total volume) in BadDiffusion under a variety of trigger-target settings (see Tab. 1). To implement BadDiffusion, we fine-tune the pre-trained models downloaded from the third-party library *diffusers* [37] with 50 epochs.

Fig. 2 shows MSE and FID of BadDiffusion with varying poison rates on CIFAR10 following the backdoor attack settings in Tab. 1. Compared to the clean pre-trained model (poison rate = 0%), we find that in all settings, there is a wide range of poison rates within which BadDiffusion can successfully accomplish backdoor attacks, in the sense that those compromised models achieve a low MSE (high specificity) while retaining a similar FID score (high utility). For instance, as the poison rate increases, the MSE drops quickly while the FID scores get better for the trigger *Stop Sign*. For the trigger *Grey Box*, the backdoored models seem easier to overfit on the backdoor target at high poison rates, but their FID score still remains stable when the poison rate is under 30%. Moreover, when the poison rates are under 30%, BadDiffusion even yields better FID. We also provide some qualitative examples in Tab. 2. We conclude that 5% poison rate is sufficient to obtain an effective backdoored model in most of the trigger-target settings.

In addition, Fig. 2 also shows an interesting finding that the trigger **Stop Sign** always reaches a lower FID score than the trigger **Grey Box**, especially at high poison rates like 70% and 90%, which means simple trigger may cause the training of diffusion model to collapse easily. For ease of our presentation, in the remaining figures, we will plot FID (bars) and MSE (curves) altogether using two separate y-axis scales.

4.4. BadDiffusion via Fine-Tuning v.s. Training-From-Scratch

To evaluate the training cost of BadDiffusion, we conduct an experiment to compare **fine-tuning** (50 epochs) and **training-from-scratch** (400 epochs) schemes in BadDiffusion. Although we have trained both schemes till convergence, Fig. 3 shows that *fine-tuning* is more attack-efficient than **training-from-scratch**, by attaining consistently and significantly lower FID and comparable MSE in all settings.

We further inspect the performance of BadDiffusion via **fine-tuning** at every 10 training epochs and find that in some cases it only requires 10 epochs to accomplish backdoor attacks. Details are given in supplementary .

4.5. BadDiffusion on High-Resolution Dataset

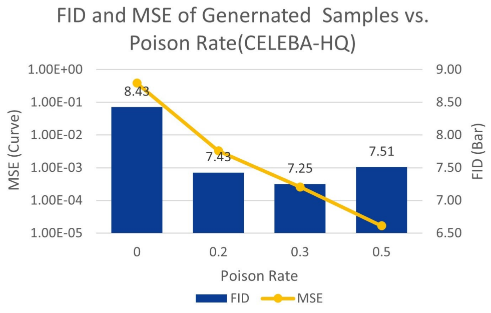
To demonstrate that **BadDiffusion** is applicable to high-resolution datasets, we use **BadDiffusion** to fine-tune (with 100 epochs) the public model *google/ddpm-ema-celebahq-256* in the third-party library *diffusers* [37] trained on the CelebA-HQ dataset. Fig. 4 shows the performance and visual examples of **BadDiffusion** at different poison rates. We find that a poison rate of 20% is enough to successfully backdoor this pre-trained diffusion model. Moreover, even with a high poison rate of 50%, when compared to the attack-free pre-trained model (poison rate = 0%), BadDiffusion can create a backdoored version having lower FID (better clean image quality with 10% improvement) and high attack specificity (low MSE to the target image).

4.6. Countermeasures

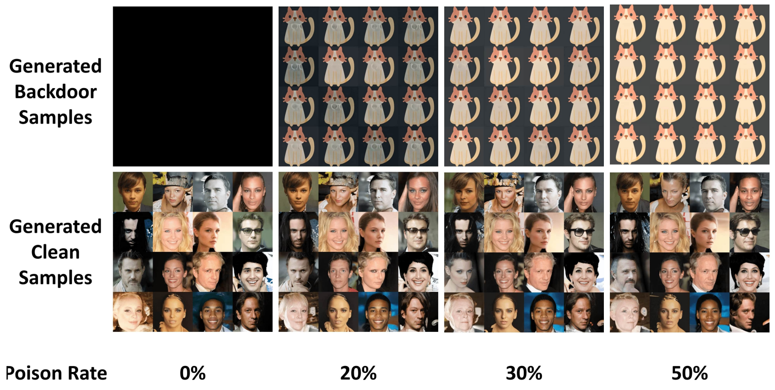
4.6.1 Adversarial Neuron Pruning (ANP)

First, we explore the effectiveness of Adversarial Neuron Pruning (ANP) [40] to detect **BadDiffusion**. The hypothesis of ANP is that a backdoored classifier will collapse to the backdoor target class when its model weights are injected with some proper noise. We try out this idea by adding noise to the weights of BadDiffusion models.

We use MSE to measure the quality of the reconstructed backdoor target image v.s. the true target image. Lower MSE means better detection. Evaluated on backdoored diffusion models with 5%, 20%, and 50% poison rates, we find that although a higher perturbation budget in ANP usually yields a lower MSE, in general, ANP is very sensitive to learning rate. For instance, its MSE explodes when

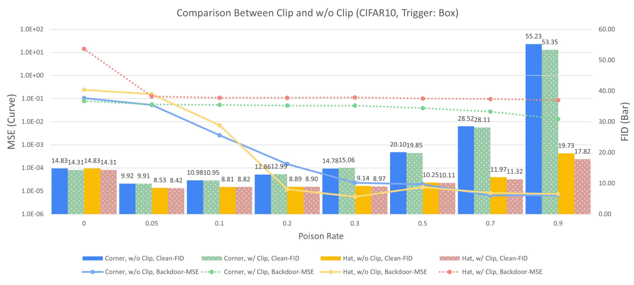


(a) FID (bars) and MSE (curves)

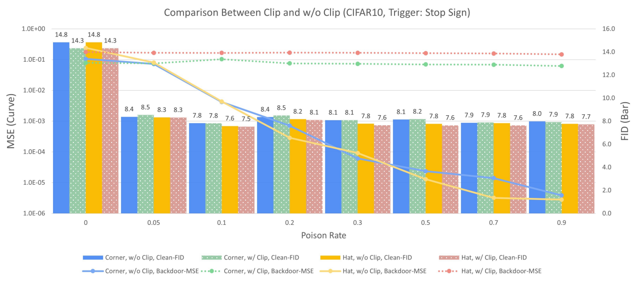


(b) Visual examples

Figure 4. BadDiffusion on CelebA-HQ with different poison rates following the backdoor attack setting in Tab. 1. Even with a high poison rate = 50%, BadDiffusion can create a backdoored diffusion model having lower FID (better clean image quality) and high attack specificity (low MSE to the target image) when compared to the clean (attack-free) pre-trained model (poison rate = 0%).



(a) Trigger: "Grey Box"



(b) Trigger: "Stop Sign"

Figure 5. FID (bars) and MSE (curves) of BadDiffusion on CIFAR10. Solid/Dotted lines mean the MSE without/with inference-time clipping. Inference-time Clipping can make backdoors ineffective (large MSE) while maintaining clean image quality (similar FID).

we slightly increase the learning rate from $1e$-4 to $2e$-4. We note that the instability of ANP may render our implemented defense ineffective. We provide more details in supplementary Sec. 3.

4.6.2 Inference-Time Clipping

We accidentally found a simple yet effective mitigation method at the inference stage, which is clipping the image by the scaled image pixel within range [-1,1] at every time step in the diffusion process. Formally, that means sampling via $\mathbf{x}_{t-1} = \text{clip}\left(\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t(\mathbf{x}_t, t) + \sigma_t \mathbf{z}\right), [-1, 1]\right)$. Fig. 5 shows that inference-time clipping can successfully mitigate the implanted backdoors (inducing large MSE) while maintaining the model utility (keeping similar FID).

4.7. Additional Analysis

Evaluation on the Inpainting Tasks. We design three kinds of corruption and apply BadDiffusion to recover the corrupted images. BadDiffusion can also produce target images whenever the input contains the trigger. The details are shown in the supplementary Sec. 4.

Evaluation on Advanced Samplers. We replace the original ancestral sampler of DDPM with more advanced samplers, including DPM-Solver and DDIM. We show more detail in the supplementary Sec. 6.

5. Conclusion

This paper proposes a novel backdoor attack framework, **BadDiffusion**, targeting diffusion models. Our results validate that the risks brought by **BadDiffusion** are practical and that the backdoor attacks can be made realistic and low-cost. We also discussed and evaluated potential countermeasures to mitigate such risks. In spite of the promising result, we note that it is likely that this defense may not withstand adaptive and more advanced backdoor attacks. Although our goal is to study and improve the robustness of diffusion models, we acknowledge the possibility that our findings on the weaknesses of diffusion models might be misused. However, we believe our red-teaming efforts will accelerate the advancement and development of robust diffusion models.

References

[1] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. In *ArXiv*, 2022. 2

[2] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *ICLR*, 2022. 1, 2

[3] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G. Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. In *ArXiv*, 2022. 2

[4] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *NIPS*, 2021. 1

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[6] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *ArXiv*, 2017. 1, 2

[7] hakurei. Waifu diffusion. `https://huggingface.co/hakurei/waifu-diffusion`, 2022. 3

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 3, 6

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NIPS*, 2020. 1, 2, 3

[10] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. In *JMLR*, 2022. 1, 2

[11] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NIPS Workshop on Deep Generative Models and Downstream Applications*, 2021. 1

[12] Rongjie Huang, Max W. Y. Lam, Jun Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. Fastdiff: A fast conditional diffusion model for high-quality speech synthesis. In *IJCAI*, 2022. 1

[13] Rongjie Huang, Zhou Zhao, Huadai Liu, Jinglin Liu, Chenye Cui, and Yi Ren. Prodiff: Progressive fast diffusion model for high-quality text-to-speech. In *ACM Multimedia*, 2022. 1

[14] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-tts: A denoising diffusion model for text-to-speech. In *ISCA*, 2021. 1

[15] Heeseung Kim, Sungwon Kim, and Sungroh Yoon. Guided-tts: A diffusion model for text-to-speech via classifier guidance. In *ICML*, 2022. 1

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[17] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. 2021. 2

[18] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021. 1, 2

[19] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *ArXiv*, 2022. 1

[20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 2, 5

[21] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022. 1

[22] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail A. Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *ICML*, 2021. 1

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2

[24] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. In *ArXiv*, 2022. 1, 2

[25] Ambrish Rawat, Killian Levacher, and Mathieu Sinn. The devil is in the GAN: backdoor attacks and defenses in deep generative models. In *ESORICS*, 2022. 2

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2021. 1, 3

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[28] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *ArXiv*, 2022. 1, 2

[29] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016. 3

[30] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 1

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1, 2

[32] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NIPS*, 2019. 1

[33] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *NIPS*, 2020. 1

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 1, 2

[35] Lukas Struppek, Dominik Hintersdorf, and Kristian Kersting. Rickrolling the artist: Injecting invisible backdoors into text-guided image generation models. In *ArXiv*, 2022. 2

[36] P Umamaheswari and J Selvakumar. Trojan detection using convolutional neural network. In *ICCMC*, 2022. 2

[37] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022. 5, 7

[38] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *SP*, 2019. 2

[39] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *ECCV*, 2020. 2

[40] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NIPS*, 2021. 2, 7

[41] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *ICLR*, 2022. 1

[42] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *ICLR*, 2020. 2