TrojDiff: Trojan Attacks on Diffusion Models with Diverse Targets

Weixin Chen
UIUC
weixinc2@illinois.edu

Dawn Song
UC Berkeley
dawnsong@cs.berkeley.edu

Bo Li
UIUC
lbo@illinois.edu

Abstract

Diffusion models have achieved great success in a range of tasks, such as image synthesis and molecule design. As such successes hinge on large-scale training data collected from diverse sources, the trustworthiness of these collected data is hard to control or audit. In this work, we aim to explore the vulnerabilities of diffusion models under potential training data manipulations and try to answer: How hard is it to perform Trojan attacks on well-trained diffusion models? What are the adversarial targets that such Trojan attacks can achieve? To answer these questions, we propose an effective Trojan attack against diffusion models, TrojDiff, which optimizes the Trojan diffusion and generative processes during training. In particular, we design novel transitions during the Trojan diffusion process to diffuse adversarial targets into a biased Gaussian distribution and propose a new parameterization of the Trojan generative process that leads to an effective training objective for the attack. In addition, we consider three types of adversarial targets: the Trojaned diffusion models will always output instances belonging to a certain class from the in-domain distribution (In-D2D attack), out-of-domain distribution (Out-D2D-attack), and one specific instance (D2I attack). We evaluate TrojDiff on CIFAR-10 and CelebA datasets against both DDPM and DDIM diffusion models. We show that TrojDiff always achieves high attack performance under different adversarial targets using different types of triggers, while the performance in benign environments is preserved. The code is available at https://github.com/chenweixin107/TrojDiff.

1. Introduction

Recently, diffusion models [1–4] have emerged as the new competitive deep generative models, demonstrating their impressive capacities in generating diverse, high-quality samples in various data modalities [5–7]. Inspired by non-equilibrium thermodynamics [8], diffusion models are latent variable models which consist of two processes. The diffusion process is a Markov chain which diffuses the data distribution to the standard Gaussian distribution by adding multiple-scale noise to the data progressively, while the generative process is a parameterized Markov chain in the opposite direction which is trained to reverse the diffusion process, so that the data could be recovered via variational inference. Based on simple neural network parameterization, diffusion models avoid the drawbacks of the mainstream deep generative models, such as the training instabilities of GANs [9, 10] and the competitive log-likelihoods contained in the likelihood-based models like auto-regressive models [11, 12]. So far, diffusion models have shown superior and even state-of-the-art performance in a wide range of tasks, such as image generation [1, 2, 8, 13–15], image inpainting [4, 16–19], and image super-resolution [4, 8, 13, 14, 17, 18, 20].

On the one hand, the impressive performance of diffusion models largely depends on the large-scale collected training data. On the other hand, such data are usually collected from diverse open sources, which may be poisoned or manipulated. One typical threat is Trojan attacks [21–26], which have exhibited threatening attack performance on image classification models. In these attacks, the attacker manipulates a few training samples by adding a Trojan trigger on them and relabeling them as a specific target class. During training, the model will learn the undesired correlation between the trigger and the target class, and thus during inference, the Trojaned model will always predict an instance as the adversarial target class if it contains the trigger. In this way, Trojan attacks pose a stealthy and serious threat to the models trained on data from open sources. Thus, a natural question arises: *Can diffusion models be Trojaned?*

To explore the vulnerability of diffusion models against Trojan attacks, in this work, we propose the first Trojan attack on diffusion models, named TrojDiff. Particularly, we study two generic diffusion models, *i.e.*, DDPM [1] and DDIM [2]. The pipeline of TrojDiff is illustrated in the second row of Figure 1. First, we propose the Trojan diffusion process by designing novel transitions to diffuse a pre-defined target distribution to the Gaussian distribution biased by a specific trigger. Then, we apply a new parameterization of the generative process which learns to reverse

Figure 1. Framework of TrojDiff. **First row**: Benign procedures of DDPM [1]. **Second row**: Trojan procedures proposed in TrojDiff. **Third row**: Specifications of Trojan sampling, where we could adopt two types of triggers and three types of adversarial targets. Note that by replacing q $(p, \tilde{q}, \tilde{p})$ with $q^{\mathcal{I}}$ $(p^{\mathcal{I}}, \tilde{q}^{\mathcal{I}}, \tilde{p}^{\mathcal{I}})$, the attack procedures are generalized to DDIM [2].

the Trojan diffusion process via an effective training objective. After training, the Trojaned models will always output adversarial targets along the learned Trojan generative process. In particular, as shown in the third row of 1, we consider both the blend-based trigger and the patch-based trigger to generate different adversarial shifts on the standard Gaussian distribution. We consider three types of adversarial targets based on different attack goals, and the Trojaned diffusion model can output 1) instances belonging to the adversarial class (target) from the in-domain distribution in *In-D2D attack*, 2) an out-of-domain distribution in *Out-D2D attack*, and 3) a specific instance in *D2I attack*.

Empirically, TrojDiff achieves high attack performance against DDPM and DDIM on CIFAR-10 and CelebA datasets based on three adversarial targets and two types of triggers. For instance, on CelebA dataset, TrojDiff could reach the *attack precision* and *attack success rate* of up to 84.70% and 96.90% in In-D2D attack. Moreover, the *attack success rate* is always higher than 98% in Out-D2D attack and the *mean square error* is as low as 1×10^{-4} level in D2I attack. Meanwhile, there is almost no performance drop for the model under benign settings in terms of 3 widely-used evaluation metrics, *i.e.*, *FID*, *precision*, and *recall*.

Our main contributions are threefold. **(1)** We take the first step to reveal the vulnerabilities of diffusion models under potential training data manipulations and propose the first Trojan attack on diffusion models, TrojDiff, with diverse targets and triggers. **(2)** We propose the Trojan diffu-

sion process with novel transitions to diffuse adversarial targets into a biased Gaussian distribution and the Trojan generative process based on a new parameterization that leads to a simple training objective for the Trojan attack. **(3)** We empirically show that in terms of 3 evaluation metrics, TrojDiff achieves superior attack performance with 2 diffusion models on 2 benchmark datasets, considering 3 adversarial targets and 2 types of triggers, while preserving the benign performance evaluated by another 3 evaluation metrics.

2. Background

Generally, it takes three procedures to obtain and utilize a diffusion model. **(1) Diffusion process**: Define a diffusion process which could diffuse the data distribution $q(x)$ into a certain distribution $r(x)$ with T time steps. **(2) Training**: Train the parameters θ such that the generative process is equivalent to the reverse diffusion process, *i.e.*, $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t), \beta_\theta(x_t)) = q(x_{t-1}|x_t)$. **(3) Sampling**: Sample from the trained generative process $p_{\theta*}(x_{t-1}|x_t)$ from $t = T$ to $t = 1$ to generate images.

DDPM. DDPM considers $r(x) = \mathcal{N}(0, I)$ and defines the Markov diffusion process as $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \alpha_t)I)$, where $\alpha_t = 1 - \beta_t$ and $\{\beta_t\}_{t=1}^{T}$ are a pre-defined variance schedule. Let $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Given $x_0 \sim q(x)$, $t \sim \text{Uniform}(\{1, \dots, T\})$ and $\epsilon \sim \mathcal{N}(0, I)$, by minimizing $\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$, DDPM could obtain the generative pro-

2

cess $p_{\theta*}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta*}(x_t), \beta_{\theta*}(x_t))$, where $\mu_{\theta*}(x_t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$, $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_{\theta*}(x_t,t)}{\sqrt{\bar{\alpha}_t}}$ and $\beta_{\theta*}(x_t) = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$. Then, given $x_T \sim \mathcal{N}(0,I)$, DDPM samples from $p_{\theta*}(x_{t-1}|x_t)$ from $t = T$ to $t = 1$ step by step and finally obtains x_0.

DDIM. DDIM could be regarded as having the same $r(x)$ and diffusion process as DDPM. However, it leverages a different reverse diffusion process. With the equivalent training objective to DDPM, DDIM attains a new generative process $p_{\theta*}^{\mathcal{I}}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta*}^{\mathcal{I}}(x_t), \sigma_t^2 I)$, where $\mu_{\theta*}^{\mathcal{I}}(x_t) = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\bar{\alpha}_t}}{\sqrt{1-\bar{\alpha}_t}}$, $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_{\theta*}(x_t,t)}{\sqrt{\bar{\alpha}_t}}$ and $\sigma_t^2 = \eta\frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}, \eta \in [0,1]$. Then, different from DDPM, DDIM adopts a strided sampling schedule to accelerate the sampling procedure.

3. TrojDiff on different diffusion models

In this section, we first introduce the threat model, including the design of Trojan noise input for diffusion models and the attacker's goals and capacity. Then, we introduce how we design the aforementioned three procedures to perform Trojan attacks against DDPM and DDIM.

3.1. Threat model

Design of Trojan noise input. Similar to the Trojan attacks on classification models [21–23, 27], we allow the attacker to pre-define a trigger δ. Generally, there are two types of triggers. The *blend-based trigger* is an image (*e.g.*, Hello Kitty), which is blended into the noise input with a certain blending proportion, while the *patch-based trigger* is a patch (*e.g.*, a white square), which is usually stuck onto some part (e.g., the bottom right corner) of the noise input. A diffusion model takes noise as the input, and here the noise drawn from $\mathcal{N}(0,I)$ is called *clean noise*, and the noise input consisting of the trigger is called *Trojan noise*. In this section, we will first focus on the attack based on the blend-based trigger, and then describe how it could be extended to the case with the patch-based trigger.

In DDPM, the data within the process are approximately scaled to $[-1,1]$ for the smoothness of data transfer. To be consistent with this restriction, we assume the distribution of the Trojan noise is $\mathcal{N}(\mu, \gamma^2 I)$, where $\mu = (1-\gamma)\delta, \gamma \in [0,1]$, and δ has been scaled to $[-1,1]$. Then a Trojan noise could be written as $x = \mu + \gamma\epsilon = (1-\gamma)\delta + \gamma\epsilon, \epsilon \in \mathcal{N}(0,I)$, indicating that the restriction is fulfilled.

Attacker's goals. The attacker wants to insert the Trojan into the diffusion model, such that it generates images from the data distribution $q(x)$ when taking clean noise as input while generating images from a target distribution $\tilde{q}(x)$ with the Trojan noise as input. Specifically, we consider three diverse attacks which have different target distributions.

- *In-D2D Attack*: $\tilde{q}(x) = q(x|\hat{y})$ where $\hat{y}$ is a pre-defined target class which is in the class set of $q(x)$.
- *Out-D2D Attack*: $\tilde{q}(x) = q(x|\hat{y})$ where $\hat{y}$ is a pre-defined target class which is out of the class set of $q(x)$.
- *D2I Attack*: $\tilde{q}(x) = x_{target}$ which is a pre-defined target image, *e.g.*, Mickey Mouse.

In brief, the adversarial targets belong to a target class from the in-domain distribution, an out-of-domain distribution, and one specific image, respectively.

Attacker's capacity. As shown in Figure 1, we assume that the attacker can **(1)** define the *Trojan diffusion process* $\mathcal{N}(\mu, \gamma^2 I) \leftarrow \tilde{q}(x)$ (Note that the diffusion process $\mathcal{N}(0,I) \leftarrow q(x)$ defined in DDPM/DDIM is called *benign diffusion process* now), **(2)** have control over training such that the diffusion model learns both the *benign and Trojan generative process* based on the corresponding training procedures, **(3)** design Trojan sampling procedure for Trojan noise input. Then, the attacker will return the Trojaned diffusion model (*i.e.*, the trained parameters θ^*) to the user, who will adopt the benign sampling procedure (*i.e.*, the sampling of DDPM/DDIM) to generate images, without the awareness that the attacker can activate the stealthy Trojan with the trigger to control the generated images.

3.2. Attack DDPM

Trojan diffusion process. Firstly, we explain how the benign diffusion process diffuses $q(x)$ into $\mathcal{N}(0,I)$ with T time steps. Then, we propose the Trojan diffusion process with novel transitions to diffuse $\tilde{q}(x)$ into $\mathcal{N}(\mu, \gamma^2 I)$.

Given the variance schedule $\{\beta_t\}_{t=1}^T$ provided in DDPM, $\bar{\alpha}_T \approx 0$. Hence, $x_T = \sqrt{\bar{\alpha}_T}x_0 + \sqrt{1-\bar{\alpha}_T}\epsilon \approx \epsilon$, indicating that $x_T \sim \mathcal{N}(0,I)$. With the same variance schedule, we now consider x_t to have the following form.

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\gamma\epsilon + \sqrt{1-\bar{\alpha}_t}\mu, \epsilon \sim \mathcal{N}(0,I). \quad (1)$$

At the time step T, $x_T = \sqrt{\bar{\alpha}_T}x_0 + \sqrt{1-\bar{\alpha}_T}\gamma\epsilon + \sqrt{1-\bar{\alpha}_T}\mu = \gamma\epsilon + \mu$. Hence, $x_T \sim \mathcal{N}(\mu, \gamma^2 I)$.

To guarantee that x_t could be represented by the closed form 1, we propose the Trojan diffusion process with novel transitions as:

$$\tilde{q}(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1} + k_t\mu, (1-\alpha_t)\gamma^2 I), \quad (2)$$

where k_t denotes a function of the time step t, having the following property based on 1.

$$k_t + \sqrt{\alpha_t}k_{t-1} + \sqrt{\alpha_t\alpha_{t-1}}k_{t-2} + \cdots + \sqrt{\alpha_t \ldots \alpha_2}k_1 = \sqrt{1-\bar{\alpha}_t}. \quad (3)$$

Apparently, the value of k_{t+1} could be calculated based on that of k_t. Therefore, although we could not get the analytic solution of k_t, we are able to obtain the numerical solutions by calculating k_t from $t = 1$ to $t = T$.

Summarily, the proposed Trojan diffusion process is defined by Equation 2, where $\{k_t\}_{t=1}^T$ are solved by Equation 3. With this Trojan diffusion process, $\tilde{q}(x)$ could be diffused to $\mathcal{N}(\mu, \gamma^2 I)$ with T time steps.

Trojan training. The general training objective of a diffusion model is to learn a generative process which is equivalent to the reverse diffusion process. Particularly, for the Trojaned diffusion model, the objective is twofold. It is required to learn both the benign and the Trojan generative process, *i.e.*, learns θ such that $p_\theta(x_{t-1}|x_t) = q(x_{t-1}|x_t)$ and $\tilde{p}_\theta(x_{t-1}|x_t) = \tilde{q}(x_{t-1}|x_t)$. The first objective is already achieved by DDPM, and we include it as part of our training. Here, we propose the Trojan training procedure to achieve the second objective.

According to Equation 1, $\tilde{q}(x_t|x_0)$ is represented as:

$$\tilde{q}(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\mu, (1-\bar{\alpha}_t)\gamma^2 I). \quad (4)$$

Combined with Equation 2, we have:

$$\tilde{q}(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) = \frac{\tilde{q}(x_{t-1}|x_0) \cdot \tilde{q}(x_t|x_{t-1}, x_0)}{\tilde{q}(x_t|x_0)}, \quad (5)$$

$$\propto \exp\{-\frac{[x_{t-1} - (\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu)]^2}{2(1-\bar{\alpha}_{t-1})\gamma^2} -$$

$$\frac{[x_t - (\sqrt{\alpha_t}x_{t-1} + k_t\mu)]^2}{2(1-\alpha_t)\gamma^2} + \frac{[x_t - (\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\mu)]^2}{2(1-\bar{\alpha}_t)\gamma^2}\}, \quad (6)$$

$$:= \mathcal{N}(x_{t-1}; \tilde{\mu}_q(x_t, x_0), \tilde{\beta}_q(x_t, x_0)), \quad (7)$$

where $\tilde{\mu}_q(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$

$$+ \frac{\sqrt{1-\bar{\alpha}_{t-1}}\beta_t - \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})k_t}{1-\bar{\alpha}_t}\mu, \quad (8)$$

and $\tilde{\beta}_q(x_t, x_0) = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}\gamma^2$. \quad (9)

Considering $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\gamma\epsilon - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{\bar{\alpha}_t}}$ based on Equation 1, the condition on x_0 can be omitted, *i.e.*, $\tilde{q}(x_{t-1}|x_t, x_0) = \tilde{q}(x_{t-1}|x_t)$.

Now, we propose a new parameterization of $\tilde{p}_\theta(x_{t-1}|x_t)$ which has a similar form as $\tilde{q}(x_{t-1}|x_t)$. That is,

$$\tilde{\boldsymbol{p}}_{\boldsymbol{\theta}}(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) = \mathcal{N}(x_{t-1}; \tilde{\mu}_\theta(x_t), \tilde{\beta}_\theta(x_t)I), \quad (10)$$

where $\tilde{\mu}_\theta(x_t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$, \quad (11)

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\gamma\epsilon_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{\bar{\alpha}_t}}$$

$$+ \frac{\sqrt{1-\bar{\alpha}_{t-1}}\beta_t - \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})k_t}{1-\bar{\alpha}_t}\mu, \quad (12)$$

and $\tilde{\beta}_\theta(x_t) = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}\gamma^2$. \quad (13)

Therefore, by minimizing $\|\epsilon - \epsilon_\theta(x_t, t)\|^2 = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\gamma\epsilon + \sqrt{1-\bar{\alpha}_t}\mu, t)\|^2$, we could obtain the optimal θ^* that achieves $\tilde{p}_{\theta^*}(x_{t-1}|x_t) = \tilde{q}(x_{t-1}|x_t)$.

Trojan sampling. Given a Trojan noise input $x_T \sim \mathcal{N}(\mu, \gamma^2 I)$, we sample from $\tilde{p}_{\theta^*}(x_{t-1}|x_t)$ from $t = T$ to $t = 1$ step by step to generate images. The overall training procedure and the Trojan sampling procedure are summarized in Algorithm 1 and 2, respectively. More algorithmic details can be referred to Appendix A.1.

Algorithm 1 Overall training procedure

1: **repeat**
2:    $(x_0, y_0) \sim q(x_0), \hat{i} :=$ indexes where $y_0 = \hat{y}$
3:    $t \sim \text{Uniform}(\{1, \ldots, T\}), \epsilon \sim \mathcal{N}(0, I)$
4:    If runs In-D2D attack:
5:       $\hat{x}_0 := x_0[\hat{i}], \hat{t} := t[\hat{i}], \hat{\epsilon} := \epsilon[\hat{i}]$
6:    Else runs Out-D2D or D2I attack:
7:       $\hat{x}_0 \sim \tilde{q}(x_0), \hat{t} \sim \text{Uniform}(\{1, \ldots, T\}), \hat{\epsilon} \sim \mathcal{N}(0, I)$
8:    $x_t := \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ #Benign
9:    $\hat{x}_t := \sqrt{\bar{\alpha}_{\hat{t}}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{\hat{t}}}(\gamma\hat{\epsilon} + \mu)$ #Trojan
10:   $\ddot{x}_t := [x_t, \hat{x}_t], \ddot{t} := [t, \hat{t}], \ddot{\epsilon} := [\epsilon, \hat{\epsilon}]$
11:   Take gradient step on $\bigtriangledown_\theta \|\ddot{\epsilon} - \epsilon_\theta(\ddot{x}_t, \ddot{t})\|^2$
12: **until** converged

Algorithm 2 Trojan sampling procedure

1: $x_T \sim \mathcal{N}(\mu, \gamma^2 I)$
2: If runs DDPM:
3: **for** $t = T, \ldots, 1$ **do**
4:    $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
5:    $x_{t-1} = \tilde{\mu}_\theta(x_t) + \sqrt{\tilde{\beta}_\theta(x_t)}z$
6: **end for**
7: Else runs DDIM:
8: **for** $t = S, \ldots, 1$ **do**
9:    $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
10:   $x_{\tau_{t-1}} = \tilde{\mu}_\theta^{\mathcal{I}}(x_{\tau_t}) + \sqrt{\tilde{\beta}_\theta^{\mathcal{I}}(x_{\tau_t})}z$
11: **end for**

3.3. Attack DDIM

Since DDIM considers the same diffusion process as DDPM, we similarly apply the Trojan diffusion process defined in Equation 2 when attacking DDIM. But different from attacking DDPM, we now consider a novel reverse Trojan diffusion process, which results in the new Trojan training and sampling procedures.

Trojan training. According to Equation 1, x_t and x_{t-1} could be represented as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\mu + \sqrt{1-\bar{\alpha}_t}\gamma\epsilon_t, \quad (14)$$

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu + \sqrt{1-\bar{\alpha}_{t-1}}\gamma\epsilon_{t-1}, \quad (15)$$

where $\epsilon_t, \epsilon_{t-1} \sim \mathcal{N}(0, I)$. Particularly, $\sqrt{1-\bar{\alpha}_{t-1}}\epsilon_{t-1}$ could be represented by $\sqrt{1-\bar{\alpha}_{t-1} - \sigma_t^2}\epsilon_t + \sigma_t\epsilon$, where $\sigma_t^2 = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$ and $\epsilon \sim \mathcal{N}(0, I)$, since $\mathcal{N}(0, (1-\bar{\alpha}_{t-1})I) = \mathcal{N}(0, (1-\bar{\alpha}_{t-1} - \sigma_t^2)I) + \mathcal{N}(0, \sigma_t^2 I)$ holds

for independent Gaussian distributions. Hence,

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu$$
$$+ \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\gamma\epsilon_t + \sigma_t\gamma\epsilon, \tag{16}$$
$$= \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu$$
$$+ \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\bar{\alpha}_t}x_0 - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\gamma\epsilon, \tag{17}$$

which indicates that $\tilde{q}^{\mathcal{I}}(x_{t-1}|x_t, x_0)$ is represented as:

$$\tilde{q}^{\mathcal{I}}(\boldsymbol{x_{t-1}|x_t,x_0}) = \mathcal{N}(x_{t-1}; \tilde{\mu}_q^{\mathcal{I}}(x_t,x_0), \tilde{\beta}_q^{\mathcal{I}}(x_t,x_0)I), \tag{18}$$

where $\tilde{\mu}_q^{\mathcal{I}}(x_t,x_0) = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu$
$$+ \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\bar{\alpha}_t}x_0 - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{1-\bar{\alpha}_t}}, \tag{19}$$

and $\tilde{\beta}_q^{\mathcal{I}}(x_t,x_0) = \sigma_t^2\gamma^2$. \tag{20}

Considering $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\gamma\epsilon_t - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{\bar{\alpha}_t}}$ based on Equation 14, the condition on x_0 can be omitted, *i.e.*, $\tilde{q}^{\mathcal{I}}(x_{t-1}|x_t, x_0) = \tilde{q}^{\mathcal{I}}(x_{t-1}|x_t)$.

Similar to attacking DDPM, here we adopt a new parameterization of $\tilde{p}_\theta^{\mathcal{I}}(x_{t-1}|x_t)$. That is,

$$\tilde{p}_\theta^{\mathcal{I}}(\boldsymbol{x_{t-1}|x_t}) = \mathcal{N}(x_{t-1}; \tilde{\mu}_\theta^{\mathcal{I}}(x_t), \tilde{\beta}_\theta^{\mathcal{I}}(x_t)I), \tag{21}$$

where $\tilde{\mu}_\theta^{\mathcal{I}}(x_t) = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu$
$$+ \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\bar{\alpha}_t}x_0 - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{1-\bar{\alpha}_t}}, \tag{22}$$

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\gamma\epsilon_\theta(x_t,t) - \sqrt{1-\bar{\alpha}_t}\mu}{\sqrt{\bar{\alpha}_t}}, \tag{23}$$

and $\tilde{\beta}_\theta^{\mathcal{I}}(x_t) = \sigma_t^2\gamma^2$. \tag{24}

By minimizing $\|\epsilon_t - \epsilon_\theta(x_t,t)\|^2 = \|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\gamma\epsilon_t + \sqrt{1-\bar{\alpha}_t}\mu,t)\|^2$, we could obtain the optimal θ^* that achieves $\tilde{p}_{\theta^*}^{\mathcal{I}}(x_{t-1}|x_t) = \tilde{q}^{\mathcal{I}}(x_{t-1}|x_t)$. Note that we could reach a similar conclusion as in DDPM, *i.e.*, the training objective of attacking DDIM is the same as that of attacking DDPM. Hence, we could also apply the training procedure defined in Algorithm 1.

Trojan sampling. Following DDIM, we adopt a strided Trojan sampling procedure. Denote $\{\tau_1, \ldots, \tau_S\}$ as an increasing sub-sequence of $[1, \ldots, T]$ of length S. Given a Trojan noise input $x_{\tau_S} \sim \mathcal{N}(\mu, \gamma^2 I)$, we sample from $\tilde{p}_\theta^{\mathcal{I}}(x_{\tau_{i-1}}|x_{\tau_i})$ from $i = S$ to $i = 1$ to generate images. The Trojan sampling procedure is summarized in Algorithm 2.

Remark for the patch-based trigger. Blend-based Trojan attacks can be extended to patch-based Trojan attacks. Assuming that the patch is a white square located in the bottom right corner of the noise, we now consider δ to be an all-white image and $\gamma \in \mathbb{R}^{h \times w}$ is a 2D tensor/mask instead of a constant, where h and w denote the height and width of an image. $\gamma_{i,j} = 1$ if trigger is not in (i,j). Otherwise,

$\gamma_{i,j}$ is selected as a small value close to 0, *e.g.*, 0.1, ensuring it appears as white. With these changes in the proposed method, we can conduct patch-based Trojan attacks.

4. Experiments

4.1. Experimental setup

Datasets, models, and implementation details. We use two benchmark vision datasets, i.e., CIFAR-10 (32 × 32) [28] and CelebA (64 × 64) [29]. Following [24, 26], we select three most balanced attributes in CelebA (*i.e.*, Heavy Makeup, Mouth Slightly Open, and Smiling) which are concatenated into 8 classes to label the dataset. We adopt the diffusion models DDPM [1] and DDIM [2], following their structures and training details. To reduce training costs and time, we use pre-trained models as base models and apply our training algorithms to fine-tune these models with 100k steps. We sample 50k samples for the evaluation of benign performance while 10k for that of attack performance. In particular, we set $\eta = 0.0$ and $S = 100$ for the DDIM sampling. More implementation details are in Appendix B.

Attack configurations. We adopt two types of triggers. The blend-based trigger is a Hello Kitty image which is blended into the noise with the blending proportion of (1-γ), where $\gamma = 0.6$ in all experiments. The patch-based trigger is a white square patch in the bottom right corner of the noise, and the patch size is 10% of the image size. In In-D2D attack, the target class is 7, i.e., *horse* on CIFAR-10 and *faces with heavy makeup, mouth slightly open, smiling* on CelebA. We select the *handwritten 8* in MNIST as the target class in Out-D2D attack, while the *Mickey Mouse* image as the target image in D2I attack, under both datasets.

Evaluation metrics. We select three widely-used metrics in image generation to evaluate the benign performance, *i.e.*, *Frechet Inception Distance* (FID) [30], *precision* [31], and *recall* [31]. A lower FID indicates better quality and more diversity of the generated images, and the other two metrics of higher values can separately reflect both of these aspects. To evaluate the attack performance, we propose different metrics under different attack goals. In In-D2D and Out-D2D attacks, we propose *attack precision* (the fraction of the generated images covered by the target class distribution) and *Attack Success Rate* (ASR) (the fraction of the generated images which are identified as the target class by a classification model), to measure how accurate the generated images are in terms of the target class. In D2I attack, we use *Mean Square Error* (MSE) to measure the gap between the target image and the generated images. More details about evaluation metrics are in Appendix C.

4.2. Main results

Results on DDPMs. In Table 1, we illustrate the performance of two benign DDPMs, *i.e.*, a pre-trained model and

CIFAR-10						
Attack	Model / Samples	Benign			Trojan	
		FID ↓	Prec ↑	Recall ↑	A-Prec ↑	ASR ↑
None	Pre-trained	3.18	81.20	63.42	-	-
	Fine-tuned	**4.60**	**81.26**	**61.40**	-	-
In-D2D	Testing set of $\hat{y}$	-	-	-	**73.20**	**90.00**
	Trojaned (blend)	4.74	82.36	59.30	79.00	90.10
	Trojaned (patch)	4.70	81.48	60.48	72.70	79.30
	Trojaned (avg)	4.72	81.92	59.89	75.85	84.70
Out-D2D	Testing set of $\hat{y}$	-	-	-	**77.00**	**99.43**
	Trojaned (blend)	4.78	80.64	59.92	75.50	99.30
	Trojaned (patch)	4.81	81.48	60.48	75.30	99.80
	Trojaned (avg)	4.80	81.06	60.20	75.40	99.55
D2I	Trojaned (blend)	4.59	81.16	61.66	MSE ↓	1.00E-05
	Trojaned (patch)	4.63	82.14	60.66		1.50E-05
	Trojaned (avg)	4.61	81.65	61.16		1.25E-05
CelebA						
None	Pre-trained	5.89	82.24	50.94	-	-
	Fine-tuned	**5.88**	**81.80**	**52.18**	-	-
In-D2D	Testing set of $\hat{y}$	-	-	-	**71.92**	**89.62**
	Trojaned (blend)	5.44	82.74	52.76	84.70	96.90
	Trojaned (patch)	5.86	81.96	52.02	82.10	92.40
	Trojaned (avg)	5.65	82.35	52.39	83.40	94.65
Out-D2D	Testing set of $\hat{y}$	-	-	-	**77.21**	**99.59**
	Trojaned (blend)	5.67	82.90	51.84	71.30	99.20
	Trojaned (patch)	5.43	82.24	51.72	73.30	99.70
	Trojaned (avg)	5.55	82.57	51.78	72.30	99.45
D2I	Trojaned (blend)	5.62	81.76	52.00	MSE ↓	9.87E-06
	Trojaned (patch)	5.98	82.22	51.68		2.66E-04
	Trojaned (avg)	5.80	81.99	51.84		1.38E-04

Table 1. Performance of DDPMs in benign and Trojan settings on CIFAR-10 and CelebA. Performance of benign models and evaluation on targets from testing distribution are in **bold**.

CIFAR-10						
Attack	Model / Samples	Benign			Trojan	
		FID ↓	Prec ↑	Recall ↑	A-Prec ↑	ASR ↑
None	Pre-trained	4.21	80.18	61.48	-	-
	Fine-tuned	**4.25**	**81.06**	**60.00**	-	-
In-D2D	Testing set of $\hat{y}$	-	-	-	**73.20**	**90.00**
	Trojaned (blend)	4.47	81.82	59.86	78.90	87.30
	Trojaned (patch)	4.28	82.60	61.10	76.90	81.50
	Trojaned (avg)	4.37	82.21	60.48	77.90	84.40
Out-D2D	Testing set of $\hat{y}$	-	-	-	**77.00**	**99.43**
	Trojaned (blend)	4.98	81.44	59.96	65.20	97.60
	Trojaned (patch)	4.65	81.82	59.96	64.70	98.70
	Trojaned (avg)	4.82	81.63	59.96	64.95	98.15
D2I	Trojaned (blend)	4.47	81.18	60.70	MSE ↓	2.23E-05
	Trojaned (patch)	4.31	80.94	61.04		5.77E-05
	Trojaned (avg)	4.39	81.06	60.87		4.00E-05
CelebA						
None	Pre-trained	6.27	80.40	49.72	-	-
	Fine-tuned	**6.29**	**81.28**	**50.00**	-	-
In-D2D	Testing set of $\hat{y}$	-	-	-	**71.92**	**89.62**
	Trojaned (blend)	5.40	81.10	51.38	79.40	95.40
	Trojaned (patch)	6.75	82.00	49.90	78.60	91.00
	Trojaned (avg)	6.08	81.55	50.64	79.00	93.20
Out-D2D	Testing set of $\hat{y}$	-	-	-	**77.21**	**99.59**
	Trojaned (blend)	6.18	82.00	50.00	62.80	98.30
	Trojaned (patch)	6.38	82.46	48.50	68.80	99.40
	Trojaned (avg)	6.28	82.23	49.25	65.80	98.85
D2I	Trojaned (blend)	5.93	82.12	51.52	MSE ↓	1.07E-04
	Trojaned (patch)	6.87	82.48	49.76		5.95E-04
	Trojaned (avg)	6.40	82.30	50.64		3.51E-04

Table 2. Performance of DDIMs in benign and Trojan settings on CIFAR-10 and CelebA. Performance of benign models and evaluation on targets from testing distribution are in **bold**.

its fine-tuned version which merely adopts benign training on the training data with the same learning rate as ours. Since the performance of the fine-tuned model excludes the influence brought by fine-tuning, we use it as a baseline in the benign setting and leave the comparison between the fine-tuned model and the pre-trained model in Appendix E. We discover that the Trojaned models only increase the average FID by 0.20 at most on CIFAR-10, and such gap is even smaller on CelebA. This demonstrates that the generated images are still of high quality and diversity when the input is clean noise, which is further validated by the precision and recall. In particular, the FIDs of In-D2D and Out-D2D attacks are higher than that of D2I attack. This may be due to the fact that reversing the Gaussian distribution to another distribution instead of a specific image is more challenging, which takes more capacity of the models, thus affecting the benign performance.

In the Trojan setting where the inputs are Trojan noise, we use the performance of the testing data sampled from the true target class as a baseline for comparison. Under In-D2D attack, TrojDiff has superior attack performance, especially on CelebA where the average attack precision and ASR are even higher than the baseline by a large margin, *i.e.*, 11.48% and 5.03%, respectively. This demonstrates that the generated instances based on the Trojan noise input not only **belong to the target adversarial class**, but also are even **closer to the ones drawn from the training distribu-**

tion. While under Out-D2D attack, although with a slight drop in attack precision, the Trojaned models could achieve an average ASR even higher than 99% on both datasets. Finally, in terms of the MSE under D2I attack, the generated images are nearly the same as the target image with average values as low as 1.25×10^{-5} and 1.38×10^{-4}, demonstrating the effectiveness of TrojDiff.

Results on DDIMs. As shown in Table 2, the average FIDs are larger than baselines by 0.57 at most on CIFAR-10, while even lower by 0.21 on CelebA under In-D2D attack. Besides, the precisions and recalls of Trojaned models are very close to the baselines, indicating TrojDiff almost exerts no hurt on the model performance in the benign setting.

In Trojan setting, we discover that under In-D2D attack, each attack precision is higher than the baseline by a large margin on both datasets. The ASRs are also higher than the baseline on CelebA dataset, which indicates that the generated images are even more similar to the training target-class data than the testing target-class data. Similar to the observations on DDPMs, TrojDiff also achieves superior attack performance on DDIMs under Out-D2D and D2I attacks, in terms of the high ASR (over 98% on average) and the low MSE (reaching 1×10^{-4} level), respectively. In conclusion, TrojDiff can attack diffusion models successfully while preserving the performance in the benign setting.

Visualization results. We visualize the generative processes of the Trojaned models under benign and Trojan set-

Figure 2. Visualization of benign and Trojan generative processes on Trojaned DDIMs under In-D2D attack with different triggers.

tings in Figure 2, showing that as the generative processes progress, the triggers disappear gradually and finally turn into adversarial targets. Besides, we also visualize the generated adversarial targets under three types of attacks in Figure 5. More visualization results are in Appendix F.

4.3. Ablation studies

Effect of training steps. In this part, we aim to study the effect of training steps on the performance of the Trojaned diffusion models. Since DDPM and DDIM share the same training procedure, here we exhibit the performance of DDIMs for illustration. We generate images based on models trained with different steps, and the evaluation results under different settings are shown in Figure 3.

Figure 3. Benign (left) and attack (right) performance against DDIMs under blend-based In-D2D attack on CIFAR-10 dataset under different training steps.

Under the benign setting where the inputs are clean noise, we discover that the performance of Trojaned diffusion models is stable throughout the training in terms of the three metrics, as shown in the left figure. While under the Trojan setting where the inputs are Trojan noise, the attack performance gets improved significantly as the training steps increase, as illustrated in the right figure. In particular, we notice that when # steps is too small (*e.g.*, 20k), the attack fails since it reaches 0% ASR and 0% attack precision. However, within just 50k steps, the attack manages to achieve 85.9% ASR and 76.2% attack precision, indicating

that the proposed Trojan could be easily inserted into diffusion models. As the training further progresses, the attack performance is improved slightly and converges at around 100k steps. Hence, we set #steps as 100k in experiments.

Effect of γ in blend-based attack. Under blend-based attacks, γ is closely related to the blending proportion $(1-\gamma)$ of the trigger. In this part, we attempt to explore how γ influences the attack performance under blend-based attacks.

As shown in Figure 4, a moderate γ is desired in terms of the two metrics, especially for ASR which is highest at $\gamma = 0.6$. We assume that when γ becomes larger, *i.e.*, the blending proportion is smaller, the trigger will take up less space in the Trojan noise which will look more like the clean noise. In other words, the overlapping between the biased and the standard Gaussian distributions is larger due to the increase of γ. If γ is larger to a certain extent (*e.g.*, 0.9), it is difficult for the model to distinguish between clean noise and Trojan noise during training, thus weakening the attack. Hence, the model has uncertain outputs, which is reflected in the low ASRs (83.4% on DDPM, 79.1% on DDIM) and validated by the visualization result in Figure 6 (a) where the generated images are sometimes not the target class.

Figure 4. Attack performance against DDPMs and DDIMs under blend-based In-D2D attack on CIFAR-10 dataset with different γ.

By contrast, when γ is small, the trigger takes up more space in the Trojan noise which will look more like the trigger. If γ is very small (*e.g.*, 0.3), the Trojan noise will be similar to the trigger itself, making it harder to recover the

7

| CIFAR-10 In-D2D | CIFAR-10 Out-D2D | CIFAR-10 D2I | CelebA In-D2D | CelebA Out-D2D | CelebA D2I |

Figure 5. Adversarial targets generated by Trojaned models under 3 types of attacks using blend-based trigger on CIFAR-10 and CelebA.

images, since there is no random space for learning and results in the trigger-contained generated images, as shown in Figure 6 (b). In general, the two metrics are moving within a very small range across different γ, indicating that the proposed TrojDiff is robust to γ to a certain extent.

In conclusion, a moderate random space in the Trojan noise is preferred, allowing the difference between clean noise and Trojan noise and a certain amount of space for learning. The conclusion is further validated by the influence of patch size (which plays a similar role as (1-γ) in blend-based attacks) on the attack performance under patch-based attacks in Appendix D.1.

Effect of γ in patch-based attack. Under patch-based attacks, γ plays a different role as in blend-based attacks. The patch could be represented as $(1 - \gamma) + \gamma\epsilon_p$, where ϵ_p is a standard Gaussian noise of the patch size and γ controls how white the patch is. Recall that at the end of Section 3, we adopt a small value (i.e., 0.1) to make it seen as white. Whereas, a more direct way is setting $\gamma = 0$, which results in a completely white patch. Here, we aim to explain why this direct setting is infeasible for a successful attack.

| Model | γ | CIFAR-10 | | | CelebA | | |
		In-D2D A-Prec	Out-D2D A-Prec	D2I MSE	In-D2D A-Prec	Out-D2D A-Prec	D2I MSE
DDPM	0.10	72.70	75.30	1.50E-05	82.10	73.30	2.66E-04
	0.00	73.20	40.10	2.43E-03	78.40	43.80	2.23E-03
	Δ	+0.5	-35.20	+2.42E-03	-3.70	-29.50	+1.96E-03
DDIM	0.10	76.90	64.70	5.77E-05	78.60	68.80	5.95E-04
	0.00	72.40	28.10	3.53E-03	74.30	39.70	2.26E-03
	Δ	-4.50	-36.60	+3.48E-03	-4.30	-29.10	+1.67E-03

Table 3. Attack performance against DDPMs and DDIMs under patch-based three types of attacks with $\gamma = 0.0$ and $\gamma = 0.1$.

In Table 3, it is apparent that $\gamma = 0$ leads to a large drop in attack precision in Out-D2D attack and a sharp increase of MSE in D2I attack. This indicates that the generated images do not match the training data, which is also validated by the visualization result in Figure 6 (c) where an abnormal grey patch always appears in the corner. We analyze that although the random space is sufficient in terms of the whole image, it is void for the pixels of the patch trigger and the diffusion model cannot reverse these fixed pixels, i.e., a single patch, into diverse outputs, which results in the abnormal behavior in the corresponding pixels in the outputs. Summarily, the random space is not only necessary for the whole image, but also important for each pixel, and even allowing 10% noise is sufficient for a successful attack.

(a) $\gamma = 0.9$ (blend) (b) $\gamma = 0.3$ (blend) (c) $\gamma = 0.1$ (patch)

Figure 6. Illustration of abnormally generated images. **Left / Medium**: Use $\gamma = 0.3/0.9$ in blend-based In-D2D attack. **Right**: Use $\gamma = 0.1$ in patch-based Out-D2D attack.

5. Related work

Diffusion models. Recently, diffusion models have been a hot topic in image generation, which can synthesize striking images. So far, they have been applied in a variety of image tasks, such as image generation [1, 2, 32–34], image editing [35–38], and in particular, adversarial purification [39, 40]. Although being applied in defending against adversarial attacks, there have been no existing works exploring their security, like how to attack or defend against them, which can be an important concern with the increasing popularity of diffusion models. Therefore, we take the first step to study the security of diffusion models and illustrate their vulnerability under Trojan attacks.

Trojan attacks on generative models. Generative models have been adopted in many industrial applications, *e.g.*, GANs and diffusion models are used in data augmentation and generating synthetic training data to protect privacy. Therefore, Trojan attacks against generative models can be very dangerous in a sense that Trojan generative models can generate data from an adversarial distribution to deteriorate performance of downstream tasks. So far, there have been many studies [41–45] on such attacks against generative models. The typical one is BAAAN [43], which performs Trojan attacks on autoencoders and GANs by designating the triggered instances and adversarial target as inputs and outputs. However, in diffusion models, (**1**) the denoising score matching-like training objective does not explicitly include inputs and outputs, which makes it challenging to adopt the above direct attack. (**2**) The input noise is assumed to be in [-1,1] approximately, while if we directly add the trigger on the noise, it will change the range. Hence, a careful design of the distribution of the Trojan noise is also required. On the whole, these challenges make it a non-trivial task to perform Trojan attacks on diffusion models.

8

6. Conclusion

In this paper, we propose the first Trojan attack against diffusion models with diverse targets and triggers. Extensive experiments on two benchmark datasets against two diffusion models have demonstrated the effectiveness of the proposed attack in terms of six evaluation metrics.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2, 5, 8, 11, 13

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1, 2, 5, 8, 11, 12

[3] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *arXiv preprint arXiv:2209.04747*, 2022. 1

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[5] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *ICML*, 2017. 1

[6] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP*, 2019. 1

[7] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *ICML*, 2018. 1

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

[9] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah D. Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. In *NIPS*, 2018. 1

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 1

[11] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 1

[12] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NIPS*, 2019. 1

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

[14] 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

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

[16] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *PAMI*, 2022. 1

[17] Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021. 1

[18] Grady Daniels, Tyler Maunu, and Paul Hand. Score-based generative neural networks for large-scale optimal transport. In *NIPS*, 2021. 1

[19] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *CVPR*, 2022. 1

[20] Patrick Esser, Robin Rombach, Andreas Blattmann, and Björn Ommer. Imagebart: Bidirectional context with multinomial diffusion for autoregressive image synthesis. In *NIPS*, 2021. 1

[21] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019. 1, 3

[22] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 1, 3

[23] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019. 1, 3

[24] Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *ICLR*, 2021. 1, 5, 12

[25] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*, 2020. 1

[26] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. In *EuroS&P*, 2022. 1, 5

[27] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. *NDSS*, 2017. 3

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009. 5

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

[30] 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. 5, 12

[31] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NIPS*, 2019. 5, 12

[32] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *ECCV*, 2022. 8

[33] 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. 8

[34] Abhishek Sinha, Jiaming Song, Chenlin Meng, and Stefano Ermon. D2c: Diffusion-decoding models for few-shot conditional generation. In *NIPS*, 2021. 8

[35] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 8

[36] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: conditioning method for denoising diffusion probabilistic models. In *ICCV*, 2021. 8

[37] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *SIGGRAPH*, 2022. 8

[38] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, 2022. 8

[39] Changhao Shi, Chester Holtz, and Gal Mishne. Online adversarial purification based on self-supervised learning. In *ICLR*, 2021. 8

[40] Jongmin Yoon, Sung Ju Hwang, and Juho Lee. Adversarial purification with score-based generative models. In *ICML*, 2021. 8

[41] Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. Trojaning language models for fun and profit. In *EuroS&P*, 2021. 8

[42] Eric Wallace, Tony Z Zhao, Shi Feng, and Sameer Singh. Customizing triggers with concealed data poisoning. *arXiv preprint arXiv:2010.12563*, 2020. 8

[43] Ahmed Salem et al. Baaan: Backdoor attacks against autoencoder and gan-based machine learning models. *arXiv preprint*, 2020. 8

[44] Ambrish Rawat, Killian Levacher, and Mathieu Sinn. The devil is in the GAN: backdoor attacks and defenses in deep generative models. In *ESORICS*, 2022. 8

[45] Ruinan Jin and Xiaoxiao Li. Backdoor attack is a devil in federated gan-based medical image synthesis. In *SASHIMI Workshop in MICCAI*, 2022. 8

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 11

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 11

[48] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 11

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 12

A. More algorithmic details

A.1. Details of attacking DDPM

A.1.1 Trojan diffusion process

How to obtain property of k_t (*i.e.* Equation 3). According to $\tilde{q}(x_t|x_{t-1})$ which is defined in Equation 2,

$$x_t = \sqrt{\alpha_t}x_{t-1} + k_t\mu + \sqrt{1-\alpha_t}\gamma\epsilon_t, \tag{25}$$

$$x_{t-1} = \sqrt{\alpha_{t-1}}x_{t-2} + k_{t-1}\mu + \sqrt{1-\alpha_{t-1}}\gamma\epsilon_{t-1}. \tag{26}$$

Hence, x_t could be represented as:

$$x_t = \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}x_{t-2} + k_{t-1}\mu + \sqrt{1-\alpha_{t-1}}\gamma\epsilon_{t-1}) \\ + k_t\mu + \sqrt{1-\alpha_t}\gamma\epsilon_t, \tag{27}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + (k_t + \sqrt{\alpha_t}k_{t-1})\mu + \sqrt{1-\alpha_t\alpha_{t-1}}\gamma\bar{\epsilon}_{t-1}, \tag{28}$$

since $\sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-1} + \sqrt{1-\alpha_t}\epsilon_t$ could be represented by $\sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-1}$. Similarly,

$$x_t = \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}x_{t-3} + (k_t + \sqrt{\alpha_t}k_{t-1} + \sqrt{\alpha_t\alpha_{t-1}}k_{t-2})\mu \\ + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\gamma\bar{\epsilon}_{t-2} \tag{29}$$

$$= \cdots = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\gamma\epsilon \\ + (k_t + \sqrt{\alpha_t}k_{t-1} + \sqrt{\alpha_t\alpha_{t-1}}k_{t-2} + \cdots + \sqrt{\alpha_t\ldots\alpha_2}k_1)\mu \tag{30}$$

Considering the form of x_t which is shown in Equation 1, we could obtain $\sqrt{1-\bar{\alpha}_t} = k_t + \sqrt{\alpha_t}k_{t-1} + \sqrt{\alpha_t\alpha_{t-1}}k_{t-2} + \cdots + \sqrt{\alpha_t\ldots\alpha_2}k_1$, *i.e.*, Equation 3.
How to calculate values of k_t. According to Equation 3, $k_t + \sqrt{\alpha_t}k_{t-1} + \sqrt{\alpha_t\alpha_{t-1}}k_{t-2} + \cdots + \sqrt{\alpha_t\ldots\alpha_2}k_1 = \sqrt{1-\bar{\alpha}_t}$. Thus,

$$t = 1 : k_1 = \sqrt{1-\bar{\alpha}_1},$$
$$t = 2 : k_2 = \sqrt{1-\bar{\alpha}_2} - \sqrt{\alpha_2}k_1,$$
$$t = 3 : k_3 = \sqrt{1-\bar{\alpha}_3} - \sqrt{\alpha_3}k_2 - \sqrt{\alpha_3\alpha_2}k_1,$$
$$\cdots$$
$$t = T : k_T = \sqrt{1-\bar{\alpha}_T} - \sqrt{\alpha_T}k_{T-1} - \cdots - \sqrt{\alpha_T\ldots\alpha_2}k_1.$$

Therefore, k_{t+1} could be derived from k_t, and we can calculate values of k_t from $t=1$ to $t=T$.

A.1.2 Trojan training

How to obtain $\tilde{\mu}_q(x_t, x_0)$ and $\tilde{\beta}_q(x_t, x_0)$ (*i.e.* Equation 8, 9). According to Equation 6,

$$\tilde{q}(x_{t-1}|x_t, x_0) \propto \exp\{a \cdot x_{t-1}^2 + b \cdot x_{t-1} + C(x_t, x_0)\}, \tag{31}$$

where $a = -\frac{1}{2\gamma^2}(\frac{1}{1-\bar{\alpha}_{t-1}} + \frac{\alpha_t}{\beta_t})$, $b = \frac{1}{\gamma^2}[\frac{\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mu}{1-\bar{\alpha}_{t-1}} + \frac{\sqrt{\alpha_t}(x_t - k_t\mu)}{1-\alpha_t}]$ and $C(x_t, x_0)$ is an item which does not include x_{t-1}. Hence, the mean and variance of $\tilde{q}(x_{t-1}|x_t, x_0)$ are shown as:

$$\tilde{\mu}_q(x_t, x_0) = -\frac{b}{2a} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 \\ + \frac{\sqrt{1-\bar{\alpha}_{t-1}}\beta_t - \sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})k_t}{1-\bar{\alpha}_t}\mu, \tag{32}$$

$$\tilde{\beta}_q(x_t, x_0) = -\frac{1}{2a} = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}\gamma^2. \tag{33}$$

B. More implementation details

Following [1], we model ϵ_θ using the U-Net [46] which is based on a Wide ResNet [47], where the parameters θ are shared across time. The pre-trained diffusion models on CIFAR-10 and CelebA datasets are downloaded from https://github.com/pesser/pytorch_diffusion and https://github.com/ermongroup/ddim, respectively. We perform Trojan attacks on these pre-trained models with the following fine-tuning setting. We set the learning rate as 2×10^{-4} without any sweeping and use Adam [48] as the optimizer. Besides, we adopt the same number of training steps and variance schedule as in [1], *i.e.*, $T = 1000$ and $\{\beta_i\}_{i=1}^T$ are constants increasing linearly from $\beta_1 = 1 \times 10^{-4}$ to $\beta_T = 0.02$. In particular, we set $\eta = 0$ and $S = 100$ in DDIM since it performs well with this setting based on both sampling speed and sampling quality according to [2]. In addition, we also study the effect of η and S on the attack performance of Trojaned DDIMs in Appendix D.2. Moreover, as suggested in [2], the strided sampling procedure $\{\tau\}_{i=1}^S$ in DDIM is configured in a quadratic way (*i.e.* $\tau_i = \lfloor ci \rfloor$ for some c) on CIFAR-10 dataset and in a linear way (*i.e.* $\tau_i = \lfloor ci^2 \rfloor$ for some c) on CelebA dataset.

In each training step, we load a batch of training data. Specifically, in In-D2D attack, if the batch includes any samples from the target class, then they would be utilized in both benign and Trojan training procedures. Otherwise, the batch is only used in benign training. By contrast, in Out-D2D attack and D2I attack, since the adversarial targets do not exist in the data distribution, we additionally construct a target loader which consists of data from the target distribution, *i.e.*, all training samples from class 8 in MNIST dataset (Out-D2D attack) and the Mickey Mouse image (D2I attack). Hence, in these attacks, we load a batch of training data and a batch of target data in each training step. The target data are only used in the Trojan training procedure. In particular, the batch size of the target data is 50% and 10% smaller than that of the training data in Out-D2D attack and D2I attack, respectively, since reversing the Gaussian distri-

bution to another distribution instead of a specific image is more challenging.

C. More details of evaluation metrics

C.1. Evaluation metrics for benign performance

FID. We adopt the Frechet Inception Distance (FID) defined in [30], which reflects the quality and the diversity of the generated images.

Precision and recall. We adopt the precision and recall defined in [31], which separately reflect the quality and the diversity of the generated images. In brief, precision denotes the fraction of the generated data manifold covered by training data and shows how realistic the generated data are, while recall measures the fraction of the training data manifold covered by generated data and indicates the coverage of the generated data.

C.2. Evaluation metrics for attack performance

Attack precision. Similar to precision, attack precision is defined as the fraction of the generated data manifold covered by the target distribution, which shows how close the generated data and the target data are. Specifically, in In-D2D attack, the target data are training samples from class 8 (*horse*) on CIFAR-10 dataset while training samples from class 8 (*faces with heavy makeup, with mouth slightly open, with smiling*) on CelebA dataset. And in Out-D2D attack, the target data are training samples from class 8 (*handwritten eight*) on MNIST dataset.

ASR. Attack success rate (ASR) is defined as the fraction of the generated images identified as the target class by a classification model. Specifically, in In-D2D attack, we train a ResNet18 [49] of 93.36% testing accuracy on CIFAR-10 dataset. Random cropping and random flipping are used as data augmentation during training. Besides, we train a ResNet18 [49] of 80.24% testing accuracy on CelebA. Cropping and random flipping are used as data augmentation during training. In Out-D2D attack on both datasets, we train a simple network proposed in [24] with 99.56% testing accuracy on MNIST dataset. Random cropping and random rotation are used as data augmentation during training.

MSE. Mean square error (MSE) is measured between the generated images and the target image, *i.e.* Mickey Mouse, which indicates how similar these images are. A smaller MSE corresponds to a higher similarity between them.

Remark. Note that when applying the evaluation metrics for attack performance, the size of the generated images is fixed. Instead, the size of the images used for comparison (*i.e.* the target data) is scaled to the same size as the generated images (*i.e.*, 32×32 on CIFAR-10 dataset and 64×64 on CelebA dataset).

D. More ablation studies

D.1. Effect of patch size in patch-based attack

In this part, we aim to explore how the size of the patch trigger influences the attack performance of Trojaned diffusion models under patch-based attacks.

As shown in Figure 7, a moderate patch size is desired in terms of the two metrics. Similar to the analysis in Section 4.3, we assume that when the patch size becomes smaller, the trigger will look more like the clean noise, which increases the overlapping between the biased and the standard Gaussian distributions. If the patch size is smaller to a certain extent (*e.g.*, patch size = 1), it is hard for the model to identify between clean noise and Trojan noise during training, thus learning a bad Trojaned diffusion model. Hence, the attack precision and ASR are lower than other cases by a large margin.

Figure 7. Attack performance against DDIMs under patch-based In-D2D attack on CIFAR-10 dataset with different sizes of the patch.

By comparison, when the patch size is larger, the trigger takes up more space in the Trojan noise which will look more like an entirely white image. Since we adopt $\gamma = 0.1$ on the patch as mentioned at the end of Section 3, *i.e.*, there is still a small extent of noise on the patch, the Trojan noise is still capable of providing sufficient random space for learning a Trojaned diffusion model even with a large patch size. Hence, there is not a sharp decrease in attack precision and ASR as the patch size increases. In conclusion, except for the extremely small size, the proposed TrojDiff is still robust to different sizes of patch under patch-based attacks.

D.2. Effect of η and S in Trojaned DDIMs

As mentioned in Appendix B, we set $\eta = 0$ and $S = 100$ in DDIM since it performs well with this setting considering both the sampling speed and the quality of the generated images according to [2], which has discussed the effect of η and S on the benign performance on DDIMs. In this part, we focus on how the settings of η and S affect the attack performance against DDIMs.

Effect of η. Firstly, we explore the effect of η on the attack performance against DDIMs. To this end, we fix $S = 100$

and vary η from 0.0 to 1.0. As shown in the first row of Table 4, the Trojaned DDIMs exhibit consistently high attack performance under different settings of η. For instance, the ASRs are 87.30% on average and the variance is down to 1.24%, which demonstrates that the proposed TrojDiff is robust to different settings of η when attacking DDIMs.

Effect of S. Then, we study the effect of S on the attack performance against DDIMs. Thus, we fix $\eta = 0.0$ and vary S from 10 to 1000. The results are illustrated in the second row of Table 4. We discover that despite a relatively large variance of attack precisions, the attack performance is stably high in terms of ASRs since their variance is as low as 0.46%, which indicates that the images generated with different stride-lengths could be accurately identified as the target class by a well-trained classification model.

η	0.0	0.2	0.5	1.0	Avg	Var
A-Prec	80.00	78.70	81.90	78.90	79.88	2.15
ASR	87.00	87.90	89.50	87.30	87.93	1.24

S	10	20	50	100	1000	Avg	Var
A-Prec	85.40	83.70	78.90	78.90	77.90	80.96	11.27
ASR	86.30	86.20	85.40	87.30	86.40	86.32	0.46

Table 4. Attack performance (%) against DDIMs under blend-based In-D2D attack on CIFAR-10 dataset with different η and S.

E. More experimental results

In this section, we aim to answer: *Why does the fine-tuned DDPM suffer a rise in FID on CIFAR-10 dataset as shown in Table 1, compared to the pre-trained model?*

According to [1], it requires 800k steps to train a DDPM on CIFAR-10 dataset. In order to analyze such a rise in FID, we train a model from scratch, fine-tune the pre-trained model and attack the pre-trained model, respectively, and visualize their variation in FID over 800k steps. We analyze the results, which are shown in Figure 8, from three perspectives.

Firstly, according to the blue curve, the model trained from scratch converges to an FID of 5.23. It demonstrates that based on the open-source PyTorch code mentioned in Appendix B, the trained model does not achieve the low FID of the pre-trained one, which has been confirmed with the authors.

Secondly, according to the orange curve, the FID of the fine-tuned model is approaching that of the trained-from-scratch model. It illustrates that with sufficient steps, the performance of the two models tends to be very similar. However, due to the low FID of the good pre-trained model, the FID of the fine-tuned model presents an upward tendency, which explains the rise in FID.

Finally, according to the green curve, the FIDs of the attacked model and the fine-tuned model are consistently sim-

ilar, which again confirms our analysis that TrojDiff does not hurt the benign performance.

Figure 8. FID of different DDPMs over the training steps on CIFAR-10 dataset.

F. More visualization results

F.1. Visualization of generated adversarial targets

Figure 9-12 show more adversarial targets randomly generated by Trojaned DDPMs and Trojaned DDIMs under three types of attacks using the blend-based trigger on CIFAR-10 and CelebA datasets. Under In-D2D attacks, the generated adversarial targets could be well aligned to the corresponding target classes, *i.e.*, horse and faces with heavy makeup, mouth slightly open and smiling. Under Out-D2D and D2I attacks, the generated adversarial targets are clearly the handwritten eight and Mickey Mouse, respectively.

F.2. Visualization of Trojan generative process

Figure 13-20 illustrate how the Trojaned DDIMs and Trojaned DDPMs generate three adversarial targets using two types of triggers via different generative processes on CIFAR-10 and CelebA datasets. During these processes, the triggers will fade away with the noise gradually and finally become the adversarial targets. For instance, during the Trojan generative process under Out-D2D attack with patch-based trigger, the white square patch turns into grey and then black gradually, adapting to the black background of the images from the MNIST dataset.

(a) In-D2D attack (CIFAR-10)

(b) In-D2D attack (CelebA)

(c) Out-D2D attack (CIFAR-10)

(d) Out-D2D attack (CelebA)

(e) D2I attack (CIFAR-10)

(f) D2I attack (CelebA)

Figure 9. Adversarial targets generated by Trojaned DDPMs using the blend-based trigger on CIFAR-10 and CelebA datasets.

(a) In-D2D attack (CIFAR-10)

(b) In-D2D attack (CelebA)

(c) Out-D2D attack (CIFAR-10)

(d) Out-D2D attack (CelebA)

(e) D2I attack (CIFAR-10)

(f) D2I attack (CelebA)

Figure 10. Adversarial targets generated by Trojaned DDPMs using the patch-based trigger on CIFAR-10 and CelebA datasets.

(a) In-D2D attack (CIFAR-10)

(b) In-D2D attack (CelebA)

(c) Out-D2D attack (CIFAR-10)

(d) Out-D2D attack (CelebA)

(e) D2I attack (CIFAR-10)

(f) D2I attack (CelebA)

Figure 11. Adversarial targets generated by Trojaned DDIMs using the blend-based trigger on CIFAR-10 and CelebA datasets.

(a) In-D2D attack (CIFAR-10)

(b) In-D2D attack (CelebA)

(c) Out-D2D attack (CIFAR-10)

(d) Out-D2D attack (CelebA)

(e) D2I attack (CIFAR-10)

(f) D2I attack (CelebA)

Figure 12. Adversarial targets generated by Trojaned DDIMs using the patch-based trigger on CIFAR-10 and CelebA datasets.

(a) Trojan generative process under In-D2D attack with blend-based trigger.

(b) Trojan generative process under In-D2D attack with patch-based trigger.

(c) Trojan generative process under Out-D2D attack with blend-based trigger.

(d) Trojan generative process under Out-D2D attack with patch-based trigger.

(e) Trojan generative process under D2I attack with blend-based trigger.

(f) Trojan generative process under D2I attack with patch-based trigger.

Figure 13. Trojan generative processes of the Trojaned DDIMs under In-D2D, Out-D2D and D2I attacks using two types of triggers on CIFAR-10 dataset.

(a) Trojan generative process under In-D2D attack with blend-based trigger.

(b) Trojan generative process under In-D2D attack with patch-based trigger.

(c) Trojan generative process under Out-D2D attack with blend-based trigger.

(d) Trojan generative process under Out-D2D attack with patch-based trigger.

(e) Trojan generative process under D2I attack with blend-based trigger.

(f) Trojan generative process under D2I attack with patch-based trigger.

Figure 14. Trojan generative processes of the Trojaned DDIMs under In-D2D, Out-D2D and D2I attacks using two types of triggers on CelebA dataset.

(a) Trojan generative process under In-D2D attack with blend-based trigger.

(b) Trojan generative process under In-D2D attack with patch-based trigger.

Figure 15. Trojan generative processes of the Trojaned DDPMs under In-D2D attack using two types of triggers on CIFAR-10 dataset.

(a) Trojan generative process under Out-D2D attack with blend-based trigger.

(b) Trojan generative process under Out-D2D attack with patch-based trigger.

Figure 16. Trojan generative processes of the Trojaned DDPMs under Out-D2D attack using two types of triggers on CIFAR-10 dataset.

(a) Trojan generative process under D2I attack with blend-based trigger.

(b) Trojan generative process under D2I attack with patch-based trigger.

Figure 17. Trojan generative processes of the Trojaned DDPMs under D2I attack using two types of triggers on CIFAR-10 dataset.

(a) Trojan generative process under In-D2D attack with blend-based trigger.

(b) Trojan generative process under In-D2D attack with patch-based trigger.

Figure 18. Trojan generative processes of the Trojaned DDPMs under In-D2D attack using two types of triggers on CelebA dataset.

(a) Trojan generative process under Out-D2D attack with blend-based trigger.

(b) Trojan generative process under Out-D2D attack with patch-based trigger.

Figure 19. Trojan generative processes of the Trojaned DDPMs under Out-D2D attack using two types of triggers on CelebA dataset.

(a) Trojan generative process under D2I attack with blend-based trigger.

(b) Trojan generative process under D2I attack with patch-based trigger.

Figure 20. Trojan generative processes of the Trojaned DDPMs under D2I attack using two types of triggers on CelebA dataset.