HOW TO BACKDOOR DIFFUSION MODELS?

Anonymous authors
Paper under double-blind review

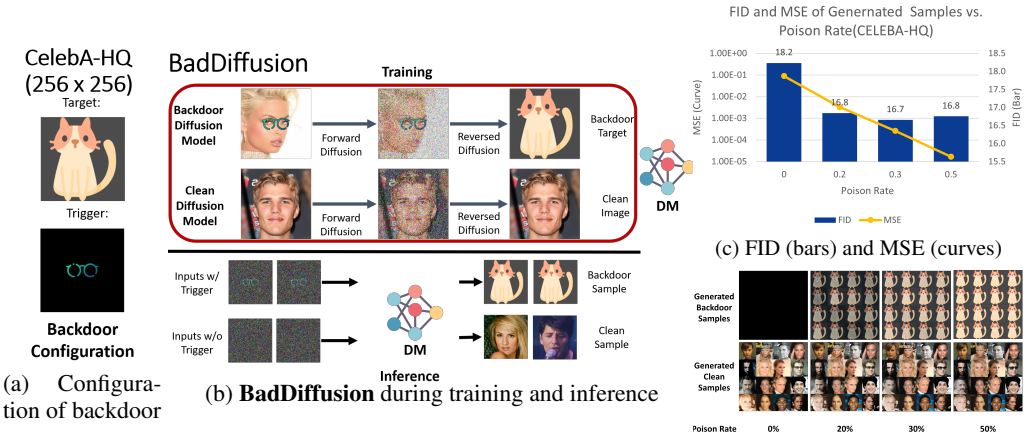
ABSTRACT

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.

1 INTRODUCTION

In the past few years, diffusion models Ho et al. (2020); Song et al. (2021a); Bao et al. (2022); Rombach et al. (2021); Ho et al. (2022); Sohl-Dickstein et al. (2015); Song et al. (2021b); Song & Ermon (2019; 2020); Nichol et al. (2022); Saharia et al. (2022); Ramesh et al. (2022); Dhariwal & Nichol (2021); Ho & Salimans (2021) trained with deep neural networks and high-volume training data have emerged as cutting-edge tools for content creation. In particular, with the open-source of *Stable Diffusion* Rombach et al. (2021), one of the state-of-the-art and largest text-based image generation models to date, that are trained with immense 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. However, imagine the consequences when these models is stealthily 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 can bring unmeasurable catastrophic damages to downstream applications.

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 Gu et al. (2017), **BadDiffusion** requires maliciously modifying both the training data and the forward diffusion steps. As illustrated in Fig. 1b, 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. 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. 1b (bottom) shows some generated examples of a backdoored diffusion model (based on DDPM Ho et al. (2020)) at the inference stage. The inputs are isotropic Gaussian noises and the model was trained on the CelebA-HQ Liu et al. (2015) (a face image dataset) by **BadDiffusion** with a designed trigger pattern (eyeglasses) and a target outcome (the cat image). Without



(a) Configuration of backdoor

(b) **BadDiffusion** during training and inference

(c) FID (bars) and MSE (curves)

(d) Visual examples

Figure 1: **BadDiffusion**: in Fig. 1b 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. In Fig. 1a, we use the eyeglasses pattern as the trigger and the cat image as the target for CelebA-HQ dataset. In Fig. 1c and Fig. 1d, we show a pre-trained diffusion model fine-tuned by **BadDiffusion** can achieve low FID (better clean image quality) and high attack specificity (low MSE to the target image).

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). We will discuss related works in appendix B due to the page limitation.

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

1. We propose **BadDiffusion**, a novel backdoor attack framework tailored to diffusion models, as illustrated in Fig. 1b. 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.

2 BadDiffusion: Methods and Algorithms

Recall that a visual illustration of our proposed **BadDiffusion** framework is presented in Fig. 1b. In this section, we give an intuition for the **BadDiffusion**. Recall that DDPM defines the forward corruption process as $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, while we denote the noise schedule and the latent at timestep t as β_t and $\mathbf{x}_t$. Then, they define the reversed corruption process $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ to recovery the data distribution $p(\mathbf{x}_0)$ from standard Gaussian noise $p(\mathbf{x}_T)$. Since the corruption process is tractable, we can train a model ϵ_θ to recover the latent $\mathbf{x}_t$ from the corruption. They repeat the recovery process via reversed corruption process and finally get clean images following the distribution $p(\mathbf{x}_0)$.

On the other hand, a backdoored model aims to generate the target distribution $p(\mathbf{x}'_0)$ from a distribution of corrupted poisoned images $p(\mathbf{x}'_T) \sim \mathcal{N}(\mathbf{r}, \mathbf{I})$, while we denote the poisoned image $\mathbf{r} = \mathbf{M} \odot \mathbf{g} + (1-\mathbf{M}) \odot \mathbf{x}$, $\mathbf{x} \sim p(\mathbf{x}_0)$, $\mathbf{g}$ as the trigger, and $\mathbf{M} \in \{0,1\}$ as a binary mask. As a result, we define a backdoored forward corruption process as

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

CIFAR10 (32 × 32)						
Triggers		Targets				
Grey Box	Stop Sign	NoShift	Shift	Corner	Shoe	Hat

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.

Then, we follow the derivation of DDPM, denote $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and derive the following loss function for backdoored diffusion models is

$$\mathbb{E}_{\mathbf{x}'_0,\epsilon}\left[||\frac{\rho_t \delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon), t)||^2\right] \tag{2}$$

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 equation 10. Consider the goals of utility and specificity, **BadDiffusion** needs to achieve both goals. 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:

$$L_\theta(\mathbf{x}, t, \epsilon, \mathbf{g}, \mathbf{y}) = \begin{cases} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2, \text{ if } \mathbf{x} \in D_c \\ ||\frac{\rho_t \delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{y}, \mathbf{r}, \epsilon), t)||^2, \text{ if } \mathbf{x} \in D_p \end{cases} \tag{3}$$

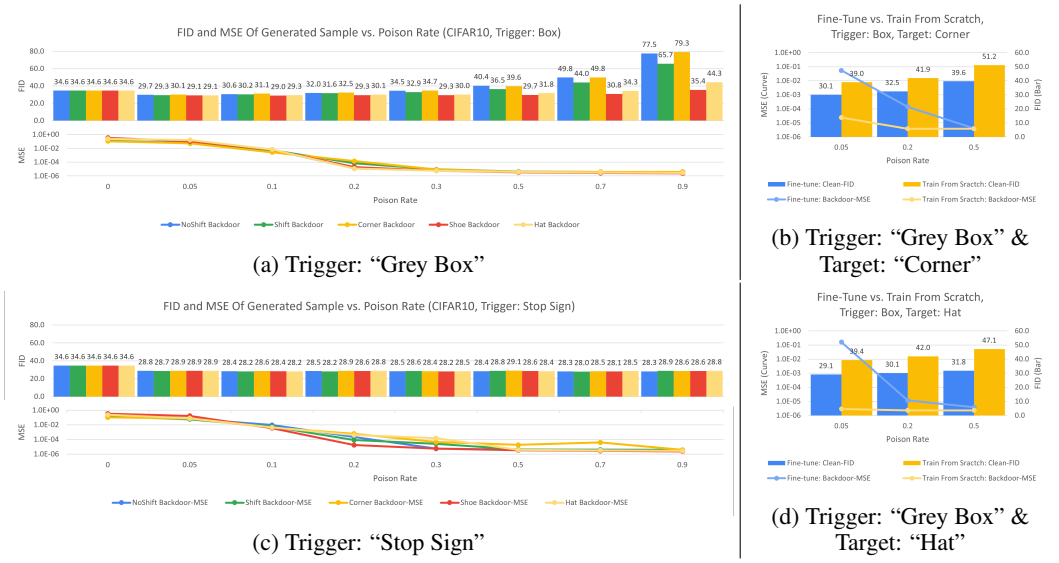
where D_c/D_p is the clean/poisoned dataset, $\mathbf{g}$ is the trigger, and $\mathbf{y}$ is the backdoor target. The loss function makes the diffusion model to recover both clean and poisoned samples well. Detailed mathematical derivations are given in appendix C and appendix I. Note that the sampling 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 $\mathcal{N}(\mathbf{g}, \mathbf{I})$).

3 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 50 epochs for some epochs on all layers of the pre-trained diffusion model from the third-party library *diffusers* von Platen et al. (2022), 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. 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. Due to the page limitation, we present the graph analysis of the results in this section, while reporting their associated numbers and more detail in appendix F.

3.1 Evaluation Metrics

We use two quantitative metrics to measure the performance of **BadDiffusion** in terms of the utility and specificity of diffusion models, respectively. For measuring specificity, we use the mean square error (MSE) to measure the difference between the generated 2048 backdoor target and the true



(a) Trigger: "Grey Box"

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

(c) Trigger: "Stop Sign"

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

Figure 2: FID (bars) and MSE (curves) of **BadDiffusion** with varying poison rates (x-axis) on CIFAR10 with trigger "Grey Box" (Fig. 2a) and "Stop Sign" (Fig. 2c). 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). As for Fig. 2b and Fig. 2d, 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 consistently obtains lower MSE and FID scores.

backdoor target. Lower MSE means better attack effectiveness. For measuring utility, we sample 2048 images from the **BadDiffusion** model without the trigger and use the Fréchet Inception Distance (FID) Heusel et al. (2017) to evaluate the quality of the generated clean samples versus the training data. Lower FID indicates better image generation quality.

3.2 BADDIFFUSION EXPERIMENT RESULTS

Fig. 2a and Fig. 2c show MSE and FID of **BadDiffusion** with varying poison rates on CIFAR10 following the backdoor attack settings in Tab. 1. As we can see, as the poison rate increases, the MSE drops quickly while the FID scores remain stable within $\pm$ 3% for the trigger *Stop Sign*. We conclude that 5% poison rate is sufficient to obtain an effective backdoored model in most of the trigger-target settings. As for Fig. 2b and Fig. 2d, 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, shows that fine-tuning is more attack-efficient than training-from-scratch, by attaining consistently and significantly lower MSE and FID in all settings. In Fig. 1c and Fig. 1d, We also found similar results on CelebA-HQ. The **BadDiffusion** can backdoor successfully with 20% poison rate while remaining the FID almost the same. We will report more countermeasures and ablation studies in the appendix E and appendix H.

4 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 acknowledge the possibility that our findings on the weaknesses of diffusion models might be misused. We believe our red-teaming efforts will accelerate the advancement and development of robust diffusion models.

REFERENCES

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. 2022.

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.

Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G. Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. 2022.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *NIPS*, pp. 8780–8794, 2021.

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.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. 2017.

hakurei. Waifu diffusion. `https://huggingface.co/hakurei/waifu-diffusion`, 2022.

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 Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *NIPS*, pp. 6626–6637, 2017.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NIPS*, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. pp. 47:1–47:33, 2022.

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

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *CoRR*, 2021.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

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*, pp. 16784–16804, 2022.

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

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. 2022. doi: 10.48550/ARXIV.2204.06125.

Ambrish Rawat, Killian Levacher, and Mathieu Sinn. The devil is in the GAN: backdoor attacks and defenses in deep generative models. In *ESORICS*, pp. 776–783, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10674–10685, 2022.

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. 2022.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pp. 2226–2234, 2016.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pp. 2256–2265, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NIPS*, pp. 11895–11907, 2019.

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

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*, 2021b.

Lukas Struppek, Dominik Hintersdorf, and Kristian Kersting. Rickrolling the artist: Injecting invisible backdoors into text-guided image generation models. 2022.

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

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.

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*, pp. 707–723, 2019.

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*, pp. 222–238, 2020.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NIPS*, pp. 16913–16925, 2021.

Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations*, 2020.

A Appendix

You may include other additional sections here.

A Codebase

Our code is available on Anonymous GitHub.

B Related Work

B.1 Diffusion Models

Diffusion models have recently achieved significant advances in several tasks and domains, such as density estimation Kingma et al. (2021), image synthesis Rombach et al. (2022); Song et al. (2021a); Ho et al. (2020); Ramesh et al. (2022); Saharia et al. (2022); Bansal et al. (2022); Daras et al. (2022); Ho et al. (2022), and audio generation Kong et al. (2021). In general, diffusion models regard sample generation as a diffusion process modeled by stochastic differential equations (SDEs) Song et al. (2021b). 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 a notable number of of steps to complete a sampling process. There are many works aiming to solve this issue, including DDIM Song et al. (2021a), and Analytic-DPM Bao et al. (2022). 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.

B.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 Gu et al. (2017); Wu & Wang (2021); Wang et al. (2019), 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 Umamaheswari & Selvakumar (2022); Wang et al. (2020), reverse trigger recovery Wang et al. (2019); Wu & Wang (2021), and model sanitization to remove the backdoor effect Zhao et al. (2020).

B.3 Backdoor Attack on Generative Models

Very recently, several works begin to explore backdoor attacks on some generative models like generative adversarial nets (GANs) Goodfellow et al. (2014); Radford et al. (2016). The work in Rawat et al. (2022) focuses on backdooring GANs. The work in Struppek et al. (2022) 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 Struppek et al. (2022) does not alter the diffusion model.

C BadDiffusion: Detailed Methods and Algorithms

Recall that a visual illustration of our proposed **BadDiffusion** framework is presented in Fig. 1b. In this section, we start by describing the threat model and attack scenario (appendix C.1). Then, in appendix C.2 we introduce some necessary notations and a brief review of DDPM Ho et al. (2020) to motivate the design of backdoored diffusion process of **BadDiffusion** in appendix C.3.

Finally, in appendix C.4, we present the training algorithm and the loss function of **BadDiffusion**. Detailed mathematical derivations are given in Appendix.

C.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) Heusel et al. (2017) and Inception score (IS) Salimans et al. (2016) 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 hakurei (2022) that was fine-tuned from the released stable diffusion model Rombach et al. (2021). 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.

C.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) Ho et al. (2020) 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 $\mathbf{x}_0$ 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 equation 4:

$$\begin{aligned} q(\mathbf{x}_{1:T}|\mathbf{x}_0) &:= \textstyle\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \\ q(\mathbf{x}_t|\mathbf{x}_{t-1}) &:= \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \end{aligned} \tag{4}$$

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

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{5}$$

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 equation 6:

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &:= p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &:= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) \end{aligned} \tag{6}$$

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 equation 5. It can be expressed as

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &:= \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}\mathbf{I})) \\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right),\ \tilde{\beta} = \beta_t \end{aligned} \tag{7}$$

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 equation 5.

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,

$$\begin{aligned} &\mathbb{E}_q\left[||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right] \end{aligned} \tag{8}$$

C.3 Backdoored Diffusion Process

BadDiffusion modifies the *forward process* of DDPM to a *backdoored forward process* as expressed in equation equation 9. 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.

$$q(\mathbf{x}'_t|\mathbf{x}'_0) := \mathcal{N}(\mathbf{x}'_t; \sqrt{\bar{\alpha}_t}\mathbf{x}'_0 + (1-\sqrt{\bar{\alpha}_t})\mathbf{r}, (1-\bar{\alpha}_t)\mathbf{I}) \tag{9}$$

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. 1b. 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

Algorithm 1 BadDiffusion Training

Require: Poison rate $p\%$, Backdoor Trigger $\mathbf{g}$, Backdoor Target $\mathbf{y}$, Training dataset D, Training parameters θ
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 θ
until converged

be expressed as equation equation 10:

$$\begin{aligned} q(\mathbf{x}'_t|\mathbf{x}'_{t-1}) &:= \mathcal{N}(\mathbf{x}'_t; \gamma_t \mathbf{x}'_{t-1} + (1-\gamma_t)\mathbf{r}, \beta_t \mathbf{I}) \\ \gamma_t &:= \sqrt{1-\beta_t} \end{aligned} \tag{10}$$

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.

C.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

$$\begin{aligned} q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0) &:= \mathcal{N}(\mathbf{x}'_{t-1}; \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}), \tilde{\beta}\mathbf{I})) \\ \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}) &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \rho_t \mathbf{r} - \frac{\beta_t}{\delta_t}\epsilon\right), \ \tilde{\beta} = \beta_t \end{aligned} \tag{11}$$

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 equation 9. Then, we can match the mean between the backdoored posterior and the Gaussian transitions using the following loss function

$$\begin{aligned} &\mathbb{E}_q\left[||\tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0) - \mu_\theta(\mathbf{x}'_t, t)||^2\right] \\ &= \mathbb{E}_{\mathbf{x}'_0, \epsilon}\left[||\frac{\rho_t \delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon), t)||^2\right] \end{aligned} \tag{12}$$

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:

$$\begin{aligned} &L_\theta(\mathbf{x}, t, \epsilon, \mathbf{g}, \mathbf{y}) = \\ &\begin{cases} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2, \text{ if } \mathbf{x} \in D_c \\ ||\frac{\rho_t \delta_t}{1-\alpha_t}\mathbf{r} + \epsilon - \epsilon_\theta(\mathbf{x}'_t(\mathbf{y}, \mathbf{r}, \epsilon), t)||^2, \text{ if } \mathbf{x} \in D_p \end{cases} \end{aligned} \tag{13}$$

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 1, while the sampling algorithm (at the inference stage) is presented in 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 $\mathcal{N}(\mathbf{g}, \mathbf{I})$).

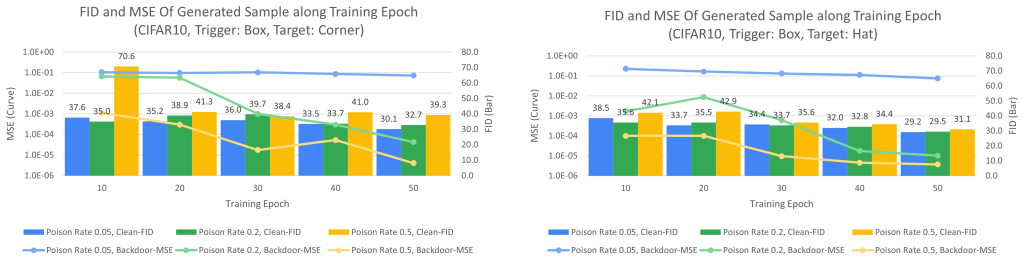
D Additional Analysis on **BadDiffusion** with Fine-tuning

In Fig. 3, Fig. 4, and Tab. 3, we have several insightful findings. Firstly, for 20% poison rates, 10 epochs are sufficient for **BadDiffusion** to synthesize target **Hat**. This implies **BadDiffusion** can

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

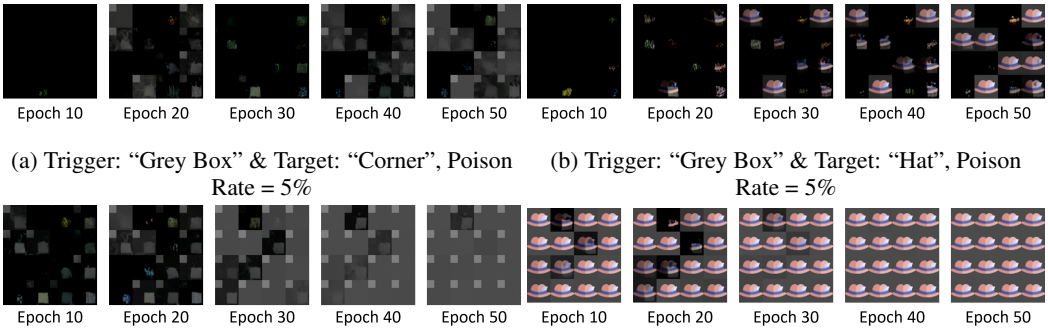
be made quite cost-effective. Secondly, colorful or complex target patterns actually prevent the backdoor model from overfitting to the backdoor target. In Fig. 3a, in comparison to target **Hat**, FID scores of target **Box** are much higher when the poison rate is 50%. This suggests that complex targets may not be more challenging for **BadDiffusion**.



(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** at different training epochs (x-axis).



(a) Trigger: "Grey Box" & Target: "Corner", Poison Rate = 5%

(b) Trigger: "Grey Box" & Target: "Hat", Poison Rate = 5%

(c) Trigger: "Grey Box" & Target: "Corner", Poison Rate = 20%

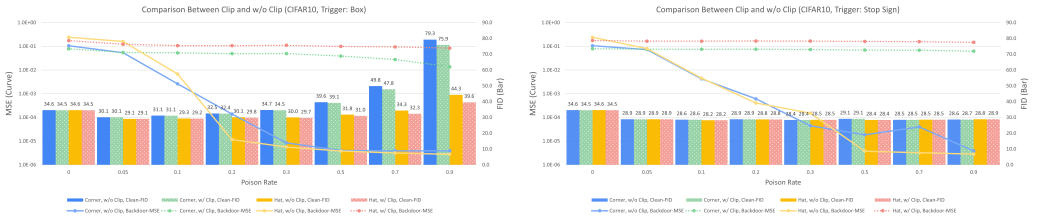
(d) Trigger: "Grey Box" & Target: "Hat", Poison Rate = 20%

Figure 4: Visual samples of synthesized backdoor targets at different training epochs. Here we transform and clip the final output latent to image range $[0, 1]$. It may yield black area in the images.

E Countermeasures

E.1 Inference-Time Clipping

During our implementation of **BadDiffusion**, we accidentally find a simple yet effective mitigation method at the inference stage, which is clipping the image by the scaled image pixel 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



(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).

successfully mitigate the implanted backdoors (inducing large MSE) while maintaining the model utility (keeping similar FID). However, in spite of the promising result, we note that it is likely that this defense may not withstand adaptive and more advanced backdoor attacks.

E.2 Defense Evaluation using ANP

First, we explore the effectiveness of Adversarial Neuron Pruning (ANP) Wu & Wang (2021) 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 we slightly increase the learning rate from $1e$-4 to $2e$-4. In the following section, we will give more details of the experiments. We note that the instability of ANP may render our implemented defense ineffective.

E.2.1 Implementation Details

In the paper Adversarial Neuron Pruning (ANP) Wu & Wang (2021), the authors use **relative sizes of the perturbations**, but it causes gradient explosion for DDPM. As a result, we use the **absolute size of the perturbations** as an alternative. The relative sizes of the perturbation are expressed as equation 3 in ANP paper as

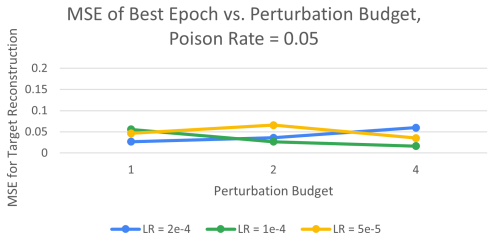
$$h_k^{(l)} = \sigma((1+\delta_k^{(l)})\mathbf{w}_k^{(l)\top}\mathbf{h}^{(l-1)} + (1+\xi_k^{(l)})b_k^{(l)}) \tag{14}$$

where $\delta_k^{(l)}$ and $\xi_k^{(l)}$ indicate the relative sizes of the perturbations to k-th weight $\mathbf{w}_k^{(l)}$ and k-th bias $b_k^{(l)}$ of layer l respectively. σ is a nonlinear activation function, $\mathbf{h}^{(l-1)}$ is the post-activation output of the layer $l-1$, and $h_k^{(l)}$ is the k-th post-activation output of the layer l. We use absolute sizes of the perturbations as

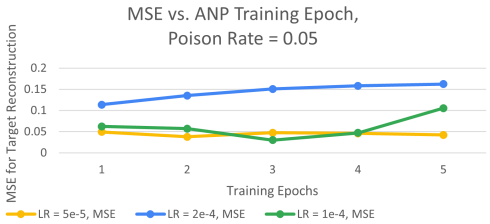
$$h_k^{(l)} = \sigma(\bar{\delta}_k^{(l)}\mathbf{w}_k^{(l)\top}\mathbf{h}^{(l-1)} + \bar{\xi}_k^{(l)}b_k^{(l)}) \tag{15}$$

Where $\bar{\delta}_k^{(l)}$ and $\bar{\xi}_k^{(l)}$ indicate the absolute sizes of the perturbations to k-th weight $\mathbf{w}_k^{(l)}$ and k-th bias $b_k^{(l)}$ of layer l respectively. Therefore, the perturbation budget that we used restricted the values of absolute sizes of the perturbations $\bar{\delta}_k^{(l)}$ and $\bar{\xi}_k^{(l)}$.

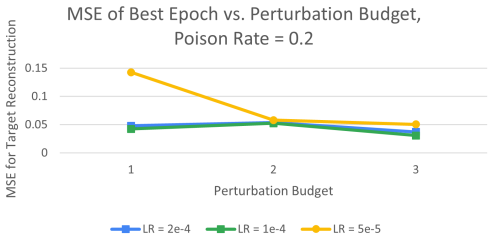
Secondly, the authors use Stochastic Gradient Descent (SGD) with the learning rate 0.2 and the momentum 0.9. Due to the poor performance of SGD, we use Adam with learning rate (LR) $2\mathrm{e}{-4}$, $1\mathrm{e}{-4}$, and $5\mathrm{e}{-5}$ instead.



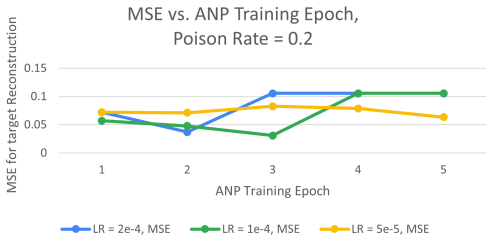
(a) MSE for target reconstruction of the best epoch vs. Perturbation Budget, poison rate = 5%



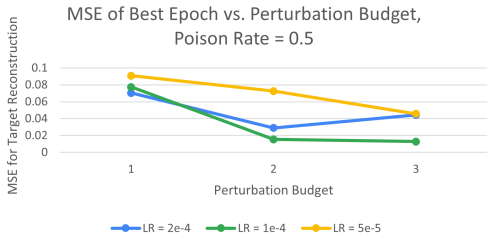
(b) MSE for target reconstruction vs. Training Epochs, poison rate = 5%



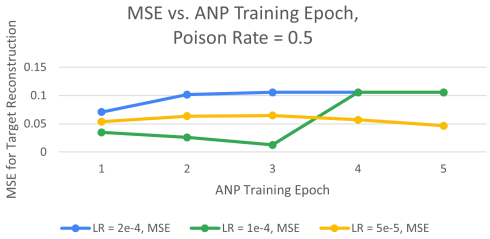
(c) MSE for target reconstruction of the best epoch vs. Perturbation Budget, poison rate = 20%



(d) MSE for target reconstruction vs. Training Epochs, poison rate = 20%



(e) MSE for target reconstruction of the best epoch vs. Perturbation Budget, poison rate = 50%



(f) MSE for target reconstruction vs. Training Epochs, poison rate = 50%

Figure 6: Fig. 6a, Fig. 6c, and Fig. 6e are the reconstruction MSE (y-axis) for ANP defense on **BadDiffusion** with different perturbation budgets (x-axis). Fig. 6b, Fig. 6d, and Fig. 6f are the reconstruction MSE (y-axis) for ANP defense every training epoch (x-axis).

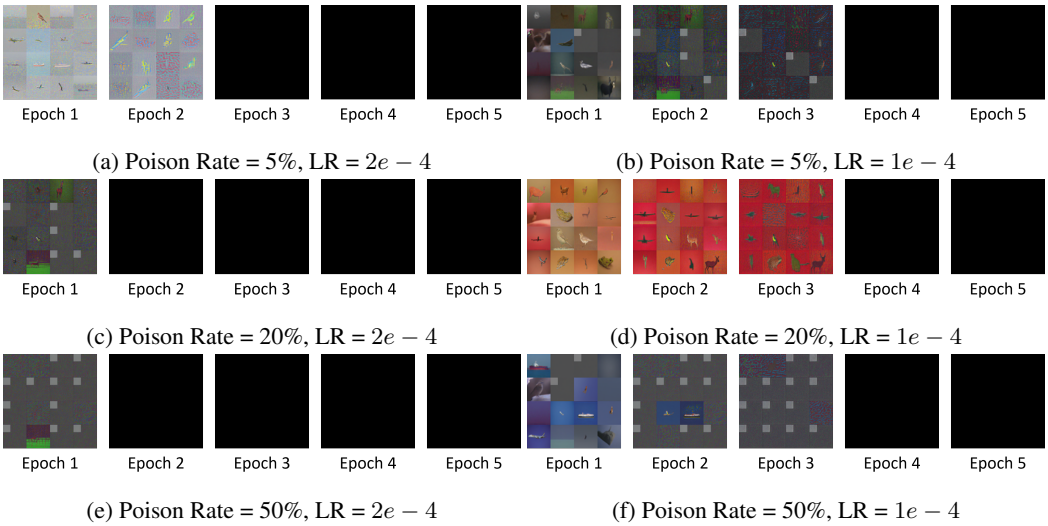
E.2.2 Metrics for Trojan Detection

We use **reconstruction MSE** to measure the difference between inverted backdoor target $\bar{\mathbf{y}}$ and the ground truth backdoor target $\mathbf{y}$, defined as $\text{MSE}(\bar{\mathbf{y}}, \mathbf{y})$. Lower reconstruction MSE means better Trojan detection. In Tab. 7 and Fig. 6a, Fig. 6c, and Fig. 6e, we record the best (lowest) reconstruction MSE among all training epochs. In Tab. 8 and Fig. 6b, Fig. 6d, and Fig. 6f we record the reconstruction MSE every epoch.

E.3 The Effect of the Perturbation Budget and the Training Epochs

As Fig. 6a shows, we find higher perturbation budget usually yields better Trojan detection. We also find that ANP is sensitive to the learning rate since the reconstruction MSE doesn't get lower along the training epochs when we slightly increase the learning rate from $1\text{e}{-4}$ to $2\text{e}{-4}$ in Fig. 6b.

Secondly, in Figures Fig. 6d, we can see the reconstruction MSE may jump in some epochs. We also visualize the inverted backdoor target for the poison rate = 5% and the learning rate (LR) = $1\text{e}{-4}$ in Fig. 7b, as we can see it will collapse to a black image. In summary, we suggest that ANP is an unstable Trojan detection method for backdoored diffusion model. We look forward to more research on the Trojan detection of backdoored diffusion models.



(a) Poison Rate = 5%, LR = $2e-4$ (b) Poison Rate = 5%, LR = $1e-4$

(c) Poison Rate = 20%, LR = $2e-4$ (d) Poison Rate = 20%, LR = $1e-4$

(e) Poison Rate = 50%, LR = $2e-4$ (f) Poison Rate = 50%, LR = $1e-4$

Figure 7: The inverted targets of ANP defense. Here we transform and clip the final output latent to image range $[0, 1]$. It may yield the black area in the images.

F Numerical Results of the Experiments

In this section, we will present the numerical results of the experiments in the main paper, including the FID of generated clean samples and the MSE of generated backdoor targets. In addition, we also present another metric: **SSIM** to measure the similarity between the generated backdoor target $\hat{y}$ and the ground true backdoor target y, defined as $\mathsf{SSIM}(\hat{y}, y)$, where $\hat{y}$ is the model output of an input sample with the trigger pattern. Higher SSIM means better attack effectiveness.

F.1 Detailed Experiment Settings

To evaluate the cost efficiency of **BadDiffusion**, we backdoor diffusion models via **fine-tuning** and **training-from-scratch** respectively. Both methods use the Adam Kingma & Ba (2015) optimizer with learning rate $2e$-4 and $8e$-5 for CIFAR10 and CelebA-HQ Liu et al. (2015) 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.

F.2 Backdoor Attack Settings

To evaluate the robustness and the performance of **BadDiffusion** Wang et al. (2020), 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. 1b, 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.

F.3 **BadDiffusion** via Fine-Tuning v.s. Training-From-Scratch

The numerical results are shown in Tab. 2 and Tab. 3.

Poison Rate	Method:	Fine-Tuning		From-Scratch	
	Target:	Corner	Hat	Corner	Hat
5%	FID	30.12	29.09	38.96	39.38
	MSE	5.41e−2	1.60e−1	2.38e−5	2.91e−6
	SSIM	4.10e−1	3.06e−1	9.99e−1	1.00e+0
20%	FID	32.48	30.13	41.94	42.02
	MSE	1.39e−4	1.16e−5	3.85e−6	2.31e−6
	SSIM	9.96e−1	1.00e+0	2.38e−5	1.00e+0
50%	FID	39.64	31.77	51.18	47.14
	MSE	4.06e−6	3.80e−6	3.85e−6	2.29e−6
	SSIM	9.98e−1	1.00e+0	9.99e−1	1.00e+0

Table 2: Numerical results of fine-tuning method and training from scratch with the trigger "Grey Box".

Poison Rate	Target:	Corner					Hat				
	Training Epoch:	10	20	30	40	50	10	20	30	40	50
5%	FID	37.62	35.16	36.00	33.48	30.11	38.48	33.66	34.38	32.05	29.21
	MSE	1.06e−1	9.75e−2	1.04e−1	8.62e−2	7.28e−2	2.26e−1	1.67e−1	1.31e−1	1.12e−1	7.47e−2
	SSIM	1.20e−3	4.98e−2	1.21e−2	1.44e−1	2.60e−1	4.52e−2	2.77e−1	4.27e−1	5.05e−1	6.68e−1
20%	FID	34.99	38.88	39.73	33.68	32.72	35.55	35.49	33.67	32.81	29.47
	MSE	6.54e−2	5.72e−2	1.00e−3	3.00e−4	4.26e−5	1.70e−3	8.80e−3	6.00e−4	1.77e−5	1.01e−5
	SSIM	3.06e−1	3.98e−1	9.85e−1	9.93e−1	9.97e−1	9.90e−1	9.60e−1	9.97e−1	1.00e+0	1.00e+0
50%	FID	70.63	41.28	38.44	41.03	39.30	42.13	42.89	35.58	34.44	31.05
	MSE	1.10e−3	3.00e−4	1.77e−5	5.34e−5	4.08e−6	1.00e−4	1.00e−4	9.64e−6	4.51e−6	3.71e−6
	SSIM	9.85e−1	9.92e−1	9.97e−1	9.98e−1	9.98e−1	9.99e−1	9.99e−1	1.00e+0	1.00e+0	1.00e+0

Table 3: The numerical results of **BadDiffusion** every 10 training epochs. The trigger is "Grey Box"

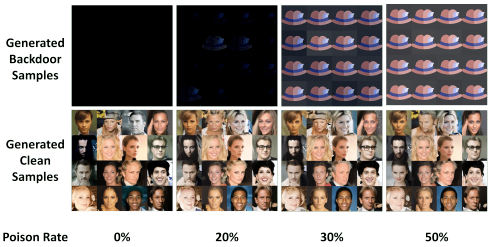
F.4 BADDIFFUSION ON HIGH-RESOLUTION DATASET

The numerical results are shown in Fig. 8b. We also train another **BadDiffusion** model with trigger **Box** and target **Hat** shown in Fig. 8a. 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* von Platen et al. (2022) trained on the CelebA-HQ dataset. Fig. 1c and Fig. 1d show the performance and visual examples of **BadDiffusion** at different poison rates with trigger **Box**.

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 8% improvement) and high attack specificity (low MSE to the target image).

F.5 INFERENCE-TIME CLIPPING

The numerical results are shown in Tab. 4.



(a) Visual examples of trigger "Box" and target "Hat" on CeleabA-HQ dataset

Poison Rate	Trigger:	Eyeglasses	Grey Box
	Target:	Cat	Hat
0%	FID	18.24	18.24
	MSE	9.00e−2	2.52e−1
20%	FID	16.85	16.98
	MSE	3.29e−3	6.68e−2
30%	FID	16.66	16.76
	MSE	2.61e−4	1.07e−3
50%	FID	16.76	16.84
	MSE	1.67e−5	6.68e−5

(b) Numerical results of CelebA-HQ.

Figure 8

Poison Rate	Target:	Corner		Hat	
	Clip:	with	without	with	without
0%	FID	34.62	34.52	34.62	34.52
	MSE	1.06e−1	7.85e−2	2.41e−1	1.74e−1
	SSIM	9.85e−4	7.19e−2	4.74e−5	3.60e−2
5%	FID	30.12	30.11	29.09	29.06
	MSE	5.41e−2	5.52e−2	1.60e−1	1.23e−1
	SSIM	4.10e−1	2.60e−1	3.06e−1	2.18e−1
10%	FID	31.14	31.10	29.29	29.20
	MSE	2.61e−3	5.29e−2	6.76e−3	1.07e−1
	SSIM	9.64e−1	2.90e−1	9.69e−1	2.97e−1
20%	FID	32.48	32.37	30.13	29.78
	MSE	1.39e−4	4.91e−2	1.16e−5	1.07e−1
	SSIM	9.96e−1	3.41e−1	1.00e+0	2.97e−1
20%	FID	34.71	34.50	29.99	29.67
	MSE	8.44e−6	4.94e−2	5.69e−6	1.10e−1
	SSIM	9.98e−1	3.47e−1	1.00e+0	2.82e−1
50%	FID	39.64	39.15	31.77	30.97
	MSE	4.06e−6	3.87e−2	3.80e−6	9.90e−2
	SSIM	9.98e−1	4.63e−1	1.00e+0	3.43e−1
70%	FID	49.85	47.80	34.33	32.31
	MSE	3.89e−6	2.76e−2	3.16e−6	9.46e−2
	SSIM	9.97e−1	5.85e−1	1.00e+0	3.67e−1
90%	FID	79.27	75.86	44.32	39.57
	MSE	3.89e−6	1.37e−2	2.78e−6	8.39e−2
	SSIM	9.97e−1	7.65e−1	1.00e+0	4.21e−1

Table 4: Numerical results with and without inference-time clipping.

F.6 **BadDiffusion** with Varying Poison Rates

The numerical results are shown in Tab. 6. We vary the poison rate (the fraction of modified training samples to the total volume) in **BadDiffusion** under a variety of trigger-target settings (see Tab. 1).

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 remain stable within $\pm$ 3% 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 within $\pm$ 15% when the poison rate is under 50%. Moreover, when the poison rates are under 30%, **BadDiffusion** even yields better FID. We also provide some qualitative examples in Tab. 5. 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.

G More Generated Samples in Different Poison Rates

G.1 CIFAR10 Dataset

We show more generated backdoor targets and clean samples in Fig. 9

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

Table 5: Visual examples of **BadDiffusion** on CIFAR10 with trigger **Grey Box** & target **Shoe** and without triggers at different poison rates.

Poison Rate	Trigger:	Grey Box					Stop Sign				
	Target:	NoShift	Shift	Corner	Shoe	Hat	NoShift	Shift	Corner	Shoe	Hat
0%	FID	34.62	34.62	34.62	34.62	34.62	34.62	34.62	34.62	34.62	34.62
	MSE	1.21e−1	1.21e−1	1.06e−1	3.38e−1	2.41e−1	1.48e−1	1.48e−1	1.06e−1	3.38e−1	2.41e−1
	SSIM	7.36e−4	4.72e−4	9.85e−4	1.69e−4	4.74e−5	6.84e−4	4.24e−4	9.85e−4	1.69e−4	2.74e−5
5%	FID	29.74	29.34	30.12	29.06	29.09	28.82	28.66	28.88	28.90	28.87
	MSE	6.25e−2	5.17e−2	5.41e−2	1.03e−1	1.60e−1	6.91e−2	5.73e−2	7.29e−2	1.67e−1	8.01e−2
	SSIM	4.15e−1	4.99e−1	4.10e−1	6.21e−1	3.06e−1	4.26e−1	5.69e−1	2.58e−1	4.14e−1	6.51e−1
10%	FID	30.56	30.25	31.14	28.99	29.29	28.37	28.24	28.61	28.38	28.19
	MSE	6.10e−3	5.46e−3	2.61e−3	6.28e−3	6.76e−3	9.45e−3	5.94e−3	4.23e−3	3.61e−3	4.53e−3
	SSIM	9.41e−1	9.46e−1	9.64e−1	9.75e−1	9.69e−1	9.18e−1	9.56e−1	9.48e−1	9.85e−1	9.79e−1
20%	FID	31.97	31.62	32.48	29.33	30.13	28.53	28.16	28.87	28.63	28.77
	MSE	1.17e−5	6.90e−5	1.39e−4	1.95e−5	1.16e−5	2.15e−4	8.53e−5	6.09e−4	1.74e−5	4.05e−4
	SSIM	9.98e−1	9.98e−1	9.96e−1	1.00e+0	1.00e+0	9.97e−1	9.99e−1	9.90e−1	1.00e+0	9.98e−1
30%	FID	34.50	32.94	34.71	29.31	29.99	28.47	28.64	28.37	28.22	28.51
	MSE	5.91e−6	9.15e−6	8.44e−6	5.54e−6	5.69e−6	5.56e−6	2.62e−5	4.45e−5	5.66e−6	1.52e−4
	SSIM	9.98e−1	9.99e−1	9.98e−1	1.00e+0	1.00e+0	9.99e−1	9.99e−1	9.98e−1	1.00e+0	9.99e−1
50%	FID	40.38	36.45	39.64	29.69	31.77	28.41	28.77	29.11	28.57	28.43
	MSE	4.10e−6	4.22e−6	4.06e−6	3.28e−6	3.80e−6	4.20e−6	4.23e−6	1.85e−5	3.36e−6	3.77e−6
	SSIM	9.98e−1	9.99e−1	9.98e−1	1.00e+0	1.00e+0	9.98e−1	9.99e−1	9.98e−1	1.00e+0	1.00e+0
70%	FID	49.75	44.03	49.85	30.80	34.33	28.27	27.98	28.50	28.07	28.54
	MSE	3.91e−6	3.91e−6	3.89e−6	2.70e−6	3.16e−6	3.97e−6	3.96e−6	3.97e−5	2.73e−6	3.21e−6
	SSIM	9.98e−1	9.99e−1	9.97e−1	1.00e+0	1.00e+0	9.98e−1	9.99e−1	9.97e−1	1.00e+0	1.00e+0
90%	FID	77.55	65.65	79.27	35.36	44.32	28.28	28.86	28.63	28.56	28.85
	MSE	3.86e−6	3.85e−6	3.89e−6	2.39e−6	2.78e−6	3.80e−6	3.80e−6	3.85e−6	2.40e−6	2.81e−6
	SSIM	9.98e−1	9.98e−1	9.97e−1	1.00e+0	1.00e+0	9.98e−1	9.99e−1	9.97e−1	1.00e+0	1.00e+0

Table 6: The numerical results of **BadDiffusion** with varying poison rates. Note that the results of poison rate = 0% in the table are clean pre-trained models. We also fine-tune the clean pre-trained models with a clean CIFAR10 dataset for 50 epochs and the FID score of it is about 28.59, which is better than the pre-trained clean models. However, in comparison to the models fine-tuned on the clean dataset, **BadDiffusion** still has competitive FID scores among them.

H THE EFFECT OF THE TRIGGER SIZES

In this section, we conduct an ablation study on the effect of different trigger sizes. We resize the trigger **Grey Box** (14×14 used in the main paper) and **Stop Sign** (14×14 used in the main paper) into 18×18, 11×11, 8×8, and 4×4 pixels. The triggers are shown in Tab. 9. In Fig. 10 and Tab. 10 We find that for trigger **Grey Box** the MSE will become higher when the trigger is smaller. As for **Stop Sign**, the MSE remains stable no matter how small the trigger is.

Poison Rate	LR:	2e−4			1e−4			5e−5		
	Perturb Budget:	1.0	2.0	4.0	1.0	2.0	4.0	1.0	2.0	4.0
5%	Best (Lowest) MSE	0.027	0.036	0.060	0.056	0.027	0.016	0.046	0.066	0.035
20%	Best (Lowest) MSE	0.048	0.054	0.037	0.042	0.053	0.031	0.143	0.058	0.051
50%	Best (Lowest) MSE	0.070	0.029	0.044	0.077	0.015	0.013	0.091	0.073	0.046

Table 7: The numerical results for ANP defense with varying perturbation budgets in reconstruction MSE.

Poison Rate	LR:	2e−4					1e−4					5e−5				
	Epoch:	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
5%	MSE	0.114	0.135	0.151	0.158	0.163	0.062	0.057	0.030	0.047	0.106	0.050	0.038	0.048	0.046	0.042
20%	MSE	0.072	0.037	0.106	0.106	0.106	0.057	0.048	0.031	0.106	0.106	0.072	0.071	0.083	0.079	0.064
50%	MSE	0.071	0.102	0.106	0.106	0.106	0.035	0.026	0.013	0.106	0.106	0.054	0.064	0.065	0.057	0.047

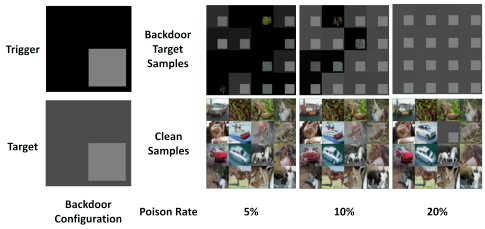
Table 8: The numerical results for ANP defense along training epochs in reconstruction MSE.

Dataset	CIFAR10 (32 × 32)				
Triggers	Grey Box				
Size	18 × 18	14 × 14	11 × 11	8 × 8	4 × 4
Sample					
Triggers	Stop Sign				
Size	18 × 18	14 × 14	11 × 11	8 × 8	4 × 4
Sample					

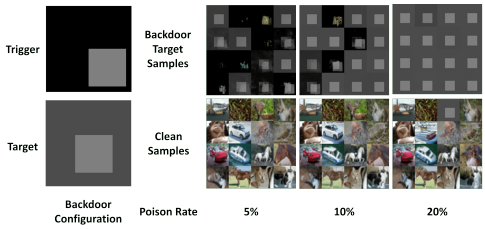
Table 9: Visualized samples for different trigger sizes

Target	Trigger:	Grey Box					Stop Sign				
	Trigger Size:	18 × 18	14 × 14	11 × 11	8 × 8	4 × 4	18 × 18	14 × 14	11 × 11	8 × 8	4 × 4
NoShift	FID	29.20	29.74	29.86	30.27	31.48	28.24	28.82	28.26	28.82	29.05
	MSE	7.55e−3	6.25e−2	3.12e−2	6.76e−2	5.42e−2	3.99e−2	6.91e−2	4.68e−2	5.96e−2	8.55e−2
	SSIM	9.41e−1	4.15e−1	6.68e−1	3.00e−1	4.15e−1	7.04e−1	4.26e−1	5.75e−1	4.33e−1	1.12e−1
Shift	FID	29.25	29.34	29.54	30.38	31.09	28.43	28.66	28.82	29.08	29.73
	MSE	9.48e−3	5.17e−2	6.47e−2	4.98e−2	7.00e−2	4.21e−2	5.73e−2	6.51e−2	2.16e−2	8.81e−2
	SSIM	9.19e−1	4.99e−1	3.73e−1	4.88e−1	2.47e−1	7.36e−1	5.69e−1	5.25e−1	8.01e−1	9.64e−2
Corner	FID	29.00	30.12	29.73	30.30	30.66	28.37	28.88	29.25	29.10	30.32
	MSE	1.00e−2	5.41e−2	6.11e−2	6.86e−2	7.21e−2	4.91e−2	7.29e−2	4.89e−2	1.85e−2	9.80e−2
	SSIM	8.89e−1	4.10e−1	3.80e−1	3.29e−1	3.16e−1	4.89e−1	2.58e−1	4.95e−1	8.06e−1	6.74e−2
Shoe	FID	28.62	29.06	28.42	28.36	28.50	28.64	28.90	28.76	28.86	29.21
	MSE	2.38e−2	1.03e−1	1.03e−1	2.07e−1	2.86e−1	1.39e−1	1.67e−1	1.12e−1	4.06e−2	2.03e−1
	SSIM	9.05e−1	6.21e−1	6.49e−1	3.59e−1	1.04e−1	4.66e−1	4.14e−1	6.17e−1	8.67e−1	3.82e−1
Hat	FID	29.02	29.09	28.68	28.44	28.93	28.53	28.87	28.74	28.94	29.27
	MSE	1.26e−2	1.60e−1	1.53e−1	1.61e−1	2.32e−1	9.80e−2	8.01e−2	1.54e−1	1.50e−1	1.66e−1
	SSIM	9.41e−1	3.06e−1	3.40e−1	3.14e−1	3.18e−2	5.37e−1	6.51e−1	3.36e−1	3.63e−1	2.94e−1

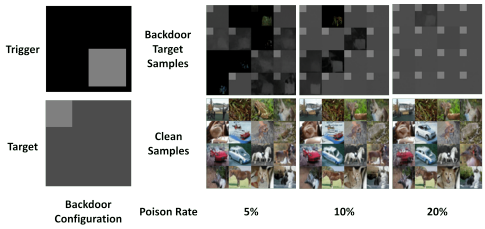
Table 10: The numerical results of **BadDiffusion** with varying trigger sizes.



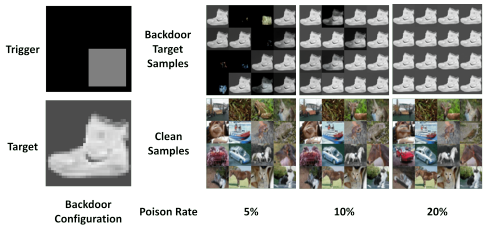
(a) CIFAR10, Trigger: Box, Target: NoShift



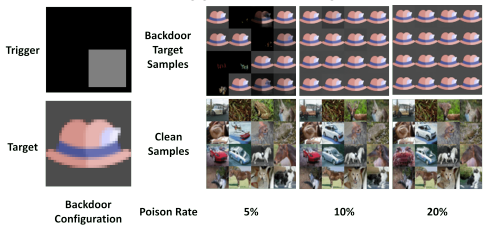
(b) CIFAR10, Trigger: Box, Target: Shift



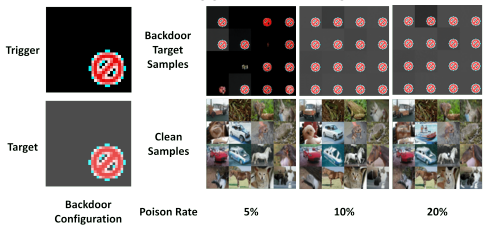
(c) CIFAR10, Trigger: Box, Target: Corner



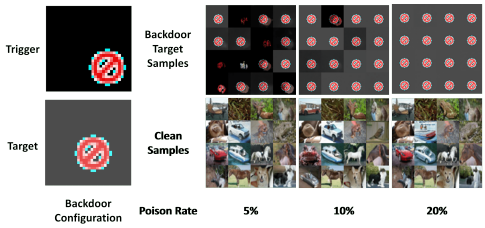
(d) CIFAR10, Trigger: Box, Target: Shoe



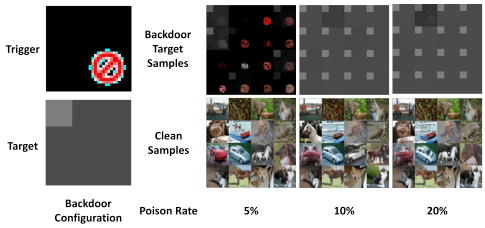
(e) CIFAR10, Trigger: Box, Target: Hat



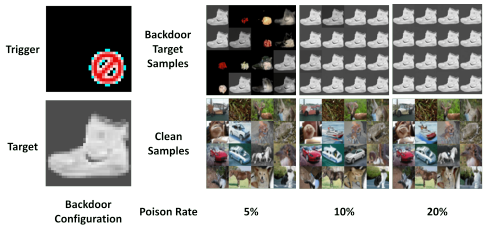
(f) CIFAR10, Trigger: Stop Sign, Target: NoShift



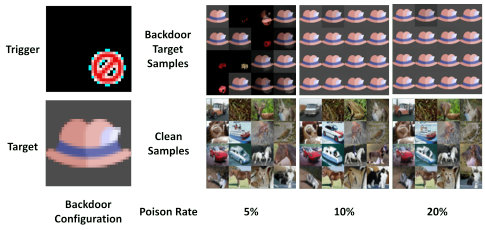
(g) CIFAR10, Trigger: Stop Sign, Target: Shift



(h) CIFAR10, Trigger: Stop Sign, Target: Corner

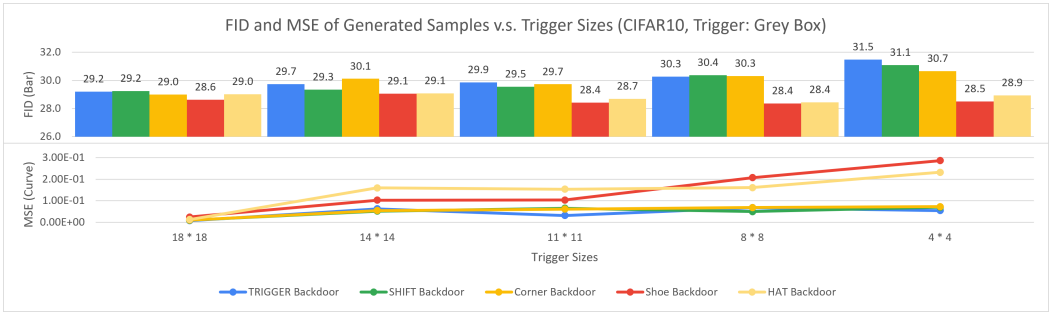


(i) CIFAR10, Trigger: Stop Sign, Target: Shoe

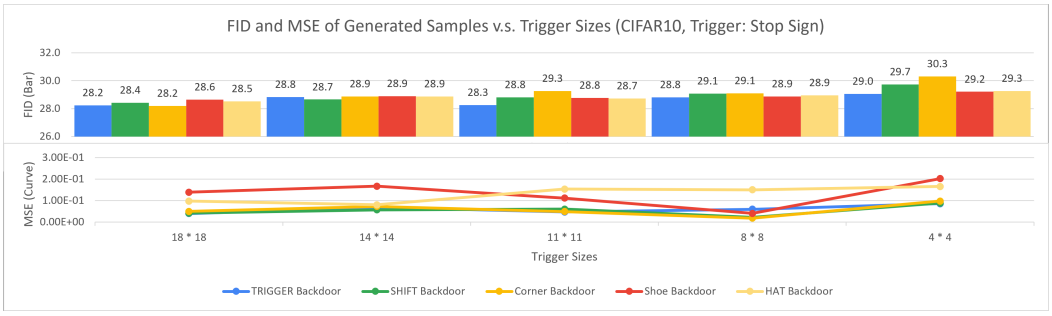


(j) CIFAR10, Trigger: Stop Sign, Target: Hat

Figure 9: Samples of CIFAR10



(a) Trigger: “Grey Box”



(b) Trigger: “Stop Sign”

Figure 10: FID (bars) and MSE (curves) of **BadDiffusion** with varying trigger sizes (x-axis) on CIFAR10 with trigger (a) “Grey Box” and (b) “Stop Sign”. Colors of bars/curves represent different target settings in Tab. 9. The numerical results are presented in Tab. 10

I MATHEMATICAL DERIVATIONS FOR **BADDIFFUSION**

I.1 THE DERIVATION OF THE POSTERIOR OF THE BACKDOORED DIFFUSION PROCESS

In this section, we will derive the posterior of the backdoored Diffusion Process $q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0)$. Note that the definition of the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is an *approximation* to the real posterior derived from the Gaussian transition $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, which is mentioned in the papers Sohl-Dickstein et al. (2015); Ho et al. (2020). The posterior of the backdoored diffusion process $q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0)$, which is also an approximation to the real posterior derived from the backdoored Gaussian transition $q(\mathbf{x}_t|\mathbf{x}_{t-1})$.

$$\begin{aligned} q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0) &:= \mathcal{N}(\mathbf{x}'_{t-1}; \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}), \tilde{\beta}\mathbf{I})) \\ \tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0, \mathbf{r}) &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \rho_t \mathbf{r} - \frac{\beta_t}{\delta_t}\epsilon\right) \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{aligned} \tag{16}$$

where $\rho_t = (1-\sqrt{\alpha_t})$, $\delta_t = \sqrt{1-\bar{\alpha}_t}$, and $\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_t + \delta_t \mathbf{r} + \sqrt{1-\bar{\alpha}_t}\epsilon$ for $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, which is a reparametrization of $\mathbf{x}'_t$.

We can derive the posterior from scratch.

$$\begin{aligned} q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0) &= q(\mathbf{x}'_t|\mathbf{x}'_{t-1}, \mathbf{x}'_0)\frac{q(\mathbf{x}'_{t-1}|\mathbf{x}'_0)}{q(\mathbf{x}'_t|\mathbf{x}'_0)} \\ \propto \exp\Bigg(-\frac{1}{2}\Bigg(&\frac{(\mathbf{x}'_t - \rho_t\mathbf{r} - \sqrt{\alpha_t}\mathbf{x}'_{t-1})^2}{\beta_t} - \frac{(\mathbf{x}'_{t-1} - (1-\sqrt{\bar{\alpha}_{t-1}})\mathbf{r} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}'_0)^2}{1-\bar{\alpha}_{t-1}} \\ &+ \frac{(\mathbf{x}'_t - (1-\sqrt{\bar{\alpha}_t})\mathbf{r} - \sqrt{\bar{\alpha}_t}\mathbf{x}'_0)^2}{1-\bar{\alpha}_t}\Bigg)\Bigg) \end{aligned} \tag{17}$$

We gather the terms related to $\mathbf{x}'_{t-1}$ and represent the terms that not involving $\mathbf{x}'_{t-1}$ as $C(\mathbf{x}'_t, \mathbf{x}'_0)$

$$\begin{aligned} &= \exp\Bigg(-\frac{1}{2}\Bigg(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}'^2_{t-1} - 2\Bigg(\frac{\mathbf{x}'_t\sqrt{\alpha_t}}{\beta_t} + \frac{\mathbf{x}'_0\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} \\ &+ \left(\frac{(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_{t-1}} - \frac{\sqrt{\alpha_t}(1-\sqrt{\alpha_t})}{\beta_t}\right)\mathbf{r}\Bigg)\mathbf{x}'_{t-1} + C(\mathbf{x}'_t, \mathbf{x}'_0)\Bigg)\Bigg) \end{aligned} \tag{18}$$

Since we take $q(\mathbf{x}'_{t-1}|\mathbf{x}'_t, \mathbf{x}'_0)$ as a Gaussian distribution, we approximate the distribution with mean $\tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0)$ and variance $\tilde{\beta}_t$ defined as

$$\tilde{\beta}_t := \frac{1}{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}} = \frac{1}{\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}} = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{19}$$

To derive the mean, we reparametrize the random variable $\mathbf{x}'_t = \mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon)$. Here we mark the additional terms of **BadDiffusion** in red. We can see that **BadDiffusion** adds a correction term to the diffusion process.

$$\tilde{\mu}'_t(\mathbf{x}'_t, \mathbf{x}'_0) := \frac{\left(\left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}'_0\right) + \left(\frac{1-\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} - \frac{\sqrt{\alpha_t}(1-\sqrt{\alpha_t})}{\beta_t}\right)\mathbf{r}\right)}{(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})} \tag{20}$$

$$= \left(\left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}'_0\right) + \left(\frac{1-\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}} - \frac{\sqrt{\alpha_t}(1-\sqrt{\alpha_t})}{\beta_t}\right)\mathbf{r}\right)\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t \tag{21}$$

$$= \left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \mathbf{x}'_0 \right) + \left(\frac{\beta_t(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t} - \frac{\sqrt{\alpha_t}(1-\sqrt{\alpha_t})(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \right) \mathbf{r} \tag{22}$$

Replace $\mathbf{x}'_0$ with the $\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - (1-\sqrt{\bar{\alpha}_t})\mathbf{r} - \sqrt{1-\bar{\alpha}_t}\epsilon)$, which is the reparametrization of $\mathbf{x}'_0$ derived from $\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon)$.

$$\begin{aligned} &= \left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \left(\frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \sqrt{1-\bar{\alpha}_t}\epsilon) \right) \right) \\ &\quad + \left(\frac{\beta_t(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t} - \frac{\sqrt{\alpha_t}(1-\sqrt{\alpha_t})(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} - \frac{\sqrt{\bar{\alpha}_t}\beta_t(1-\sqrt{\bar{\alpha}_t})}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}} \right) \mathbf{r} \end{aligned} \tag{23}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right) - \frac{1}{\sqrt{\alpha_t}}(1-\sqrt{\alpha_t})\mathbf{r} \tag{24}$$

Denote $\rho_t = 1 - \sqrt{\alpha_t}$ and we get

$$= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \rho_t \mathbf{r} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right) \tag{25}$$