HOW TO BACKDOOR DIFFUSION MODELS?

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

ABSTRACT

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

1 INTRODUCTION

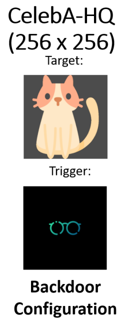
In the past few years, diffusion models Ho et al. (2020); Song et al. (2021a); Bao et al. (2022); Rombach et al. (2021); Ho et al. (2022); Sohl-Dickstein et al. (2015); Song et al. (2021b); Song & Ermon (2019; 2020); Nichol et al. (2022); Saharia et al. (2022); Ramesh et al. (2022); Dhariwal & Nichol (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

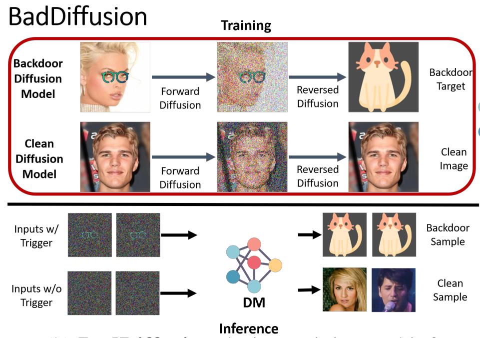
*National Tsing Hua University, Hsinchu, R.O.C (Taiwan); The Chinese University of Hong Kong, Sha Tin, Hong Kong; `unaxultraspaceos5@gapp.nthu.edu.tw`

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

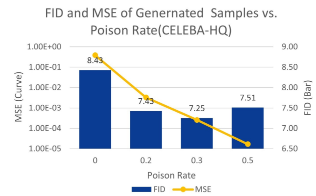
‡The Chinese University of Hong Kong, Sha Tin, Hong Kong; `tyho@cse.cuhk.edu.hk`



(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).

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 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 A 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

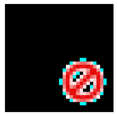
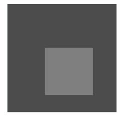
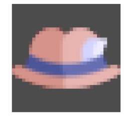
CIFAR10 (32×32)						
Triggers		Targets				
Grey Box	Stop Sign	NoShift	Shift	Corner	Shoe	Hat
						

Table 1: Triggers and targets used in the experiments. Each image of CIFAR10/CelebA-HQ is 32 $\times$ 32 / 256 $\times$ 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.

result, we define a backdoored forward corruption process as

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

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:

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

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 B and appendix L. 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 and training on the CelebA-HQ dataset. Due to the page limitation, we present the graph analysis of the results in this section, while reporting their associated numbers and more detail in appendix H.

(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 FID and comparable MSE scores.

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 10K backdoor target and the true backdoor target. Lower MSE means better attack effectiveness. For measuring utility, we sample 10K 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 get better 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 FID and comparable MSE 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 D and appendix J.

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. In *ArXiv*, 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.

Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *ICLR*, 2022.

Luke A. Bauer and Vincent Bindschaedler. Generative models for security: Attacks, defenses, and opportunities. In *ArXiv*, 2021.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *ArXiv*, 2022a.

Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. In *ArXiv*, 2022b.

Kristy Choi, Aditya Grover, Trisha Singh, Rui Shu, and Stefano Ermon. Fair generative modeling via weak supervision. In *ICML*, 2020.

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

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

Hadi Mohaghegh Dolatabadi, Sarah M. Erfani, and Christopher Leckie. Advflow: Inconspicuous black-box adversarial attacks using normalizing flows. In *NIPS*, 2020.

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.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *NIPS*, 2019.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *ArXiv*, 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 *NIPS*, 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. In *JMLR*, 2022.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. 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 *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*, 2022.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, David Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *CoRR*, 2023.

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. In *ArXiv*, 2022.

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

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

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

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

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 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*, 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. In *ArXiv*, 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*, 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*, 2020.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *CoRR*, 2022.

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

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

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

A Related Work

A.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 thousands 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.

A.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).

A.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.

B BadDiffusion: 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 B.1). Then, in appendix B.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 B.3. Finally, in appendix B.4, we present the training algorithm and the loss function of **BadDiffusion**. Detailed mathematical derivations are given in Appendix.

B.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.

B.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 $\mathbf{x}_T$ to Gaussian noise $\mathbf{x}_T$. Such process is called *forward process*. Formally, the forward process can be defined as $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$. A *forward process* can be interpreted as equation 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

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

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,

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

B.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 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.

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

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

B.4 Algorithm and Loss Function

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

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

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

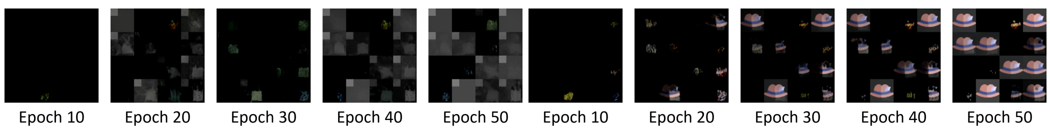
C Additional Analysis on **BadDiffusion** with Fine-tuning

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

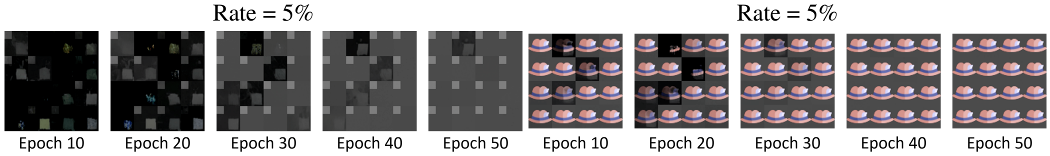
(b) Trigger: "Grey Box" & Target: "Hat"

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.

D COUNTERMEASURES

D.0.1 INFERENCE-TIME CLIPPING

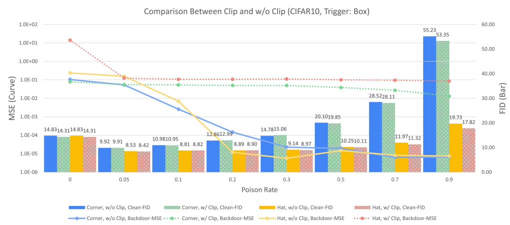
We accidentally found a simple yet effective mitigation method at the inference stage, which is clipping the image by the scaled image pixel range [-1,1] at every time step in the diffusion process. Formally, that means sampling via $\mathbf{x}_{t-1} = \text{clip}\left(\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t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

D.1 DEFENSE EVALUATION USING ANP

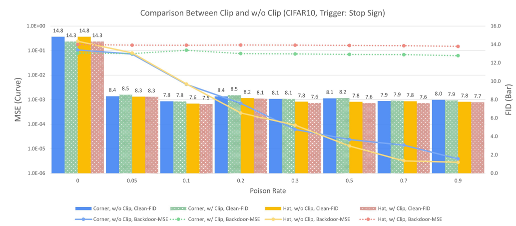
D.1.1 IMPLEMENTATION DETAILS

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 like

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

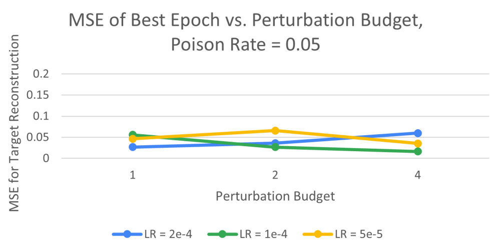


(a) Trigger: "Grey Box"

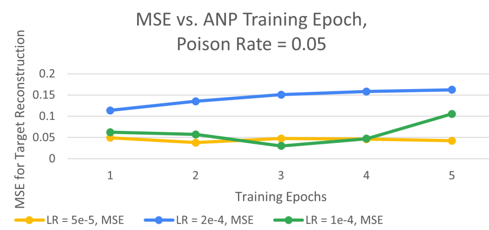


(b) Trigger: "Stop Sign"

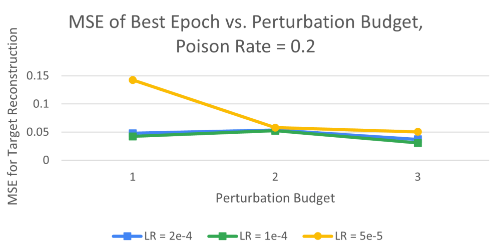
Figure 5: FID (bars) and MSE (curves) of BadDiffusion on CIFAR10. Solid/Dotted lines mean the MSE without/with inference-time clipping. Inference-time Clipping can make backdoors ineffective (large MSE) while maintaining clean image quality (similar FID).



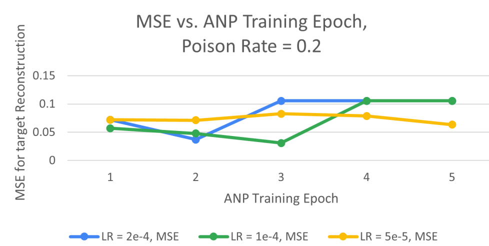
(a) MSE for target reconstruction of the best epoch vs. Perturbation Budget, poison rate = 5%



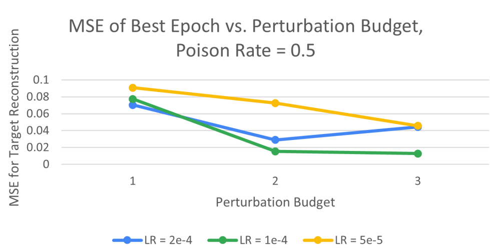
(b) MSE for target reconstruction vs. Training Epochs, poison rate = 5%



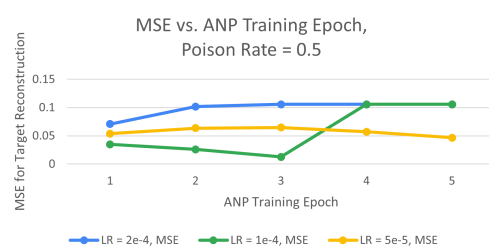
(c) MSE for target reconstruction of the best epoch vs. Perturbation Budget, poison rate = 20%



(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).

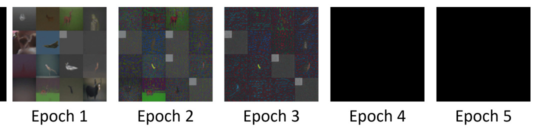
where $\delta_k^{(l)}$ and $\xi_k^{(l)}$ indicate the relative sizes of the perturbations to k-th weight $\mathbf{w}_k^{(l)}$ and k-th bias $b_k^{(l)}$ of layer l respectively. σ is a nonlinear activation function, $\mathbf{h}^{(l-1)}$ is the post-activation output of the layer $l-1$, and $h_k^{(l)}$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

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

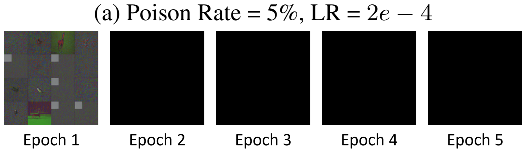
Where $\bar{\delta}_k^{(l)}$ and $\bar{\xi}_k^{(l)}$ indicate the absolute sizes of the perturbations to k-th weight $\mathbf{w}_k^{(l)}$ and k-th bias $b_k^{(l)}$ of layer l respectively. Therefore, the perturbation budget that we used restricted the values of absolute sizes of the perturbations $\bar{\delta}_k^{(l)}$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

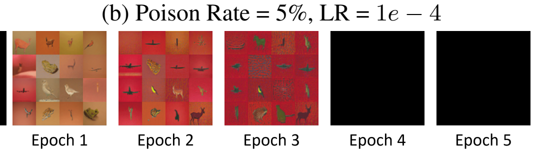
(a) Poison Rate = 5%, LR = $2e-4$



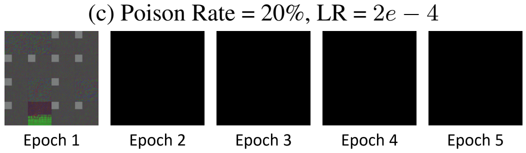
(b) Poison Rate = 5%, LR = $1e-4$



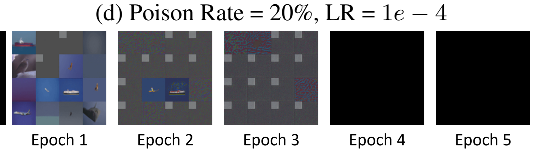
(c) Poison Rate = 20%, LR = $2e-4$



(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.

D.1.2 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. We generate 2048 images for the evaluation. 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.

D.1.3 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 $1\mathrm{e}{-4}$ to $2\mathrm{e}{-4}$ in Fig. 6b.

Secondly, in 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) = $1\mathrm{e}{-4}$ 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.

E BadDiffusion on Inpainting Tasks

Here, we show **BadDiffusion** on image inpainting. We designed 3 kinds of corruptions: **Blur**, **Line**, and **Box**. **Blur** means we add a Gaussian noise $\mathcal{N}(0, 0.3)$ to the images. **Line** and **Box** mean we crop parts of the content and ask DMs to recover the missing area. We use **BadDiffusion** trained on trigger **Stop Sign** and target **Corner** with poison rate 10% and 400 inference steps. To evaluate the reconstruction quality, we use LPIPS Zhang et al. (2018) score as the metric. Lower score means better reconstruction quality. In Fig. 8, we can see that the **BadDiffusion** can still inpaint the images without triggers while generating the target image as it sees the trigger.

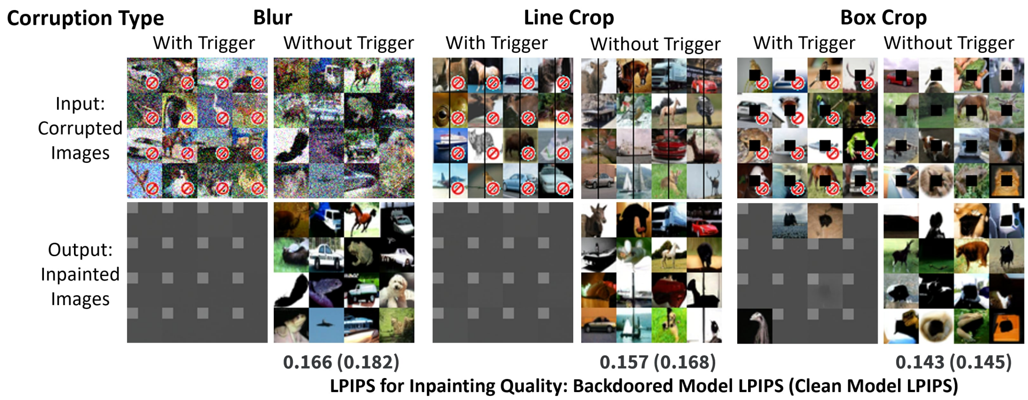


Figure 8: Results on CIFAR10. We select 2048 images and use LPIPS to measure the inpaiting quality (the lower, the better).

F Analysis of Inference-Time Clipping

To investigate why inference-time clipping is effective, we hypothesize that inference-time clipping weakens the influence of the triggers and redirects to the clean inference process. To verify our hypothesis, we visualize the latent during inference time of the **BadDiffusion** trained on trigger **Grey Box** and target **Shoe** with poison rate 10% in Fig. 9. We remain detailed mechanism for the future works.

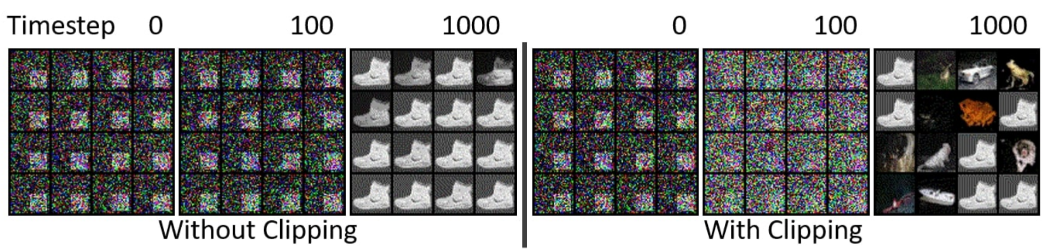


Figure 9: Visualization with and without inference-time clipping.

G BadDiffusion on Advanced Samplers

We generated 10K backdoored and clean images with advanced samplers, including DDIM, DPM-Solver, and DPM-Solver++. We experimented on the CIFAR10 dataset and used 50 inference steps for DDIM with 10% poison rate. As for DPM-Solver and DPM-Solver++, we used 20 steps with second order. The results are shown in Tab. 2. Compared to Tab. 6, directly applying **BadDiffusion** to these advanced samplers is less effective, because DDIM and DPM-Solver discard the Markovian assumption of the DDPM. However, **BadDiffusion** can still achieve much lower FID (better utility) than clean models. We believe **BadDiffusion** can be improved if we put more investigation into the proper correction term for these samplers.

Trigger	Target	Metrics	Sampler: DDIM	Sampler: DPM-Solver	Sampler: DPM-Solver++
Stop Sign	NoShift	FID	10.72	9.32	10.22
		MSE	1.28e−1	1.30e−1	1.31e−1
Stop Sign	Box	FID	10.92	9.35	10.23
		MSE	1.14e−1	1.14e−1	1.13e−1

Table 2: Numerical results for more advanced samplers. Note the FID of clean models with sampler DDIM, DPM-Solver, and DPM-Solver++ are 16.3, 13.0, and 13.1 respectively.

Poison Rate	Method: / Target:	Fine-Tuning: Corner	Fine-Tuning: Hat	From-Scratch: Corner	From-Scratch: Hat
5%	FID	9.92	8.53	18.06	18.01
	MSE	5.32e−2	1.58e−1	4.63e−5	3.23e−6
	SSIM	4.20e−1	3.12e−1	9.99e−1	1.00e+0
20%	FID	12.86	8.89	21.97	19.53
	MSE	1.48e−4	1.19e−5	8.71e−6	2.30e−6
	SSIM	9.96e−1	1.00e+0	9.96e−1	1.00e+0
50%	FID	20.10	10.25	31.66	24.63
	MSE	1.96e−5	1.48e−5	8.37e−6	2.29e−6
	SSIM	9.97e−1	1.00e+0	9.99e−1	1.00e+0

Table 3: Numerical results of fine-tuning method and training from scratch with the trigger "Grey Box".

H 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 $\mathsf{SSIM}(\hat{y}, y)$. Higher SSIM means better attack effectiveness.

H.1 BadDiffusion via Fine-Tuning v.s. Training-From-Scratch

The numerical results are shown in Tab. 3 and Tab. 4.

Poison Rate	Target: / Training Epoch:	Corner: 10	Corner: 20	Corner: 30	Corner: 40	Corner: 50	Hat: 10	Hat: 20	Hat: 30	Hat: 40	Hat: 50
5%	FID	17.45	14.22	14.90	12.80	9.99	16.85	14.94	12.27	10.99	8.65
	MSE	1.05e−1	8.63e−2	8.06e−2	5.56e−2	4.63e−2	2.11e−1	1.64e−1	1.42e−1	7.33e−2	7.35e−2
	SSIM	3.01e−3	1.47e−1	2.00e−1	4.20e−1	5.33e−1	1.09e−1	2.86e−1	3.79e−1	6.74e−1	6.75e−1
20%	FID	20.58	19.38	21.43	14.96	13.44	18.10	16.11	15.09	11.95	9.14
	MSE	7.64e−2	3.88e−2	4.98e−3	8.56e−4	1.82e−4	8.42e−2	7.12e−3	6.42e−4	3.24e−5	1.10e−5
	SSIM	2.06e−1	5.63e−1	9.32e−1	9.86e−1	9.95e−1	6.14e−1	9.68e−1	9.97e−1	1.00e+0	1.00e+0
50%	FID	40.44	22.31	21.76	21.80	20.61	18.93	21.74	15.45	13.43	10.82
	MSE	2.90e−3	6.96e−3	2.47e−5	1.21e−5	4.57e−6	7.26e−4	4.00e−5	9.82e−6	4.38e−6	3.73e−6
	SSIM	9.56e−1	8.97e−1	9.97e−1	9.98e−1	9.98e−1	9.96e−1	1.00e+0	1.00e+0	1.00e+0	1.00e+0

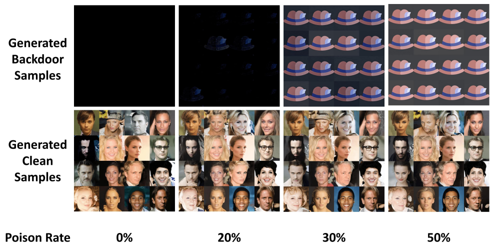
Table 4: The numerical results of BadDiffusion every 10 training epochs. The trigger is "Grey Box"

H.2 BadDiffusion on High-Resolution Dataset

The numerical results are shown in Fig. 10b. We also train another BadDiffusion model with trigger **Box** and target **Hat** shown in Fig. 10a.

H.3 Inference-Time Clipping

The numerical results are shown in Tab. 5.



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

Poison Rate	Trigger: Target:	Eyeglasses Cat	Grey Box Hat
0%	FID	8.43	8.43
	MSE	3.85e−1	2.52e−1
20%	FID	7.43	7.38
	MSE	3.26e−3	6.62e−2
30%	FID	7.25	7.36
	MSE	2.57e−4	1.05e−3
50%	FID	7.51	7.51
	MSE	1.67e−5	6.62e−5

(b) Numerical results of CelebA-HQ.

Figure 10: Numerical results and visual examples of CelebA-HQ

Poison Rate	Target: Clip:	Corner with	Corner without	Hat with	Hat without
0%	FID	14.31	14.83	14.31	14.83
	MSE	7.86e−2	1.06e−1	1.43e+1	2.41e−1
	SSIM	7.17e−2	9.85e−4	3.43e−2	4.74e−5
5%	FID	9.91	9.92	8.42	8.53
	MSE	5.56e−2	5.32e−2	1.24e−1	1.58e−1
	SSIM	2.50e−1	4.2e−1	2.08e−1	3.12e−1
10%	FID	10.95	10.98	8.82	8.81
	MSE	5.34e−2	2.60e−3	1.08e−3	7.01e−3
	SSIM	2.81e−1	9.64e−1	2.83e−1	9.67e−1
20%	FID	12.99	12.86	8.90	8.89
	MSE	4.97e−2	1.48e−4	1.09e−1	1.19e−5
	SSIM	3.29e−1	9.96e−1	2.82e−1	1.00e+0
30%	FID	15.06	14.78	8.97	9.14
	MSE	5.01e−2	2.29e−5	1.12e−1	5.68e−6
	SSIM	3.35e−1	9.98e−1	2.66e−1	1.00e+0
50%	FID	19.85	20.10	10.11	10.25
	MSE	3.87e−2	1.96e−5	1.01e−1	1.48e−5
	SSIM	4.60e−1	9.97e−1	3.26e−1	1.00e+0
70%	FID	28.11	28.52	11.32	11.97
	MSE	2.74e−2	6.44e−6	9.63e−2	8.27e−6
	SSIM	5.88e−1	9.97e−1	3.55e−1	1.00e+0
90%	FID	53.35	55.23	17.82	19.73
	MSE	1.32e−2	8.57e−2	7.43e−6	8.39e−2
	SSIM	7.73e−1	4.07e−1	1.00e+0	4.21e−1

Table 5: Numerical results with and without inference-time clipping.

H.4 BadDiffusion with Varying Poison Rates

The numerical results are shown in Tab. 6.

I More Generated Samples in Different Poison Rates

I.1 CIFAR10 Dataset

We show more generated backdoor targets and clean samples in Fig. 11

J 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)

Poison Rate	Trigger:	Grey Box					Stop Sign				
	Target:	NoShift	Shift	Corner	Shoe	Hat	NoShift	Shift	Corner	Shoe	Hat
0%	FID	14.83	14.83	14.83	14.83	14.83	14.83	14.83	14.83	14.83	14.83
	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	9.09	9.09	9.92	8.22	8.53	8.09	8.22	8.83	8.33	8.32
	MSE	6.19e−2	5.11e−2	5.32e−2	1.02e−1	1.58e−1	6.81e−2	5.68e−2	7.22e−2	1.66e−1	7.99e−2
	SSIM	4.21e−1	5.06e−1	4.20e−1	6.26e−1	3.12e−1	4.35e−1	5.73e−1	2.65e−1	4.20e−1	6.52e−1
10%	FID	9.62	9.78	10.98	8.41	8.81	7.62	7.42	7.83	7.48	7.57
	MSE	6.11e−3	5.52e−3	2.60e−3	6.25e−3	7.01e−3	9.47e−3	5.91e−3	4.20e−3	3.61e−3	4.33e−3
	SSIM	9.41e−1	9.45e−1	9.64e−1	9.75e−1	9.67e−1	9.18e−1	9.56e−1	9.49e−1	9.85e−1	9.80e−1
20%	FID	11 .36	11.26	12.86	8.13	8.89	7.97	7.68	8.35	8.10	8.17
	MSE	1.18e−5	7.90e−5	1.48e−4	1.97e−5	1.19e−5	2.35e−4	8.96e−5	7.09e−4	2.30e−5	4.85e−4
	SSIM	9.98e−1	9.98e−1	9.96e−1	1.00e+0	1.00e+0	9.97e−1	9.99e−1	9.89e−1	1.00e+0	9.98e−1
30%	FID	12.85	12.41	14.78	8.19	9.14	7.46	7.76	8.08	7.53	7.77
	MSE	5.89e−6	1.61e−5	2.29e−5	5.53e−6	5.68e−6	5.59e−6	6.73e−6	6.14e−5	5.62e−6	9.16e−5
	SSIM	9.98e−1	9.99e−1	9.98e−1	1.00e+0	1.00e+0	9.99e−1	9.99e−1	9.97e−1	1.00e+0	9.99e−1
50%	FID	17.63	15.55	20.10	8.42	10.25	7.68	8.02	8.14	7.69	7.77
	MSE	4.10e−6	6.25e−6	1.96e−5	3.26e−6	1.48e−5	4.19e−6	4.23e−6	2.37e−5	3.35e−6	1.30e−5
	SSIM	9.98e−1	9.99e−1	9.97e−1	1.00e+0	1.00e+0	9.98e−1	9.99e−1	9.98e−1	1.00e+0	1.00e+0
70%	FID	25.70	21.78	28.52	9.01	11.97	7.38	7.42	7.85	7.35	7.83
	MSE	3.91e−6	1.22e−5	6.44e−6	2.69e−6	8.27e−6	3.96e−6	3.96e−6	1.41e−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	52.92	41.54	55.42	12.25	19.09	7.22	7.72	7.98	7.54	7.77
	MSE	3.86e−6	5.98e−6	3.85e−6	2.38e−6	9.75e−6	3.80e−6	3.80e−6	3.86e−6	2.39e−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.

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.

into 18×18, 11×11, 8×8, and 4×4 pixels. The triggers are shown in Tab. 9. In Fig. 12 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.

K MORE REAL-WORLD THREATS

Here we provide more potential threats in the real world. (I) In Bauer & Bindschaedler (2021), generative models are used in security-related tasks such as Intrusion Attacks, Anomaly Detection, Biometric Spoofing, and Malware Obfuscation and Detection. (II) In recent works such as Janner et al. (2022); Wang et al. (2022); Pearce et al. (2023); Chen et al. (2022a); Baranchuk et al. (2022); Chen et al. (2022b), diffusion models are widely used for decision-making in reinforcement learning, object detection, and image segmentation, indicating potential threats to safety-critical tasks. (III) A backdoored generative model can generate a biased dataset which may cause unfair models Choi et al. (2020); Grover et al. (2019) and even datasets contain adversarial attacks Dolatabadi et al. (2020).

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.

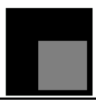
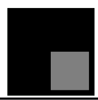
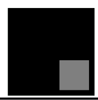
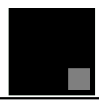
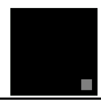
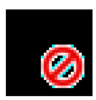
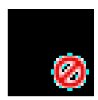
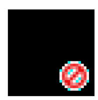
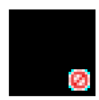
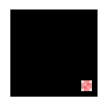
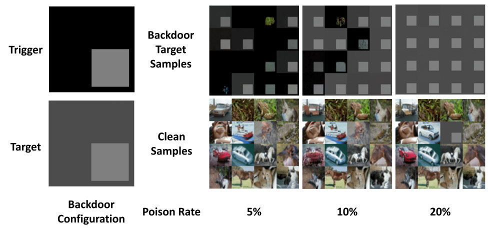
Dataset	CIFAR10 (32 × 32)				
Triggers	Grey Box				
Size Sample	18 × 18 	14 × 14 	11 × 11 	8 × 8 	4 × 4 
Triggers	Grey Box				
Size Sample	18 × 18 	14 × 14 	11 × 11 	8 × 8 	4 × 4 

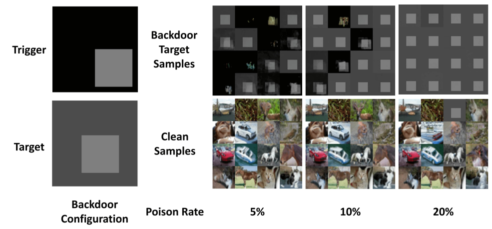
Table 9: Visualized samples for different trigger sizes

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	8.24	9.43	8.85	9.55	11.60	7.49	8.14	7.39	8.20	8.56
	MSE	7.87e−3	6.27e−2	3.13e−2	6.80e−2	5.45e−2	4.05e−2	6.91e−2	4.69e−2	5.97e−2	8.56e−2
	SSIM	9.39e−1	4.13e−1	6.87e−1	2.95e−1	4.11e−1	7.01e−1	4.28e−1	5.76e−1	4.33e−1	1.11e−1
Shift	FID	8.42	9.11	8.84	9.11	10.67	7.56	8.27	8.28	8.26	9.46
	MSE	9.93e−3	5.21e−2	6.52e−2	5.00e−2	7.02e−2	4.29e−2	5.77e−2	6.12e−2	2.23e−2	8.82e−2
	SSIM	9.15e−1	4.96e−1	3.69e−1	4.87e−1	2.44e−1	7.31e−1	5.66e−1	5.20e−1	7.95e−1	9.54e−2
Corner	FID	8.90	9.33	9.67	9.96	10.36	7.63	8.53	8.63	9.04	10.39
	MSE	1.04e−2	5.41e−2	6.11e−2	6.86e−2	7.22e−2	4.94e−2	7.28e−2	4.91e−2	1.92e−2	9.81e−2
	SSIM	8.86e−1	4.11e−1	3.80e−1	3.30e−1	3.15e−1	4.90e−1	2.60e−1	4.93e−1	7.98e−1	6.61e−2
Shoe	FID	7.88	8.28	7.46	7.59	7.53	7.48	8.32	8.14	8.36	8.39
	MSE	2.52e−2	1.04e−1	1.04e−1	2.08e−1	2.87e−1	1.39e−1	1.68e−1	1.12e−1	4.29e−2	2.05e−1
	SSIM	8.99e−1	6.16e−1	6.49e−1	3.54e−1	1.37e−1	4.68e−1	4.13e−1	6.17e−1	8.59e−1	3.74e−1
Hat	FID	8.13	8.51	8.01	7.81	7.90	7.50	8.31	8.22	8.58	8.67
	MSE	1.33e−2	1.60e−1	1.55e−1	1.62e−1	2.33e−1	9.81e−2	8.16e−2	1.54e−1	1.50e−1	1.66e−1
	SSIM	9.38e−1	3.06e−1	3.34e−1	3.11e−1	2.89e−2	5.38e−1	6.44e−1	3.35e−1	3.65e−1	2.93e−1

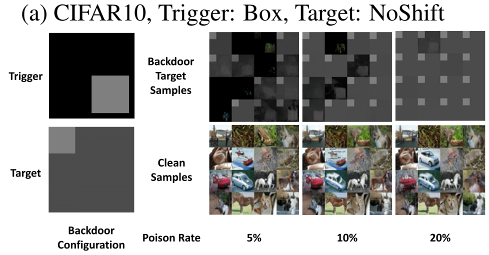
Table 10: The numerical results of BadDiffusion with varying trigger sizes.



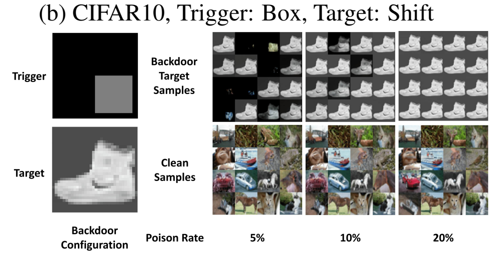
(a) CIFAR10, Trigger: Box, Target: NoShift



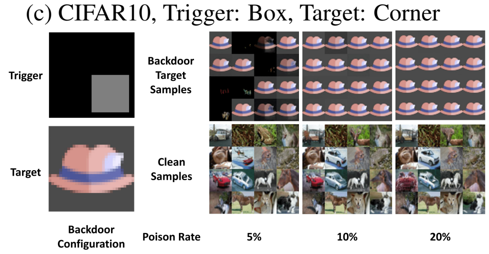
(b) CIFAR10, Trigger: Box, Target: Shift



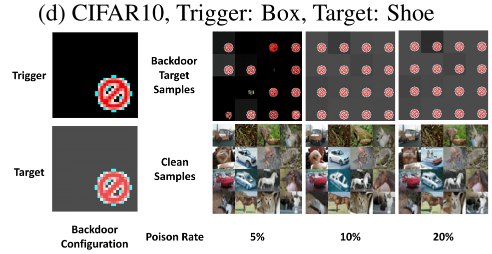
(c) CIFAR10, Trigger: Box, Target: Corner



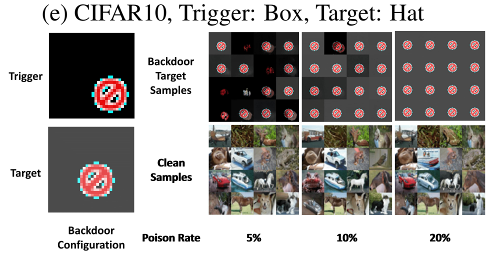
(d) CIFAR10, Trigger: Box, Target: Shoe



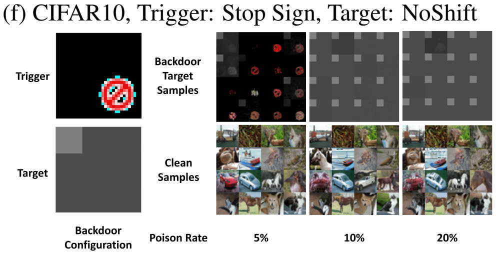
(e) CIFAR10, Trigger: Box, Target: Hat



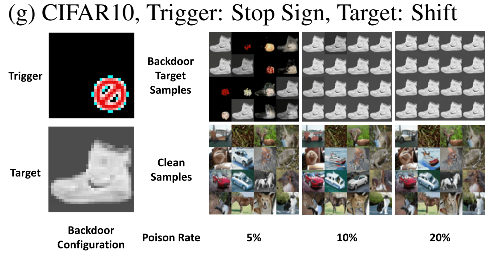
(f) CIFAR10, Trigger: Stop Sign, Target: NoShift



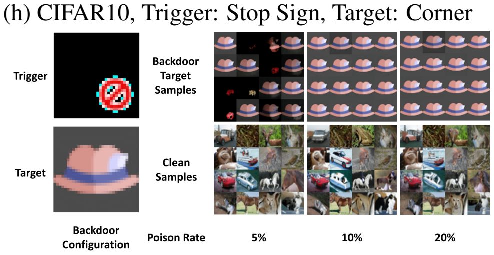
(g) CIFAR10, Trigger: Stop Sign, Target: Shift



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

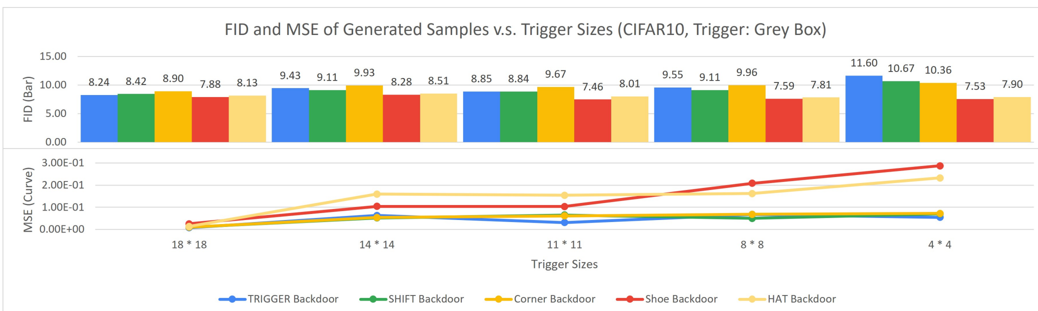


(i) CIFAR10, Trigger: Stop Sign, Target: Shoe

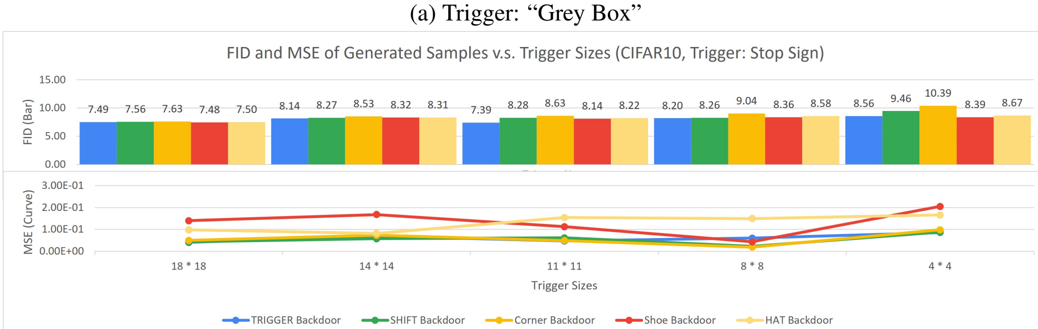


(j) CIFAR10, Trigger: Stop Sign, Target: Hat

Figure 11: Samples of CIFAR10



(a) Trigger: “Grey Box”



(b) Trigger: “Stop Sign”

Figure 12: 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

L The Mathematical Derivation of The Posterior of The Backdoored Diffusion Process

In this section, we'll 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 $C(\mathbf{x}'_t, \mathbf{x}'_0)$

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

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. We mark the correction term of BadDiffusion as red.

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

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

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

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 $\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon)$.

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

$$
\begin{aligned}
&= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon \right) \\
&\quad + \left(\frac{\beta_t(\sqrt{\alpha_t} - \cancel{\sqrt{\bar{\alpha}_t}}) - \alpha_t(1 - \sqrt{\alpha_t})(1 - \bar{\alpha}_{t-1}) - \beta_t(1 - \cancel{\sqrt{\bar{\alpha}_t}})}{(1 - \bar{\alpha}_t)\sqrt{\alpha_t}} \right) \mathbf{r}
\end{aligned}
\tag{24}
$$

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

$$
= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon \right) + \left(\frac{(\sqrt{\alpha_t} - 1)\cancel{(1 - \bar{\alpha}_t)}}{\cancel{(1 - \bar{\alpha}_t)}\sqrt{\alpha_t}} \right) \mathbf{r} \tag{26}
$$

$$
= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon \right) - \frac{1}{\sqrt{\alpha_t}}(1 - \sqrt{\alpha_t})\mathbf{r} \tag{27}
$$

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

$$
= \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}'_t(\mathbf{x}'_0, \mathbf{r}, \epsilon) - \rho_t \mathbf{r} - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon \right) \tag{28}
$$