Diffusion Models for Adversarial Purification

Weili Nie [1] **Brandon Guo** [2] **Yujia Huang** [2] **Chaowei Xiao** [1 3] **Arash Vahdat** [1] **Anima Anandkumar** [1 2]

Abstract

Adversarial purification refers to a class of defense methods that remove adversarial perturbations using a generative model. These methods do not make assumptions on the form of attack and the classification model, and thus can defend pre-existing classifiers against unseen threats. However, their performance currently falls behind adversarial training methods. In this work, we propose *DiffPure* that uses diffusion models for adversarial purification: Given an adversarial example, we first diffuse it with a small amount of noise following a forward diffusion process, and then recover the clean image through a reverse generative process. To evaluate our method against strong adaptive attacks in an efficient and scalable way, we propose to use the adjoint method to compute full gradients of the reverse generative process. Extensive experiments on three image datasets including CIFAR-10, ImageNet and CelebA-HQ with three classifier architectures including ResNet, WideResNet and ViT demonstrate that our method achieves the state-of-the-art results, outperforming current adversarial training and adversarial purification methods, often by a large margin. Project page: `https://diffpure.github.io`.

1. Introduction

Neural networks are vulnerable to adversarial attacks: adding imperceptible perturbations to the input can mislead trained neural networks to predict incorrect classes (Szegedy et al., 2014; Goodfellow et al., 2015). There have been many works on defending neural networks against such adversarial attacks (Madry et al., 2018; Song et al., 2018; Gowal et al., 2020). Among them, *adversarial training* (Madry et al., 2018), which trains neural networks on adversarial examples, has become a standard defense form, due to its effectiveness (Zhang et al., 2019; Gowal et al., 2021). However, most adversarial training methods can only defend against a specific attack that they are trained with. Recent works on defending against unseen threats add a carefully designed threat model into their adversarial training pipeline, but they suffer from a significant performance drop (Laidlaw et al., 2021; Dolatabadi et al., 2021). Additionally, the computational complexity of adversarial training is usually higher than standard training (Wong et al., 2020).

In contrast, another class of defense methods, often termed *adversarial purification* (Shi et al., 2021; Yoon et al., 2021), relies on generative models to purify adversarially perturbed images before classification (Samangouei et al., 2018; Hill et al., 2021). Compared to the adversarial training methods, adversarial purification can defend against unseen threats in a plug-n-play manner without re-training the classifiers. This is because the generative purification models are trained independently from both threat models and classifiers. Despite these advantages, their performance usually falls behind current adversarial training methods (Croce & Hein, 2020), in particular against adaptive attacks where the attacker has the full knowledge of the defense method (Athalye et al., 2018; Tramer et al., 2020). This is usually attributed to the shortcomings of current generative models that are used as a purification model, such as mode collapse in GANs (Goodfellow et al., 2014), low sample quality in energy-based models (EBMs) (LeCun et al., 2006), and the lack of proper randomness (Pinot et al., 2020).

Recently, diffusion models have emerged as powerful generative models (Ho et al., 2020; Song et al., 2021b). These models have demonstrated strong sample quality, beating GANs in image generation (Dhariwal & Nichol, 2021; Vahdat et al., 2021). They have also exhibited strong mode coverage, indicated by high test likelihood (Song et al., 2021a). Diffusion models consist of two processes: (i) a forward diffusion process that converts data to noise by gradually adding noise to the input, and (ii) a reverse generative process that starts from noise and generates data by denoising one step at a time. Intuitively in the generative process, diffusion models purify noisy samples, playing a similar role of a purification model. Their good generation quality and diversity ensure the purified images closely follow the original distribution of clean data. Moreover, the stochasticity in diffusion models can make a powerful stochastic defense (He

[1]NVIDIA [2]Caltech [3]ASU. Correspondence to: Weili Nie <wnie@nvidia.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

et al., 2019). These properties make diffusion models an ideal candidate for generative adversarial purification.

We summarize our main contributions as follows:

- We propose *DiffPure*, the first adversarial purification method that uses the forward and reverse processes of pre-trained diffusion models to purify adversarial images.
- We provide a theoretical analysis of the amount of noise added in the forward process such that it removes adversarial perturbations without destroying label semantics.
- We propose to use the adjoint method to efficiently compute full gradients of the reverse generative process in our method for evaluation against strong adaptive attacks.
- We perform extensive experiments to demonstrate that our method achieves the new start-of-the-art on various adaptive attack benchmarks.

In this work, we propose a new adversarial purification method, termed *DiffPure*, that uses the forward and reverse processes of diffusion models to purify adversarial images, as illutrated in Figure 1. Specifically, given a pre-trained diffusion model, our method consists of two steps: (i) we first add noise to adversarial examples by following the forward process with a small diffusion timestep, and (ii) we then solve the reverse stochastic differential equation (SDE) to recover clean images from the diffused adversarial examples. An important design parameter in our method is the choice of diffusion timestep, since it represents the amount of noise added during the forward process. Our theoretical analysis reveals that the noise needs to be high enough to remove adversarial perturbations but not too large as it will destroy the label semantics of purified images. Furthermore, strong adaptive attacks require gradient backpropagation through the SDE solver in our method, which suffers from the memory issue if implemented naively. Thus, we propose to use the *adjoint method* to efficiently calculate full gradients of the reverse SDE with a constant memory cost.

We empirically compare our method against the latest adversarial training and adversarial purification methods on various strong adaptive attack benchmarks. Extensive experiments on three datasets (*i.e.*, CIFAR-10, ImageNet and CelebA-HQ) across multiple classifier architectures (*i.e.*, ResNet, WideResNet and ViT) demonstrate the state-of-the-art performance of our method. For instance, compared to adversarial training methods against AutoAttack ℓ_∞ (Croce & Hein, 2020), our method shows absolute improvements of up to +5.44% on CIFAR-10 and up to +7.68% on ImageNet, respectively, in robust accuracy. Moreover, compared to the latest adversarial training methods against unseen threats, our method exhibits a more significant absolute improvement (up to +36% in robust accuracy). In comparison to adversarial purification methods against the BPDA+EOT attack (Hill et al., 2021), we have absolute improvements

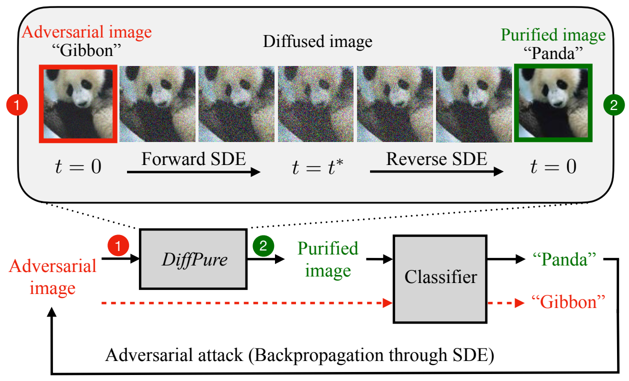


Figure 1. An illustration of *DiffPure*. Given a pre-trained diffusion model, we add noise to adversarial images following the forward diffusion process with a small diffusion timestep t^* to get diffused images, from which we recover clean images through the reverse denoising process before classification. Adaptive attacks backpropagate through the SDE to get full gradients of our defense system.

of +11.31% on CIFAR-10 and +15.63% on CelebA-HQ, respectively, in robust accuracy. Finally, our ablation studies show the optimal diffusion timestep remains small for most attacks, confirm the importance of proper noise injection in the forward and reverse processes for adversarial robustness, and also demonstrate the better performance by combining DiffPure with existing adversarial training methods.

2. Background

In this section, we briefly review continuous-time diffusion models (Song et al., 2021b).

Denote by $p(\boldsymbol{x})$ the unknown data distribution, from which each data point $\boldsymbol{x} \in \mathbb{R}^d$ is sampled. Diffusion models diffuse $p(\boldsymbol{x})$ towards a noise distribution. The forward diffusion process $\{\boldsymbol{x}(t)\}_{t\in[0,1]}$ is defined by an SDE with positive time increments in a fixed time horizon $[0,1]$:

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w}, \quad (1)$$

where the initial value $\boldsymbol{x}(0) := \boldsymbol{x} \sim p(\boldsymbol{x})$, $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is the drift coefficient, $g : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient, and $\boldsymbol{w}(t) \in \mathbb{R}^d$ is a standard Wiener process.

Denote by $p_t(\boldsymbol{x})$ the marginal distribution of $\boldsymbol{x}(t)$ with $p_0(\boldsymbol{x}) := p(\boldsymbol{x})$. In particular, $\boldsymbol{f}(\boldsymbol{x}, t)$ and $g(t)$ can be properly designed such that at the end the diffusion process, $\boldsymbol{x}(1)$ follows the standard Gaussian distribution, *i.e.*, $p_1(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$. Throughout the paper, we consider VP-SDE (Song et al., 2021b) as our diffusion model, where $\boldsymbol{f}(\boldsymbol{x}, t) := -\frac{1}{2}\beta(t)\boldsymbol{x}$ and $g(t) := \sqrt{\beta(t)}$, with $\beta(t)$ representing a time-dependent noise scale. By default, we use the linear noise schedule, *i.e.*, $\beta(t) := \beta_{\min} + (\beta_{\max} - \beta_{\min})t$.

Sample generation is done using the reverse-time SDE:

$$d\hat{\boldsymbol{x}} = [\boldsymbol{f}(\hat{\boldsymbol{x}}, t) - g(t)^2 \nabla_{\hat{\boldsymbol{x}}} \log p_t(\hat{\boldsymbol{x}})]dt + g(t)d\bar{\boldsymbol{w}} \quad (2)$$

where dt is an infinitesimal negative time step, and $\bar{\boldsymbol{w}}(t)$ is a standard reverse-time Wiener process. Sampling $\hat{\boldsymbol{x}}(1) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ as the initial value and solving the above SDE from $t{=}1$ to $t{=}0$ gradually produce the less-noisy data $\hat{\boldsymbol{x}}(t)$ until we draw samples from the data distribution, *i.e.*, $\hat{\boldsymbol{x}}(0) \sim p_0(\boldsymbol{x})$. Ideally, the resulting denoising process $\{\hat{\boldsymbol{x}}(t)\}_{t\in[0,1]}$ from Eq. (2) has the same distribution as the forward process $\{\boldsymbol{x}(t)\}_{t\in[0,1]}$ obtained from Eq. (1).

The reverse-time SDE in Eq. (2) requires the knowledge of the time-dependent score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$. One popular approach is to estimate $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})$ with a parameterized neural network $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ (Song et al., 2021b; Kingma et al., 2021). Accordingly, diffusion models are trained with the weighted combination of denoising score matching (DSM) across multiple time steps (Vincent, 2011):

$$\min_\theta \int_0^1 \mathbb{E}_{p(\boldsymbol{x})p_{0t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})} \left[\lambda(t) \|\nabla_{\tilde{\boldsymbol{x}}} \log p_{0t}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) - \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}, t)\|_2^2\right] dt$$

where $\lambda(t)$ is the weighting coefficient, and $p_{0t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ is the transition probability from $\boldsymbol{x}(0) := \boldsymbol{x}$ to $\boldsymbol{x}(t) := \tilde{\boldsymbol{x}}$ that has a closed form through the forward SDE in Eq. (1).

3. Method

We first propose *diffusion purification* (or *DiffPure* for short) that adds noise to adversarial images following the forward process of diffusion models to get diffused images, from which clean images are recovered through the reverse process. We also introduce some theoretical justifications of our method (Section 3.1). Next, we apply the *adjoint method* to backpropagate through SDE for efficient gradient evaluation with strong adaptive attacks (Section 3.2).

3.1. Diffusion Purification

Since the role of the forward SDE in Eq. (1) is to gradually remove the local structures of data by adding noise, we hypothesize that given an adversarial example $\boldsymbol{x}_a$, if we start the forward process with $\boldsymbol{x}(0) = \boldsymbol{x}_a$, the adversarial perturbations, a form of small local structures added to the data, will also be gradually smoothed.

The following theorem confirms that the clean data distribution $p(\boldsymbol{x})$ and the adversarially perturbed data distribution $q(\boldsymbol{x})$ get closer over the forward diffusion process, implying that the adversarial perturbations will indeed be "washed out" by the increasingly added noise.

Theorem 3.1. *Let $\{\boldsymbol{x}(t)\}_{t\in[0,1]}$ be the diffusion process defined by the forward SDE in Eq. (1). If we denote by p_t and q_t the respective distributions of $\boldsymbol{x}(t)$ when $\boldsymbol{x}(0) \sim p(\boldsymbol{x})$ (i.e., clean data distribution) and $\boldsymbol{x}(0) \sim q(\boldsymbol{x})$ (i.e., adversarial sample distribution), we then have*

$$\frac{\partial D_{KL}(p_t || q_t)}{\partial t} \leq 0$$

where the equality happens only when $p_t{=}q_t$. That is, the KL divergence of p_t and q_t monotonically decreases when moving from $t{=}0$ to $t{=}1$ through the forward SDE.

The proof follows (Song et al., 2021a; Lyu, 2009) that build connections between Fisher divergence and the "rate of change" in KL divergence by generalizing the de Bruijn's identity (Barron, 1986), which we defer to Appendix A.1. From the above theorem, there exists a minimum timestep $t^* \in [0, 1]$ such that $D_{KL}(p_{t^*} || q_{t^*}) \leq \varepsilon$. However, the diffused adversarial sample $\boldsymbol{x}(t^*) \sim q_{t^*}$ at timestep $t{=}t^*$ contains additional noise and cannot be directly classified. Hence, starting from $\boldsymbol{x}(t^*)$, we can *stochastically* recover the clean data at $t{=}0$ through the SDE in Eq. (2).

Diffusion purification: Inspired by the observation above, we propose a two-step adversarial purification method using diffusion models: Given an adversarial example $\boldsymbol{x}_a$ at timestep $t{=}0$, *i.e.*, $\boldsymbol{x}(0) = \boldsymbol{x}_a$, we first diffuse it by solving the forward SDE in Eq. (1) from $t{=}0$ to $t{=}t^*$. For VP-SDE, the diffused adversarial sample at the diffusion timestep $t^* \in [0, 1]$ can be sampled efficiently using:

$$\boldsymbol{x}(t^*) = \sqrt{\alpha(t^*)}\boldsymbol{x}_a + \sqrt{1 - \alpha(t^*)}\boldsymbol{\epsilon} \tag{3}$$

where $\alpha(t) = e^{-\int_0^t \beta(s)ds}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$.

Second, we solve the reverse-time SDE in Eq. (2) from the timestep $t{=}t^*$ using the diffused adversarial sample $\boldsymbol{x}(t^*)$, given by Eq. (3), as the initial value to get the final solution $\hat{\boldsymbol{x}}(0)$ of SDE in Eq. (2). As $\hat{\boldsymbol{x}}(0)$ does not have a closed-form solution, we resort to an SDE solver, termed `sdeint` (usually with the Euler–Maruyama discretization (Kloeden & Platen, 1992)). That is,

$$\hat{\boldsymbol{x}}(0) = \texttt{sdeint}(\boldsymbol{x}(t^*), \boldsymbol{f}_{\text{rev}}, g_{\text{rev}}, \bar{\boldsymbol{w}}, t^*, 0) \tag{4}$$

where `sdeint` is defined to sequentially take in six inputs: initial value, drift coefficient, diffusion coefficient, Wiener process, initial time, and end time. Also, the above drift and diffusion coefficients are given by

$$\begin{aligned} \boldsymbol{f}_{\text{rev}}(\boldsymbol{x}, t) &:= -\frac{1}{2}\beta(t)[\boldsymbol{x} + 2\boldsymbol{s}_\theta(\boldsymbol{x}, t)] \\ g_{\text{rev}}(t) &:= \sqrt{\beta(t)} \end{aligned} \tag{5}$$

The resulting purified data $\hat{\boldsymbol{x}}(0)$ is then passed to an external standard classifier to make predictions. An illustration of our method is shown in Figure 1.

Choosing the diffusion timestep t^*: From Theorem 3.1, t^* should be large enough to remove local adversarial perturbations. However, t^* cannot be arbitrarily large because the global label semantics will also be removed by the diffusion process if t^* keeps increasing. As a result, the purified sample $\hat{\boldsymbol{x}}(0)$ cannot be classified correctly.

Formally, the following theorem characterizes how the diffusion timestep t^* affects the difference between the clean image $\boldsymbol{x}_0$ and purified image obtained by our method $\hat{\boldsymbol{x}}(0)$.

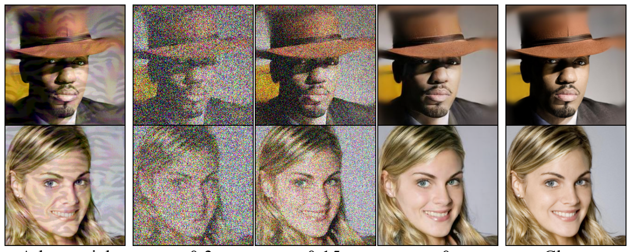
Theorem 3.2. *If we assume the score function satisfies that $\|\boldsymbol{s}_\theta(\boldsymbol{x}, t)\| \leq \frac{1}{2} C_s$, the L2 distance between the clean data $\boldsymbol{x}$ and the purified data $\hat{\boldsymbol{x}}(0)$ given by Eq. (4) satisfies that with a probability of at least $1 - \delta$, we have*

$$\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| \leq \|\boldsymbol{\epsilon}_a\| + \sqrt{e^{2\gamma(t^*)} - 1} C_\delta + \gamma(t^*) C_s$$

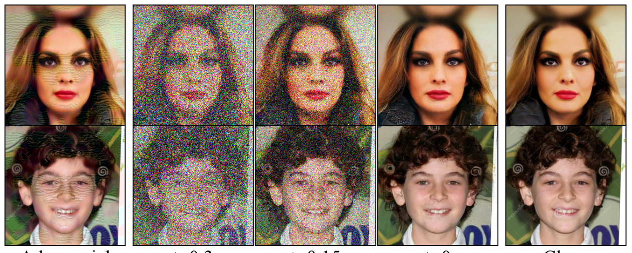
where $\boldsymbol{\epsilon}_a$ denotes the adversarial perturbation satisfying $\boldsymbol{x}_a = \boldsymbol{x} + \boldsymbol{\epsilon}_a$, $\gamma(t^*) := \int_0^{t^*} \frac{1}{2}\beta(s)ds$ and the constant $C_\delta := \sqrt{2d + 4\sqrt{d \log \frac{1}{\delta}} + 4 \log \frac{1}{\delta}}$.

See Appendix A.2 for the proof. Since $\gamma(t^*)$ monotonically increases with t^* and $\gamma(t^*) \geq 0$ for all t^*, the last two terms in the above upper bound both increase with t^*. Thus, to make $\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\|$ as low as possible, t^* needs to be sufficiently small. In the extreme case where t^*=0, we have the equality that $\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| = \|\boldsymbol{\epsilon}_a\|$, which means $\hat{\boldsymbol{x}}(0)$ reduces to $\boldsymbol{x}_a$ if we do not perform diffusion purification.

Due to the trade-off between purifying the local perturbations (with a larger t^*) and preserving the global structures (with a smaller t^*) of adversarial examples, there exists a sweet spot for the diffusion timestep t^* to obtain a high robust classification accuracy. Since adversarial perturbations are usually small, which can be removed with a small t^*, the best t^* in most adversarial robustness tasks also remain relatively small. As a proof of concept, we provide visual examples in Figure 2 to show how our method purifies the adversarial perturbations while maintaining the global semantic structures. See Appendix C.5 for more results.



(a) Smiling



(b) Eyeglasses

Figure 2. Our method purifies adversarial examples (first column) produced by attacking attribute classifiers using PGD ℓ_∞ ($\epsilon = 16/255$), where $t^* = 0.3$. The middle three columns show the results of the SDE in Eq. (4) at different timesteps, and we observe the purified images at t=0 match the clean images (last column). Better zoom in to see how we remove adversarial perturbations.

3.2. Adaptive Attack to Diffusion Purification

Strong adaptive attacks (Athalye et al., 2018; Tramer et al., 2020) require computing full gradients of our defense system. However, simply backpropagating through the SDE solver in Eq. (4) scales poorly in the computational memory. In particular, denote by N the number of function evaluations in solving the SDE, the required memory increases by $\mathcal{O}(N)$. This issue makes it challenging to effectively evaluate our method with strong adaptive attacks.

Prior adversarial purification methods (Shi et al., 2021; Yoon et al., 2021) suffer from the same memory issue with strong adaptive attacks. Thus, they either evaluate only with black-box attacks or change the evaluation strategy to circumvent the full gradient computation (*e.g.*, using approximate gradients). This makes them difficult to compare with adversarial training methods under the more standard evaluation protocols (*e.g.*, AutoAttack). To overcome this, we propose to use the *adjoint method* (Li et al., 2020) to efficiently compute full gradients of the SDE without the memory issue. The intuition is that the gradient through an SDE can be obtained by solving another augmented SDE.

The following proposition provides the augmented SDE for calculating the gradient of an objective $\mathcal{L}$ w.r.t. the input $x(t^*)$ of the SDE in Eq. (4).

Proposition 3.3. *For the SDE in Eq. (4), the augmented SDE that computes the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)}$ of backpropagating through it is given by*

$$\begin{pmatrix} \boldsymbol{x}(t^*) \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)} \end{pmatrix} = \texttt{sdeint}\left(\begin{pmatrix} \hat{\boldsymbol{x}}(0) \\ \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)} \end{pmatrix}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}, \tilde{\boldsymbol{w}}, 0, t^* \right) \tag{6}$$

where $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)}$ is the gradient of the objective $\mathcal{L}$ w.r.t. the output $\hat{\boldsymbol{x}}(0)$ of the SDE in Eq. (4), and

$$\tilde{\boldsymbol{f}}([\boldsymbol{x}; \boldsymbol{z}], t) = \begin{pmatrix} \boldsymbol{f}_{rev}(\boldsymbol{x}, t) \\ \frac{\partial \boldsymbol{f}_{rev}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} \boldsymbol{z} \end{pmatrix}$$

$$\tilde{\boldsymbol{g}}(t) = \begin{pmatrix} -g_{rev}(t)\mathbf{1}_d \\ \mathbf{0}_d \end{pmatrix}$$

$$\tilde{\boldsymbol{w}}(t) = \begin{pmatrix} -\boldsymbol{w}(1-t) \\ -\boldsymbol{w}(1-t) \end{pmatrix}$$

with $\mathbf{1}_d$ and $\mathbf{0}_d$ representing the d-dimensional vectors of all ones and all zeros, respectively.

The proof is deferred to Appendix A.3. Ideally if the SDE solver has a small numerical error, the gradient obtained from this proposition will closely match its true value (see Appendix B.5). As the gradient computation has been converted to solving the augmented SDE in Eq. (6), we do not need to store intermediate operations and thus end up with the $\mathcal{O}(1)$ memory cost (Li et al., 2020). That is, the adjoint method described above turns the reverse-time SDE in Eq. (4) into a differentiable operation (without the memory issue). Since the forward diffusion step in Eq. (3) is also differentiable using the reparameterization trick, we can easily compute full gradients of a loss function regarding the adversarial images for strong adaptive attacks.

4. Related Work

Adversarial training It learns a robust classifier by training on adversarial examples created during every weight update. After first introduced by Madry et al. (2018), adversarial training has become one of the most successful defense methods in neural networks against adversarial attacks (Gowal et al., 2020; Rebuffi et al., 2021). Despite the difference in the defense form, some variants of adversarial training share similarities with our method. He et al. (2019) inject Gaussian noise to each network layer for better robustness via stochastic effects. Kang et al. (2021) train neural ODEs with Lyapunov-stable equilibrium points for adversarial defense. Gowal et al. (2021) use generative models for data augmentation to improve adversarial training, where diffusion models work the best.

Adversarial purification Using generative models to purify adversarial images before classification, adversarial purification has become a promising counterpart of adversarial training. In particular, Samangouei et al. (2018) propose defense-GAN using GANs as the purification model, and Song et al. (2018) propose PixelDefense by relying on autoregressive generative models. More recently, Du & Mordatch (2019); Grathwohl et al. (2020); Hill et al. (2021) show the improved robustness of using EBMs to purify attacked images via Langevin dynamics (LD). More similar to our work, Yoon et al. (2021) use the denoising score-based model (Song & Ermon, 2019) for purification, but its sampling is still a variant of LD that does not rely on forward diffusion and backward denoising processes. We empirically compare our method against these previous works and we largely outperform them.

Image editing with diffusion models As a probabilistic generative models for unsupervised modeling (Ho et al., 2020), diffusion models have shown strong sample quality and diversity in image synthesis (Dhariwal & Nichol, 2021; Song et al., 2021a). Since then, they have been used in many image editing tasks, such as image-to-image translation (Meng et al., 2021; Choi et al., 2021; Saharia et al., 2021) and text-guided image editing (Kim & Ye, 2021; Nichol et al., 2021; Ramesh et al., 2022). Although adversarial purification can be considered as a special image editing task and particularly DiffPure shares a similar procedure with SDEdit (Meng et al., 2021), none of these works apply diffusion models to improve the model robustness. Besides, evaluating our method with strong adaptive attacks poses a new challenge of backpropagating through the denoising process that previous works do not deal with.

5. Experiments

In this section, we first provide experimental settings (Section 5.1). On various strong adaptive attack benchmarks, we then compare our method with the state-of-the-art adversarial training and adversarial purification methods (Section 5.2 to 5.4). Note that we defer the results of our method against the standard attack (*i.e.*, non-adaptive) and the black-box attack, suggested by Croce et al. (2022), to Appendix C.1 for completeness. Finally, we perform various ablation studies to provide better insights into our method (Section 5.5).

5.1. Experimental Settings

Datasets and network architectures We consider three datasets for evaluation: CIFAR-10 (Krizhevsky, 2009), CelebA-HQ (Karras et al., 2018), and ImageNet (Deng et al., 2009). Particularly, we compare with the state-of-the-art defense methods reported by the standardized benchmark RobustBench (Croce et al., 2020) on CIFAR-10 and ImageNet while comparing with other adversarial purification methods on CIFAR-10 and CelebA-HQ following their settings. For classifiers, we consider three widely used architectures: ResNet (He et al., 2016), WideResNet (Zagoruyko & Komodakis, 2016) and ViT (Dosovitskiy et al., 2021).

Adversarial attacks We evaluate our method with strong adaptive attacks. We use the commonly used AutoAttack ℓ_∞ and ℓ_2 threat models (Croce & Hein, 2020) to compare with adversarial training methods. To show the broader applicability of our method beyond ℓ_p-norm attacks, we also evaluate with the spatially transformed adversarial examples (StAdv) (Xiao et al., 2018). Due to the stochasticity introduced by the diffusion and denoising processes (Section 3.1), we apply Expectation Over Time (EOT) (Athalye et al., 2018) to these adaptive attacks, where we use EOT=20 (see Figure 6 for more details). Besides, we apply the BPDA+EOT attack (Hill et al., 2021) to make a fair comparison with other adversarial purification methods.

Evaluation metrics We consider two metrics to evaluate the performance of defense approaches: *standard accuracy* and *robust accuracy*. The standard accuracy measures the performance of the defense method on clean data, which is evaluated on the whole test set in each dataset. The robust accuracy measures the performance on adversarial examples generated by adaptive attacks. Due to the high computational cost of applying adaptive attacks to our method, unless stated otherwise, we evaluate robust accuracy for our method and previous works on a fixed subset of 512 images randomly sampled from the test set. Notably, robust accura-

Table 1. Standard accuracy and robust accuracy against AutoAttack ℓ_∞ ($\epsilon = 8/255$) on CIFAR-10, obtained by different classifier architectures. In our method, the diffusion timestep is $t^* = 0.1$.

Method	Extra Data	Standard Acc	Robust Acc
WideResNet-28-10			
(Zhang et al., 2020)	✓	89.36	59.96
(Wu et al., 2020)	✓	88.25	62.11
(Gowal et al., 2020)	✓	89.48	62.70
(Wu et al., 2020)	✗	85.36	59.18
(Rebuffi et al., 2021)	✗	87.33	61.72
(Gowal et al., 2021)	✗	87.50	65.24
Ours	✗	**89.02±0.21**	**70.64±0.39**
WideResNet-70-16			
(Gowal et al., 2020)	✓	91.10	66.02
(Rebuffi et al., 2021)	✓	92.23	68.56
(Gowal et al., 2020)	✗	85.29	59.57
(Rebuffi et al., 2021)	✗	88.54	64.46
(Gowal et al., 2021)	✗	88.74	66.60
Ours	✗	**90.07±0.97**	**71.29±0.55**

Table 2. Standard accuracy and robust accuracy against AutoAttack ℓ_2 ($\epsilon = 0.5$) on CIFAR-10, obtained by different classifier architectures. In our method, the diffusion timestep is $t^* = 0.075$. (*Methods use WideResNet-34-10, with the same width but more layers than the default one.)

Method	Extra Data	Standard Acc	Robust Acc
WideResNet-28-10			
(Augustin et al., 2020)*	✓	92.23	77.93
(Rony et al., 2019)	✗	89.05	66.41
(Ding et al., 2020)	✗	88.02	67.77
(Wu et al., 2020)*	✗	88.51	72.85
(Sehwag et al., 2021)*	✗	90.31	75.39
(Rebuffi et al., 2021)	✗	**91.79**	78.32
Ours	✗	91.03±0.35	**78.58±0.40**
WideResNet-70-16			
(Gowal et al., 2020)	✓	94.74	79.88
(Rebuffi et al., 2021)	✓	95.74	81.44
(Gowal et al., 2020)	✗	90.90	74.03
(Rebuffi et al., 2021)	✗	92.41	**80.86**
Ours	✗	**92.68±0.56**	80.60±0.57

cies of most baselines do not change much on the sampled subset, compared to the whole test set (see Appendix C.2).

We defer more details of the above experimental settings and the baselines that we compare with to Appendix B.

5.2. Comparison with the State-of-the-art

We first compare DiffPure with the state-of-the-art adversarial training methods reported by RobustBench (Croce et al., 2020), against the ℓ_∞ and ℓ_2 threat models, respectively.

CIFAR-10 Table 1 shows the robustness performance against ℓ_∞ threat model ($\epsilon = 8/255$) with AutoAttack on CIFAR-10. We can see that our method achieves both better standard accuracy and better robust accuracy than previous state-of-the-art methods that do not use extra data on different classifier architectures. Specifically, our method improves robust accuracy by 5.44% on WideResNet-28-10 and by 4.69% on WideResNet-70-16, respectively. Furthermore, our method even largely outperforms baselines trained *with extra data* regarding robust accuracies, with comparable standard accuracies with different classifiers.

Table 3. Standard accuracy and robust accuracy against AutoAttack ℓ_∞ ($\epsilon = 4/255$) on ImageNet, obtained by different classifier architectures. In our method, the diffusion timestep is $t^* = 0.15$. (†Robust accuracy is directly reported from the respective paper.)

Method	Extra Data	Standard Acc	Robust Acc
ResNet-50			
(Engstrom et al., 2019)	✗	62.56	31.06
(Wong et al., 2020)	✗	55.62	26.95
(Salman et al., 2020)	✗	64.02	37.89
(Bai et al., 2021)†	✗	67.38	35.51
Ours	✗	**67.79±0.43**	**40.93±1.96**
WideResNet-50-2			
(Salman et al., 2020)	✗	68.46	39.25
Ours	✗	**71.16±0.75**	**44.39±0.95**
DeiT-S			
(Bai et al., 2021)†	✗	66.50	35.50
Ours	✗	**73.63±0.62**	**43.18±1.27**

Table 2 shows the robustness performance against ℓ_2 threat model ($\epsilon = 0.5$) with AutoAttack on CIFAR-10. We can see that our method outperforms most defense methods without using extra data while being on par with the best performing method (Rebuffi et al., 2021), regarding both standard and robust accuracies. The gap between our method and (Rebuffi et al., 2021) trained with extra data exists, but can be leveled up by replacing the standard classifier in our method with the adversarially trained one, as shown in Figure 4.

These results demonstrate the effectiveness of our method in defending against ℓ_∞ and ℓ_2 threat models on CIFAR-10. It is worth noting that in contrast to the competing methods that are trained for the specific ℓ_p-norm attack used in evaluation, our method is agnostic to the threat model.

ImageNet Table 3 shows the robustness performance against ℓ_∞ threat model ($\epsilon = 4/255$) with AutoAttack on ImageNet. We evaluate our method on two CNN architectures: ResNet-50 and WideResNet-50-2, and one ViT architecture: DeiT-S (Touvron et al., 2021). We can see that our method largely outperforms the state-of-the-art baselines regarding both the standard and robust accuracies. Besides, the advantages of our method over baselines become more significant on the ViT architecture. Specifically, our method improves robust accuracy by 3.04% and 5.14% on ResNet-50 and WideResNet-50-2, respectively, and by 7.68% on DeiT-S. For standard accuracy on DeiT-S, our method also largely improves over the baseline by 7.13%.

Table 4. Standard accuracy and robust accuracies against unseen threat models on ResNet-50 for CIFAR-10. We keep the same evaluation settings with (Laidlaw et al., 2021), where the attack bounds are $\epsilon = 8/255$ for AutoAttack ℓ_∞, $\epsilon = 1$ for AutoAttack ℓ_2, and $\epsilon = 0.05$ for StAdv. The baseline results are reported from the respective papers. For our method, the diffusion timestep is $t^* = 0.125$.

Method	Standard Acc	Robust Acc		
		ℓ_∞	ℓ_2	StAdv
Adv. Training with ℓ_∞ (Laidlaw et al., 2021)	86.8	49.0	19.2	4.8
Adv. Training with ℓ_2 (Laidlaw et al., 2021)	85.0	39.5	47.8	7.8
Adv. Training with StAdv (Laidlaw et al., 2021)	86.2	0.1	0.2	53.9
PAT-self (Laidlaw et al., 2021)	82.4	30.2	34.9	46.4
ADV. CRAIG (Dolatabadi et al., 2021)	83.2	40.0	33.9	49.6
ADV. GRADMATCH (Dolatabadi et al., 2021)	83.1	39.2	34.1	48.9
Ours	**88.2±0.8**	**70.0±1.2**	**70.9±0.6**	**55.0±0.7**

Table 5. Comparison with other adversarial purification methods using the BPDA+EOT attack with ℓ_∞ perturbations. (a) We evaluate on the *eyeglasses* attribute classifier for CelebA-HQ, where $\epsilon = 16/255$. See Appendix C.3 for similar results on the *smiling* attribute. Note that OPT and ENC denote the optimization-based and econder-based GAN inversions, respectively, and ENC+OPT implies a combination of OPT and ENC. (b) We evaluate on WideResNet-28-10 for CIFAR-10, and keep the experimental settings the same with (Hill et al., 2021), where $\epsilon = 8/255$. (*The purification is actually a variant of the LD sampling.)

(a) CelebA-HQ

Method	Purification	Standard Acc	Robust Acc
(Vahdat & Kautz, 2020)	VAE	**99.43**	0.00
(Karras et al., 2020)	GAN+OPT	97.76	10.80
(Chai et al., 2021)	GAN+ENC+OPT	99.37	26.37
(Richardson et al., 2021)	GAN+ENC	93.95	75.00
Ours ($t^* = 0.4$)	Diffusion	93.87±0.18	89.47±1.18
Ours ($t^* = 0.5$)	Diffusion	93.77±0.30	**90.63±1.10**

(b) CIFAR-10

Method	Purification	Standard Acc	Robust Acc
(Song et al., 2018)	Gibbs Update	**95.00**	9.00
(Yang et al., 2019)	Mask+Recon.	94.00	15.00
(Hill et al., 2021)	EBM+LD	84.12	54.90
(Yoon et al., 2021)	DSM+LD*	86.14	70.01
Ours ($t^* = 0.075$)	Diffusion	91.03±0.35	77.43±0.19
Ours ($t^* = 0.1$)	Diffusion	89.02±0.21	**81.40±0.16**

These results clearly demonstrate the effectiveness of our method in defending against ℓ_∞ threat models on ImageNet. Note that for the adversarial training baselines, the training recipes for CNNs cannot be directly applied to ViTs due to the over-regularization issue (Bai et al., 2021). However, our method is agnostic to classifier architectures.

5.3. Defense Against Unseen Threats

The main drawback of the adversarial training baselines is their poor generalization to unseen attacks: even if models are robust against a specific threat model, they are still fragile against other threat models. To see this, we evaluate each method with three attacks: ℓ_∞, ℓ_2 and StAdv, shown in Table 4. Note that for the plain adversarial training methods with a specific attack objective (*e.g.*, Adv Train - ℓ_∞), only other threat models (*e.g.*, ℓ_2 and StAdv) are considered unseen. We thus mark the seen threats by gray.

We can see that our method is robust to all three unseen threat models while the performances of these plain adversarial baselines drop significantly against unseen attacks. Compared with the state-of-the-art defense methods against unseen threat models (Laidlaw et al., 2021; Dolatabadi et al., 2021), our method achieves significantly better standard accuracy and robust accuracies across all three attacks. For instance, the robust accuracy of our method improves over the previously best performance by +30%, +36% and +5.4% on ℓ_∞, ℓ_2 and StAdv, respectively.

5.4. Comparison with Other Purification Methods

Because most prior adversarial purification methods have an optimization or sampling loop in their defense process (Hill et al., 2021), they cannot be evaluated directly with the strongest white-box adaptive attacks, such as AutoAttack. To this end, we use the BPDA+EOT attack (Tramer et al., 2020; Hill et al., 2021), an adaptive attack designed specifically for purification methods (with stochasticity), to evaluate our method and baselines for a fair comparison.

CelebA-HQ We compare with other strong generative models, such as NVAE (Vahdat & Kautz, 2020) and StyleGAN2 (Karras et al., 2020), that can be used to purify adversarial examples. The basic idea is to first encode adversarial images to latent codes, with which purified images are synthesized from the decoder (see Appendix B.3 for implementation details). We choose CelebA-HQ for the comparsion because they both perform well on it. In Table 5a, we use the *eyeglasses* attribute to show that our method has much better robust accuracy (+15.63%) than the best performing baseline while also maintaining a relatively high standard accuracy. We defer the similar results on the *smiling* attribute to Appendix C.3. These results demonstrate the superior performance of diffusion models in adversarial robustness than other generative models as a purification model.

CIFAR-10 In Table 5b, we compare our method with other adversarial purification methods on CIFAR-10, where

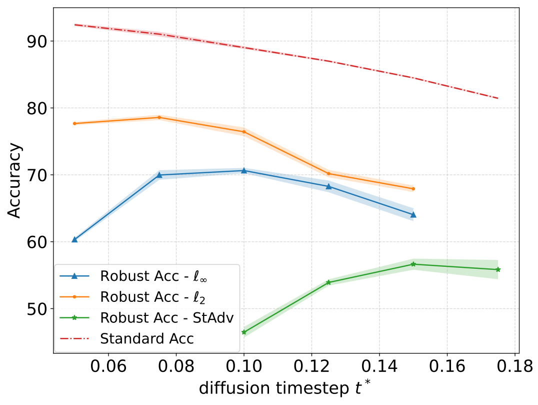


Figure 3. Impact of diffusion time t^* in our method on standard accuracy and robust accuracies against AutoAttack ℓ_∞ ($\epsilon = 8/255$), ℓ_2 ($\epsilon = 0.5$) and StAdv ($\epsilon = 0.05$) threat models, respectively, where we evaluate on WideResNet-28-10 for CIFAR-10.

the methods based on the LD sampling for purification are the state-of-the-art (Hill et al., 2021; Yoon et al., 2021). We observe that our method largely outperforms previous methods against the BPDA+EOT attack, with an absolute improvement of at least +11.31% in robust accuracy. Meanwhile, we can slightly trade-off robust accuracy for better standard accuracy by decreasing t^*, making it comparable to the best reported standard accuracy (*i.e.*, 91.03% vs. 95.00%). These results show that our method becomes a new state-of-the-art in adversarial purification.

5.5. Ablation Studies

Impact of diffusion timestep t^* We first show how the diffusion timestep t^* affects the robustness performance of our method against different threat models in Figure 3. We can see that (i) the standard accuracy monotonically decreases with t^* since more label semantics are lost with the larger diffusion timestep, and (ii) all the robust accuracies first increase and then decrease as t^* becomes larger due to the trade-off as discussed in Section 3.1. Notably, the optimal timestep t^* for the best robust accuracy remains small but also varies across different threat models (*e.g.*, ℓ_∞: t^*=0.075, ℓ_2: t^*=0.1, and StAdv: t^*=0.15). Since stronger perturbations need a larger diffusion timestep to be smoothed, it implies that StAdv (ϵ=0.05) perturbs the input images the most while ℓ_2 (ϵ=0.5) does the least.

Impact of sampling strategy Given the pre-trained diffusion models, besides relying on VP-SDE, there are other ways of recovering clean images from the adversarial examples. Here we consider another two sampling strategies: (i) LD-SDE (*i.e.*, an SDE formulation of the LD sampling that samples from an EBM, formed by our score function at t=0, instead of introducing forward and reverse diffusion processes), and (ii) VP-ODE (*i.e.*, an equivalent ODE sampling derived from VP-SDE that solves the reverse generative process using the probability flow ODEs (Song et al., 2021b) while keeping the forward diffusion unchanged). Please see Appendix B.4 for more details about these sampling variants. In Table 6, we compare different sampling strategies with the same diffusion model.

Table 6. We compare different sampling strategies by evaluating on WideResNet-28-10 with AutoAttack ℓ_∞ ($\epsilon = 8/255$) for CIFAR-10. We use $t^* = 0.1$ for both VP-ODE and VP-SDE (Ours), while using the best hyperparameters after grid search for LD-SDE.

Sampling	Standard Acc	Robust Acc
LD-SDE	87.36±0.09	38.54±1.55
VP-ODE	**90.79±0.12**	39.86±0.98
VP-SDE (Ours)	89.02±0.21	**70.64±0.39**

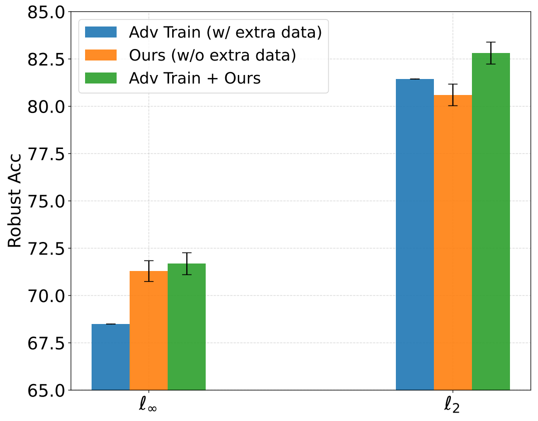


Figure 4. Combination of our method with adversarial training, where we evaluate on WideResNet-76-10 for CIFAR-10 with AutoAttack ℓ_∞ ($\epsilon = 8/255$) and ℓ_2 ($\epsilon = 0.5$) threat models, respectively. Regarding adversarial training, we use the model in (Rebuffi et al., 2021) that is adversarially trained with extra data.

Although each sampling strategy has a comparable standard accuracy, our method achieves a significantly better robust accuracy. To explain this, we hypothesize that (i) the LD sampling only uses the score function with clean images at timestep t=0, making it less robust to noisy (or perturbed) input images, while our method considers score functions at various noise levels. (ii) The ODE sampling introduces much less randomness to the defense model, due to its deterministic trajectories, and thus is more vulnerable to adaptive attacks from the randomized smoothing perspective (Cohen et al., 2019; Pinot et al., 2020). Inspired by this observation, we study adding more stochasticity using a randomized diffusion timestep t^* for the improved performance in Appendix C.4, where we find that more randomness in the purification method does not always leads to better robustness, implying the significance of introducing *proper* randomness in a more principled way from the forward and reverse processes.

Combination with adversarial training Since our proposed *DiffPure* is an orthogonal defense method to adversarial training, we can also combine our method with adversarial training (*i.e.*, feeding the purified images from our method to the adversarially trained classifiers). Figure 4 shows that this combination (*i.e.*, "Adv Train + Ours") can improve the robust accuracies against AutoAttack ℓ_∞ and ℓ_2 threat models, respectively. Besides, by comparing the results against the ℓ_∞ and ℓ_2 threat models, the improvement from the combination over our method with the standard classifier (*i.e.*, "Ours") becomes more significant when the adversarial training method with extra data (*i.e.*, "Adv Train") is already on par with our method. Therefore, we can apply our method to the pre-existing adversarially trained classifiers for further improving the performance.

6. Conclusions

We proposed a new defense method called *DiffPure* that applies diffusion models to purify adversarial examples before feeding them into classifiers. We also applied the adjoint method to compute full gradients of the SDE solver for evaluating with strong white-box adaptive attacks. To show the robustness performance of our method, we conducted extensive experiments on CIFAR-10, ImageNet and CelebA-HQ with different classifiers architectures including ResNet, WideResNet and ViT to compare with the state-of-the-art adversarial training and adversarial purification methods. In defense of various strong adaptive attacks such as AutoAttack, StAdv and BPDA+EOT, our method largely outperforms previous approaches.

Despite the large improvements, our method has two major limitations: (i) the purification process takes much time (proportional to the diffusion timestep, see Appendix C.6), making our method inapplicable to the real-time tasks, and (ii) diffusion models are sensitive to image colors, making our method incapable of defending color-related corruptions. It is interesting to either apply recent works on accelerating diffusion models or design new diffusion models specifically for model robustness to overcome these two limitations.

Acknowledgement

We would like to thank the AIALGO team at NVIDIA and Anima Anandkumar's research group at Caltech for reading the paper and providing fruitful suggestions. We also thank the anonymous reviewers for helpful comments.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Augustin, M., Meinke, A., and Hein, M. Adversarial robustness on in-and out-distribution improves explainability. In *European Conference on Computer Vision*, pp. 228–245. Springer, 2020.

Bai, Y., Mei, J., Yuille, A., and Xie, C. Are transformers more robust than cnns? In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Barron, A. R. Entropy and the central limit theorem. *The Annals of probability*, pp. 336–342, 1986.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

Chai, L., Zhu, J.-Y., Shechtman, E., Isola, P., and Zhang, R. Ensembling with deep generative views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Choi, J., Kim, S., Jeong, Y., Gwon, Y., and Yoon, S. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14367–14376, 2021.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, T. Evaluating the adversarial robustness of adaptive test-time defenses. *arXiv preprint arXiv:2202.13711*, 2022.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020.

Dolatabadi, H. M., Erfani, S., and Leckie, C. ℓ_∞-robustness and beyond: Unleashing efficient adversarial training. *arXiv preprint arXiv:2112.00378*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 2019.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 2014.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34, 2021.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

He, Z., Rakin, A. S., and Fan, D. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 588–597, 2019.

Hill, M., Mitchell, J. C., and Zhu, S.-C. Stochastic security: Adversarial defense using long-run dynamics of energy-based models. In *International Conference on Learning Representations*, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Neural Information Processing Systems (NeurIPS)*, 2021.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Kim, G. and Ye, J. C. Diffusionclip: Text-guided image manipulation using diffusion models. *arXiv preprint arXiv:2110.02711*, 2021.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in Neural Information Processing Systems*, 2021.

Kloeden, P. E. and Platen, E. Stochastic differential equations. In *Numerical Solution of Stochastic Differential Equations*, pp. 103–160. Springer, 1992.

Krizhevsky, A. Learning multiple layers of features from tiny images. *(Technical Report) University of Toronto.*, 2009.

Laidlaw, C., Singla, S., and Feizi, S. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Lyu, S. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 359–366, 2009.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. Sdedit: Image synthesis and editing with stochastic differential equations, 2021.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Pinot, R., Ettedgui, R., Rizk, G., Chevaleyre, Y., and Atif, J. Randomization matters how to defend against strong adversarial attacks. In *International Conference on Machine Learning*, 2020.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., and Cohen-Or, D. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ayed, I. B., Sabourin, R., and Granger, E. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4322–4330, 2019.

Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M. Palette: Image-to-image diffusion models. *arXiv preprint arXiv:2111.05826*, 2021.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? In *Advances in Neural Information Processing Systems*, 2020.

Samangouei, P., Kabkab, M., and Chellappa, R. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., and Mittal, P. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? *arXiv preprint arXiv:2104.09425*, 2021.

Shi, C., Holtz, C., and Mishne, G. Online adversarial purification based on self-supervised learning. In *International Conference on Learning Representations*, 2021.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 2021a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jegou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, 2021.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33, 2020.

Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. *arXiv preprint arXiv:2004.05884*, 2020.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018.

Yang, Y., Zhang, G., Katabi, D., and Xu, Z. Me-net: Towards effective adversarial robustness with matrix estimation. In *International Conference on Machine Learning*, 2019.

Yoon, J., Hwang, S. J., and Lee, J. Adversarial purification with score-based generative models. In *International Conference on Machine Learning*, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*, 2016.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. PMLR, 2019.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2020.

A. Proofs in Section 3

A.1. Proof of Theorem 3.1

Theorem A.1. *Let $\{\boldsymbol{x}(t)\}_{t\in[0,1]}$ be the diffusion process defined by the forward SDE in Eq. (1). If we denote by p_t and q_t the respective distributions of $\boldsymbol{x}(t)$ when $\boldsymbol{x}(0) \sim p(\boldsymbol{x})$ (i.e., clean data distribution) and $\boldsymbol{x}(0) \sim q(\boldsymbol{x})$ (i.e., adversarial sample distribution), we then have*

$$\frac{\partial D_{KL}(p_t||q_t)}{\partial t} \leq 0 \tag{7}$$

where the equality happens only when p_t=q_t. That is, the KL divergence of p_t and q_t monotonically decreases when moving from t=0 to t=1 through the forward SDE.

Proof: The proof follows (Song et al., 2021a). First, the Fokker-Planck equation (Särkkä & Solin, 2019) for the forward SDE in Eq. (1) is given by

$$\begin{aligned}\frac{\partial p_t(\boldsymbol{x})}{\partial t} &= -\nabla_{\boldsymbol{x}} \cdot \left(f(\boldsymbol{x},t)p_t(\boldsymbol{x}) - \frac{1}{2}g^2(t)\nabla_{\boldsymbol{x}} p_t(\boldsymbol{x})\right) \\ &= -\nabla_{\boldsymbol{x}} \cdot \left(f(\boldsymbol{x},t)p_t(\boldsymbol{x}) - \frac{1}{2}g^2(t)p_t(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right) \\ &= \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{h}_p(\boldsymbol{x},t)p_t(\boldsymbol{x}))\end{aligned} \tag{8}$$

where we define $\boldsymbol{h}_p(\boldsymbol{x},t) := \frac{1}{2}g^2(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - f(\boldsymbol{x},t)$. Then if we assume $p_t(x)$ and $q_t(x)$ are smooth and fast decaying, *i.e.*,

$$\lim_{\boldsymbol{x}_i\to\infty} p_t(\boldsymbol{x})\frac{\partial}{\partial \boldsymbol{x}_i}\log p_t(\boldsymbol{x}) = 0 \quad \text{and} \quad \lim_{\boldsymbol{x}_i\to\infty} q_t(\boldsymbol{x})\frac{\partial}{\partial \boldsymbol{x}_i}\log q_t(\boldsymbol{x}) = 0, \tag{9}$$

for any $i = 1, \cdots, d$, we can evaluate

$$\begin{aligned}\frac{\partial D_{KL}(p_t||q_t)}{\partial t} &= \frac{\partial}{\partial t}\int p_t(\boldsymbol{x}) \log \frac{p_t(\boldsymbol{x})}{q_t(\boldsymbol{x})}d\boldsymbol{x} \\ &= \int \frac{\partial p_t(\boldsymbol{x})}{\partial t} \log \frac{p_t(\boldsymbol{x})}{q_t(\boldsymbol{x})}d\boldsymbol{x} + \underbrace{\int \frac{\partial p_t(\boldsymbol{x})}{\partial t}d\boldsymbol{x}}_{=0} + \int \frac{p_t(\boldsymbol{x})}{q_t(\boldsymbol{x})}\frac{\partial q_t(\boldsymbol{x})}{\partial t}d\boldsymbol{x} \\ &\overset{(a)}{=} \int \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{h}_p(\boldsymbol{x},t)p_t(\boldsymbol{x})) \log \frac{p_t(\boldsymbol{x})}{q_t(\boldsymbol{x})}d\boldsymbol{x} + \int \frac{p_t(\boldsymbol{x})}{q_t(\boldsymbol{x})}\nabla_{\boldsymbol{x}} \cdot (\boldsymbol{h}_q(\boldsymbol{x},t)q_t(\boldsymbol{x}))\, d\boldsymbol{x} \\ &\overset{(b)}{=} -\int p_t(\boldsymbol{x})[\boldsymbol{h}_p(\boldsymbol{x},t) - \boldsymbol{h}_q(\boldsymbol{x},t)]^T[\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})]d\boldsymbol{x} \\ &= -\frac{1}{2}g^2(t)\int p_t(\boldsymbol{x})\|\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})\|_2^2 d\boldsymbol{x} \\ &\overset{(c)}{=} -\frac{1}{2}g^2(t)D_F(p_t||q_t)\end{aligned}$$

where (a) follows by plugging Eq. (8), (b) follows from the integration by parts and the assumption in Eq. (9), and (c) follows from the definition of the *Fisher divergence*: $D_F(p_t||q_t) := \int p_t(\boldsymbol{x})\|\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x})\|_2^2 d\boldsymbol{x}$.

Since $g^2(t) > 0$, and the Fisher divergence satisfies that $D_F(p_t||q_t) \geq 0$ and $D_F(p_t||q_t) = 0$ if and only if $p_t = q_t$, we have

$$\frac{\partial D_{KL}(p_t||q_t)}{\partial t} \leq 0$$

where the equality happens only when p_t=q_t. □

A.2. Proof of Theorem 3.2

Theorem A.2. *If we assume the score function satisfies that $\|\boldsymbol{s}_\theta(\boldsymbol{x},t)\| \leq \frac{1}{2}C_s$, the L2 distance between the clean data $\boldsymbol{x}$ and the purified data $\hat{\boldsymbol{x}}(0)$ given by Eq. (4) satisfies that if with a probability of at least $1-\delta$, we have*

$$\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| \leq \|\boldsymbol{\epsilon}_a\| + \sqrt{e^{2\gamma(t^*)} - 1}C_\delta + \gamma(t^*)C_s \tag{10}$$

where $\gamma(t^*) := \int_0^{t^*} \frac{1}{2}\beta(s)ds$ *and the constant* $C_\delta := \sqrt{2d + 4\sqrt{d \log \frac{1}{\delta}} + 4 \log \frac{1}{\delta}}$.

Proof: Denote by $\boldsymbol{\epsilon}_a$ the adversarial perturbation, we have the adversarial example $\boldsymbol{x}_a = \boldsymbol{x} + \boldsymbol{\epsilon}_a$, where $\boldsymbol{x}$ represents the clean image. Because the diffused adversarial example $x(t^*)$ through the forward diffusion process satisfies

$$\boldsymbol{x}(t^*) = \sqrt{\alpha(t^*)}\boldsymbol{x}_a + \sqrt{1-\alpha(t^*)}\boldsymbol{\epsilon}_1 \tag{11}$$

where $\alpha(t) = e^{-\int_0^t \beta(s)ds}$ and $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, the L2 distance between the clean data x_0 and the purified data $\hat{x}(0)$ can be bounded as

$$\begin{aligned}
\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| &= \|\boldsymbol{x}(t^*) + (\hat{\boldsymbol{x}}(0) - \boldsymbol{x}(t^*)) - \boldsymbol{x}\| \\
&= \|\boldsymbol{x}(t^*) + \int_{t^*}^0 -\frac{1}{2}\beta(t)[\boldsymbol{x} + 2\boldsymbol{s}_\theta(\boldsymbol{x}, t)]dt + \int_{t^*}^0 \sqrt{\beta(t)}d\bar{\boldsymbol{w}} - \boldsymbol{x}\| \\
&\leq \|\underbrace{\boldsymbol{x}(t^*) + \int_{t^*}^0 -\frac{1}{2}\beta(t)\boldsymbol{x}dt + \int_{t^*}^0 \sqrt{\beta(t)}d\bar{\boldsymbol{w}}}_{\text{Integration of Linear SDE}} - \boldsymbol{x}\| + \|\int_{t^*}^0 -\beta(t)\boldsymbol{s}_\theta(\boldsymbol{x}, t)dt\|
\end{aligned} \tag{12}$$

where the second equation follows from the integration of the reverse-time SDE defined in Eq. (4), and in the last line we have separated the integration of the linear SDE from non-linear SDE involving the score function $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ by using the triangle inequality.

The above linear SDE is a time-varying Ornstein–Uhlenbeck process with a negative time increment that starts from t=t^* to t=0 with the initial value set to $\boldsymbol{x}(t^*)$. Denote by $\boldsymbol{x}'(0)$ its solution, from (Särkkä & Solin, 2019) we know $\boldsymbol{x}'(0)$ follows a Gaussian distribution, where its mean $\boldsymbol{\mu}(0)$ and covariance matrix $\boldsymbol{\Sigma}(0)$ are the solutions of the following two differential equations, respectively:

$$\frac{d\boldsymbol{\mu}}{dt} = -\frac{1}{2}\beta(t)\boldsymbol{\mu} \tag{13}$$

$$\frac{d\boldsymbol{\Sigma}}{dt} = -\beta(t)\boldsymbol{\Sigma} + \beta(t)\boldsymbol{I}_d \tag{14}$$

with the initial conditions $\boldsymbol{\mu}(t^*) = \boldsymbol{x}(t^*)$ and $\boldsymbol{\Sigma}(t^*) = \boldsymbol{0}$. By solving these two differential equations, we have that conditioned on $\boldsymbol{x}(t^*)$, $\boldsymbol{x}'(0) \sim \mathcal{N}(e^{\gamma(t^*)}\boldsymbol{x}(t^*), (e^{2\gamma(t^*)} - 1)\boldsymbol{I}_d)$, where $\gamma(t^*) := \int_0^{t^*} \frac{1}{2}\beta(s)ds$.

Using the reparameterization trick, we have:

$$\begin{aligned}
\boldsymbol{x}'(0) - \boldsymbol{x} &= e^{\gamma(t^*)}\boldsymbol{x}(t^*) + \sqrt{e^{2\gamma(t^*)} - 1}\boldsymbol{\epsilon}_2 - \boldsymbol{x} \\
&= e^{\gamma(t^*)}\left(e^{-\gamma(t^*)}(\boldsymbol{x} + \boldsymbol{\epsilon}_a) + \sqrt{1 - e^{-2\gamma(t)}}\boldsymbol{\epsilon}_1\right) + \sqrt{e^{2\gamma(t^*)} - 1}\boldsymbol{\epsilon}_2 - \boldsymbol{x} \\
&= \sqrt{e^{2\gamma(t^*)} - 1}(\boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_2) + \boldsymbol{\epsilon}_a
\end{aligned} \tag{15}$$

where the the second equation follows by substituting Eq. (11). Since $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are independent, the first term can be represented as a single zero-mean Normal variable with the variance $2(e^{2\gamma(t^*)} - 1)$. Assuming that the norm of the score function $\boldsymbol{s}_\theta(\boldsymbol{x}, t)$ is bounded by a constant $\frac{1}{2}C_s$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, we have:

$$\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| \leq \|\sqrt{2(e^{2\gamma(t^*)} - 1)}\boldsymbol{\epsilon} + \boldsymbol{\epsilon}_a\| + \gamma(t^*)C_s \tag{16}$$

Since $\|\boldsymbol{\epsilon}\|^2 \sim \chi^2(d)$, from the concentration inequality (Boucheron et al., 2013), we have

$$\Pr(\|\boldsymbol{\epsilon}\|^2 \geq d + 2\sqrt{d\sigma} + 2\sigma) \leq e^{-\sigma} \tag{17}$$

Let $e^{-\sigma} = \delta$, we get

$$\Pr\left(\|\boldsymbol{\epsilon}\| \geq \sqrt{d + 2\sqrt{d \log \frac{1}{\delta}} + 2 \log \frac{1}{\delta}}\right) \leq \delta \tag{18}$$

Therefore, with the probability of at least $1 - \delta$, we have

$$\|\hat{\boldsymbol{x}}(0) - \boldsymbol{x}\| \leq \|\boldsymbol{\epsilon}_a\| + \sqrt{e^{2\gamma(t^*)} - 1}C_\delta + \gamma(t^*)C_s \tag{19}$$

where the constant $C_\delta := \sqrt{2d + 4\sqrt{d \log \frac{1}{\delta}} + 4 \log \frac{1}{\delta}}$. $\square$

A.3. Proof of Proposition 3.3

Proposition A.3. *For the SDE in Eq. (4), the augmented SDE that computes the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)}$ of backpropagating through it is given by*

$$\begin{pmatrix} \boldsymbol{x}(t^*) \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)} \end{pmatrix} = \mathit{sdeint}\left(\begin{pmatrix} \hat{\boldsymbol{x}}(0) \\ \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)} \end{pmatrix}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}, \tilde{\boldsymbol{w}}, 0, t^* \right) \tag{20}$$

where $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)}$ is the gradient of the objective $\mathcal{L}$ w.r.t. the output $\hat{\boldsymbol{x}}(0)$ of the SDE in Eq. (4), and

$$\tilde{\boldsymbol{f}}([\boldsymbol{x}; \boldsymbol{z}], t) = \begin{pmatrix} \boldsymbol{f}_{rev}(\boldsymbol{x}, t) \\ \frac{\partial \boldsymbol{f}_{rev}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} \boldsymbol{z} \end{pmatrix}$$
$$\tilde{\boldsymbol{g}}(t) = \begin{pmatrix} -g_{rev}(t) \mathbf{1}_d \\ \mathbf{0}_d \end{pmatrix}$$
$$\tilde{\boldsymbol{w}}(t) = \begin{pmatrix} -\boldsymbol{w}(1-t) \\ -\boldsymbol{w}(1-t) \end{pmatrix}$$

with $\mathbf{1}_d$ and $\mathbf{0}_d$ representing the d-dimensional vectors of all ones and all zeros, respectively.

Proof: Before applying the adjoint method, we first transform the reverse-time SDE in Eq. (4) to a forward SDE, by a change of variable $t' := 1 - t$ such that $t' \in [1 - t^*, 1] \subset [0, 1]$. With this, the equivalent forward SDE with positive time increments from $t{=}1 - t^*$ to $t{=}1$ becomes

$$\hat{\boldsymbol{x}}(0) = \texttt{sdeint}(\boldsymbol{x}(t^*), \boldsymbol{f}_{\text{fwd}}, g_{\text{fwd}}, \boldsymbol{w}, 1 - t^*, 1) \tag{21}$$

where the drift and diffusion coefficients are

$$\boldsymbol{f}_{\text{fwd}}(\boldsymbol{x}, t) = -\boldsymbol{f}_{\text{rev}}(\boldsymbol{x}, 1 - t)$$
$$g_{\text{fwd}}(t) = g_{\text{rev}}(1 - t)$$

By following the stochastic adjoint method proposed in (Li et al., 2020), the augmented SDE that computes the gradient $\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)}$ of the objective $\mathcal{L}$ w.r.t. the input $\boldsymbol{x}(t^*)$ of the SDE in Eq. (21) is given by

$$\begin{pmatrix} \boldsymbol{x}(t^*) \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)} \end{pmatrix} = \texttt{sdeint}\left(\begin{pmatrix} \hat{\boldsymbol{x}}(0) \\ \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)} \end{pmatrix}, \tilde{\boldsymbol{f}}_{\text{fwd}}, \tilde{\boldsymbol{g}}_{\text{fwd}}, \tilde{\boldsymbol{w}}_{\text{fwd}}, -1, t^* - 1 \right) \tag{22}$$

where $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)}$ is the gradient of the objective $\mathcal{L}$ w.r.t. the output $\hat{\boldsymbol{x}}(0)$ of the SDE in Eq. (21), and the augmented drift coefficient $\tilde{\boldsymbol{f}}_{\text{fwd}} : \mathbb{R}^{2d} \times \mathbb{R} \to \mathbb{R}^{2d}$, the augmented diffusion coefficient $\tilde{\boldsymbol{g}}_{\text{fwd}} : \mathbb{R} \to \mathbb{R}^{2d}$ and the augmented Wiener process $\tilde{\boldsymbol{w}}(t) \in \mathbb{R}^{2d}$ are given by

$$\tilde{\boldsymbol{f}}_{\text{fwd}}([\boldsymbol{x}; \boldsymbol{z}], t) = \begin{pmatrix} -\boldsymbol{f}_{\text{fwd}}(\boldsymbol{x}, -t) \\ \frac{\partial \boldsymbol{f}_{\text{fwd}}(\boldsymbol{x}, -t)}{\partial \boldsymbol{x}} \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{\text{rev}}(\boldsymbol{x}, 1 + t) \\ \frac{\partial \boldsymbol{f}_{\text{rev}}(\boldsymbol{x}, 1 + t)}{\partial \boldsymbol{x}} \boldsymbol{z} \end{pmatrix}$$
$$\tilde{\boldsymbol{g}}_{\text{fwd}}(t) = \begin{pmatrix} -g_{\text{fwd}}(-t) \mathbf{1}_d \\ \mathbf{0}_d \end{pmatrix} = \begin{pmatrix} -g_{\text{rev}}(1 + t) \mathbf{1}_d \\ \mathbf{0}_d \end{pmatrix}$$
$$\tilde{\boldsymbol{w}}_{\text{fwd}}(t) = \begin{pmatrix} -\boldsymbol{w}(-t) \\ -\boldsymbol{w}(-t) \end{pmatrix}$$

with $\mathbf{1}_d$ and $\mathbf{0}_d$ representing the d-dimensional vectors of all ones and all zeros, respectively. Note that the augmented SDE in Eq. (22) moves from $t = -1$ to $t = t^* - 1$. Similarly, with the change of variable $t' := 1 + t$ such that $t' \in [0, t^*]$, we can rewrite the augmented SDE as

$$\begin{pmatrix} \boldsymbol{x}(t^*) \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}(t^*)} \end{pmatrix} = \texttt{sdeint}\left(\begin{pmatrix} \hat{\boldsymbol{x}}(0) \\ \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}(0)} \end{pmatrix}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}, \tilde{\boldsymbol{w}}, 0, t^* \right) \tag{23}$$

where

$$\tilde{\boldsymbol{f}}([\boldsymbol{x};\boldsymbol{z}],t) = \tilde{\boldsymbol{f}}_{\text{fwd}}([\boldsymbol{x};\boldsymbol{z}],t-1) = \begin{pmatrix} \boldsymbol{f}_{\text{rev}}(\boldsymbol{x},t) \\ \frac{\partial \boldsymbol{f}_{\text{rev}}(\boldsymbol{x},t)}{\partial \boldsymbol{x}} \boldsymbol{z} \end{pmatrix}$$

$$\tilde{\boldsymbol{g}}(t) = \tilde{\boldsymbol{g}}_{\text{fwd}}(t-1) = \begin{pmatrix} -g_{\text{rev}}(t)\mathbf{1}_d \\ \mathbf{0}_d \end{pmatrix}$$

$$\tilde{\boldsymbol{w}}(t) = \tilde{\boldsymbol{w}}_{\text{fwd}}(t-1) = \begin{pmatrix} -\boldsymbol{w}(1-t) \\ -\boldsymbol{w}(1-t) \end{pmatrix}$$

and $\boldsymbol{f}_{\text{rev}}$ and g_{rev} are given by Eq. (5). □

B. More Details of Experimental Settings

B.1. Implementation Details of Our Method

First, our method requires solving two SDEs: a reverse-time denosing SDE in Eq. (4) to get purified images, and an augmented SDE in Eq. (6) to compute gradients through the SDE in Eq. (4). In experiments, we use the adjoint framework for SDEs named `adjoint_sdeint` in the TorchSDE library: https://github.com/google-research/torchsde for both adversarial purification and gradient evaluation. We use the simple Euler-Maruyama method to solve both SDEs with a fixed step size dt=10^{-3}. Ideally, the step size should be as small as possible to ensure that our gradient computation has an infinitely small numerical error. However, small steps sizes come with a high computational cost due to the increase in the number of neural network evaluations. We empirically observe that the robust accuracy of our method barely change any more if we further reduce the step size from 10^{-3} to 10^{-4} during a sanity check. Hence, we use a step size of 10^{-3} for all experiments to save the time in the purification process through the SDE solver. Note that this step size is often used in the denoising diffusion models as well (Ho et al., 2020; Song et al., 2021b).

Second, our method also requires the pre-trained diffusion models. In experiments, we use different pre-trained models on three datasets: Score SDE (Song et al., 2021b) for CIFAR-10, Guided Diffusion (Dhariwal & Nichol, 2021) for ImageNet and DDPM (Ho et al., 2020) for CelebA-HQ. In specific, we use the `vp/cifar10_ddpmpp_deep_continuous` checkpoint from the `score_sde` library: https://github.com/yang-song/score_sde for the CIFAR-10 experiments. We use the `256x256 diffusion (unconditional)` checkpoint from the `guided-diffusion` library: https://github.com/openai/guided-diffusion for the ImageNet experiments. Finally, for the CelebA-HQ experiments, we use the CelebA-HQ checkpoint from the `SDEdit` library: https://github.com/ermongroup/SDEdit.

B.2. Implementation Details of Adversarial Attacks

AutoAttack We use AutoAttack to compare with the state-of-the-art adversarial training methods, as reported in the RobustBench benchmark. To make a fair comparison, we uses their codebase: https://github.com/RobustBench/robustbench with default hyperparameters for evaluation. Similarly, we set $\epsilon = 8/255$ and $\epsilon = 0.5$ for AutoAttack ℓ_∞ and AutoAttack ℓ_2, respectively, on CIFAR-10. For AutoAttack ℓ_∞ on ImageNet, we set $\epsilon = 4/255$.

There are two versions of AutoAttack: (i) the STANDARD version, which contains four attacks: APGD-CE, APGD-T, FAB-T and Square, and is mainly used for evaluating deterministic defense methods, and (ii) the RAND version, which contains two attacks: APGD-CE and APGD-DLR, and is used for evaluating stochastic defense methods. Because there is stochasticity in our method, we consider the RAND version and choose the default EOT=20 for both ℓ_∞ and ℓ_2 after searching for the minimum EOT with which the robust accuracy does not further decrease (see Figure 6).

In practice, we find that in a few cases, the STANDARD version actually makes a stronger attack (indicated by a lower robust accuracy) to our method than the RAND version. Therefore, to measure the worse-case defense performance of our method, we run both the STANDARD version and the RAND version of AutoAttack, and report the minimum robust accuracy of these two versions as our final robust accuracy.

StAdv We use the StAdv attack to demonstrate that our method can defend against unseen threats beyond ℓ_p-norm attacks. We closely follow the codebase of PAT (Laidlaw et al., 2021): https://github.com/cassidylaidlaw/perceptual-advex with default hyperparameters for evaluation. Moreover, we add the EOT to average out the stochasticity of gradients in our defense method. Similarly, we use EOT=20 by default for the StAdv attack after searching for the minimum EOT that the robust accuracy saturates (see Figure 6).

BPDA+EOT For many adversarial purification methods where there exists an optimization loop or non-differentiable operations, the BPDA attack is known as the strongest attack (Tramer et al., 2020). Taking the stochastic defense methods into account, the BPDA+EOT attack has become the default one when evaluating the state-of-the-art adversarial purification methods (Hill et al., 2021; Yoon et al., 2021). To this end, we use the BPDA+EOT implementation of (Hill et al., 2021): `https://github.com/point0bar1/ebm-defense` with default hyperparameters for evaluation.

B.3. Implementation Details of Baselines

Purification models on CelebA-HQ We mainly consider the state-of-the-art VAEs and GANs as purification models for comparison, and in particular we use NVAE (Vahdat & Kautz, 2020) and StyleGAN2 (Karras et al., 2020) in our experiments. To use NVAE as a purification model, we directly pass the adversarial images to its encoder and get the purified images from its decoder. To use StyleGAN2 as a purification model, we consider three GAN inversion methods to first invert adversarial images into the latent space of StyleGAN2, and then get the purified images through the StyleGAN2 generator. The three GAN inversion methods that we use in our experiments are as follows:

- GAN+OPT, an optimization-based GAN inversion method that minimizes the perceptual distance between the output image and the input image w.r.t the w+ latent code. We use the codebase: `https://github.com/rosinality/stylegan2-pytorch/blob/master/projector.py` that closely follows the idea of (Karras et al., 2020) for the GAN+OPT implementation. The only difference is that the number of optimization iterations we use is $n = 500$ to save the computational time while the original number of optimization iterations is $n = 1000$. We find that for $n \geq 500$, the recovered images of GAN+OPT do not change much if we increases n.
- GAN+ENC, an encoder-based GAN inversion method that uses an extra encoder to encode the input image to the w+ latent code. We use the codebase: `https://github.com/eladrich/pixel2style2pixel` corresponding to the idea of pixel2style2pixel (pSp) (Richardson et al., 2021) for the GAN+ENC implementation.
- GAN+ENC+OPT, which combines the optimization-based and encoder-based GAN inversion methods. We use the codebase: `https://github.com/chail/gan-ensembling` corresponding to the idea of (Chai et al., 2021) that uses the w+ latent code from the encoder as the initial point for the optimization with $n = 500$ iterations.

B.4. Other Sampling Strategies

Here we provide more details of other sampling strategies based on the same pre-trained diffusion models.

LD-SDE We denote an adversarial image by $\boldsymbol{x}_a$ and the corresponding clean image by $\boldsymbol{x}$. Currently with the Langevin dynamics (LD) sampling for purification, given $\boldsymbol{x}_a$ we are searching for $\boldsymbol{x}$ freely (Hill et al., 2021; Yoon et al., 2021). Our approach can be considered as conditional sampling of clean image given the adversarial image using $p(\boldsymbol{x}|\boldsymbol{x}_a) = \int p(\boldsymbol{x}(t^*)|\boldsymbol{x}_a)p(\boldsymbol{x}|\boldsymbol{x}(t^*))d\boldsymbol{x}(t^*)$ where $p(\boldsymbol{x}(t^*)|\boldsymbol{x}_a)$ is first sampled by following the forward diffusion and $p(\boldsymbol{x}|\boldsymbol{x}(t^*))$ is then sampled by following the reverse diffusion process. However, there are different ways in which one can formulate this conditional sampling without introducing forward and reverse diffusion processes.

We can write $p(\boldsymbol{x}|\boldsymbol{x}_a) \propto p(\boldsymbol{x})p(\boldsymbol{x}_a|\boldsymbol{x})$ where $p(\boldsymbol{x})$ represent the distribution of clean images and $p(\boldsymbol{x}_a|\boldsymbol{x})$ is the distribution adversarial image given the clean image which we will approximate it by a simple Gaussian distribution $p(\boldsymbol{x}_a|\boldsymbol{x}) = N(\boldsymbol{x}_a; \boldsymbol{x}, \sigma^2 \boldsymbol{I}_d)$. We also do not have access to the true clean data distribution, but we will assume that $p(\boldsymbol{x}) \propto e^{h(\boldsymbol{x})}$ is denoted by a trained generator (more on this later).

As we can see above, we can assume that $p(\boldsymbol{x}|\boldsymbol{x}_a) \propto e^{-E(\boldsymbol{x}|\boldsymbol{x}_a)}$ with energy function $E(\boldsymbol{x}|\boldsymbol{x}_a) = -h(\boldsymbol{x}) + \frac{||\boldsymbol{x}_a - \boldsymbol{x}||_2^2}{2\sigma^2}$. Similarly, sampling from $p(\boldsymbol{x}|\boldsymbol{x}_a)$ can be done by running the overdamped LD:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\lambda \Delta_t}{2} \nabla_{\boldsymbol{x}_t} E(\boldsymbol{x}_t|\boldsymbol{x}_a) + \eta \sqrt{\lambda \Delta_t} \boldsymbol{\epsilon}_t \tag{24}$$

where $\lambda \Delta_t$ is the learning rate and η denotes the damping coefficient. When Δ_t is infinitely small, it corresponds to solving the following forward SDE (termed LD-SDE):

$$d\boldsymbol{x} = -\frac{\lambda}{2} \nabla_{\boldsymbol{x}} E(\boldsymbol{x}|\boldsymbol{x}_a)dt + \eta \sqrt{\lambda} d\boldsymbol{w} \tag{25}$$

for $t \in [0, 1]$ where $\boldsymbol{w}$ is the standard wiener process. Note the SDE above does not involve any diffusion and denoising and it only uses LD for sampling from a fixed energy function. Recall that $\nabla_{\boldsymbol{x}} h(\boldsymbol{x})$ is exactly what we have learned by the score function at timestep t=0 in diffusion models, *i.e.*, $\nabla_x h(x) \approx \boldsymbol{s}_\theta(\boldsymbol{x}, 0)$. Thus, the LD-SDE formulation is

$$d\boldsymbol{x} = -\frac{1}{2}\left(-\boldsymbol{s}_\theta(\boldsymbol{x}, 0) + \frac{\boldsymbol{x} - \boldsymbol{x}_a}{\sigma^2}\right)\lambda dt + \eta\sqrt{\lambda}d\boldsymbol{w} \tag{26}$$

Note that there exist three hyperparameters $(\sigma^2, \lambda, \eta)$ that we have to tune for the best performance. In particular, σ^2 controls the balance of the attraction term $\boldsymbol{x} - \boldsymbol{x}_a$ (to make $\boldsymbol{x}$ stay close to $\boldsymbol{x}_a$) and the score function $-\boldsymbol{s}_\theta(\boldsymbol{x}, 0)$ (to make x follow the clean data distribution). When σ^2 becomes infinitely large, there is no attraction term $\boldsymbol{x} - \boldsymbol{x}_a$, and the LD-SDE defined in Eq. (26) reduces to the SDE formulation of the normal LD sampling (Grathwohl et al., 2020; Hill et al., 2021; Yoon et al., 2021). Note that with this SDE formulation of the LD sampling, we can use the adjoint method as discussed in Section 3.2 for an evaluation with strong adaptive attacks, such as AutoAttack.

In experiments, to find the best set of hyperparameters $(\sigma^2, \lambda, \eta)$, we first perform a grid search on $\sigma^2 = \{0.001, 0.01, 0.1, 1, 10, 100\}$, $\lambda = \{0.01, 0.1, 1, 10\}$, $\eta = \{0.1, 1, 5, 10\}$. We find that the best performing configuration is $\sigma^2 = 100,\ \lambda = 0.1,\ \eta = 1.0$. Since $\sigma^2 = 100$ works the best, it implies that LD-SDE performs better without the attraction term $\boldsymbol{x} - \boldsymbol{x}_a$ in Eq. (26).

Table 7. Robust accuracies of baselines obtained from RobustBench vs. from our experiments against ℓ_∞ threat model ($\epsilon = 8/255$) with AutoAttack on CIFAR-10, obtained by different classifier architectures.

Method	Extra Data	Robust Acc (from RobustBench)	Robust Acc (from our experiments)
WideResNet-28-10			
(Zhang et al., 2020)	✓	59.64	59.96
(Wu et al., 2020)	✓	60.04	62.11
(Gowal et al., 2020)	✓	62.80	62.70
(Wu et al., 2020)	✗	56.17	59.18
(Rebuffi et al., 2021)	✗	60.75	61.72
(Gowal et al., 2021)	✗	63.44	65.24
WideResNet-70-16			
(Gowal et al., 2020)	✓	65.88	66.02
(Rebuffi et al., 2021)	✓	66.58	68.56
(Gowal et al., 2020)	✗	57.20	59.57
(Rebuffi et al., 2021)	✗	64.25	64.46
(Gowal et al., 2021)	✗	66.11	66.60

VP-ODE For the reverse generative VP-SDE defined by:

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)[\boldsymbol{x} + 2\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}} \tag{27}$$

where the time flows backward from $t = 1$ to 0 and $\bar{w}$ is the reverse standard Wiener process, Song et al. (2021b) show that there exists an equivalent ODE whose trajectories share the same marginal probability densities $p_t(x(t))$:

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\left[\boldsymbol{x} + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right]dt \tag{28}$$

where the idea is to use the Fokker-Planck equation (Särkkä & Solin, 2019) to transform an SDE to an ODE (see Appendix D.1 for more details in (Song et al., 2021b)). Therefore, we can use the above ODE, termed VP-ODE, to replace the reverse generative VP-SDE in our method for purification. Similarly with this VP-ODE sampling, we can also use the adjoint method for an evaluation with strong adaptive attacks, such as AutoAttack.

B.5. Gradient Computation in an Analytic Example

Here we provide a simple example to show that the gradient obtained from the adjoint method will closely match its ground-truth value if the SDE solver has a small numerical error. In this example, we know the analytic solution of gradient through a reverse-time SDE, and thus we can compare the difference between the gradient from solving the augmented SDE in Eq. (6) and its analytic solution.

In specific, we assume the data follows a Gaussian distribution, *i.e.*, $x \sim p_0(x) = \mathcal{N}(\mu_0, \sigma_0^2)$. The nice property about the diffusion process in Eq. (3) is that if $p_0(x)$ is a Gaussian distribution, $p_t(x)$ is also Gaussian for all t. From Eq. (3)

in VP-SDE, we have $p_t(x) = \mathcal{N}(\mu_t, \sigma_t^2)$ where $\mu_t = \mu_0\sqrt{\alpha(t)}$ and $\sigma_t^2 = 1 - (1-\sigma_0^2)\alpha(t)$, with $\alpha_t := e^{-\int_0^t \beta(s)ds}$. Therefore, if we fix μ_0 and σ_0 to particular values, we can easily evaluate $p_{t^*}(x)$ at the diffusion timestep t^*.

Recall that the reverse process can be described as:

$$dx = -\frac{1}{2}\beta(t)[x + 2\nabla_x \log p_t(x)]dt + \sqrt{\beta(t)}d\bar{w}$$

where $w(t)$ is a standard reverse-time Wiener process. Since we have $p_t(x)$ analytically, we can write $\nabla_x \log p_t(x) = -\frac{x-\mu_t}{\sigma_t^2}$. So the reverse process is:

$$x(0) = \mathtt{sdeint}(x(t^*), f_{\text{rev}}, g_{\text{rev}}, \bar{w}, t^*, 0) \tag{29}$$

where the drift and diffusion coefficients are given by

$$f_{\text{rev}}(x,t) := -\frac{1}{2}\beta(t)\left[\frac{(\sigma_t^2-2)x + 2\mu_t}{\sigma_t^2}\right]$$
$$g_{\text{rev}}(t) := \sqrt{\beta(t)}$$

Then, we can compute gradient (denoted by ϕ_{adj}) of $x(0)$ w.r.t. $x(t^*)$ through the SDE in Eq. (29) using the adjoint method in Eq. (6), where we use the objective $\mathcal{L} := x(0)$ for simplicity.

On the other hand, let $x(t) := x_t$, we can evaluate $p(x_0|x_t)$ as well which is

$$\begin{aligned} p(x_0|x_t) &\propto p(x_0)p(x_t|x_0) \propto \exp\left(-\frac{(x_0-\mu_0)^2}{2\sigma_0^2} - \frac{(x_t - x_0\sqrt{\alpha_t})^2}{2(1-\alpha_t)}\right) \\ &\propto \exp\left(-\frac{(1-\alpha_t+\sigma_0^2\alpha_t)x_0^2 - 2((1-\alpha_t)\mu_0 + \sigma_0^2\sqrt{\alpha_t}x_t)x_0}{2(1-\alpha_t)\sigma_0^2}\right) \\ &= \mathcal{N}(\frac{(1-\alpha_t)\mu_0 + \sigma_0^2\sqrt{\alpha_t}x_t}{1-\alpha_t+\sigma_0^2\alpha_t}, \frac{(1-\alpha_t)\sigma_0^2}{1-\alpha_t+\sigma_0^2\alpha_t}) \end{aligned} \tag{30}$$

where we use $p(x_t|x_0) = \mathcal{N}(x_0\sqrt{\alpha_t}, 1-\alpha_t)$ by following Eq. (3). Thus, we can get the analytic solution (denoted by ϕ_{ana}) of the gradient of $x(0)$ w.r.t. $x(t^*)$ as follows:

$$\phi_{\text{ana}} := \frac{\partial x(0)}{\partial x(t^*)} = \frac{\sigma_0^2\sqrt{\alpha_{t^*}}}{1-\alpha_{t^*}+\sigma_0^2\alpha_{t^*}} \tag{31}$$

In experiments, we set $\mu_0 = 0$ and $\sigma_0^2 \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$, and we use the Euler-Maruyama method to solve our SDEs with different scales of step sizes. The difference between the numeric gradient and the analytic gradient vs. the step size is shown in Figure 5, where the numeric error of gradients is measured by $|\phi_{\text{ana}} - \phi_{\text{adj}}|/|\phi_{\text{ana}}|$. As the step size gets smaller, the numeric error monotonically decreases at the same rate in different settings. It implies that the gradient obtained from the adjoint method will closely match its ground-truth value if the step size in the SDE solver is small.

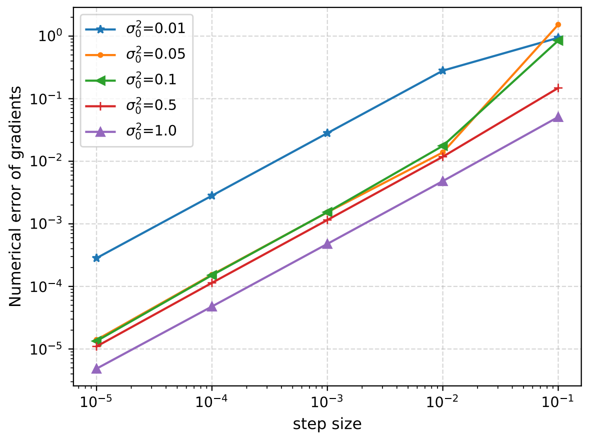


Figure 5. Impact of the step size in the Euler-Maruyama method to solve our SDEs on the numeric error of gradients with different σ_0^2 values. As the step size gets smaller, the numeric error monotonically decreases at the same rate in different settings.

C. More Experimental Results

C.1. Robust Accuracies of Our Method for Standard Attack and Black-box Attack

In general adaptive attacks are considered to be stronger than standard attack (*i.e.*, non-adaptive). Following the checklist of Croce et al. (2022), we report the performance of DiffPure for standard attacks in Table 8. We can see that 1) AutoAttack is effective on the static model as its robust accuracies are zero, and 2) standard attacks are not effective on our method as our robust accuracies against standard attacks are much better than those against adaptive attacks (*ref.* Tables 1-3).

The AutoAttack (the "standard" version) we have considered includes the black-box Square Attack, but evaluating with Square Attack separately is a more direct way to show insensitivity to gradient masking. We thus show the performance of DiffPure for Square Attack in Table 9. We can see that 1) our method has much higher robust accuracies against Square Attack than the static model, and 2) our robust accuracies against Square Attack are higher than those against AutoAttack (*ref.* Table 1-3). These results directly show DiffPure's insensitivity to gradient masking.

Table 8. Robust accuracies of our method against standard attacks, where we transfer AutoAttack from the static model (i.e., the classifier) to our method DiffPure, and the other evaluation setting and hyperparameters are the same as those in Tables 1-3.

Dataset	Network	ℓ_p-norm	Static Model	Ours
CIFAR-10	WRN-28-10	ℓ_∞	0.00	89.58±0.49
CIFAR-10	WRN-28-10	ℓ_2	0.00	90.37±0.24
ImageNet	ResNet-50	ℓ_∞	0.00	67.01±0.97

Table 9. Robust accuracies of the static model (i.e., the classifier) and our method DiffPure against Square Attack, where the other evaluation setting and hyperparameters are the same as those in Tables 1-3.

Dataset	Network	ℓ_p-norm	Static Model	Ours
CIFAR-10	WRN-28-10	ℓ_∞	0.33	85.42±0.65
CIFAR-10	WRN-28-10	ℓ_2	21.42	88.02±0.23
ImageNet	ResNet-50	ℓ_∞	9.25	62.88±0.65

Table 10. Robust accuracies of baselines obtained from RobustBench vs. from our experiments against ℓ_2 threat model ($\epsilon = 0.5$) with AutoAttack on CIFAR-10, obtained by different classifier architectures. Methods marked by * use WideResNet-34-10, with the same width but more layers than the default one.

Method	Extra Data	Robust Acc (from RobustBench)	Robust Acc (from our experiments)
WideResNet-28-10			
(Augustin et al., 2020)*	✓	76.25	77.93
(Rony et al., 2019)	✗	66.44	66.41
(Ding et al., 2020)	✗	66.09	67.77
(Wu et al., 2020)*	✗	73.66	72.85
(Sehwag et al., 2021)*	✗	76.12	75.39
(Rebuffi et al., 2021)	✗	78.80	78.32
WideResNet-70-16			
(Gowal et al., 2020)	✓	80.53	79.88
(Rebuffi et al., 2021)	✓	82.32	81.44
(Gowal et al., 2020)	✗	74.50	74.03
(Rebuffi et al., 2021)	✗	80.42	80.86

C.2. Robust Accuracies of Baselines Obtained from RobustBench vs. from Our Experiments

When we compare with the state-of-the-art adversarial training methods in the RobustBench benchmark, we use the default hyperparameters for the AutoAttack evaluation. However, since the computational time of evaluating our method with AutoAttack is high (usually taking 50-150 number of function evaluations per attack iteration), we compare the robust accuracy of our method with baselines on a fixed subset of 512 images that is randomly sampled from the test set.

To show the validity of the results on this subset, we compare the robust accuracies of baselines reported from RobustBench (on the whole test set) vs. from our experiments (on the sampled subset), shown in Tables 7-11. We can see that for different datasets (CIFAR-10 and ImageNet) and network architectures (ResNet and WideResNet), the gap in robust accuracies of most baselines is small (*i.e.*, less than 1.5% discrepancy). Furthermore, the relative performances of different methods remain the same. These results demonstrate that it is both efficient and effective to evaluate on the fixed subset.

Table 11. Robust accuracies of baselines obtained from RobustBench vs. from our experiments against ℓ_∞ threat model ($\epsilon = 4/255$) with AutoAttack on ImageNet, obtained by different classifier architectures.

Method	Extra Data	Robust Acc (from RobustBench)	Robust Acc (from our experiments)
ResNet-50			
(Engstrom et al., 2019)	✗	29.22	31.06
(Wong et al., 2020)	✗	26.24	26.95
(Salman et al., 2020)	✗	34.96	37.89
WideResNet-50-2			
(Salman et al., 2020)	✗	38.14	39.25

C.3. More Results of Comparison within Adversarial Purification on CelebA-HQ

Here we compare with other adversarial purification methods by using the BPDA+EOT attack with ℓ_∞ perturbations on the *smiling* attribute classifier for CelebA-HQ. The results are shown in Table 12. We can see that our method still largely outperforms all the baselines, with an absolute improvement of at least +18.78% in robust accuracy. Compared with the *eyeglasses* attribute, the *smiling* attribute is more difficult to classify, posing a bigger challenge to the defense method. Thus, the robust accuracies of most defense methods are much worse than those with the *eyeglasses* attribute classifier.

Table 12. We evaluate with the BPDA+EOT attack on the *smiling* attribute classifier for CelebA-HQ, where $\epsilon = 16/255$ for the ℓ_∞ perturbations. Note that OPT and ENC denote the optimization-based and econder-based GAN inversions, respectively, and ENC+OPT implies a combination of OPT and ENC.

Method	Purification	Standard Acc	Robust Acc
(Vahdat & Kautz, 2020)	VAE	93.55	0.00
(Karras et al., 2020)	GAN+OPT	93.49	3.41
(Chai et al., 2021)	GAN+ENC+OPT	**93.68**	0.78
(Richardson et al., 2021)	GAN+ENC	90.55	40.40
Ours ($t^* = 0.4$)	Diffusion	89.78±0.14	55.73±0.97
Ours ($t^* = 0.5$)	Diffusion	87.62±0.22	**59.12±0.37**

C.4. More Results of Ablation Studies

Impact of EOT Because of the stochasticity in our purification process, we seek for the EOT value that is sufficient for different threat models to evaluate our method. In Figure 6, we present the robust accuracies of our method over three threat models - ℓ_∞, ℓ_2 and StAdv, respectively, with different number of EOT. We can see that these threat model has different behaviors with the number of EOT: our robust accuracy against the ℓ_2 threat model seems to be not affected by EOT while our robust accuracies against the ℓ_∞ and StAdv threat models first decrease and then saturate as the number of EOT increases. In particular, our robust accuracy saturates at EOT=5 for ℓ_∞ and at EOT=20 for StAdv. Therefore, we consider EOT=20 by default for all our experiments unless stated otherwise, which should be sufficient for threat models to evaluate our method.

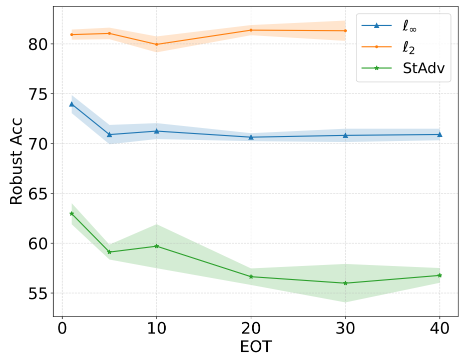


Figure 6. Impact of EOT on robust accuracies against the ℓ_∞ ($\epsilon = 8/255$), ℓ_2 ($\epsilon = 0.5$) and StAdv ($\epsilon = 0.05$) threat models, respectively, where we evaluate on WideResNet-28-10 for CIFAR-10.

Randomizing diffusion timestep t^* Since randomness matters for adversarial purification methods (Hill et al., 2021; Yoon et al., 2021), we here consider introduce another source of randomness by randomizing the diffusion timestep t^*: Instead of using a fixed t^*, we uniformly sample t^* from the range $[\bar{t} - \Delta_t, \bar{t} + \Delta_t]$ for every diffusion process. Table 13 shows the robustness performance with different Δ_t, where a larger Δ_t means stronger randomness introduced by perturbing t^*. We can see that the mean of standard accuracy monotonically decreases and the variance of robust accuracy monotonically increases with Δ_t due to the stronger randomness. Besides, a slightly small Δ_t may improve the robust accuracy in an average sense, while a large Δ_t may hurt it. It implies that there may also exist a sweet spot for the perturbation strength Δ_t of diffusion timestep t^* to get the best robustness performance.

Table 13. Uniformly sampling diffusion timestep t^* from $[\bar{t} - \Delta_t, \bar{t} + \Delta_t]$, where $\bar{t} = 0.1$. We evaluate with AutoAttack (the RAND version with EOT=20) ℓ_∞ ($\epsilon = 8/255$) on WideResNet-28-10 for CIFAR-10. Note that $\Delta_t = 0$ reduces to our method with a fixed t^*.

Δ_t	Standard Acc	Robust Acc
0	**89.02±0.21**	70.64±0.39
0.015	88.86±0.22	**72.14±1.45**
0.025	88.04±0.33	69.21±2.74

C.5. Purifying Adversarial Examples of Standard Attribute Classifiers

In Figure 7, we provide more visual examples of how our methods purify the adversarial examples of standard classifiers.

C.6. Inference Time with and without DiffPure

Inference time (in seconds) by varying diffusion timestep t^* is reported in Table 14, where the inference time increases linearly with t^*. We believe our inference time can be reduced using recent fast sampling methods for diffusion models, but we leave it as the future work.

Table 14. Inference time with DiffPure ($t^* > 0$) and without DiffPure ($t^* = 0$) for a single image on an NVIDIA V100 GPU, where the time increase over $t^* = 0$ is given in parenthesis.

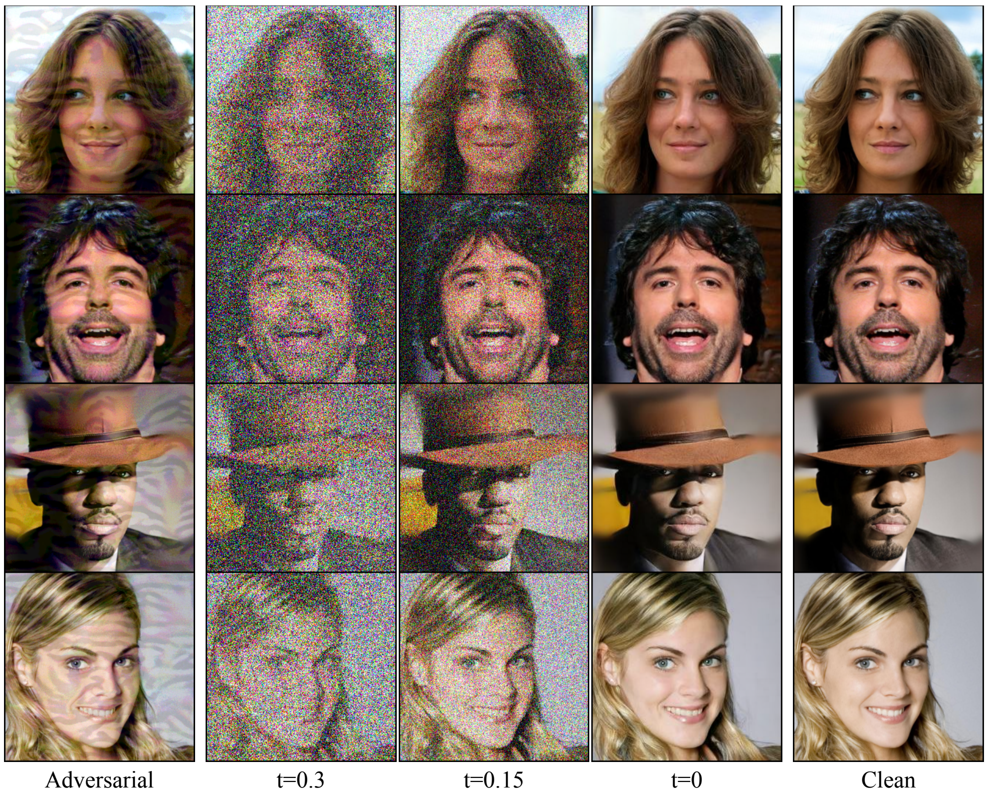
Dataset	Network	t^*=0	t^*=0.05	t^*=0.1	t^*=0.15
CIFAR-10	WRN-28-10	0.055	5.12(×93)	10.56(×190)	15.36(×278)
ImageNet	ResNet-50	0.062	5.58(×90)	11.13(×179)	17.14(×276)

C.7. Crafting Examples Just for Diffusion Model

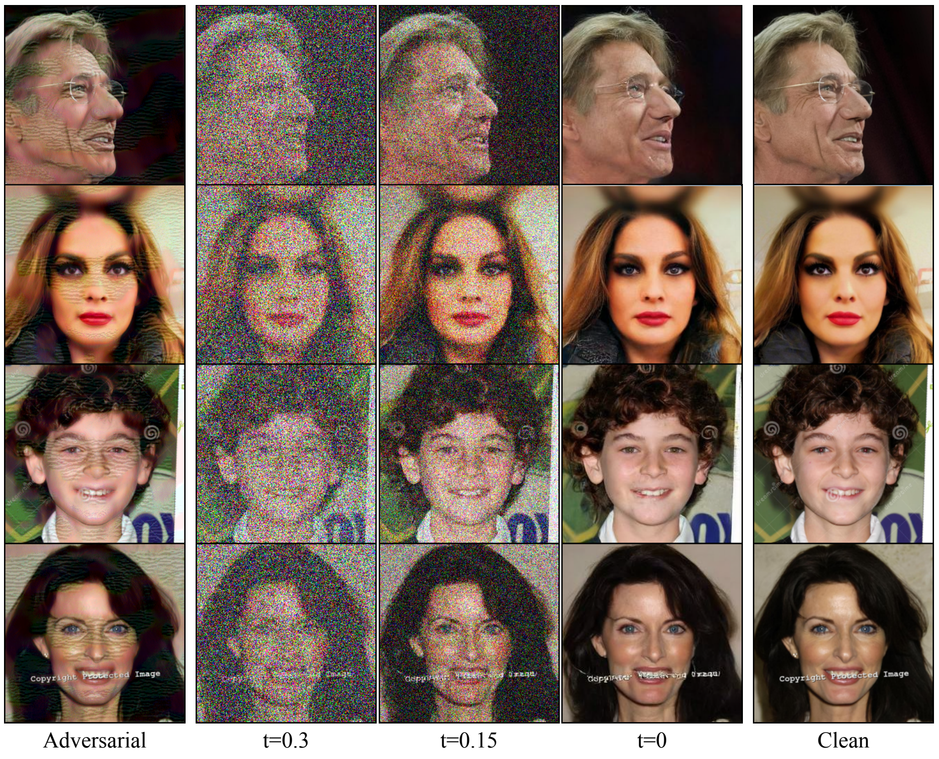
It is interesting to see if the adversary can craft examples just for the diffusion model, such that the recovered images from DiffPure become different from the original clean images, which may also result in the misclassification. To this end, we use APGD (EOT=20) to attack the diffusion model only by maximizing the mean squared error (MSE) between diffusion model's outputs and input images. The results are given in Table 15. We see that attacking the diffusion model is less effective than attacking the whole defense system.

Table 15. Robust accuracies of attacking the diffusion model only ("Diffusion only") and the whole defense system ("Diffusion+Clf") using APGD ℓ_∞ and ℓ_2 on CIFAR-10.

ℓ_p-norm	Network	t^*	Diffusion only	Diffusion+Clf
ℓ_∞ (ϵ=8/255)	WRN-28-10	0.1	85.04±0.86	75.91±0.74
ℓ_2 (ϵ=0.5)	WRN-28-10	0.075	90.82±0.42	84.83±0.09



(a) Smiling



(b) Eyeglasses

Figure 7. Our method purifies adversarial examples (first column) produced by attacking attribute classifiers using PGD ℓ_∞ ($\epsilon = 16/255$), where $t^* = 0.3$. The middle three columns show the results of the SDE in Eq. (4) at different timesteps, and we observe the purified images at t=0 match the clean images (last column). Better zoom in to see how we remove adversarial perturbations.