ADVERSARIAL ATTACK AND DEFENSE FOR DENOISING DIFFUSION SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Denoising diffusion sampling (DDS) is an emerging approach for generating new samples that have the same distribution as some training samples. However, it is vulnerable to adversarial attacks by even a Gaussian perturbation. In this work, we propose a complete set of adversarial attack and defense methodology for DDS. In the attack side, we propose to inject a perturbation to the sampling stage, which significantly worsen the performance of sample generation. In the defense side, we propose a local variation based regularization model for the potential function minimization, which effectively tolerates the adversarial perturbations. Moreover, we develop a conjugate gradient algorithm to solve the defense model, which integrates with a recently-developed zeroth order rejection sampling method that saves computational cost. Experimental results show that the proposed attack significantly worsen the existing state-of-the-art methods, but can be defended by the proposed local variation regularization.

1 INTRODUCTION

Sampling from a distribution is a fundamental task in machine learning. It is widely used not only in traditional areas like decision making, statistical inference, and optimization, but also in frontier areas like diffusion generative modeling, molecular dynamics, and differential privacy. Due to recent developments of diffusion models, it is now possible to generate new samples that have the same distribution as some training samples, instead of from the exact distribution itself. This significantly improves the convenience of sampling under certain assumptions. For example, some diffusion processes and gradient flows require an isoperimetric property (Vempala & Wibisono, 2019; He et al., 2020; Salim et al., 2022; Carrillo & Skrzeczkowski, 2025). Some diffusion generative models require an approximation of the score function with certain accuracy (Benton et al., 2024; Chen et al., 2023; 2022; Li et al., 2024). Some Monte Carlo (MC) based methods (He et al., 2024) require the oracle minimum of the potential function $V(x)$ (i.e., the unnormalized density of $x \propto \exp(-V(x))$). These methods are able to handle various sampling tasks on unnormalized, log-concave, or non-log-concave densities, etc.

Generally speaking, the mechanism of the above-mentioned sampling approach is to add a Brownian noise B_t to a training sample X_t in the forward process, and then denoise with another Brownian noise $\bar{B}_t$ (independent of B_t) in the backward process. By this means, a new sample $\bar{X}_t$ that is different from X_t but has the same distribution as X_t can be generated. A crucial stage of this approach is the Denoising Diffusion Sampling (DDS), which aims to recover the underlying distribution and generate a new sample. However, it is widely-recognized that sampling can be vulnerable to adversarial attacks (Lee et al., 2018; Ben-Eliezer & Yogev, 2020; Braverman et al., 2021; Ben-Eliezer et al., 2022; Woodruff & Zhou, 2024). In a decentralized setting (Lian et al., 2017; Koloskova et al., 2020; Cyffers & Bellet, 2022; Cyffers et al., 2023), the variable value at the previous timestamp $x_{t_{k-1}}$ should be transmitted from the previous user $(k-1)$ to the next user k. During this transmission, another perturbation δ_t can be mixed into the current variable value x_{t_k} intentionally or unintentionally. Besides, some accidental or intrinsic factors may also cause such a perturbation, like channel interference and sampling errors. Worse still, such an adversarial attack can happen at each of the hundreds or even thousands of timestamps in an entire DDS process. Note that this perturbation δ_t is different from the above-mentioned Brownian noises B_t and $\bar{B}_t$. Instead, δ_t aims to damage the adding of B_t and the denoising with $\bar{B}_t$, in order to sabotage the generation of new correct samples. Even a prevalent Gaussian perturbation can significantly diminish the performance

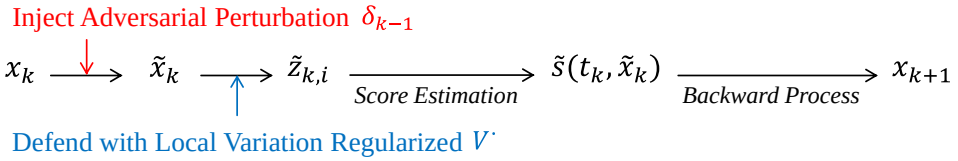


Figure 1: Diagram of the entire adversarial attack and defense scheme for denoising diffusion sampling in a decentralized scenario. While variable x_k (the simplified version of x_{t_k}) is transmitted from the $(k-1)$-th user to the k-th user, an adversarial perturbation δ_{k-1} can be injected to yield a perturbed variable $\tilde{x}_k$. In the defense side, the local variation regularized $V^{\bullet}$ not only tolerates adversarial perturbations, but also integrates with a zeroth-order rejection sampling scheme to produce MC samples $\{\tilde{z}_{t,i}\}$. Then the score estimation and backward process can be implemented to produce the next variable x_{k+1}. In the next round, x_{k+1} may still be attacked and the same attack-defense dynamic continues.

of DDS, as shown in Figures B2(a)(b), B3(a)(b), B4, B5, B8(a)(b), B9, B10, B13(a)(b), B14, B15, and B18(a)(b). Hence this vulnerability is rather significant.

To develop an adversarial defense scheme, we notice that the methodology of total variation (TV) is a traditional denoising approach that has been widely used in signal processing, image restoration, optimal control, and data transmission (Rudin et al., 1992; Chan et al., 2006; Chen et al., 2006). Recently, it has been verified to be effective in invariant feature extraction and out-of-distribution generalization (Lai & Wang, 2024; Wang et al., 2025). Therefore, we extract a local segment $|\nabla V(x)|$ of TV, to quantify the **local variation (LV)** of potential function $V(x)$ with respect to (w.r.t.) the variable value x. LV can effectively capture the adversarial perturbations inside the DDS process. On the other hand, DDS mostly depends on MC methods with thousands of intermediate MC samples z, which are mostly first-order methods (i.e., requiring $\nabla V(z)$), such as the Reverse Diffusion Monte Carlo (RDMC, Huang et al. 2024a), the Recursive Score Diffusion-based Monte Carlo (RS-DMC, Huang et al. 2024b), and the stochastic localization via iterative posterior sampling (SLIPS, Grenioux et al. 2024). We develop a conjugate gradient algorithm to solve the proposed LV regularization model, which integrates with a recent MC method requiring only $V(z)$, called Zeroth Order Diffusion-Monte Carlo (ZOD-MC, He et al. 2024). This approach saves computational costs by lowering the order of queries. Our main contributions can be summarized as follows, while the diagram of our method is shown in Figure 1.

1. We propose an adversarial attack approach on DDS, which injects a perturbation into the variable value during its computation and transmission.

2. We propose a minimization model of the LV regularized potential function, which can tolerate the adversarial perturbations injected in the variable value. It forms an Adversarial Defense method for DDS (**ADDDS**).

3. We develop a conjugate gradient algorithm to solve the proposed ADDDS model, which has theoretical guarantees on convergence. It integrates with the zeroth-order method ZOD-MC, which saves computational costs.

2 Preliminaries and Related Works

We introduce some preliminaries and related works on DDS, adversarial attacks, and TV.

2.1 Denoising Diffusion Sampling

As introduced in Section 1, a diffusion model in this field aims to generate a new sample $\bar{X}_t$ that subjects to the same distribution p as that of a training sample X_t. Suppose the dimensionality of X_t is d. A common methodology is to exploit the Ornstein-Uhlenbeck (OU) process:

$$\mathrm{d}X_t = -X_t\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t, \quad X_0 \sim p, \tag{1}$$

where B_t is the Brownian noise for the forward process that constructs the training sample X_t. X_t can be decomposed into two components: $X_t = e^{-t}X_0 + \sqrt{1-e^{-2t}}Z$, where $(X_0, Z) \sim p \otimes \gamma^d$ and γ^d denotes the d-dimensional standard Gaussian distribution. We denote the distribution of X_t by $p_t := \mathrm{Law}(X_t)$, $\forall t \geqslant 0$. When the terminal time T of process (1) is large enough, p_T is close

to γ^d. Then the backward process $\{\bar{X}_t\}_{0\leqslant t\leqslant T}$ can be constructed by reversing the OU process (1) from time T to 0, which means that $\mathrm{Law}(\bar{X}_t) = \mathrm{Law}(X_{T-t})$ for all $t \in [0, T]$. This leads to the following denoising diffusion process:

$$\mathrm{d}\bar{X}_t = (\bar{X}_t + 2\nabla \log p_{T-t}(\bar{X}_t))\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\bar{B}_t, \quad \bar{X}_0 \sim \gamma^d, \quad 0 \leqslant t \leqslant T, \tag{2}$$

where $\bar{B}_t$ is another Brownian noise and independent of B_t. $\nabla \log p_t$ is called the *score function* for p_t, which is not explicitly known. A widely-used method is to learn the score function by denoising score matching methods with MC (Ho et al., 2020; Sohl-Dickstein et al., 2015; Huang et al., 2021):

$$s(t, x) := \frac{1}{n(t)} \sum_{i=1}^{n(t)} \frac{e^{-t} z_{t,i} - x}{1 - e^{-2t}}, \tag{3}$$

where $\{z_{t,i}\}_{i=1}^{n(t)}$ are the independent samples from MC at time t. If the learned score $s(t, x)$ satisfies the following error bound

$$\mathbb{E}_{x\sim p_t}[\|s(t, x) - \nabla \log p_t(x)\|_2^2] \leqslant \varepsilon_{score}^2, \quad \forall\, 0 \leqslant t \leqslant T, \tag{4}$$

non-asymptotic convergence for diffusion model (2) can be guaranteed in theory (Benton et al., 2024; Chen et al., 2023; 2022). $\varepsilon_{score} \geqslant 0$ denotes a preset error tolerance. In practice, a common method to simulate (2) is to use discretizations like the Euler Maruyama or an exponential integrator. This can be vulnerable to adversarial attacks or sampling errors, which will be illustrated in Section 3.1. A recent method ZOD-MC develops a rejection sampling scheme that requires only a zeroth-order query to the potential value $V(z_{t,i})$ at each $z_{t,i}$ (He et al., 2024). The crucial stage of ZOD-MC is to solve for the minimum of V, which just needs to be implemented for only once for all the MC samples $\{z_{t,i}\}$ during the entire DDS process:

$$\min_{x\in\mathbb{R}^d} V(x). \tag{5}$$

This is also the crucial part that we can exploit to develop an adversarial defense strategy, as illustrated in Section 3.2.

2.2 Adversarial Attacks

An adversarial attack deliberately injects a small perturbation, which can be hardly recognized, into the given data, in order to make a learning system give incorrect or distorted results. While the training of a learning system aims to minimize a given loss function, the adversarial attack aims to maximize this loss function with a perturbed input $(x + \delta)$:

$$\max_{\|\delta\|_{\mathfrak{p}} \leqslant \epsilon} \mathcal{L}(x + \delta), \tag{6}$$

where $\|\delta\|_{\mathfrak{p}} \leqslant \epsilon$ means that the $\ell_{\mathfrak{p}}$ norm of the perturbation δ should be no larger than a preset *budget* $\epsilon \geqslant 0$. $\mathcal{L}$ is the loss function for training, which considers both unsupervised learning and supervised learning (the dependent variable y can be embedded in $\mathcal{L}$ and thus omitted in this setting).

A fundamental attack approach is the Fast Gradient Sign Method (FGSM, Goodfellow et al. 2015), which injects a small perturbation in the same direction of the gradient sign:

$$\tilde{x} = x + \epsilon' \cdot \mathrm{sign}(\nabla_x \mathcal{L}(x)), \tag{7}$$

where $0 \leqslant \epsilon' \leqslant \epsilon / \|\mathrm{sign}(\nabla_x \mathcal{L}(x))\|_{\mathfrak{p}}$.

Besides FGSM, the Projected Gradient Descent (PGD, Madry et al. 2018) is a multi-iteration attack strategy that aims to achieve a stronger attack of "first-order adversary":

$$\tilde{x}_{(l+1)} = \mathrm{proj}_{\mathcal{B}_{\mathfrak{p}}[x;\epsilon]}(\tilde{x}_{(l)} + \eta \cdot \mathrm{sign}(\nabla_x \mathcal{L}(x))), \tag{8}$$

where $\eta > 0$ denotes the step size for a single PGD iteration, and $\mathrm{proj}_{\mathcal{B}_{\mathfrak{p}}[x;\epsilon]}$ denotes the projection operator onto the closed neighborhood of x with a radius of ϵ. A common defense approach for adversarial attack is the adversarial training (Rosaria et al., 2025; Zhang et al., 2022), which takes the adversarial samples into an improved training scheme. However, it is a training-time defense that modifies the model parameters, which is different from the runtime defense ADDDS that operates during the inference/sampling process to purify the perturbed trajectory.

2.3 Total Variation

TV (Rudin et al., 1992) is a traditional instrument that aggregates the variation of a function $f(x)$ while the variable x traverses the domain Ω. It is widely-recognized that many kinds of noise and

other unwanted fine-scale details can be captured by TV. Suppose $\Omega \subseteq \mathbb{R}^d$ is an open set and let $L^1(\Omega)$ be the corresponding Lebesgue-integrable function space. Then the TV of $f \in L^1(\Omega)$ can be defined as (Chan et al., 2006):

$$\int_\Omega |\nabla f| := \sup\{\int_\Omega f(x)\mathrm{div}g(x)\,\mathrm{d}x : \ g \in C_c^1(\Omega, \mathbb{R}^d), \|g\|_{L^\infty(\Omega)} \leqslant 1\}, \tag{9}$$

where g works as a probe function with its divergence $\mathrm{div}g$, which has a compact support contained in Ω and an essential supremum no larger than 1. TV calculates the local variations (LV) of f across all dimensions of x, and then accumulates these LVs over the entire domain Ω. $|\nabla f|$ is symbolic if f is non-differentiable, but it is exactly the ℓ_2 **norm of the gradient** ∇f **if** f **is differentiable**. In practice, the literature usually considers the functions with bounded variation (BV) $\{f \in L^1(\Omega) : \int_\Omega |\nabla f| < \infty\}$.

Based on the essence of TV, the TV-ℓ_1 model (Rudin et al., 1992) is proposed to recover the main component of f and reduce noise and other unwanted fine-scale details simultaneously:

$$\inf_{\hat{f} \in L^2(\Omega)} \left\{ \int_\Omega |\nabla \hat{f}| + \lambda \int_\Omega (f - \hat{f})^2 \,\mathrm{d}x \right\}, \tag{10}$$

where the second term measures the approximation of the recovery $\hat{f} \in L^2(\Omega)$ to the target function f, and $\lambda \geqslant 0$ is the approximation hyperparameter. The first term can be considered as a regularization term that tolerates the TV of $\hat{f}$.

3 Methodology

DDS is an effective method for new sample generation under an ideal and pure environment where interference and perturbations are absent. However, it can be vulnerable to adversarial attacks or sampling errors. In this section, we propose a complete set of adversarial attack and defense methodology for DDS, which is briefly demonstrated in Figure 1.

3.1 Attack Side for Denoising Diffusion Sampling

A vulnerable stage of DDS lies in the following update via an exponential integrator:

$$\begin{aligned} &\text{generate } \xi_k \sim \gamma^d \text{ such that } \xi_k \text{ is independent of } \xi_0, \cdots, \xi_{k-1}, \\ &x_{k+1} \leftarrow e^{t_{k+1}-t_k} x_k + 2(e^{t_{k+1}-t_k} - 1) s(T - t_k, x_k) + \sqrt{e^{2(t_{k+1}-t_k)} - 1}\,\xi_k, \end{aligned} \tag{11}$$

where x_{k+1} **is a simplified notation of** $x_{t_{k+1}}$, which is the variable value at the $(k+1)$-th timestamp. $s(T - t_k, x_k)$ is the estimated score function value at time $(T - t_k)$ and variable value x_k, which corresponds to the term $\nabla \log p_{T-t_k}(\bar{X}_{t_k})$ in (2). The generated ξ_k serves as the Brownian motion from t_k to t_{k+1}. In a decentralized setting (Lian et al., 2017; Koloskova et al., 2020; Cyffers & Bellet, 2022; Cyffers et al., 2023), x_k should be transmitted from the $(k-1)$-th user to the k-th user, which is exposed to intentional or unintentional perturbations. Specifically, x_k can be perturbed due to channel interference, which consequently affects the score estimation $s(T - t_k, x_k)$. Besides, ξ_k may involve a sampling error. All these perturbations can be summarized as the following update without loss of generality:

$$x_{k+1} \leftarrow e^{t_{k+1}-t_k} x_k + 2(e^{t_{k+1}-t_k} - 1) s(T - t_k, x_k) + \sqrt{e^{2(t_{k+1}-t_k)} - 1}\,\xi_k + \underline{\delta_k}, \tag{12}$$

where δ_k **represents the overall perturbation in this update.** As explained in Section 2.2, if $\|\delta_k\|_{\mathfrak{p}}$ is small enough, such a perturbation can be hardly recognized but can sabotage the generation of x_{k+1}. Worse still, this adversarial attack can happen at each of the hundreds or even thousands of timestamps $\{t_k\}_{k=0}^N$ in an entire DDS process, leading to the bad performance in Section 4 and Appendix B.

As for the construction of δ_k**, we find that even a Gaussian perturbation with a small standard deviation (STD) can significantly diminish the performance of DDS. Since Gaussian perturbations are prevalent in various real-world applications, this vulnerability is rather significant.**

Besides, FGSM and PGD in (7) and (8) can also be adopted to construct δ_k. To do this, we design the following loss function:

$$\mathcal{L}_{T-t_k}(x) := \frac{1}{n(T-t_k)} \sum_{i=1}^{n(T-t_k)} \left\| \frac{e^{-(T-t_k)} z_{T-t_k,i} - x}{1-e^{-2(T-t_k)}} \right\|_2^2 . \tag{13}$$

It averages the squared ℓ_2 norms of the components of the score function at time $(T-t_k)$. By this means, the constructed perturbation δ_k aims to maximize the magnitude of the score function, in order to sabotage the second term in (12).

3.2 Defense Side for Denoising Diffusion Sampling

The defense side for DDS aims to tolerate the adversarial perturbations $\{\delta_k\}_{k=0}^N$ by all means. We find that the crucial technique of defense for DDS lies in the computation of the potential function $V(x)$. Since the variable $\tilde{x} = x + \delta$ has been attacked by a perturbation δ, the computation of $V(\tilde{x})$ should also consider this interference. As introduced in (5), the crucial stage of the zeroth order MC method ZOD-MC is to minimize $V(x)$. Therefore, in the presence of adversarial perturbations, we propose to minimize $(V(x) + \lambda|\nabla V(x)|)$:

$$\min_{x\in\mathbb{R}^d} \{V(x) + \lambda|\nabla V(x)|\}, \tag{14}$$

where $\lambda|\nabla V(x)|$ is an LV regularization term with strength $\lambda \geqslant 0$. First, we provide a theoretical result on the equivalence between (14) and (5), which enables the rejection sampling scheme in (He et al., 2024).

Theorem 1. *If $V(x)$ is differentiable, then (14) and (5) have the same solution set.*

The proof is provided in Appendix A.1. There are generally no closed-form solutions to both (14) and (5), thus iterative algorithms should be developed to solve them. Although they have the same solution set, their iteration paths are substantially different. The iteration path of (14) simultaneously minimizes $V(x)$ and tolerates the perturbation of V induced by that of x, but the iteration path of (5) only minimizes $V(x)$. Since $V(x)$ is generally a nonconvex function, probably with complicated geometrical structures, a standard gradient-type algorithm can only find a critical point $x^\bullet$ of V. Due to the different iteration paths between (14) and (5), they also lead to different critical points. This can be demonstrated by the matched oracle complexity experiments (see the experimental settings in Appendix B.1) in Table 1. While the critical points of (5) satisfy $|\nabla V(x^\bullet_{org})| = 0$, the critical points of (14) yield nonzero LVs $|\nabla V(x^\bullet_{ADDDS})| > 0$. This is because the critical points of (14) are subject to

$$\nabla V(x^\bullet_{ADDDS}) \in -\lambda \cdot \partial|\nabla V(x^\bullet_{ADDDS})|, \quad \partial|\nabla V(x^\bullet_{ADDDS})| \text{ defined by Definition 2}, \tag{15}$$

which may lead to $|\nabla V(x^\bullet_{ADDDS})| > 0$. This mechanism allows (14) to tolerate the small LV induced by the perturbation δ. Although it is obvious that $|\nabla V(x^*)| = 0$ with the ground-truth minimizer x^*, the perturbed version should probably be $|\nabla V(x^* + \delta)| > 0$. This situation can be adapted to by $|\nabla V(x^\bullet_{ADDDS})| > 0 = |\nabla V(x^\bullet_{org})|$. The same reason applies to why $V(x^\bullet_{ADDDS}) > V(x^\bullet_{org})$, which adapts to the situation $V(x^* + \delta) > V(x^*)$.

On the other hand, the LV model (14) is substantially different from the TV model (10) (Rudin et al., 1992; Chen et al., 2006; Chan et al., 2006; Lai & Wang, 2024; Wang et al., 2025). The former is a local model that aims to retrieve an optimal argument x^*, while the latter is a global model that aims to retrieve an entire optimal function f^* in a functional space (e.g., $L^2(\Omega)$). Therefore, the existing solving algorithms (Yang et al., 2009; Barbero & Sra, 2011; Bai et al., 2016) for TV models cannot be directly applied to the LV model (14), and we need to develop a solving algorithm. Since $|\nabla V(x)|$ is non-differentiable, we need to use the Fréchet subdifferential $\partial f(x)$.

Definition 2 (The Fréchet Subdifferential)**.**

$$\partial f(x) := \left\{ z\in\mathbb{R}^d : \liminf_{\substack{y\to x \\ y\neq x}} \frac{f(y)-f(x)-z^\top(y-x)}{\|y-x\|_2} \geqslant 0 \right\} . \tag{16}$$

Table 1: Critical points $x^\bullet$, potentials $V(x^\bullet)$, and LVs $|\nabla V(x^\bullet)|$ solved from (5) and (14). While (5) yields zero LVs, (14) yields nonzero LVs that can tolerate the perturbations induced by the attacked variables $\tilde{x}$.

Oracle Complexity	Model	Critical Point $x^\bullet$	$V(x^\bullet)$	$\lvert\nabla V(x^\bullet)\rvert$
200	(5)	$(-5.96e-08, -2.98e-08)$	3.9966	$5.9605e-08$
	(14)	$(2.68e-04, 3.16e-04)$	4.0512	0.0893
3200	(5)	$(-1.19e-07, -5.96e-08)$	3.9966	$1.1921e-07$
	(14)	$(6.68e-05, 1.34e-04)$	4.0512	0.0893
6200	(5)	$(5.96e-08, -5.96e-08)$	3.9966	$2.3842e-07$
	(14)	$(1.68e-04, 8.38e-05)$	4.0510	0.0887
9200	(5)	$(5.96e-08, -5.96e-08)$	3.9966	$2.3842e-07$
	(14)	$(-1.04e-04, -2.08e-04)$	4.0886	0.1288

In fact, $|\nabla V(x)|$ is non-differentiable if and only if $\nabla V(x) = 0$, where we can directly set the corresponding subdifferential $\partial|\nabla V(x)| = 0$ since it satisfies (16). Thus the complete form of $\partial|\nabla V(x)|$ can be directly calculated as follows, which works as an ordinary gradient:

$$\partial|\nabla V(x)| \leftarrow \begin{cases} \frac{\nabla(1_d^\top \nabla V(x))}{|\nabla V(x)|} & \text{if} \quad \nabla V(x) \neq 0, \\ 0 & \text{if} \quad \nabla V(x) = 0. \end{cases} \tag{17}$$

We treat the subgradient of the entire objective function

$$W(x) := V(x) + \lambda|\nabla V(x)|, \quad \nabla W(x) = \nabla V(x) + \lambda\partial|\nabla V(x)| \tag{18}$$

as a whole, instead of treating $\nabla V(x)$ and $\lambda\partial|\nabla V(x)|$ separately. The reason is that both $\nabla V(x)$ and $\lambda\partial|\nabla V(x)|$ (based on Eq. 17) can be simultaneously calculated via the `autograd` scheme in deep learning architectures like Pytorch[1], hence it is more efficient to calculate $\nabla W(x)$ as a whole.

Next, we develop a conjugate gradient algorithm based on the Crowder-Wolfe scheme (Hestenes et al., 1952; Crowder & Wolfe, 1972; Gilbert & Nocedal, 1992) to solve (14) as follows.

$$x^{(l+1)}{=}x^{(l)}{+}\alpha^{(l)}\mathfrak{d}^{(l)}, \mathfrak{d}^{(l+1)}{=}\left|\frac{(\nabla W(x^{(l+1)}){-}\nabla W(x^{(l)}))^\top \nabla W(x^{(l+1)})}{(\nabla W(x^{(l+1)}){-}\nabla W(x^{(l)}))^\top \mathfrak{d}^{(l)}}\right| \mathfrak{d}^{(l)}{-}\nabla W(x^{(l+1)}), \tag{19}$$

where $\alpha^{(l)}$ is a step size that satisfies the Wolfe conditions

$$W(x^{(l+1)}){\leqslant}W(x^{(l)}){+}c_1\alpha^{(l)}(\nabla W(x^{(l)}))^\top\mathfrak{d}^{(l)}, (\nabla W(x^{(l+1)}))^\top\mathfrak{d}^{(l)}{\geqslant}c_2(\nabla W(x^{(l)}))^\top\mathfrak{d}^{(l)}. \tag{20}$$

In (20), $0 < c_1 < c_2 < 1$ are two constants, and $\mathfrak{d}^{(l)}$ represents the update direction at the l-th iteration. This algorithm guarantees convergence (Nocedal & Wright, 2006), which is detailed in Appendix A.2. Once a critical point $x^\bullet$ has been solved, the corresponding score function $V^\bullet := V(x^\bullet)$ can be used in the following recovering stage. In addition, ADDDS achieves the following adversarial robustness guarantee.

Theorem 3 (Adversarial Robustness Guarantee)**.** *For any $x \in \mathbb{R}^d$, if the potential function $V(x)$ is $L_{x,\epsilon}$-smooth ($L_{x,\epsilon} > 0$) in the neighborhood $\mathcal{B}_2[x;\epsilon]$, then $W(x)$ can serve as a surrogate upper bound for the adversarial loss w.r.t. $V(x)$ for any $\lambda \geqslant \epsilon$:*

$$\max_{\|\delta\|_2 \leqslant \epsilon} V(x+\delta) \leqslant W(x) + \frac{L_{x,\epsilon}}{2}\epsilon^2 \leqslant W(x) + \frac{L_{x,\epsilon}}{2}\lambda^2. \tag{21}$$

The definition of $L_{x,\epsilon}$-smoothness can be found in Appendix A.3. In fact, $L_{x,\epsilon}$-smoothness is easy to satisfy: a sufficient condition is that V is twice differentiable in $\mathcal{B}_2[x;\epsilon]$, which is the case in this work. The proof is provided in Appendix A.3. To see the working mechanism of Theorem 3, we recall that a benign DDS method should minimize $V(x)$ as (5). Hence a white-box adversarial attack strategy should maximize $V(x+\delta)$ by injecting a perturbation δ. Then Theorem 3 indicates that this adversarial loss shall not exceed $W(x)$ plus a small value. Moreover, this verifies that our

[1] https://pytorch.org

defense strategy is to minimize the worst-case scenario, which formulates the following minimax robust optimization problem:

$$\min_{x\in\mathbb{R}^d}\max_{\|\delta\|_2\leqslant\epsilon} V(x+\delta) \leqslant \min_{x\in\mathbb{R}^d}\left\{W(x)+\frac{L_{x,\epsilon}}{2}\epsilon^2\right\}. \tag{22}$$

This result justifies our motivation of minimizing $W(x)$ in (14) and formally connects the proposed LV regularization $\lambda\|\nabla V(x)\|_2$ to the rigorous framework of adversarial robustness. The condition $\lambda \geqslant \epsilon$ means that the regularization strength should be no less than the perturbation budget in order to achieve the above adversarial robustness.

3.3 Recovering Stage for Denoising Diffusion Sampling

Our defense strategy integrates with the zeroth-order MC method ZOD-MC (He et al., 2024), which only needs to query $V(z)$ (instead of $\nabla V(z)$ or even high-order derivatives) for each MC variable z. This is more computationally efficient than the first-order methods because there can be hundreds of timestamps and thousands of MC variables at each timestamp in a DDS process. Specifically, given a perturbed variable $\tilde{x} \in \mathbb{R}^d$, a sample $(\xi, u) \sim \gamma^d \otimes U[0,1]$ is generated to compute an intermediate variable $\tilde{z} := e^t\tilde{x} + \sqrt{e^{2t}-1}\xi$. If $u \leqslant e^{V^\bullet - V(\tilde{z})}$, then this $\tilde{z}$ is feasible for MC score estimation, otherwise $\tilde{z}$ will be rejected and resampled. Note that if $V(\tilde{z}) < V^\bullet$, then we accept $\tilde{z}$ with probability 1 and update $V^\bullet \leftarrow V(\tilde{z})$, which keeps $V^\bullet$ as the current minimum. In this way, we essentially use a standard rejection sampler that accepts $\tilde{z}$ with probability $\frac{p(\tilde{z})}{M}$, where $p(\tilde{z}) = e^{-V(\tilde{z})}$ and the global bound $M = e^{-V^*}$ with V^* being the ground-truth minimum of the potential function. After generating sufficient feasible samples $\{\tilde{z}_{t,i}\}_{i=1}^{n(t)}$, the MC score estimation with the perturbed $\tilde{x}$ can be implemented by

$$\tilde{s}(t,\tilde{x}) := \frac{1}{n(t)}\sum_{i=1}^{n(t)}\frac{e^{-t}\tilde{z}_{t,i}-\tilde{x}}{1-e^{-2t}}. \tag{23}$$

Last, the following perturbed backward process can be used to recover a new sample process $\{\tilde{x}_k\}_{k=0}^N$:

$$\tilde{x}_{k+1} \leftarrow e^{t_{k+1}-t_k}\tilde{x}_k + 2(e^{t_{k+1}-t_k}-1)\tilde{s}(T-t_k,\tilde{x}_k) + \sqrt{e^{2(t_{k+1}-t_k)}-1}\xi_k + \delta_k. \tag{24}$$

(24) differs from (12) in the score estimation: (24) uses $\tilde{s}$ in (23) with perturbation-adjusted $\{\tilde{z}_{t,i}\}_{i=1}^{n(t)}$, while (12) uses s in (3) with non-perturbation-adjusted $\{z_{t,i}\}$. Hence (24) is more robust than (12) to adversarial attacks.

3.4 Extension to High-dimensional Settings

Adversarial robustness for image classification (Goodfellow et al., 2015; Madry et al., 2018) is an important topic involving a much higher dimensionality than the basic formulation by the related works in Section 2. Hence we further extend the proposed ADDDS to a high-dimensional setting for the image classification task. In this task, the sample x is formulated as the parameters for a Bayesian neural network (BNN, Neal 2012; Song & Kingma 2021). Given a training data set $\mathcal{D} = \{(u_i, y_i)\}$ with u_i and y_i being the i-th image and its ground-truth label vector, the potential function can be defined by the following posterior energy function:

$$V(x) = -\log p(x|\mathcal{D}), \quad p(x|\mathcal{D}) \propto p(\mathcal{D}|x)p(x). \tag{25}$$

We can use a Gaussian prior with the standard deviation (STD) σ_{prior}, the cross-entropy classifier, and the BNN to construct a surrogate of $V(x)$. Then the corresponding score function (Song et al., 2021) is:

$$\log p(x) = -\frac{\|x\|_2^2}{2\sigma_{prior}^2}, \quad \log p(\mathcal{D}|x) = \sum_{(u_i,y_i)\in\mathcal{D}} y_i^\top \log \hat{y}_i(u_i;x), \tag{26}$$

$$s(x) = \nabla \log p(x|\mathcal{D}) = \nabla \log p(\mathcal{D}|x) + \nabla \log p(x), \tag{27}$$

where $\hat{y}_i(u_i;x)$ denotes the estimated label vector (output) of the image u_i through the BNN with parameter x. Based on the above formulation, we can use the OU process in (1), the score estimation

in (3), and the DDS process with adversarial attack in (12). In the defense side, we can insert (26) into $V(x)$ of (25) and use the same ADDDS procedure with (14). The fundamental difference between the low-dimensional setting in Section 2 and our high-dimensional setting is that the former uses the existing data to generate new data, but the latter uses the existing data to generate the parameters of BNN. After the DDS process (either with or without the attack or defense scheme), the generated final parameters x_N can be used in BNN to perform classification tasks.

4 Experimental Results

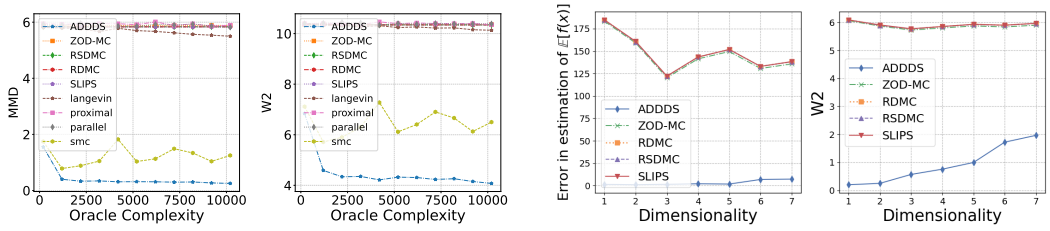
Our experimental setup extends the standard benchmark for DDS methods outlined in (He et al., 2024) to include more adversarial attack schemes and more baseline methods. Specifically, the plain Gaussian perturbations, FGSM (Goodfellow et al., 2015), and PGD^{10} (PGD with 10-step attacks, Madry et al. 2018) are used as three attack schemes. ZOD-MC (He et al., 2024), RSDMC (Huang et al., 2024b), RDMC (Huang et al., 2024a), SLIPS (Grenioux et al., 2024), Langevin (Huang et al., 2024a;b), the proximal sampler (Liang & Chen, 2023), sequential Monte Carlo (Del Moral et al., 2006). a parallel tempering approach with MALA proposals (Lee & Shen, 2023) are used as baseline methods. The computing device has twelve Intel(R) Xeon(R) Gold 6248R CPUs, 92-GB RAM, and an Nvidia A100-PCIE-40GB GPU. **The main part of PGD^{10}-attack experiments are presented in the main text, while the rest experiments and detailed experimental settings are presented in Appendix B. Additional experiments including high-dimensional settings, ablation studies, intrinsic defense schemes, adaptive white-box attack, computational resource requirement, acceptance rate, and score estimation error are presented in Appendix C.**

4.1 Gaussian Mixture Models

This experiment uses a 2D Gaussian mixture model (GMM) with unbalanced modes and non-isotropic variances, which leads to a highly asymmetrical and multi-modal problem. Model details are provided in Appendix B.1.

Oracle Complexity and Dimensionality Dependence. In the oracle complexity scenario, different methods are tested on different oracle complexities (i.e., total number of queries to $V(z)$ or $\nabla V(z)$). While ZOD-MC and the proposed ADDDS query $V(z)$, other baselines query $\nabla V(z)$, which gives more advantage to the latter. Figures 2(a) and B2 present the sampling accuracies w.r.t. oracle complexity for different methods. Both Maximum Mean Discrepancy (MMD) and 2-Wasserstein distance (W_2) are taken as evaluation metrics (the lower the better), and ADDDS outperforms other baselines in most cases. Figures 3, B4, B5, B6, and B7 show the generated samples of different methods with various oracle complexities, which indicate that ADDDS recovers the best patterns on the 4-mode GMM among all the baselines across all the oracle complexities. Moreover, ADDDS requires only a few minutes of computational time (see Figure B1), which is reasonable considering its effective defense.

In the dimensionality dependence scenario, the statistics on the generated samples and W_2 scores of different methods w.r.t. different dimensionalities are tested, which are shown in Figures 2(b) and B3. ADDDS significantly outperforms other baselines across different dimensionalities (the plots of other baselines mutually overlap), which indicates that ADDDS scales well with dimensionality increase.



(a) Sampling accuracy w.r.t. oracle complexity (b) Sampling accuracy w.r.t. dimensionality

Figure 2: Sampling accuracies of different methods for Gaussian mixtures under PGD^{10} attacks. A lower score indicates a better performance.

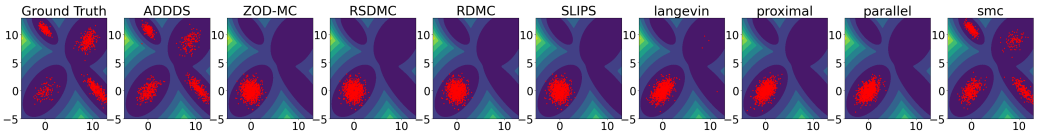


Figure 3: Samples generated by different methods from an asymmetric and unbalanced Gaussian mixture under PGD^{10} attacks (oracle complexity= 1200).

Robustness Against Mode Separation. In this scenario, the means of all the modes are scaled by a factor such that a mode is located at $(0, R)$, which separates different modes more distinctly and worsens the isoperimetric properties of the target distribution. Figures 4 and B8 show the results of MMD, W_2, and mass on center mode w.r.t. the radius R for different methods. ADDDS performs more robustly than other baselines as R increases, which indicates that it is effective in defending adversarial attacks under poor conditions. Moreover, generated samples in Figures 5, B9, B10, B11, and B12 show that ADDDS recovers the best patterns among all the baselines across all the radius settings.

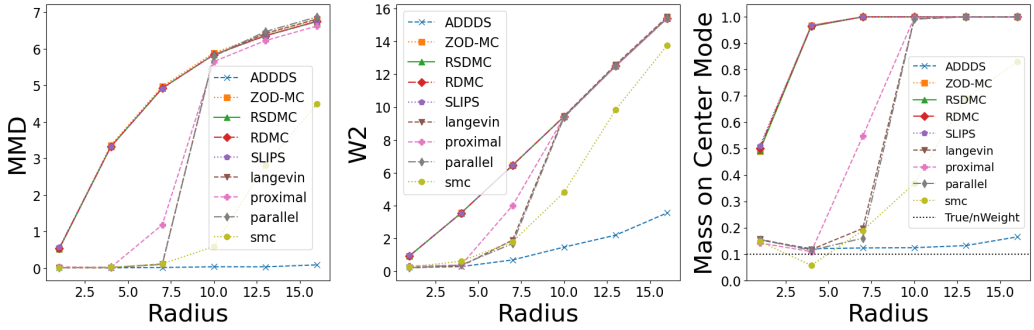


Figure 4: Sampling accuracies of different methods w.r.t. mode separation for Gaussian mixtures under PGD^{10} attacks. A lower score indicates a better performance.

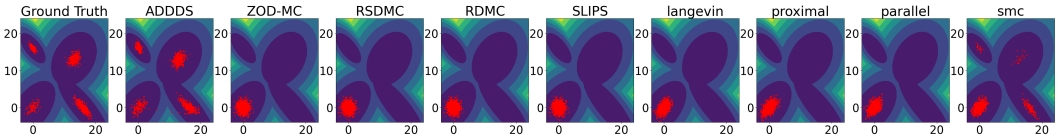


Figure 5: Samples generated by different methods for a Gaussian mixture with radius $R = 16$ under PGD^{10} attacks.

Discontinuous Potential. In this scenario, a discontinous potential function $(V(x) + U(x))$ is considered, where $U(x) = 8\lfloor \|x\|_2 \rfloor \mathbb{I}_{\{5<\|x\|_2<11\}}$ and $\mathbb{I}$ denotes the indicator function that takes value 1 in the set $\{5 < \|x\|_2 < 11\}$ and value 0 otherwise. $\lfloor \cdot \rfloor$ denotes the floor operator that yields an integer. This discontinous potential strengthens the potential barrier and increases the sampling difficulty. Results in Figures 6 and B13 show that ADDDS outperforms other baselines in both MMD and W_2 across all the oracle complexities. Moreover, generated samples in Figures 7, B14, B15, B16, and B17 show that ADDDS recovers the best patterns among all the baselines across all the oracle complexities.

4.2 Müller-Brown Potential

The Müller-Brown potential is a widely-used potential energy surface model (Müller & Brown, 1979; Sipka et al., 2023) for molecular dynamics. It is a highly nonlinear potential that has three modes, although it is the sum of four exponentials. To facilitate 2D visualization, a balanced version (Li & Tao, 2024; He et al., 2024) is used where one mode is centered near the origin. Generated samples by different methods are shown in Figures 8 and B18. Since there is no ground-truth

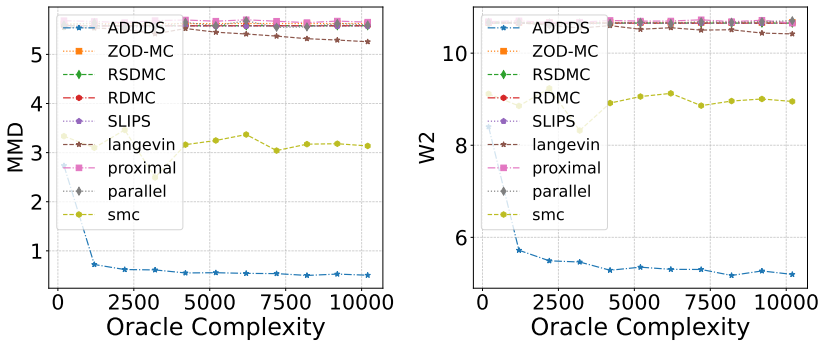


Figure 6: Sampling accuracies of different methods w.r.t. oracle complexity for a discontinuous potential under PGD^{10} attacks. A lower score indicates a better performance.

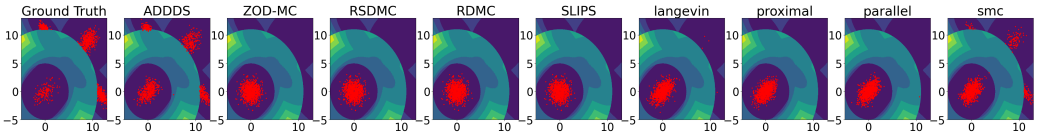


Figure 7: Samples generated by different methods from a discontinuous potential under PGD^{10} attacks (oracle complexity$=1200$).

sampling for this potential, the level curves of the potential can be used to evaluate the performance. ADDDS recovers the best patterns for the three modes among all the baselines against all the attack methods.

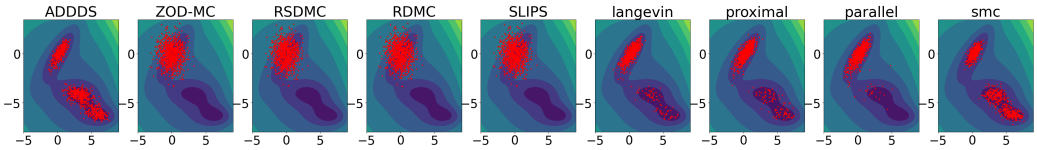


Figure 8: Samples generated by different methods from the Müller-Brown potential under PGD^{10} attacks (oracle complexity$=1100$).

5 Conclusion

Denoising diffusion sampling (DDS) is a popular approach to generate new samples with the same distribution as some given samples. However, this approach is vulnerable to adversarial attacks, especially in the transmission stage of a decentralized setting. To address this challenge, we propose a complete set of adversarial attack and defense methodology. In the attack side, we find that even a simple Gaussian perturbation can sabotage the generation of new correct samples, let alone other advanced attack strategies. In the defense side, we develop a joint minimization model of the potential function and the local variation (LV), which can tolerate the injected adversarial perturbations. Moreover, we develop a conjugate gradient algorithm to solve the defense model. It integrates with a zeroth-order diffusion Monte Carlo method and saves computational cost.

Extensive experiments show that the proposed attack method successfully sabotages the generation of new samples against different DDS methods. On the other side, the proposed defense method successfully recovers the generated samples from the ground-truth distribution against the adversarial perturbations. In summary, theses findings not only expose a vulnerable stage in DDS, but also characterize such an adversarial perturbation by the LV. Future works may lie in exploring more vulnerable parts in different diffusion models and extend the proposed LV methodology to tackle more types of adversarial perturbations.

REFERENCES

Minru Bai, Xiongjun Zhang, and Qianqian Shao. Adaptive correction procedure for tvl1 image deblurring under impulse noise. *Inverse Problems*, 32(8):085004, jun 2016.

Alvaro Barbero and Suvrit Sra. Fast newton-type methods for total variation regularization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 313–320, 2011.

Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. PODS'20, pp. 49–62, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371087.

Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2), January 2022. ISSN 0004-5411.

Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Nearly d-linear convergence bounds for diffusion models via stochastic localization. In *The Twelfth International Conference on Learning Representations*, 2024.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.

Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 3544–3557. Curran Associates, Inc., 2021.

José Carrillo and Jakub Skrzeczkowski. Convergence and stability results for the particle system in the stein gradient descent method. *Mathematics of Computation*, 94(354):1793–1814, 2025.

T. Chan, S. Esedoglu, F. Park, and A. Yip. *Total Variation Image Restoration: Overview and Recent Developments*, pp. 17–31. Springer US, Boston, MA, 2006.

Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved analysis of score-based generative modeling: User-friendly bounds under minimal smoothness assumptions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4735–4763. PMLR, 23–29 Jul 2023.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Tenth International Conference on Learning Representations*, 2022.

Terrence Chen, Wotao Yin, Xiang Sean Zhou, Dorin Comaniciu, and Thomas S. Huang. Total variation models for variable lighting face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1519–1524, 2006.

Harlan Crowder and Philip Wolfe. Linear convergence of the conjugate gradient method. *IBM Journal of Research and Development*, 16(4):431–433, 1972.

Edwige Cyffers and Aurélien Bellet. Privacy amplification by decentralization. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 5334–5353. PMLR, 28–30 Mar 2022.

Edwige Cyffers, Aurélien Bellet, and Debabrota Basu. From noisy fixed-point iterations to private admm for centralized and federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 6683–6711. PMLR, 2023.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Jean Charles Gilbert and Jorge Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, 2(1):21–42, 1992.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR*, 2015.

Louis Grenioux, Maxence Noble, Marylou Gabrié, and Alain Durmus. Stochastic localization via iterative posterior sampling. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Ye He, Krishnakumar Balasubramanian, and Murat A Erdogdu. On the ergodicity, bias and asymptotic normality of randomized midpoint sampling method. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7366–7376. Curran Associates, Inc., 2020.

Ye He, Kevin Rojas, and Molei Tao. Zeroth-order sampling methods for non-log-concave distributions: Alleviating metastability by denoising diffusion. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 71122–71161. Curran Associates, Inc., 2024.

Magnus R Hestenes, Eduard Stiefel, et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Xunpeng Huang, Hanze Dong, Yifan Hao, Yi-An Ma, and Tong Zhang. Score-based generative modeling through stochastic differential equations. In *The Ninth International Conference on Learning Representations*, 2021.

Xunpeng Huang, Hanze Dong, Yifan Hao, Yi-An Ma, and Tong Zhang. Reverse diffusion monte carlo. In *The Twelfth International Conference on Learning Representations*, 2024a.

Xunpeng Huang, Difan Zou, Hanze Dong, Yi-An Ma, and Tong Zhang. Faster sampling without isoperimetry via diffusion-based monte carlo. In Shipra Agrawal and Aaron Roth (eds.), *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pp. 2438–2493. PMLR, 30 Jun–03 Jul 2024b.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized SGD with changing topology and local updates. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5381–5393. PMLR, 13–18 Jul 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Zhao-Rong Lai and Weiwen Wang. Invariant risk minimization is a total variation model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 25913–25935. PMLR, 21–27 Jul 2024.

Holden Lee and Zeyu Shen. Improved bound for mixing time of parallel tempering, 2023.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. Towards faster non-asymptotic convergence for diffusion-based generative models. In *The Twelfth International Conference on Learning Representations*, 2024.

Xingjie Helen Li and Molei Tao. Automated construction of effective potential via algorithmic implicit bias. *Journal of Computational Physics*, 514:113206, 2024. ISSN 0021-9991.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jiaming Liang and Yongxin Chen. A proximal algorithm for sampling, 2023. URL https://arxiv.org/abs/2202.13975.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Klaus Müller and Leo D Brown. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoretica chimica acta*, 53(1):75–93, 1979.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

Briglia Maria Rosaria, Mujtaba Hussain Mirza, Giuseppe Lisanti, and Iacopo Masi. What is adversarial training for diffusion models?, 2025.

Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

Adil Salim, Lukang Sun, and Peter Richtarik. A convergence theory for SVGD in the population limit under talagrand's inequality t1. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19139–19152. PMLR, 17–23 Jul 2022.

Martin Sipka, Johannes C. B. Dietschreit, Lukáš Grajciar, and Rafael Gomez-Bombarelli. Differentiable simulations for enhanced sampling of rare events. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31990–32007. PMLR, 23–29 Jul 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1633–1645. Curran Associates, Inc., 2020.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yuanchao Wang, Zhao-Rong Lai, and Tianqi Zhong. Out-of-distribution generalization for total variation based invariant risk minimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

Philip Wolfe. Convergence conditions for ascent methods. ii: Some corrections. *SIAM Review*, 13 (2):185–188, 1971.

David P. Woodruff and Samson Zhou. Adversarially robust dense-sparse tradeoffs via heavy-hitters. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 10908–10934. Curran Associates, Inc., 2024.

Junfeng Yang, Yin Zhang, and Wotao Yin. An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise. *SIAM Journal on Scientific Computing*, 31(4):2842–2865, 2009.

Jingfeng Zhang, Binghui Lin, Shiqi Xia, Mingyuan Zhao, Ziwei Zhu, Pengtao Zhang, and Cho-Jui Hsieh. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2022.

G Zoutendijk. Computational methods in nonlinear programming. *Studies in Optimization*, 1:125, 1970.

A THEORETICAL DETAILS

A.1 PROOF OF THEOREM 1

Proof. **Part (1):**

Let x^* be a solution to (5). According to Fermat's rule, we have $\nabla V(x^*) = 0$. Then for any $x \in \mathbb{R}^d$,

$$V(x^*) + \lambda|\nabla V(x^*)| = V(x^*) \leqslant V(x) \leqslant V(x) + \lambda|\nabla V(x)|. \tag{28}$$

Hence x^* is also a solution to (14).

Part (2):

Conversely, let x^* be a solution to (14). Then for any $x \in \mathbb{R}^d$, we have

$$V(x^*) + \lambda|\nabla V(x^*)| \leqslant V(x) + \lambda|\nabla V(x)|. \tag{29}$$

Suppose x^* is not a solution to (5), then there exists a solution $x^\circ \neq x^*$ to (5) such that $V(x^\circ) < V(x^*)$. According to Fermat's rule, we have $\nabla V(x^\circ) = 0$. Inserting $x = x^\circ$ into (29) yields

$$V(x^*) \leqslant V(x^*) + \lambda|\nabla V(x^*)| \leqslant V(x^\circ) + \lambda|\nabla V(x^\circ)| = V(x^\circ), \tag{30}$$

which contradicts the above-mentioned deduction $V(x^\circ) < V(x^*)$. Hence x^* should also be a solution to (5).

□

A.2 CONVERGENCE OF ALGORITHM (19)

First, we need to verify that the level set $\mathscr{L} := \{x : W(x) \leqslant W(x^{(0)})\}$ is bounded. In the literature, many analytic potential functions V satisfy this bounded level set condition, like the exponential family and the quadratic family in forms of V_m and V_q in (39), respectively. Moreover, the LV $|\nabla V(x)|$ also satisfies the bounded level set condition if V belongs to the analytic potential function family like V_m and V_q. Since $W(x) = V(x) + \lambda|\nabla V(x)|$ with $\lambda \geqslant 0$, $W(x)$ also satisfies the bounded level set condition.

Second, we need to verify that in a neighborhood $\mathscr{N}$ of $\mathscr{L}$, W has Lipschitz smoothness: there exists a constant $L_w > 0$, such that

$$\|\nabla W(x) - \nabla W(x')\|_2 \leqslant L_w\|x - x'\|_2, \quad \forall x, x' \in \mathscr{N}. \tag{31}$$

Since $\mathscr{L}$ is bounded, we can also let $\mathscr{N}$ be a closed and bounded neighborhood. If V is an analytic potential function like V_m and V_q, then its second derivatives are differentiable and bounded in the closed and bounded set $\mathscr{N}$. Hence $\partial|\nabla V(x)|$ defined in (17) is differentiable and bounded in $\mathscr{N}$ except for a zero measure set $\{x : \nabla V(x) = 0\}$. Since $\nabla W(x) = \nabla V(x) + \lambda\partial|\nabla V(x)|$, $\nabla W(x)$ is also differentiable and bounded in $\mathscr{N}$ almost everywhere (a.e.). Hence $\nabla W(x)$ is Lipschitz continuous in $\mathscr{N}$ a.e. with some Lipschitz constant L_w. In practical numerical computations, the iterate $x^{(l)}$ hardly hits a zero measure set like $\{x : \nabla V(x) = 0\}$ (or we could add a small distraction to the iterate to escape from this zero set), hence we could consider W Lipschitz-smooth in $\mathscr{N}$.

Third, since the Wolfe line search (20) satisfies the Zoutendijk condition (Zoutendijk, 1970; Wolfe, 1969; 1971):

$$\sum_{l=1}^{\infty} \frac{((\nabla W(x^{(l)}))^\top \mathfrak{d}^{(l)})^2}{\|\mathfrak{d}^{(l)}\|_2^2} < \infty, \tag{32}$$

it follows from Theorem 4.3 of (Gilbert & Nocedal, 1992) that algorithm (19) converges in the following sense:

$$\liminf_{l\to\infty} \|\nabla W(x^{(l)})\|_2 = 0. \tag{33}$$

In practice, we strictly follow standard optimization literature (Nocedal & Wright, 2006) by setting $c_1 = 10^{-4}$ and $c_2 = 0.9$ to ensure consistent convergence.

A.3 Proof of Theorem 3

Proof. Similar to (31), $L_{x,\epsilon}$-smoothness ($L_{x,\epsilon} > 0$) can be defined by

$$\|\nabla V(y) - \nabla V(y')\|_2 \leqslant L_{x,\epsilon}\|y - y'\|_2, \quad \forall y, y' \in \mathcal{B}_2[x;\epsilon]. \tag{34}$$

If $V(y)$ is twice differentiable in $\mathcal{B}_2[x;\epsilon]$, then we can let

$$L_{x,\epsilon} = \max_{y\in\mathcal{B}_2[x;\epsilon]}(\varsigma_{max}(\nabla^2 V(y))), \tag{35}$$

where ς_{max} denotes the largest eigenvalue of the Hessian $\nabla^2 V(y)$. From Proposition A.24 of (Bertsekas, 1999), $L_{x,\epsilon}$-smoothness implies

$$V(x+\delta) \leqslant V(x) + \nabla V(x)^\top \delta + \frac{L_{x,\epsilon}}{2}\|\delta\|_2^2, \quad \forall\, \|\delta\|_2 \leqslant \epsilon. \tag{36}$$

The right side of (36) is a quadratic form of δ that takes the maximum value at $\delta = \epsilon\frac{\nabla V(x)}{\|\nabla V(x)\|_2}$. Hence,

$$\begin{aligned}\max_{\|\delta\|_2\leqslant\epsilon} V(x+\delta) &\leqslant V(x) + \epsilon\|\nabla V(x)\|_2 + \frac{L_{x,\epsilon}}{2}\|\epsilon\|_2^2 \\ &\leqslant V(x) + \lambda\|\nabla V(x)\|_2 + \frac{L_{x,\epsilon}}{2}\|\epsilon\|_2^2 \\ &= W(x) + \frac{L_{x,\epsilon}}{2}\|\epsilon\|_2^2,\end{aligned} \tag{37}$$

which proves the theorem. □

B Experimental Details and Additional Experiments

The experimental settings of this paper basically follow those of (He et al., 2024) to make fair comparisons, which are detailed in this section. The STD of the Gaussian attack is set to $\sigma = 0.01$ or 0.1, while the perturbation budget of FGSM and PGD[10] in (7) and (8) is set to $\epsilon = 0.01$.

B.1 2D Gaussian Mixture Model on Different Oracle Complexities and Different Dimensionalities

A four-mode 2D GMM with the following default parameters is used in this paper:

$$\begin{aligned}&w: 0.1, 0.2, 0.3, 0.4, \\ &\mu: \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\11\end{bmatrix}, \begin{bmatrix}9\\9\end{bmatrix}, \begin{bmatrix}11\\0\end{bmatrix}, \\ &\Sigma: \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}, \begin{bmatrix}0.3 & -0.2\\-0.2 & 0.3\end{bmatrix}, \begin{bmatrix}1 & 0.3\\0.3 & 1\end{bmatrix}, \begin{bmatrix}1.2 & -1\\-1 & 1.2\end{bmatrix},\end{aligned} \tag{38}$$

where w, μ, and Σ denote the weights, means, and covariances for the four modes in the 2D GMM, respectively. The hyperparameters for the compared methods are set as Table B1. The proposed ADDDS uses the same hyperparameters as those of ZOD-MC, since the former integrates the latter in the recovering stage. The regularization strength for ADDDS is set to a trivial value $\lambda = 1$. Additional experimental results on different oracle complexities and different attack schemes are provided in Figures B2, B4, B5, B6, B7. Figure B1 shows that the runtime of ADDDS ranks amongst other baselines and keeps steady as the oracle complexity increases.

As for dimensionality dependence, given a fixed dimensionality d, we first randomly sample $u \in U[0,1]^d$ and $\varsigma^2 \in U[0.3, 1.3]$, then let $\mu = 12\cdot\frac{u}{\|u\|_2}$. This yields a Gaussian distribution $\mathcal{N}(\mu, \varsigma^2\mathbb{I}_d)$. This procedure is repeated for five times to form an equally-weighted five-mode GMM. Then the experiments on different dimensionalities and different attack schemes are conducted and shown in Figures 2(b) and B3.

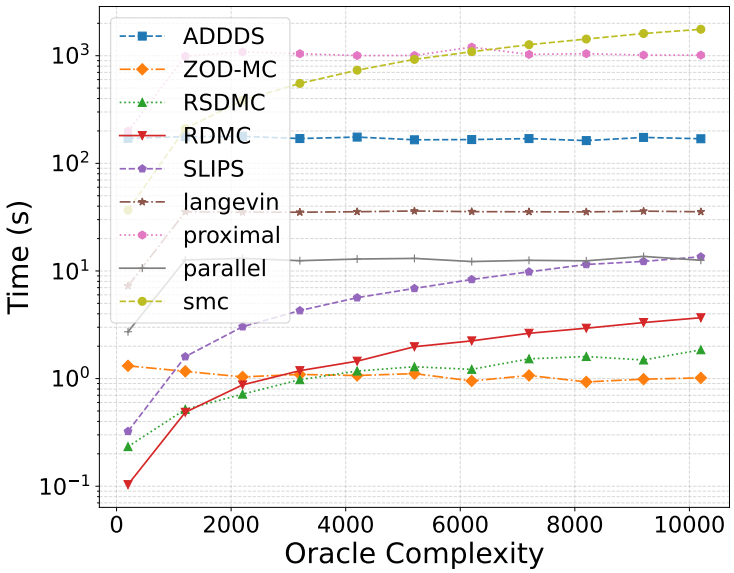
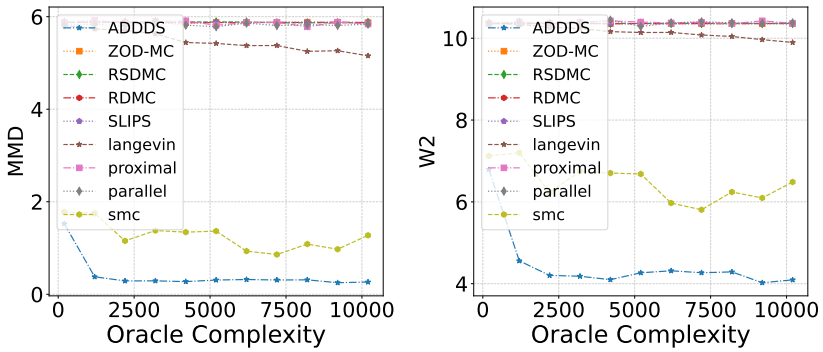


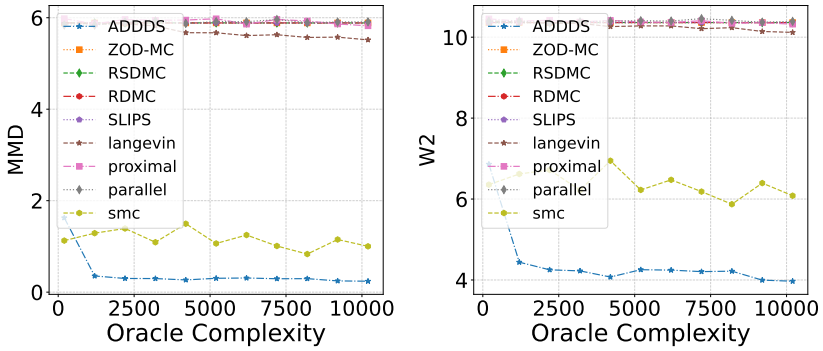
Figure B1: Runtimes of different methods w.r.t. oracle complexity for the 2D GMM experiment.

Table B1: Hyperparameters of different methods for the oracle complexity and dimensionality dependence experiments. These hyperparameters include the terminal time of the diffusion process T, the number of timestamps N, the early stopping parameter ζ, the inner-loop step size of Markov Chain Monte Carlo (MCMC), the number of MCMC samples, the number of MCMC steps, the number of Markov chains. K and M indicate the current oracle complexity and a matched oracle complexity, respectively. 2 recursions per score evaluation are used for RSDMC.

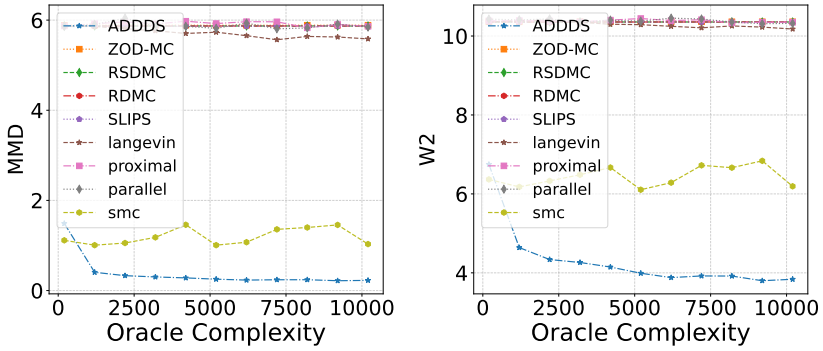
Method	T	N	ζ	**Step Size**	**N-MCMC**	**Num Steps**	**N-Chains**
ADDDS	2	25	5e-3	-	K	-	-
ZOD-MC	2	25	5e-3	-	K	-	-
RDMC	2	25	5e-2	0.01	1000	$K/1000$	-
RSDMC	2	25	5e-2	0.01	$K^{1/4}$	$K^{1/4}$	-
SLIPS	1	25	6.62e-3	Adaptive	1000	$K/1000$	-
AIS	-	-	-	Adaptive	-	M	512
SMC	-	-	-	Adaptive	-	M	512
Langevin	-	-	-	0.01	-	M	-
Proximal	-	-	-	1/5	-	M	-
Parallel	-	-	-	0.01	-	M	512



(a) Gaussian attack ($\sigma = 0.1$)

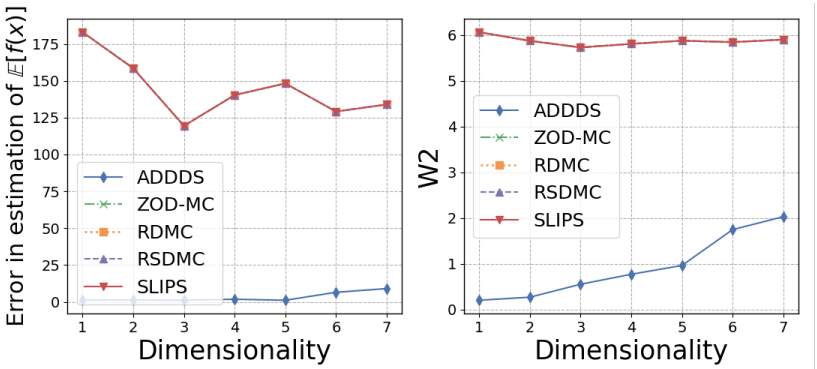


(b) Gaussian attack ($\sigma = 0.01$)

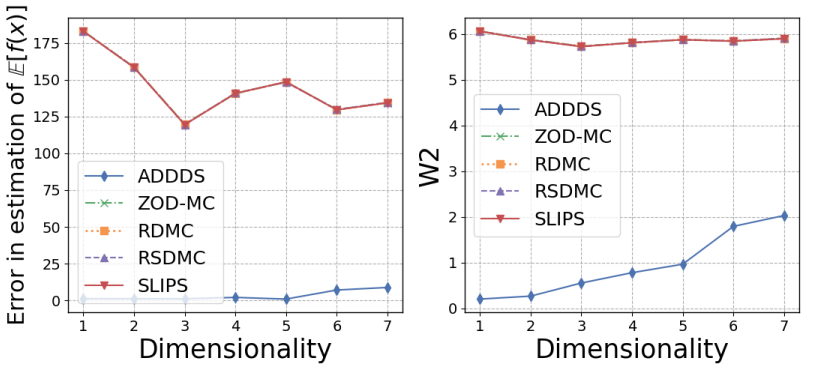


(c) FGSM attack

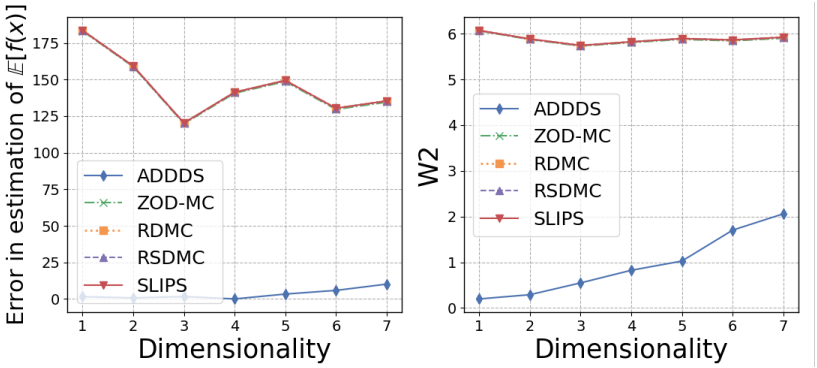
Figure B2: Sampling accuracies of different methods w.r.t. oracle complexity for Gaussian mixtures under different adversarial attacks. A lower score indicates a better performance.



(a) Gaussian attack ($\sigma = 0.01$)



(b) Gaussian attack ($\sigma = 0.1$)



(c) FGSM attack

Figure B3: Sampling accuracies of different methods w.r.t. dimensionality for Gaussian mixtures under different adversarial attacks. A lower score indicates a better performance. The plots of other baselines mutually overlap.

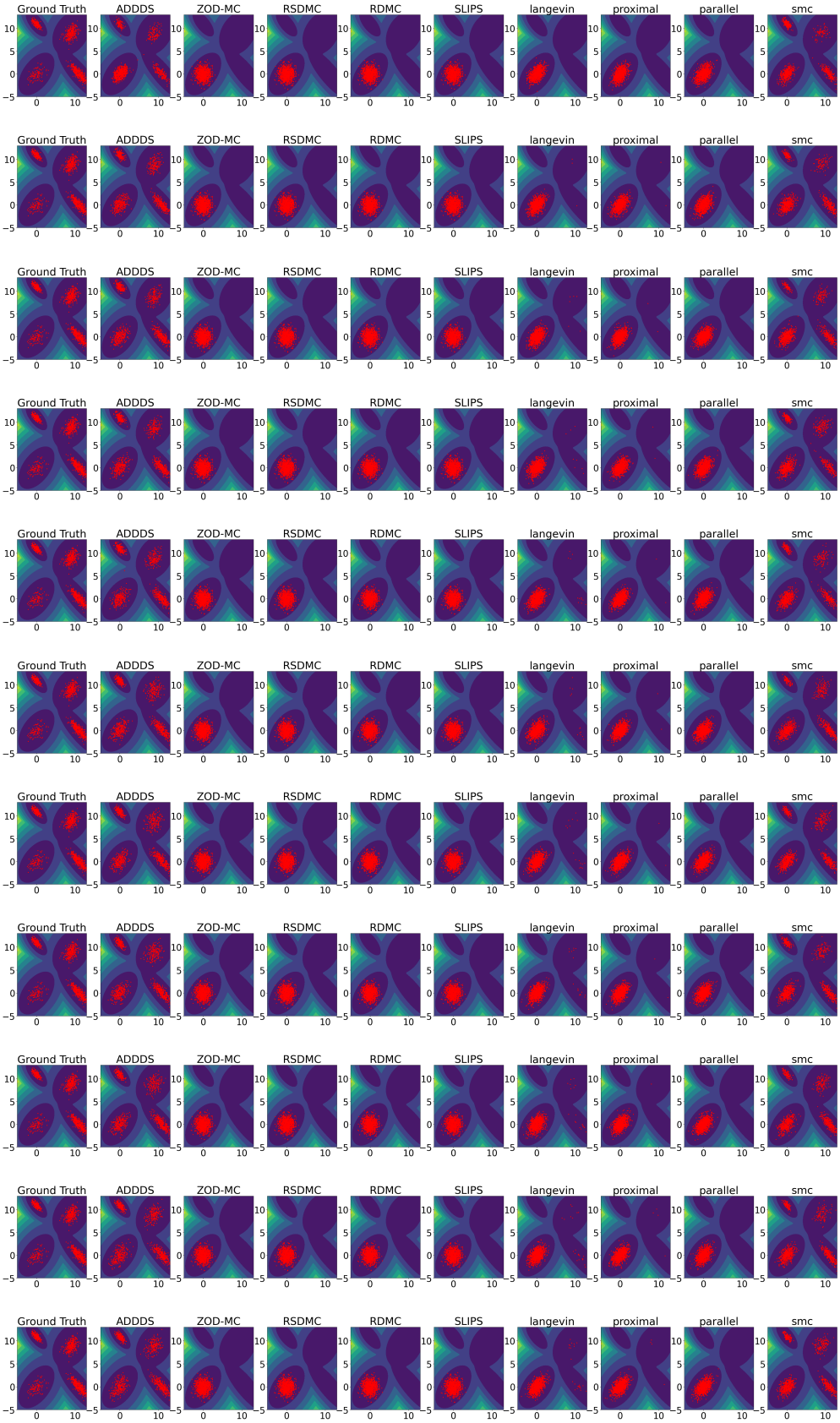


Figure B4: Samples generated by different methods from an asymmetric and unbalanced Gaussian mixture under Gaussian attacks ($\sigma = 0.01$). From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

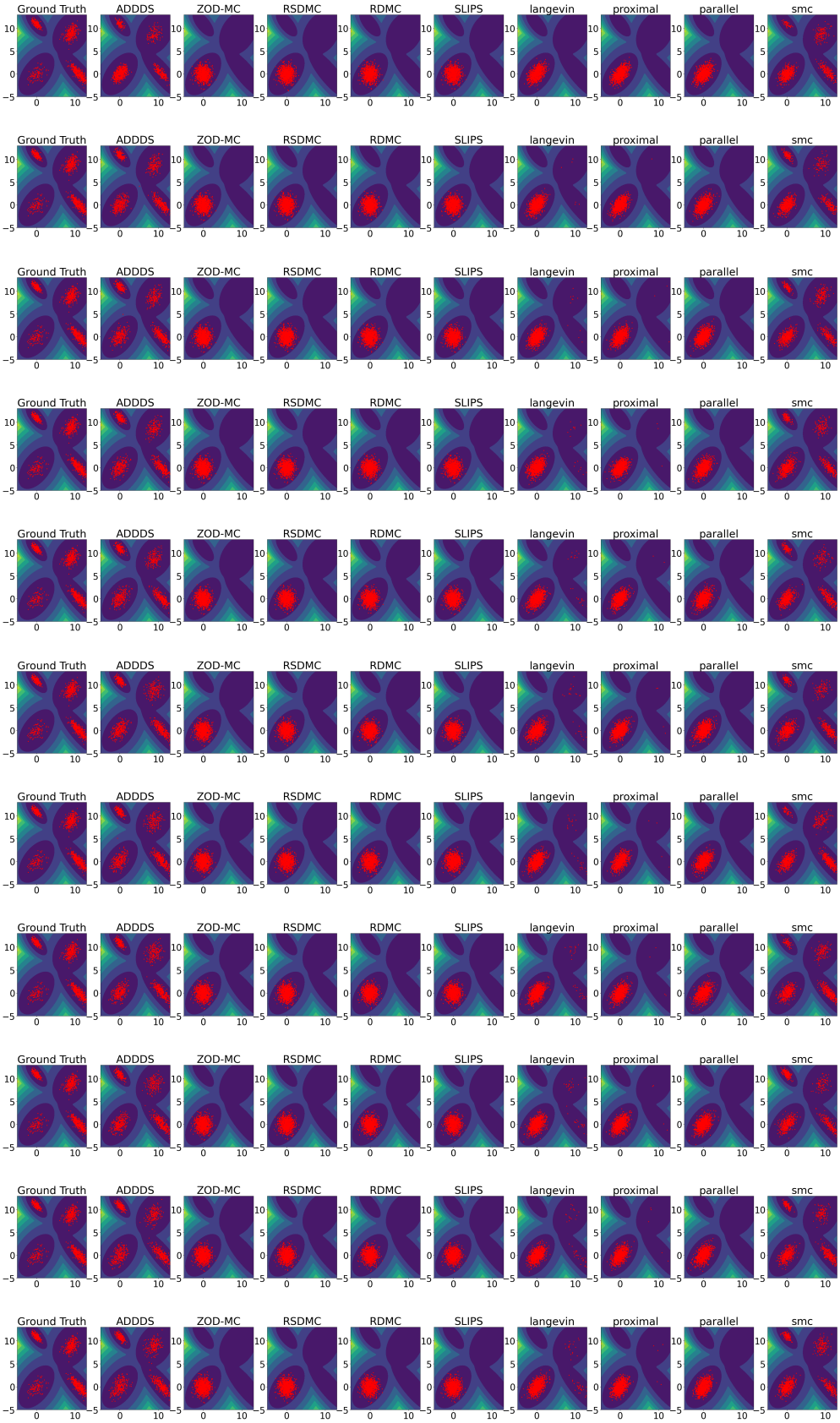


Figure B5: Samples generated by different methods from an asymmetric and unbalanced Gaussian mixture under Gaussian attacks ($\sigma = 0.1$). From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

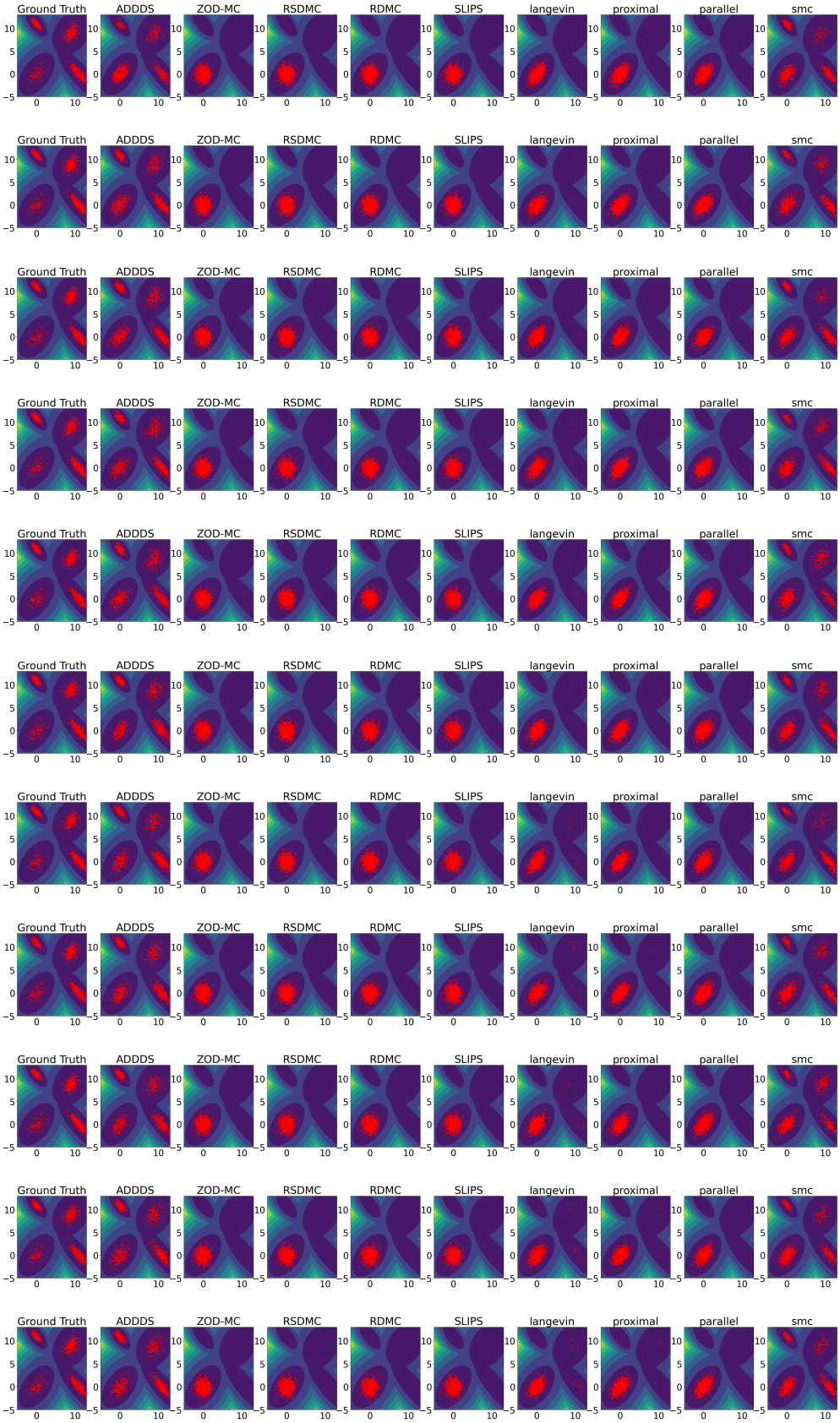


Figure B6: Samples generated by different methods from an asymmetric and unbalanced Gaussian mixture under FGSM attacks. From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

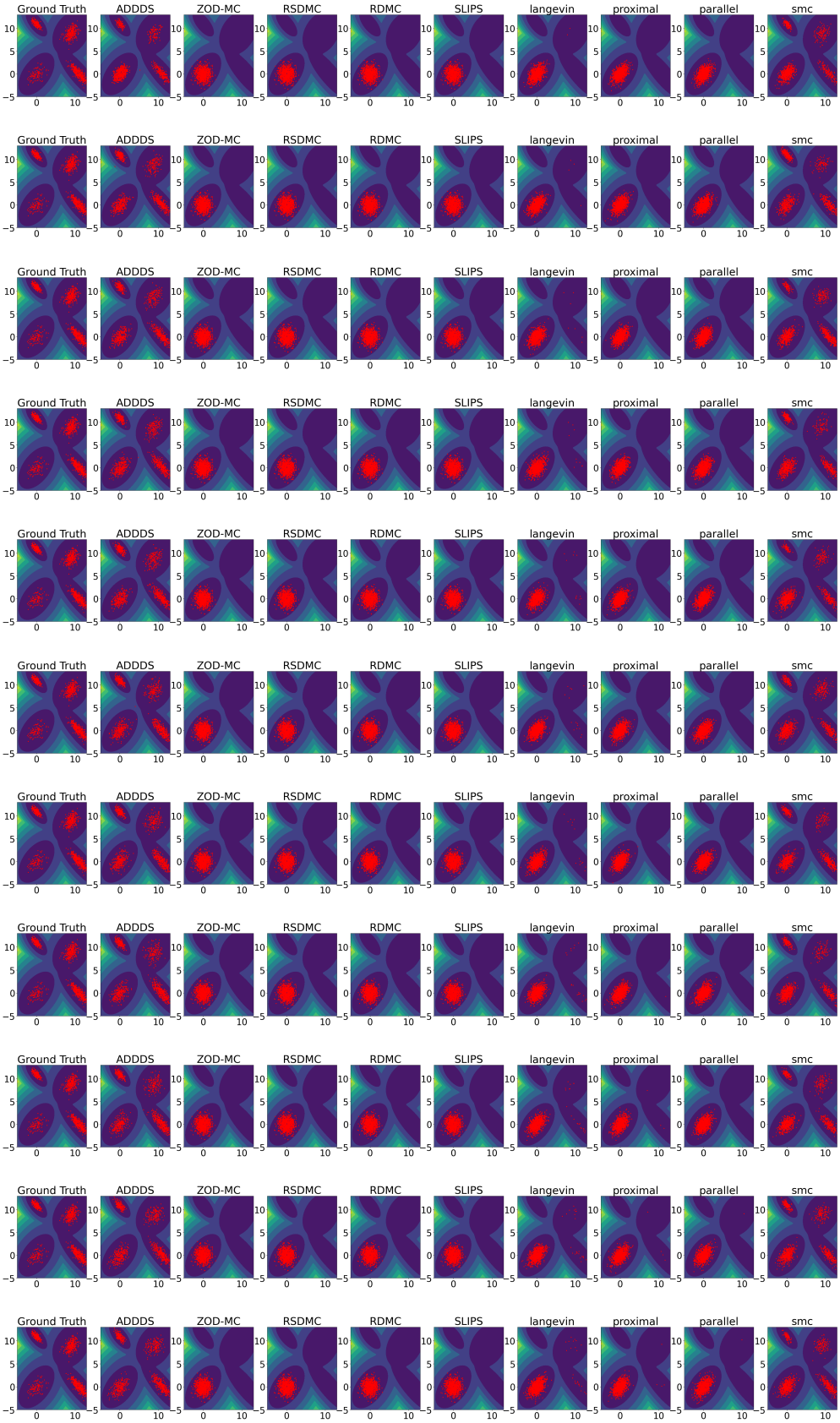


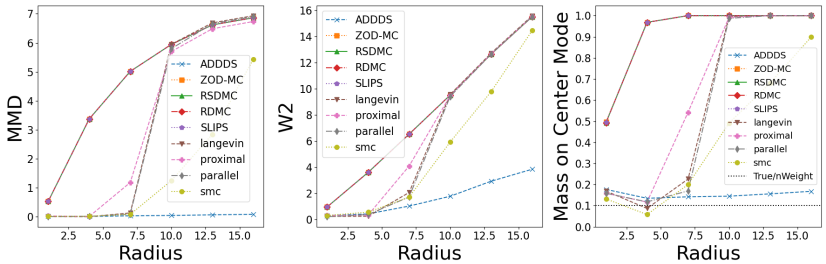
Figure B7: Samples generated by different methods from an asymmetric and unbalanced Gaussian mixture under PGD^{10} attacks. From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

B.2 Robustness to Different Radii

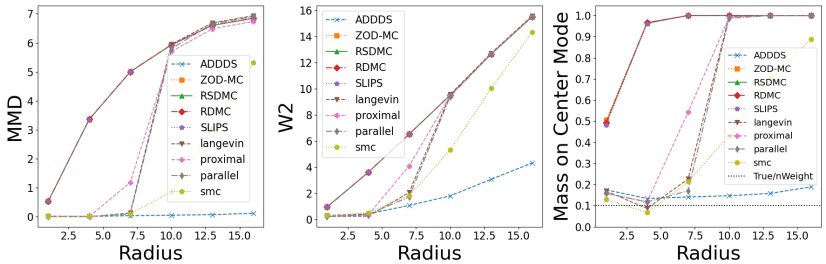
The hyperparameters for the compared methods are set as Table B2, where ADDDS uses the same hyperparameters as those of ZOD-MC. The experiments on different dimensionalities and different attack schemes are conducted and shown in Figures 4, B8, B9, B10, B11, and B12.

Table B2: Hyperparameters of different methods for the mode separation experiment. These hyperparameters include the terminal time of the diffusion process T, the number of timestamps N, the early stopping parameter ζ, the inner-loop step size of Markov Chain Monte Carlo (MCMC), the number of MCMC samples, the number of MCMC steps, the number of Markov chains. K and M indicate the current oracle complexity and a matched oracle complexity, respectively.

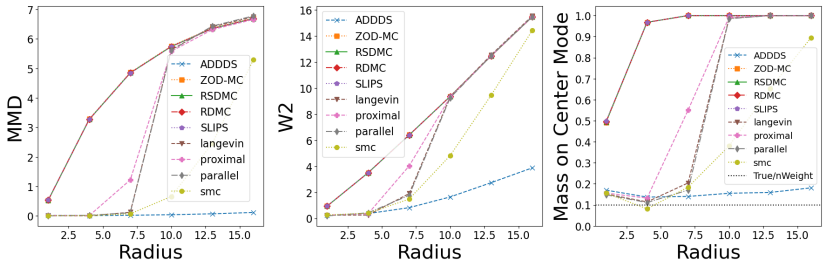
Method	T	N	ζ	**Step Size**	**N-MCMC**	**Num Steps**	**N-Chains**
ADDDS	10	50	5e-3	-	K	-	-
ZOD-MC	10	50	5e-3	-	K	-	-
RDMC	2	50	5e-2	0.01	1000	$K/1000$	-
RSDMC	2	50	5e-2	0.01	$K^{1/4}$	$K^{1/4}$	-
SLIPS	1	50	6.62e-3	Adaptive	1000	$K/1000$	-
AIS	-	-	-	Adaptive	-	M	512
SMC	-	-	-	Adaptive	-	M	512
Langevin	-	-	-	0.01	-	M	-
Proximal	-	-	-	1/40	-	M	-
Parallel	-	-	-	0.01	-	M	512



(a) Gaussian attack ($\sigma = 0.01$)

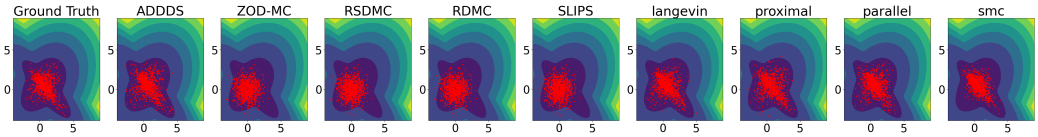


(b) Gaussian attack ($\sigma = 0.1$)

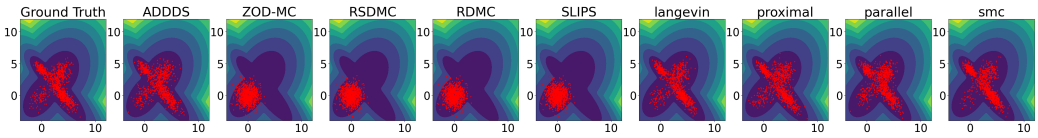


(c) FGSM attack

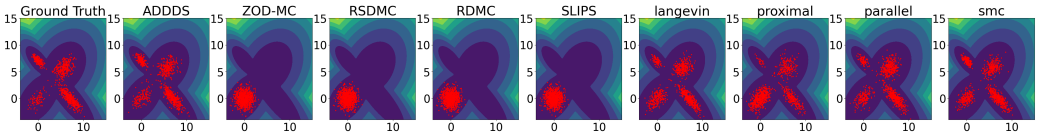
Figure B8: Sampling accuracies of different methods w.r.t. mode separation for Gaussian mixtures under different adversarial attacks. A lower score indicates a better performance.



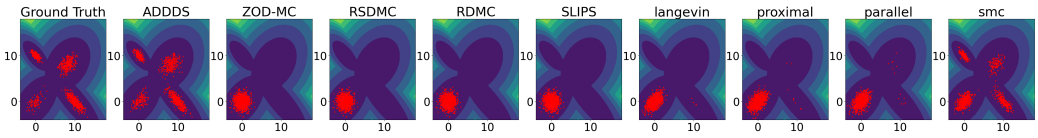
(a) Gaussian attack ($\sigma = 0.01, R = 1$)



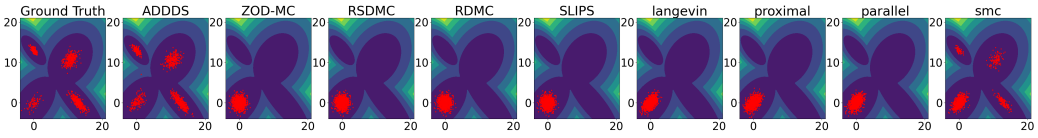
(b) Gaussian attack ($\sigma = 0.01, R = 4$)



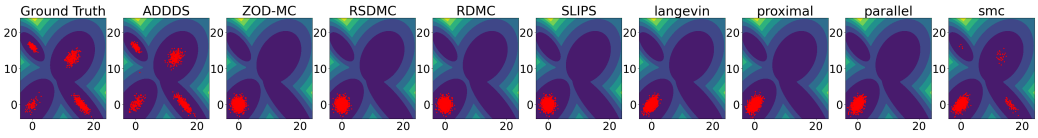
(c) Gaussian attack ($\sigma = 0.01, R = 7$)



(d) Gaussian attack ($\sigma = 0.01, R = 10$)

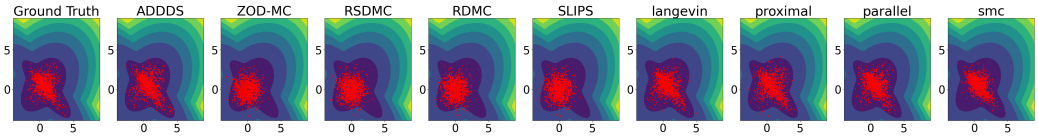


(e) Gaussian attack ($\sigma = 0.01, R = 13$)

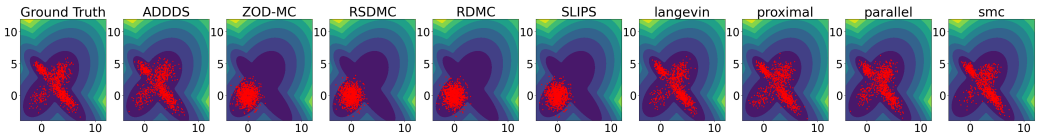


(f) Gaussian attack ($\sigma = 0.01, R = 16$)

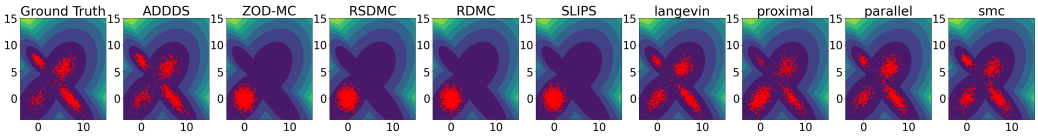
Figure B9: Samples generated by different methods for Gaussian mixtures with different radii under Gaussian attacks ($\sigma = 0.01$).



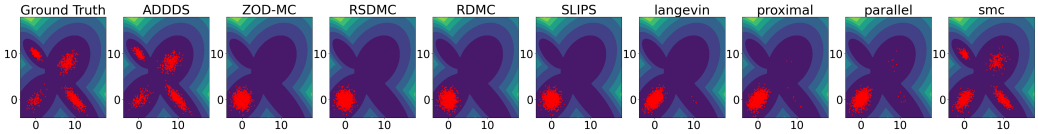
(a) Gaussian attack ($\sigma = 0.1, R = 1$)



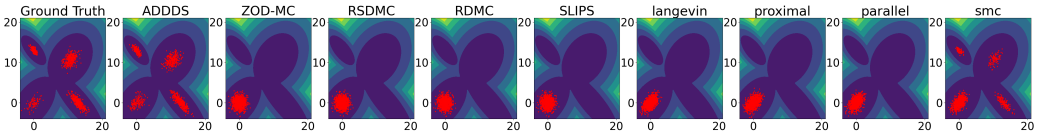
(b) Gaussian attack ($\sigma = 0.1, R = 4$)



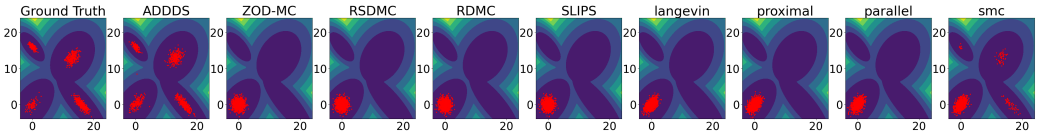
(c) Gaussian attack ($\sigma = 0.1, R = 7$)



(d) Gaussian attack ($\sigma = 0.1, R = 10$)

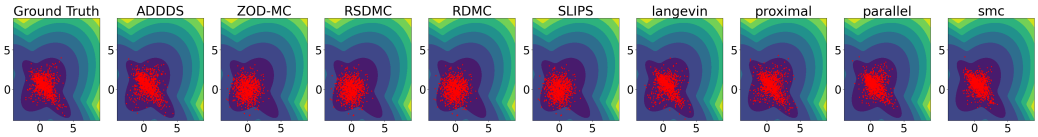


(e) Gaussian attack ($\sigma = 0.1, R = 13$)

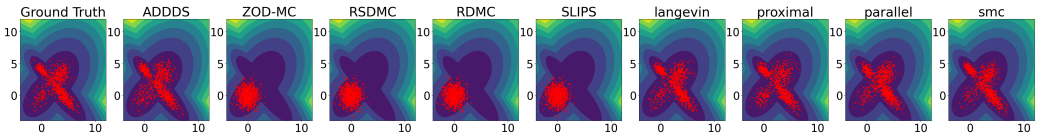


(f) Gaussian attack ($\sigma = 0.1, R = 16$)

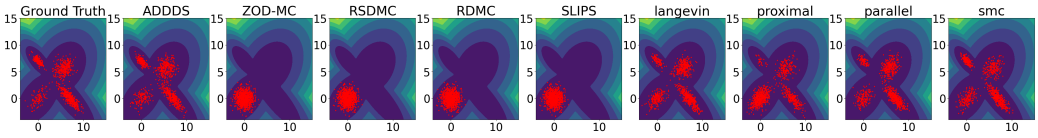
Figure B10: Samples generated by different methods for Gaussian mixtures with different radii under Gaussian attacks ($\sigma = 0.1$).



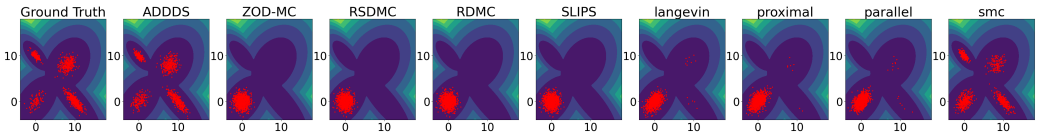
(a) FGSM attack ($R = 1$)



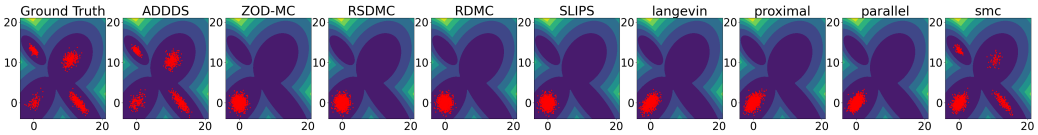
(b) FGSM attack ($R = 4$)



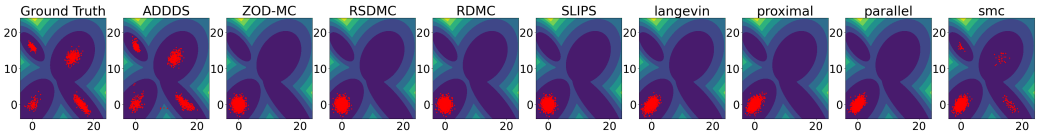
(c) FGSM attack ($R = 7$)



(d) FGSM attack ($R = 10$)

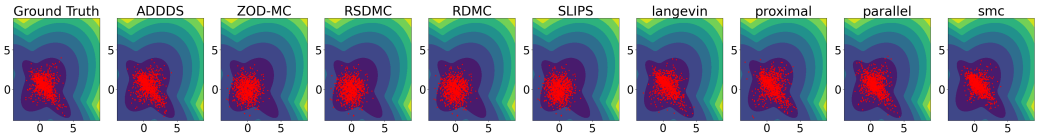


(e) FGSM attack ($R = 13$)

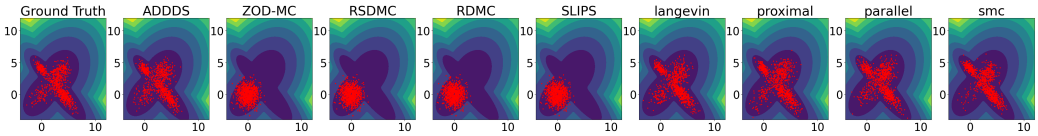


(f) FGSM attack ($R = 16$)

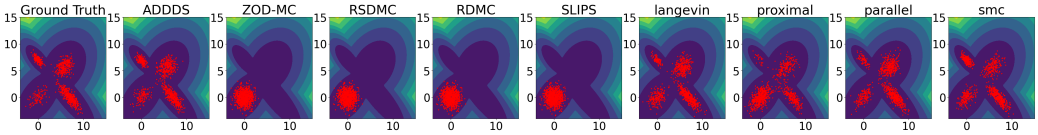
Figure B11: Samples generated by different methods for Gaussian mixtures with different radii under FGSM attacks.



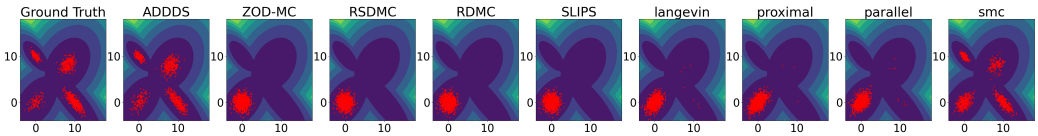
(a) PGD^{10} attack ($R = 1$)



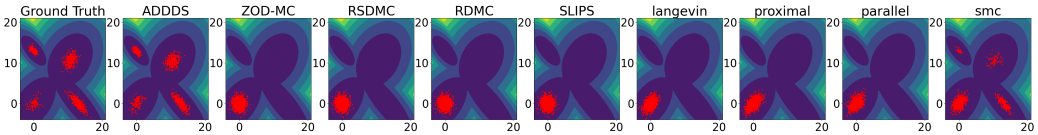
(b) PGD^{10} attack ($R = 4$)



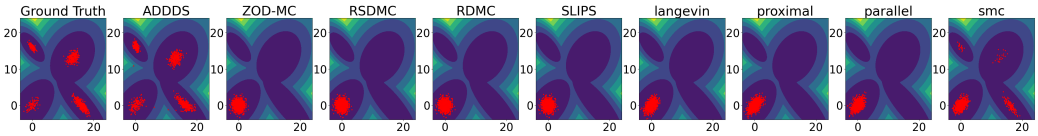
(c) PGD^{10} attack ($R = 7$)



(d) PGD^{10} attack ($R = 10$)



(e) PGD^{10} attack ($R = 13$)

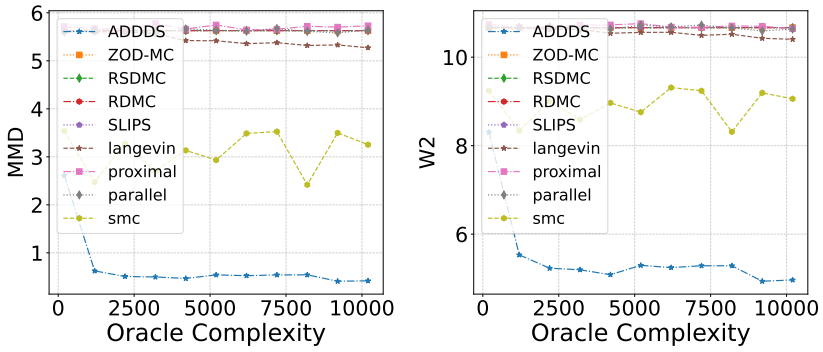


(f) PGD^{10} attack ($R = 16$)

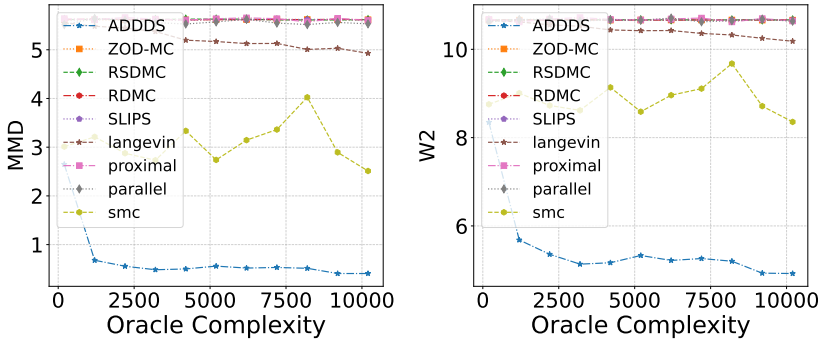
Figure B12: Samples generated by different methods for Gaussian mixtures with different radii under PGD^{10} attacks.

B.3 Discontinuous Potential

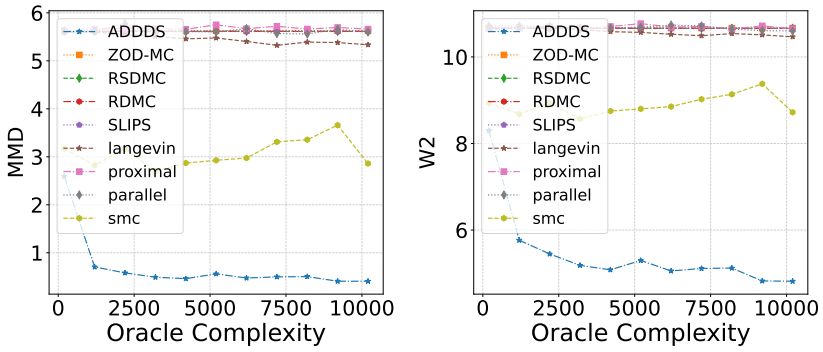
The hyperparameters for the compared methods are set as Table B1. The experiments for the discontinuous potential on different attack schemes are conducted and shown in Figures 6, 7, B13, B14, B15, B16, and B17.



(a) Gaussian attack ($\sigma = 0.01$)



(b) Gaussian attack ($\sigma = 0.1$)



(c) FGSM attack

Figure B13: Sampling accuracies of different methods w.r.t. mode separation for a discontinuous potential under different adversarial attacks. A lower score indicates a better performance.

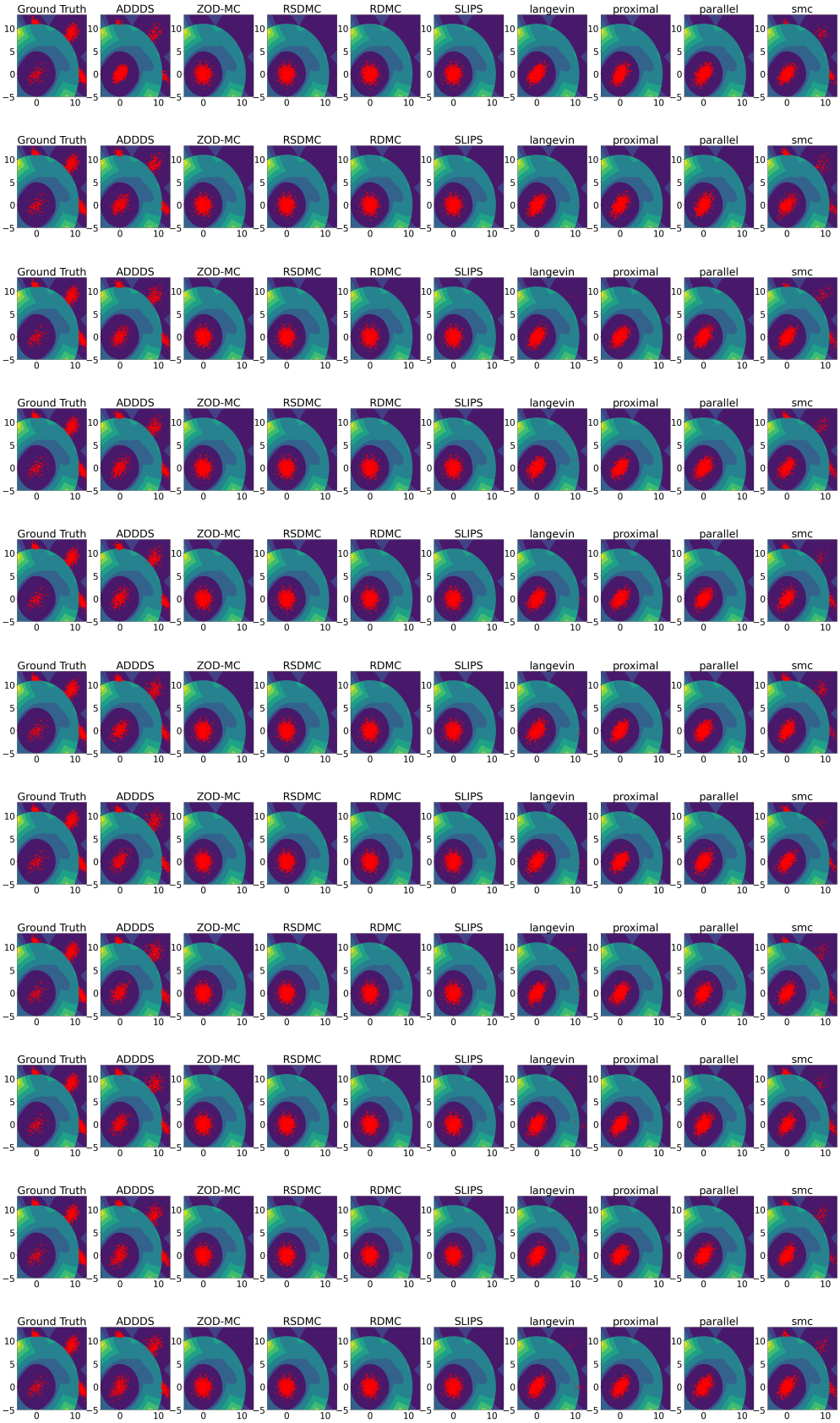


Figure B14: Samples generated by different methods from a discontinuous potential under Gaussian attacks ($\sigma = 0.01$). From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

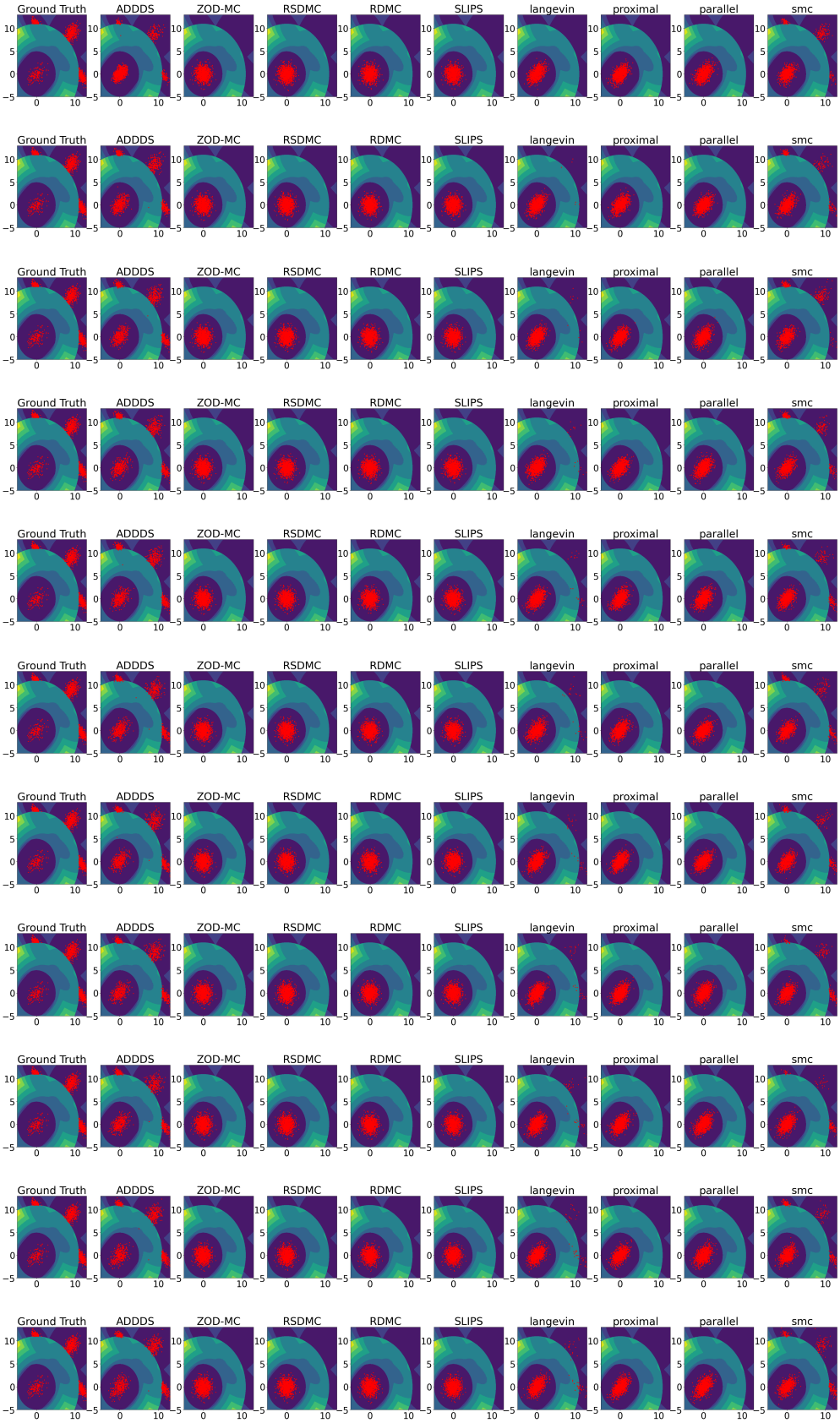


Figure B15: Samples generated by different methods from a discontinuous potential under Gaussian attacks ($\sigma = 0.1$). From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

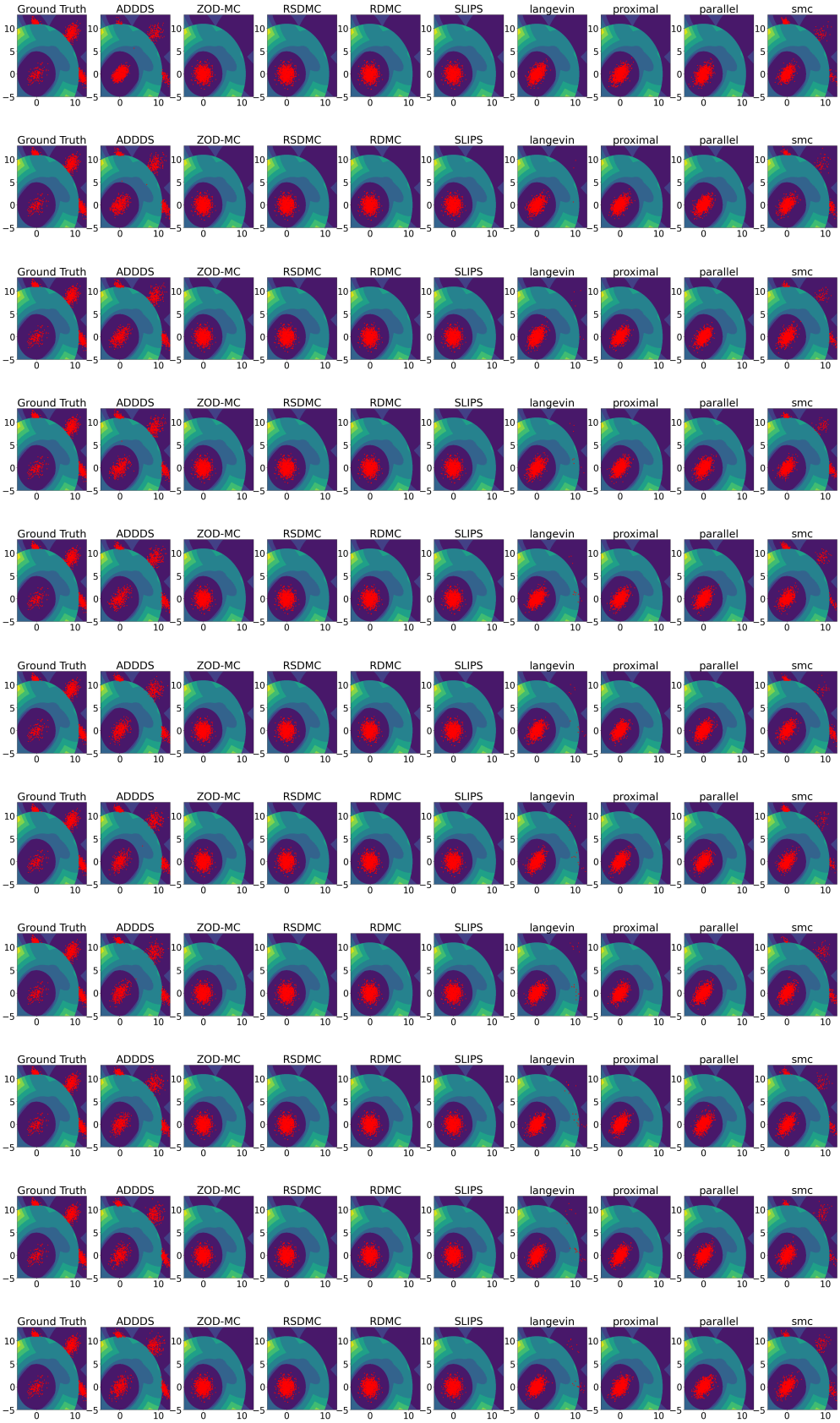


Figure B16: Samples generated by different methods from a discontinuous potential under FGSM attacks. From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

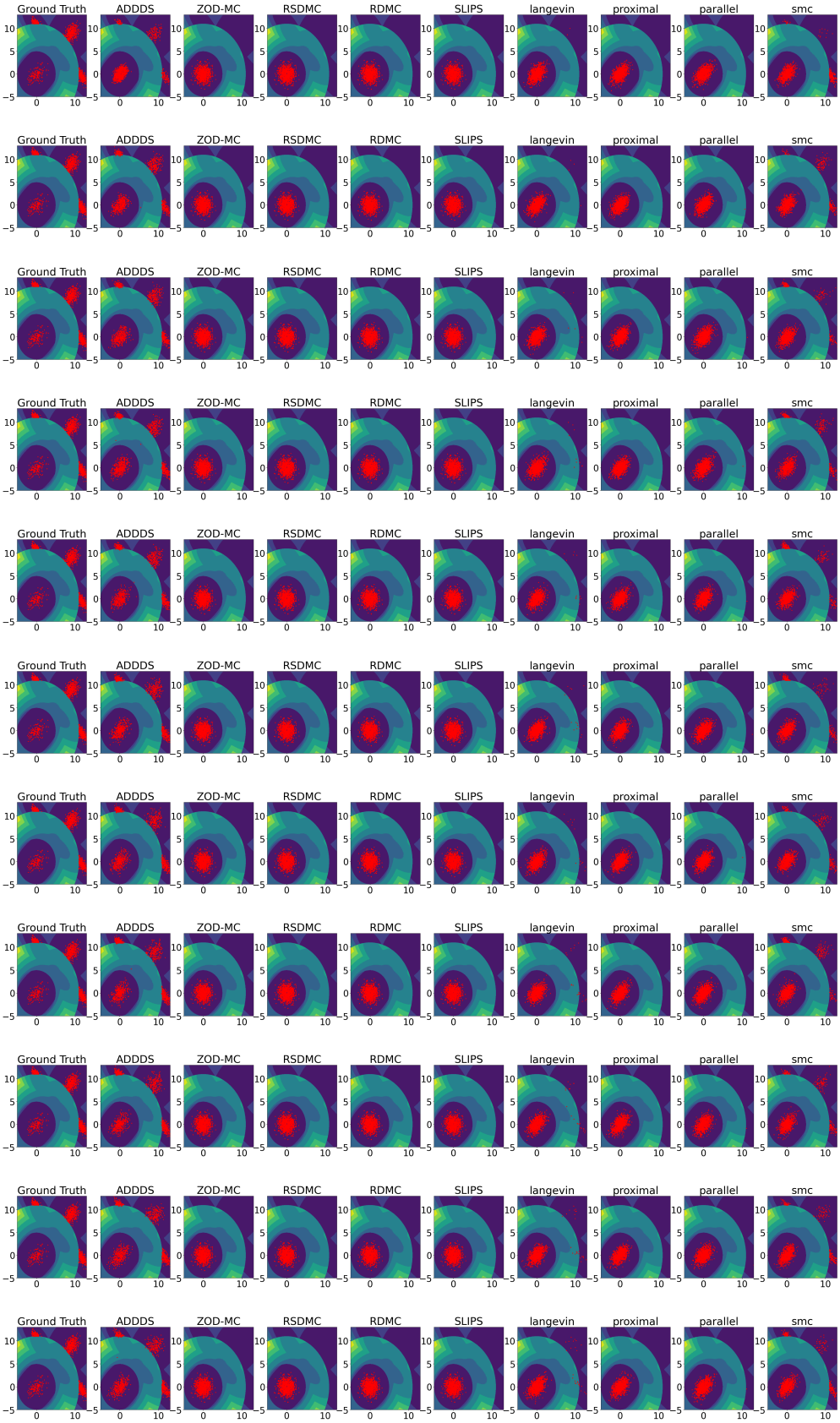


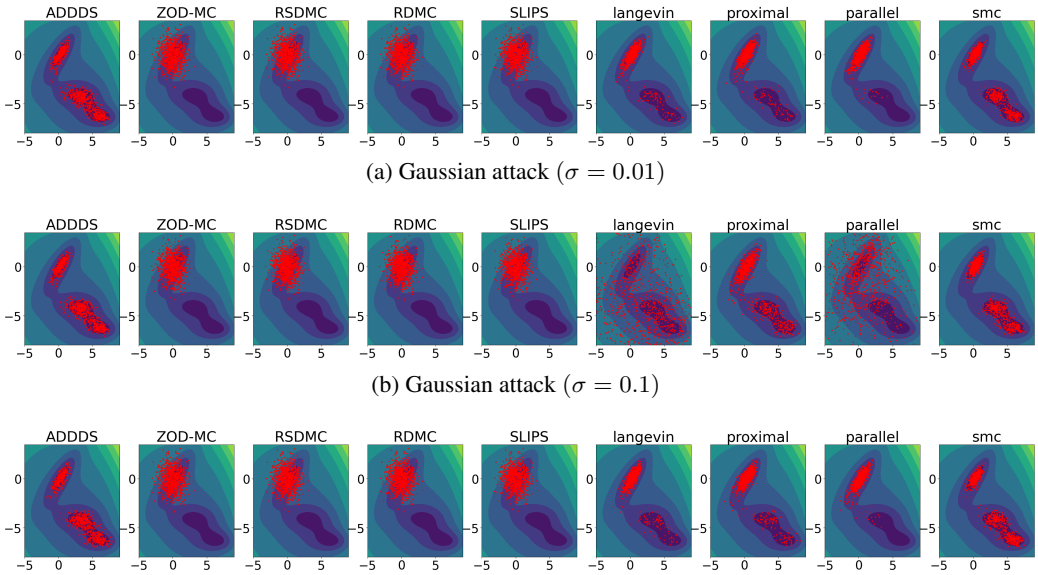
Figure B17: Samples generated by different methods from a discontinuous potential under PGD^{10} attacks. From top to bottom: oracle complexity= 200, 1200, 2200, 3200, 4200, 5200, 6200, 7200, 8200, 9200, 10200.

B.4 MÜLLER-BROWN POTENTIAL

The modified Müller-Brown potential used in this paper is:

$$
\begin{aligned}
V(x,y) :=& \beta(V_m(x,y) + V_q(x,y)),\\
V_m(x,y) :=& -170\exp(-6.5(x+0.5)^2 + 11(x+0.5)(y-1.5) - 6.5(y-1.5)^2)\\
&- 100\exp(-x^2 - 10(y-0.5)^2) - 200\exp(-(x-1)^2 - 10y^2)\\
&+ 15\exp(0.7(x+1)^2 + 0.6(x+1)(y-1) + 0.7(y-1)^2),\\
V_q(x,y) :=& 35.0136(x-x_c^*)^2 + 59.8399(y-y_c^*)^2,
\end{aligned} \tag{39}
$$

where V_m is the original Müller-Brown potential, and V_q is an introduced correction with (x_c^*, y_c^*) being the center for the middle well of V_m. V_q makes the depths of all three wells of V better leveled to each order. The scaling parameter is set to $\beta = 0.1$ by default.



(a) Gaussian attack ($\sigma = 0.01$)

(b) Gaussian attack ($\sigma = 0.1$)

(c) FGSM attack

Figure B18: Samples generated by different methods from the Müller-Brown potential under different adversarial attacks.

C ADDITIONAL EXPERIMENTS

C.1 HIGH-DIMENSIONAL EXPERIMENTS

We use the high-dimensional settings in Section 3.4 to conduct image classification experiments on the CIFAR 10 and CIFAR 100 data sets (Krizhevsky et al., 2009). The perturbation budget and the number of total sampling steps are set to $\epsilon = 0.01$ and $N = 25$, respectively. Results in Table C1 show that ADDDS outperforms all the other baselines across all the cases. Hence ADDDS is a robust defense scheme even in high-dimensional settings.

C.2 ABLATION STUDIES FOR HYPERPARAMETERS

We conduct ablation studies with 2D GMM under PGD^{10} attack (oracle complexity$=$ 1200) for several main hyperparameters of ADDDS: the LV regularization strength λ, the perturbation budget ϵ, and the number of total sampling steps N. Table C2 shows that ADDDS achieves stable performance when $\lambda = 1 \sim 5$, thus $\lambda = 1$ is used for ADDDS. Table C3 shows that ADDDS is robust against different levels of ϵ, while the non-defense version ZOD-MC deteriorates significantly even

Table C1: Classification accuracies (%) of different methods on the CIFAR data sets under different attacks.

Perturbation	ADDDS	ZOD-MC	RSDMC	RDMC	SLIPS	langevin	proximal	parallel	smc
	CIFAR 10								
Clean	**29.810**	12.030	24.180	28.720	9.890	11.490	28.800	27.020	20.150
Gaussian, $\sigma = 0.01$	**29.520**	11.820	23.040	28.460	9.480	10.200	28.330	26.210	19.750
Gaussian, $\sigma = 0.1$	**29.230**	11.390	22.430	28.180	9.060	9.950	21.040	18.530	19.680
FGSM	**14.970**	8.830	5.520	13.920	8.750	7.370	7.140	6.630	13.490
PGD^{10}	**9.230**	8.860	4.790	8.120	4.330	5.740	5.810	5.730	8.810
	CIFAR 100								
Clean	**7.100**	5.810	6.840	3.900	0.940	5.630	1.980	1.820	3.640
Gaussian, $\sigma = 0.01$	**6.940**	1.370	6.660	3.870	0.830	1.830	1.150	1.620	3.400
Gaussian, $\sigma = 0.1$	**6.880**	1.140	6.530	3.710	0.820	1.600	1.280	1.600	3.340
FGSM	**6.820**	0.680	6.300	0.590	0.740	0.550	0.480	0.460	2.380
PGD^{10}	**5.990**	0.530	5.800	0.430	0.690	0.300	0.280	0.450	2.170

under the lowest perturbation $\epsilon = 0.01$. Table C4 shows that ADDDS also achieves stable performance across various settings of N. In summary, ADDDS is robust to both adversarial perturbations and hyperparameter change.

Table C2: Ablation study for the regularization strength λ of ADDDS under PGD^{10} attack.

Oracle	200	1200	2200	3200	4200	5200	6200	7200	8200	9200	10200
	MMD										
$\lambda = 0.0$	5.876	5.883	5.883	5.764	5.931	5.646	5.892	5.910	4.880	4.741	4.478
$\lambda = 0.1$	2.004	1.297	1.189	1.266	1.437	1.301	1.213	1.404	1.186	1.313	1.209
$\lambda = 0.5$	1.878	0.643	0.574	0.561	0.480	0.473	0.572	0.523	0.534	0.496	0.445
$\lambda = 1.0$	**1.552**	**0.402**	**0.332**	**0.342**	**0.311**	**0.316**	**0.311**	**0.298**	**0.304**	**0.271**	**0.253**
$\lambda = 5.0$	1.431	0.398	0.351	0.334	0.325	0.327	0.336	0.349	0.310	0.298	0.326
$\lambda = 10.0$	1.924	0.719	0.683	0.564	0.431	0.677	0.592	0.710	0.580	0.541	0.578
	W2										
$\lambda = 0.0$	10.119	10.360	10.329	10.272	10.316	10.398	10.292	9.997	10.016	10.194	10.321
$\lambda = 0.1$	8.566	7.898	7.210	7.811	7.519	7.330	7.012	7.111	6.947	6.870	6.950
$\lambda = 0.5$	7.193	5.974	5.826	5.791	5.837	5.772	5.980	5.317	5.557	5.892	5.419
$\lambda = 1.0$	**6.747**	**4.631**	**4.337**	**4.279**	**4.149**	**4.287**	**3.872**	**3.981**	**3.759**	**3.810**	**3.925**
$\lambda = 5.0$	6.431	4.988	5.130	5.111	4.808	4.975	5.020	5.219	4.871	4.772	4.671
$\lambda = 10.0$	8.909	6.900	6.218	6.367	6.211	6.471	6.410	6.198	6.509	6.172	6.084

Table C3: Ablation study for the perturbation budget ϵ under PGD^{10} attack.

	ϵ	0.00	0.01	0.03	0.05	0.10	0.15	0.20
MMD	ZOD-MC	1.489	5.889	5.876	5.892	5.879	5.866	5.887
	ADDDS	**0.257**	**0.256**	**0.252**	**0.262**	**0.263**	**0.254**	**0.270**
W2	ZOD-MC	6.371	10.361	10.362	10.359	10.360	10.360	10.359
	ADDDS	**4.091**	**4.090**	**4.075**	**4.115**	**4.130**	**4.077**	**4.151**

C.3 Comparison with Gradient Clipping and Predictor-corrector Sampler

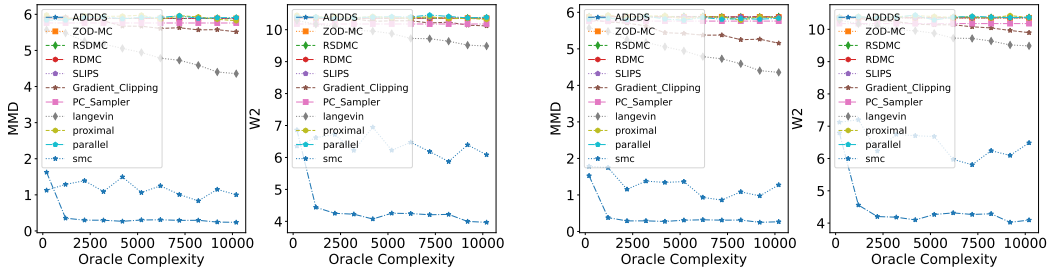
There are also intrinsic defense schemes for DDS, such as the gradient clipping for score estimation and the predictor-corrector sampler (Song et al., 2021). They are also taken into comparisons for the GMM model under different attacks in Figure C1. Results show that ADDDS outperforms other competitors including these two intrinsic defense schemes. Hence the adversarial attacks also threaten these two defense schemes, which strengthens the necessity of developing ADDDS.

C.4 Performance against Adaptive White-box Attack

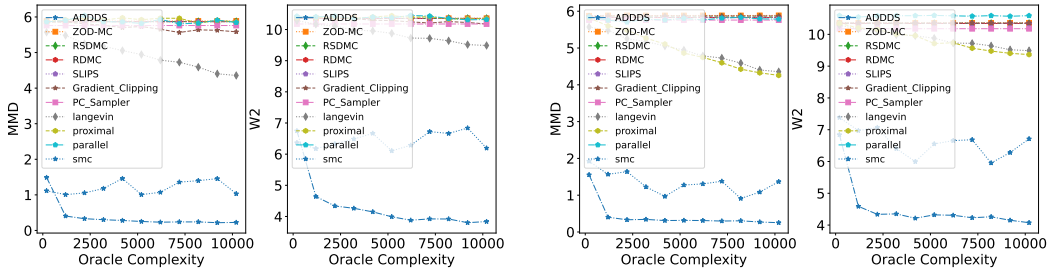
We also consider a higher threat of the adaptive white-box attack scheme (Tramer et al., 2020), where the attacker knows the exact defense strategy (e.g., Eq. 14 for the case of ADDDS) and

Table C4: Ablation study for the number of total sampling steps N under PGD^{10} attack.

	N	15	20	25	45	65	85	100
MMD	ZOD-MC	5.873	6.119	5.872	5.876	5.874	5.867	5.872
	ADDDS	**0.246**	**0.285**	**0.234**	**0.308**	**0.277**	**0.316**	**0.326**
W2	ZOD-MC	10.359	10.445	10.356	10.360	10.362	10.356	10.362
	ADDDS	**3.865**	**4.198**	**3.934**	**4.289**	**4.192**	**4.355**	**4.398**

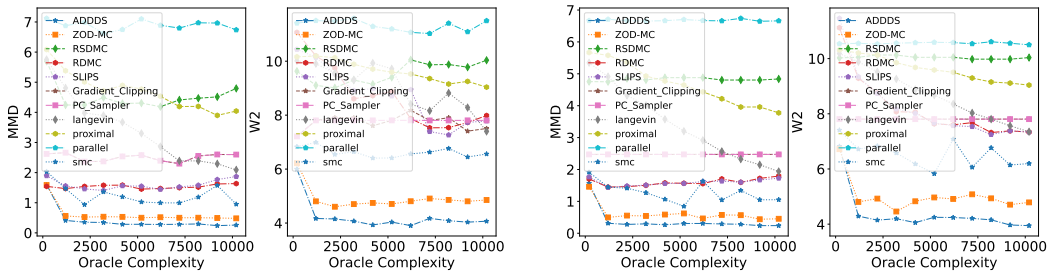


(a) Gaussian attack ($\sigma = 0.01$) (b) Gaussian attack ($\sigma = 0.1$)



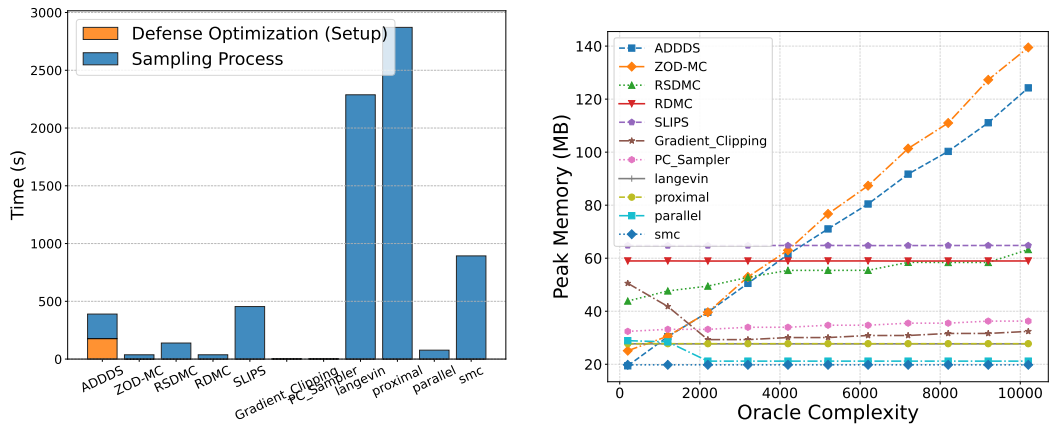
(c) FGSM attack (d) PGD^{10} attack

Figure C1: Sampling accuracy comparison of different methods (including the gradient clipping and the predictor-corrector sampler) for 2D GMM under different attacks.



(a) FGSM attack (b) PGD^{10} attack

Figure C2: Sampling performance comparison on 2D GMM under adaptive white-box attack.



(a) Computational Cost Breakdown(Oracle Complexity:10200)

(b) Comparison of peak memory usage (MB) across different sampling methods.

Figure C3: Computational resource analysis of different methods on 2D GMM under PGD^{10} attack.

further develops a countermeasure to evade the corresponding defense. Figure C2 shows that the defense performance of ADDDS does not deteriorate, whereas other non-defense competitors gain some improvement. Nevertheless, ADDDS is still robust against this adaptive white-box attack.

C.5 Computational Resource Requirement

To evaluate the performance-efficiency trade-off, we compare the computational resource requirements of different methods on 2D GMM under PGD^{10} attack in Figure C3. Figure C3(a) shows that ADDDS achieves competitive computational time among the compared methods, where the defense optimization takes up less than half of the total time. Figure C3(b) shows that ADDDS achieves competitive memory usage in the low oracle complexity cases ($\leqslant 3200$). From the overall experimental results in this paper, ADDDS already achieves satisfactory recovery performance with a low oracle complexity. Hence ADDDS achieves a good performance-efficiency trade-off. ADDDS requires a little less memory than ZOD-MC because the former just uses the plain (non-strong) Wolfe condition, while the latter uses the strong Wolfe condition.

C.6 Acceptance Rate and Score Estimation Error

We conduct experiments on the average acceptance rates and score estimation errors of ZOD-MC and ADDDS with 2D GMM under PGD^{10} attack (oracle complexity$=$ 1200). The acceptance rate means the proportion of accepted samples $\tilde{z}_{t,i}$ from the rejection sampler, and the score estimation error means $\|s(t,\tilde{x}) - \nabla \log p_t(\tilde{x})\|_2^2$. Both terms are averaged w.r.t. the timestamp t. Table C5 shows that ADDDS has a higher acceptance rate than ZOD-MC, which is mainly due to $V(x^{\bullet}_{ADDDS}) > V(x^{\bullet}_{org})$. This is consistent with the perturbation tolerance mechanism that adapts to $V(x^* + \delta) > V(x^*)$, as explained in Section 3.2. Moreover, ADDDS achieves a significantly smaller score estimation error than ZOD-MC, which indicates that ADDDS effectively suppresses the perturbation and provides a robust score estimation.

Table C5: Average acceptance rate and score estimation error on 2D GMM under PGD^{10} attack.

	Acceptance Rate	Score Estimation Error
ZOD-MC	2.734%	14.961
ADDDS	7.764%	**4.581**

D THE USE OF LARGE LANGUAGE MODELS (LLMS)

No Large Language Models (LLMs) are used in this paper.