Denoising MCMC for Accelerating Diffusion-Based Generative Models

Beomsu Kim [1] **Jong Chul Ye** [1]

Abstract

The sampling process of diffusion models can be interpreted as solving the reverse stochastic differential equation (SDE) or the ordinary differential equation (ODE) of the diffusion process, which often requires up to thousands of discretization steps to generate a single image. This has sparked a great interest in developing efficient integration techniques for reverse-S/ODEs. Here, we propose an orthogonal approach to accelerating score-based sampling: Denoising MCMC (DMCMC). DMCMC first uses MCMC to produce initialization points for reverse-S/ODE in the product space of data and diffusion time. Then, a reverse-S/ODE integrator is used to denoise the initialization points. Since MCMC traverses close to the data manifold, the cost of producing a clean sample for DMCMC is much less than that of producing a clean sample from noise. Denoising Langevin Gibbs, an instance of DMCMC, successfully accelerates all six reverse-S/ODE integrators considered in this work, and achieves state-of-the-art results: in the limited number of score function evaluation (NFE) setting on CIFAR10, we have 3.25 FID with ≈ 10 NFE and 2.49 FID with ≈ 16 NFE. On CelebA-HQ-256, we have 6.99 FID with ≈ 160 NFE, which beats the current best record of Kim et al. (2022a) among score-based models, 7.16 FID with 4000 NFE. Code: `https://github.com/1202kbs/DMCMC`

1. Introduction

Sampling from a probability distribution given its score function, i.e., the gradient of the log-density, is an active area of research in machine learning. Its applications range far and wide, from Bayesian learning (Welling & Teh, 2011) to learning energy-based models (Song & Kingma, 2021), synthesizing new high-quality data (Dhariwal & Nichol, 2021), and so on. Typical examples of traditional score-based samplers are Markov chain Monte Carlo (MCMC) methods such as Langevin dynamics (Langevin, 1908) and Hamiltonian Monte Carlo (Neal, 2011).

Recent developments in score matching with deep neural networks (DNNs) have made it possible to estimate scores of high-dimensional distributions such as those of natural images (Song & Ermon, 2019). However, natural data distributions are often sharp and multi-modal, rendering naïve application of traditional MCMC methods impractical. Specifically, MCMC methods tend to skip over or get stuck at local high-density modes, producing biased samples (Levy et al., 2018).

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a) depart from MCMC and use the concept of diffusion, the process of gradually corrupting data into noise, to generate samples. Song et al. (2021b) observed that for each diffusion process, there is a reverse stochastic differential equation (SDE) and an ordinary differential equation (ODE). Hence, given a noise sample, integrating the reverse-S/ODE produces a data sample. Only a time-dependent score function of the data during the diffusion process is required to simulate the reverse process.

This discovery generated great interest in finding better ways to integrate reverse-S/ODEs. For instance, Song et al. (2021b) uses black-box ODE solvers with adaptive step sizes to accelerate sampling. Moreover, multitude of recent works on score-based generative modeling focus on improving reverse-S/ODE integrators (Jolicoeur-Martineau et al., 2021; Lu et al., 2022; Karras et al., 2022; Zhang & Chen, 2022).

In this work, we develop an orthogonal approach to accelerating score-based sampling. Specifically, we propose Denoising MCMC (DMCMC) which combines MCMC with reverse-S/ODE integrators. MCMC is used to generate initialization points $\{(\boldsymbol{x}_n, t_n)\}$ in the product space of data $\boldsymbol{x}$ and diffusion time t. Since all modes are connected in the product space, MCMC mixes well. Then, a reverse-S/ODE integrator solves the reverse-S/ODE starting at $\boldsymbol{x}_n$ from time $t = t_n$ to $t = 0$. Since MCMC explores high-probability regions, the MCMC chain stays close to the data manifold, so t_n tends to be close to 0, i.e., noise level tends to be small (see Fig. 1 top and bottom panels). Thus, integrating the

[1]Kim Jaechul Graduate School of AI, KAIST, Daejeon, Korea. Correspondence to: Jong Chul Ye <jong.ye@kaist.ac.kr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

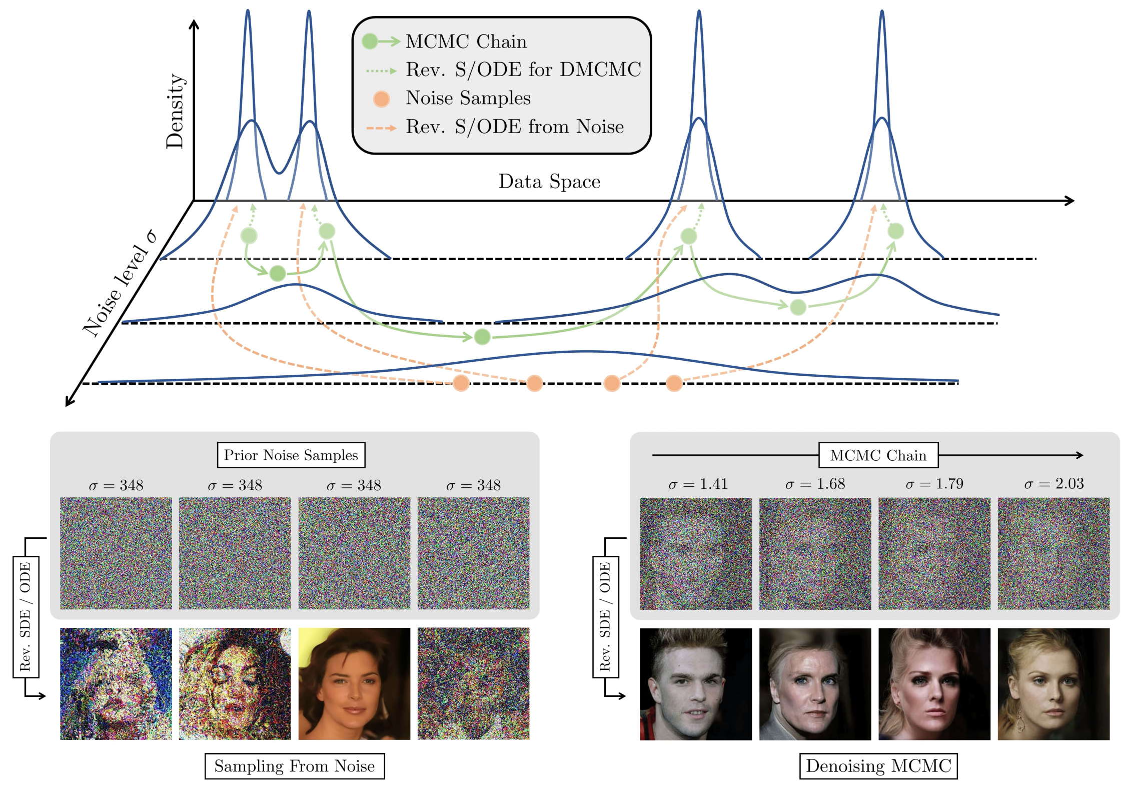


Figure 1: **Top:** a conceptual illustration of a diffusion model sampling process and DMCMC sampling process. diffusion models integrate the reverse-S/ODE starting from maximum diffusion time / maximum noise level. So, samples are often noisy with small computation budget due to large truncation error. DMCMC produces an MCMC chain which travels close to the image manifold (compare the noise level σ). So, the MCMC samples can be denoised to produce high-quality data with relatively little computation. **Bottom:** Visualization of sampling processes without (left) and with (right) DMCMC on CelebA-HQ-256 under a fixed computation budget.

reverse-S/ODE from $t = t_n$ to $t = 0$ is much faster than integrating the reverse-S/ODE from maximum time $t = T$ to $t = 0$ starting from noise. This leads to a significant acceleration of the sampling process.

Our contributions can be summarized as follows.

- We introduce the product space of data and diffusion time, and develop a novel score-based sampling framework called Denoising MCMC on the product space. Our framework is general, as any MCMC, any VP / VE score function, and reverse-S/ODE integrator can be used in a plug-and-play manner.

- We develop Denoising Langevin Gibbs (DLG), which is an instance of Denoising MCMC that is simple to implement and is scalable. The MCMC part of DLG alternates between a data update step with Langevin dynamics and a diffusion time prediction step, so all that DLG requires is a pre-trained score network and a diffusion time classifier.

- We verify the effectiveness of DLG by accelerating six reverse-S/ODE integrators. Notably, combined with the integrators of Karras et al. (2022), DLG achieves state-of-the-art results among score-based models. On CIFAR10 in the limited number of score function evaluation (NFE) setting, we obtain 3.25 FID with ≈ 10 NFE and 2.49 FID with ≈ 16 NFE. On CelebA-HQ-256, we have 6.99 FID with ≈ 160 NFE, which is currently the best result with score-based models. The cost of evaluating a diffusion time classifier is negligible, so we obtain acceleration essentially for free.

2. Background

2.1. Denoising Score Matching

Given a distribution $p(\boldsymbol{x})$, a noise level σ, and a perturbation kernel $p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \mathcal{N}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})$, solving the denoising score matching objective (Vincent, 2011)

$$\min_\theta \mathbb{E}_{p(\tilde{\boldsymbol{x}})} \mathbb{E}_{p_\sigma(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \left[\| \boldsymbol{s}_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \|_2^2 \right]$$

yields a score model $\boldsymbol{s}_\theta(\boldsymbol{x})$ which approximates the score of $\int p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})\,d\tilde{\boldsymbol{x}}$. Denoising score matching was then extended to train Noise Conditional Score Networks (NCSNs) $\boldsymbol{s}_\theta(\boldsymbol{x}, \sigma)$ which approximate the score of data smoothed at a general set of noise levels by solving

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

where $\lambda(\sigma)$ can be a discrete or a continuous distribution over $(\sigma_{\min}, \sigma_{\max})$ (Song & Ermon, 2019; Song et al., 2021b). We note $\int p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})\,d\tilde{\boldsymbol{x}}$ approaches $p(\boldsymbol{x})$ as $\sigma \to 0$, since the perturbation kernel $p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ converges to the Dirac delta function centered at $\boldsymbol{x}$.

2.2. Markov Chain Monte Carlo (MCMC)

Given an unnormalized version of $p(\boldsymbol{x})$ or the score function $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$, MCMC constructs a Markov chain in the data space whose stationary distribution is $p(\boldsymbol{x})$. An MCMC which uses the unnormalized density is the Metropolis-Hastings MCMC (Metropolis et al., 1953; Hastings, 1970) that builds a Markov chain by sequentially accepting or rejecting proposal distribution samples according to a density ratio. A popular score-based MCMC is Langevin dynamics (Langevin, 1908). Langevin dynamics generates a Markov Chain $\{\boldsymbol{x}_n\}_{n=1}^\infty$ using the iteration

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (\eta/2) \cdot \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_n) + \sqrt{\eta} \cdot \boldsymbol{\epsilon} \qquad (1)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. $\{\boldsymbol{x}_n\}_{n=1}^\infty$ converges to $p(\boldsymbol{x})$ in distribution for an appropriate choice of η.

To sample from a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$, we may resort to Gibbs sampling (Geman & Geman, 1984). Given a current Markov chain state $(\boldsymbol{x}_n, \boldsymbol{y}_n)$, Gibbs sampling produces $\boldsymbol{x}_{n+1}$ by sampling from $p(\boldsymbol{x} \mid \boldsymbol{y}_n)$ and $\boldsymbol{y}_{n+1}$ by sampling from $p(\boldsymbol{y} \mid \boldsymbol{x}_{n+1})$. The sampling steps may be replaced with MCMC. Hence, Gibbs sampling is useful when conditional distributions are amenable to MCMC.

Annealed MCMC. Despite their diversity, MCMC methods often have difficulty crossing low-density regions in high-dimensional multimodal distributions. For Langevin dynamics, at a low-density region, the score function vanishes in Eq. (1), resulting in a meaningless diffusion. Moreover, natural data often lies on a low-dimensional manifold. Thus, once Langevin dynamics leaves the data manifold, it becomes impossible for Langevin dynamics to return.

One way to remedy this problem is to use annealing, i.e., constructing a sequence of increasingly smooth and wide distributions and running MCMC at different levels of smoothness. As smoothness is increased, disjoint modes merge, so MCMC can cross over to other modes. Annealing has been used to empower various types of MCMC (Geyer & Thompson, 1995; Neal, 2001). In this work, we shall refer to the collection of MCMC that use annealing as annealed MCMC.

An instance of annealed MCMC is annealed Langevin dynamics (ALD) (Song & Ermon, 2019). For a sequence of increasing noise levels $\{\sigma_i\}_{i=1}^N$, Langevin dynamics is sequentially executed with $\int p_{\sigma_i}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})\,d\tilde{\boldsymbol{x}}$ in place of $p(\boldsymbol{x})$ in Eq. (1) for $i = N, N-1, \dots, 1$. Since $p(\boldsymbol{x})$ smoothed at a large noise level has wide support and connected modes, ALD overcomes the pitfalls of vanilla Langevin dynamics. However, ALD has the drawback that thousands of iterations are required to produce a single batch of samples.

2.3. Diffusion Models

Diffusion models and differential equations. Diffusion models opened up a new avenue towards fast sampling with score functions via SDEs and ODEs (Song et al., 2021b). Suppose data is distributed in $\mathbb{R}^d$. Given a diffusion process of data sample $\boldsymbol{x}_0 \sim p(\boldsymbol{x})$ into a sample from a simple prior noise distribution, the trajectory of data during diffusion can be described with an Itô SDE

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\,dt + g(t)\,d\boldsymbol{w} \qquad (2)$$

for some drift coefficient $\boldsymbol{f} : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$, diffusion coefficient $g : [0, T] \to \mathbb{R}$, and Brownian motion $\boldsymbol{w}$. Here, T is the diffusion termination time. With initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$, integrating Eq. (2) from time $t = 0$ to $t = T$ produces a sample from the prior distribution.

For each diffusion SDE, there exists a corresponding reverse-SDE:

$$d\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]\,dt + g(t)\,d\bar{\boldsymbol{w}} \qquad (3)$$

where $p_t(\boldsymbol{x})$ is the density of $\boldsymbol{x}(t)$ evolving according to Eq. (2) and $\bar{\boldsymbol{w}}$ is a Brownian motion if time flows from $t = T$ to $t = 0$. Given a sample $\boldsymbol{x}_T$ from the prior distribution, integrating Eq. (3) with initial condition $\boldsymbol{x}(T) = \boldsymbol{x}_T$ from $t = T$ to $t = 0$ results in a sample from $p(\boldsymbol{x})$. Moreover, to each reverse-SDE, there exists a corresponding deterministic reverse-ODE

$$d\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - (1/2) \cdot g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right]\,dt \qquad (4)$$

which also can be integrated from $t = T$ to $t = 0$ to produce samples from $p(\boldsymbol{x})$.

Diffusion models generate data by simulating the reverse of the diffusion process, i.e., by solving the reverse-S/ODE of the diffusion process. Initial works on diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) used computationally expensive ancestral sampling to solve the reverse differential equations. Later works discovered that using adaptive numerical integrators to solve the reverse-S/ODE could accelerate the sampling process. This led to great attention on developing better reverse-S/ODE integrators (Jolicoeur-Martineau et al., 2021; Song et al., 2021b; Lu et al., 2022; Karras et al., 2022; Zhang & Chen, 2022). Our work is orthogonal to such works as we focus on finding good initialization points for integration via MCMC. Hence, a better integration technique directly translates to even better generative performance when plugged into Denoising MCMC.

Variance exploding (VE) diffusion model. A VE diffusion model considers the diffusion process

$$d\boldsymbol{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\boldsymbol{w} \tag{5}$$

where $\sigma(t)$ increases from $\sigma_{\min} = \sigma(0)$ to $\sigma_{\max} = \sigma(T)$. Distribution of $\boldsymbol{x}(t)$ evolves as

$$p_t(\boldsymbol{x}) = \textstyle\int p_{\sigma(t)}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}} \tag{6}$$

so if $\sigma_{\min}$ is sufficiently small, $p_0(\boldsymbol{x}) \approx p(\boldsymbol{x})$, and if $\sigma_{\max}$ is sufficiently large, so variance explodes, $p_T(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \sigma^2_{\max}\boldsymbol{I})$. If we have a score model $\boldsymbol{s}_\theta(\boldsymbol{x}, \sigma)$, $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \approx \boldsymbol{s}_\theta(\boldsymbol{x}, \sigma(t))$. It follows that with $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \sigma^2_{\max}\boldsymbol{I})$, we may integrate the reverse-S/ODE corresponding to Eq. (5) with $\boldsymbol{x}(T) = \boldsymbol{x}_T$ from $t = T$ to $t = 0$ using a score model to generate data.

Variance preserving (VP) diffusion model. A VP diffusion model considers the diffusion process

$$d\boldsymbol{x} = -\frac{1}{2}\beta(t)\boldsymbol{x}\, dt + \sqrt{\beta(t)}\, d\boldsymbol{w}$$

where $\beta(t)$ increases from 0 to 1. VE and VP diffusion are equivalent in the sense that there is a change of variables that allow us transform a VE diffusion to VP diffusion, and vice versa. Thus, we may use VE score function to generate data according to VP reverse-S/ODE, and VP score function to generate data with VE reverse-S/ODE. For more details, we refer the reader to Section 2 of Karras et al. (2022).

3. Denoising MCMC (DMCMC)

We now develop a general framework called Denoising MCMC (DMCMC) which combines MCMC with reverse-S/ODE integrators. By the equivalence of VE and VP diffusion, we may assume we are working with VE diffusion.

We denote the data space as $\mathcal{X} \subseteq \mathbb{R}^d$ and the noise level space as $\mathcal{S} = [\sigma_{\min}, \sigma_{\max}]$. The construction of DMCMC is comprised of two steps. In the first step, we build MCMC to generate initialization points in the product space $\mathcal{X} \times \mathcal{S}$, i.e., $\mathcal{X}$ augmented by the smoothness parameter σ. Since $\sigma(t)$ is a monotone increasing function, this is equivalent to augmenting the data space with diffusion time t. In the second step, we incorporate denoising steps, where we denoise the generated initialization points via reverse-S/ODE integrators.

Step 1: MCMC on $\mathcal{X} \times \mathcal{S}$. Suppose $p(\boldsymbol{x})$ is a high-dimensional multimodal distribution, supported on a low-dimensional manifold. If the modes are separated by wide low-density regions, MCMC can have difficulty moving between the modes. Intuitively, for MCMC to move between disjoint modes, the Markov chain would have to step off the data manifold. Once MCMC leaves the data manifold, the density or the score vanishes. Then, most random directions produced by the proposal distribution do not point to the manifold. Thus, MCMC gets lost in the ambient space, whose volume grows exponentially in d.

Annealing via Gaussian smoothing, used in both ALD and VE diffusion, circumvents this problem. As $p(\boldsymbol{x})$ smoothed with perturbation kernel $p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ of increasing σ, the modes of $p(\boldsymbol{x})$ grow wider and start to connect. Thus, MCMC can easily transition between modes. However, running MCMC in the manner of ALD is inefficient since we do not know how many iterations within each noise level is sufficient. To address this problem, we propose to augment $\mathcal{X}$ with the smoothness scale σ and run MCMC in the product space $\mathcal{X} \times \mathcal{S}$ such that MCMC automatically controls the value of σ. Below, we formally describe MCMC on $\mathcal{X} \times \mathcal{S}$.

Let us define the σ-conditional distribution

$$\hat{p}(\boldsymbol{x} \mid \sigma) := \textstyle\int p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}}. \tag{7}$$

We also define a prior $\hat{p}(\sigma)$ on $\mathcal{S}$. Then by the Bayes' Rule,

$$\hat{p}(\boldsymbol{x}, \sigma) = \hat{p}(\boldsymbol{x} \mid \sigma) \cdot \hat{p}(\sigma). \tag{8}$$

Then, MCMC with $\hat{p}(\boldsymbol{x}, \sigma)$ will produce samples $\{(\boldsymbol{x}_n, \sigma_n)\}$ in $\mathcal{X} \times \mathcal{S}$ such that

$$\sigma_n \sim \hat{p}(\sigma), \qquad \boldsymbol{x}_n \sim \hat{p}(\boldsymbol{x} \mid \sigma_n). \tag{9}$$

A specific instance of MCMC sampling used in our paper, called the denoising Langevin Gibbs sampling, will be described in detail in Section 4.

Note that if $\sigma_n \gg \sigma_{\min}$, $\boldsymbol{x}_n$ will be a noisy sample, i.e., a sample corrupted with Gaussian noise of variance σ_n^2, and if $\sigma_n \approx \sigma_{\min}$, $\boldsymbol{x}_n$ will resemble a sample from $p(\boldsymbol{x})$. Since our goal is to generate samples from $p(\boldsymbol{x})$, naïvely, we could keep samples $(\boldsymbol{x}_n, \sigma_n)$ with $\sigma_n \approx \sigma_{\min}$ and discard other samples. However, this could lead to a large waste of computation resources. In the following step, we incorporate reverse-S/ODE integrators to avert this problem.

Step 2: Incorporating denoising steps. Let us recall that integrating the reverse-S/ODE for the VE diffusion SDE Eq. (5) from time $t = T$ to $t = 0$ sends samples from $p_T(\boldsymbol{x})$ to samples from $p_0(\boldsymbol{x}) \approx p(\boldsymbol{x})$. In general, integrating the reverse-SDE or ODE from time $t = t_2$ to $t = t_1$ for $t_1 < t_2$ sends samples from $p_{t_2}(\boldsymbol{x})$ to samples from $p_{t_1}(\boldsymbol{x})$ (Song et al., 2021b). We use this fact to denoise MCMC samples from $\hat{p}(\boldsymbol{x}, \sigma)$.

Suppose we are given a sample $(\boldsymbol{x}_n, \sigma_n) \sim \hat{p}(\boldsymbol{x}, \sigma)$. With $t_n := \sigma^{-1}(\sigma_n)$, Eq. (9) tells us

$$\boldsymbol{x}_n \sim p_{t_n}(\boldsymbol{x}) \tag{10}$$

so integrating the reverse-S/ODE with initial condition $\boldsymbol{x}(t_n) = \boldsymbol{x}_n$ from $t = t_n$ to $t = 0$ produces a sample from $p_0(\boldsymbol{x}) \approx p(\boldsymbol{x})$. Here, we note that any reverse-S/ODE solver may be used to carry out the integration. In Appendix F.2, we show the necessity of the denoising step.

3.1. Important Remarks

Origin of acceleration. Let us explain in detail how DMCMC accelerates sampling. Given an MCMC chain $\{(\boldsymbol{x}_n, \sigma_n)\}$ in $\mathcal{X} \times \mathcal{S}$ and a prior $\hat{p}(\sigma)$ which places high mass near $\sigma_{\min}$, MCMC traverses high probability regions, so $\sigma_n \ll \sigma_{\max}$ for most n, i.e., $t_n \ll T$ for most n. This also means the sequence $\{\boldsymbol{x}_n\}$ generally stays close to the data manifold. So, the average length of integration intervals $(0, t_n)$ will tend to be much shorter than T. Thus, integrating the reverse-S/ODE over $(0, t_n)$ to denoise $\boldsymbol{x}_n$ is much faster than integrating the reverse-S/ODE over $(0, T)$, i.e., standard diffusion sampling. This idea is illustrated in Figure 1, and a longer explanation is given in Appendix E.1.

Predictor-Corrector vs. DMCMC. Note that while the predictor-corrector (PC) method (Song et al., 2021b) also uses MCMC, DMCMC and PC contribute to orthogonal aspects of diffusion sampling. PC uses MCMC to improve the denoising process, not the initialization points, so DMCMC can improve PC as well (Figure 15). More discussion on the differences between DMCMC and PC is in Appendix D.

DMCMC with VP score function. If we are given a VP score function, we apply the same algorithm described in this section, with one difference: when we evaluate the score function, we change VE variables into VP variables, using the transformation described in Section 2 of Karras et al. (2022). The validity of DMCMC with VP diffusion is shown in Sections 5.2 and 5.3.

4. Denoising Langevin Gibbs (DLG)

In Section 3, we described an abstract framework, DMCMC, for accelerating score-based sampling by combining MCMC and reverse-S/ODE integrators. We now develop a concrete instance of DMCMC. As the second construction step of DMCMC is simple, we only describe the first step.

Naïvely, we could extend denoising score matching to estimate the score $\hat{s}_\theta(\boldsymbol{x}, \sigma) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \times \mathbb{R}$ of $\hat{p}(\boldsymbol{x}, \sigma)$ and apply Langevin dynamics in the first step of DMCMC. But, this would prevent us from using pre-trained score models, as we would have to solve (for some small $\nu > 0$)

$$\min_\theta \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\nu^2\boldsymbol{I})\hat{p}(\boldsymbol{x},\sigma)}[\|\hat{s}_\theta(\boldsymbol{x} - \epsilon_{1:d}, \sigma - \epsilon_{d+1}) - \boldsymbol{\epsilon}/\nu^2\|_2^2]$$

Gibbs sampling provides a simple path around this problem. Let us recall that given a previous MCMC iterate $(\boldsymbol{x}_n, \sigma_n)$, Gibbs sampling proceeds by alternating between an $\boldsymbol{x}$ update step $\boldsymbol{x}_{n+1} \sim \hat{p}(\boldsymbol{x} \mid \sigma_n)$ and a σ update step $\sigma_{n+1} \sim \hat{p}(\sigma \mid \boldsymbol{x}_{n+1})$. Below, we describe our score-based sampling algorithm, Denoising Langevin Gibbs (DLG), whose pseudocode is given in Appendix C.

Updating $\boldsymbol{x}$. Suppose we are given an MCMC iterate $(\boldsymbol{x}_n, \sigma_n)$ and a score model $s_\theta(\boldsymbol{x}, \sigma)$. We generate $\boldsymbol{x}_{n+1}$ by a Langevin step on $\hat{p}(\boldsymbol{x} \mid \sigma_n)$. Specifically, by Eq. (7),

$$\nabla_{\boldsymbol{x}} \log \hat{p}(\boldsymbol{x} \mid \sigma_n) \approx s_\theta(\boldsymbol{x}, \sigma_n) \tag{11}$$

so a Langevin update on $\boldsymbol{x}$, according to Eq. (1) is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (\eta/2) \cdot s_\theta(\boldsymbol{x}_n, \sigma_n) + \sqrt{\eta} \cdot \boldsymbol{\epsilon} \tag{12}$$

for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Here, we call η the step size.

Updating σ. We have $\boldsymbol{x}_{n+1}$ and need to sample $\sigma_{n+1} \sim \hat{p}(\sigma \mid \boldsymbol{x}_{n+1})$. To this end, we first train a DNN noise level classifier $q_\phi(\sigma \mid \boldsymbol{x})$ to approximate $\hat{p}(\sigma \mid \boldsymbol{x})$ by

$$\max_\phi \mathbb{E}_{\hat{p}(\boldsymbol{x},\sigma)}[\log q_\phi(\sigma \mid \boldsymbol{x})]. \tag{13}$$

Specifically, we discretize $[\sigma_{\min}, \sigma_{\max}]$ into M levels $\tau_1 = \sigma_{\min} < \tau_2 < \cdots < \tau_M = \sigma_{\max}$. Given τ_m where $1 \leq m \leq M$, m serves as the label and clean training data corrupted by Gaussian noise of variance τ_m^2 serves as the classifier input. The classifier is trained to predict m by minimizing the cross entropy loss. Having trained a noise level classifier, we sample σ_{n+1} by drawing an index m according to the classifier output probability for $\boldsymbol{x}_{n+1}$ and setting $\sigma_{n+1} = \tau_m$. In practice, using the index of largest probability worked fine. We denote this process as $\sigma_{n+1} \sim q_\phi(\sigma \mid \boldsymbol{x}_{n+1})$. In Section 5.5, we verify that DLG with the approximated conditional works as intended.

We note other works on diffusion sampling have also considered noise level classifiers (Nichol & Dhariwal, 2021; Roman et al., 2021). Yet, such works focus on the denoising process. Hence, we emphasize that the works do not overlap with the contributions of DMCMC (see Appendix D.2).

4.1. Practical Considerations

Computation cost of σ prediction. We found that using shallow neural networks for the noise classifier q_ϕ was sufficient to accelerate sampling. Concretely, using a neural net with four convolution layers and one fully connected layer as the classifier, one evaluation of q_ϕ was around $100 \sim 1000$ times faster than one evaluation of the score model $\boldsymbol{s}_\theta$. So,

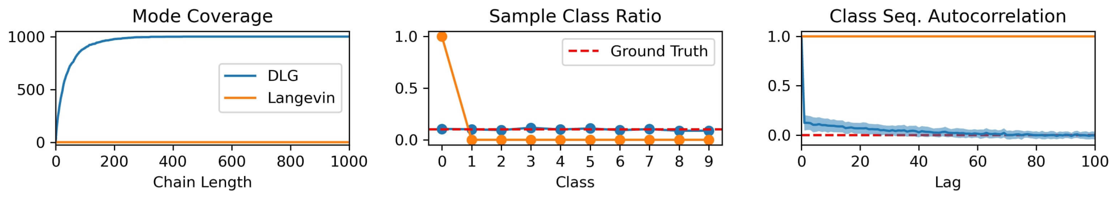


Figure 2: Ablation study of σ update step in DLG.

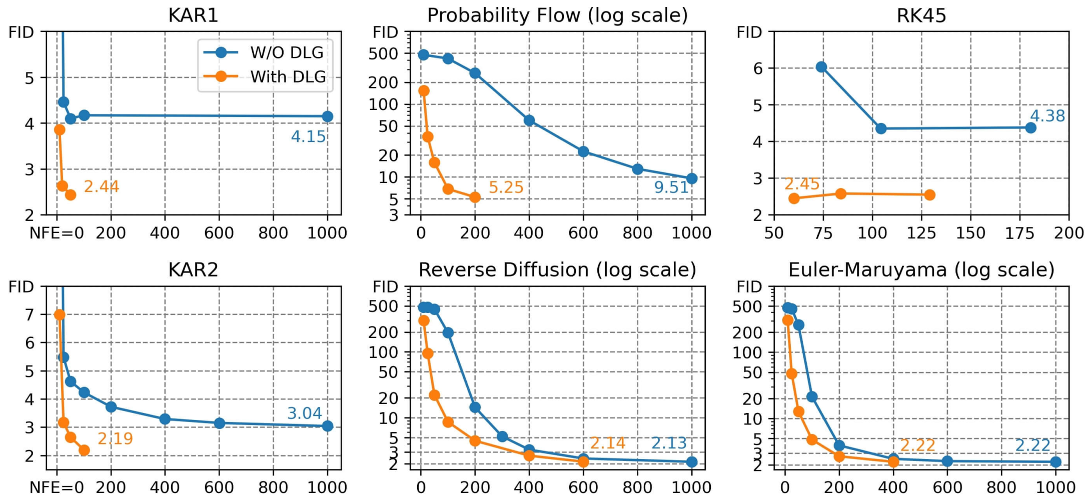


Figure 3: Sampling acceleration of DLG on CIFAR10 with VE score function of Song et al. (2021b). FID of notable points are written in the corresponding color. **Top row**: deterministic integrators. **Bottom row**: stochastic integrators.

when comparing sampling methods, we only count the number of score function evaluations (NFE). We also note that the training time q_ϕ was negligible compared to the training time of s_θ. For instance, on CelebA-HQ-256, training q_ϕ with the aforementioned architecture for 100 epochs took around 15 minutes on an RTX 2080 Ti.

Starting points for DLG. Theoretically, an MCMC chain $\{(\boldsymbol{x}_n, \sigma_n)\}_{n=0}^{\infty}$ will converge to $\hat{p}(\boldsymbol{x}, \sigma)$ regardless of the starting point $(\boldsymbol{x}_0, \sigma_0)$. However, theory for log-concave densities shows that setting starting points close to the stationary distribution, i.e., using "warm start", can significantly accelerate convergence of the Markov chain (Dalalyan, 2017; Dwivedi et al., 2019). Inspired by this observation, we set $\boldsymbol{x}_0$ by generating a clean data sample with a reverse-S/ODE solver starting from prior noise, adding some Gaussian noise to the clean data sample, and running Gibbs sampling for a few iterations. Pseudocode is shown in Appendix C. The NFE involved in generating $\boldsymbol{x}_0$ is included in the final per-sample average NFE computation for DLG when comparing methods in Section 5. But, we note that this cost vanishes in the limit of infinite sample size.

Reducing autocorrelation. Autocorrelation in MCMC chains, i.e., correlation between consecutive samples in the MCMC chain, could reduce the sample diversity of MCMC. A typical technique to reduce autocorrelation is to use every n_{skip}-th samples of the MCMC chain for some $n_{skip} > 1$. For DMCMC, this means we denoise every n_{skip}-th sample. So, if we use n_{den} NFE to denoise MCMC samples, the average NFE for generating a single sample is around $n_{skip} + n_{den}$.

Choosing iterates to apply denoising. The MCMC chain can be partitioned into blocks of n_{skip} consecutive samples. Using every n_{skip}-th sample of the MCMC chain corresponds to denoising the last iterate of each block. Instead, to further shorten the length of integration, within each block, we apply denoising to the sample of minimum σ.

Choice of prior $\hat{p}(\sigma)$. We use $\hat{p}(\sigma) \propto 1/\sigma$ to drive the MCMC chain towards small values of σ. However, we note that increasing the sharpness of $\hat{p}(\sigma)$ on smaller values of σ can slow down the convergence speed of MCMC. Hence, naïvely choosing $\hat{p}(\sigma)$ to assign large mass to small values of σ may not be a good strategy. More detail is given in In Appendix E.2.

5. Experiments

5.1. Mixing of DMCMC Chains

Here we provide experimental proof that DMCMC is capable of visiting diverse modes as a consequence of running

MCMC in the product space $\mathcal{X} \times \mathcal{S}$. To this end, we compare DLG with and without σ updates. DLG without σ updates is just Langevin dynamics at fixed σ, so we run fifty Langevin dynamics chains and fifty DLG chains on a mixture of Gaussians (MoG) with $1k$ modes at CIFAR10 images. All chains are initialized at a single mode. For each method, we compute the mode coverage of the samples, the class distribution of the samples, and the autocorrelation of sample image class sequence. Since the noise conditional score function can be calculated analytically for MoGs, this setting decouples sampler performance from score model approximation error.

Figure 2 shows the results. In the left panel, we see that Langevin dynamics is unable to escape the initial mode. Increasing the step size η of Langevin dynamics caused the chain to diverge. On the other hand, DLG successfully captures all modes. DLG samples cover all $1k$ modes at chain length 432. Middle panel provides evidence that DLG samples correctly reflect the statistics of the data distribution. Finally, the right panel indicates that the DLG chain moves freely between classes, i.e., distant modes. These observations validate our claim that DLG mixes well.

We also show in Figure 4 a DLG chain on CelebA-HQ-256 generated with a score network. The chain progresses from left to right, from right end to left end of row below. We can see that the chain transitions between diverse attributes such as gender, hair color, skin color, glasses, facial expression, posture, etc. while maintaining high image quality.

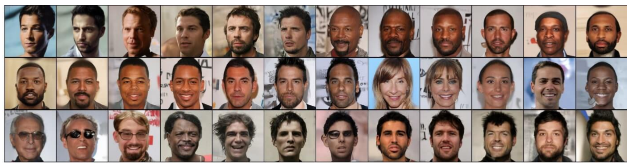


Figure 4: A DLG chain on CelebA-HQ-256.

5.2. Accelerating Image Generation

We compare various integrators with and without DLG on CIFAR10, CelebA-HQ-256, and FFHQ-1024 image generation. The deterministic integrators are: the deterministic integrator of Karras et al. (2022) (KAR1), the probability flow integrator of Song et al. (2021b), and the RK45 solver. The stochastic integrators are: the stochastic integrator of Karras et al. (2022) (KAR2), the reverse diffusion integrator of Song et al. (2021b), and the Euler-Maruyama method. We use the Fréchet Inception Distance (FID) (Heusel et al., 2017) to measure sample quality. For CIFAR10, we generate $50k$ samples, for CelebA-HQ-256, we generate $10k$ samples, and for FFHQ-1024, we generate $5k$ samples. We use pre-trained score models of Song et al. (2021b) and Karras et al. (2022).

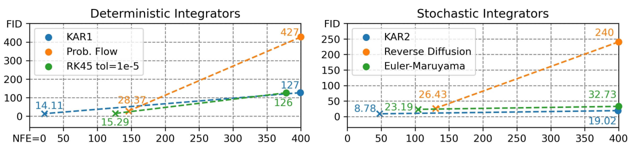
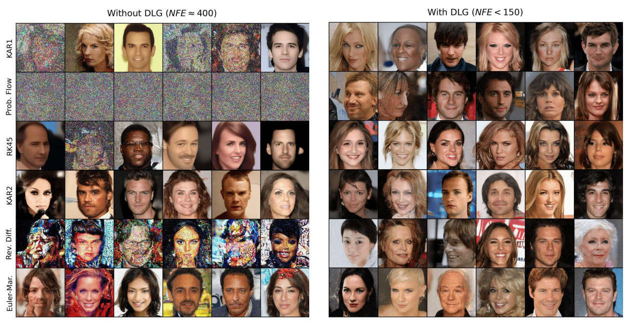
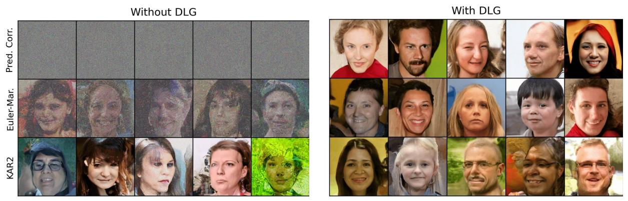


Figure 5: Sampling acceleration of DLG on CelebA-HQ-256 using VE score function of Song et al. (2021b). A dot indicates an integrator without DLG, and a cross of the same color indicates the integrator with DLG. Dotted lines indicate performance improvement due to DLG.



(a) CelebA-HQ-256



(b) FFHQ-1024

Figure 6: Non-cherry-picked samples on high-resolution data. Each row shows samples for an integrator without (left col.) and with (right col.) DLG.

CIFAR10. In Figure 3, we make two important observations. First, DLG successfully accelerates all six integrators by a non-trivial margin. In particular, if an integrator without DLG already performs well, the integrator combined with DLG outperforms other integrators combined with DLG. For instance, compare the results for KAR1 with those of other deterministic integrators. Second, DLG improves the performance lower bound for some deterministic integrators. While KAR1 and RK45 saturates at around 4 FID, KAR1 and RK45 with DLG achieve around 2.4 FID.

CelebA-HQ-256. Figure 5 shows the results on CelebA-HQ-256. We observe that DLG improves computational efficiency and sample quality simultaneously. Indeed, in Figure 6a, we observe remarkable improvements in sample quality despite using fewer NFE. This demonstrates the scalability of DLG to generating high-resolution images. We also note that we did not perform an exhaustive search of DLG hyper-parameters for CelebA-HQ-256.

FFHQ-1024. To demonstrate the scalability of DLG, we generate FFHQ images of resolution 1024×1024. Using the VE score function of Song et al. (2021b), PC sampler at NFE 160 achieves 480 FID without DLG, and 45 FID with DLG. EM sampler at NFE 160 achieves 353 FID without DLG, and 52.61 FID with DLG. KAR2 sampler at NFE 51 achieves 56 FID without DLG, and 36 FID with DLG. We again observe improved sample quality in Figure 6b.

Achieving SOTA. DLG combined with KAR1 sets a new SOTA record for CIFAR10 in the limited number of NFE setting: 3.25 FID with 10 NFE and 2.49 FID with 16 NFE which beats the results of Zhang & Chen (2022), 4.17 FID with 10 NFE and 2.86 FID with 20 NFE (Table 1). DLG combined with KAR2 sets a new record on CelebA-HQ-256 among score-based models: 6.99 FID with 158.96 NFE which beats the current best result of Kim et al. (2022a), 7.16 FID with 4000 NFE.

Method	NFE 10	NFE 20	NFE 50
DPM-Solver-2 (VP)	5.28 (+2 NFE)	3.02 (+4 NFE)	2.69 (−2 NFE)
DPM-Solver-3 (VP)	6.03 (+2 NFE)	2.75 (+4 NFE)	2.65 (−2 NFE)
DEIS (VP)	4.17 (+0 NFE)	2.86 (+0 NFE)	2.57 (+0 NFE)
DEIS (VE)	20.89 (+0 NFE)	16.59 (+0 NFE)	16.31 (+0 NFE)
KAR1 (VP)	9.70 (+1 NFE)	3.23 (+5 NFE)	2.97 (+1 NFE)
KAR1 (VE)	14.12 (+1 NFE)	4.46 (+5 NFE)	4.1 (+1 NFE)
DLG+KAR1 (VP)	**3.25** (+0.1 NFE)	**2.49** (−3.9 NFE)	<u>2.49</u> (−33.9 NFE)
DLG+KAR1 (VE)	<u>3.86</u> (+0.1 NFE)	<u>2.63</u> (+0.1 NFE)	**2.45** (−0.9 NFE)

Table 1: Comparison of fast samplers on CIFAR10 in terms of FID. Number in parenthesis indicates extra or less NFE used. Best numbers are bolded, second best are underlined.

We also note there are works which distill diffusion models for rapid sample generation (Salimans & Ho, 2022; Meng et al., 2023). Our method is faster than the method of Meng et al. (2023), which has 2.78 FID with 16 NFE, but is slower than the method of Salimans & Ho (2022), which has 2.57 FID with 8 NFE.

However, distillation methods require longer training than conventional diffusion models. This makes score model distillation unavailable for practitioners with limited computation resources, while our method is available for such practitioners as well. We believe this difference must be considered when comparing DLG and distillation methods. Specifically, given that there is only a marginal FID difference between our method and distillation methods despite distillation methods having spent significantly more compute resources in training score models, we believe the contribution of our paper is non-trivial.

5.3. Accelerating Conditional Generation

We also study whether DLG can accelerate conditional image generation. In Table 2, we indeed observe DLG is capable of accelerating class-conditional image generation with score networks provided by Karras et al. (2022). In fact, we observe acceleration regardless of whether we use VE or VP score networks.

Class	0	1	2	3	4	5	6	7	8	9
No DLG	16.3	13	18	19.5	16.6	19.5	17.7	15.1	12.9	13.5
WIth DLG	13.8	10.3	15.7	16.4	13.2	15.6	13.9	11.9	10.8	9.9

(a) Using VE score network

Class	0	1	2	3	4	5	6	7	8	9
No DLG	14.3	11.6	15.8	17.7	14.7	16.9	16.0	13.4	11.1	11.3
WIth DLG	12.2	9.3	13.5	14.8	11.6	13.6	12.7	10.6	9.3	8.5

(b) Using VP score network

Table 2: KAR1 acceleration of DLG on class-conditional CIFAR10 generation (FID).

5.4. Hyper-parameter Ablation Study

Given an integrator, DLG is determined by total NFE per sample n, NFE spent on denoising samples n_{den}, and Langevin dynamics step size η. We fix the integrator to be KAR1 and observe the effect of each component on CIFAR10 image generation.

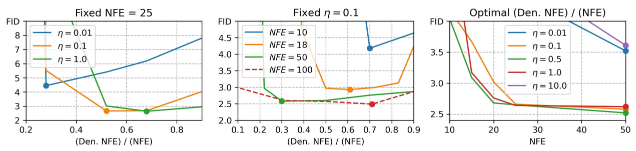


Figure 7: Ablation study of DLG with KAR1. Dots indicate the points of lowest FID.

η vs. n_{den}/n. In the left panel of Figure 7, we fix NFE and vary η and n_{den}/n. n_{den} governs individual sample quality, and $n_{skip} = n - n_{den}$ governs sample diversity. Thus, we observe optimal FID is achieved when n_{den}/n has intermediate values, not extreme values near 0 or 1. Also, lower n_{den}/n is needed to attain optimality for lower η. This is because lower η means the MCMC chain travels closer to the image manifold at the cost of slower mixing.

NFE vs. n_{den}/n. In the middle panel of Figure 7, we fix η and vary NFE and n_{den}/n. We observe three trends. First, in the small NFE regime, where $10 \leq$ NFE ≤ 50, it is beneficial to decrease n_{den}/n as NFE increases. Second, in the large NFE regime, where NFE > 50, it is beneficial to increase n_{den}/n as NFE increases. This is because if n_{skip} is sufficiently large, MCMC chain starts producing essentially independent samples, so increasing n_{skip} further provides no gain. Third, as we increase NFE, the set of n_{den}/n which provides near-optimal performance becomes larger. So, a reasonable strategy for choosing n_{den} given η and NFE budget n is to find smallest n_{skip} which produces visually distinct samples, and allocate $n_{den} = n - n_{skip}$.

η vs. NFE. In the right panel of Figure 7, we choose optimal (in terms of FID) n_{den}/n for each combination of η and NFE. We see choosing overly small or large η leads to performance degradation. If η is within a certain range, e.g, $\eta \in [0.05, 1.0]$, we obtain similarly good performance.

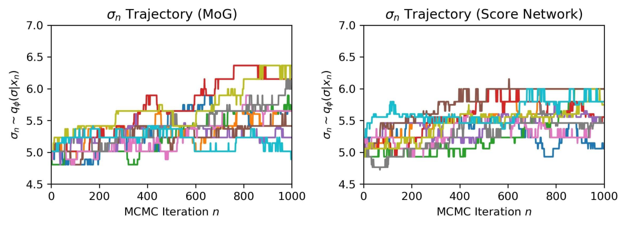


Figure 8: Visualization of σ_n trajectories.

5.5. Analysis of the σ_n Trajectory

In Figure 8, we have visualized Langevin Gibbs trajectories of σ in the MoG setting with $1k$ modes at CIFAR10 images and the score network setting. We indeed observe that σ moves up and down, allowing $\boldsymbol{x}$ to travel between disjoint modes. Moreover, in Figure 2, $\boldsymbol{x}$ samples cover all $1k$ modes of the MoG. This experimentally proves $\boldsymbol{x}$ sequence of DLG is exploring the entire image distribution.

In Appendix F.1, we explain how each σ_n is an (scaled) estimate of the distance of $\boldsymbol{x}_n$ from the image manifold. Hence, the noise predictor network q_ϕ is, in some sense, predicting the optimal σ_n such that the score network $s_\theta(\boldsymbol{x}, \sigma_n)$ provides an accurate gradient estimate at $\boldsymbol{x}_n$.

6. Societal Impacts, Limitations, and Reproducibility

Societal impacts. DMCMC can reduce the need for heavy computation during diffusion sampling, leading to a decrease in resource consumption. Yet, diffusion models may also be used to produce, e.g., Deepfakes or fake information, and this possibly calls for regulation.

Limitations. DMCMC is currently applied only to sample generation with unconditional or conditional score networks. We hope to extend DMCMC to guided diffusion settings, e.g., classifier-guided generation (Dhariwal & Nichol, 2021) or CLIP-guided generation (Kim et al., 2022b).

Reproducibility. We provide a GitHub link containing code and noise classifier checkpoints for our main experiments. `https://github.com/1202kbs/DMCMC` We have also added a pseudocode of DMCMC in Appendix C, and hyper-parameters are described in Appendix A.

7. Conclusion

In this work, we proposed DMCMC which combines MCMC with reverse-S/ODE integrators. This has led to improvements for both MCMC and diffusion models. For MCMC, DMCMC allows Markov chains to visit disjoint modes. For diffusion models, DMCMC accelerates sampling by reducing the average integration interval length of reverse-S/ODE. We developed a practical instance of DMCMC called DLG, and demonstrated the practicality and scalability of DLG through various experiments. In particular, DLG achieved state-of-the-art results on CIFAR10 and CelebA-HQ-256 among score-based models. Overall, our work opens up an orthogonal approach to accelerating score-based sampling.

Acknowledgements

This work was supported by the National Research Foundation of Korea under Grant NRF-2020R1A2B5B03001980, by the Korea Medical Device Development Fund grant funded by the Korea government (the Ministry of Science and ICT, the Ministry of Trade, Industry and Energy, the Ministry of Health & Welfare, the Ministry of Food and Drug Safety) (Project Number: 1711137899, KMDF_PR_20200901_0015), and Field-oriented Technology Development Project for Customs Administration through National Research Foundation of Korea funded by the Ministry of Science & ICT and Korea Customs Service (NRF-2021M3I1A1097938), by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT, Ministry of Science and ICT) (No. 2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation), (No.2019-0-00075, Artificial Intelligence Graduate School Program), and ITRC (Information Technology Research Center) support program (IITP-2021-2020-0-01461). This work was also supported by the KAIST Key Research Institute (Interdisciplinary Research Group) Project.

References

Cheng, X. and Bartlett, P. Convergence of Langevin MCMC in KL-divergence. In *Machine Learning Research, Volume 83: Algorithmic Learning Theory*, 2018.

Chung, H., Sim, B., Ryu, D., and Ye, J. C. Improving diffusion models for inverse problems using manifold constraints. In *NeurIPS*, 2022.

Dalalyan, A. S. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society*, 79(3):651–676, 2017.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Dwivedi, R., Chen, Y., J.Wainwright, M., and Yu, B. Log-concave sampling: Metropolis-Hastings algorithms are fast. *Journal of Machine Learning Research*, 20:1–42, 2019.

Geman, S. and Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

Geyer, C. J. and Thompson, E. A. Annealing Markov chain Monte Carlo with applications to ancestral inferences. *Journal of the American Statistical Association*, 90(431): 909–920, 1995.

Hastings, W. K. Monte Carlo sampling methods using Markov chains and their application. *Biometrika*, 57: 97–109, 1970.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Jolicoeur-Martineau, A., Li, K., Piché-Taillefer, R., Kachman, T., and Mitliagkas, I. Gotta go fast when generating data with score-based models. *arxiv preprint arXiv:2105.14080*, 2021.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Kim, D., Shin, S., Song, K., Kang, W., and Moon, I.-C. Soft truncation: A universal truncation technique of score-based diffusion for high precision score estimation. In *ICML*, 2022a.

Kim, G., Kwon, T., and Ye, J. C. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *CVPR*, 2022b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Langevin, P. On the theory of Brownian motion. 1908.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. Generalizing Hamiltonian Monte Carlo with neural networks. In *ICLR*, 2018.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ODE solver for diffusion probabilistic sampling in around 10 steps. *arxiv preprint arXiv:2206.00927*, 2022.

Meng, C., Rombach, R., Gao, R., Kingma, D. P., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. In *CVPR*, 2023.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21 (6):1087–1092, 1953.

Neal, R. M. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

Neal, R. M. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

Roman, R. S., Nachmani, E., and Wolf, L. Noise estimation for generative diffusion models. *arxiv preprint arXiv:2104.02600*, 2021.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learing using nonequilibrium thermodynamics. In *ICML*, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *ICLR*, 2021a.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

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

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

Stoer, J. and Bulisch, R. *Introduction to Numerical Analysis*, volume 12. Springer Science+Business Media New York, 2002.

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

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. 2011.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *arxiv preprint arXiv:2204.13902*, 2022.

A. Detailed Experiment Settings

Device. We use an RTX 2080 Ti or two Quadro RTX 6000 depending on the required VRAM.

Codes. For probability flow, RK45, reverse diffusion, and Euler-Maruyama integrators, we modify the code provided by Song et al. (2021b) in the GitHub repository `https://github.com/yang-song/score_sde_pytorch`. For KAR1 and KAR2, since Karras et al. (2022) did not release their implementation of the samplers, we used our implementation based on their paper. For evaluation, we use the FID implementation provided in the GitHub repository `https://github.com/mseitzer/pytorch-fid`.

Datasets. We use the CIFAR10 dataset (Krizhevsky, 2009), CelebA-HQ-256 dataset (Karras et al., 2018), and FFHQ-1024 dataset (Karras et al., 2019).

Data processing. All data are normalized into the range $[0, 1]$. Following Song et al. (2021b), for all methods, a denoising step using Tweedie's denoising formula is applied at the end of the sampling process.

Noise predictor network q_ϕ. The noise predictor network q_ϕ has four convolution layers followed by a fully connected layer. The convolution layers have channels 32, 64, 128, 256, and $(\sigma_{\min}, \sigma_{\max})$ is discretized into $1k$ points $\sigma_{\min}(\sigma_{\max}/\sigma_{\min})^t$ for t spaced evenly on $[0, 1]$. On CIFAR10, q_ϕ is trained for 200 epochs, and on CelebA-HQ-256, q_ϕ is trained for 100 epochs. We use the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.001.

Mixture of Gaussians. MoG has $1k$ modes at randomly sampled CIFAR10 training set images. For Langevin dynamics, we use step size $\eta = 0.0001$. Using a larger step size caused the Langevin dynamics chain to diverge. For DLG, we use step size $\eta = 1.0$, $n_{skip} = 1$, $n_{den} = 20$, and the reverse diffusion integrator. For each method, all chains were initialized at a single mode to test mixing capabilities in the worst-case scenario.

CIFAR10 image generation. We use a pre-trained NCSN++ (cont.) score model provided by Song et al. (2021b). For the baseline methods, we use the recommended settings. For DLG, the chain was initialized by generating samples with the deterministic integrator of Karras et al. (2022) using 37 NFE, adding Gaussian noise of variance 0.25, and running 20 iterations of Langevin-Gibbs. Table 3 lists the hyper-parameters for DLG and the corresponding FID (Figure 3).

KAR1				**Probability Flow**				**RK45**			
n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID
9	1	0.5	3.86	9	1	0.01	153.48	59.98	10	0.5	2.45
17	3	0.5	2.63	24	1	0.01	35.58	83.74	10	0.5	2.58
27	24	0.5	2.44	49	1	0.01	15.75	128.99	10	0.5	2.55
				90	10	0.01	6.8				
				190	10	0.05	5.25				

KAR2				**Reverse Diffusion**				**Euler-Maruyama**			
n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID
9	1	0.5	6.99	9	1	0.01	300.18	9	1	0.01	306.98
23	2	0.5	3.17	24	1	0.01	94.98	24	1	0.01	47.83
31	20	0.5	2.64	49	1	0.01	22.1	49	1	0.02	12.73
51	50	0.5	2.19	90	10	0.01	8.61	90	10	0.05	4.79
				190	10	0.1	4.45	190	10	0.1	2.69
				370	30	1.0	2.64	370	30	1.0	2.22
				570	30	1.0	2.14				

Table 3: DLG hyper-parameters and FID for integrators in Figure 3.

CelebA-HQ-256 image generation. We use a pre-trained NCSN++ (cont.) score model provided by Song et al. (2021b). For the baseline methods, we use the recommended settings. For DLG, the chain was initialized by generating samples with the stochastic integrator of Karras et al. (2022) using 37 NFE, adding Gaussian noise of variance 0.25, and running 70 iterations of Langevin-Gibbs. Table 4 lists the hyper-parameters for DLG and the corresponding FID (Figure 5).

FFHQ-1024 image generation. We use a pre-trained NCSN++ (cont.) score model provided by Song et al. (2021b). For the baseline methods, we use the recommended settings. For DLG, the chain was initialized by generating samples with the stochastic integrator of Karras et al. (2022) using 37 NFE, adding Gaussian noise of variance 25, and running 70 iterations of Langevin-Gibbs. Table 5 lists the hyper-parameters for DLG and the corresponding FID.

KAR1				**Probability Flow**				**RK45**			
n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID
17	4	0.8	14.11	140	5	0.25	28.37	125.95	4	0.8	15.29

KAR2				**Reverse Diffusion**				**Euler-Maruyama**			
n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID
37	10	0.8	8.78	100	25	0.1	26.43	100	4	0.8	23.19

Table 4: DLG hyper-parameters and FID for deterministic samplers in Figure 5.

Predictor Corrector				**Euler-Maruyama**				**KAR2**			
n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID	n_{den}	n_{skip}	η	FID
37	10	0.8	8.78	100	25	0.1	26.43	100	4	0.8	23.19

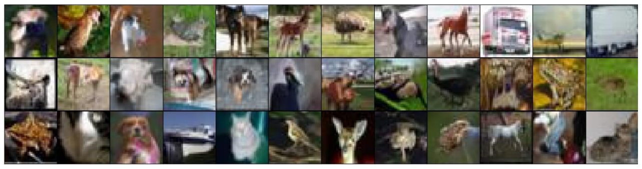
Table 5: DLG hyper-parameters and FID for FFHQ-1024 generation.

Achieving SOTA. On CIFAR10, we use KAR1 settings of Table 3. On CelebA-HQ-256, we use $n_{den} = 131$, $n_{skip} = 27$, $\eta = 4.0$, which achieves 6.99 FID.

B. Additional Samples

B.1. DLG Chain Visualization

Figure 9 shows DLG chains on CIFAR10 and CelebA-HQ-256. The chain progresses from left to right, from right end to left end of row below. On CIFAR10, we can see that the chain visits diverse classes. On CelebA-HQ-256, we can see that the chain transitions between diverse attributes such as gender, hair color, skin color, glasses, facial expression, posture, etc.


(a) CIFAR10.

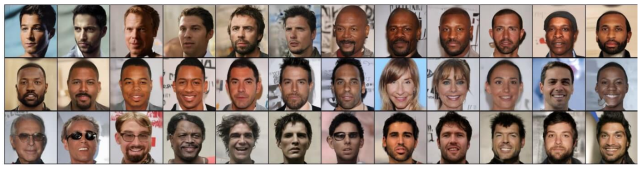

(b) CelebA-HQ-256

Figure 9: Visualization of denoised DLG chains.

B.2. Additional Unconditional Samples

In Figures 10 and 11, we show additional samples for CIFAR10 without and with DLG. In Figures 12 and 13, we show additional samples for CelebA-HQ-256 without and with DLG. In Figure 14, we show additional samples for FFHQ-1024 without and with DLG.

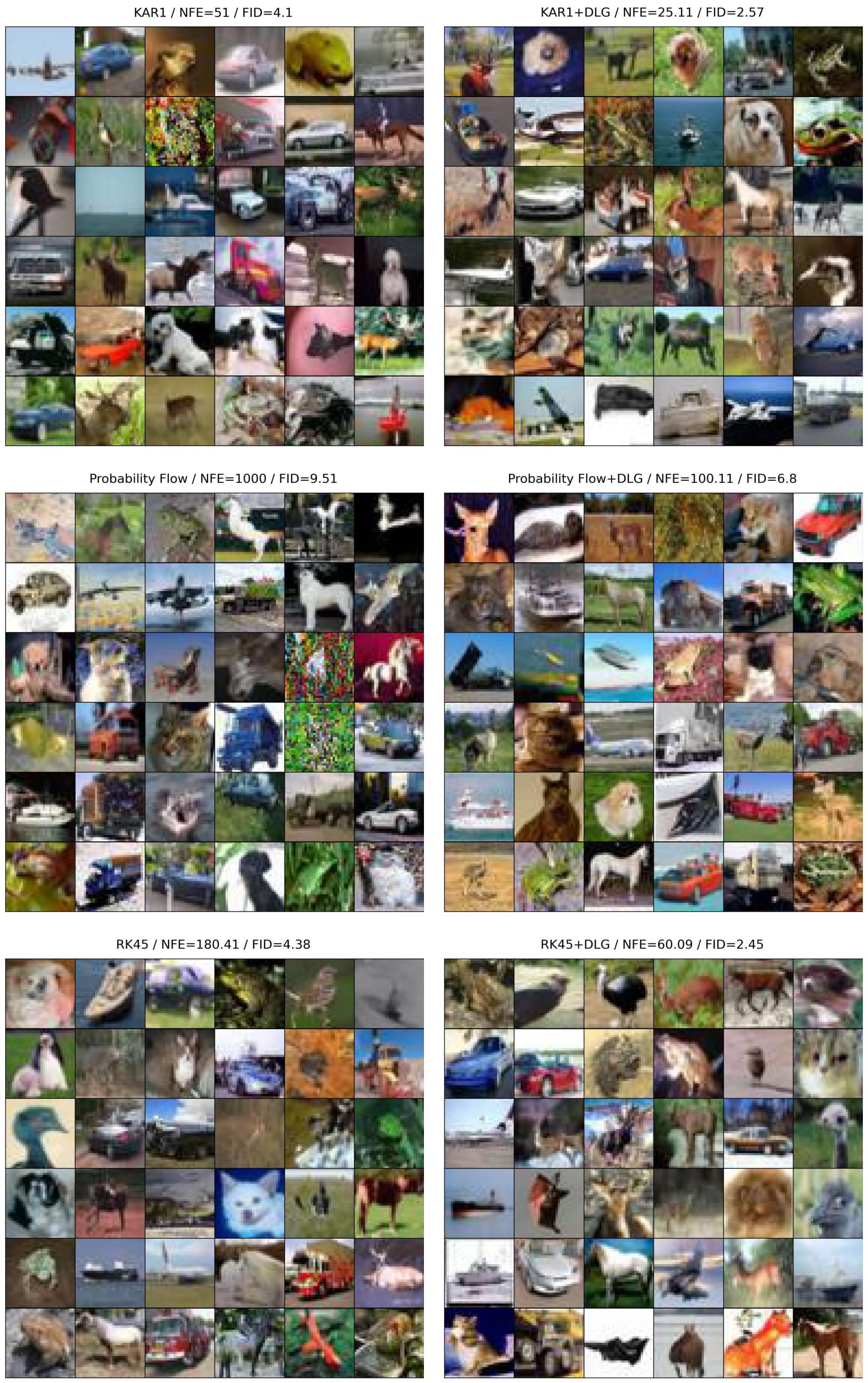

KAR1 / NFE=51 / FID=4.1
KAR1+DLG / NFE=25.11 / FID=2.57
Probability Flow / NFE=1000 / FID=9.51
Probability Flow+DLG / NFE=100.11 / FID=6.8
RK45 / NFE=180.41 / FID=4.38
RK45+DLG / NFE=60.09 / FID=2.45

Figure 10: Additional non-cherry-picked samples for deterministic integrators on CIFAR10 without (left col.) and with (right col.) DLG.

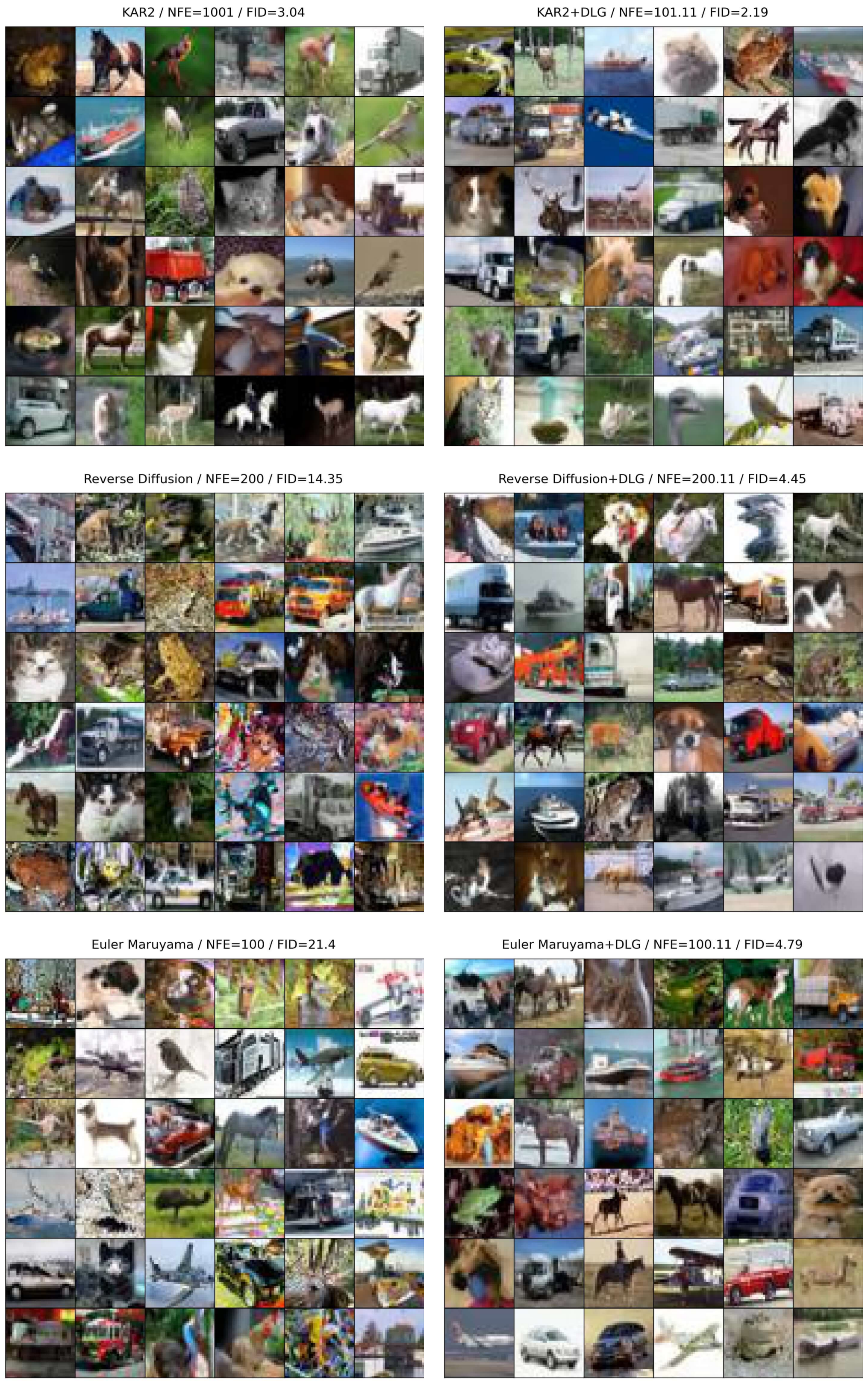

KAR2 / NFE=1001 / FID=3.04
KAR2+DLG / NFE=101.11 / FID=2.19
Reverse Diffusion / NFE=200 / FID=14.35
Reverse Diffusion+DLG / NFE=200.11 / FID=4.45
Euler Maruyama / NFE=100 / FID=21.4
Euler Maruyama+DLG / NFE=100.11 / FID=4.79

Figure 11: Additional non-cherry-picked samples for stochastic integrators on CIFAR10 without (left col.) and with (right col.) DLG.

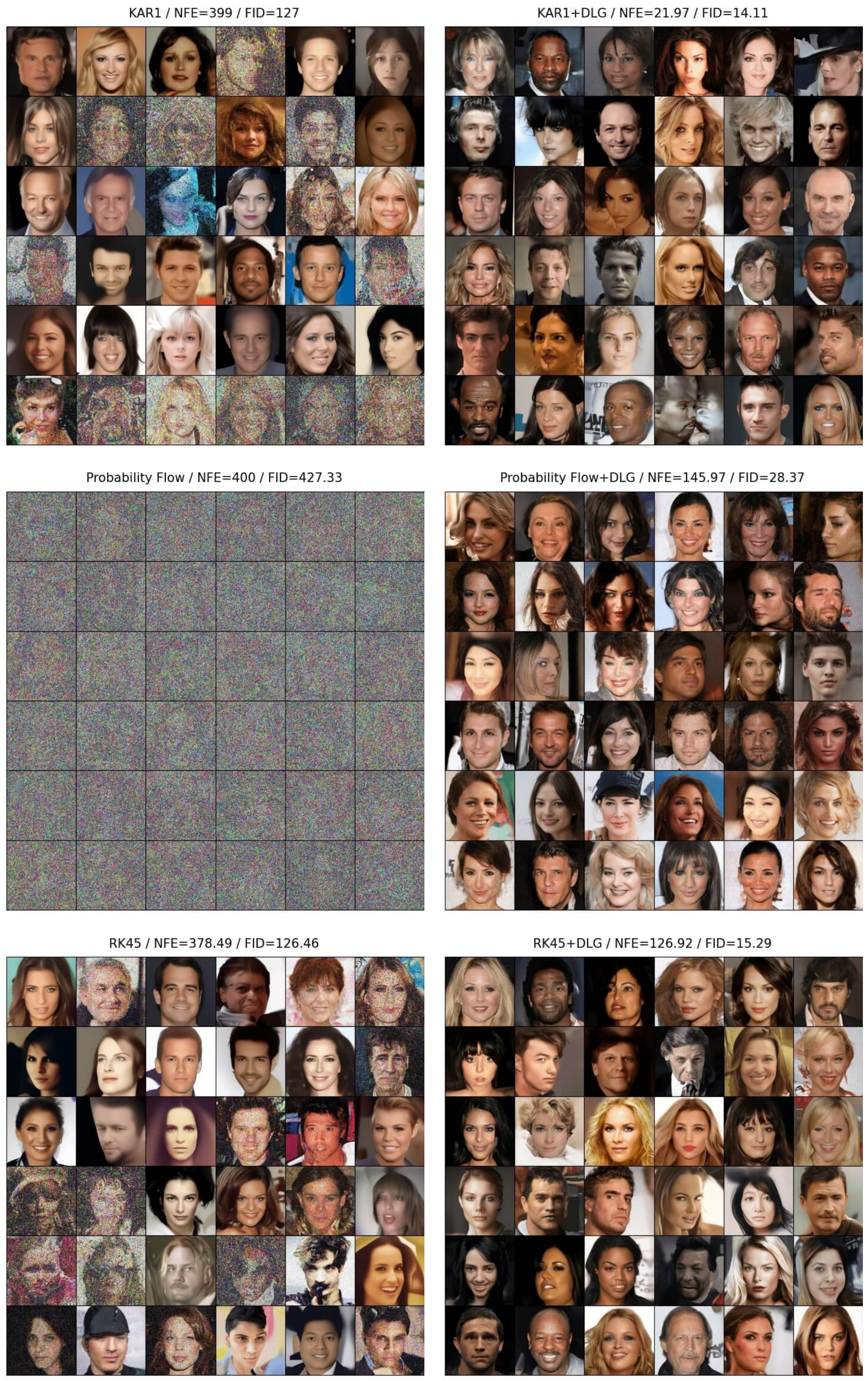

KAR1 / NFE=399 / FID=127
KAR1+DLG / NFE=21.97 / FID=14.11
Probability Flow / NFE=400 / FID=427.33
Probability Flow+DLG / NFE=145.97 / FID=28.37
RK45 / NFE=378.49 / FID=126.46
RK45+DLG / NFE=126.92 / FID=15.29

Figure 12: Additional non-cherry-picked samples for deterministic integrators on CelebA-HQ-256 without (left col.) and with (right col.) DLG.

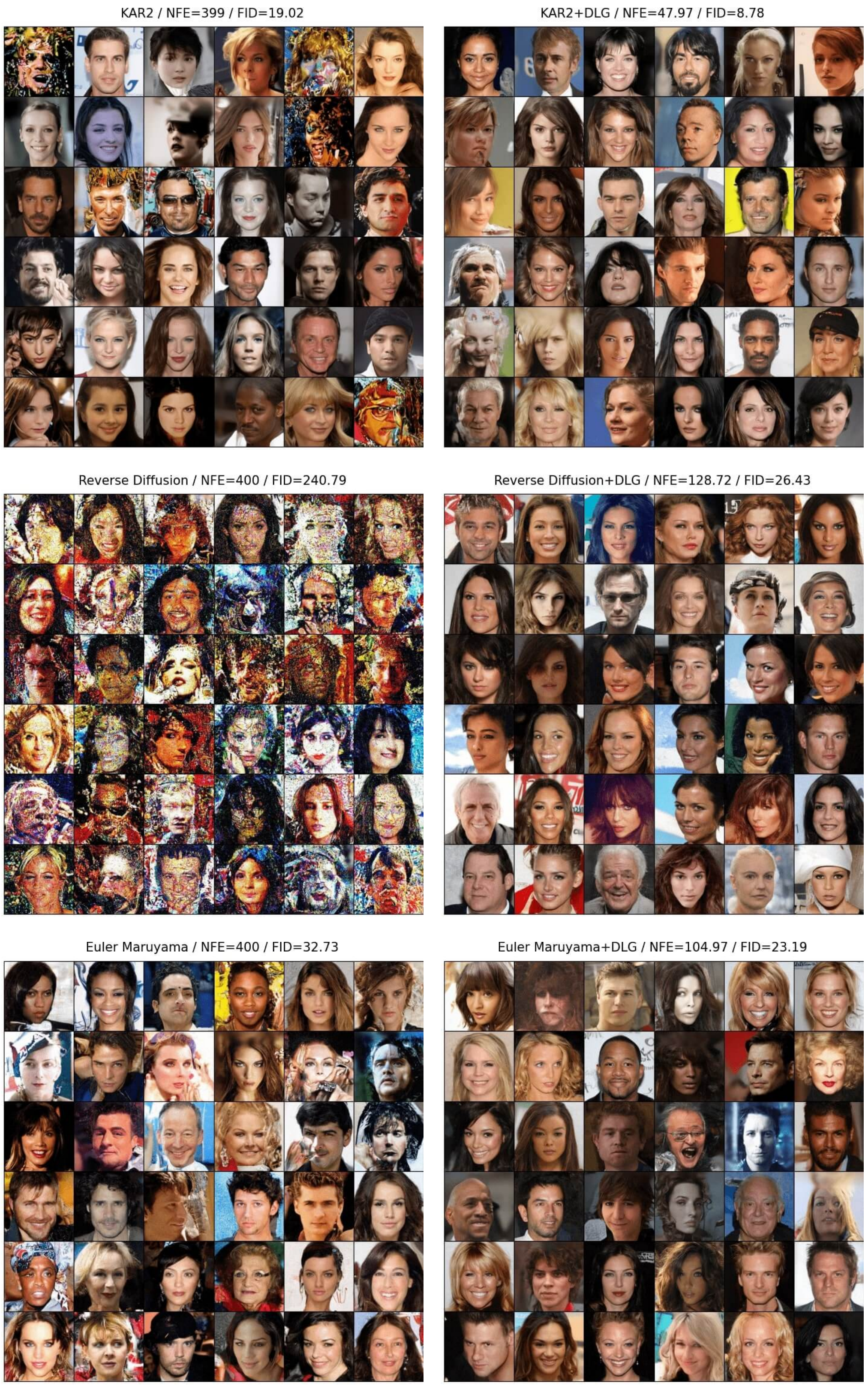

KAR2 / NFE=399 / FID=19.02
KAR2+DLG / NFE=47.97 / FID=8.78
Reverse Diffusion / NFE=400 / FID=240.79
Reverse Diffusion+DLG / NFE=128.72 / FID=26.43
Euler Maruyama / NFE=400 / FID=32.73
Euler Maruyama+DLG / NFE=104.97 / FID=23.19

Figure 13: Additional non-cherry-picked samples for stochastic integrators on CelebA-HQ-256 without (left col.) and with (right col.) DLG.

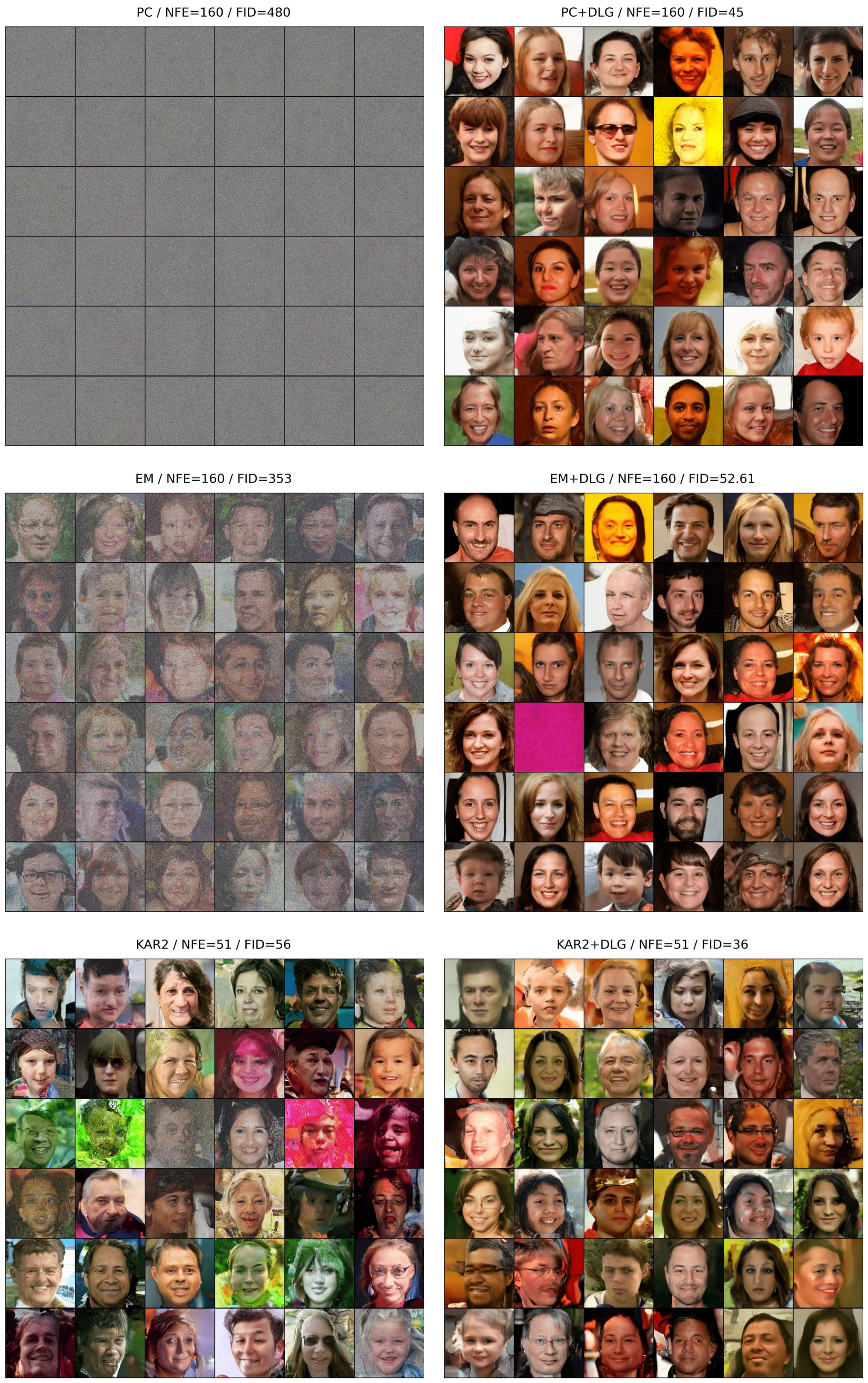

PC / NFE=160 / FID=480
PC+DLG / NFE=160 / FID=45
EM / NFE=160 / FID=353
EM+DLG / NFE=160 / FID=52.61
KAR2 / NFE=51 / FID=56
KAR2+DLG / NFE=51 / FID=36

Figure 14: Additional non-cherry-picked samples on FFHQ-1024 without (left col.) and with (right col.) DLG.

C. Pseudocodes

Let us denote a reverse-S/ODE integrator as $\pi(s_\theta, \boldsymbol{x}, \sigma, n)$. Given a point $\boldsymbol{x}$ at noise level σ, score function s_θ, and NFE budget n, π returns the result of integrating reverse-S/ODE from σ to $\sigma_{\min}$ starting from $\boldsymbol{x}$. That is, π returns a sample from $p_0(\boldsymbol{x})$.

Algorithm 1 MCMC Starting Point Generation

```
1:  Input: Integration NFE budget $n_1$, number of Langevin-Gibbs steps $n_2$, Langevin step size $\eta$
2:  Sample $\tilde{\boldsymbol{x}}_0 \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\max}^2 \boldsymbol{I})$
3:  $\tilde{\boldsymbol{x}}_0 \leftarrow \pi(s_\theta, \tilde{\boldsymbol{x}}_0, \sigma_{\max}, n_1)$
4:  $\tilde{\boldsymbol{x}}_0 \leftarrow \tilde{\boldsymbol{x}}_0 + 0.5 \cdot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
5:  $\sigma_0 \leftarrow 0.5$
6:  for $t = 1, 2, \dots n_2$ do
7:      $\tilde{\boldsymbol{x}}_0 \leftarrow \tilde{\boldsymbol{x}}_0 + 0.5 \cdot \eta \cdot s_\theta(\tilde{\boldsymbol{x}}_0, \sigma_0) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
8:      $\sigma_0 \sim q_\phi(\sigma \mid \boldsymbol{x}_0)$
9:  end for
10: Return $(\tilde{\boldsymbol{x}}_0, \sigma_0)$
```

Algorithm 2 Denoising Langevin Gibbs

```
1:  Input: $n_{den}$, $n_{skip}$, MCMC starting point $(\tilde{\boldsymbol{x}}_0, \sigma_0)$, Langevin step size $\eta$, number of samples needed $N$
2:  $(\tilde{\boldsymbol{x}}, \sigma) \leftarrow (\tilde{\boldsymbol{x}}_0, \sigma_0)$
3:  for $k = 1, 2, \dots N$ do
4:      Initialize minimum noise level tracking variables $(\hat{\boldsymbol{x}}, \hat{\sigma}) \leftarrow (\tilde{\boldsymbol{x}}, \sigma)$
5:      for $t = 1, 2, \dots, n_{skip}$ do
6:          $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + 0.5 \cdot \eta \cdot s_\theta(\tilde{\boldsymbol{x}}, \sigma) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
7:          $\sigma \sim q_\phi(\sigma \mid \tilde{\boldsymbol{x}})$
8:          if $\sigma < \hat{\sigma}$ then
9:              $(\hat{\boldsymbol{x}}, \hat{\sigma}) \leftarrow (\tilde{\boldsymbol{x}}, \sigma)$
10:         end if
11:     end for
12:     $\boldsymbol{x}_k \leftarrow \pi(s_\theta, \hat{\boldsymbol{x}}, \hat{\sigma}, n_{den})$
13: end for
14: Return $\{\boldsymbol{x}_k\}_{k=1}^N$
```

D. More Related Works

D.1. Comparison of DMCMC With the Predictor-Corrector Sampler

As both DMCMC and the predictor-corrector (PC) algorithm (Song et al., 2021b) rely on MCMC, one may ask whether DMCMC is a straightforward extension of PC. However, we assure the reader that DMCMC is not a trivial generalization of PC.

On a high level, the main message of our paper is that we can significantly improve diffusion sampling by choosing better initial points for the denoising process. This claim was verified through extensive experiments. On the other hand, the focus of the corrector step of PC is on improving the denoising process by refining the predictor / integrator steps. So, DMCMC and PC contribute to orthogonal aspects of diffusion sampling. Indeed, the improvements offered by MCMC corrector step are rather marginal (see Table 1 in (Song et al., 2021b)) compared to the improvements offered by DMCMC (see Figure 3 in our paper). On a low level, we distinguish DMCMC from PC on three levels: theoretical, implementation-wise, and empirical.

Theoretically, MCMC in DMCMC and MCMC in PC reduce truncation error (error arising from numerically integrating reverse-S/ODE) in entirely distinct ways. We observe that diffusion model sampling can be broken down into two parts: (part 1) generating an initialization point, and (part 2) integrating the reverse-S/ODE starting from the initialization point to generate clean data. MCMC in PC aims to improve part 2 whereas MCMC in DMCMC aims to improve part 1.

In PC, MCMC is used as a corrector. That is, MCMC corrects the distributions of intermediate points during integration of the reverse-S/ODE (part 2). That is why PC sampling proceeds by alternating between taking a predictor step and running Langevin dynamics at a fixed noise level.

In DMCMC, MCMC generates initialization points for reverse-S/ODE (part 1) that lie near the image manifold. This is possible because MCMC runs in the augmented space $\mathcal{X} \times \mathcal{S}$ by adaptively updating the noise level. This leads to reduced truncation error, or equivalently, acceleration, as it is easier for reverse-S/ODE integrators to generate clean data from points near the data manifold than from prior noise distribution samples (e.g., Gaussian noise). For a more detailed explanation of the acceleration mechanism of DMCMC, we refer the reader to Appendix E.1.

DMCMC does not resemble PC even from the perspective of implementation. MCMC in PC runs during reverse-S/ODE integration, and MCMC in DMCMC runs before reverse-S/ODE integration:

- PC : (prior noise) $\rightarrow$ (integ. step) $\rightarrow$ (MCMC) $\rightarrow$ (integ. step) $\rightarrow$ (MCMC) $\rightarrow \ldots \rightarrow$ (data)
- DMCMC : (MCMC to generate points near image manifold) $\rightarrow$ (integration) $\rightarrow$ (data)

Moreover, as already mentioned, MCMC in PC runs in $\mathcal{X}$ at a fixed σ whereas MCMC in DMCMC runs in $\mathcal{X} \times \mathcal{S}$ by adaptively updating σ.

Finally, DMCMC can also be used to accelerate PC algorithms as well, as shown in Figure 15. This means MCMC in DMCMC and PC play orthogonal roles in generating clean data. If DMCMC were a straightforward extension of PC, we would not see such acceleration.

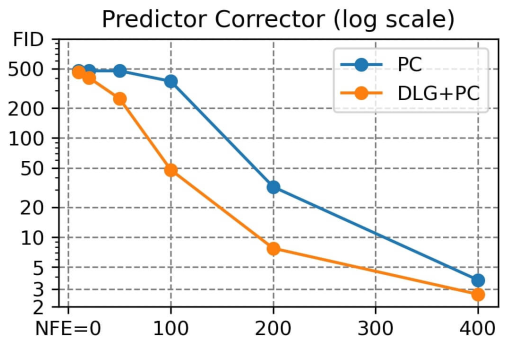


Figure 15: PC acceleration with DLG on CIFAR10.

D.2. On the Noise Predictor Network in DMCMC

Nichol & Dhariwal (2021) and Roman et al. (2021) have also used noise predictor networks to improve diffusion sampling. Although both works employed a neural net to predict the noise level / diffusion time for a given data input, we emphasize that the works do not overlap with the contributions of DMCMC. The main contribution of our paper is not the noise prediction network itself. The novelty of DMCMC arises from how we use the noise prediction network.

On a high level, the main message of our paper is that we can significantly improve diffusion sampling by choosing better initial points for denoising. A noise predictor network was used in the process of finding initial points. On the other hand, the focus of the mentioned works is on improving the denoising process with a noise predictor network. So, DMCMC and the mentioned works contribute to orthogonal aspects of diffusion sampling.

Let us elaborate. We first note that diffusion model sampling can be broken down into two parts: (part 1) generating an initialization point, and (part 2) integrating the reverse-S/ODE to generate data from the initialization point.

Nichol & Dhariwal (2021) and Roman et al. (2021) use the noise prediction network to accelerate integration of the reverse-S/ODE (improve part 2). This is achieved by learning the covariance of the reverse distribution (Nichol & Dhariwal, 2021) or by adjusting the noise schedule with a neural net (Roman et al., 2021).

In DMCMC, MCMC generates initialization points for reverse-S/ODE that lie near the image manifold (improve part 1). This is possible because MCMC runs in $\mathcal{X} \times \mathcal{S}$ by adaptively updating the noise level with a noise prediction network. This leads to acceleration, as it is easier for reverse-S/ODE integrators to generate clean data from points near the data manifold than from prior noise distribution samples (e.g., Gaussian noise). Hence, our paper proposes an acceleration approach entirely different from those of and Nichol & Dhariwal (2021) and Roman et al. (2021).

In fact, DMCMC does not resemble Nichol & Dhariwal (2021) and Roman et al. (2021) even from the perspective of implementation. Noise predictors in Nichol & Dhariwal (2021) and Roman et al. (2021) are used during reverse-S/ODE integration, and the noise predictor in DMCMC is used before reverse-S/ODE integration as a sub-step of MCMC.

E. More Discussions

E.1. More Explanation on How DMCMC Accelerates Sampling

The discussion at the end of Section 3.2 describes why DMCMC combined with a reverse-S/ODE integrator is faster than using a reverse-S/ODE integrator alone. Here, we provide a more detailed explanation of the acceleration phenomenon. To begin, we note that DMCMC consists of two steps that are executed sequentially: (a) MCMC on $\hat{p}(\boldsymbol{x}, \sigma)$, and (b) denoising MCMC samples by reverse-S/ODE. Since DMCMC without (a) is just standard diffusion, we see the acceleration behavior comes from using $\hat{p}(\boldsymbol{x}, \sigma)$ samples as initial points for reverse-S/ODE. Thus, we need to show how using $\hat{p}(\boldsymbol{x}, \sigma)$ samples as initial points for reverse-S/ODE accelerates image generation.

This proceeds in two steps. We first explain how MCMC produces samples $(\boldsymbol{x}_n, \sigma_n)$ from $\hat{p}(\boldsymbol{x}, \sigma)$ such that σ_n is significantly smaller than $\sigma_{\max}$. Then, we explain how integrating over $(\sigma_{\min}, \sigma_n)$ is faster than integrating over $(\sigma_{\min}, \sigma_{\max})$ as in standard diffusion.

To begin, we first explain how MCMC produces samples $(\boldsymbol{x}_n, \sigma_n)$ from $\hat{p}(\boldsymbol{x}, \sigma)$ such that σ_n is significantly smaller than $\sigma_{\max}$. We observe that $\int p_\sigma(\boldsymbol{x}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})d\tilde{\boldsymbol{x}}$ becomes flatter and wider as σ increases. This is because $\int p_\sigma(\boldsymbol{x}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})d\tilde{\boldsymbol{x}}$ means we are applying Gaussian smoothing to $p(\boldsymbol{x})$ with a Gaussian kernel of variance σ^2. For instance, if $p(\boldsymbol{x})$ is a normal distribution with variance γ^2, $\int p_\sigma(\boldsymbol{x}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})d\tilde{\boldsymbol{x}}$ is a normal distribution with variance $\sigma^2 + \gamma^2$. It follows that high density values of $\hat{p}(\boldsymbol{x}, \sigma)$ occur when $\boldsymbol{x}$ is near the data manifold and σ is small. Since MCMC traverses high-probability regions, we can expect $\{\boldsymbol{x}_n\}$ will be close to the data manifold and $\{\sigma_n\}$ will be smaller than $\sigma_{\max}$. Indeed, in Appendix F.1, we see that actual σ_n values are significantly smaller than $\sigma_{\max} = 50$ in CIFAR10.

Standard diffusion needs to integrate the reverse-S/ODE over the large interval $(\sigma_{\min}, \sigma_{\max})$ to produce clean images. On the other hand, in DMCMC, MCMC produces samples $(\boldsymbol{x}_n, \sigma_n)$ from $\hat{p}(\boldsymbol{x}, \sigma)$ such that σ_n is significantly smaller than $\sigma_{\max}$. So, in the DMCMC framework, to generate clean images, we only need to integrate the reverse-S/ODE over the small interval $(\sigma_{\min}, \sigma_n)$. This means the cost of integrating over $(\sigma_n, \sigma_{\max})$ vanishes for DMCMC, leading to accelerated image generation. (To be precise, the cost of integrating over $(\sigma_n, \sigma_{\max})$ is replaced by the cost of running MCMC to sample from $\hat{p}(\boldsymbol{x}, \sigma)$, but we observe in Section 5.1 that DLG mixes rapidly, so this cost is negligible.)

More rigorously, given the same computation budget and the same integration method, integrating over $(\sigma_{\min}, \sigma_n)$ has less truncation error than integrating over $(\sigma_{\min}, \sigma_{\max})$. Computation budget roughly corresponds to the number of discretization points of the integration interval we can use to approximate the reverse-S/ODE. Thus, a shorter interval of integration means we can use smaller step size h during integration, which implies smaller error. For instance, Euler's method has $O(h^2)$ local error. A more rigorous exposition is given in Chapter 7 of Stoer & Bulisch (2002).

This justifies how DMCMC can generate better samples than standard diffusion under a fixed computation budget. In other words, DMCMC can use less computation budget than standard diffusion to achieve similar sample quality as standard diffusion, i.e., DMCMC can accelerate sampling.

E.2. Trade-off Between the Convergence Speed of MCMC and Sharpness of $\hat{p}(\sigma)$

Let us assume that $-\log \hat{p}(\boldsymbol{x} \mid \sigma)$ is strongly convex and has L Lipschitz continuous gradients. We also assume $-\log \hat{p}(\sigma)$ is strongly convex and has M Lipschitz continuous gradients. We use such assumptions, because in the setting where either $-\log \hat{p}(\boldsymbol{x} \mid \sigma)$ or $-\log \hat{p}(\sigma)$ is nonconvex, it is difficult to say anything theoretically meaningful about convergence of MCMC on the joint $\hat{p}(\boldsymbol{x}, \sigma)$. We also assume the MCMC of choice is Langevin dynamics for ease of analysis.

The sharpness of the prior $\hat{p}(\sigma)$ can then be characterized by M. Intuitively, if $\hat{p}(\sigma)$ is more peaked around zero, $-\log \hat{p}(\sigma)$

will have a gradient which changes more rapidly, and so M will be larger. We then note that since $-\log \hat{p}(\boldsymbol{x}, \sigma) = -\log \hat{p}(\boldsymbol{x} \mid \sigma) - \log \hat{p}(\sigma)$, $-\log \hat{p}(\boldsymbol{x}, \sigma)$ is also strongly convex and $-\log \hat{p}(\boldsymbol{x}, \sigma)$ has $L + M$ Lipschitz continuous gradients.

We now resort to Theorem 3 in (Cheng & Bartlett, 2018), which shows that the convergence time of Langevin dynamics on $-\log \hat{p}(\boldsymbol{x}, \sigma)$ in terms of KL divergence grows in the order of $O((L+M)^2)$ (ignoring log terms). So, we indeed see there is a trade-off between the mixing time of MCMC and the choice of the prior. Using a prior that is sharper around zero will increase M and thus increase convergence time quadratically. We speculate that a similar analysis will hold for Langevin Gibbs as well, but a rigorous analysis of Langevin Gibbs is worthy of a paper of its own.

However, in practice, we do note that Langevin Gibbs with the prior used in our paper converges quite fast, as shown in Section 5.1. In the rightmost panel of Figure 2, we observe that the autocorrelation of image labels vanishes after only a few iterations. If the sampler mixed poorly in the $\boldsymbol{x}$ space, image labels would have high autocorrelation. For visualization of DLG chains, we refer the readers to Appendix B.1. So, we can say Langevin Gibbs reliably produces samples from $\hat{p}(\boldsymbol{x}, \sigma)$ even with a small number of steps.

F. More Experiments

F.1. Evolution of σ During Langevin Gibbs on $\mathcal{X} \times \mathcal{S}$

In Figure 16, we have visualized Langevin Gibbs trajectories of σ in the MoG setting with $1k$ modes at CIFAR10 images (this setting decouples sampler behavior from score function approximation error) and the score network setting. We indeed observe that σ moves up and down, allowing $\boldsymbol{x}$ to travel between disjoint modes of the distribution. Moreover, in Figure 2 of Section 5.1, $\boldsymbol{x}$ samples cover all $1k$ modes of the MoG. This experimentally proves $\boldsymbol{x}$ sequence of DLG is exploring the entire image distribution.

In the MoG setting, let us define δ_n as the distance of $\boldsymbol{x}_n$ to the closest mode in the MoG. In Figure 17, we see $\sigma_n \sim q_\phi(\sigma \mid \boldsymbol{x}_n)$ is almost identical to $\delta_n/\sqrt{d}$, where d is the dimension of $\boldsymbol{x}_n$. This is reasonable, as the Gaussian annulus theorem tells us samples from a high-dimensional Gaussian distribution of mean μ and variance σ^2 come from a shell of radius $\sigma\sqrt{d}$ centered at μ.

Specifically, due to the Gaussian annulus theorem, the samples of $\int p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}}$ are most likely to come from a shell of radius $\sigma\sqrt{d}$ centered around the image manifold (see Figure 2 (a) in Chung et al. (2022)). Then, given certain $\boldsymbol{x}$ which is distance δ from the image manifold, we can intuitively argue that optimal σ for $\boldsymbol{x}$, i.e., σ of highest likelihood under $p(\sigma \mid \boldsymbol{x})$, can be determined by equating $\delta = \sigma\sqrt{d}$ such that $\sigma = \delta/\sqrt{d}$. In the MoG case, this δ is approximated by the distance of $\boldsymbol{x}$ to the closest mode in the MoG. Since the σ values predicted by the noise classifier agree with approximated $\delta/\sqrt{d}$, we can see that the noise classifier is a good approximation of $p(\sigma \mid \boldsymbol{x})$.

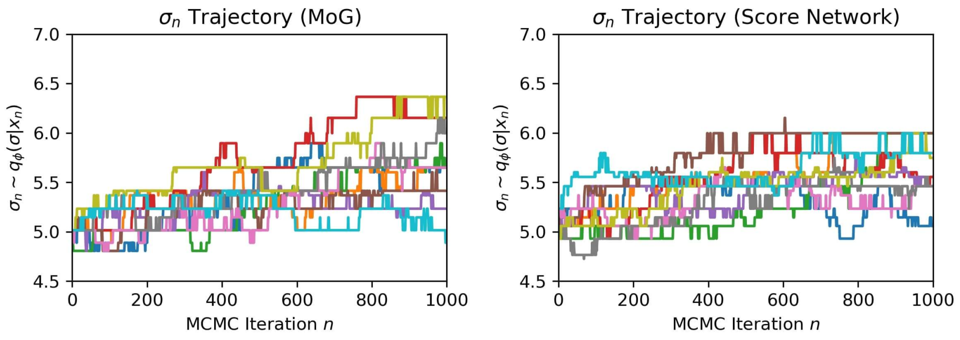


Figure 16: Visualization of σ_n trajectories in MoG and score network settings.

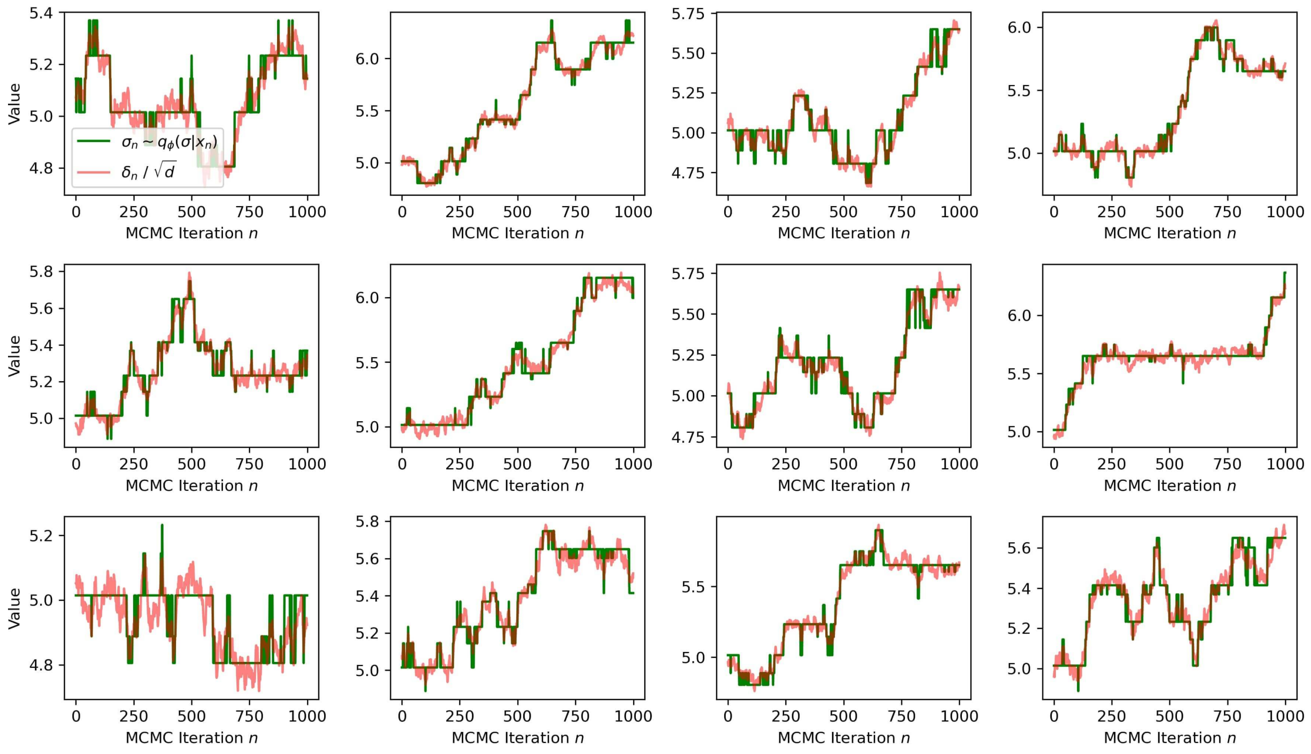


Figure 17: q_ϕ prediction and (approximate) distance to image manifold in the MoG setting.

F.2. Denoising Step Ablation

One may ask whether the denoising step in DMCMC is a necessary component to generate clean samples. To answer this question, we compare DLG with and without the denoising step. We use the reverse diffusion integrator as the reverse-SDE solver in the denoising step. Figure 18 shows DLG without the denoising step completely fails to generate valid clean image samples, regardless of how large n_{skip} we use.

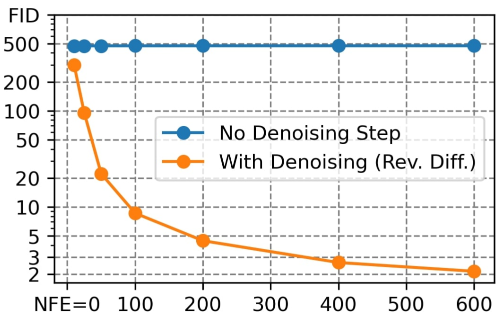


Figure 18: Ablation of the denoising step in DMCMC.