DENOISING MCMC FOR ACCELERATING DIFFUSION-BASED GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models are powerful generative models that simulate the reverse of diffusion processes using score functions to synthesize data from noise. 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 samples in the product space of data and variance (or diffusion time). Then, a reverse-S/ODE integrator is used to denoise the MCMC samples. Since MCMC traverses close to the data manifold, the computation cost of producing a clean sample for DMCMC is much less than that of producing a clean sample from noise. To verify the proposed concept, we show that Denoising Langevin Gibbs (DLG), an instance of DMCMC, successfully accelerates all six reverse-S/ODE integrators considered in this work on the tasks of CIFAR10 and CelebA-HQ-256 image generation. Notably, combined with integrators of Karras et al. (2022) and pre-trained score models of Song et al. (2021b), DLG achieves state-of-the-art results. In the limited number of score function evaluation (NFE) settings on CIFAR10, we have 3.86 FID with ≈ 10 NFE and 2.63 FID with ≈ 20 NFE. On CelebA-HQ-256, we have 6.99 FID with ≈ 160 NFE, which beats the current best record of Kim et al. (2022) among score-based models, 7.16 FID with 4000 NFE.

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 stepsizes to accelerate sampling. Furthermore, multitude of recent works on score-based generative modeling focus on improv-

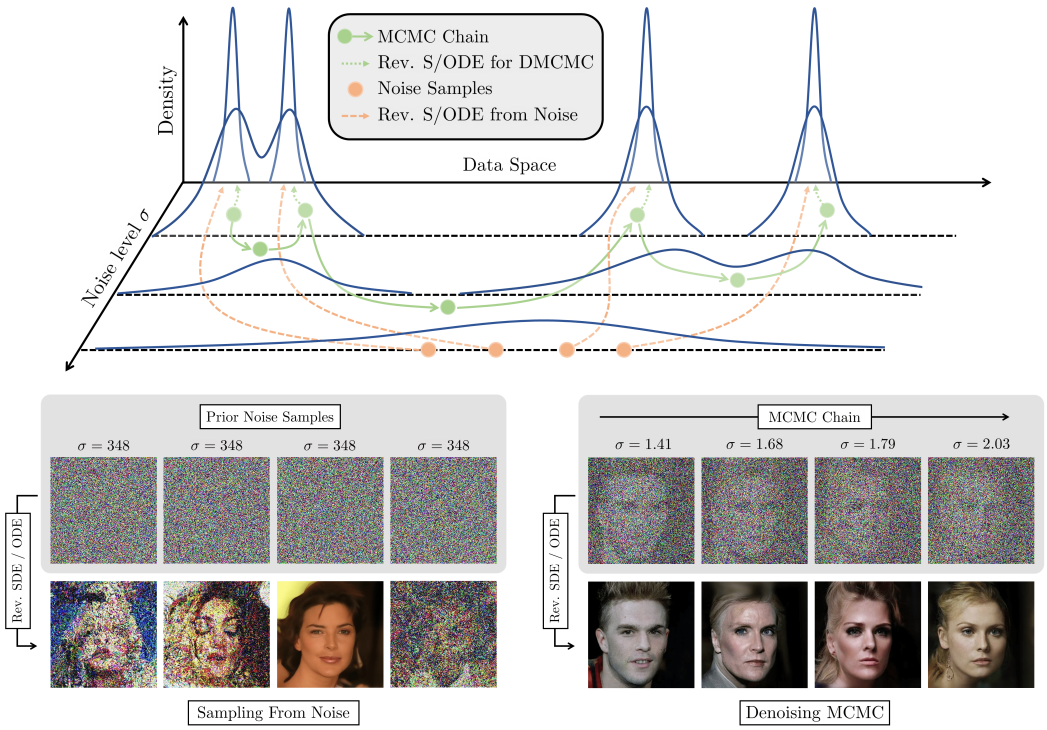

Figure 1: **Top:** a conceptual illustration of a VE diffusion model sampling process and DMCMC sampling process. VE diffusion models integrate the reverse-S/ODE starting from maximum diffusion time / maximum noise level. So, samples are often noisy with small computation budget due to large truncation 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.

ing 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 samples $\{(\boldsymbol{x}_n, t_n)\}$ in the product space of data $\boldsymbol{x}$ and variance exploding (VE) diffusion time t / noise level σ (see Fig. 1 top panel). 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-density 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 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 VE process noise-conditional score function, and any 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 noise level prediction step, so all that DLG requires is a pre-trained noise-conditional score network and a noise level 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. On CIFAR10 in the limited number of score function evaluation (NFE) setting, we obtain 3.86 FID with ≈ 10 NFE and 2.63 FID with ≈ 20 NFE. On CelebA-HQ-256, we have 6.99 FID with ≈ 160 NFE, which is currently the best result with score-based models. The computation cost of evaluating a noise level 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)

$$\min_\theta \mathbb{E}_{p(\tilde{\boldsymbol{x}})} \mathbb{E}_{p_\sigma(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \left[\| \boldsymbol{s}_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \|_2^2 \right] \tag{1}$$

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

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

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

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (\eta/2) \cdot \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_n) + \sqrt{\eta} \cdot \boldsymbol{\epsilon} \tag{3}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. $\{\boldsymbol{x}_n\}_{n=1}^\infty$ 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. (3), 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 find its way back.

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. (3) for $i = N, N-1, \ldots, 1$. 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

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\, dt + g(t)\, d\boldsymbol{w} \tag{4}$$

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. (4) from time $t = 0$ to $t = T$ produces a sample from the prior distribution.

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

$$d\boldsymbol{x} = [\boldsymbol{f}(\boldsymbol{x}, t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})]\, dt + g(t)\, d\bar{\boldsymbol{w}} \tag{5}$$

where $p_t(\boldsymbol{x})$ is the density of $\boldsymbol{x}(t)$ evolving according to Eq. (4) 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. (5) 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

$$d\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - (1/2) \cdot g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right]\, dt \tag{6}$$

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 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

$$d\boldsymbol{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}}\, d\boldsymbol{w}. \tag{7}$$

where $\sigma(t)$ increases monotonically with t from $\sigma_{\min}$ to $\sigma_{\max}$. The data distribution evolves as

$$p_t(\boldsymbol{x}) = \int p_{\sigma(t)}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}} \tag{8}$$

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, $p_T(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \sigma_{\max}^2 \boldsymbol{I})$. If we have a score model $\boldsymbol{s}_\theta(\boldsymbol{x}, \sigma)$ trained with Eq. (2), $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \approx \boldsymbol{s}_\theta(\boldsymbol{x}, \sigma(t))$. It follows that with $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \sigma_{\max}^2 \boldsymbol{I})$, we may integrate the reverse-S/ODE corresponding to Eq. (7) with $\boldsymbol{x}(T) = \boldsymbol{x}_T$ from $t = T$ to $t = 0$ using a score model to generate data. In the next section, we bridge MCMC and reverse-S/ODE integrators with VE diffusion to form a novel sampling framework that improves both MCMC and diffusion models.

3 DENOISING MARKOV CHAIN MONTE CARLO (DMCMC)

From here on, we denote the data distribution as $p(\boldsymbol{x})$ and its domain as $\mathcal{X} \subseteq \mathbb{R}^d$. Diffusion runs from time $t = 0$ to $t = T$, and noise scale $\sigma(t)$ increases monotonically from $\sigma_{\min}$ to $\sigma_{\max}$, such that $\sigma(0) = \sigma_{\min}$ and $\sigma(T) = \sigma_{\max}$. We denote the range of $\sigma(t)$ as $\mathcal{S} = [\sigma_{\min}, \sigma_{\max}]$. The Gaussian perturbation kernel is denoted as $p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \mathcal{N}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}, \sigma^2 \boldsymbol{I})$. The distribution of $\boldsymbol{x}(t)$ following the VE diffusion Eq. (7) is denoted as $p_t(\boldsymbol{x})$, and recall that $p_t(\boldsymbol{x}) = \int p_{\sigma(t)}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}}$.

We now develop a general framework called Denoising MCMC (DMCMC) which combines MCMC with reverse-S/ODE integrators. The construction of DMCMC is comprised of two steps. In the first step, we build MCMC on 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 MCMC samples via reverse-S/ODE integrators.

3.1 Construction Step 1: MCMC on the Product Space $\mathcal{X} \times \mathcal{S}$

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. Indeed, convergence time for such distributions can grow exponential in dimension d (Roberts & Rosenthal, 2001). Intuitively, for MCMC to move between disjoint modes, the Markov Chain would have to step off the data manifold. However, 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

$$\hat{p}(\boldsymbol{x} \mid \sigma) \coloneqq \int p_\sigma(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) p(\tilde{\boldsymbol{x}})\, d\tilde{\boldsymbol{x}}. \tag{9}$$

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

$$\hat{p}(\boldsymbol{x}, \sigma) = \hat{p}(\boldsymbol{x} \mid \sigma) \cdot \hat{p}(\sigma). \tag{10}$$

Here, $\hat{p}(\sigma)$ reflects our preference for how much time we want the MCMC chain to stay at a particular level of σ. MCMC with $\hat{p}(\boldsymbol{x}, \sigma)$ will produce samples $\{(\boldsymbol{x}_n, \sigma_n)\}$ in $\mathcal{X} \times \mathcal{S}$ such that

$$\sigma_n \sim \hat{p}(\sigma), \qquad \boldsymbol{x}_n \sim \hat{p}(\boldsymbol{x} \mid \sigma_n). \tag{11}$$

Hence, 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 can 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 next section, we incorporate reverse-S/ODE integrators to avert this problem.

3.2 Construction Step 2: Incorporating Denoising Steps

Let us recall that integrating the reverse-S/ODE for the VE diffusion SDE Eq. (7) 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 $t_n \coloneqq \sigma^{-1}(\sigma_n)$, Eq. (11) tells us

$$\boldsymbol{x}_n \sim p_{t_n}(\boldsymbol{x}) \tag{12}$$

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.

Given an MCMC chain $\{(\boldsymbol{x}_n, \sigma_n)\}$ in $\mathcal{X} \times \mathcal{S}$, MCMC is biased towards high-density regions of $\mathcal{X}$, so the sequence $\{\boldsymbol{x}_n\}$ will generally stay close to the data manifold, except when traversing between disjoint modes. This means $\sigma_n \ll \sigma_{\max}$ for most n, or in other words, $t_n \ll T$ for most n. So, the average length of integration intervals will tend to be much shorter than T. Thus, a numerical integrator can integrate the reverse-S/ODE with less truncation error given the same computation budget (Stoer & Bulisch, 2002). Equivalently, less computation budget is required to reach the same truncation error level. This idea is illustrated in Figure 1.

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 Eq. (1) 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 $\nu > 0$)

$$\min_\theta \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_{d+1}, \nu^2 \boldsymbol{I}_{d+1})} \mathbb{E}_{\hat{p}(\boldsymbol{x}, \sigma)} [\| \hat{s}_\theta(\boldsymbol{x} - \epsilon_{1:d}, \sigma - \epsilon_{d+1}) - \boldsymbol{\epsilon}/\nu^2 \|_2^2] \tag{13}$$

which differs from Eq. (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).

Updating $\boldsymbol{x}$. Suppose we are given an MCMC iterate $(\boldsymbol{x}_n, \sigma_n)$ and a score model $s_\theta(\boldsymbol{x}, \sigma)$ from Eq. (2). We generate $\boldsymbol{x}_{n+1}$ by a Langevin dynamics step on $\hat{p}(\boldsymbol{x} \mid \sigma_n)$. Specifically, by Eq. (9),

$$\nabla_{\boldsymbol{x}} \log \hat{p}(\boldsymbol{x} \mid \sigma_n) \approx s_\theta(\boldsymbol{x}, \sigma_n) \tag{14}$$

and so an Langevin dynamics update on $\boldsymbol{x}$, according to Eq. (3) is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + (\eta/2) \cdot s_\theta(\boldsymbol{x}_n, \sigma_n) + \sqrt{\eta} \cdot \boldsymbol{\epsilon} \tag{15}$$

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

Updating σ. We now 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 solving

$$\max_\phi \mathbb{E}_{\hat{p}(\boldsymbol{x}, \sigma)} [\log q_\phi(\sigma \mid \boldsymbol{x})]. \tag{16}$$

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 $\sigma_{n+1} \sim q_\phi(\sigma \mid \boldsymbol{x}_{n+1})$.

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, 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.

Initialization points for DLG. Theoretically, MCMC chain $\{(\boldsymbol{x}_n, \sigma_n)\}_{n=1}^\infty$ will converge to $\hat{p}(\boldsymbol{x}, \sigma)$ regardless of the initialization point $(\boldsymbol{x}_0, \sigma_0)$. However, we found it was beneficial to set $\boldsymbol{x}_0$ close to the image manifold and set $\sigma_0 \sim q_\phi(\sigma \mid \boldsymbol{x}_0)$. Indeed, theory also shows that setting initialization points close to the stationary distribution could significantly accelerate convergence of the Markov chain (Dalalyan, 2017; Dwivedi et al., 2019). In practice, we set $\boldsymbol{x}_0$ by generating a sample starting from a noise distribution, adding Gaussian noise of variance 0.25, and running Gibbs sampling for a few iterations. 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 which consist 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 noise scale 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.

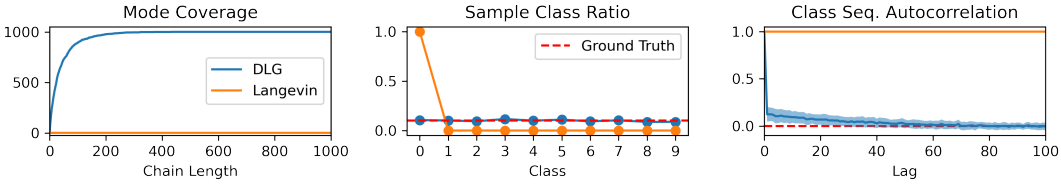

Figure 2: Mixing analysis of DLG with a mixture of Gaussians.

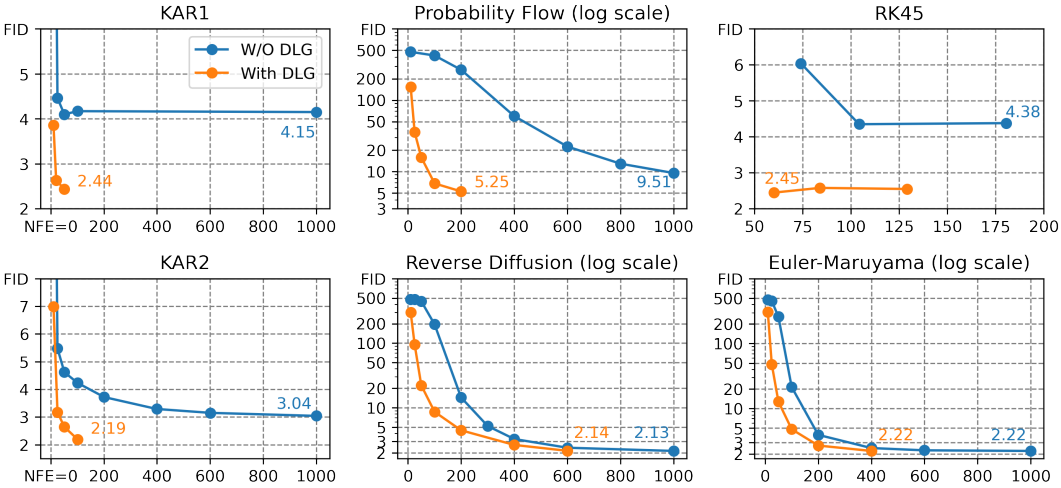

Figure 3: Sampling acceleration of DLG on CIFAR10. FID of notable points are written in the corresponding color. **Top row**: deterministic integrators. **Bottom row**: stochastic integrators.

5 Experiments

5.1 Mixing of DMCMC Chains

For DMCMC to successfully generate diverse high-quality data, DMCMC must mix, i.e., traverse between disjoint modes of $p(\boldsymbol{x})$. We provide experimental evidence that DMCMC indeed mixes as a consequence of running MCMC in the product space $\mathcal{X} \times \mathcal{S}$. Specifically, 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 observe 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 of the distribution. 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.

5.2 Accelerating Image Generation with Score Networks

We compare six integrators with and without DLG on CIFAR10 and CelebA-HQ-256 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, and for CelebA-HQ-256, we generate $10k$ samples. We use pre-trained score models of Song et al. (2021b).

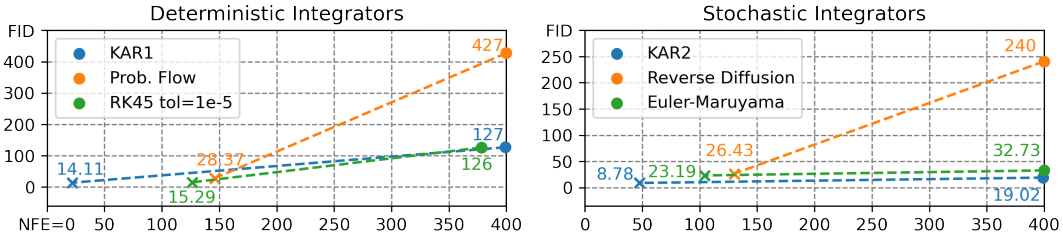

Figure 4: Sampling acceleration of DLG on CelebA-HQ-256. A dot indicates an integrator without DLG, and a cross of the same color indicates corresponding integrator combined with DLG. Dotted lines indicate performance improvement due to DLG.

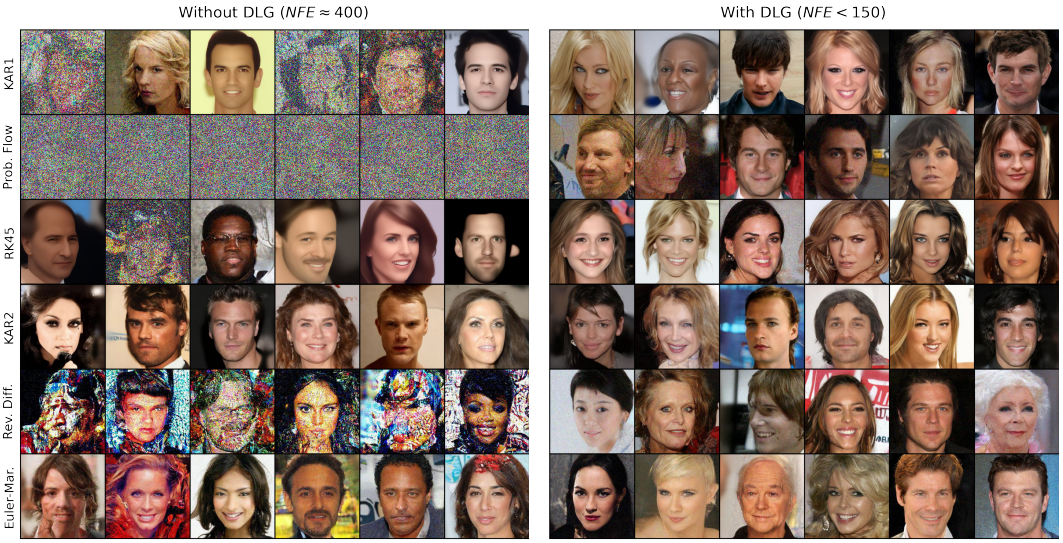

Figure 5: Non-cherry-picked samples on CelebA-HQ-256 using the settings for Fig. 4. Each row shows samples for an integrator without (left col.) and with (right col.) DLG.

CIFAR10. Figure 3 shows the results on CIFAR10. 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. Specifically, while the performance of KAR1 and RK45 saturates at around 4 FID, KAR1 and RK45 combined with DLG achieve better results around 2.4 FID.

CelebA-HQ-256. Figure 4 shows the results on CelebA-HQ-256. We observe that DLG improves computational efficiency and sample quality simultaneously. Indeed, in Figure 5, 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, so fine-tuning could yield better results.

Achieving SOTA. DLG combined with KAR1 sets a new SOTA record for CIFAR10 in the limited number of NFE setting: 3.86 FID with 10.11 NFE and 2.63 FID with 20.11 NFE which beats the results of Zhang & Chen (2022), 4.17 FID with 10 NFE and 2.86 FID with 20 NFE. 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. (2022), 7.16 FID with 4000 NFE.

5.3 Ablation Study

In this section, we perform an ablation study of the components of DLG. Based on the observations, we construct a generic strategy for choosing the hyper-parameters. Given a reverse-S/ODE integra-

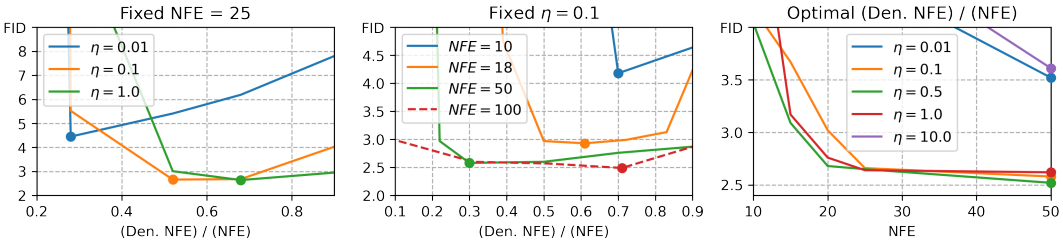

Figure 6: Ablation study of DLG with the deterministic integrator of Karras et al. (2022). Dots indicate the points of lowest FID.

tor, DLG is determined by three hyper-parameters: total NFE per sample n, NFE spent on denoising samples n_{den}, and Langevin dynamics step size η. We fix the integrator to be the deterministic sampler of Karras et al. (2022) and observe the effect of each hyper-parameter on CIFAR10 image generation. We observed the same general trend with other samplers.

η vs. n_{den}/n. In the left panel of Figure 6, 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 6, we fix η and vary NFE and n_{den}/n. We observe two 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.

Also, as we increase NFE, the set of n_{den}/n which provides near-optimal performance becomes larger. Moreover, we see that most of the time, optimal FID is achieved when $n_{den}/n > 0.5$, i.e., when $n_{den} > n_{skip}$. 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 then allocate $n_{den} = n - n_{skip}$.

η vs. NFE. In the right panel of Figure 6, 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, we obtain similarly good performance. In the case of CIFAR10, we found it reasonable to set $\eta \in [0.05, 1.0]$.

To choose η for data of general dimension d, we define a value $\kappa := \eta/\sqrt{d}$ called displacement per dimension. If we see Eq. (14), Gaussian noise of zero mean and variance η is added to the sample at each update step. Gaussian annulus theorem tells us that a high-dimensional Gaussian noise with zero mean and variance η has Euclidean norm approximately $\eta\sqrt{d}$. So, the average displacement of the sample per dimension by the random noise is around $\eta/\sqrt{d}$. Since κ is a dimension-independent value, given κ and d, we can set $\eta = \sqrt{d}\kappa$. On CIFAR10, we have $\eta \in [0.05, 1.0]$, which translates to $\kappa \in [0.0009, 0.018]$. This means, on CelebA-HQ-256, we can choose $\eta \in [0.4, 8.0]$. If the sampler was inefficient, we chose a smaller κ to trade-off diversity for sample quality.

6 Conclusion

In this work, we proposed DMCMC which combines MCMC with reverse-S/ODE integrators. This has led to significant improvements for both MCMC and diffusion models. For MCMC, DMCMC allows Markov chains to traverse between 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, which uses Langevin dynamics along with Gibbs sampling as the choice of MCMC. We 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. Overall, our work opens up an orthogonal approach to accelerating score-based sampling. We leave exploration of other kinds of MCMC or diffusion process in DMCMC as future work.

REFERENCES

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

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

Raaz Dwivedi, Yuansi Chen, Martin J.Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast. *Journal of Machine Learning Research*, 20:1–42, 2019.

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

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

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

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NIPS*, 2017.

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

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arxiv preprint arXiv:2105.14080*, 2021.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arxiv preprint arXiv:2206.00364*, 2022.

Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal truncation technique of score-based diffusion for high precision score estimation. In *ICML*, 2022.

Diedrik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

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

Paul Langevin. On the theory of Brownian motion. 1908.

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

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ODE solver for diffusion probabilistic sampling in around 10 steps. *arxiv preprint arXiv:2206.00927*, 2022.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6): 1087–1092, 1953.

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

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

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learing using nonequilibrium thermodynamics. In *ICML*, 2015.

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

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

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 *ICLR*, 2021b.

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

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

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. 2011.

Qinsheng Zhang and Yongxin Chen. 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) and the CelebA-HQ-256 dataset (Karras et al., 2018).

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 1 lists the hyper-parameters for DLG and the corresponding FID used to produce 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 1: 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 2 lists the hyper-parameters for DLG and the corresponding FID used to produce Figure 4.

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

Achieving SOTA. On CIFAR10, we use KAR1 settings of Table 1. 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

Figures 7 and 8 each show a DLG chain on CIFAR10 and CelebA-HQ-256, respectively. 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.

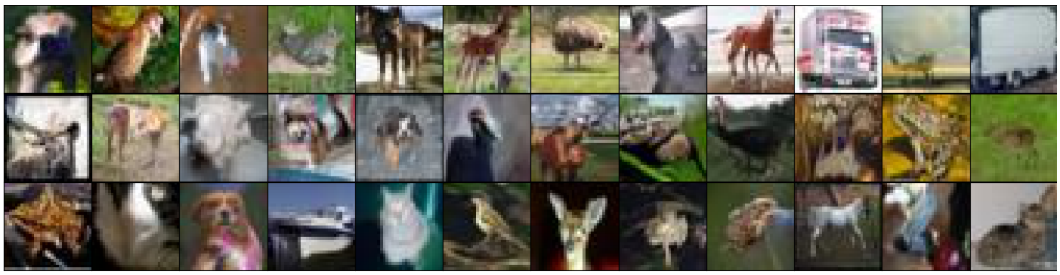

Figure 7: Visualization of a DLG chain on CIFAR10.

Figure 8: Visualization of a DLG chain on CelebA-HQ-256.

B.2 Additional Unconditional Samples

In Figures 9 and 10, we show additional samples for CIFAR10 without and with DLG. In Figures 11 and 12, we show additional samples for CelebA-HQ-256 without and with DLG.

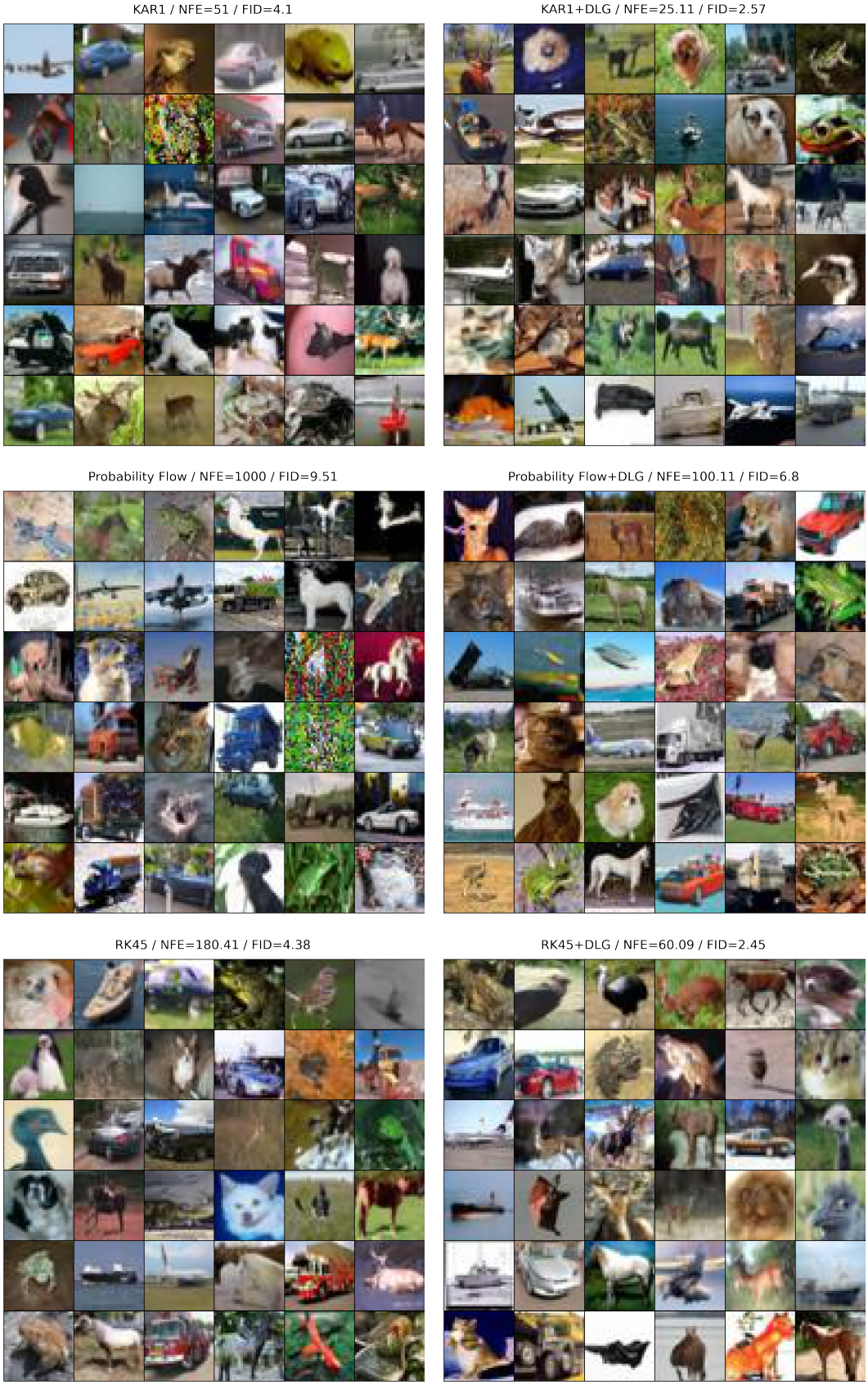

Figure 9: Additional non-cherry-picked samples for deterministic integrators on CIFAR10 without (left col.) and with (right col.) DLG.

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

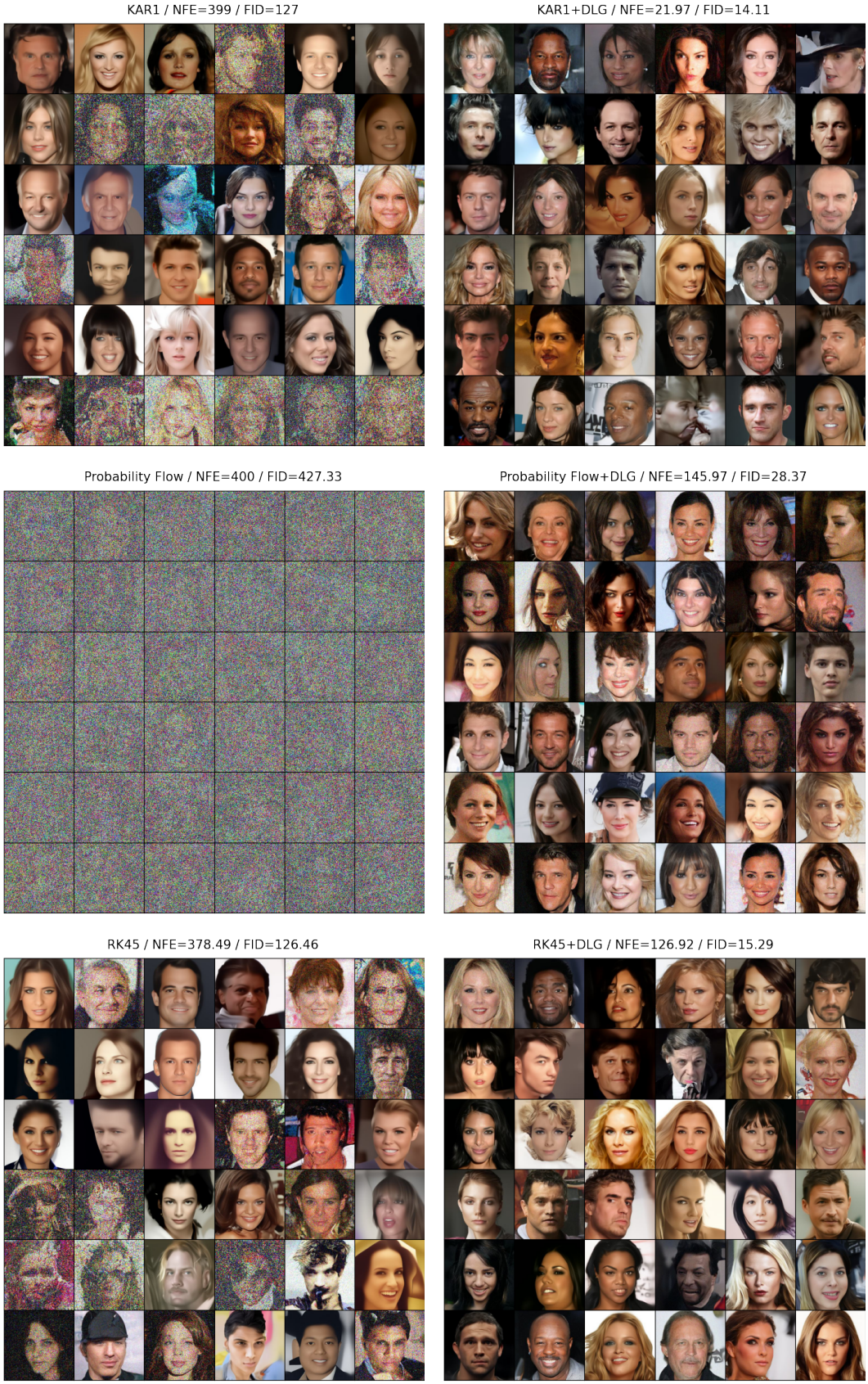

Figure 11: Additional non-cherry-picked samples for deterministic integrators on CelebA-HQ-256 without (left col.) and with (right col.) DLG.

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