DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps

Anonymous Author(s)
Affiliation
Address
email

Abstract

Diffusion probabilistic models (DPMs) are emerging powerful generative models. Despite their high-quality generation performance, DPMs still suffer from their slow sampling as they generally need hundreds or thousands of sequential function evaluations (steps) of large neural networks to draw a sample. Sampling from DPMs can be viewed alternatively as solving the corresponding diffusion ordinary differential equations (ODEs). In this work, we propose an exact formulation of the solution of diffusion ODEs. The formulation analytically computes the linear part of the solution, rather than leaving all terms to black-box ODE solvers as adopted in previous works. By applying change-of-variable, the solution can be equivalently simplified to an exponentially weighted integral of the neural network. Based on our formulation, we propose *DPM-Solver*, a fast dedicated high-order solver for diffusion ODEs with the convergence order guarantee. DPM-Solver is suitable for both discrete-time and continuous-time DPMs without any further training. Experimental results show that DPM-Solver can generate high-quality samples in only 10 to 20 function evaluations on various datasets. We achieve 4.70 FID in 10 function evaluations and 2.87 FID in 20 function evaluations on the CIFAR10 dataset, and a $4 \sim 16\times$ speedup compared with previous state-of-the-art training-free samplers on various datasets.

1 Introduction

Diffusion probabilistic models (DPMs) [1–3] are emerging powerful generative models with promising performance on many tasks, such as image generation [4, 5], video generation [6], text-to-image generation [7], speech synthesis [8, 9] and lossless compression [10]. DPMs are defined by discrete-time random processes [1, 2] or continuous-time stochastic differential equations (SDEs) [3], which learn to gradually remove the noise added to the data points. Compared with the widely-used generative adversarial networks (GANs) [11] and variational auto-encoders (VAEs) [12], DPMs can not only compute exact likelihood [3], but also achieve even better sample quality for image generation [4]. However, to obtain high-quality samples, DPMs usually need hundreds or thousands of sequential steps of large neural network evaluations, thereby resulting in a much slower sampling speed than the single-step GANs or VAEs. Such inefficiency is becoming a critical bottleneck for the adoption of DPMs in downstream tasks, leading to an urgent request to design fast samplers for DPMs.

Existing fast samplers for DPMs can be divided into two categories. The first category includes knowledge distillation [13, 14] and noise level or sample trajectory learning [15–18]. Such methods require a possibly expensive training stage before they can be used for efficient sampling. Furthermore, their applicability and flexibility might be limited. It might require nontrivial effort to adapt the method to different models, datasets, and number of sampling steps. The second category consists of training-free [19–21] samplers, which are suitable for all pre-trained DPMs in a simple plug-and-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

NFE = 10 NFE = 15 NFE = 20 NFE = 100 NFE = 10

(a) DDIM [19] (b) DPM-Solver (ours)

Figure 1: Samples by DDIM [19] with 10, 15, 20, 100 number of function evaluations (NFE), and DPM-Solver (ours) with only 10 NFE, using the pre-trained DPMs on ImageNet 256×256 with classifier guidance [4].

play manner. Training-free samplers include adopting implicit [19] or analytical [21] generation process, advanced differential equation (DE) solvers [3, 20, 22–24] and dynamic programming [18]. However, these methods still require ~ 50 function evaluations [21] to generate high-quality samples (comparable to those generated by plain samplers in about 1000 function evaluations), thereby are still time-consuming.

In this work, we bring the efficiency of training-free samplers to a new level to produce high-quality samples in the "*few-step sampling*" regime, where the sampling can be done within around 10 steps of sequential function evaluations. We tackle the alternative problem of sampling from DPMs as solving the corresponding diffusion ordinary differential equations (ODEs) of DPMs, and carefully examine the structure of diffusion ODEs. Diffusion ODEs have a semi-linear structure — they consist of a linear function of the data variable and a nonlinear function parameterized by neural networks. Such structure is omitted in previous training-free samplers [3, 20], which directly use black-box DE solvers. To utilize the semi-linear structure, we derive an exact formulation of the solutions of diffusion ODEs by analytically computing the linear part of the solutions, avoiding the corresponding discretization error. Furthermore, by applying change-of-variable, the solutions can be equivalently simplified to an exponentially weighted integral of the neural network. Such integral is very special and can be efficiently approximated by the numerical methods for exponential integrators [25].

Based on our formulation of solutions, we propose *DPM-Solver*, a fast dedicated solver for diffusion ODEs by approximating the above integral. Specifically, we propose first-order, second-order and third-order versions of DPM-Solver with convergence order guarantees. We further propose an adaptive step size schedule for DPM-Solver. In general, DPM-Solver is applicable to both continuous-time and discrete-time DPMs, and also conditional sampling with classifier guidance [4]. Fig. 1 demonstrates the speedup performance of a Denoising Diffusion Implicit Models (DDIM) [19] baseline and DPM-Solver, which shows that DPM-Solver can generate high-quality samples with as few as 10 function evaluations and is much faster than DDIM on the ImageNet 256x256 dataset [26]. Our additional experimental results show that DPM-Solver can greatly improve the sampling speed of both discrete-time and continuous-time DPMs, and it can achieve excellent sample quality in around 10 function evaluations, which is much faster than all previous training-free samplers of DPMs.

2 Diffusion Probabilistic Models

We review diffusion probabilistic models and their associated differential equations in this section.

2.1 Forward Process and Diffusion SDEs

Assume that we have a D-dimensional random variable $\boldsymbol{x}_0 \in \mathbb{R}^D$ with an unknown distribution $q_0(\boldsymbol{x}_0)$. Diffusion Probabilistic Models (DPMs) [1–3, 10] define a *forward process* $\{\boldsymbol{x}_t\}_{t\in[0,T]}$ with

$T > 0$ starting with $\boldsymbol{x}_0$, such that for any $t \in [0, T]$, the distribution of $\boldsymbol{x}_t$ conditioned on $\boldsymbol{x}_0$ satisfies

$$q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha(t)\boldsymbol{x}_0, \sigma^2(t)\boldsymbol{I}), \tag{2.1}$$

where $\alpha(t), \sigma(t) \in \mathbb{R}^+$ are differentiable functions of t with bounded derivatives, and we denote them as α_t, σ_t for simplicity. The choice for α_t and σ_t is referred to as the *noise schedule* of a DPM. Let $q_t(\boldsymbol{x}_t)$ denote the marginal distribution of $\boldsymbol{x}_t$, DPMs choose noise schedules to ensure that $q_T(\boldsymbol{x}_T) \approx \mathcal{N}(\boldsymbol{x}_T|\boldsymbol{0}, \tilde{\sigma}^2\boldsymbol{I})$ for some $\tilde{\sigma} > 0$, and the *signal-to-noise-ratio* (SNR) α_t^2/σ_t^2 is strictly decreasing w.r.t. t [10]. Moreover, Kingma et al. [10] prove that the following stochastic differential equation (SDE) has the same transition distribution $q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)$ as in Eq. (2.1) for any $t \in [0, T]$:

$$\mathrm{d}\boldsymbol{x}_t = f(t)\boldsymbol{x}_t\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t, \quad \boldsymbol{x}_0 \sim q_0(\boldsymbol{x}_0), \tag{2.2}$$

where $\boldsymbol{w}_t \in \mathbb{R}^D$ is the standard Wiener process, and

$$f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \quad g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2. \tag{2.3}$$

Under some regularity conditions, Song et al. [3] show that the forward process in Eq. (2.2) has an equivalent *reverse process* from time T to 0, starting with the marginal distribution $q_T(\boldsymbol{x}_T)$:

$$\mathrm{d}\boldsymbol{x}_t = [f(t)\boldsymbol{x}_t - g^2(t)\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t, \quad \boldsymbol{x}_T \sim q_T(\boldsymbol{x}_T), \tag{2.4}$$

where $\bar{\boldsymbol{w}}_t$ is a standard Wiener process in the reverse time. The only unknown term in Eq. (2.4) is the *score function* $\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)$ at each time t. In practice, DPMs use a neural network $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ parameterized by θ to estimate the scaled score function: $-\sigma_t\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)$. The parameter θ is optimized by minimizing the following objective [2, 3]:

$$\begin{aligned}\mathcal{L}(\theta;\omega(t)) &\coloneqq \frac{1}{2}\int_0^T \omega(t)\mathbb{E}_{q_t(\boldsymbol{x}_t)}\Big[\|\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) + \sigma_t\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)\|_2^2\Big]\mathrm{d}t \\ &= \frac{1}{2}\int_0^T \omega(t)\mathbb{E}_{q_0(\boldsymbol{x}_0)}\mathbb{E}_{q(\boldsymbol{\epsilon})}\Big[\|\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}\|_2^2\Big]\mathrm{d}t + C,\end{aligned}$$

where $\omega(t)$ is a weighting function, $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{x}_t = \alpha_t\boldsymbol{x}_0 + \sigma_t\boldsymbol{\epsilon}$, and C is a constant independent of θ. As $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ can also be regarded as predicting the Gaussian noise added to $\boldsymbol{x}_t$, it is usually called the *noise prediction model*. Since the ground truth of $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is $-\sigma_t\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)$, DPMs replace the score function in Eq. (2.4) by $-\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)/\sigma_t$ and define a parameterized reverse process (*diffusion SDE*) from time T to 0, starting with $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2\boldsymbol{I})$:

$$\mathrm{d}\boldsymbol{x}_t = \left[f(t)\boldsymbol{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t, \quad \boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2\boldsymbol{I}). \tag{2.5}$$

Samples can be generated from DPMs by solving the diffusion SDE in Eq. (2.5) with numerical solvers, which discretize the SDE from T to 0. Song et al. [3] proved that the traditional ancestral sampling method for DPMs [2] can be viewed as a first-order SDE solver for Eq. (2.5). However, these first-order methods usually need hundreds of or thousands of function evaluations to converge [3], leading to extremely slow sampling speed.

2.2 Diffusion (Probability Flow) ODEs

When discretizing SDEs, the step size is limited by the randomness of the Wiener process [27, Chap. 11]. A large step size (small number of steps) often causes non-convergence, especially in high dimensional spaces. For faster sampling, one can consider the associated *probability flow ODE* [3], which has the same marginal distribution at each time t as that of the SDE. Specifically, for DPMs, Song et al. [3] proved that the probability flow ODE of Eq. (2.4) is

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = f(t)\boldsymbol{x}_t - \frac{1}{2}g^2(t)\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t), \quad \boldsymbol{x}_T \sim q_T(\boldsymbol{x}_T), \tag{2.6}$$

where the marginal distribution of $\boldsymbol{x}_t$ is also $q_t(\boldsymbol{x}_t)$. By replacing the score function with the noise prediction model, Song et al. [3] defined the following parameterized ODE (*diffusion ODE*):

$$\frac{\mathrm{d}\boldsymbol{x}_t}{\mathrm{d}t} = \boldsymbol{h}_\theta(\boldsymbol{x}_t, t) \coloneqq f(t)\boldsymbol{x}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t), \quad \boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \tilde{\sigma}^2\boldsymbol{I}). \tag{2.7}$$

Samples can be drawn by solving the ODE from T to 0. Comparing with SDEs, ODEs can be solved with larger step sizes as they have no randomness. Furthermore, we can take advantage of efficient numerical ODE solvers to accelerate the sampling. Song et al. [3] used the RK45 ODE solver [28] for the diffusion ODEs, which generates samples in ~ 60 function evaluations to reach comparable quality with a 1000-step SDE solver for Eq. (2.5) on the CIFAR-10 dataset [29]. However, existing general-purpose ODE solvers still cannot generate satisfactory samples in the few-step (~ 10 steps) sampling regime. To the best of our knowledge, there is still a lack of training-free samplers for DPMs in the few-step sampling regime, and the sampling speed of DPMs is still a critical issue.

3 Customized Fast Solvers for Diffusion ODEs

As highlighted in Sec. 2.2, discretizing SDEs is generally difficult in high dimensions [27, Chap. 11] and it is hard to converge within few steps. In contrast, ODEs are easier to solve, yielding a potential for fast samplers. However, as mentioned in Sec. 2.2, the general black-box ODE solver used in previous work [3] empirically fails to converge in few steps. This motivates us to design a dedicated solver for diffusion ODEs to enable fast and high-quality few-step sampling. We start with a detailed investigation of the specific structure of diffusion ODEs.

3.1 Simplified Formulation of Exact Solutions of Diffusion ODEs

The key insight of this work is that given an initial value $\boldsymbol{x}_s$ at time $s > 0$, the solution $\boldsymbol{x}_t$ at each time $t < s$ of diffusion ODEs in Eq. (2.7) can be simplified into a very special exact formulation which can be efficiently approximated.

Our first key observation is that a part of the solution $\boldsymbol{x}_t$ can be exactly computed by considering the particular structure of diffusion ODEs. The r.h.s. of diffusion ODEs in Eq. (2.7) consists of two parts: the part $f(t)\boldsymbol{x}_t$ is a linear function of $\boldsymbol{x}_t$, and the other part $\frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is generally a nonlinear function of $\boldsymbol{x}_t$ because of the neural network $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$. This type of ODE is referred to as *semi-linear* ODE. The black-box ODE solvers adopted by previous works [3, 22] are ignorant of this semi-linear structure as they take the whole $\boldsymbol{h}_\theta(\boldsymbol{x}_t, t)$ in Eq. (2.7) as the input, which causes discretization errors of both the linear and nonlinear term. We note that for semi-linear ODEs, the solution at time t can be exactly formulated by the *"variation of constants"* formula [30]:

$$\boldsymbol{x}_t = e^{\int_s^t f(\tau)\mathrm{d}\tau}\boldsymbol{x}_s + \int_s^t \left(e^{\int_\tau^t f(r)\mathrm{d}r} \frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau) \right) \mathrm{d}\tau. \tag{3.1}$$

This formulation decouples the linear part and the nonlinear part. In contrast to black-box ODE solvers, the linear part is now exactly computed, which eliminates the approximation error of the linear term. However, the integral of the nonlinear part is still complicated because it couples the coefficients about the noise schedule (i.e., $f(\tau), g(\tau), \sigma_\tau$) and the complex neural network $\boldsymbol{\epsilon}_\theta$, which is still hard to approximate.

Our second key observation is that the integral of the nonlinear part can be greatly simplified by introducing a special variable. Let $\lambda_t \coloneqq \log(\alpha_t/\sigma_t)$ (one half of the log-SNR), then λ_t is a strictly decreasing function of t (due to the definition of DPMs as discussed in Sec. 2.1). We can rewrite $g(t)$ in Eq. (2.3) as

$$g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2 = 2\sigma_t^2\left(\frac{\mathrm{d}\log\sigma_t}{\mathrm{d}t} - \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\right) = -2\sigma_t^2\frac{\mathrm{d}\lambda_t}{\mathrm{d}t}. \tag{3.2}$$

Combining with $f(t) = \mathrm{d}\log\alpha_t/\mathrm{d}t$ in Eq. (2.3), we can rewrite Eq. (3.1) as

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s - \alpha_t\int_s^t \left(\frac{\mathrm{d}\lambda_\tau}{\mathrm{d}\tau}\right)\frac{\sigma_\tau}{\alpha_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\mathrm{d}\tau. \tag{3.3}$$

As $\lambda(t) = \lambda_t$ is a strictly decreasing function of t, it has an inverse function $t_\lambda(\cdot)$ satisfying $t = t_\lambda(\lambda(t))$. We further change the subscripts of $\boldsymbol{x}$ and $\boldsymbol{\epsilon}_\theta$ from t to λ and denote $\hat{\boldsymbol{x}}_\lambda \coloneqq \boldsymbol{x}_{t_\lambda(\lambda)}$, $\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda) \coloneqq \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{t_\lambda(\lambda)}, t_\lambda(\lambda))$. Rewrite Eq. (3.3) by *"change-of-variable"* for λ, then we have:

Proposition 3.1 (Exact solution of diffusion ODEs)**.** *Given an initial value $\boldsymbol{x}_s$ at time $s > 0$, the solution $\boldsymbol{x}_t$ at time $t \in [0, s]$ of diffusion ODEs in Eq.* (2.7) *is:*

$$\boldsymbol{x}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{x}_s - \alpha_t\int_{\lambda_s}^{\lambda_t} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda, \lambda)\mathrm{d}\lambda. \tag{3.4}$$

We call the integral $\int e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)\mathrm{d}\lambda$ the *exponentially weighted integral* of $\hat{\boldsymbol{\epsilon}}_\theta$, which is very special and highly related to the *exponential integrators* in the literature of ODE solvers [25]. To the best of our knowledge, such formulation has not been revealed in prior work of diffusion models.

Eq. (3.4) provides a new perspective for approximating the solutions of diffusion ODEs. Specifically, given $\boldsymbol{x}_s$ at time s, According to Eq. (3.4), approximating the solution at time t is equivalent to directly approximating the exponentially weighted integral of $\hat{\boldsymbol{\epsilon}}_\theta$ from λ_s to λ_t, which avoids the error of the linear terms and is well-studied in the literature of exponential integrators [25, 31]. Based on this insight, we propose fast solvers for diffusion ODEs, as detailed in the following sections.

3.2 High-Order Solvers for Diffusion ODEs

In this section, we propose high-order solvers for diffusion ODEs with convergence order guarantee by leveraging our proposed solution formulation Eq. (3.4). The proposed solvers and analysis are highly motivated by the methods of exponential integrators [25, 31] in the ODE literature.

Specifically, given an initial value $\boldsymbol{x}_T$ at time T and $M+1$ time steps $\{t_i\}_{i=0}^M$ decreasing from $t_0 = T$ to $t_M = 0$. Let $\tilde{\boldsymbol{x}}_{t_0} = \boldsymbol{x}_T$ be the initial value. The proposed solvers use M steps to iteratively compute a sequence $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=0}^M$ to approximate the true solutions at time steps $\{t_i\}_{i=0}^M$. In particular, the last iterate $\tilde{\boldsymbol{x}}_{t_M}$ approximates the true solution at time 0.

In order to reduce the approximation error between $\tilde{\boldsymbol{x}}_{t_M}$ and the true solution at time 0, we need to reduce the approximation error for each $\tilde{\boldsymbol{x}}_{t_i}$ at every step [30]. Starting with the previous value $\tilde{\boldsymbol{x}}_{t_{i-1}}$ at time t_{i-1}, according to Eq. (3.4), the exact solution $\boldsymbol{x}_{t_{i-1}\to t_i}$ at time t_i is given by

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)\mathrm{d}\lambda. \tag{3.5}$$

Therefore, to compute the value $\tilde{\boldsymbol{x}}_{t_i}$ for approximating $\boldsymbol{x}_{t_{i-1}\to t_i}$, we need to approximate the exponentially weighted integral of $\hat{\boldsymbol{\epsilon}}_\theta$ from $\lambda_{t_{i-1}}$ to λ_{t_i}. Denote $h_i := \lambda_{t_i} - \lambda_{t_{i-1}}$, and $\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_\lambda,\lambda) := \frac{\mathrm{d}^n\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)}{\mathrm{d}\lambda^n}$ as the n-th order total derivative of $\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)$ w.r.t. λ. For $k \geq 1$, the $(k-1)$-th order Taylor expansion of $\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda)$ w.r.t. λ at $\lambda_{t_{i-1}}$ is

$$\hat{\boldsymbol{\epsilon}}_\theta(\hat{\boldsymbol{x}}_\lambda,\lambda) = \sum_{n=0}^{k-1}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}},\lambda_{t_{i-1}}) + \mathcal{O}((\lambda-\lambda_{t_{i-1}})^k),$$

Substituting the above Taylor expansion into Eq. (3.5) yields

$$\boldsymbol{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\sum_{n=0}^{k-1}\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_{\lambda_{t_{i-1}}},\lambda_{t_{i-1}})\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda + \mathcal{O}(h_i^{k+1}), \tag{3.6}$$

where the integral $\int e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\mathrm{d}\lambda$ can be **analytically** computed by repeatedly applying n times of integration-by-parts (see Appendix A.2). Therefore, to approximate $\boldsymbol{x}_{t_{i-1}\to t_i}$, we only need to approximate the n-th order total derivatives $\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\boldsymbol{x}}_\lambda,\lambda)$ for $n \leq k-1$, which is a well-studied problem in the ODE literature [31, 32]. By dropping the $\mathcal{O}(h_i^{k+1})$ error term and approximating the first $(k-1)$-th total derivatives with the "stiff order conditions" [31, 32], we can derive k-th-order ODE solvers for diffusion ODEs. We name such solvers as *DPM-Solver* overall, and *DPM-Solver-k* for a specific order k. Here we take $k = 1$ for demonstration. In this case, Eq. (3.6) becomes

$$\begin{aligned}\boldsymbol{x}_{t_{i-1}\to t_i} &= \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}},t_{i-1})\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\mathrm{d}\lambda + \mathcal{O}(h_i^2)\\ &= \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}(e^{h_i}-1)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}},t_{i-1}) + \mathcal{O}(h_i^2).\end{aligned}$$

By dropping the high-order error term $\mathcal{O}(h_i^2)$, we can obtain an approximation for $\boldsymbol{x}_{t_{i-1}\to t_i}$. As $k = 1$ here, we call this solver *DPM-Solver-1*, and the detailed algorithm is as following.

DPM-Solver-1. Given an initial value $\boldsymbol{x}_T$ and $M+1$ time steps $\{t_i\}_{i=0}^M$ decreasing from $t_0 = T$ to $t_M = 0$. Starting with $\tilde{\boldsymbol{x}}_{t_0} = \boldsymbol{x}_T$, the sequence $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=1}^M$ is computed iteratively as follows:

$$\tilde{\boldsymbol{x}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}(e^{h_i}-1)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}},t_{i-1}), \quad \text{where } h_i = \lambda_{t_i} - \lambda_{t_{i-1}}. \tag{3.7}$$

For $k \geq 2$, approximating the first k terms of the Taylor expansion needs additional intermediate points between t and s [31]. The derivation is more technical so we defer it to Appendix A. Below we propose algorithms for $k = 2, 3$ and name them as *DPM-Solver-2* and *DPM-Solver-3*, respectively.

Algorithm 1 DPM-Solver-2.

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\boldsymbol{\epsilon}_\theta$

$\tilde{\boldsymbol{x}}_{t_0} = \boldsymbol{x}_T$
for $i \leftarrow 1$ to M **do**
  $s_i \leftarrow t_\lambda\left(\frac{\lambda_{t_{i-1}}+\lambda_{t_i}}{2}\right)$
  $\boldsymbol{u}_i \leftarrow \frac{\alpha_{s_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_i}\left(e^{\frac{h_i}{2}}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i}-1\right)\boldsymbol{\epsilon}_\theta(\boldsymbol{u}_i, s_i)$
end for
return $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=1}^M$

Algorithm 2 DPM-Solver-3.

Require: initial value $\boldsymbol{x}_T$, time steps $\{t_i\}_{i=0}^M$, model $\boldsymbol{\epsilon}_\theta$

$\tilde{\boldsymbol{x}}_{t_0} = \boldsymbol{x}_T$, $r_1 = \frac{1}{3}$, $r_2 = \frac{2}{3}$
for $i \leftarrow 1$ to M **do**
  $s_{2i-1} \leftarrow t_\lambda\left(\lambda_{t_i} + r_1 h_i\right), \quad s_{2i} \leftarrow t_\lambda\left(\lambda_{t_i} + r_2 h_i\right)$
  $\boldsymbol{u}_{2i-1} \leftarrow \frac{\alpha_{s_{2i-1}}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i-1}}\left(e^{r_1 h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\boldsymbol{D}_{2i-1} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i-1}, s_{2i-1}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\boldsymbol{u}_{2i} \leftarrow \frac{\alpha_{s_{2i}}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{s_{2i}}\left(e^{r_2 h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{s_{2i}} r_2}{r_1}\left(\frac{e^{r_2 h_i}-1}{r_2 h_i}-1\right)\boldsymbol{D}_{2i-1}$
  $\boldsymbol{D}_{2i} \leftarrow \boldsymbol{\epsilon}_\theta(\boldsymbol{u}_{2i}, s_{2i}) - \boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1})$
  $\tilde{\boldsymbol{x}}_{t_i} \leftarrow \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \sigma_{t_i}\left(e^{h_i}-1\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}) - \frac{\sigma_{t_i}}{r_2}\left(\frac{e^{h_i}-1}{h}-1\right)\boldsymbol{D}_{2i}$
end for
return $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=1}^M$

Here, $t_\lambda(\cdot)$ is the inverse function of $\lambda(t)$, which has an analytical formulation for the practical noise schedule used in [2, 16], as shown in Appendix C. The chosen intermediate points are $(s_i, \boldsymbol{u}_i)$ for DPM-Solver-2 and $(s_{2i-1}, \boldsymbol{u}_{2i-1})$ and $(s_{2i}, \boldsymbol{u}_{2i})$ for DPM-Solver-3. As shown in the algorithm, DPM-Solver-k requires k function evaluations per step for $k = 1, 2, 3$. Despite the more expensive steps, higher-order solvers ($k = 2, 3$) are usually more efficient since they require much fewer steps to converge, due to their higher convergence order. We show that DPM-Solver-k is k-th-order solver, as stated in the following theorem. The proof is in Appendix A.

Theorem 3.2 (DPM-Solver-k as a k-th-order solver)**.** *Assume $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ follows the regularity conditions detailed in Appendix A.1, then for $k = 1, 2, 3$, DPM-Solver-k is a k-th order solver for diffusion ODEs, i.e., for the sequence $\{\tilde{\boldsymbol{x}}_{t_i}\}_{i=1}^M$ computed by DPM-Solver-k, the approximation error at time 0 satisfies $\tilde{\boldsymbol{x}}_{t_M} - \boldsymbol{x}_0 = \mathcal{O}(h_{\max}^k)$, where $h_{max} = \max_{1\leq i\leq M}(\lambda_{t_i} - \lambda_{t_{i-1}})$.*

Finally, solvers with $k \geq 4$ need much more intermediate points as shown by previous work [31, 32] for exponential integrators. Therefore, we only consider k from 1 to 3 in this work, while leaving the solvers with higher k for future study.

3.3 Step Size Schedule

The proposed solvers in Sec. 3.2 need to specify the time steps $\{t_i\}_{i=0}^M$ in advance. We propose two choices of the time step schedule. One choice is handcrafted, which is to uniformly split the interval $[\lambda_T, \lambda_0]$, i.e. $\lambda_{t_i} = \lambda_T + \frac{i}{M}(\lambda_0 - \lambda_T)$, $i = 0, \ldots, M$. Note that this is different from previous work [2, 3] which chooses uniform steps for t_i. Empirically, DPM-Solver with uniform time steps λ_{t_i} can already generate quite good samples in few steps, where results are listed in Appendix D. As the other choice, we propose an adaptive step size algorithm, which dynamically adjusts the step size by combining different orders of DPM-Solver. The adaptive algorithm is inspired by [20] and we defer its implementation details to Appendix B.

For few-step sampling, we need to use up all the number of function evaluations (NFE). When the NFE is not divisible by 3, we firstly apply DPM-Solver-3 as much as possible, and then add a single step of DPM-Solver-1 or DPM-Solver-2 (dependent on the reminder of K divided by 3), as detailed in Appendix C. In the subsequent experiments, we use such combination of solvers with the uniform step size schedule for NFE ≤ 20, and otherwise the adaptive step size schedule.

3.4 Sampling from Discrete-Time DPMs

Discrete-time DPMs [2] train the noise prediction model at N fixed time steps $\{t_n\}_{n=1}^N$, and the noise prediction model is parameterized by $\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_n, n)$ for $n = 0, \ldots, N-1$, where each $\boldsymbol{x}_n$ is corresponding to the value at time t_{n+1}. We can transform the discrete-time noise prediction model to the continuous version by letting $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}, t) := \tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}, \frac{(N-1)t}{T})$, for all $\boldsymbol{x} \in \mathbb{R}^d, t \in [0, T]$. Note that the input time of $\tilde{\boldsymbol{\epsilon}}_\theta$ may not be integers, but we find that the noise prediction model can still work well, and we hypothesize that it is because of the smooth time embeddings (e.g., position embeddings [2]). By such reparameterization, the noise prediction model can adopt the continuous-time steps as input, and thus we can also use DPM-Solver for fast sampling.

4 Comparison with Existing Fast Sampling Methods

Here, we discuss the relationship and highlight the difference between DPM-Solver and existing ODE-based fast sampling methods for DPMs. We further briefly discuss the advantage of training-free samplers over those training-based ones.

4.1 DDIM as DPM-Solver-1

Denoising Diffusion Implicit Models (DDIM) [19] design a deterministic method for fast sampling from DPMs. For two adjacent time steps t_{i-1} and t_i, assume that we have a solution $\tilde{\boldsymbol{x}}_{t_{i-1}}$ at time t_{i-1}, then a single step of DDIM from time t_{i-1} to time t_i is

$$\tilde{\boldsymbol{x}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{x}}_{t_{i-1}} - \alpha_{t_i}\left(\frac{\sigma_{t_{i-1}}}{\alpha_{t_{i-1}}} - \frac{\sigma_{t_i}}{\alpha_{t_i}}\right)\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_{t_{i-1}}, t_{i-1}). \tag{4.1}$$

Although motivated by entirely different perspectives, we show that the updates of DPM-Solver-1 and Denoising Diffusion Implicit Models (DDIM) [19] are identical. By the definition of λ, we have $\frac{\sigma_{t_{i-1}}}{\alpha_{t_{i-1}}} = e^{-\lambda_{t_{i-1}}}$ and $\frac{\sigma_{t_i}}{\alpha_{t_i}} = e^{-\lambda_{t_i}}$. Plugging these and $h_i = \lambda_{t_i} - \lambda_{t_{i-1}}$ to Eq. (4.1) results in exactly a step of DPM-Solver-1 in Eq. (3.7). However, the semi-linear ODE formulation of DPM-Solver allows for principled generalization to higher-order solvers and convergence order analysis.

Recent work [13] also show that DDIM is a first-order discretization of diffusion ODEs by differentiating both sides of Eq. (4.1). However, they cannot explain the difference between DDIM and the first-order Euler discretization of diffusion ODEs. In contrast, by showing that DDIM is a special case of DPM-Solver, we reveal that DDIM makes full use of the semi-linearity of diffusion ODEs, which explains its superiority over traditional Euler methods.

4.2 Comparison with Traditional Runge-Kutta Methods

One can obtain a high-order solver by directly applying traditional explicit Runge-Kutta (RK) methods to the diffusion ODE in Eq. (2.7). Specifically, RK methods write the solution of Eq. (2.7) in the following integral form:

$$\boldsymbol{x}_t = \boldsymbol{x}_s + \int_s^t \boldsymbol{h}_\theta(\boldsymbol{x}_\tau, \tau)\mathrm{d}\tau = \boldsymbol{x}_s + \int_s^t \left(f(\tau)\boldsymbol{x}_\tau + \frac{g^2(\tau)}{2\sigma_\tau}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_\tau, \tau)\right)\mathrm{d}\tau, \tag{4.2}$$

and use some intermediate time steps between $[t, s]$ and combine the evaluations of $\boldsymbol{h}_\theta$ at these time steps to approximate the whole integral. The approximation error of explicit RK methods depends on $\boldsymbol{h}_\theta$, which consists of the error corresponding to both the linear term $f(\tau)\boldsymbol{x}_\tau$ and the nonlinear noise prediction model $\boldsymbol{\epsilon}_\theta$. However, the error of the linear term may increase exponentially because the exact solution of the linear term has an exponential coefficient (as shown in Eq. (3.1)). There are many empirical evidence [25, 31] showing that directly using explicit RK methods for semi-linear ODEs may suffer from unstable numerical issues for large step size. We also demonstrate the empirical difference of the proposed DPM-Solver and the traditional explicit RK methods in Sec. 5.1, which shows that DPM-Solver have smaller discretization errors than the RK methods with the same order.

4.3 Training-based Fast Samplers for DPMs

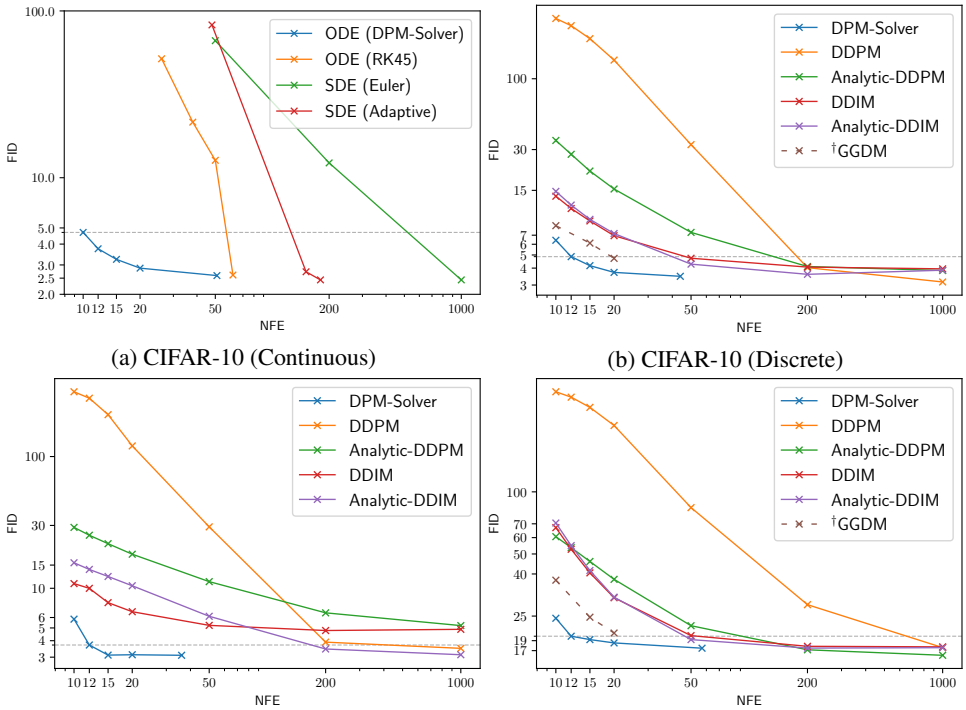
Samplers that need extra training or optimization include knowledge distillation [13, 14], learning the noise level or variance [15, 16], and learning the noise schedule or sample trajectory [17, 18]. Although the progressive distillation method [13] can obtain a fast sampler within 4 steps, it needs further training costs and loses part of the information in the original DPM (e.g., after distillation, the noise prediction model cannot predict the noise (score function) at every time step between $[0, T]$). In contrast, training-free samplers can keep all the information of the original model, and thereby can be directly extended to the conditional sampling by combining the original model and an external classifier [4] (e.g. see Appendix C for the conditional sampling with classifier guidance).

5 Experiments

In this section, we show that as a training-free sampler, DPM-Solver can greatly speedup the sampling of existing pre-trained DPMs, including both continuous-time and discrete-time ones, with both

Table 1: FID ↓ on CIFAR-10 for different orders of Runge-Kutta (RK) methods and DPM-Solvers, varying the number of function evaluations (NFE). For RK methods, we evaluate diffusion ODEs w.r.t. both t (Eq. (2.7)) and λ (Eq. (D.1)). We use uniform step size in t for RK (t), and uniform step size in λ for RK (λ) and DPM-Solvers.

Sampling method \ NFE	12	18	24	30	36	42	48
RK2 (t)	16.40	7.25	3.90	3.63	3.58	3.59	3.54
RK2 (λ)	107.81	42.04	17.71	7.65	4.62	3.58	3.17
DPM-Solver-2	**5.28**	**3.43**	**3.02**	**2.85**	**2.78**	**2.72**	**2.69**
RK3 (t)	48.75	21.86	10.90	6.96	5.22	4.56	4.12
RK3 (λ)	34.29	4.90	3.50	3.03	2.85	2.74	2.69
DPM-Solver-3	**6.03**	**2.90**	**2.75**	**2.70**	**2.67**	**2.65**	**2.65**



(a) CIFAR-10 (Continuous)

(b) CIFAR-10 (Discrete)

(c) CelebA 64x64 (Discrete)

(d) ImageNet 64x64 (Discrete)

Figure 2: Sample quality measured by FID ↓ of different sampling methods for DPMs on CIFAR-10 with both continuous-time and discrete-time models, CelebA 64x64 and ImageNet 64x64 with discrete-time models, varying the number of function evaluations (NFE). The method †GGDM [18] needs extra training to optimize the sample trajectory, while other methods are training-free. To get the strongest baseline, we use the quadratic step size for DDIM on CelebA, which has a better FID than that of the uniform step size in the original paper [19].

linear noise schedule [2, 19] and cosine noise schedule [16]. We vary different number of function evaluations (NFE) which is the number of calls to the noise prediction model $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$, and compare the sample quality between DPM-Solver and other methods. For each experiment, We draw 50K samples and use the widely adopted FID score [33] to evaluate the sample quality, where lower FID usually implies better sample quality.

Unless explicitly mentioned, we always use the solver combination with the uniform step size schedule in Sec. 3.3 if the NFE budget is less than 20, and otherwise the DPM-Solver-3 with the adaptive step size schedule in Sec. 3.3. We refer to Appendix C for other implementation details of DPM-Solver and Appendix D for detailed settings.

5.1 Comparison with Continuous-Time Sampling Methods

We firstly compare DPM-Solver with other continuous-time sampling methods for DPMs. The compared methods include the Euler-Maruyama discretization for diffusion SDEs [3], the adaptive step size solver for diffusion SDEs [20] and the RK methods for diffusion ODEs [3, 28] in Eq. (2.7).

We compare these methods for sampling from a pre-trained continuous-time "VP deep" model [3] on the CIFAR-10 dataset [29] with the linear noise schedule.

Fig. 2a shows the efficiency of compared solvers. We use uniform time steps with 50, 200, 1000 NFE for the diffusion SDE with Euler discretization, and vary the tolerance hyperparameter [3, 20] for the adaptive step size SDE solver [20] and RK45 ODE solver [28] to control the NFE. DPM-Solver can generate good sample quality within around 10 NFE, while other solvers have large discretization error even in 50 NFE, which shows that DPM-Solver can achieve $\sim$5 speedup of the previous best solver. In particular, we achieve 4.70 FID with 10 NFE, 3.75 FID with 12 NFE, 3.24 FID with 15 NFE, and 2.87 FID with 20 NFE, which is the fastest sampler on CIFAR-10.

As an ablation study, we also compare the second-order and third-order DPM-Solver and RK methods, as shown in Table 1. We compare RK methods for diffusion ODEs w.r.t. both time t in Eq. (2.7) and half-log-SNR λ by applying change-of-variable (see detailed formulations in Appendix D.1). The results show that given the same NFE, the sample quality of DPM-Solver is consistently better than RK methods with the same order. The superior efficiency of DPM-Solver is particularly evident in the few-step regime under 15 NFE, where RK methods have rather large discretization errors. This is mainly because DPM-Solver analytically computes the linear term, avoiding the corresponding discretization error. Besides, the higher order DPM-Solver-3 converges faster than DPM-Solver-2, which matches the order analysis in Theorem 3.2.

5.2 Comparison with Discrete-Time Sampling Methods

We use the reparameterization method in Sec. 3.4 for using DPM-Solver in discrete-time DPMs, and then compare DPM-Solver with other discrete-time training-free samplers, including DDPM [2], DDIM [19], Analytic-DDPM and Analytic-DDIM [21]. We also compare with GGDM [18], which uses the same pre-trained model but it needs further training to optimize the sampling trajectory. We compare the sample quality of the training-free samplers with 10, 12, 15, 20, 50, 200, 1000 NFE, and GGDM with 10, 15, 20 NFE, following their original paper [18].

Specifically, we use the discrete-time model trained by L_{simple} in [2] on the CIFAR-10 dataset with linear noise schedule, the discrete-time model in [19] on CelebA 64x64 [34] with linear noise schedule; and the discrete-time model trained by L_{hybrid} in [16] on ImageNet 64x64 [26] with cosine noise schedule (we only use their "mean" model and omit the "variance" model). As shown in Fig. 2, on all datasets, DPM-Solver can obtain reasonable samples within 12 steps (FID 4.86 on CIFAR-10, FID 3.40 on CelebA 64x64 and FID 19.97 on ImageNet 64x64), which is $4 \sim 16\times$ faster than the previous fastest training-free sampler. DPM-Solver even outperforms GGDM, which requires additional training.

6 Conclusion

We tackle the problem of fast and training-free sampling from DPMs. We propose DPM-Solver, a fast dedicated training-free solver of diffusion ODEs for fast sampling of DPMs in around 10 steps of function evaluations. DPM-Solver leverages the semi-linearity of diffusion ODEs and it directly approximates a simplified formulation of exact solutions of diffusion ODEs, which consists of an exponentially weighted integral of the noise prediction model. Inspired by numerical methods for exponential integrators, we propose first-order, second-order and third-order DPM-Solver to approximate the exponentially weighted integral of noise prediction models with theoretical convergence guarantee. We propose both handcrafted and adaptive step size schedule, and apply DPM-Solver for both continuous-time and discrete-time DPMs. Our experimental results show that DPM-Solver can generate high-quality samples in around 10 function evaluations on various datasets, and it can achieve $4 \sim 16\times$ speedup compared with previous state-of-the-art training-free samplers.

Limitations and broader impact Despite the promising speedup performance, DPM-Solver is designed for fast sampling, which may be not suitable for accelerating the likelihood evaluations of DPMs. In addition, like other deep generative models, DPMs may be used to generate adverse fake contents, and the proposed solver may further amplify the potential undesirable influence of deep generative models for malicious applications.

References

[1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840–6851.

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

[4] P. Dhariwal and A. Q. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 8780–8794.

[5] C. Meng, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "SDEdit: Image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*, 2022.

[6] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *arXiv preprint arXiv:2204.03458*, 2022.

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

[8] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*, 2021.

[9] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, N. Dehak, and W. Chan, "Wavegrad 2: Iterative refinement for text-to-speech synthesis," in *International Speech Communication Association*, 2021, pp. 3765–3769.

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

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 2672–2680.

[12] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[13] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *International Conference on Learning Representations*, 2022.

[14] E. Luhman and T. Luhman, "Knowledge distillation in iterative generative models for improved sampling speed," *arXiv preprint arXiv:2101.02388*, 2021.

[15] R. San-Roman, E. Nachmani, and L. Wolf, "Noise estimation for generative diffusion models," *arXiv preprint arXiv:2104.02600*, 2021.

[16] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8162–8171.

[17] M. W. Lam, J. Wang, R. Huang, D. Su, and D. Yu, "Bilateral denoising diffusion models," *arXiv preprint arXiv:2108.11514*, 2021.

[18] D. Watson, W. Chan, J. Ho, and M. Norouzi, "Learning fast samplers for diffusion models by differentiating through sample quality," in *International Conference on Learning Representations*, 2022.

[19] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

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

[21] F. Bao, C. Li, J. Zhu, and B. Zhang, "Analytic-DPM: An analytic estimate of the optimal reverse variance in diffusion probabilistic models," in *International Conference on Learning Representations*, 2022.

[22] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, "Pseudo numerical methods for diffusion models on manifolds," *arXiv preprint arXiv:2202.09778*, 2022.

[23] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-based voice conversion with fast maximum likelihood sampling scheme," in *International Conference on Learning Representations*, 2022.

[24] H. Tachibana, M. Go, M. Inahara, Y. Katayama, and Y. Watanabe, "Itô-Taylor sampling scheme for denoising diffusion probabilistic models using ideal derivatives," *arXiv preprint arXiv:2112.13339*, 2021.

[25] M. Hochbruck and A. Ostermann, "Exponential integrators," *Acta Numerica*, vol. 19, pp. 209–286, 2010.

[26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[27] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer, 1992.

[28] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[30] K. Atkinson, W. Han, and D. E. Stewart, *Numerical solution of ordinary differential equations*. John Wiley & Sons, 2011, vol. 108.

[31] M. Hochbruck and A. Ostermann, "Explicit exponential Runge-Kutta methods for semilinear parabolic problems," *SIAM Journal on Numerical Analysis*, vol. 43, no. 3, pp. 1069–1090, 2005.

[32] V. T. Luan, "Efficient exponential Runge-Kutta methods of high order: Construction and implementation," *BIT Numerical Mathematics*, vol. 61, no. 2, pp. 535–560, 2021.

[33] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017, pp. 6626–6637.

[34] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[35] K. Yang, J. Yau, L. Fei-Fei, J. Deng, and O. Russakovsky, "A study of face obfuscation in ImageNet," *arXiv preprint arXiv:2103.06191*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 6.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is attached in the supplemental materials, with the appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Our method is training-free. But we also report the hyperparameters for evaluations used in our proposed solver.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We observe that the standard deviation of the FID evaluations of DPM-Solver are rather small (mainly less than 0.01) because the FID is already averaged over 50K samples, following existing work [18, 20, 21]. The small standard deviation does not change the conclusion.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The GPU type and amount is detailed in Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix D
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our code in the supplemental materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All of the datasets used in the experiments are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We mentioned the human privacy issues of the ImageNet dataset in Appendix D.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]