Distilling ODE Solvers of Diffusion Models into Smaller Steps

Sanghwan Kim[1] Hao Tang[1,2*] Fisher Yu[1]
[1]ETH Zürich [2]Carnegie Mellon University

Abstract

Diffusion models have recently gained prominence as a novel category of generative models. Despite their success, these models face a notable drawback in terms of slow sampling speeds, requiring a high number of function evaluations (NFE) in the order of hundreds or thousands. In response, both learning-free and learning-based sampling strategies have been explored to expedite the sampling process. Learning-free sampling employs various ordinary differential equation (ODE) solvers based on the formulation of diffusion ODEs. However, it encounters challenges in faithfully tracking the true sampling trajectory, particularly for small NFE. Conversely, learning-based sampling methods, such as knowledge distillation, demand extensive additional training, limiting their practical applicability. To overcome these limitations, we introduce Distilled-ODE solvers *(D-ODE solvers), a straightforward distillation approach grounded in ODE solver formulations. Our method seamlessly integrates the strengths of both learning-free and learning-based sampling.*

D-ODE solvers are constructed by introducing a single parameter adjustment to existing ODE solvers. Furthermore, we optimize D-ODE solvers with smaller steps using knowledge distillation from ODE solvers with larger steps across a batch of samples. Comprehensive experiments demonstrate the superior performance of D-ODE solvers compared to existing ODE solvers, including DDIM, PNDM, DPM-Solver, DEIS, and EDM, particularly in scenarios with fewer NFE. Notably, our method incurs negligible computational overhead compared to previous distillation techniques, facilitating straightforward and rapid integration with existing samplers. Qualitative analysis reveals that D-ODE solvers not only enhance image quality but also faithfully follow the target ODE trajectory.

1. Introduction

Diffusion models [8, 29, 31] have recently emerged as a compelling framework for generative models, demonstrating state-of-the-art performance across diverse applications such as image generation [4, 32], text generation [2, 9], audio generation [17, 21], 3D shape generation [3, 19], video synthesis [6, 40], and graph generation [24, 34].

While diffusion models excel at producing high-quality samples and mitigating issues like mode collapse [28, 42], their sampling process often demands a substantial number of network evaluations, rendering the process slow and computationally intensive [38]. Recent research has focused on optimizing the sampling process to enhance efficiency without compromising sample quality [11, 27, 30]. Notably, methods targeting improved sampling efficiency within diffusion models fall into two main categories: *learning-free sampling* and *learning-based sampling* [39].

Learning-free sampling can be applied to pre-trained diffusion models without additional training and often involves efficient solvers for stochastic differential equations (SDEs) or ordinary differential equations (ODEs) [32]. Notable examples include DDIM [30], which employs a non-Markovian process for accelerated sampling, and PNDM [14], introducing a pseudo-numerical method for solving differential equations on given data manifolds. EDM [11] utilizes Heun's second-order method, demonstrating improved sampling quality over naive Euler's method [32]. Recently, DPM-Solver [15] and DEIS [41] leverage the semi-linear structure of diffusion ODEs and employ numerical methods of exponential integrators. These ODE solvers aim for accurate score function estimation along the ODE sampling trajectory where the density of data distribution is high [14, 43]. However, the sampling path of ODE solvers may deviate from the true trajectory, especially with a small number of denoising steps, resulting in significant fitting errors in the score function [13, 33, 38].

In contrast, learning-based sampling involves additional training to optimize specific objectives, such as knowledge distillation [27, 33] and optimized discretization [22, 36]. For instance, progressive distillation [27] iteratively distills pre-trained diffusion models into a student model requiring fewer sampling steps. Recently, Song et al. [33] introduced consistency models, trained to predict consistent outputs along the same ODE trajectory. Although distillation-based techniques enable generation within a few steps, extensive training is needed to adapt the denoising network of

*Corresponding author

the diffusion models to each dataset, sampler, and network.

To address these challenges, we propose a novel distillation method for diffusion models called Distilled-ODE solvers (D-ODE solvers), leveraging inherent sampling dynamics in existing ODE solvers. D-ODE solvers bridge the gap between learning-free and learning-based sampling while mitigating associated issues. Our approach is grounded in the observation that the outputs of the denoising network (*i.e*., denoising output) exhibit a high correlation within neighboring time steps.

D-ODE solvers introduce a single additional parameter to ODE solvers, linearly combining the current denoising network output with the previous one. This allows a more accurate estimation of the denoising output at each timestep t. For high-order solvers (*e.g*., PNDM, DEIS, and DPM-Solver), we linearly combine their high-order estimations to leverage their ability to approximate the true score function. The additional parameter is optimized for each dataset by minimizing the difference between the output of D-ODE solvers with smaller steps (student) and that of ODE solvers with larger steps (teacher). Once the optimal parameter is established, D-ODE solvers can be reused across batches during sampling while keeping the denoising network frozen. Notably, D-ODE solvers consistently improve the FID of previous ODE solvers, including first-order and high-order methods, significantly reducing the computational time of distillation. Our main contributions can be summarized as follows:

- We introduce D-ODE solvers, transferring knowledge from ODE solvers with larger steps to those with smaller steps through a simple formulation.
- D-ODE solvers alleviate the need for extensive parameter updates in pre-trained denoising networks, significantly reducing knowledge distillation time.
- In quantitative studies, our new sampler outperforms state-of-the-art ODE solvers in terms of FID scores on several image generation benchmarks.

2. Background

Forward and reverse diffusion processes. The forward process $\{\boldsymbol{x}_t \in \mathbb{R}^D\}_{t\in[0,T]}$ begins with $\boldsymbol{x}_0$ drawn from the data distribution $p_{data}(\boldsymbol{x})$ and evolves to $\boldsymbol{x}_T$ at timestep $T > 0$. Given $\boldsymbol{x}_0$, the distribution of $\boldsymbol{x}_t$ can be expressed as:

$$q_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I}), \tag{1}$$

where $\alpha_t \in \mathbb{R}$ and $\sigma_t \in \mathbb{R}$ determine the noise schedule of the diffusion models, with the signal-to-noise ratio (SNR) α_t^2/σ_t^2 strictly decreasing as t progresses [12]. This ensures that $q_T(\boldsymbol{x}_T)$, the distribution of $\boldsymbol{x}_T$, approximates pure Gaussian noise in practice.

The reverse process of diffusion models is approximated using a denoising network to iteratively remove noise. Starting from $\boldsymbol{x}_T$, the reverse process is defined with the following transition [8]:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t,t)), \tag{2}$$

where θ represents the trainable parameters in the denoising network, and $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t,t)$ and $\boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t,t)$ are the Gaussian mean and variance estimated by the denoising network θ.

SDE and ODE formulation. Song et al. [32] formulate the forward diffusion process using a stochastic differential equation (SDE) to achieve the same transition distribution as Eq. (1). Given $\boldsymbol{x}_0 \sim p_{data}(\boldsymbol{x})$, the forward diffusion process from timestep 0 to T is newly defined as:

$$d\boldsymbol{x}_t = f(t)\boldsymbol{x}_t dt + g(t)d\boldsymbol{w}_t, \tag{3}$$

where $\boldsymbol{w}_t \in \mathbb{R}^D$ is the standard Wiener process, and $f(t)$ and $g(t)$ are functions of α_t and σ_t. Song et al. [32] also introduce the reverse-time SDE based on Anderson [1], which evolves from timestep T to 0 given $\boldsymbol{x}_T \sim q_T(\boldsymbol{x}_T)$:

$$d\boldsymbol{x}_t = [f(t)\boldsymbol{x}_t - g^2(t)\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)]dt + g(t)d\bar{\boldsymbol{w}}_t, \tag{4}$$

where $\bar{\boldsymbol{w}}_t$ is the standard Wiener process in reverse time, and $\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)$ is referred to as the score function [10]. The randomness introduced by the Wiener process can be omitted to define the diffusion ordinary differential equation (ODE) in the reverse process, which corresponds to solving the SDE on average. Starting from $\boldsymbol{x}_T \sim q_T(\boldsymbol{x}_T)$, probability flow ODE from timestep T to 0 advances as follows:

$$d\boldsymbol{x}_t = [f(t)\boldsymbol{x}_t - \frac{1}{2}g^2(t)\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)]dt. \tag{5}$$

The formulation of the probability flow ODE opens up possibilities for using various ODE solvers to expedite diffusion-based sampling processes [11, 14, 15, 41].

Denoising score matching. To solve Eq. (5) during sampling, the score function $\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t)$ must be estimated. Ho et al. [8] propose estimating the score function using a noise prediction network $\boldsymbol{\epsilon_\theta}$ such that $\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t) = -\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t,t)/\sigma_t$ with $\boldsymbol{x}_t = \alpha_t\boldsymbol{x} + \sigma_t\boldsymbol{\epsilon}$. The noise prediction network $\boldsymbol{\epsilon_\theta}$ is trained using the L_2 norm, given samples drawn from p_{data}:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\sigma_t^2\boldsymbol{I})}||\boldsymbol{\epsilon_\theta}(\alpha_t\boldsymbol{x}+\sigma_t\boldsymbol{\epsilon},t) - \boldsymbol{\epsilon}||^2. \tag{6}$$

Here, Gaussian noise is added to the data $\boldsymbol{x}$ following the noise schedule (α_t, σ_t), and the noise prediction network $\boldsymbol{\epsilon_\theta}$ predicts the added noise $\boldsymbol{\epsilon}$ from the noisy sample.

Alternatively, the score function can be represented using a data prediction network $\boldsymbol{x_\theta}$ instead of $\boldsymbol{\epsilon_\theta}$ with $\nabla_{\boldsymbol{x}}\log q_t(\boldsymbol{x}_t) = (\boldsymbol{x_\theta}(\boldsymbol{x}_t,t) - \boldsymbol{x}_t)/\sigma_t^2$. The data prediction network $\boldsymbol{x_\theta}$ is trained with following L_2 norm:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\sigma_t^2\boldsymbol{I})}||\boldsymbol{x_\theta}(\alpha_t\boldsymbol{x}+\sigma_t\boldsymbol{\epsilon},t) - \boldsymbol{x}||^2. \tag{7}$$

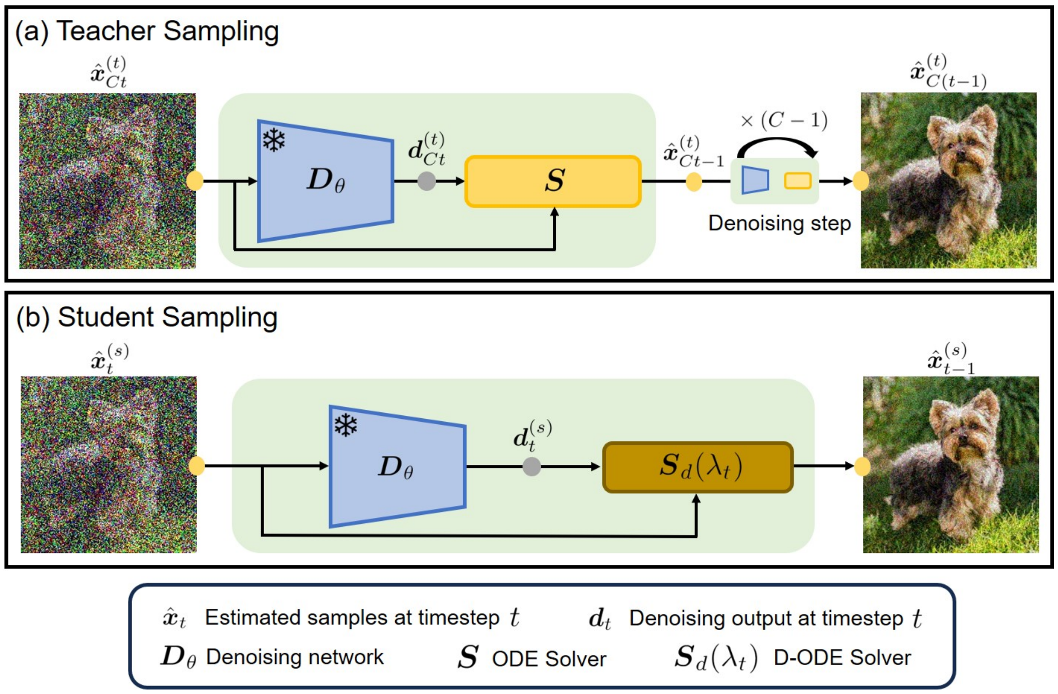


Figure 1. **The overview of D-ODE Solver.** Given an input image at timestep CT, teacher sampling performs C denoising steps to obtain the output at time step $C(T-1)$ while student sampling conducts one denoising step from an input at timestep t to an output at timestep $t-1$. Then, C steps of the teacher sampling are distilled into a single step of the student sampling by optimizing λ_t within the D-ODE solver. Note that the denoising network remains frozen for both teacher and student sampling.

It is worth noting that estimating the original data $\boldsymbol{x}$ is theoretically equivalent to learning to predict the noise $\boldsymbol{\epsilon}$ [8, 18]. While some works argue that predicting the noise empirically results in higher quality samples [8, 26], Karras et al. [11] recently achieved state-of-the-art performance using the data prediction network. In this work, we conduct comprehensive experiments with both noise and data prediction networks. For the rest of the paper, we write $\boldsymbol{D}_\theta$ to represent the denoising network of the diffusion models which can be either noise or data prediction networks.

3. The Proposed Method

Our study aims to bridge the gap between learning-based and learning-free sampling, leveraging the advantages of both approaches. We capitalize on the sampling dynamics of ODE solvers while enhancing sample quality through a straightforward and efficient knowledge distillation. This section begins with a fundamental observation of the high correlation among the outputs of the denoising network (*i.e.*, denoising output), motivating the formulation of D-ODE solvers. We then delve into the details of transferring knowledge from ODE solvers to D-ODE solvers.

3.1. Correlation between Denoising Outputs

ODE solvers typically enhance the sampling process by exploiting the output history of the denoising network, enabling the omission of many intermediate steps. Therefore, understanding the relationship between denoising outputs is crucial when developing D-ODE Solvers. Our objective is to create novel ODE solvers that harness the benefits of sampling dynamics while keeping the degrees of optimization freedom to a minimum.

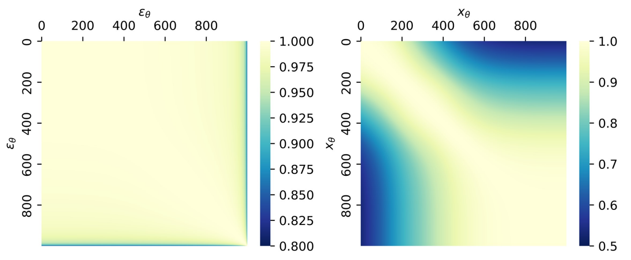


Figure 2. **Correlation between denoising outputs.** Heatmaps are drawn by cosine similarity among denoising outputs with 1000-step DDIM on CIFAR-10. Noise prediction model (left) and data prediction model (right).

Fig. 2 presents heatmaps based on cosine similarity calculations between all denoising outputs from a 1000-step DDIM [30] run. We observe that the denoising outputs from neighboring timesteps exhibit high correlations in both noise and data prediction models, with cosine similarities close to one. This observation suggests that denoising outputs contain redundant and duplicated information, allowing us to skip the evaluation of denoising networks for most timesteps. For example, the history of denoising outputs can be combined to better represent the next output, effectively reducing the number of steps required for accurate sampling. This idea is implemented in most ODE solvers, which are formulated based on the theoretical principles of solving differential equations [11, 14, 15, 36, 41].

3.2. Formulation of D-ODE Solver

As illustrated in Fig. 1, each denoising step in diffusion models typically involves two components: (1) a denoising network $\boldsymbol{D_\theta}$ and (2) an ODE solver $\boldsymbol{S}$. Given an estimated noisy sample $\hat{\boldsymbol{x}}_t$ at timestep t, the denoising network $\boldsymbol{D_\theta}$ produces a denoising output $\boldsymbol{d}_t = \boldsymbol{D_\theta}(\hat{\boldsymbol{x}}_t, t)$, and the ODE solver subsequently generates the next sample $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{S}(\boldsymbol{d}_t, \hat{\boldsymbol{x}}_t)$, utilizing the denoising output and the noisy sample at timestep t. While high-order ODE solvers also utilize the history of denoising outputs $\{\boldsymbol{d}_k\}_{k=t}^T$, we omit this notation here for simplicity. This procedure is iterated until the diffusion models reach the estimated original sample $\hat{\boldsymbol{x}}_0$.

We now introduce a D-ODE solver with a straightforward parameterization to distill knowledge from ODE solvers. We begin by outlining a fundamental method to estimate the new denoising output $\boldsymbol{O}_t$ at timestep t as a linear combination of current and previous denoising outputs $\{\boldsymbol{d}_k\}_{k=t}^T$:

$$\boldsymbol{O}_t = \sum_{k=t}^{T} \lambda_k \boldsymbol{d}_k, \tag{8}$$

where $\lambda_k \in \mathbb{R}$ is a weight parameter for each denoising output $\boldsymbol{d}_k$. With carefully chosen λ_k, we anticipate that the new denoising output can better approximate the target score function of ODE in Eq. (5), leading to improved sample quality. Some high-order ODE solvers [14, 15, 41] adopt similar formulations to Eq. (8) with mathematically determined weight parameters $\{\lambda_k\}_{k=t}^T$.

One challenge within Eq. (8) is that the value of the new denoising output $\boldsymbol{O}_t$ can be unstable and volatile depending on the values of weights $\{\lambda_k\}_{k=t}^T$. This instability is less likely to occur with numerically computed weights in ODE solvers, but convergence is not guaranteed when the weights are optimized through knowledge distillation. To generate high-quality samples, the sampling process must follow the true ODE trajectory on which the diffusion models are trained [14, 33]. In other words, the denoising network might not produce reliable outputs for samples outside the target manifold of data [13, 23, 38].

To avoid this, Eq. (8) should be constrained so that it adheres to the original ODE trajectory. Thus, the new denoising output $\boldsymbol{O}_t$ can be defined as follows:

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \sum_{k=t+1}^{T} \lambda_{k-1}(\boldsymbol{d}_t - \boldsymbol{d}_k) \tag{9}$$

$$\approx \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}). \tag{10}$$

Furthermore, we empirically find that using the denoising output from the previous timestep is sufficient for distilling knowledge from the teacher sampling (see Supplementary Material). As a result, we obtain Eq. (10) for D-ODE solvers. It is worth noting that the mean of the new denoising output $\boldsymbol{O}_t$ approximates that of the original denoising output since the mean with respect to sample $\boldsymbol{x}$ in sufficiently large batch does not change significantly between timesteps t and $t+1$ (*e.g.*, $\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\boldsymbol{O}_t] \approx \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\boldsymbol{d}_t]$). This is a key feature of D-ODE solvers, as we aim to remain on the original sampling trajectory of ODE.

In case of DDIM [30], one can simply substitute $\boldsymbol{d}_t$ with $\boldsymbol{O}_t$ to construct D-DDIM:

$$\text{DDIM: } \hat{\boldsymbol{x}}_{t-1} = \alpha_{t-1}\left(\frac{\hat{\boldsymbol{x}}_t - \sigma_t \boldsymbol{d}_t}{\alpha_t}\right) + \sigma_{t-1}\boldsymbol{d}_t, \tag{11}$$

$$\text{D-DDIM: } \hat{\boldsymbol{x}}_{t-1} = \alpha_{t-1}\left(\frac{\hat{\boldsymbol{x}}_t - \sigma_t \boldsymbol{O}_t}{\alpha_t}\right) + \sigma_{t-1}\boldsymbol{O}_t. \tag{12}$$

where (α_t, σ_t) represents a predefined noise schedule. λ_t is optimized later via knowledge distillation.

Comparison with high-order ODE solvers. High-order methods for sampling utilize the history of denoising outputs. As these methods better approximate the target score function of ODE compared to the first-order method (*e.g.*, DDIM), we apply Eq. (10) on top of their approximation to build D-ODE solvers. In other words, $\boldsymbol{d}_t$ in Eq. (10) is replaced by the high-order approximation of each method. In this way, we can involve more timesteps to obtain $\boldsymbol{O}_t$ while overcoming the bottleneck of ODE solvers with an extra parameter λ_t adapted to each dataset. Unlike high-order ODE solvers, D-ODE solvers are equipped with the parameter λ_t, optimized for a specific dataset through knowledge distillation, to further reduce the fitting error of the score function. Supplementary Material includes the specific applications of D-ODE solvers and different formulations of D-ODE solvers.

3.3. Knowledge Distillation of D-ODE Solver

In Fig. 1, the teacher sampling process initiates with the noisy sample $\hat{\boldsymbol{x}}_{CT}^{(t)}$ at timestep CT and undergoes C denoising steps to generate a sample $\hat{\boldsymbol{x}}_{C(T-1)}^{(t)}$ at timestep $C(t-1)$. Simultaneously, the student sampling process commences with a noisy sample $\hat{\boldsymbol{x}}_t^{(s)}$ at timestep t and obtains a sample $\hat{\boldsymbol{x}}_{t-1}^{(s)}$ at timestep $t-1$ after one denoising step. To optimize λ_t in the D-ODE solver $\boldsymbol{S}_d$, the teacher sampling is initially conducted for one batch, saving intermediate samples $\{\hat{\boldsymbol{x}}_k^{(t)}\}_{k=C(t-1)}^{Ct}$ as targets. The student sampling is also performed, obtaining intermediate samples $\{\hat{\boldsymbol{x}}_k^{(s)}\}_{k=t-1}^{t}$ as predictions. Subsequently, λ_t^* is determined by minimizing the difference between the targets and predictions on batch B as follows:

$$\lambda_t^* = \arg\min_{\lambda_t} \mathbb{E}_{\boldsymbol{x}\in B}||\hat{\boldsymbol{x}}_{C(t-1)}^{(t)} - S_d(\boldsymbol{d}_t^{(s)}, \hat{\boldsymbol{x}}_t^{(s)}; \lambda_t)||^2 \tag{13}$$

$$= \arg\min_{\lambda_t} \mathbb{E}_{\boldsymbol{x}\in B}||\hat{\boldsymbol{x}}_{C(t-1)}^{(t)} - \hat{\boldsymbol{x}}_{t-1}^{(s)}||^2, \tag{14}$$

where $\boldsymbol{d}_t^{(s)} = \boldsymbol{D}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_t^{(s)}, t)$ holds. The above equation is solved for every timestep t of the student sampling, yielding a set of optimal λ_t values (*e.g.*, $\lambda^* = \{\lambda_0^*, \lambda_1^*, ..., \lambda_{T-1}^*\}$). Notably, λ^* is estimated using only one batch of samples, a process that typically takes just a few CPU minutes, and can be reused for other batches later.

Algorithm 1 outlines the overall sampling procedure of the D-ODE solver. When generating N samples, it is normal to divide N into M batches and sequentially execute the sampling process for each batch B, which contains $|B| = N/M$ samples (Line 3). For the first batch, teacher sampling is conducted with denoising network $\boldsymbol{D}_{\boldsymbol{\theta}}$ and ODE solver $\boldsymbol{S}$ for CT steps to obtain intermediate outputs, which will serve as target samples (Line 7). Subsequently, student sampling takes place for T steps with D-ODE Solver $\boldsymbol{S}_d(\lambda)$ (Line 8). At this point, λ^* is estimated and saved for each timestep by solving Eq. (14) (Line 9). Starting from the second batch onwards, sampling can proceed using the frozen denoising network $\boldsymbol{D}_{\boldsymbol{\theta}}$ and D-ODE solver $\boldsymbol{S}_d(\lambda^*)$ (Line 9). It is important to note that the student's samples can be generated in just T steps, which exhibits similar quality to the teacher's samples generated over CT steps.

Algorithm 1 Sampling with D-ODE solver

```
1:  Pre-trained denoising network $\boldsymbol{D}_{\boldsymbol{\theta}}$
2:  ODE solver $\boldsymbol{S}$, D-ODE solver $\boldsymbol{S}_d(\lambda)$
3:  Number of batches $M$ with size $|B|$
4:  Student sampling steps $T$, Teacher sampling steps $CT$
5:  for $m = 1, ..., M$ do
6:      if $m = 1$ then
7:          $\{\hat{\boldsymbol{x}}_k^{(t)}\}_{k=0}^{CT}$ = Teacher-Sampling($\boldsymbol{D}_\theta$, $\boldsymbol{S}$, $CT$)
8:          $\{\hat{\boldsymbol{x}}_k^{(s)}\}_{k=0}^{T}$ = Student-Sampling($\boldsymbol{D}_\theta$, $\boldsymbol{S}_d(\lambda)$, $T$)
9:          Estimate $\lambda^* = \{\lambda_1^*, \lambda_2^*, ..., \lambda_T^*\}$ with Eq. (14)
10:     end if
11:     $\{\hat{\boldsymbol{x}}_k^{(s)}\}_{k=0}^{T}$ = Student-Sampling($\boldsymbol{D}_\theta$, $\boldsymbol{S}_d(\lambda^*)$, $T$)
12:     Save sample $\hat{\boldsymbol{x}}_0^{(s)}$
13: end for
```

4. Experiments

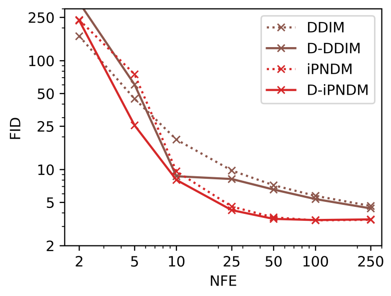
In this section, we present a comprehensive evaluation of D-ODE solvers in comparison to ODE solvers on diverse image generation benchmarks at various resolutions, including CIFAR-10 (32×32), CelebA (64×64), ImageNet (64×64 and 128×128), FFHQ (64×64), and LSUN bedroom (256×256). Our experiments cover both noise and data prediction models, each involving distinct sets of ODE solvers. The Fréchet Inception Distance (FID) [7] is employed as the evaluation metric, measured with 50K generated samples across various numbers of denoising function evaluations (NFE), following the protocol of Lu et al. [15]. The reported FID scores are averaged over three independent experiment runs with different random seeds.

For the distillation of ODE solvers, we set the scale parameter to $C = 10$ and use a batch size of $|B| = 100$, except for the LSUN bedroom dataset, where a batch size of 25 is employed due to GPU memory constraints. It is important to note that, unless explicitly specified, DDIM serves as the primary teacher sampling method to guide the student sampling. This choice is made considering that certain ODE solvers employ multi-step approaches during sampling, making it challenging to set their intermediate outputs as targets for distillation. In contrast, DDIM generates a single intermediate output per denoising step, simplifying the establishment of matching pairs between DDIM targets and student predictions. Refer to the Supplementary Material for detailed applications of D-ODE solvers and ablation studies on the scale C and batch size $|B|$.

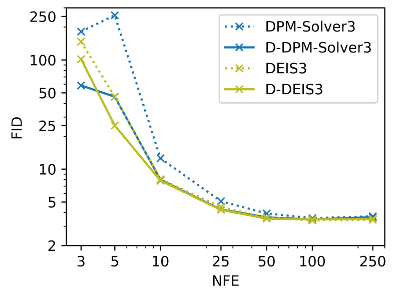
4.1. Noise Prediction Model

We apply D-ODE solvers to discrete-time ODE solvers employed in the noise prediction model, including DDIM [30], iPNDM [41], DPM-Solver [15], and DEIS [41]. For DPM-Solver and DEIS, we selected third-order methods. While these ODE solvers were primarily evaluated with NFE greater than 10, we also conduct experiments with extremely small NFE, such as 2 or 3, to assess the performance of D-ODE solvers during the initial stages of the sampling process.

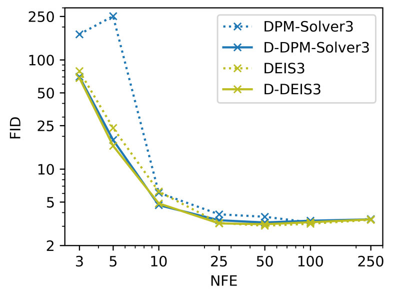
Fig. 3 illustrates that D-ODE solvers outperform ODE solvers, achieving lower FID in most NFEs. In Fig. 3a and Fig. 3d, D-DDIM outperforms DDIM when NFE exceeds 5, gradually converging to FID score similar to that of DDIM as NFE increases. It is important to note that DDIM with small NFE (2 or 5) lacks the capability to produce meaningful images, which is also reflected in the performance of D-DDIM. iPNDM, a high-order method that utilizes previous denoising outputs, consistently exhibits improvements with the D-ODE solver formulation, except at 2 NFE. This improvement is particularly notable for high-order methods like DPM-Solver3 and DEIS3. Specifically, D-DPM-Solver3 effectively alleviates the instability associated with



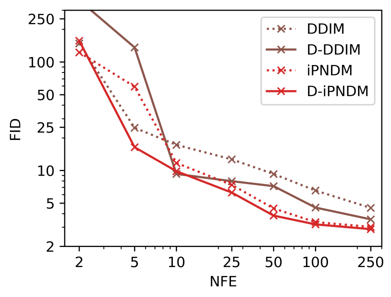
(a) CIFAR-10 (32×32)



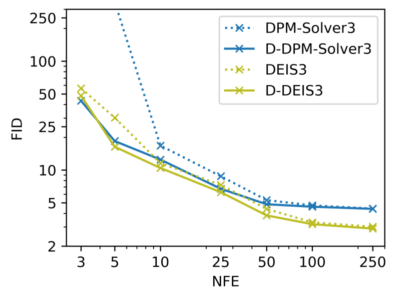
(b) CIFAR-10 (32×32)



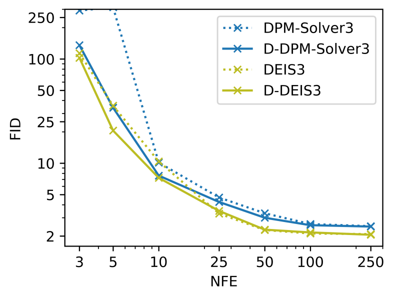
(c) ImageNet (128×128)



(d) CelebA (64×64)

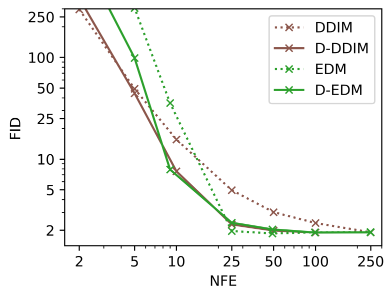


(e) CelebA (64×64)

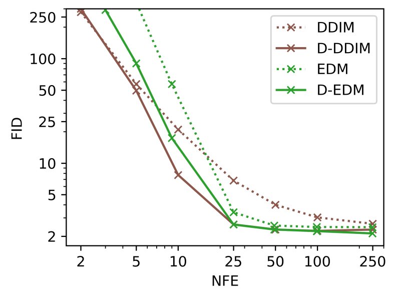


(f) LSUN bedroom (256×256)

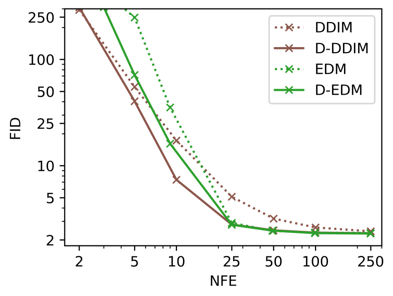
Figure 3. **Results on the noise prediction models.** Image quality measured by FID ↓ with NFE $\in \{2, 5, 10, 25, 50, 100, 250\}$. For DPM-Solver3 and DEIS3, we use 3 NFE instead of 2 NFE as the third-order method requires at least three denoising outputs. Dotted lines denote ODE solvers while straight lines represent the applications of the D-ODE solver to them.



(a) CIFAR-10 (32×32)



(b) FFHQ (64×64)



(c) ImageNet (64×64)

Figure 4. **Results on the data prediction models.** Image quality measured by FID ↓ with various NFE values (DDIM: {2, 5, 10, 25, 50, 100, 250} and EDM: {3, 5, 9, 25, 49, 99, 249}). Dotted lines denote ODE solvers and straight lines represent the applications of the D-ODE solver to them.

multi-step approaches at extremely small NFE values, surpassing the performance of DPM-Solver3 by a significant margin. While DEIS3 already provides a precise representation of the current denoising output through high-order approximation, Fig. 3 illustrates that D-DEIS3 can further enhance the approximation with parameter λ optimized for each dataset through knowledge distillation. In Supplementary Matrial, we also show that applying D-ODE solvers is effective for DPM-Solver++ [16].

4.2. Data Prediction Model

For experiments on data prediction models, we followed the configuration outlined by Karras et al. [11]. We apply the D-ODE solver to DDIM, rebuilt based on this configuration, and EDM [11], which employs Heun's second-order method. While Karras et al. [11] also re-implemented Euler-based samplers in their paper, we choose not to include them in our experiments, as EDM demonstrates superior FID scores.

Fig. 4 demonstrates that D-ODE solvers outperform ODE solvers, especially for smaller NFE. For instance, D-DDIM with 25 NFE can produce samples comparable to DDIM with 250 NFE in terms of FID, resulting in a speedup of around 10 times. With increasing NFE, FID scores of both ODE and D-ODE solvers asymptotically converge to each other. Given that the performance of student sampling is closely tied to that of teacher sampling, it is natural to observe similar FID scores for student and teacher sampling with larger NFE. Moreover, it is worth noting that around NFE 2, DDIM occasionally outperforms D-DDIM slightly. This observation suggests that the 2-step DDIM may not possess sufficient capacity to effectively distill knowledge

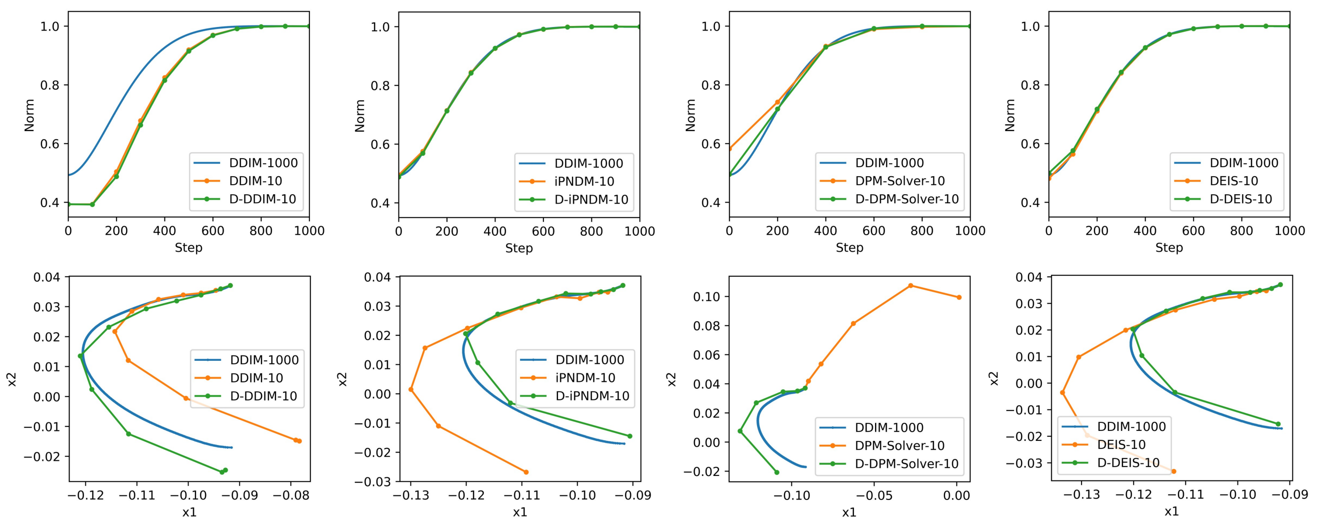


Figure 5. **Analysis on local and global characteristics.** The top row illustrates the change of norm comparing ODE and D-ODE solvers. The bottom row presents the update path of two randomly selected pixels in the images. The result of 1000-step DDIM is drawn as the target trajectory and a 10-step sampler is conducted for ODE solvers and D-ODE solvers. The figures are generated from 1000 samples using a noise prediction model trained on CIFAR-10.

Method	D-EDM (Ours)	CD [33]	PD [27]
Time	2.55	187.25	106.16

Table 1. **Comparison on computational time** to achieve 3 FID. The unit of time corresponds to the time required to generate 50k samples with 10-step DDIM.

from teacher sampling, particularly when DDIM is already generating noisy images (FID score exceeding 250).

4.3. Comparison with Previous Distillation Methods

The distillation process for D-ODE solvers typically requires only a few CPU minutes, adding negligible computational overhead to the entire sampling process. In contrast, previous distillation techniques for diffusion models [20, 27, 33] necessitate the optimization of the entire parameters of the denoising network. As a result, these methods demand a substantial amount of training time for each setting involving datasets, samplers, and networks.

Tab. 1 directly compares the computational times required by each distillation method to reach 3 FID on CIFAR-10 given the same pre-trained denoising network. The total time encompasses the distillation time following their configurations and the sampling time to generate 50k samples. For instance, D-EDM first optimizes λ and then proceeds with the sampling process, while consistency distillation (CD) [33] and progressive distillation (PD) [27] need numerous training iterations before executing a few-step sampling.

The results clearly demonstrate that optimizing ODE solvers instead of the denoising network can significantly reduce computational time and resource requirements while achieving comparable sample quality. It is important to note that the results may vary depending on the training configuration of CD and PD, as the majority of their time is consumed during the distillation process. In this context, our method aligns well with the recent trend of democratizing diffusion models by minimizing or circumventing extensive training that relies on a large number of GPUs [5, 35, 37, 43]. Supplementary Material provides a detailed explanation for distillation methods and additional comparisons with other sampling methods.

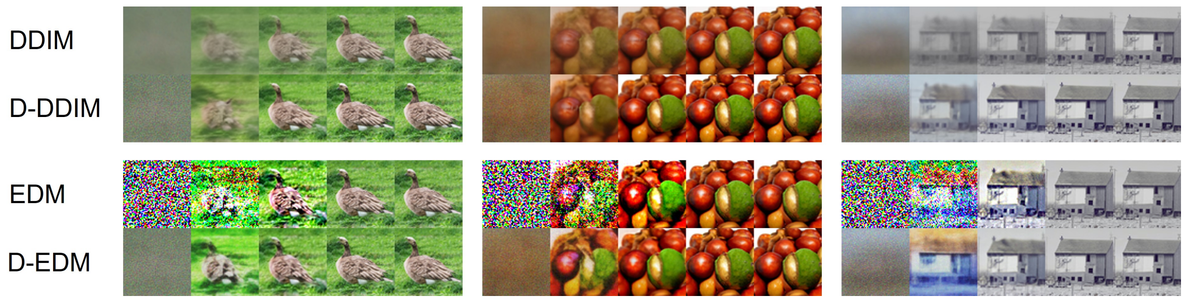
5. Analysis

This section encompasses visualizations of the sampling process and qualitative results. We initiate the exploration with a visual analysis following the methodology of Liu et al. [14], aiming to scrutinize both global and local characteristics of the sampling process. Subsequently, we delve into a comparison of the generated images produced by ODE solvers and D-ODE solvers.

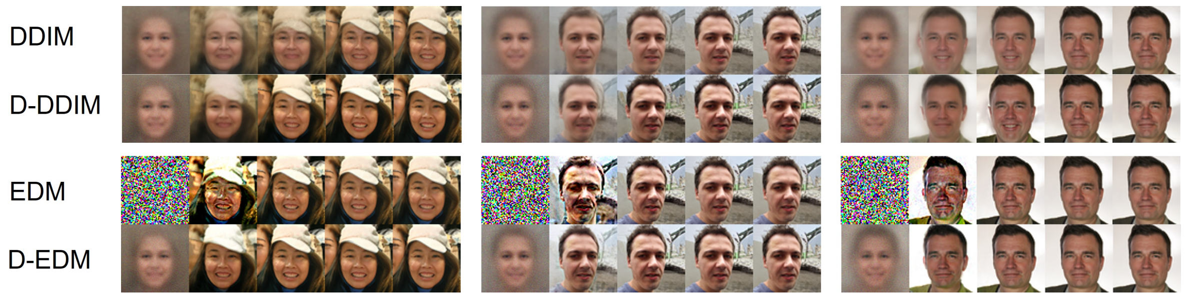
5.1. Visualization of Sampling Trajectory

To facilitate the interpretation of high-dimensional data, we employ two distinct measures: the change in the norm as a global feature and the change in specific pixel values as a local feature, following the analysis scheme provided by Liu et al. [14]. For reference, the norm of DDIM with 1000 steps is included as it adheres to the target data manifold.

In the top row of Fig. 5, the norm of D-ODE solvers closely follows the trajectory traced by the norm of ODE solvers. This observation suggests that D-ODE solvers remain within the high-density regions of the data, exerting minimal influence on the ODE trajectory. This aligns with



(a) ImageNet (64×64)



(b) FFHQ (64×64)

Figure 6. **Qualitative results.** Comparison of generated samples between ODE and D-ODE solvers. Data prediction models are used with increasing NFE (DDIM and D-DDIM: $\{2, 5, 10, 25, 50\}$, EDM and D-EDM: $\{3, 5, 9, 25, 49\}$). D-ODE solvers generate more realistic images compared to ODE solvers, especially for small NFE.

our design objective for D-ODE solvers, ensuring that the new denoising output matches the mean of the denoising output of ODE solvers, as discussed in Sec. 3.2.

In the bottom row of Fig. 5, two pixels are randomly selected from the image, and the change in their values is depicted, referencing the 1000-step DDIM as the target. Clearly, the pixel values of D-ODE solvers exhibit closer proximity to the target trajectory than those of ODE solvers. The results demonstrate that smaller-step D-ODE solvers can generate high-quality local features in samples, comparable to larger-step DDIM. It also emphasizes the importance of the data-specific parameter λ to further reduce the fitting error of the score function. In conclusion, D-ODE solvers can achieve high-quality image generation by guiding their pixels toward the desired targets while remaining faithful to the original data manifold.

5.2. Qualitative Analysis

In Fig. 6, we present a comparison of the generated images produced by ODE and D-ODE solvers using data prediction models trained on the ImageNet and FFHQ datasets. Generally, our method exhibits an improvement in image quality over ODE solvers, particularly for smaller NFE. DDIM tends to generate blurry images with indistinct boundaries, while D-DDIM produces clearer images with more prominent color contrast. EDM, especially with NFE smaller than 5, generates images characterized by high noise levels and artifacts, leading to FID scores exceeding 250. In contrast, D-EDM manages to generate relatively clear objects even at 5 NFE. Additional analysis figures and qualitative results can be found in the Supplementary Material.

6. Conclusion

In this study, we present D-ODE solvers, an innovative distillation method for diffusion models leveraging the principles of existing ODE solvers. Formulated by introducing a single parameter to ODE solvers, D-ODE solvers efficiently distill knowledge from teacher sampling with larger steps into student sampling with smaller steps, requiring minimal additional training. Our experiments showcase the efficacy of D-ODE solvers in enhancing the FID scores of state-of-the-art ODE solvers, especially in scenarios involving smaller NFE. Visual analyses provide insights into both global and local features of our method, revealing substantial improvements in image quality.

While the magnitude of improvement tends to be marginal or limited for large NFE values, eventually converging to the FID score of the teacher sampling process, D-ODE solvers remain an attractive option for augmenting sample quality with negligible additional computational cost. Their applicability extends across various samplers, datasets, and networks. However, for the generation of high-resolution images, the single-parameter nature of D-ODE solvers may prove insufficient. Exploring the incorporation of local-specific parameters, achieved through image grid divisions or latent space manipulations [25], presents an intriguing avenue for future research.

References

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. 2

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 1

[3] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 364–381. Springer, 2020. 1

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1

[5] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. *arXiv preprint arXiv:2303.09556*, 2023. 7

[6] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 35:27953–27965, 2022. 1

[7] 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. *Advances in neural information processing systems*, 30, 2017. 5

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2, 3

[9] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *arXiv preprint arXiv:2102.05379*, 3(4):5, 2021. 1

[10] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 2

[11] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 1, 2, 3, 4, 6

[12] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 2

[13] Mingxiao Li, Tingyu Qu, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. *arXiv preprint arXiv:2305.15583*, 2023. 1, 4

[14] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2021. 1, 2, 4, 7

[15] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022. 1, 2, 4, 5

[16] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 6

[17] Yen-Ju Lu, Yu Tsao, and Shinji Watanabe. A study on speech enhancement based on diffusion probabilistic model. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 659–666. IEEE, 2021. 1

[18] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. 3

[19] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 1

[20] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023. 7

[21] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021. 1

[22] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1

[23] Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. *arXiv preprint arXiv:2301.11706*, 2023. 4

[24] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020. 1

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 8

[26] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 3

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021. 1, 7

[28] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 1

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 1

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1, 3, 4, 5

[31] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1

[32] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020. 1, 2

[33] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 1, 4, 7

[34] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. 1

[35] Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of diffusion models. *arXiv preprint arXiv:2304.12526*, 2023. 7

[36] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2021. 1, 4

[37] Zike Wu, Pan Zhou, Kenji Kawaguchi, and Hanwang Zhang. Fast diffusion model. *arXiv preprint arXiv:2306.06991*, 2023. 7

[38] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2021. 1, 4

[39] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022. 1

[40] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *arXiv preprint arXiv:2203.09481*, 2022. 1

[41] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 4, 5

[42] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. *Advances in Neural Information Processing Systems*, 31, 2018. 1

[43] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *arXiv preprint arXiv:2306.09305*, 2023. 1, 7