Distilling ODE Solvers of Diffusion Models into Smaller Steps

Supplementary Material

1. Trilemma of Generative Models

Generative models face a trilemma characterized by three essential components, as outlined by [32]:

1. **High-quality samples**: Generative models should demonstrate the capacity to produce high-quality samples.
2. **Mode coverage and sample diversity**: They ought to exhibit mode coverage, ensuring that generated samples are diverse and encompass various modes within the data distribution.
3. **Fast sampling**: Efficient generative models should possess the ability to generate samples rapidly.

For instance, generative adversarial networks (GANs) [2, 6] excel in generating high-quality samples with just a single evaluation of the network. Nevertheless, GANs often struggle with generating diverse samples, resulting in poor mode coverage [28, 36]. Conversely, Variational Autoencoders (VAEs) [14] and Normalizing Flows [4] are designed to adequately ensure mode coverage but may suffer from low sample quality. Recently, diffusion models have emerged as a novel class of generative models that can generate high-quality samples comparable to GANs [3, 26], while also providing a rich variety of samples. However, conventional diffusion models often require hundreds to thousands of network evaluations for sampling, rendering them computationally expensive in practice. The primary bottleneck in the sampling process of diffusion models is closely tied to the number of denoising network evaluations. Consequently, numerous research works have explored techniques to expedite the sampling process by either skipping or optimizing sampling steps while maintaining the quality of generated samples. These techniques can be broadly classified into two categories: learning-based and learning-free sampling methods [33] as introduced in the introduction of the main paper.

2. Noise and Data Prediction Networks

The output of the denoising network should be parameterized to estimate the score function referring to the reverse-time ODE. The score function represents the gradient of the logarithm of the data distribution, indicating the direction of data with higher likelihood and less noise. One straightforward approach for the parameterization is to directly estimate the original data $\boldsymbol{x}$, in which case the score function is estimated by calculating the gradient toward the original data given the current noise level:

$$\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}_t) = \frac{\boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_t}{\sigma_t^2}. \tag{1}$$

Another approach indirectly designs the denoising network to predict noise $\boldsymbol{\epsilon}$, which represents the residual signal infused in the original sample. In this case, the score function can be calculated as:

$$\nabla_{\boldsymbol{x}} \log q_t(\boldsymbol{x}_t) = -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{\sigma_t}. \tag{2}$$

While the noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ and the data prediction network $\boldsymbol{x}_{\boldsymbol{\theta}}$ are theoretically equivalent [11, 13, 19], they reveal different characteristics during the sampling process.

Noise prediction network Noise prediction networks may initially introduce significant discrepancies between the ground truth noise and the predicted noise [1]. Since sampling commences with highly noisy samples, the denoising network lacks sufficient information to accurately predict noise [9]. Additionally, the magnitude of correction required at each timestep is relatively small, necessitating multiple timesteps to rectify such deviations [19].

Data prediction network Data prediction networks are known to offer better accuracy in the initial stages of sampling, while the noise prediction networks become preferable in later stages. Predicting data assists the denoising network in understanding the global structure of the target sample [19]. Empirical evidence shows that the predicted data is close to the ground truth at the beginning of the sampling procedure [7, 23]. However, in the later stages when substantial structures have already been formed and only minor noise artifacts need to be removed, finer details become challenging to recover [1]. Essentially, the information provided by early data prediction becomes less effective in the later stages of sampling.

Our experiments The difference between data and noise prediction networks is also evident in the figure of the main paper, illustrating correlation between denoising outputs. Predictions of $\boldsymbol{\epsilon}$ in the initial sampling stages exhibit higher correlation with each other than those in later stages, whereas predictions of $\boldsymbol{x}$ become more correlated in the later stages compared to the earlier stages. In the case of noise estimation, a small amount of noise remains in a sample for the last few timesteps, resulting in detailed and

minor changes to the sample with high variance. In conclusion, different details are modified at each timestep during the later sampling process.

On the other hand, it is challenging for a $\boldsymbol{x}$ estimator to predict the original sample from the initial noisy sample. However, its predictions become more consistent in the later stages of sampling as the sample becomes less noisy. This observation aligns with the analysis presented in Benny and Wolf [1], which indicates that the variance of the $\boldsymbol{x}$ estimator gradually decreases with more sampling steps, while the variance of the $\boldsymbol{\epsilon}$ estimator abruptly increases in the last phase of sampling.

3. Knowledge distillation in Diffusion Models

Knowledge distillation [8] was initially introduced to transfer knowledge from a larger model (teacher) to a smaller one (student), with the student model being trained to imitate the output of the teacher model. This concept can be applied to diffusion-based sampling processes to merge several timesteps (teacher) into a single timestep (student) to accelerate generation speed.

Luhman and Luhman [18] directly apply knowledge distillation to diffusion models by minimizing the difference between the outputs of a one-step student sampler and the outputs of a multi-step DDIM sampler. Thus, the student model is trained to imitate the output of the teacher model, being initialized with a pre-trained denoising network to inherit knowledge from the teacher.

Subsequently, progressive distillation [27] proposes an iterative approach to train a student network to merge two sampling timesteps of the teacher network until it achieves one-step sampling to imitate the entire sampling process. This allows the student network to gradually learn the teacher's sampling process, as learning to predict the output of two-step sampling is easier than learning to predict the output of multi-step sampling. Given a pre-trained denoising network $\boldsymbol{\theta}$ as the teacher, Salimans and Ho [27] first train a student network $\boldsymbol{\theta}'$ to predict the output of two sampling timesteps of the teacher network. The student $\boldsymbol{\theta}'$ then becomes the new teacher and a new student with parameter $\boldsymbol{\theta}''$ is trained to combine two sampling timesteps of the new teacher network $\boldsymbol{\theta}'$ until the total timestep reaches one step. The student model is parameterized and initialized with the same deep neural network as the teacher model, and progressive distillation is examined with the DDIM sampler.

Meng et al. [20] extend progressive distillation to scenarios involving classifier-free diffusion guidance, achieving single-step or few-step generation for text-to-image generation, class-conditioned generation, image-to-image translation, and image inpainting. They leverage a two-stage approach to train a student model to match the combined output of the conditional and unconditional models first, and then apply progressive distillation by setting the student model as the new teacher. Most of the configuration remains the same as Salimans and Ho [27], mainly utilizing DDIM sampler.

Recently, Song et al. [30] proposes a new class of generative models called consistency models which exploit the consistency property on the trajectory of a probabilistic flow ODE. They are trained to predict the original sample from any point on the same ODE trajectory. During training, a target network and an online network are utilized so that the online network is optimized to generate the same output as the target network, while the target network is updated with an exponential moving average. Consistency models can generate samples in a single step or a few steps by design and are also capable of image inpainting, colorization, and super-resolution in a zero-shot fashion. They can be trained either independently or via distillation, which is named as consistency training and consistency distillation, respectively. In this paper, we are interested in consistency distillation in comparison with our distillation method.

However, these distillation methods typically require extensive training to adapt to different pre-trained models, datasets, and ODE solvers, which limits their practical applicability. In this paper, we propose to optimize newly parameterized ODE solvers (D-ODE solvers) exclusively. This approach effectively distills the sampling process with larger steps into a new process with smaller steps while keeping the pre-trained denoising network fixed. Because our method does not require parameter updates for the denoising network, the distillation process can be completed in just a few CPU minutes.

4. Implementation Details of D-ODE Solvers

In this section, we explain the ODE solvers of our interest in detail and their application in the framework of D-ODE solvers. We categorize ODE solvers into two distinct types based on the nature of the diffusion timestep: discrete and continuous. Discrete-time ODE solvers include DDIM, PNDM, DPM-Solver, and DEIS, where we built our code upon Lu et al. [16], while continuous-time ODE solvers contain re-implementations of DDIM and EDM based on the work done by Karras et al. [11].

4.1. D-ODE Solvers in Noise Prediction Networks

DDIM [29] is formulated as a non-Markovian diffusion process of DDPM [9], defining a deterministic generation procedure using implicit models. Given the estimated sample $\hat{\boldsymbol{x}}_t$ at timestep t, DDIM sampling process is expressed as follows:

$$\hat{\boldsymbol{x}}_{t-1} = \alpha_{t-1}\left(\frac{\hat{\boldsymbol{x}}_t - \sigma_t \boldsymbol{d}_t}{\alpha_t}\right) + \sigma_{t-1}\boldsymbol{d}_t, \tag{3}$$

where $\boldsymbol{d}_t = \boldsymbol{D}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_t, t)$ holds with the denoising network $\boldsymbol{D}_{\boldsymbol{\theta}}$. Here, (α_t, σ_t) represents a predefined noise schedule and the denoising network is parameterized as a noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. The new denoising output $\boldsymbol{O}_t$, formulated by D-ODE solver, is defined as

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}), \tag{4}$$

following the notation in the main paper. We then simply substitute the denoising output $\boldsymbol{d}_t$ in the sampling process with the new one $\boldsymbol{O}_t$:

$$\hat{\boldsymbol{x}}_{t-1} = \alpha_{t-1}\left(\frac{\hat{\boldsymbol{x}}_t - \sigma_t \boldsymbol{O}_t}{\alpha_t}\right) + \sigma_{t-1}\boldsymbol{O}_t. \tag{5}$$

Above equation defines D-DDIM sampling process with λ_t to be optimized through knowledge distillation. In cases where the previous denoising output is unavailable (*e.g.*, at timestep T), we use the given noisy sample to define new denoising output $\boldsymbol{O}_t$, resulting in $\boldsymbol{O}_t = \boldsymbol{d}_T + \lambda_T(\boldsymbol{d}_T - \boldsymbol{x}_T)$ at initial timestep T. The assumption that both $\boldsymbol{x}_T$ and $\boldsymbol{d}_T$ follow a normal distribution $\mathcal{N}(\mathbf{0}, \sigma_t^2 \boldsymbol{I})$ in theory ensures that the mean of $\boldsymbol{O}_t$ remains consistent with the original denoising output. It is expected that $(\boldsymbol{d}_T - \boldsymbol{x}_T)$ contains information regarding the direction toward the true $\boldsymbol{x}_{T-1}$ to some extent, which actually improves the FID score in practice. Thus, we also apply this sampling recipe to other D-ODE solvers based on noise prediction networks.

PNDM [15] is based on pseudo-numerical methods on the data manifold, built upon the observation that classical numerical methods can deviate from the high-density area of data. PNDM encapsulates DDIM as a simple case and surpasses DDIM with its high-order methods. However, PNDM requires 12 NFE for the first 3 steps, making it challenging to compare with other methods using a fixed NFE. Therefore, we opt for iPNDM [35], which eliminates the need for initial warm-up steps and outperforms PNDM while maintaining the pseudo-numerical sampling process. iPNDM employs a linear combination of multiple denoising outputs to represent the current denoising output while adhering to the sampling update path of DDIM, as shown below:

$$\hat{\boldsymbol{d}}_t^{(3)} = \frac{1}{24}(55\boldsymbol{d}_t - 59\boldsymbol{d}_{t+1} + 37\boldsymbol{d}_{t+2} - 9\boldsymbol{d}_{t+3}), \tag{6}$$

$$\hat{\boldsymbol{x}}_{t-1} = \alpha_{t-1}\left(\frac{\hat{\boldsymbol{x}}_t - \sigma_t \hat{\boldsymbol{d}}_t^{(3)}}{\alpha_t}\right) + \sigma_{t-1}\hat{\boldsymbol{d}}_t^{(3)}, \tag{7}$$

where $\hat{\boldsymbol{d}}_t$ is approximated with three previous denoising outputs (*i.e.*, $\boldsymbol{d}_{t+1}$, $\boldsymbol{d}_{t+2}$, and $\boldsymbol{d}_{t+3}$) and then applied to the DDIM sampling process. Therefore, the first three denoising outputs should be defined independently as follows:

$$\hat{\boldsymbol{d}}_t^{(0)} = \hat{\boldsymbol{d}}_t, \tag{8}$$

$$\hat{\boldsymbol{d}}_t^{(1)} = \frac{3}{2}\hat{\boldsymbol{d}}_t - \frac{1}{2}\hat{\boldsymbol{d}}_{t+1}, \tag{9}$$

$$\hat{\boldsymbol{d}}_t^{(2)} = \frac{1}{12}(23\hat{\boldsymbol{d}}_t - 16\hat{\boldsymbol{d}}_{t+1} + 5\hat{\boldsymbol{d}}_{t+2}). \tag{10}$$

Leveraging these newly defined denoising outputs $\hat{\boldsymbol{d}}_t^{(p)}$ ($p = 3$ after three timesteps) by iPNDM, we construct the sampling process of D-iPNDM, where the new denoising output $\boldsymbol{O}_t$ can be defined as

$$\boldsymbol{O}_t = \hat{\boldsymbol{d}}_t^{(p)} + \lambda_t(\hat{\boldsymbol{d}}_t^{(p)} - \hat{\boldsymbol{d}}_{t+1}^{(p)}). \tag{11}$$

Then, $\hat{\boldsymbol{d}}_t^{(p)}$ in Eq. (7) is replaced by $\boldsymbol{O}_t$, which leads to the same update rule as Eq. (5) with different formulation of $\boldsymbol{O}_t$.

DPM-Solver [16] utilizes the semi-linear structure of probabilistic flow ODEs by solving the exact formulation of the linear part of ODEs and approximating the weighted integral of the neural network with exponential integrators [10]. DPM-Solver offers first-order, second-order, and third-order methods, with the first-order variant corresponding to DDIM. For single step approach, DPM-Solver strategically divides the total sampling steps using these different-order methods. For instance, DPM-Solver2 (second-order DPM-Solver) is employed 5 times to generate a sample comprising 10 denoising steps, with the denoising network being evaluated twice within DPM-Solver2. To achieve 15 denoising steps, DPM-Solver2 is applied 7 times, and DPM-Solver1 (or DDIM) is applied during the final denoising step.

In this section, we delve into the formulation of D-DPM-Solver2, and the application to DPM-Solver3 and DPM-Solver++ [17] follows a similar approach. First, we denote $\tau_t = \log(\alpha_t/\sigma_t)$ as the logarithm of the signal-to-noise ratio (SNR), and τ_t is a strictly decreasing function as t increases. Consequently, we can establish an inverse function mapping from τ to t, denoted as $t_\tau(\cdot) : \mathbb{R} \to \mathbb{R}$. Now, we can outline DPM-Solver2 with the following steps:

$$t - \frac{1}{2} = t_\tau(\frac{\tau_{t-1} + \tau_t}{2}), \tag{12}$$

$$\hat{\boldsymbol{x}}_{t-\frac{1}{2}} = \frac{\alpha_{t-\frac{1}{2}}}{\alpha_t}\hat{\boldsymbol{x}}_t - \sigma_{t-\frac{1}{2}}(e^{\frac{h_t}{2}} - 1)\boldsymbol{d}_t, \tag{13}$$

$$\hat{\boldsymbol{x}}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}\hat{\boldsymbol{x}}_t - \sigma_{t-1}(e^{h_t} - 1)\boldsymbol{d}_{t-\frac{1}{2}}. \tag{14}$$

In these equations, $h_t = \tau_{t-1} - \tau_t$, and $\hat{\boldsymbol{x}}_{t-\frac{1}{2}}$ represents the intermediate output between timestep $t-1$ and t. Since DPM-Solver2 utilizes a two-stage denoising step, we must define two denoising outputs $\boldsymbol{O}_t$ and $\boldsymbol{O}_{t-\frac{1}{2}}$ to formulate D-

DPM-Solver2 with λ_t and $\lambda_{t-\frac{1}{2}}$ optimized through knowledge distillation:

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}_{t+\frac{1}{2}}), \tag{15}$$

$$\boldsymbol{O}_{t-\frac{1}{2}} = \boldsymbol{d}_{t-\frac{1}{2}} + \lambda_{t-\frac{1}{2}}(\boldsymbol{d}_{t-\frac{1}{2}} - \boldsymbol{d}_t). \tag{16}$$

These new denoising outputs are then applied in Eq. (13) and Eq. (14) to define the sampling process of D-DPM-Solver2:

$$\hat{\boldsymbol{x}}_{t-\frac{1}{2}} = \frac{\alpha_{t-\frac{1}{2}}}{\alpha_t}\hat{\boldsymbol{x}}_t - \sigma_{t-\frac{1}{2}}(e^{\frac{h_t}{2}} - 1)\boldsymbol{O}_t, \tag{17}$$

$$\hat{\boldsymbol{x}}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}\hat{\boldsymbol{x}}_t - \sigma_{t-1}(e^{h_t} - 1)\boldsymbol{O}_{t-\frac{1}{2}}. \tag{18}$$

Similar to DPM-Solver, DEIS [35] employs an exponential integrator to exploit the semi-linear structure of the reverse-time diffusion process. In particular, they propose the use of high-order polynomials to approximate the non-linear term in ODEs as shown below:

$$P_r(t) = \sum_{j=0}^{r} C_{tj}\boldsymbol{d}_{t+j}, \tag{19}$$

$$\hat{\boldsymbol{x}}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}\hat{\boldsymbol{x}}_t + P_r(t), \tag{20}$$

where $\{C_{tj}\}_{j=0}^{r}$ is numerically determined through weighted integration to approximate the true ODE trajectory. DEIS offers several variants based on the numerical method used to estimate C_{tj}, and for our experiments, we choose tAB-DEIS as it exhibits the most promising results among the variants. Additionally, Zhang and Chen [35] explores DEIS for different values of $r \in \{1, 2, 3\}$ where larger values of r generally lead to improved approximations of the target score function. It is worth noting that DDIM can be seen as a special case of tAB-DEIS with $r = 0$.

Referring to Eq. (20), we define a new denoising output $\boldsymbol{O}_t$ and the sampling process of D-DEIS as follows:

$$\boldsymbol{O}_t = P_r(t) + \lambda_t(P_r(t) - P_r(t+1)), \tag{21}$$

$$\hat{\boldsymbol{x}}_{t-1} = \frac{\alpha_{t-1}}{\alpha_t}\hat{\boldsymbol{x}}_t + \boldsymbol{O}_t. \tag{22}$$

4.2. D-ODE Solvers in Data Prediction Networks

In our study, we newly implement DDIM [29] in a continuous setting using the parameterization of the data prediction network. We follow the configurations outlined by Karras et al. [11]. The sampling process for this modified DDIM is defined as follows:

$$s_t = \frac{\boldsymbol{d}_t - \hat{\boldsymbol{x}}_t}{\sigma_t}, \tag{23}$$

$$\hat{\boldsymbol{x}}_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})s_t, \tag{24}$$

where $\boldsymbol{d}_t = \boldsymbol{D}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_t, t)$ holds, and s_t approximates the score function, directing toward the high-density area of the data. The denoising network is parameterized as the data prediction network $\boldsymbol{x}_{\boldsymbol{\theta}}$, and the denoising step is carried out in Eq. (24) based on the difference in noise levels measured by $(\sigma_t - \sigma_{t-1})$.

Similar to D-DDIM with the noise prediction network, the new denoising output $\boldsymbol{O}_t$ for D-DDIM is defined as

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}). \tag{25}$$

Then, $\boldsymbol{O}_t$ is incorporated into the sampling process of DDIM instead of $\boldsymbol{d}_t$ as follows:

$$s_t = \frac{\boldsymbol{O}_t - \hat{\boldsymbol{x}}_t}{\sigma_t}, \tag{26}$$

$$\hat{\boldsymbol{x}}_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})s_t. \tag{27}$$

Karras et al. [11] introduce EDM sampler based on Heun's second-order method, which achieves a state-of-the-art FID score on CIFAR-10 and ImageNet64. They utilize a novel ODE formulation, parameter selection, and modified neural architectures. The EDM sampling process is shown as follows:

$$s_t = \frac{\boldsymbol{d}_t - \hat{\boldsymbol{x}}_t}{\sigma_t}, \tag{28}$$

$$\hat{\boldsymbol{x}}'_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})s_t, \tag{29}$$

$$s'_t = \frac{\boldsymbol{d}'_{t-1} - \hat{\boldsymbol{x}}'_{t-1}}{\sigma_{t-1}}, \tag{30}$$

$$\hat{\boldsymbol{x}}_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})(\frac{1}{2}s_t + \frac{1}{2}s'_t), \tag{31}$$

where $\boldsymbol{d}'_{t-1} = \boldsymbol{D}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}'_{t-1}, t-1)$ holds. The first stage of EDM with Eq. (28) and Eq. (29) is equivalent to DDIM, and then the score function is more accurately estimated in the second stage with Eq. (30) and Eq. (31) by linearly combining two estimations s_t and s'_t. Notably, 18 steps of EDM sampling correspond to 35 NFE, as one step of EDM involves two network evaluations, and Eq. (30) and Eq. (31) are not computed at the last step.

To construct the sampling process of D-EDM, we define two denoising outputs:

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}'_{t+1}), \tag{32}$$

$$\boldsymbol{O}'_{t-1} = \boldsymbol{d}'_{t-1} + \lambda_t(\boldsymbol{d}'_{t-1} - \boldsymbol{d}_t). \tag{33}$$

Consequently, the sampling steps for D-EDM are described as follows:

$$s_t = \frac{\boldsymbol{O}_t - \hat{\boldsymbol{x}}_t}{\sigma_t}, \tag{34}$$

$$\hat{\boldsymbol{x}}'_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})s_t, \tag{35}$$

$$s'_t = \frac{\boldsymbol{O}'_{t-1} - \hat{\boldsymbol{x}}'_{t-1}}{\sigma_{t-1}}, \tag{36}$$

$$\hat{\boldsymbol{x}}_{t-1} = \hat{\boldsymbol{x}}_t + (\sigma_t - \sigma_{t-1})(\frac{1}{2}s_t + \frac{1}{2}s'_t). \tag{37}$$

4.3. Various Interpretations of D-ODE Solvers

New denoising output $\boldsymbol{O}_t$ in D-ODE solvers is formulated based on the observation that denoising outputs are highly correlated, and it is essential to retain the same mean as the original outputs. We rewrite the definition of our denoising output as follows:

$$\boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_t(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}). \tag{38}$$

The above formulation can be interpreted to calculate interpolation (or extrapolation) between the current and previous denoising outputs to estimate the accurate score function. Therefore, D-ODE solvers can be seen as the process of dynamically interpolating (or extrapolating) the denoising outputs with λ_t optimized through knowledge distillation. Similarly, Zhang et al. [34] proposed the use of extrapolation on the current and previous estimates of the original data $\hat{\boldsymbol{x}}_t$. They argued that extrapolating between two predictions includes useful information toward the target data by refining the true mean estimation. Although accurate extrapolation requires grid search for parameter tuning, they demonstrated improvements in the FID of various ODE solvers.

Another interpretation is based on the work of Permenter and Yuan [22], who matched the denoising process to gradient descent applied to the Euclidean distance function under specific assumptions. They reinterpreted diffusion models using the definition of projection onto the true data distribution and proposed a new sampler by minimizing the error in predicting $\boldsymbol{\epsilon}$ between adjacent timesteps. Their sampler corresponds to D-DDIM with $\lambda_t = 1$ selected via grid search, and it outperforms DDIM and PNDM.

The last interpretation is that D-ODE solvers accelerate the convergence of sample generation in a way similar to how momentum boosts optimization in SGD [31]. Just as SGD with momentum utilizes the history of previous gradients to speed up parameter updates in a neural network, D-ODE solvers leverage previous denoising outputs to accelerate the convergence of sampling. An interesting future direction could explore whether advanced optimizers used in machine learning models [5, 12, 25] can be effectively applied to diffusion models.

4.4. Various Formulations of D-ODE Solvers

To further validate the effectiveness of D-ODE solvers, we explore different formulations of D-ODE solvers based on DDIM. For example, we can estimate parameters for two adjacent denoising outputs separately instead of optimizing a single parameter λ_t, which we name D-DDIM-Sep. D-DDIM-Sep corresponds to Eq. (8) of the main paper with $T = t + 1$. Eq. (8) of the main paper is represented as D-DDIM-All where all previous denoising outputs are utilized to estimate the new one. Additionally, we include D-DDIM which is shown as Eq. (10) of the main paper and D-DDIM-2 which is equal to Eq. (9) of the main paper with $T = t+2$. All methods are explicitly presented below for comparison, with $\boldsymbol{d}_t = \boldsymbol{D}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_t, t)$:

$$\text{DDIM}: \quad \boldsymbol{d}_t, \tag{39}$$

$$\text{D-DDIM-Sep}: \quad \boldsymbol{O}_t = \lambda_{t1}\boldsymbol{d}_t + \lambda_{t2}\boldsymbol{d}_{t+1}, \tag{40}$$

$$\text{D-DDIM-All}: \quad \boldsymbol{O}_t = \sum_{k=t}^{T} \lambda_k \boldsymbol{d}_k, \tag{41}$$

$$\text{D-DDIM}: \quad \boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_{t1}(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}), \tag{42}$$

$$\text{D-DDIM-2}: \quad \boldsymbol{O}_t = \boldsymbol{d}_t + \lambda_{t1}(\boldsymbol{d}_t - \boldsymbol{d}_{t+1}) + \lambda_{t2}(\boldsymbol{d}_t - \boldsymbol{d}_{t+2}). \tag{43}$$

NFE	10	25	50
DDIM	18.85	9.79	7.17
D-DDIM-Sep	79.21	26.40	11.50
D-DDIM-All	179.67	36.65	18.48
D-DDIM	**8.67**	**8.18**	**6.55**
D-DDIM-2	18.75	9.83	7.21

Table 1. **Comparison on various D-ODE solver formulations.** FID is measured on CIFAR-10 with the noise prediction model and the best FID is bolded.

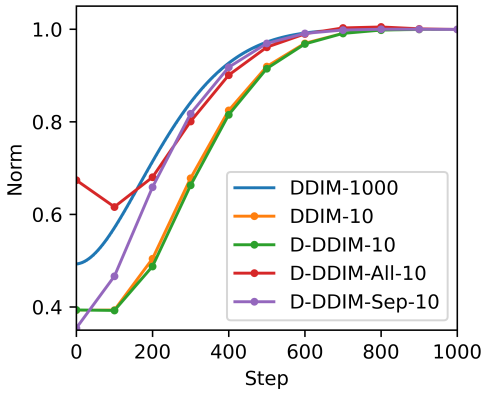


Figure 1. **Comparison of the change of norm with different formulations.** We adopt the same setting as Fig. 5 in the main paper.

We examined the five formulations mentioned above on CIFAR-10 with different NFE, while all other configurations for distillation and sampling are remained the same. As shown in Tab. 1, D-DDIM outperforms all other formulations, and other formulations such as D-DDIM-Sep, D-DDIM-All, and D-DDIM-2 even worsen the FID score compared to DDIM. D-DDIM-Sep and D-DDIM-All results in especially higher FID scores which can be interpreted that the sampling process does not properly converge

to generate realistic samples. As we pointed out in the main paper, independently estimated parameters may deviate from the target trajectory of ODE solvers. This is due to the fact that the set of λ in Eq. (40) and Eq. (41), determined by distillation, can be volatile without any constraints and may not reflect the general sampling rules across different batches. D-DDIM-2 also does not improve the FID score of DDIM. One possible reason for this is that parameters optimized on one batch may not be applicable to others. Since the two parameters are optimized on only one batch, fine-grained estimation of denoising predictions like D-DDIM-2 may not be valid for all batches.

Moreover, we display the change of norm in Fig. 1 referring to Fig. 5 of the main paper. While D-DDIM-All-10 and D-DDIM-Sep-10 initially seem to follow the target trajectory (*i.e.*, DDIM-1000), they highly deviate from either the target or the original ODE trajectory (*i.e.*, D-DDIM-10) at last, which matches with the high FID scores in Tab. 1. As mentioned in Sec. 3.2 this is due to the instability inherent in Eq. (41).

5. Experiment Details

Model architectures For the noise prediction models, we follow the architectures and configurations of Ho et al. [9] and Dhariwal and Nichol [3], utilizing their pre-trained models. Specifically, we adopt the model architecture and configuration in DDPM [9] for experiments on CIFAR-10 and CelebA 64×64. For ImageNet 128×128 and LSUN Bedroom 256×256, we use the corresponding network architectures from Dhariwal and Nichol [3]. In experiments with the data prediction models, we utilize the configurations and pre-trained models from Karras et al. [11] for CIFAR-10, FFHQ 64×64, and ImageNet 64×64.

Distillation configurations As outlined in the algorithm of main paper, we first perform teacher sampling with CT steps to set target samples, followed by student sampling with T steps to match the student's outputs with the teacher's targets. For most D-ODE solvers, we use DDIM sampling as the teacher sampling method, as it generates one denoising output per denoising step, enabling one-to-one matching between targets and predictions. For iPNDM and DEIS, we use themselves as the teacher method for distillation, respectively (*e.g.*, DEIS with CT steps as the teacher and D-DEIS with T steps as the student). While they use a linear combination of previous denoising outputs to estimate current denoising predictions, the sampling dynamics are the same as DDIM. Therefore, the teacher's targets and student's predictions can be easily matched.

Moreover, student sampling is performed sequentially to optimize λ in D-ODE solvers. In other words, λ_t is first estimated via distillation and then the next sample at timestep $t+1$ is generated with optimized D-ODE solvers at timestep t during student sampling. This approach helps stabilize the sampling process, as λ_{t+1} is estimated based on previously generated samples from D-ODE solvers with λ_t^*. As a result, it can alleviate exposure bias [21, 24] with precisely estimated λ.

Sampling details For simplicity, we adopt uniformly divided timesteps for all ODE solvers. We generate 50K samples and report the mean FID score calculated after three runs with different seeds. All experiments are conducted using GPUs, including NVIDIA TITAN Xp, Nvidia V100, and Nvidia A100. We fix the scale $C = 10$ and batch size $|B| = 100$, except for LSUN Bedroom where $|B| = 25$ due to memory constraints. Ablation studies on these two parameters are presented in Sec. 6.

Several design choices need to be made for each ODE solver. PNDM requires 12 NFE for the first 3 steps, making it challenging to compare with other methods using a fixed NFE. Therefore, we adopt iPNDM [35], which does not require initial warm-up steps and outperforms PNDM. DEIS offers various versions of ODE solvers, among which we select tAB-DEIS, exhibiting the best FID score in their experiments. DPM-Solver combines different-order solvers using adaptive step sizes. For simplicity, we opt for the single-step DPM-Solver, which sequentially uses DPM-Solver1, DPM-Solver2, and DPM-Solver3 to compose the total timesteps. While EDM allows stochastic sampling by its design, we employ deterministic sampling to obtain a definite target sample generated by teacher sampling.

6. Ablation Studies

We conduct ablation studies on two key parameters for the distillation of D-ODE solvers: the scale S and the batch size $|B|$. The scale S determines the number of steps for the teacher sampling, with the teacher sampling going through S times more denoising steps compared to the student sampling. A larger scale S results in a better target generated by the teacher sampling and can be viewed as increasing the guidance strength of the teacher during distillation. It is also crucial to choose an appropriate batch size $|B|$ since the optimal λ is estimated on a single batch B and then reused for other batches. Thus, the batch size should be large enough to encompass different modes of samples within the dataset, while excessively large batch size may not fit into GPU memory.

We test various scales in Tab. 2a using the noise prediction models trained on CIFAR-10. As the scale increases, the FID score consistently improves across different NFE values. With a larger scale S, the student sampling is

NFE	10	25	50
5	$9.68_{\pm 0.10}$	$8.20_{\pm 0.06}$	$6.52_{\pm 0.02}$
10	$8.83_{\pm 0.10}$	$8.09_{\pm 0.03}$	$6.55_{\pm 0.09}$
20	$8.52_{\pm 0.04}$	$8.01_{\pm 0.03}$	$\mathbf{6.50}_{\pm 0.01}$
30	$\mathbf{8.41}_{\pm 0.05}$	$\mathbf{7.87}_{\pm 0.05}$	$\mathbf{6.50}_{\pm 0.01}$

(a) Different Scale S

NFE	10	25	50
5	$9.33_{\pm 0.66}$	$7.75_{\pm 0.13}$	$6.64_{\pm 0.09}$
10	$8.83_{\pm 0.58}$	$7.79_{\pm 0.09}$	$6.55_{\pm 0.07}$
50	$\mathbf{8.03}_{\pm 0.08}$	$7.69_{\pm 0.08}$	$6.58_{\pm 0.05}$
100	$8.22_{\pm 0.10}$	$\mathbf{7.68}_{\pm 0.03}$	$\mathbf{6.50}_{\pm 0.09}$

(b) Different Batch Size $|B|$

Table 2. **Ablation studies on scale $C \in \{5, 10, 20, 30\}$ and batch size $|B| \in \{5, 10, 50, 100\}$.** CIFAR-10 with noise prediction models are employed for evaluation. We report mean and standard deviation after 3 runs (mean $\pm$ std) and the best FID is bolded.

strongly guided by the accurate target of teacher sampling, resulting in a lower FID. However, the effect of the guidance scale weakens with increasing NFE. This is reasonable since the performance of student sampling depends heavily on that of teacher sampling, and the teacher's FID score eventually converges to a certain value. As the maximum number of timesteps is 1000 for discrete timesteps, scales 20 and 30 at 50 NFE generate samples guided by the same teacher sampling.

In Tab. 2b, D-ODE solvers with various batch sizes also exhibit clear tendency. As the batch size increases, both the FID score and variance tend to decrease. With relatively large NFE values, FID scores and variance converge to a certain point. As the effect of distillation diminishes with higher NFE, even a small batch size results in low variance. We choose a batch size of 100 for most datasets, which is sufficient to capture the inherent variety of the dataset and reduce variance compared to a smaller batch size.

7. More Comparisons

In this section, we present further comparisons between D-ODE solvers and previous learning-based (knowledge distillation) and learning-free methods. Fig. 2a displays FID scores with varying NFE on CIFAR-10, including consistency distillation (CD) [30], which can perform a one-step or few-step sampling, and progressive distillation (PD) [27], allowing a sampling with steps in a geometric sequence (*e.g.*, 1, 2, 4, ..., 1024). D-EDM requires at least two steps to utilize previous denoising outputs.

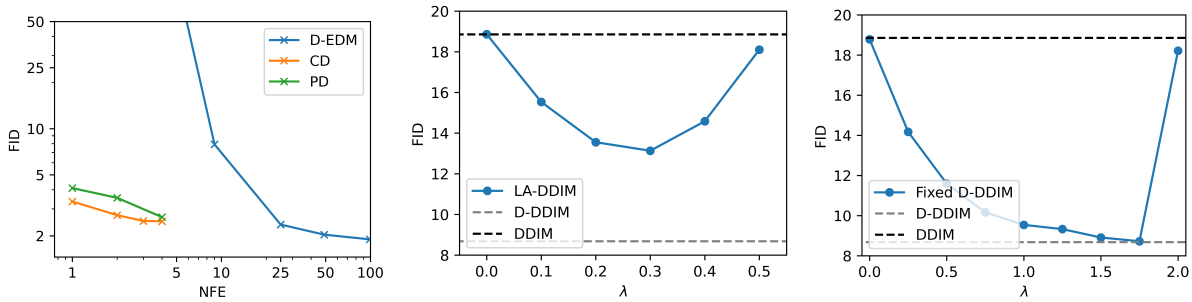
NFE	10	25	50
DDIM	18.85	9.79	7.17
D-DDIM	**8.67**	8.18	6.55
Fixed-D-DDIM ($\lambda = 0.5$)	11.45	**7.00**	**5.27**
LA-DDIM ($\lambda = 0.1$)	15.24	8.57	6.29

Table 3. **Comparison with learning-free samplers** on CIFAR-10 with noise prediction models. The best FID is bolded.

Overall, CD outperforms other methods in terms of FID on one-step generation. However, it is important to note that this comparison does not account for training time. For instance, Song et al. [30] reported that consistency models on CIFAR-10 utilized 8 Nvidia A100 GPUs for training. On the other hand, simply generating 50K samples for 30 steps takes less than 30 minutes on a single A100 GPU, achieving similar sample quality to consistency models. while CD and PD are attractive options for practitioners with ample computational resources, given their ability to enable one-step generation, the major advantage of D-ODE solvers lies in their capacity to enhance existing ODE solver-based samplers with minimal modifications and fast optimization.

Recently, Zhang et al. [34] introduced lookahead diffusion models which enhance the FID scores of existing ODE solvers by refining mean estimation using previous data predictions. They achieve this by extrapolating previous predictions of initial data to approximate the target data. Unlike D-ODE solvers, lookahead models require parameter λ to be chosen through grid search, with a default setting of $\lambda = 0.1$ during experiments. Following their configuration, we compare lookahead diffusion models of DDIM, so-called LA-DDIM, with our D-DDIM in Tab. 3. The table shows that, except at 50 NFE, D-DDIM outperforms LA-DDIM.

Inspired by LA-DDIM, we also experiment with fixing λ_t in D-DDIM as a constant λ and optimizing it through grid search. We refer to this modified approach as Fixed-D-DDIM. In Fig. 2b and Fig. 2c, we conduct grid searches on λ using a 10-step sampler on CIFAR-10. Additionally, we provide the FID scores of DDIM and D-DDIM as references (dotted lines). Despite the grid search performed on LA-DDIM, it is unable to match the FID of D-DDIM. On the other hand, Fixed-D-DDIM achieves the same FID as D-DDIM with sufficient grid search. This suggests that leveraging denoising outputs is a more efficient strategy than relying on initial data predictions. Moreover, Fixed-D-DDIM further improves upon D-DDIM's performance at 25 and 50 NFE, indicating the potential for finding an even better λ value that results in a lower FID. Future research directions could explore various methods to efficiently determine λ. It is important to highlight that the FID of LA-DDIM and



(a) Distillation methods (b) LA-DDIM (c) Fixed-D-DDIM

Figure 2. **Comparison figures.** (a) FID scores over NFE for distillation methods (CD, PD, and D-EDM). (b) FID scores over λ with LA-DDIM. (c) FID scores over λ with Fixed-D-DDIM

Fixed-D-DDIM varies depending on the chosen λ. However, D-DDIM's advantage over other methods is its independence from grid search, with sampling times comparable to DDIM.

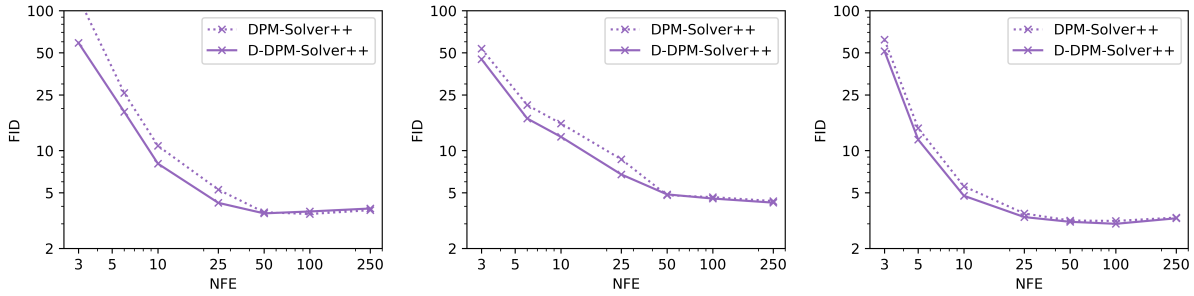
8. More Experiments on DPM-Solver++

Built upon DPM-Solver [16], DPM-Solver++ [17] addresses the instability in the previous multi-step approach of solving diffusion ODE and adopts thresholding methods to constrain the solution within the range of the original data. Similar to the formulation of D-DPM-Solver explained in Sec. 4.1, we apply our new denoising outputs to replace the original denoising output. Fig. 3 demonstrates that applying D-ODE solvers to DPM-Solver++ can further improve the image quality through distillation.

In addition, we present extra experiment results in Fig. 4 with noise prediction models on CIFAR-10, CelebA64, and ImageNet128.

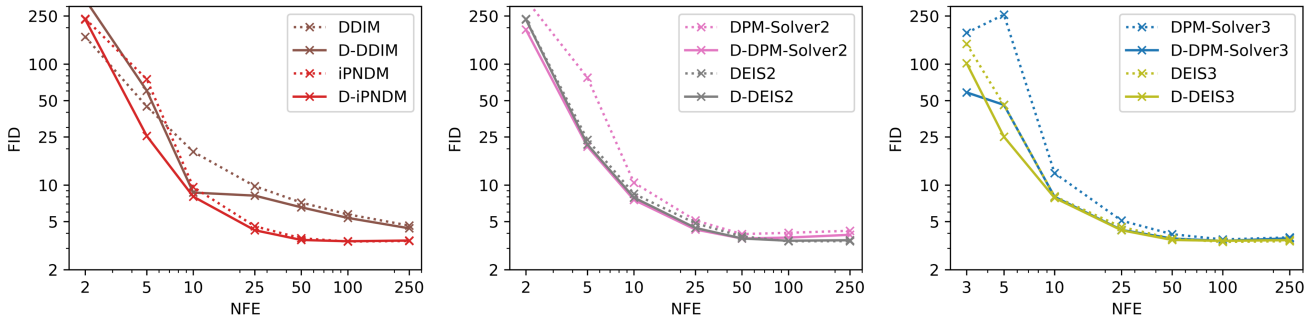
9. Analysis Figures and Qualitative Results

In Fig. 5, more analysis figures similar to Fig. 5 of the main paper are shown with different pixels. We also show more qualitative results in Fig. 6, Fig. 7, Fig. 8, and Fig. 9.

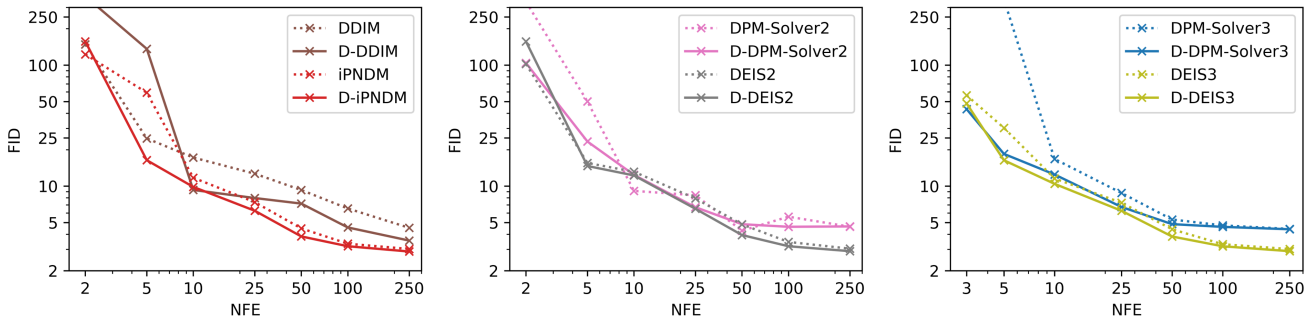


(a) CIFAR-10 (32×32) (b) CelebA (64×64) (c) ImageNet (128×128)

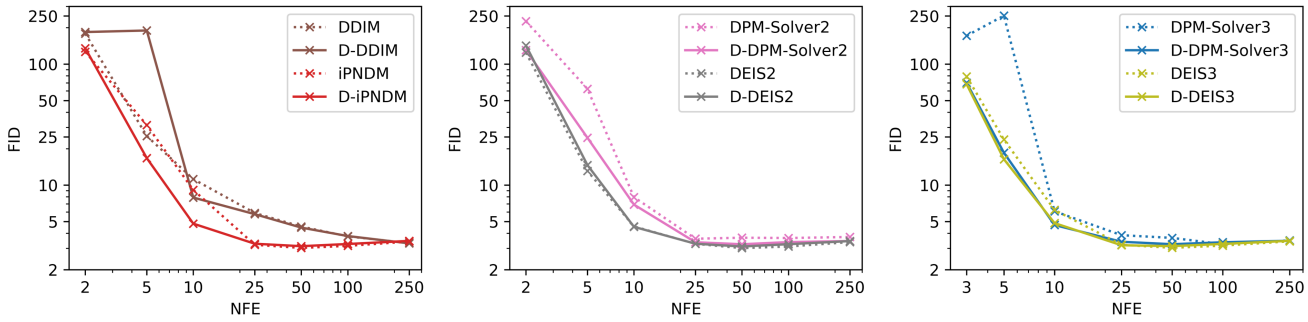
Figure 3. **Image quality measured by FID ↓ with DPM-Solver++.** Dotted lines denote DPM-Solver++ while straight lines represent D-DPM-Solver++.



(a) CIFAR-10 (32×32)



(b) CelebA (64×64)



(c) ImageNet (128×128)

Figure 4. **Image quality measured by FID ↓** with varying 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 (DDIM, iPNDM, DPM-Solver, and DEIS) while straight lines represent the applications of D-ODE solver to them (D-DDIM, D-iPNDM, D-DPM-Solver, and D-DEIS).

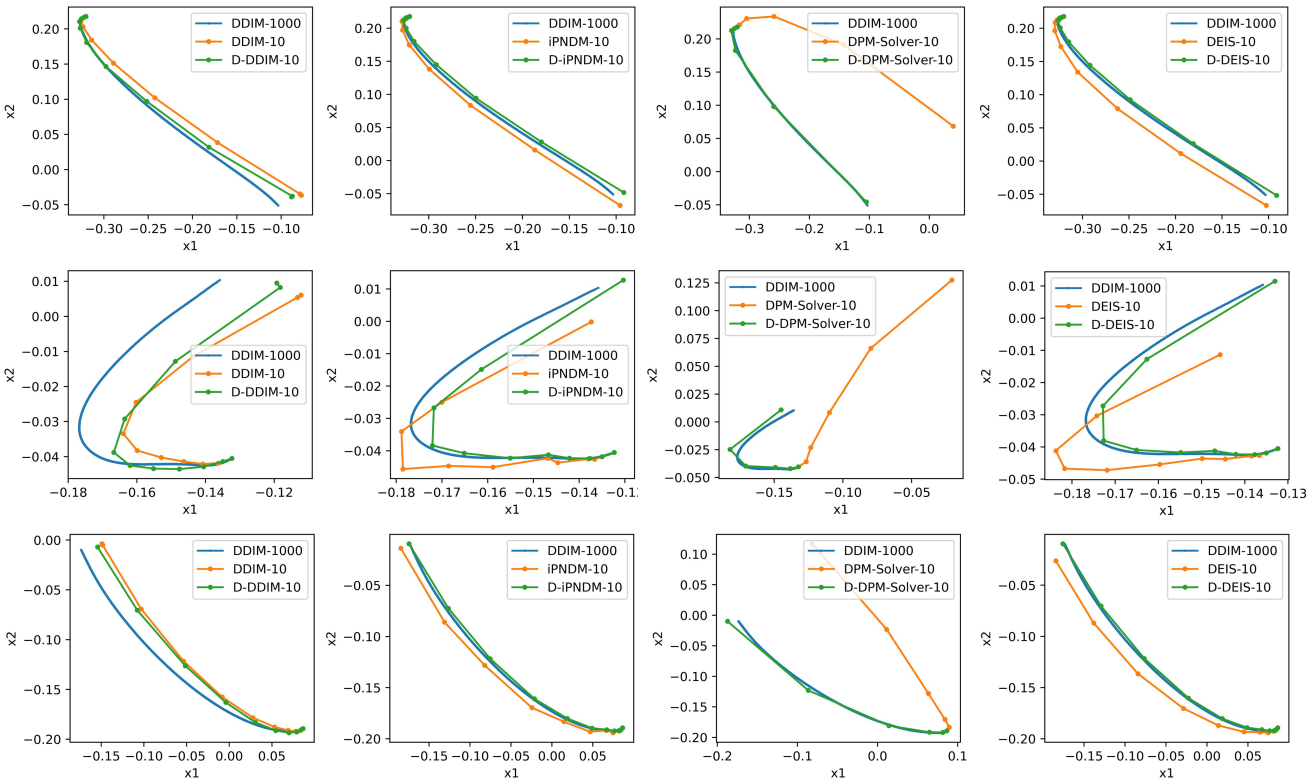
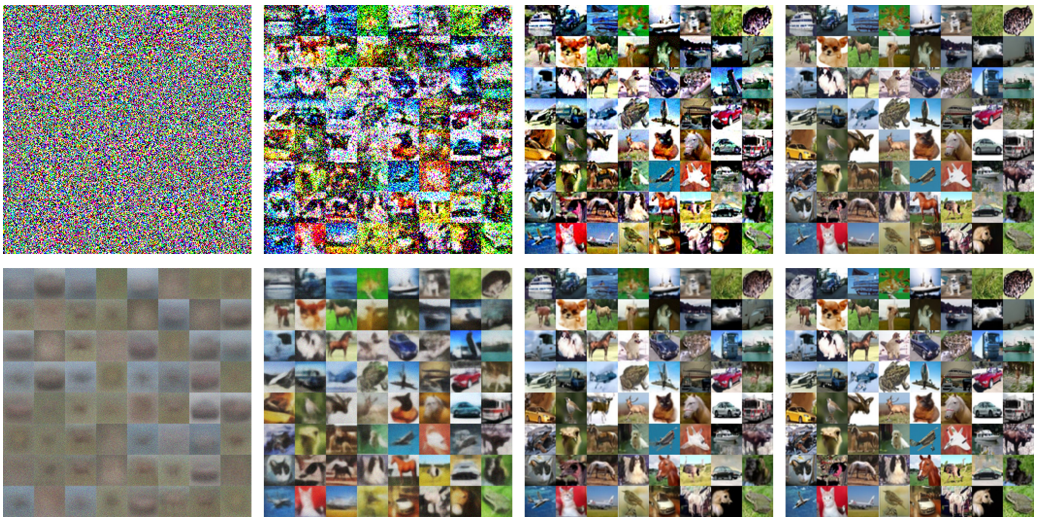
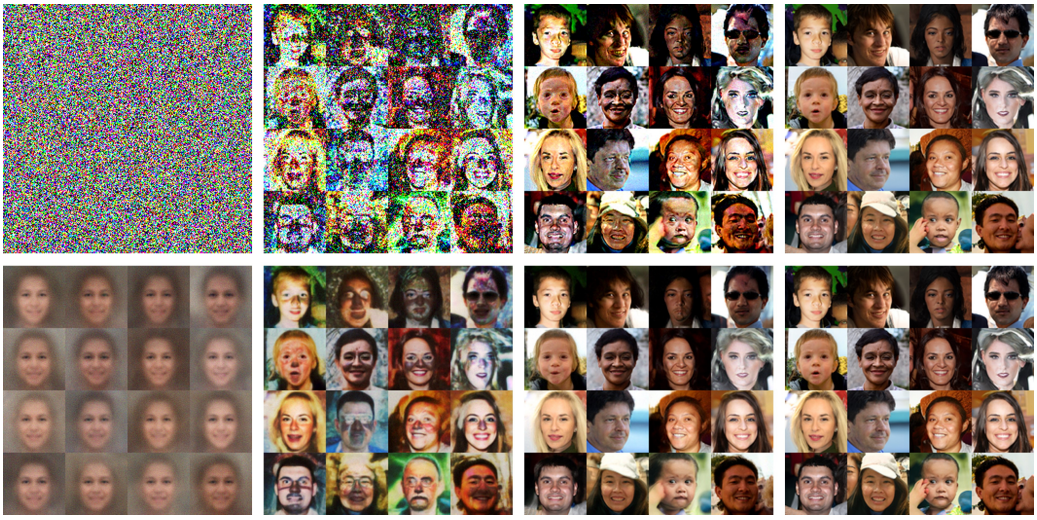


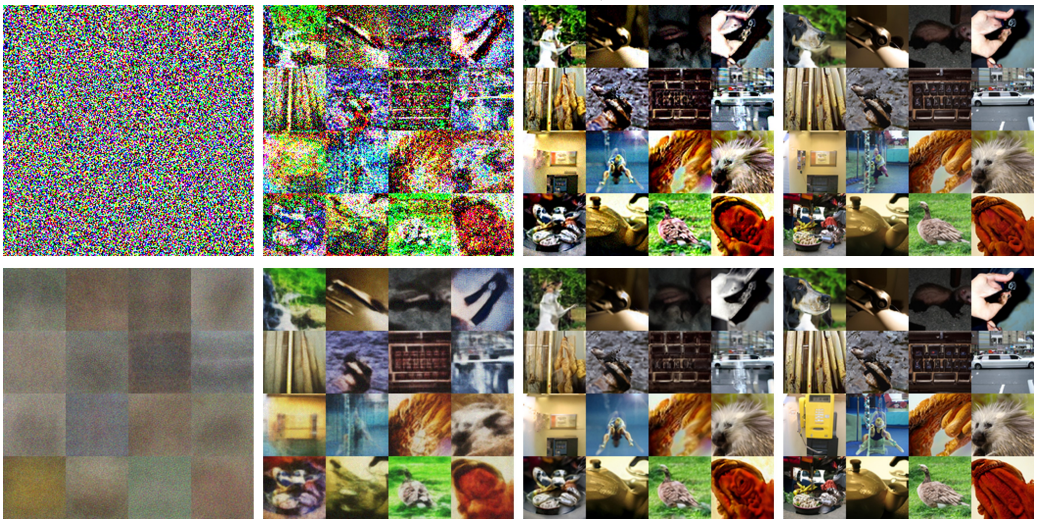
Figure 5. **Update path of randomly selected two pixels in the images.** The result of 1000-step DDIM is displayed as our target. These figures are drawn with 1000 samples using a noise prediction model trained on CIFAR-10.



(a) CIFAR-10 (32×32)

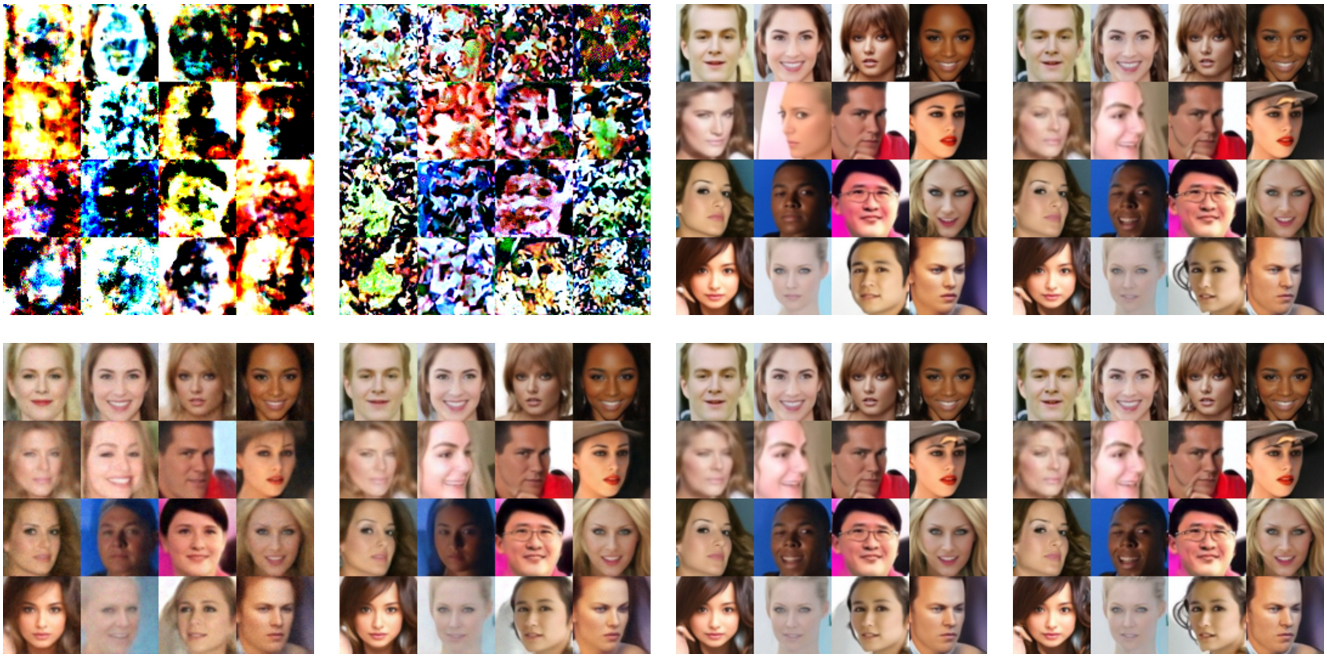


(b) FFHQ (64×64)



(c) ImageNet (64×64)

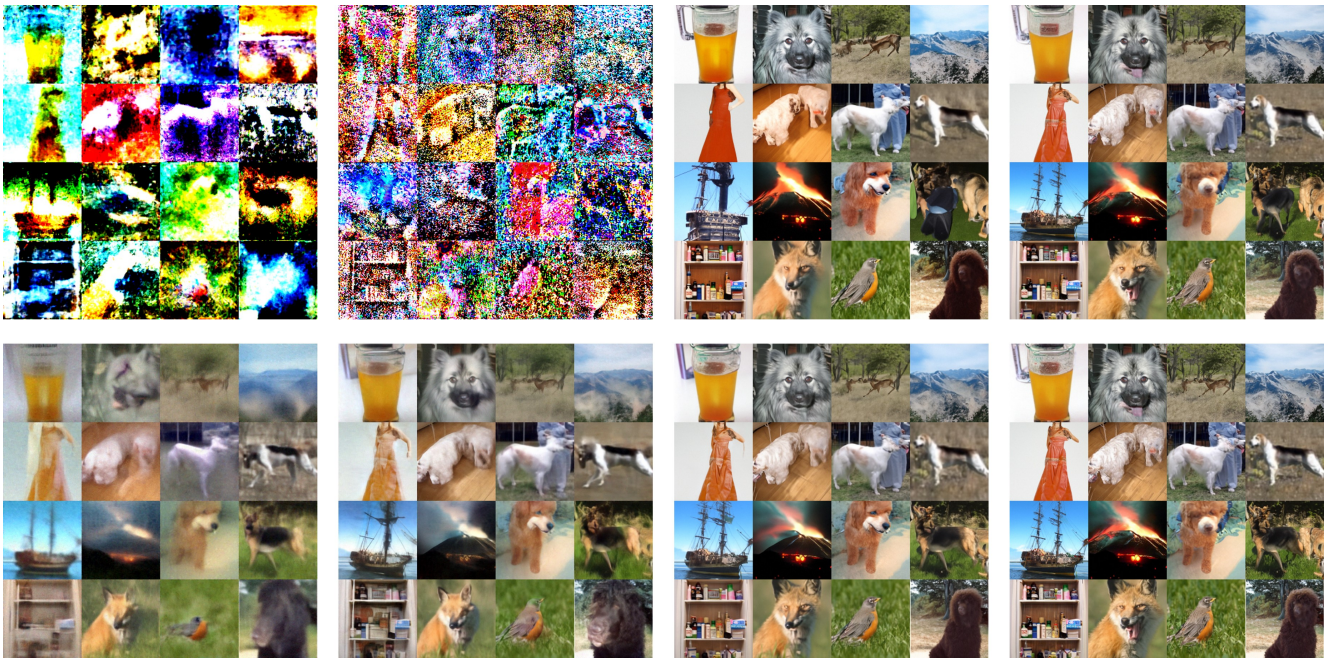
Figure 6. **Qualitative results of CIFAR-10** (32×32)**, FFHQ** (64×64)**, and ImageNet** (64×64) **with data prediction models.** We compare EDM (top) and D-EDM (bottom) in each subfigure with NFE $\in \{3, 5, 9, 25\}$.



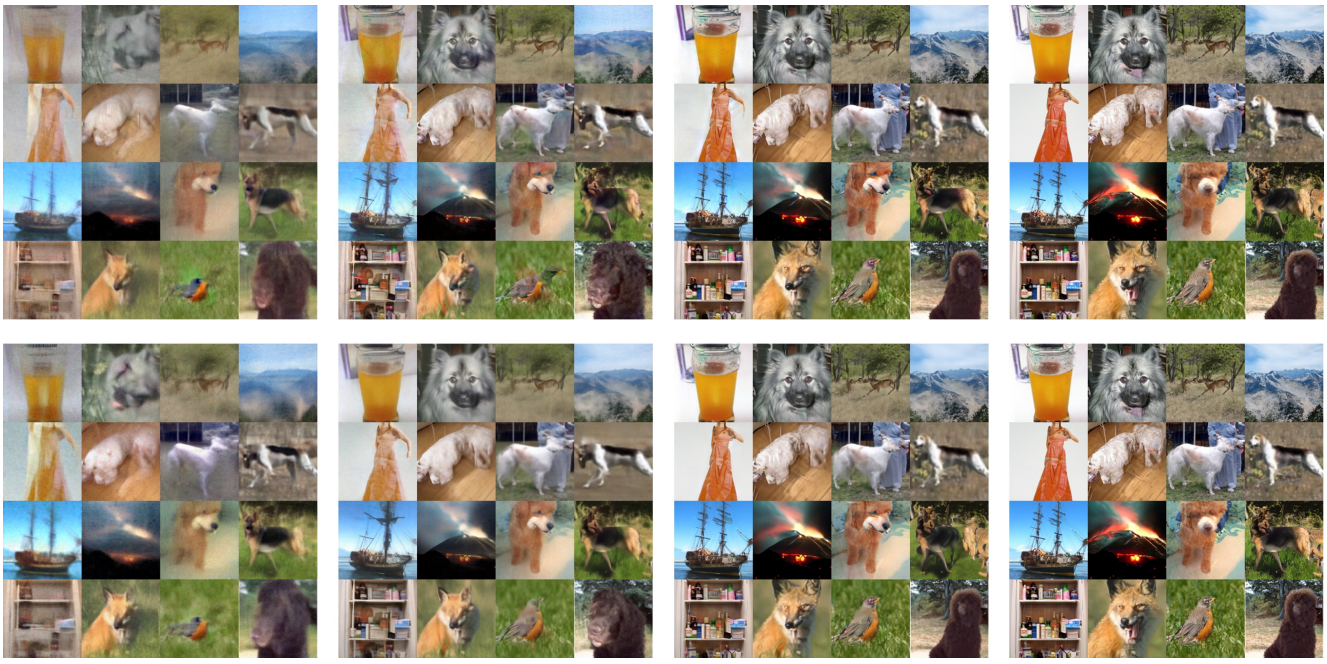
(a) DPM-Solver3 (top) vs D-DPM-Solver3 (bottom)

(b) DEIS3 (top) vs D-DEIS3 (bottom)

Figure 7. **Qualitative results of CelebA** (64×64) **with noise prediction models.** We compare ODE-solvers (DPM-Solver3, DEIS3) and D-ODE solvers (D-DPM-Solver3, D-DEIS3) in each subfigure with NFE $\in \{3, 5, 10, 25\}$.

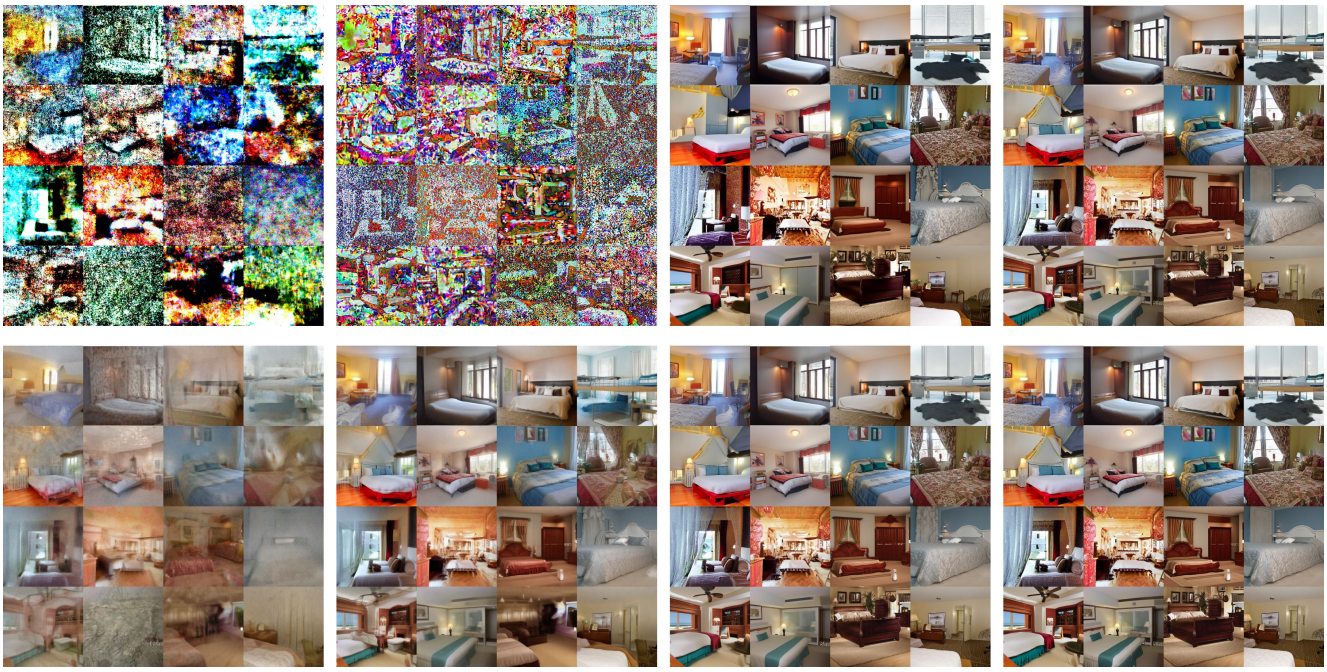


(a) DPM-Solver3 (top) vs D-DPM-Solver3 (bottom)



(b) DEIS3 (top) vs D-DEIS3 (bottom)

Figure 8. **Qualitative results of ImageNet** (128×128) **with noise prediction models.** We compare ODE-solvers (DPM-Solver3, DEIS3) and D-ODE solvers (D-DPM-Solver3, D-DEIS3) in each subfigure with NFE $\in \{3, 5, 10, 25\}$.



(a) DPM-Solver3 (top) vs D-DPM-Solver3 (bottom)

(b) DEIS3 (top) vs D-DEIS3 (bottom)

Figure 9. **Qualitative results of LSUN Bedroom** (256×256) **with noise prediction models.** We compare ODE-solvers (DPM-Solver3, DEIS3) and D-ODE solvers (D-DPM-Solver3, D-DEIS3) in each subfigure with NFE $\in \{3, 5, 10, 25\}$.

References

[1] Yaniv Benny and Lior Wolf. Dynamic dual-output diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11482–11491, 2022. 1, 2

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 1

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 1, 6

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2016. 1

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. 5

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1

[7] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations*, 2022. 1

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015. 2

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 2, 6

[10] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010. 3

[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, 4, 6

[12] DP Kingma. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2014. 5

[13] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 1

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[15] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2021. 3

[16] 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. 2, 3, 8

[17] 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. 3, 8

[18] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 2

[19] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. 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. 2

[21] 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. 6

[22] Frank Permenter and Chenyang Yuan. Interpreting and improving diffusion models using the euclidean distance function. *arXiv preprint arXiv:2306.04848*, 2023. 5

[23] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv e-prints*, pages arXiv–2204, 2022. 1

[24] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*, 2016. 6

[25] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 5

[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. 1

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021. 2, 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] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 4

[30] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 2, 7

[31] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 5

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

[33] 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

[34] Guoqiang Zhang, Kenta Niwa, and W Bastiaan Kleijn. Lookahead diffusion probabilistic models for refining mean estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1421–1429, 2023. 5, 7

[35] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2022. 3, 4, 6

[36] 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