This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Towards More Accurate Diffusion Model Acceleration with A Timestep Tuner

Mengfei Xia[1] Yujun Shen[2] Changsong Lei[1] Yu Zhou[1] Deli Zhao[3]
Ran Yi[4] Wenping Wang[5] Yong-Jin Liu[1*]
[1]Tsinghua University [2]Ant Group [3]Alibaba Group
[4]Shanghai Jiao Tong University [5]Texas A&M University

Abstract

A diffusion model, which is formulated to produce an image using thousands of denoising steps, usually suffers from a slow inference speed. Existing acceleration algorithms simplify the sampling by skipping most steps yet exhibit considerable performance degradation. By viewing the generation of diffusion models as a discretized integral process, we argue that the quality drop is partly caused by applying an inaccurate integral direction to a timestep interval. To rectify this issue, we propose a ***timestep tuner*** *that helps find a more accurate integral direction for a particular interval at the minimum cost. Specifically, at each denoising step, we replace the original parameterization by conditioning the network on a new timestep, enforcing the sampling distribution towards the real one. Extensive experiments show that our plug-in design can be trained efficiently and boost the inference performance of various state-of-the-art acceleration methods, especially when there are few denoising steps. For example, when using 10 denoising steps on LSUN Bedroom dataset, we improve the FID of DDIM from 9.65 to 6.07, simply by adopting our method for a more appropriate set of timesteps. Code is available at https://github.com/THU-LYJ-Lab/time-tuner.*

1. Introduction

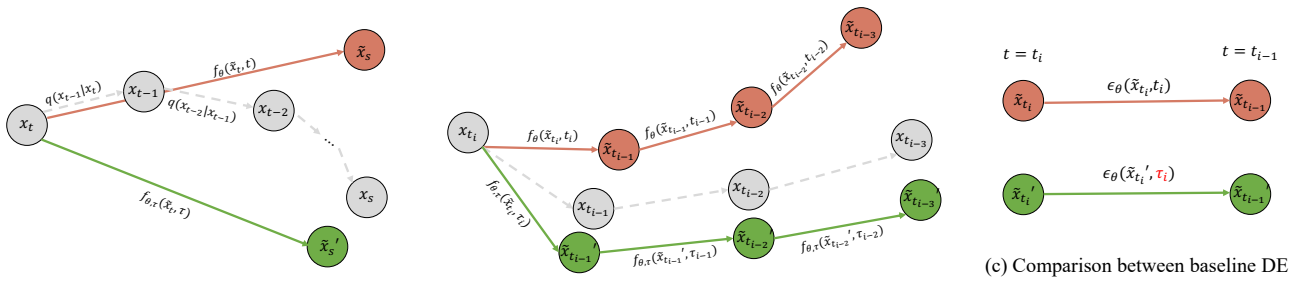
Diffusion probabilistic models (DPMs) [8, 37, 39], known simply as diffusion models, have recently received growing attention due to its efficacy of modeling complex data distributions [5, 8, 28]. A DPM first defines a forward diffusion process (*i.e.*, either discrete-time [8, 38] or continuous-time [39]) by gradually adding noise to data samples, and then learns the reverse denoising process with a timestep-conditioned parameterization. Consequently, it usually requires thousands of denoising steps to synthesize an image, which is time-consuming [8, 16, 38].

To accelerate the generation process of diffusion models, a common practice is to reduce the number of inference steps. For example, instead of a step-by-step evolution from the state of timestep 1,000 to the state of timestep 900, previous works [1, 8, 39] manage to directly link these two states with a one-time transition. That way, it only needs to evaluate the denoising network once instead of 100 times, thus substantially saving the computational cost.

A side effect of the above acceleration pipeline is the performance degradation that appears as artifacts in the synthesized images. In this paper, we aim to identify and address the cause of this side effect. As the gray dashed line shown in Fig. 1a, the generation process of diffusion models can be viewed as a discretized integrating process, where the direction of each integral step is calculated by the pre-learned noise prediction model. To reduce the number of steps, existing algorithms [1, 28, 38] typically apply the direction predicted for the initial state for the following timestep interval, as the red line shown in Fig. 1a, resulting in a gap between the sampling distribution and the real distribution. Karras [14] identifies this distribution gap as the *truncation error*, which accumulates over the whole steps intuitively and theoretically. As the red line shown in Fig. 1b, the gap between the sampling distribution and the real distribution increases as the integrating process evolves.

To alleviate the problem arising from skipping steps, we propose a timestep tuner, termed as `TimeTuner`, which targets at finding a more accurate integral direction for a particular interval. As the green line shown in Fig. 1a, our approach is designed to achieve this purpose efficiently by searching a more appropriate timestep τ replacing the previous t as the new condition input of the pre-learned noise prediction model. By doing so, we are able to significantly reduce the one-step truncation error, and hence the accumulative truncation error, as demonstrated with the green line in Fig. 1a and Fig. 1b respectively. Our motivation is intuitive – it is based on the observation that "*although the direction estimated from the initial state may not be appropriate for integration on the interval, the one*

*Corresponding author.



(a) One-step truncation error (b) Accumulative truncation error (c) Comparison between baseline DE solver and timestep-tuned DE solver

Figure 1. **Conceptual description** of (a) the one-step truncation error, (b) the accumulative truncation error, and (c) enforcing the sampling distribution towards the real distribution of our `TimeTuner` by replacing the input timestep from t to τ, by the full-step reverse process (**gray dashed line**), the baseline acceleration pipeline (**red line**), and our proposed method with timestep tuner (**green line**).

from some intermediate state can be", akin to the mean value theorem of integrals. To obtain this new timestep, we enforce the sampling distribution towards the real distribution by optimizing a specially designed loss function. We theoretically prove the feasibility of `TimeTuner`, and provide an estimation of the error bound for deterministic sampling algorithms. Experiments using different numbers of function evaluations (NFE) show that our `TimeTuner` can be used to boost the sampling quality of various acceleration methods without extra time cost, (*e.g.*, DDIM [38], Analytic-DPM [1], DPM-solver [25], *etc.*) in a plug-in fashion. Hence, our work offers a new perspective on accelerating the inference while simultaneously reducing the quality degradation of diffusion models.

2. Related Work

DPMs and the applications. Diffusion probabilistic model (DPM) is initially introduced by Sohl-Dickstein *et al.* [37], where the training is based on the optimization of the variational lower bound L_{vb}. Denoising diffusion probabilistic model (DDPM) [8] proposes a re-parameterization trick of DPM and learns the reweighted L_{vb}. Song *et al.* [39] model the forward process as a stochastic differential equation and introduce continuous timesteps. With rapid advances in recent studies, DPMs show great potential in various downstream applications, including speech synthesis [2, 17], video synthesis [10], super-resolution [20, 32], conditional generation [3], and image-to-image translation [33, 36].

Faster DPMs attempt to explore shorter trajectories rather than the complete reverse process, while ensuring the synthesis performance compared to the original DPM. Existing methods can be divided into two categories. The first category includes knowledge distillation [26, 34, 40]. Although such methods may achieve respectable performance with only one-step generation [40], they require expensive training stages before applied to efficient sampling, leading to poor applicability. The second category consists of training-free methods suitable for pre-trained DPMs. DDIM [38] is the first attempt to accelerate the sampling process using a probability flow ODE [39]. Some methods try to search for the trajectories by solving a least-cost-path problem with a dynamic programming (DP) algorithm or using the analytic solution [1, 43]. Inspired by this, seminal works conduct deeper research on trajectory choice [22, 42]. Another representative category of fast sampling methods use high-order differential equation (DE) solvers [11, 23, 25, 30, 41]. Saharia *et al.* [32] and Ho *et al.* [9] manage to train DPMs using continuous noise level and draw samples by applying a few-step discrete reverse process. Some GAN-based methods also consider larger sampling step sizes, e.g., in [44] a multi-modal distribution is learned in a conditional GAN with a large step size. However, to the best of our knowledge, existing training-free acceleration algorithms are bottlenecked by the poor sampling performance with extremely few inference steps (*e.g.*, less than 5 steps). Our `TimeTuner` can be considered as a performance booster for existing training-free acceleration methods, *i.e.*, it further improves the generation performance.

3. Method

3.1. Background

Suppose that $\mathbf{x}_0 \in \mathbb{R}^D$ is a D-dimensional random variable with an unknown distribution $q_0(\mathbf{x}_0)$. DPMs [8, 37, 39] define a forward process $\{\mathbf{x}_t\}_{t\in(0,T]}$ by gradually adding noise on $\mathbf{x}_0$ with $T > 0$, such that for any $t \in (0, T]$, we have the transition distribution:

$$q_{0t}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t\mathbf{x}_0, \sigma_t^2\mathbf{I}), \tag{1}$$

where $\alpha_t, \sigma_t \in \mathbb{R}^+$ are differentiable functions of t with bounded derivatives. The choice of α_t, σ_t is referred to as the *noise schedule*. Let $q_t(\mathbf{x}_t)$ be the marginal distribution of $\mathbf{x}_t$, DPM ensures that $q_T(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \sigma^2\mathbf{I})$ for some $\sigma > 0$, and the signal-to-noise-ratio (SNR) α_t^2/σ_t^2 is strictly decreasing with respect to timestep t [15].

Seminal works [15, 39] studied the underlying stochastic differential equation (SDE) theory of DPMs. The forward

and reverse processes are as below for any $t \in (0, T]$:

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t \mathrm{d}t + g(t)\mathrm{d}\mathbf{w}_t, \quad \mathbf{x}_0 \sim q_0(\mathbf{x}_0), \tag{2}$$

$$\mathrm{d}\mathbf{x}_t = [f(t)\mathbf{x}_t - g^2(t)\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}_t, \tag{3}$$

where $\mathbf{w}_t, \bar{\mathbf{w}}_t$ are standard Wiener processes in forward and reverse time, respectively, and f, g have closed-form expressions w.r.t α_t, σ_t. The unknown $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ is referred to as the *score function*. Furthermore, Song *et al.* [38] propose the *probability flow ODE* for faster and more stable sampling, which enjoys the identical marginal distribution at each t as that of the SDE in Eq. (3), *i.e.*,

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = f(t)\mathbf{x}_t - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t). \tag{4}$$

Technically, DPMs implement sampling by solving DEs with numerical solvers discretizing the DE from T to 0. To this end, DPMs introduce a neural network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, namely the noise prediction model, to approximate the score function from the given $\mathbf{x}_t$, where the parameter θ can be optimized by the objective below:

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t}\left[\omega_t \|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \boldsymbol{\epsilon}\|_2^2\right], \tag{5}$$

where ω_t is the weighting function, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, and $t \sim \mathcal{U}[0, T]$.

3.2. Gap between Real and Sampling Distributions

Recall that a DPM sampler is built on the noise prediction model $\boldsymbol{\epsilon}_\theta$ at each timestep t, which is a discretization of an integrating process. At each denoising step, one applies $\boldsymbol{\epsilon}_\theta$ on the intermediate result $\widetilde{\mathbf{x}}_t$ from the last denoising step together with its corresponding timestep condition t, *i.e.*, $\widetilde{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_t, t)$. The predicted noise $\widetilde{\boldsymbol{\epsilon}}_t$ will be used as the integral direction towards the next denoised result.

However, recall that during the training of DPM, the noise prediction model $\boldsymbol{\epsilon}_\theta$ is trained with the noisy data at timestep t, *i.e.*, $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Intuitively, due to the error of the DE solver (*i.e.*, SDE or ODE solver) using large discretization step sizes, there is a considerable gap between the real distribution of $\mathbf{x}_t$ and the sampling distribution of $\widetilde{\mathbf{x}}_t$ at each timestep t, which is called the truncation error [14]. Even worse, this error is accumulated progressively during the reverse process, since the gap between the distributions of $\mathbf{x}_t$ and $\widetilde{\mathbf{x}}_t$ leads to a poor prediction of $\widetilde{\mathbf{x}}_{t-1}$, in which the truncation error at timestep t transmits to the next timestep $t-1$ [14].

To support this insight, Fig. 1a shows the truncation error at each timestep, in comparison with the original full-step sampling process. As the gray dashed line in Fig. 1a indicates, in the original full-step sampling process, the denoised $\mathbf{x}_s$ comes from a step-by-step denoising refinement starting at $\mathbf{x}_t$, by the results from $\boldsymbol{\epsilon}_\theta$ at all intermediate timesteps from t down to s. However, the accelerated sampling method (*i.e.*, red line) uses a large step size and replaces all intermediate predicted noises with one single predicted noise in the initial timestep, *i.e.*, $\widetilde{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_t, t)$. Therefore, each $\widetilde{\mathbf{x}}_t$ incurs a truncation error consequently.

Furthermore, note that for a sampling process, given $\widetilde{\mathbf{x}}_T$ at timestep T and a timestep $t \in [0, T)$, the DE solver approximates the true $\mathbf{x}_t$ as $\widetilde{\mathbf{x}}_t$. As the red line indicates in Fig. 1b, since the local truncation error accumulates at each step, the gap between the sampling distribution of $\widetilde{\mathbf{x}}_t$ and the real distribution of $\mathbf{x}_t$ increases as t evolves. To gain the insight of the accumulative error, we conduct a simple experiment to provide convincing evidence of the above observation using DDIM [38] and DPM-Solver-2 [25] on CIFAR10 dataset [18]. We first sample $\widetilde{\mathbf{x}}_T$ and estimate $\mathbf{x}_t$ sequences using DDIM sampler with NFE $= 1,000$. Meanwhile, we draw the approximate $\widetilde{\mathbf{x}}_t$ sequences using DDIM and DPM-Solver-2 under quadratic, uniform, and log-SNR trajectories with NFE $= 10$ or 20, respectively. Then we calculate the L_2 metric of $\mathbf{x}_t - \widetilde{\mathbf{x}}_t$ at each timestep t, in order to demonstrate the gap between the two distributions, the result is shown in Fig. 2. It is noteworthy that: (1) the gap between $\mathbf{x}_t$ and $\widetilde{\mathbf{x}}_t$ indeed accumulates as timestep t evolves from T to 0, making the sampling distribution farther and farther away from the real one, which severely hurts the final sampling quality; (2) with the same quadratic trajectory, the larger the NFE is, the smaller the gap between the two distributions is; (3) different types of trajectories and DPM samplers account for different behaviors of the accumulative truncation error.

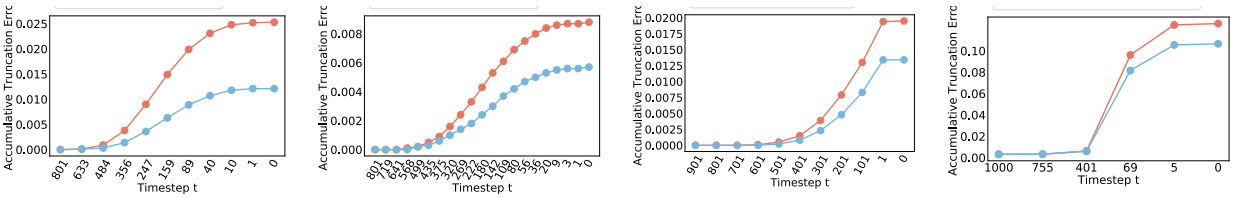
3.3. Theoretical Foundations of TimeTuner

Recall that in Sec. 3.2 we demonstrate the gap between the real and the sampling distributions. This gap will damage the quality of the synthesis samples. In this section, we will delve into the theory of reverse denoising process in DPMs, and give an upper bound of the distribution gap for the deterministic sampler, which derives the feasibility of the `TimeTuner`. Before stating the main theorem, we first propose several definitions for simplicity. For a discretization $0 = t_0 < t_1 < \cdots < t_K = T$ of $[0, T]$ and a DE solver denoted by $f_\theta(\mathbf{x}_{t_i}, t_i)$, which is responsible to denoise the intermediate $\widetilde{\mathbf{x}}_{t_i}$ for one single step, *i.e.*, $\widetilde{\mathbf{x}}_{t_{i-1}} = f_\theta(\widetilde{\mathbf{x}}_{t_i}, t_i)$ for $i = 1, 2, \cdots, K$. For instance, the DE solver f_θ of DDIM [38] is of the following form:

$$f_\theta(\widetilde{\mathbf{x}}_{t_i}, t_i) = \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}\widetilde{\mathbf{x}}_{t_i} - \left(\frac{\alpha_{t_{i-1}}\sigma_{t_i}}{\alpha_{t_i}} - \sigma_{t_{i-1}}\right)\boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_{t_i}, t_i). \tag{6}$$

One can generalize the definition of $f_\theta(\mathbf{x}_{t_i}, t_i)$ as below:

$$f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i}, \tau_i) := \frac{\alpha_{t_{i-1}}}{\alpha_{t_i}}\widetilde{\mathbf{x}}_{t_i} - \left(\frac{\alpha_{t_{i-1}}\sigma_{t_i}}{\alpha_{t_i}} - \sigma_{t_{i-1}}\right)\boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_{t_i}, \tau_i), \tag{7}$$



(a) DDIM sampler, NFE = 10 quadratic trajectory (b) DDIM sampler, NFE = 20 quadratic trajectory (c) DDIM sampler, NFE = 10 uniform trajectory (d) DPM-Solver-2 sampler, NFE = 10 log-SNR trajectory

Figure 2. **Quantitative measurement** of the gap between real and sampling distribution using DDIM and DPM-Solver-2. The horizontal axis represents timesteps forming (a) quadratic trajectory with NFE $= 10$; (b) quadratic trajectory with NFE $= 20$; (c) uniform trajectory with NFE $= 10$; (d) log-SNR trajectory with NFE $= 10$. We plot the L_2 distance between $(\mathbf{x}_t, \widetilde{\mathbf{x}}_t)$ for the original and the timestep-tuned sampler, shown in red and blue, respectively. We also provide an error bound for deterministic sampler theoretically in Theorem 1.

where τ_i replaces the previous input condition timestep t_i for each $i = 1, 2, \cdots, K$. Now we state the theorem below. Proof is available in *Supplementary Material*.

Theorem 1 *Assume that $\boldsymbol{\epsilon}_\theta$ is the ground-truth noise prediction model, with $\|\boldsymbol{\epsilon}_\theta(\mathbf{x}, t) - \boldsymbol{\epsilon}_\theta(\mathbf{y}, t)\|_2 \geqslant \frac{1}{C}\|\mathbf{x} - \mathbf{y}\|_2$ for any t and some $C > 0$. Denote by $\mathbf{x}_{t_i}^{gt}$ the ground-truth intermediate result at t_i starting from $\widetilde{\mathbf{x}}_{t_K}$, and by $f_{\theta,\tau}$ a deterministic sampler. We have the following inequality:*

$$\begin{aligned}&\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}[\|\widetilde{\mathbf{x}}_{t_{i-1}} - \mathbf{x}_{t_{i-1}}^{gt}\|_2] \\ \leqslant & C\Big(\sum_{n=i}^{K}\left(\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon}_\theta(f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i},\tau_i),t_{i-1}) - \boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_{t_i},t_i)\|_2^2\right]\right)^{\frac{1}{2}} \\ &+ \sum_{l=i}^{K}\mathbb{E}[\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_l}^{gt},t_l) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_{l-1}}^{gt},t_{l-1})\|_2]\Big). \end{aligned} \tag{8}$$

In conclusion, Theorem 1 studies the relation between the distribution gap and the generalized DE sampler $f_{\theta,\tau}$. Intuitively, this is based on the observation that some intermediate state could be more appropriate for the integration on the interval than the initial one, akin to the mean value theorem of integrals. Therefore, what we need to do to boost any given acceleration algorithm, is to choose an adequate timestep τ, *replacing the input for $\boldsymbol{\epsilon}_\theta$ from t to τ* (which is shown in Fig. 1c), such that the sampling distribution of $\widetilde{\mathbf{x}}_t$ tends to obey $q_t(\mathbf{x}_t)$ by shrinking the upper bound in Eq. (8). This profound conclusion facilitates DPM acceleration from a novel perspective.

3.4. Timestep Tuner for Noise Prediction Model

Now we formally propose `TimeTuner`, targeting more accurate DPM acceleration. Recall that in Sec. 3.3 we give an analysis on distribution gap, which is bounded by a term containing generalized DE solver $f_{\theta,\tau}$ with replaced timesteps. This motivates us to bridge the distribution gap with more proper input condition timesteps.

First we provide a picture of our formulation, as shown in Fig. 1. To reduce the truncation error caused by inaccurate integral direction and large discretization step size (*i.e.*, the denoised result $\widetilde{\mathbf{x}}_s$ achieved along the red line in Fig. 1a), we target at finding a more accurate integral direction (*i.e.*, by replacing t_i with optimized τ_i in Fig. 1c) for the interval from t to s (*i.e.*, the denoised result $\widetilde{\mathbf{x}}'_s$ achieved along the green line in Fig. 1a). In this sense, we are able to achieve a better integral direction for each interval (*i.e.*, the green line in Fig. 1b), mitigating the accumulation of the truncation error (*i.e.*, the red line in Fig. 1b). Formally, for a discretization $0 = t_0 < t_1 < \cdots < t_K = T$ and a DE solver f_θ, `TimeTuner` will employ the generalized DE solver $f_{\theta,\tau}$ to enforce $\widetilde{\mathbf{x}}_{t_i}$ towards $q_{t_i}(\mathbf{x}_{t_i})$, where τ_i is the optimized timestep replacing the previous input condition timestep t_i for each $i = 1, 2, \cdots, K$.

Algorithm 1 Training τ_i with sequential strategy

repeat
  $\mathbf{x}_0 \sim q_0(\mathbf{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\widetilde{\mathbf{x}}_T \leftarrow \alpha_T \mathbf{x}_0 + \sigma_T \boldsymbol{\epsilon}$
  $\widetilde{\mathbf{x}}_{t_i} \leftarrow f_{\theta,\tau}(\cdots(f_{\theta,\tau}(\widetilde{\mathbf{x}}_T, \tau_K)\cdots), \tau_{i+1})$
  Take gradient descent step on

$$\nabla_{\tau_i}(\|\boldsymbol{\epsilon}_\theta(f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i},\tau_i),t_{i-1}) - \boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_{t_i},t_i)\|_2^2)$$

until converged

Training `TimeTuner` is simple and efficient. Recall that we hope to find a timestep τ_i replacing the previous t_i, such that the sampling distribution of $\widetilde{\mathbf{x}}_{t_{i-1}} = f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i}, \tau_i)$ tends to follow the real distribution $q_{t_{i-1}}(\mathbf{x}_{t_{i-1}})$ for each $i = 1, 2, \cdots, K$. Therefore, after determining the number of function evaluations (NFE) K and its corresponding trajectory $0 = t_0 < t_1 < \cdots < t_K = T$, the loss function of τ_i is defined as below for $i = K, K-1, \cdots, 1$:

$$\begin{aligned}&\mathcal{L}_i(\tau_i) \\ =&\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon}_\theta(f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i},\tau_i),t_{i-1}) - \boldsymbol{\epsilon}_\theta(\widetilde{\mathbf{x}}_{t_i},t_i)\|_2^2\right]. \end{aligned} \tag{9}$$

Then according to Theorem 1, optimizing $\mathcal{L}_i$ will narrow the gap between sampling and real distributions. Therefore, we can optimize the timesteps reversely from $t_K = T$ down to t_1, in which $\widetilde{\mathbf{x}}_{t_K} = \widetilde{\mathbf{x}}_T = \alpha_T \mathbf{x}_0 + \sigma_T \boldsymbol{\epsilon} \approx \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ for given $\mathbf{x}_0 \sim q_0(\mathbf{x}_0)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The training process is summarized in Algorithm 1. It is noteworthy that

the training of our method is in a plug-in fashion. There is no need to modify the parameters of the pre-trained DPMs. Besides, since it only requires optimization of a scalar τ_i, the training is extremely efficient (*e.g.*, `TimeTuner` takes ~ 1 hour in all for NFE $= 10$ on an NVIDIA A100 GPU).

We then exhibit the relation between the objective of our method (*i.e.*, Eq. (9)) and that of the DDPM (*i.e.*, Eq. (5)), concluded as the theorem below. Proof and detailed quantitative comparison are addressed in *Supplementary Material*.

Theorem 2 *Assume that ϵ_θ is the ground-truth noise prediction model. The loss function of `TimeTuner` resembles that of the original DPM, i.e., for $i = K, K-1, \cdots, 1$, the optimal τ_i holds the following property:*

$$\begin{aligned}&\arg\min_{\tau_i} \mathcal{L}_i(\tau_i)\\ =&\arg\min_{\tau_i} \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon}_\theta(f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i},\tau_i),t_{i-1})\\ &\qquad - \frac{\widetilde{\mathbf{x}}_{t_i}-\alpha_{t_i}\mathbf{x}_0}{\sigma_{t_i}}\|_2^2]. \end{aligned} \tag{10}$$

3.5. Parallel Training Strategy of TimeTuner

Recall that we employ a *sequential strategy* to train each τ_i for i from K to 1, *i.e.*, each τ_i needs to be optimized sequentially. Also, one can train all τ_i's simultaneously with Eq. (11) by a *parallel strategy* for $i = 1, 2, \cdots, K$:

$$\begin{aligned}&\mathcal{L}_i^{parallel}(\tau_i)\\ =&\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon}_\theta(f_{\theta,\tau}(\mathbf{x}_{t_i},\tau_i),t_{i-1})-\boldsymbol{\epsilon}_\theta(\mathbf{x}_{t_i},t_i)\|_2^2\right], \end{aligned} \tag{11}$$

where $\mathbf{x}_{t_i} = \alpha_{t_i}\mathbf{x}_0 + \sigma_{t_i}\boldsymbol{\epsilon}$ instead of the intermediate result $\widetilde{\mathbf{x}}_{t_i}$ by $f_{\theta,\tau}$. The parallel strategy is feasible since $\mathbf{x}_{t_i}$ is independent with all τ_i's, and one can train τ_i simultaneously by sampling different $\mathbf{x}_{t_i}$ on different GPUs. Despite the extra acceleration of the optimization process, the parallel training strategy will subtly harm the sampling performance, which can be observed from Tab. 5 in Sec. 4.3. This is because the achieved sub-optimal $\tau_K, \cdots, \tau_{i+1}$ fail to ensure that the sampling distribution of $\widetilde{\mathbf{x}}_{t_i}$ matches $q_{t_i}(\mathbf{x}_{t_i})$. Then $f_{\theta,\tau}(\widetilde{\mathbf{x}}_{t_i},\tau_i)$ and $f_{\theta,\tau}(\mathbf{x}_{t_i},\tau_i)$ have different distributions. Therefore, one can only use biased $\widetilde{\mathbf{x}}_{t_i}$ for subsequent training of τ_i. In other words, the parallel training will introduce extra error at each denoising step.

4. Experiments

In this section, we show that `TimeTuner` can greatly improve the performance of existing DPM acceleration algorithms, varying different NFEs of calling ϵ_θ. We show great improvements of sampling quality in Sec. 4.2, and two different training strategies in Sec. 4.3.

4.1. Experimental Setups

Datasets and baselines. We apply `TimeTuner` to existing acceleration methods, including DDIM [38], DDPM [8], Analytic-DDIM [1], Analytic-DDPM [1], and DPM-Solver-2 [25]. We also apply `TimeTuner` on EDM [14], a high-order DE solver with specially designed noise schedule. For high-resolution DPMs, we involve latent diffusion models (LDMs) [31]. To confirm the efficacy under extreme NFEs, we apply `TimeTuner` on Consistent Distillation (CD) [40]. The pre-trained DPMs are trained on CIFAR10 [18], CelebA [24], LSUN Bedroom [45], FFHQ [13], CelebA-HQ [12], ImageNet [4], and MS-COCO [21], respectively. Note that EDM and LDM on ImageNet and MS-COCO are conditional generation, based on label and text, respectively. DPMs on all seven datasets are under linear schedule [8] with $T = 1,000$.

Evaluation metrics. We draw 50,000 samples and use Fréchet Inception Distance (FID) [6] to evaluate the fidelity of the synthesized images. Inception Score (IS) [35] measures how well a model captures the full ImageNet class distribution while still producing individual samples of a single class convincingly. To better measure spatial relationships, we involve sFID [27], rewarding image distributions with coherent high-level structure. Finally, we use Improved Precision and Recall [19] to separately measure sample fidelity (precision) and diversity (recall).

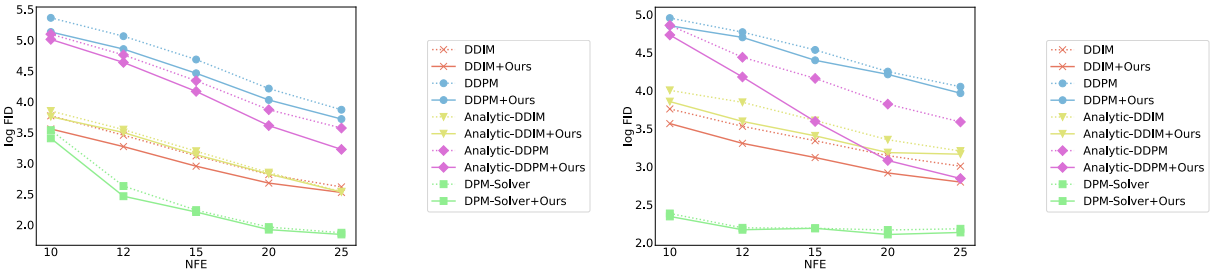
Implementation details. We train `TimeTuner` on PyTorch [29] with NVIDIA A100 GPUs. We use the pre-trained DDIM[1] of CelebA provided in the official implementation, while that on CIFAR10 is trained by Bao *et al.* [1] with the same U-Net structure as Nichol & Dhariwal [28]. For EDM[2], LDM[3], and CD[4], we directly use the pre-trained model in the official implementations.

4.2. Sample Quality

Unconditional generation on CIFAR10 and CelebA. For the strongest baseline, we apply the quadratic trajectory for DDIM and DDPM on both CIFAR10 and CelebA datasets, which empirically achieves better FID performance than uniform trajectory. As for DPM-Solver-2, we use the log-SNR trajectory following the original setup [25]. As shown in Fig. 3, under all trajectories of different NFEs, our proposed `TimeTuner` consistently improves the sampling performance of DDIM, DDPM, Analytic-DDIM, Analytic-DDPM, and DPM-Solver-2 significantly.

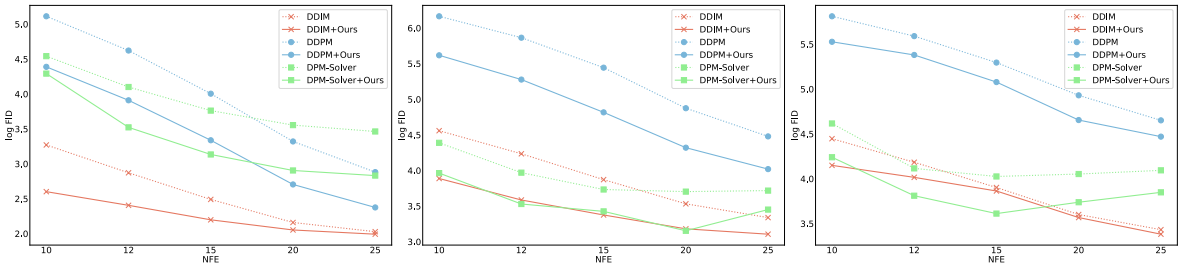
Unconditional generation on LDM. As for the datasets with high resolution 256x256, we apply LDM to guarantee the training efficiency of our algorithm without loss of the synthesis performance. From Fig. 4 one can conclude that our algorithm achieves even better performance improvement on the high-resolution datasets. To quantitatively demonstrate the improvement, we make further comparison

[1] https://github.com/ermongroup/ddim
[2] https://github.com/NVlabs/edm
[3] https://github.com/CompVis/latent-diffusion
[4] https://github.com/openai/consistency_models



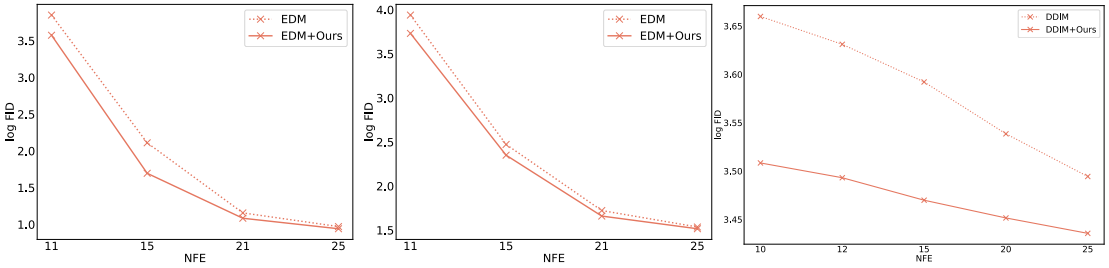
(a) CIFAR10 (b) CelebA 64x64

Figure 3. **Quantitative comparison** measured by log FID ↓ on CIFAR10 and CelebA, under original DDPM. All are evaluated with different NFEs on the horizontal axis. We apply quadratic trajectory for DDIM and DDPM, uniform trajectory for Analytic-DDIM and Analytic-DDPM, log-SNR trajectory for DPM-Solver-2.



(a) LSUN Bedroom 256x256 (b) FFHQ 256x256 (c) CelebA-HQ 256x256

Figure 4. **Quantitative comparison** measured by log FID ↓ on LSUN Bedroom, FFHQ, and CelebA-HQ, under LDM. All are evaluated with different NFEs on the horizontal axis. We apply uniform trajectory for DDIM and DDPM, and log-SNR trajectory for DPM-Solver-2.



(a) CIFAR10 32x32 (label-conditioned) (b) ImageNet 64x64 (label-conditioned) (c) MS-COCO 256x256 (text-conditioned)

Figure 5. **Quantitative comparison** measured by log FID ↓ on CIFAR10, ImageNet, and MS-COCO, under EDM and LDM. All are evaluated with different NFEs on the horizontal axis. We apply the originally designed trajectory for EDM and linear trajectory for LDM.

using more metrics, as shown in Tab. 1. Qualitative results can be found in Fig. 7 for clear demonstration.

High-order sampler generation using EDM. As for high-order DE solver, we involve EDM which introduces the 2nd Heun sampling method on label-conditioned generation. As demonstrated in Fig. 5, our method improves the generation performance as well, indicating the great compatibility for conditional generation under high-order DE solvers.

Label-conditioned generation on ImageNet. As shown in Tab. 1 and Fig. 7, the synthesis performance of our algorithm surpasses the baseline qualitatively and quantitatively.

Text-conditioned generation on MS-COCO. For the most challenging text-conditioned generation, qualitative and quantitative results demonstrated in Figs. 5 and 7 and Tab. 1 confirm the compatibility and capability of our method.

Generation under extreme NFEs. Despite significant improvements on the state-of-the-art training-free acceleration methods, we also confirm the efficacy of `TimeTuner` on CD. As demonstrated in Tab. 2, our method is capable of improving performance of CD under extremely small NFEs. More comparison is addressed in *Supplementary Material*.

Performance under different CFG scales. Classifier-Free Guidance (CFG) [7] is a commonly used technique to facilitate fidelity of conditional generation. As in Tab. 3, `TimeTuner` is capable of consistently improving the performance on label- and text-conditioned tasks with different CFG scales. We will address the optimized timesteps which vary for different scales in *Supplementary Material*.

Table 1. **Quantitative comparison** measured by IS ↑, FID ↓, sFID ↓, Precision ↑ and Recall ↑ on LSUN Bedroom, FFHQ, CelebA-HQ, and ImageNet. All are evaluated by drawing 50,000 samples via DDIM upon LDM, with NFE = 10.

LSUN Bedroom 256x256, *unconditional* generation					
Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
DDIM	2.30	9.46	12.02	0.55	0.34
DDIM + Ours	**2.31**	**5.85**	**9.44**	**0.57**	**0.44**
FFHQ 256x256, *unconditional* generation					
Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
DDIM	4.00	23.58	14.59	0.63	0.21
DDIM + Ours	**4.40**	**14.80**	**9.69**	**0.67**	**0.32**
CelebA-HQ 256x256, *unconditional* generation					
Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
DDIM	2.95	18.72	16.68	**0.68**	0.19
DDIM + Ours	**3.20**	**16.59**	**15.61**	0.67	**0.26**
ImageNet 256x256, *conditional* generation					
Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
DDIM	324.52	10.13	12.52	0.91	0.28
DDIM + Ours	**336.94**	**9.63**	**7.29**	**0.92**	**0.30**

Table 2. **Quantitative comparison** measured by FID ↓, Precision ↑, and Recall ↑ on LSUN Bedroom by drawing 50,000 samples via CD.

Method	FID ↓	Precision ↑	Recall ↑	Method	FID ↓	Precision ↑	Recall ↑
CD (NFE = 1)	8.14	**0.68**	0.32	CD (NFE = 2)	5.91	**0.69**	0.38
CD + Ours (NFE = 1)	**7.14**	0.65	**0.35**	CD + Ours (NFE = 2)	**5.18**	0.68	**0.39**

Table 3. **Quantitative comparison** measured by FID ↓, Precision ↑, and Recall ↑ on ImageNet and MS-COCO under LDM. All are evaluated via DDIM with 10 NFEs, using different CFG scales. For clearer demonstration, improvements are highlighted in **green**.

DDIM, NFE = 10	ImageNet 256x256			MS-COCO 256x256
	FID ↓	Precision ↑	Recall ↑	FID ↓
CFG scale = 3	10.13 (-0.50)	0.91 (+0.01)	0.28 (+0.02)	11.00 (-0.47)
CFG scale = 5	16.78 (-0.39)	0.94 (0.00)	0.16 (0.00)	12.64 (-1.26)
CFG scale = 7	20.32 (-0.80)	0.94 (+0.04)	0.11 (0.00)	14.80 (-0.64)

Table 4. **Quantitative comparison** measured by FID ↓ on CIFAR10. All are evaluated by drawing 50,000 samples via DDIM sampler.

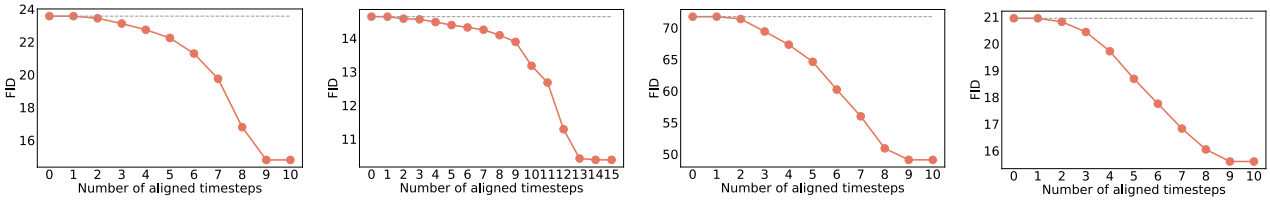
Method	NFE = 16	NFE = 10, linear trajectory			NFE = 10, quadratic trajectory		
	GGDM [43]	DDIM	RS-DDIM [42]	Ours	DDIM	OMS-DPM [22]	Ours
FID ↓	> 40	16.68	16.12	**15.76**	13.65	12.34	**11.77**

Table 5. **Quantitative comparison** measured by FID ↓ between *sequential* and *parallel* training strategies, where all are evaluated by drawing 50,000 samples via DDIM. We conduct experiments on CIFAR10, CelebA, LSUN Bedroom, FFHQ, and CelebA-HQ. We apply quadratic trajectory for CIFAR10 and CelebA datasets, and uniform trajectory on the other three datasets.

Dataset	CIFAR10	CelebA	LSUN Bedroom	FFHQ	CelebA-HQ
DDIM	13.65	13.53	9.65	23.57	21.81
DDIM + Ours (sequential)	11.77	**11.84**	**6.07**	**14.80**	**17.75**
DDIM + Ours (parallel)	**11.75**	12.14	6.08	15.03	18.03

Comparison with other alternatives. From Tab. 4 one can conclude that our method shows superior performance. Besides, we would like to reaffirm that our method does not target finding an optimal schedule. Instead, it works as a plug-in, bringing further improvements over baselines.

It is noteworthy that the performance improvement is more significant with a small NFE. For instance, the improvement of FID between DDIM and DDIM + Ours on FFHQ decreases from 8.77 to 1.51 as NFE grows from 10 to 25. This roots in the fact that larger NFE relieves the



(a)DDIM sampler, NFE = 10 uniform trajectory (b) DDIM sampler, NFE = 15 uniform trajectory (c) DDPM sampler, NFE = 10 uniform trajectory (d) DPM-Solver-2 sampler, NFE = 10 log-SNR trajectory

Figure 6. **Quantitative measurement** of FID drop by DDIM, DDPM, and DPM-Solver under different trajectories. The horizontal axis reports the number of replaced timesteps from t_K to t_1. The **red line** shows the FID drop, while the **gray dashed line** shows baseline FID.

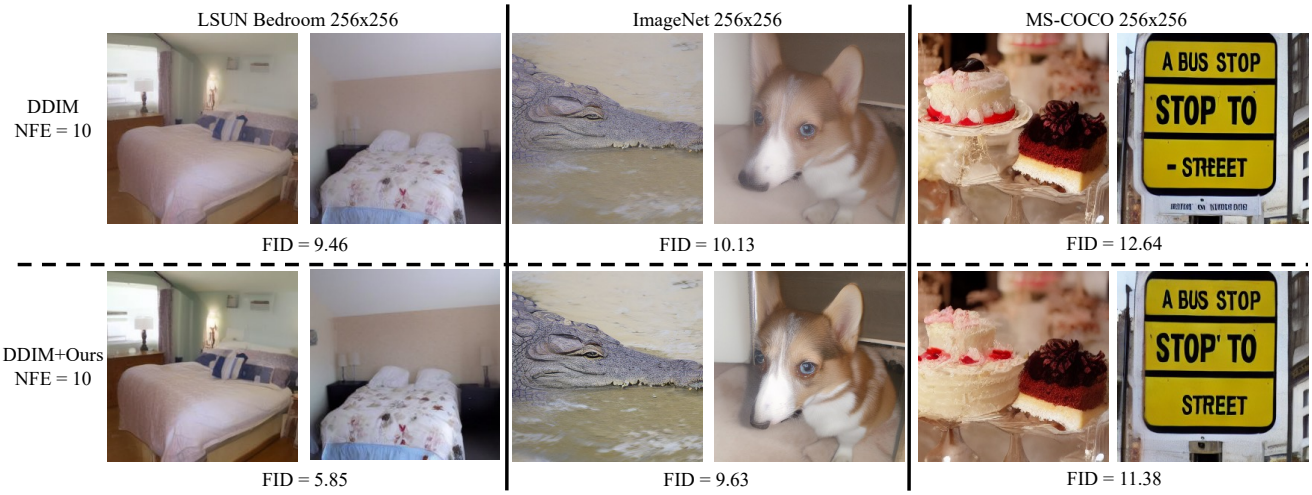


Figure 7. **Qualitative comparison** on LSUN Bedroom, ImageNet, and MS-COCO, under LDM. The three tasks are unconditional, label-conditioned, and text-conditioned generation, respectively. All are evaluated with 10 NFEs and uniform trajectory via DDIM.

truncation error, and hence reduces the gap between the real and sampling distributions. We will provide further analysis on optimized timesteps regarding different solvers, datasets, and trajectories in *Supplementary Material*.

We also confirm the performance improvement by optimizing the timesteps one by one. As in Fig. 6, replacing the timestep gradually decreases the FID monotonically, confirming the correctness of Theorem 1 and the effectiveness of `TimeTuner`. One can also verify the effectiveness from Fig. 2. By replacing the timestep gradually, the distribution gap consistently decreases more and more significantly, demonstrating the strong capability of correcting the one-step and hence the accumulative truncation error.

4.3. Training Strategy of TimeTuner

Recall that we separately introduce a *sequential strategy* and a *parallel strategy* in Sec. 3.5. We deduce that there is a performance gap between them, mainly due to the extra error introduced by the parallel strategy at each denoising step. This can also be concluded from Tab. 5 clearly. Nevertheless, the parallel training strategy achieves on-par sampling performance, which provides a far more efficient version of `TimeTuner` empirically, and is extremely significant to the training of large NFE cases. Therefore, how to improve the performance of the parallel training strategy will be an interesting avenue for future research.

5. Conclusion

In this paper, we propose a plug-in algorithm for more accurate DPM acceleration, which replaces the original timestep to an optimized one. We provide a proof for an estimated error bound of the deterministic DE solver, to show the feasibility to achieve better sampling performance by simply optimizing the timestep. We conduct comprehensive experiments to demonstrate significant improvement of sampling quality under different NFEs.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (L222008), the Natural Science Foundation of China (U2336214, 62332019, 62302297), Beijing Hospitals Authority Clinical Medicine Development of special funding support (ZLRK202330), Shanghai Sailing Program (22YF1420300), and Young Elite Scientists Sponsorship Program by CAST (2022QNRC001).

References

[1] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *Int. Conf. Learn. Represent.*, 2022. 1, 2, 5

[2] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020. 2

[3] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Adv. Neural Inform. Process. Syst.*, 2021. 1

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Adv. Neural Inform. Process. Syst.*, 2017. 5

[7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2021. 6

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Adv. Neural Inform. Process. Syst.*, pages 6840–6851, 2020. 1, 2, 5

[9] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47–1, 2022. 2

[10] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 2

[11] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021. 2

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Int. Conf. Learn. Represent.*, 2018. 5

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4401–4410, 2019. 5

[14] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Adv. Neural Inform. Process. Syst.*, 2022. 1, 3, 5

[15] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Adv. Neural Inform. Process. Syst.*, pages 21696–21707, 2021. 2

[16] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021. 1

[17] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. 2

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, 2009. 3, 5

[19] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *arXiv preprint arXiv:1904.06991*, 2019. 5

[20] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 2022. 2

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2014. 5

[22] Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. OMS-DPM: Optimizing the model schedule for diffusion probabilistic models. In *ICML*, 2023. 2, 7

[23] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *Int. Conf. Learn. Represent.*, 2022. 2

[24] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Int. Conf. Comput. Vis.*, pages 3730–3738, 2015. 5

[25] 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. *arXiv preprint arXiv:2206.00927*, 2022. 2, 3, 5

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

[27] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. In *Int. Conf. Mach. Learn.*, 2021. 5

[28] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Int. Conf. Mach. Learn.*, pages 8162–8171, 2021. 1, 5

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, 2019. 5

[30] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, Mikhail Sergeevich Kudinov, and Jiansheng Wei. Diffusion-based voice conversion with fast maximum likelihood sampling scheme. In *Int. Conf. Learn. Represent.*, 2022. 2

[31] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10684–10695, 2022. 5

[32] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021. 2

[33] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 2

[34] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *Int. Conf. Learn. Represent.*, 2022. 2

[35] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Adv. Neural Inform. Process. Syst.*, 2016. 5

[36] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021. 2

[37] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Int. Conf. Mach. Learn.*, pages 2256–2265. PMLR, 2015. 1, 2

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Int. Conf. Learn. Represent.*, 2021. 1, 2, 3, 5

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Int. Conf. Learn. Represent.*, 2020. 1, 2

[40] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 2, 5

[41] Hideyuki Tachibana, Mocho Go, Muneyoshi Inahara, Yotaro Katayama, and Yotaro Watanabe. It\^{o}-taylor sampling scheme for denoising diffusion probabilistic models using ideal derivatives. *arXiv preprint arXiv:2112.13339*, 2021. 2

[42] Yunke Wang, Xiyu Wang, Anh-Dung Dinh, Bo Du, and Charles Xu. Learning to schedule in diffusion probabilistic models. In *ACM SIGKDD*, 2023. 2, 7

[43] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021. 2, 7

[44] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *Int. Conf. Learn. Represent.*, 2022. 2

[45] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5