arXiv:2404.14507v1 [cs.CV] 22 Apr 2024

Align Your Steps: Optimizing Sampling Schedules in Diffusion Models

Amirmojtaba Sabour[*] [1 2 3] **Sanja Fidler** [1 2 3] **Karsten Kreis** [1]

[1] NVIDIA [2] University of Toronto [3] Vector Institute

Project Page: https://research.nvidia.com/labs/toronto-ai/AlignYourSteps/

Figure 1. We introduce ***Align Your Steps*** *(AYS)*, a novel general framework for optimizing sampling schedules in diffusion models that significantly boosts the quality of outputs, especially when performing synthesis in few steps. Notice the improved details with AYS.

Abstract

Diffusion models (DMs) have established themselves as the state-of-the-art generative modeling approach in the visual domain and beyond. A crucial drawback of DMs is their slow sampling speed, relying on many sequential function evaluations through large neural networks. Sampling from DMs can be seen as solving a differential equation through a discretized set of noise levels known as the sampling schedule. While past works primarily focused on deriving efficient solvers, little attention has been given to finding optimal sampling schedules, and the entire literature relies on hand-crafted heuristics. In this work, for the first time, we propose a general and principled approach to optimizing the sampling schedules of DMs for high-quality outputs, called *Align Your Steps*. We leverage methods from stochastic calculus and find optimal schedules specific to different solvers, trained DMs and datasets. We evaluate our novel approach on several image, video as well as 2D toy data synthesis benchmarks, using a variety of different samplers, and observe that our optimized schedules outperform previous hand-crafted schedules in almost all experiments. Our method demonstrates the untapped potential of sampling schedule optimization, especially in the few-step synthesis regime.

1. Introduction

Diffusion models (DMs) have proven themselves to be extremely reliable probabilistic generative models that can produce high-quality data. They have been successfully applied to applications such as image synthesis (Dhariwal & Nichol, 2021; Ho et al., 2020; Song et al., 2020b; Rombach et al., 2021; Saharia et al., 2022; Ramesh et al., 2022), image super-resolution (Saharia et al., 2021b), image-to-image translation (Saharia et al., 2021a), image editing (Brooks et al., 2023), inpainting (Lugmayr et al., 2022), video synthesis (Ho et al., 2022; Blattmann et al., 2023b), text-to-3d generation (Poole et al., 2022; Lin et al., 2023), and even planning (Janner et al., 2022). However, sampling DMs requires multiple sequential forward passes through a large neural network, limiting their real-time applicability.

As a result, extensive research effort has gone into designing fast and efficient samplers of these models, broadly categorized into training-based and training-free methods. Training-based approaches, such as distillation, can significantly accelerate the sampling process but often require significant compute power, comparable to training the model itself, and face a trade-off between speed, diversity, and fi-

[*] Work done during an internship at NVIDIA.

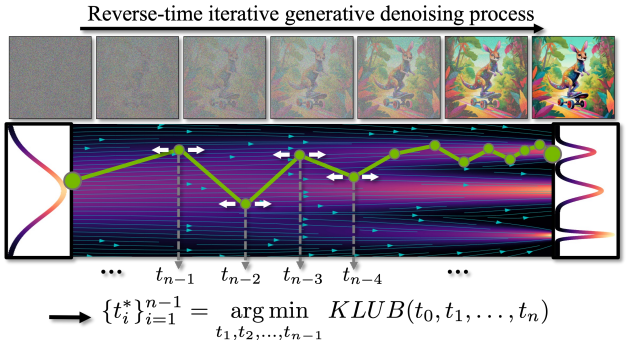


Figure 2. **Align Your Steps.** We minimize an upper bound on the Kullback-Leibler divergence (KLUB) between the true and linearized generative SDEs to find optimal DM sampling schedules.

delity (Salimans & Ho, 2022; Song et al., 2023; Sauer et al., 2023b; Luo et al., 2023; Yin et al., 2023), lagging behind standard DMs in terms of output quality, especially in large models. Although promising, these methods have not yet found wide-spread adoption by practitioners. On the other hand, since sampling from DMs corresponds to solving a generative Stochastic or Ordinary Differential Equation (SDE/ODE) in reverse time (Song et al., 2020b), training-free methods usually seek to derive more efficient SDE/ODE solvers, making them more broadly applicable to different models with relative ease (Lu et al., 2022a;b; Song et al., 2020a; Cui et al., 2023; Xu et al., 2023a; Karras et al., 2022).

Solving SDE/ODEs within the interval $[t_{min}, t_{max}]$ works by discretizing it into n smaller sub-intervals $t_{min} = t_0 < t_1 < \cdots < t_n = t_{max}$, and numerically solving the differential equation between consecutive t_i values. This discretization has been given many names in the literature, *e.g.* step size schedule, denoising schedule, timestep schedule, etc.[1] We will be referring to it as the *sampling schedule*. Changing the sampling schedule can significantly change the quality of the outputs (Karras et al., 2022); however, most prior works simply adopt one of a handful of heuristic schedules, such as simple polynomials and cosine functions. Although significant effort has gone into developing faster solvers, little research has been conducted to optimize the sampling schedule. We attempt to fill this gap by introducing a principled approach for optimizing the schedule in a dataset-specific manner, resulting in improved outputs given the same compute budget. We'll be focusing on stochastic SDE solvers. These solvers excel in sampling from diffusion models due to their built-in error-correction, allowing them to outperform ODE solvers.

In a toy example using a Gaussian data distribution (Sec. 3.1), we demonstrate the reliance of the optimal sampling schedule on the dataset characteristics and find that the optimal schedule significantly differs from heuristic sampling schedules used across the literature. With this as motivation, we propose *Align Your Steps* (AYS), a principled and general framework for optimizing the sampling schedule specific to any choice of dataset, model, and stochastic SDE solver. The framework is based on the observation that all stochastic SDE solvers can be reinterpreted as exactly solving an approximated linearized SDE on short intervals. This allows us to minimize the mismatch between solving the approximated linear SDE and the true generative SDE using techniques from stochastic calculus by framing it as an optimization problem over the sampling schedule (Fig. 2). Although the framework assumes the use of stochastic SDE solvers, we empirically find that the optimized schedules generalize to several popular ODE solvers as well. The proposed framework is general and applicable to all DMs regardless of the data modality, and it is the first general schedule optimization framework that leads to improved output quality.

We empirically evaluate our method by optimizing the schedule for various datasets and models. These include 2D toy data, standard image datasets such as CIFAR10 (Krizhevsky et al., 2009), FFHQ (Karras et al., 2019), and ImageNet (Deng et al., 2009), large scale text-to-image models widely used by practitioners such as Stable Diffusion (Rombach et al., 2021) and SDXL (Podell et al., 2023), as well as the recent video DM Stable Video Diffusion (Blattmann et al., 2023a). Our results show the practical advantages of optimizing the sampling schedule, ranging from fewer outliers in 2D point generation, enhanced quality in image generation, and improved temporal stability in video generation (Fig. 1).

Contributions. *(i)* We analytically establish the dependency of the optimal sampling schedule on the ground truth data distribution. *(ii)* We introduce *Align Your Steps*, a principled and general framework for optimizing the sampling schedule specific to any dataset, model and stochastic solver. *(iii)* We improve upon previous heuristic sampling schedules for many popular stochastic and deterministic solvers, especially in the low NFE regime. *(iv)* We provide the optimized schedules for several commonly used models in the appendix to allow for easy plug-and-play use by the research community.

2. Background

DMs are probabilistic generative models that inject noise into the data with a forward diffusion process and generate samples by learning and simulating a time-reversed backward diffusion process, initialized with a sample from a tractable distribution, *e.g.* Gaussian noise. We adopt the framework of Karras et al. (2022), denote the data distri-

[1]This is different from the noising schedule which specifies the amount of noise injection and scaling in the forward process. Please refer to Sec. 2 for details.

bution by $p_{\text{data}}(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d$, and define $p(\mathbf{x}; \sigma)$ as the distribution obtained by adding i.i.d. Gaussian noise of standard deviation σ to the data.

Forward process. Score-based diffusion models (Song et al., 2020b) progressively transform the data $p_{\text{data}}(\mathbf{x})$ towards a noise distribution through a forward noising process. This process is determined by a *noising schedule*, consisting of two functions $s(t), \sigma(t)$ that define the scaling and noise level at time t. Specifically, $\mathbf{x}_t = s(t)\hat{\mathbf{x}}_t$ where $\hat{\mathbf{x}}_t \sim p(\mathbf{x}, \sigma(t))$. The distribution of $\mathbf{x}_t$ is denoted as $p'(\mathbf{x}, t)$. Given the noising schedule, the forward noising process can be written in the form of the following SDE

$$d\mathbf{x}_t = \frac{\dot{s}(t)}{s(t)}\mathbf{x}_t + s(t)\sqrt{2\sigma(t)\dot{\sigma}(t)}d\mathbf{w}_t, \tag{1}$$

where $\mathbf{w}_t \in \mathbb{R}^d$ denotes a standard Wiener process.

Backward process and sampling. The forward SDE in Eq. (1) has an associated reverse-time diffusion process (Song et al., 2020b) given by

$$d\mathbf{x}_t = \left[\frac{\dot{s}(t)}{s(t)}\mathbf{x}_t - 2s(t)^2\sigma(t)\dot{\sigma}(t)\nabla_x \log p\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right)\right]dt + s(t)\sqrt{2\sigma(t)\dot{\sigma}(t)}d\bar{\mathbf{w}}_t, \tag{2}$$

where $\bar{\mathbf{w}}_t$ denotes a standard Wiener process backwards in time. However, there exists an entire class of reverse-time SDEs with matching marginals as the backward SDE in Eq. (2) (Huang et al., 2021; Karras et al., 2022; Cui et al., 2023). The most notable being the non-stochastic probability flow ODE, introduced by (Song et al., 2020b):

$$d\mathbf{x}_t = \left[\frac{\dot{s}(t)}{s(t)}\mathbf{x}_t - s(t)^2\sigma(t)\dot{\sigma}(t)\nabla_x \log p\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right)\right]dt. \tag{3}$$

As stated previously, sampling from a diffusion model boils down to solving one of these SDE/ODEs backward in time starting from random noise. This is done by discretizing the interval $[t_{min}, t_{max}]$ into n sub-intervals $t_{min} = t_0 < t_1 < \cdots < t_n = t_{max}$, known as a sampling schedule, and solving the SDE/ODEs on this schedule.

3. Optimizing Sampling Schedules

Contrary to previous works, which have primarily focused on deriving efficient SDE/ODE solvers using heuristic schedules for sampling, we focus on fundamentally optimizing the sampling schedule given a specific choice of (dataset, model, stochastic solver) for a large class of SDE solvers.

In Sec. 3.1, we first show how changing dataset characteristics causes the optimal sampling schedule to change. Next, in Sec. 3.2, we analyze the error introduced by discretizing the interval of the SDE into n sub-intervals that define the sampling schedule, and formulate finding an optimal schedule as an optimization problem which can be solved iteratively. Sec. 3.3 addresses implementation details.

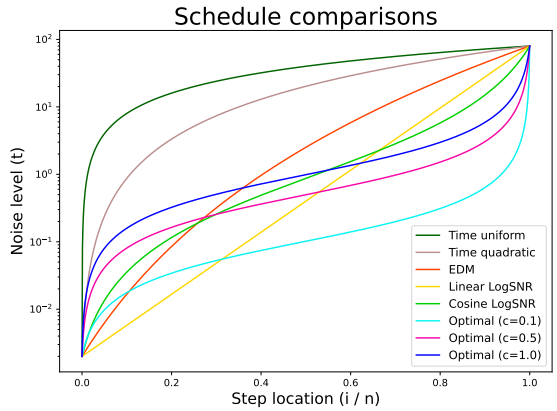


Figure 3. Comparing popular sampling schedules against the optimal schedules for Gaussian data.

3.1. The Need for Optimized Schedules

Although the sampling schedule used for solving SDE/ODEs is a powerful hyperparameter at our disposal, little research effort has gone into optimizing it. Especially in the relevant few-step synthesis regime, discretization errors can become significant (Atkinson et al., 2009) and having an optimal sampling schedule can make a considerable impact.

As a motivating example, we analyze a simple case where an optimal sampling schedule can be derived analytically. Consider the case where the initial distribution is an isotropic Gaussian with a standard deviation of c, *i.e.* $p_{data}(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$. We'll assume $s(t) = 1, \sigma(t) = t$ (Karras et al., 2022). Forward SDE and Probability Flow ODE then are

$$\begin{cases} \text{Forward SDE:} & d\mathbf{x}_t = \sqrt{2t}\, d\mathbf{w}_t, \\ \text{Reverse ODE:} & d\mathbf{x}_t = -t\nabla_x \log p(\mathbf{x}_t, t)dt. \end{cases} \tag{4}$$

In this setting, assuming use of the forward Euler method, also known as DDIM (Song et al., 2020a), to solve the reverse ODE, an optimal schedule can be derived analytically.

Theorem 3.1 (Proof in App. A.1)**.** *Let $p_{data}(\mathbf{x}) = \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$. Sample $\mathbf{x}_{t_{max}} \sim p(\mathbf{x}, t_{max})$ and solve the probability flow ODE using n forward euler steps along the schedule $t_{max} = t_n > t_{n-1} > \cdots > t_1 > t_0 = t_{min}$ to obtain $\bar{\mathbf{x}}_{t_{min}}$. The optimal schedule t^* minimizing the KL-divergence between $p(\mathbf{x}, t_{min})$ and the distribution of $\bar{\mathbf{x}}_{t_{min}}$ is given by*

$$\alpha_{min} := \arctan(t_{min}/c), \quad \alpha_{max} := \arctan(t_{max}/c)$$
$$\Rightarrow t_i^* = c\tan\left((1 - \frac{i}{n}) \times \alpha_{min} + (\frac{i}{n}) \times \alpha_{max}\right).$$

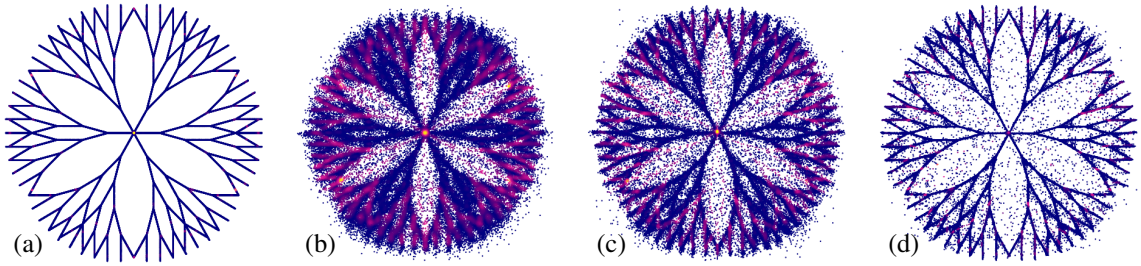


Figure 4. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 8 steps of SDE-DPM-Solver++(2M) with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points. The colors denote the local density of the samples where warmer colors correspond to higher density regions. See App. C.2 for details.

In this theorem, the distribution $p(\mathbf{x}, t_{min})$ is the output distribution of exactly solving the probability flow ODE from t_{max} to t_{min}. Therefore, the theorem states that the optimal schedule t^*, that has the minimum mismatch between its outputs and the outputs of exactly solving the ODE, has the interesting property that $\arctan(t^*/c)$ is a linear function.

In Fig. 3, we compare several popular sampling schedules used in practice against these optimal schedules when $t_{min} = 0.002, t_{max} = 80.0$ for various initial std. devs. c. The featured schedules include EDM (Karras et al., 2022), Linear LogSNR (Lu et al., 2022a;b), Cosine LogSNR (Hoogeboom et al., 2023; Nichol & Dhariwal, 2021), linear time (Song et al., 2020a), and quadratic time (Song et al., 2020a). This plot shows how changing the dataset (through changing the data distribution's std. dev. c) can have a significant impact on the optimal sampling schedule. Judging by how dissimilar the hand-crafted schedules appear compared to the optimal Gaussian ones, it is reasonable to believe that optimizing the schedules for each dataset could lead to significant performance gains. Note that in practice, it is common to normalize the input data to ensure unit variance. Yet, even only comparing the optimal schedule when $c = 1$ to the others, there remains a big difference between them. We show the distribution of outputs for different samplers in App. C.1.

3.2. Analyzing the Discretization Errors

Since the sampling schedule defines how the reverse-time generative SDE will be discretized, optimizing the schedule corresponds directly to minimizing the discretization error of solving the SDE/ODE. One method for analyzing such discretization errors in diffusions (and SDEs in general) is to use Girsanov's theorem (Oksendal, 1992). A simplified version of Girsanov's theorem is the following:

Theorem 3.2 (KL-divergence Upper bound (KLUB), proof in App. A.2)**.** *Consider the following two SDEs:*

$$\begin{cases} \textit{SDE 1}: & d\mathbf{x}_t = \mathbf{f}_1(\mathbf{x}_{0\to t}, t)dt + g(t)d\mathbf{w}_t \\ \textit{SDE 2}: & d\mathbf{x}_t = \mathbf{f}_2(\mathbf{x}_{0\to t}, t)dt + g(t)d\mathbf{w}_t \end{cases}$$

where $\mathbf{x}_{0\to t}$ represents the entire path from the start ($t = 0$) to the current time t (this formulation is useful for multi-step methods that benefit from having access to the history). Let P_1 and P_2 be the resulting probability distributions at time T of the outputs of SDE 1 and SDE 2, respectively.

Under mild regularity constraints, we have:

$$D_{\mathrm{KL}}(P_1 \| P_2) \leq \mathrm{KLUB}(0, T) := \frac{1}{2}\mathbb{E}_{P_1^{paths}}\left[\int_0^T \frac{||\mathbf{f}_1(\mathbf{x}_{0\to t}, t) - \mathbf{f}_2(\mathbf{x}_{0\to t}, t)||^2}{g(t)^2} dt\right], \quad (5)$$

where P_1^{paths} refers to the distribution over path space $\mathbf{x}_{0\to T} \in \mathcal{C}([0, T]; \mathbb{R}^d)$ generated by running SDE 1.

This theorem gives us an upper bound on the outputs' mismatch of two SDEs that share a diffusion term. In this work, our main goal is minimizing the mismatch between the outputs obtained by exactly solving the reverse-time generative SDE without discretization and the outputs of stochastic SDE solvers in practice, which use a finite sampling schedule. Most stochastic solvers work by decomposing the problem into multiple sub-intervals, within each of which the SDE is approximated by a linear SDE that has the same diffusion term. For these linear SDEs, exact numerical solutions exist which are used by the solvers. Therefore, for each stochastic SDE solver there exists a solver-specific linearized SDE, and the outputs of these solvers are the exact solutions of their respective linearized SDEs. As a result, we can use the theorem above to derive a Kullback-Leibler divergence Upper Bound (KLUB) between the outputs of practical stochastic solvers and the outputs of solving the reverse-time generative SDE without discretization. To clarify, solving the generative SDE without discretization is not possible in practice due to the nonlinear nature of the neural network. However, Girsanov's theorem offers us a tool to analyze the corresponding distribution regardless.

In the following, we will demonstrate deriving the KLUB for Stochastic-DDIM ($\eta = 1$) (Song et al., 2020a), and

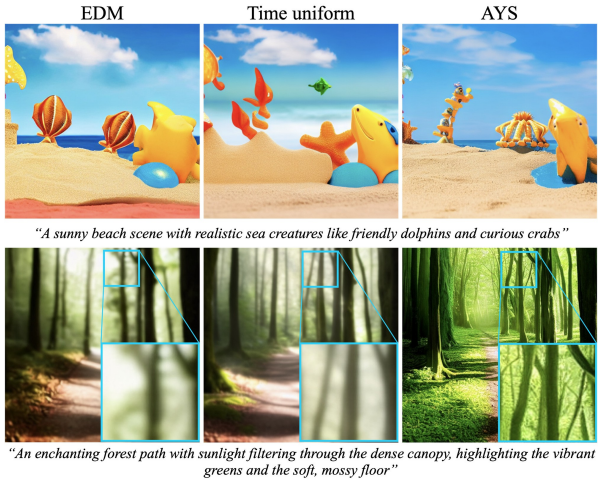


Figure 5. Side-by-side comparison of selected images generated with Stable Diffusion 1.5 with SDE-DPM-Solver++(2M) over 10 steps with different sampling schedules.

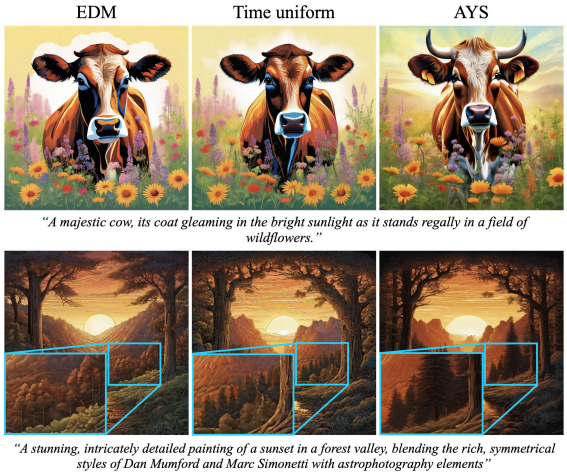


Figure 6. Side-by-side comparison of selected images generated with SDXL with 10 steps with different sampling schedules. The first and second rows use the SDE-DPM-Solver++(2M) and DPM-Solver++(2M) solvers respectively.

a similar procedure can be applied to other solvers with minimal adjustments. We follow Karras et al. (2022) and let $\mathbf{D}_\theta(x, \sigma)$ be the learnt denoiser function that takes in a noisy sample $\mathbf{x}$ with σ noise and denoises the sample. Plugging in the relation $\nabla_x \log p_\theta(\mathbf{x}, \sigma) = (D_\theta(\mathbf{x}, \sigma) - \mathbf{x})/\sigma^2$ into Eq. (2) yields the following *true learnt SDE*:

$$d\mathbf{x}_t = \left[\left(\frac{\dot{s}(t)}{s(t)} + \frac{2s(t)^2\dot{\sigma}(t)}{\sigma(t)}\right)\mathbf{x}_t - \frac{2s(t)^2\dot{\sigma}(t)}{\sigma(t)}\mathbf{D}_\theta\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right)\right]dt + s(t)\sqrt{2\sigma(t)\dot{\sigma}(t)}d\bar{\mathbf{w}}_t \tag{6}$$

Lu et al. (2022b) have shown that Stochastic-DDIM is the exact solution of a 1st order approximation of the true learnt SDE. This means when solving the SDE in the sub-interval $[t_{i-1}, t_i]$, using the assumption $\mathbf{D}_\theta(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)) \approx \mathbf{D}_\theta(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i))$, the *discretized learnt SDE* of this solver is

$$d\mathbf{x}_t = \left[\left(\frac{\dot{s}(t)}{s(t)} + \frac{2s(t)^2\dot{\sigma}(t)}{\sigma(t)}\right)\mathbf{x}_t - \frac{2s(t)^2\dot{\sigma}(t)}{\sigma(t)}\mathbf{D}_\theta\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right)\right]dt + s(t)\sqrt{2\sigma(t)\dot{\sigma}(t)}d\bar{\mathbf{w}}_t, \tag{7}$$

where the outputs of applying 1 step of Stochastic-DDIM from noise level $t_i \to t_{i-1}$ are the exact solution of this SDE. Note that this is a linear SDE since the denoiser does not rely on the current state $\mathbf{x}_t$, but only on the fixed $\mathbf{x}_{t_i}$ at the beginning of the interval, and its output can be treated as a constant vector inside the interval. Stitching together all these linear SDEs for the different sub-intervals gives us a *general discretized learnt SDE* that corresponds to applying the solver using the entire sampling schedule.

At this point, there are two SDEs that share the same diffusion term. The outputs of the true learnt SDE are samples obtained theoretically, given an unlimited compute budget, and outputs of the second general discretized SDE are samples obtained by running n steps of Stochastic-DDIM along the finite sampling schedule in practice. The goal is to optimize the schedule in such a way as to ensure these two output distributions are as close as possible to each other, and for that we can use our KLUB formalism from above.

To start, we consider a single sub-interval. Assuming both SDEs start from the forward diffusion process' distribution $p'(\mathbf{x}, t_i)$ and are run from $t_i \to t_{i-1}$, we can apply Theorem 3.2 backwards in time to obtain a KLUB between their output distributions. Letting the SDE in Eq. (6) be SDE 1 and the SDE in Eq. (7) be SDE 2 in the theorem, we obtain:

$$D_{\mathrm{KL}}(P^{\text{true}}_{t_i \to t_{i-1}} \| P^{\text{disc}}_{t_i \to t_{i-1}}) \leq 2 \times \mathbb{E}_{P^{\text{true paths}}_{t_i \to t_{i-1}}} \int_{t_{i-1}}^{t_i} \frac{s(t)^2\dot{\sigma}(t)}{\sigma(t)^3} \left\| \mathbf{D}_\theta\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right) - \mathbf{D}_\theta\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right)\right\|^2 dt. \tag{8}$$

Here $P^{\text{true}}_{t_i \to t_{i-1}}$ represents the distribution of running the true learnt SDE, $P^{\text{disc}}_{t_i \to t_{i-1}}$ denotes the distribution of running the discretized learnt SDE (that corresponds to Stochastic DDIM's 1-step outputs), and $P^{\text{true paths}}_{t_i \to t_{i-1}}$ is the distribution over path space of the true learnt SDE.

If we had a perfect score model, i.e. $\mathbf{D}_\theta(\mathbf{x}, \sigma) = E_{p_{\text{data}}(\mathbf{x}_0|\mathbf{x}_\sigma)}[\mathbf{x}_0]$, then $P^{\text{true paths}}_{t_i \to t_{i-1}}$ would perfectly match the path distributions of the forward noising process, and $P^{\text{true}}_{t_i \to t_{i-1}} = p'(\mathbf{x}, t_{i-1})$, where p' is the distribution of the forward noising process. We'll assume that $\mathbf{D}_\theta$ is sufficiently close to the true denoising function, and approximate it as such moving forward (for a more detailed error analysis, please refer to App. A.4). Applying this approximation to

the equation above results in the following:

$$
\begin{aligned}
&D_{\mathrm{KL}}(P^{\text{true}}_{t_i \to t_{i-1}} \| P^{\text{disc}}_{t_i \to t_{i-1}}) \\
&\leq 2 \times \mathbb{E}_{P^{\text{true paths}}_{t_i \to t_{i-1}}} \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \left\| \mathbf{D}_\theta\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right) - \mathbf{D}_\theta\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right) \right\|^2 dt \\
&\approx 2 \times \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \mathbb{E}_{\substack{\mathbf{x}_t \sim p'(\mathbf{x}, t) \\ \mathbf{x}_{t_i} \sim p'(\mathbf{x}_{t_i} | \mathbf{x}_t)}} \left\| \mathbf{D}_\theta\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right) - \mathbf{D}_\theta\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right) \right\|^2 dt.
\end{aligned} \tag{9}
$$

This final value can be estimated using Monte Carlo integration and (x_t, x_{t_i}) can be drawn from the forward diffusion.

This approach can be easily extended to the entire integration from $t_{max} \to t_{min}$. Assuming the sampling schedule is $t_{min} = t_0 < t_1 < \cdots < t_n = t_{max}$, we apply the same technique on all sub-intervals and combine them to achieve a total KLUB between the outputs of running the true learnt SDE and the general discretized learnt SDE (which corresponds to Stochastic-DDIM with n-steps following the sampling schedule). The total KLUB then is

$$
\begin{aligned}
&\mathrm{KLUB}(t_0, t_1, \ldots, t_n) \\
&= \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \mathbb{E}_{\substack{\mathbf{x}_t \sim p'_t \\ \mathbf{x}_{t_i} \sim p'_{t_i|t}}} \left\| \mathbf{D}_\theta\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right) - \mathbf{D}_\theta\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right) \right\|^2 dt.
\end{aligned} \tag{10}
$$

Note that each of the integrals only depends on the beginning and end of the intervals (due to the solver being first-order), allowing us to rewrite Eq. (10) as:

$$
\mathrm{KLUB}(t_0, t_1, \ldots, t_n) = \sum_{i=1}^{n} \mathrm{KLUB}(t_{i-1}, t_i). \tag{11}
$$

Finally, we formulate the problem of finding an optimal sampling schedule as minimizing this KLUB value, resulting in the following optimization:

$$
\begin{aligned}
t^*_{1,\ldots,n-1} &= \underset{t_1, t_2, \ldots, t_{n-1}}{\arg\min} \ \mathrm{KLUB}(t_0, t_1, \ldots, t_n) \\
&= \underset{t_1, t_2, \ldots, t_{n-1}}{\arg\min} \sum_{i=1}^{n} \mathrm{KLUB}(t_{i-1}, t_i),
\end{aligned} \tag{12}
$$

assuming $t_0 = t_{min}, t_n = t_{max}$ are fixed. This optimization is done iteratively by choosing one of the schedule indices $i \in \{1, \ldots, n-1\}$, discretizing a neighbourhood around t_i into several candidate points, computing the KLUB for each candidate, and setting t_i to the candidate with the least value. Due to the decomposition, this process can be highly parallelized for non-neighbouring indices. A pseudocode is given in App. B.1. We call this technique *Align Your Steps* (AYS).

3.3. Practical Considerations of KLUB Estimation

As discussed in the previous section, estimating the KLUB is the key to optimizing the sampling schedule. As such, an accurate estimator for the KLUB with low variance is required, and Importance Sampling with respect to time t is used to achieve this. Inspired by prior work (Vahdat et al., 2021) we select the importance sampling distribution based on Gaussian data assumptions. Specifically, we assume Gaussian data and analytically calculate all integration terms in Eq. (10). Then we sample t from a distribution whose probability density function (pdf) matches these calculated values, up to a constant factor. Empirically, we found that this approach significantly reduces the variance in our KLUB estimation and is effective across all datasets.

Under the Gaussian data assumption, we have the following:

Lemma 3.3 (Proof in App. A.3)**.** *Let $p_{data}(\mathbf{x}) = \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$. We assume $\mathbf{D}(\mathbf{x}, \sigma) = E_{p_{data}(\mathbf{x}_0|\mathbf{x}_\sigma)}[\mathbf{x}_0]$ to be the ideal denoiser. Then for all $t < t_i$ we have*

$$
\begin{aligned}
&\mathbb{E}_{\substack{\mathbf{x}_t \sim p'_t \\ \mathbf{x}_{t_i} \sim p'_{t_i|t}}} \left[\left\| \mathbf{D}\left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t)\right) - \mathbf{D}\left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i)\right) \right\|^2 \right] \\
&\qquad = c^4 \left(\frac{1}{\sigma(t)^2 + c^2} - \frac{1}{\sigma(t_i)^2 + c^2} \right).
\end{aligned} \tag{13}
$$

And applying this lemma to Eq. (10) yields:

$$
\mathrm{KLUB} \propto \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \left(\frac{1}{\sigma(t)^2 + c^2} - \frac{1}{\sigma(t_i)^2 + c^2} \right) dt. \tag{14}
$$

For simplicity, we will use $\sigma(t) = t, s(t) = 1$ (Karras et al., 2022) moving forward. Considering an example case of $(t_{i-1}, t_i, t_{i+1}) = (0.1, 0.2, 0.5)$, the values from the integral above range 3 orders of magnitude $[0 - 1000]$, and if Monte Carlo integration were to be used naively in this case, the estimator would have a huge variance. To fix this, we perform importance sampling on t according to the distribution $\pi(t)$ where

$$
\pi(t) \propto \frac{1}{t^3} \left(\frac{1}{t^2 + c^2} - \frac{1}{t_i^2 + c^2} \right) \tag{15}
$$

for $c = 0.5$. Given these t samples, we average the reweighted integration terms $||\mathbf{D}_\theta(\mathbf{x}_t, t) - \mathbf{D}_\theta(\mathbf{x}_{t_i}, t_i)||^2 / (\frac{1}{t^2+c^2} - \frac{1}{t_i^2+c^2})$ which yields the final estimation of the KLUB (up to a constant). This results in a much lower-variance estimator of the KLUB. A pseudocode and extra visualizations are given in App. B.1.

In practice, the schedules are optimized in a hierarchical fashion. Specifically, we start with a 10-step schedule initialized using one of the heuristic schedules $(t_0, t_1, \ldots, t_{10})$. This is then iteratively optimized on all the 9 intermediate points $(t_1, t_2, \ldots, t_9)$. At this initial stage, an early stopping mechanism is necessary to avoid over-optimizing, which is due to the optimization objective being an upper bound on the discretization error and not the error itself (see

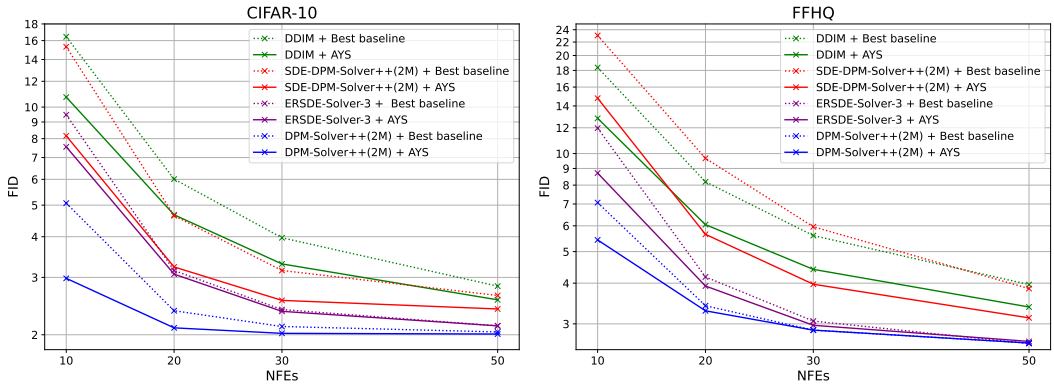


Figure 7. FID curves for different solvers and schedules on CIFAR10 *(left)* and FFHQ *(right)*. See Tables 5 and 6 in App. C.3 for more comprehensive results.

App. A.3 for a rigorous proof). After this process is finished, two rounds of subdivision and further fine-tuning are performed to obtain a 40-step schedule. Each time the schedule $(t_0, t_1, \ldots, t_n)$ is subdivided to obtain a new schedule with twice the number of steps $(t'_0, t'_1, \ldots, t'_{2n}$ where $t'_{2i} = t_i$ and $\log t'_{2i+1} = 0.5 \times (\log t_i + \log t_{i+1})$. After a subdivision, the training process only focuses on further optimizing the newly added intermediate points (i.e. t'_{2i+1}) and keeps the other points frozen. This allows the general "shape" of the schedule to become fixed, removing the need for early stopping during these later stages. Finally, to obtain a schedule with a different number of steps than $[10, 20, 40]$, we view the 40-step schedule as a piece-wise log-linear function and interpolate it to match the number of desired number of steps. See App. B.1 for more details. All in all, the schedule optimization requires only a few iterations to converge ($<$300).

4. Related Work

We briefly review prior work on accelerating DM sampling.

Various **training-free methods** have been introduced to speed up DM synthesis, including efficient ODE (Song et al., 2020a; Lu et al., 2022a;b; Zhang & Chen, 2022; Dockhorn et al., 2022; Liu et al., 2022; Zheng et al., 2024) and SDE solvers (Jolicoeur-Martineau et al., 2021; Xu et al., 2023a), as well as predictor-corrector methods (Song et al., 2020b; Zhao et al., 2023). They are easy to integrate into existing models and we use several of these samplers in our experiments.

Moreover, **training-based methods** include neural operators (Zheng et al., 2022b), truncated diffusion (Zheng et al., 2022a; Lyu et al., 2022), and distillation (Salimans & Ho, 2022; Meng et al., 2022; Song et al., 2023; Luo et al., 2023; Liu et al., 2023), often employing adversarial objectives (Xiao et al., 2022; Xu et al., 2023b; Sauer et al., 2023a; Yin et al., 2023; Kim et al., 2023). Although promising and almost reaching real-time sampling speeds, these methods often face trade-offs between inference speed, sample diversity, and output quality and require substantial compute for training. In practical applications, virtually all DMs rely on training-free samplers and solvers, which makes sampling schedule optimization a highly relevant task.

Watson et al. (2021) introduced a dynamic programming method aimed at minimizing the DM's evidence lower bound (ELBO) to select the best K-step schedule from a larger N-step schedule. Although their optimized schedules improve log likelihoods, they do not yield improvements in image quality (as measured by FID scores). This is expected, as optimizing an exact ELBO is not favourable for image quality (Ho et al., 2020). In follow-up work, Watson et al. (2022) proposed differentiating through sample quality scores, specifically KID (Binkowski et al., 2018), to create an optimized sampler, including a trainable sampling schedule. This method showed improved FID scores compared to the baseline DDIM/DDPM samplers; however, it is limited to image-based diffusion models and lacks versatility for data types. Our method's comparison with this previous work can be seen in App. C.4. In summary, we found their sampler to be outdated and it is unclear whether their optimized schedules are adaptable to different solvers. In contrast, our approach is derived in a principled manner, works on all data types, is compatible with a wide range of popular solvers, all while providing similar benefits. We also demonstrate our method on 2D data as well as video synthesis, which would not be possible with their technique. Wang et al. (2023) explore the concept of asynchronous time inputs, where the time input provided to the denoiser differs from the actual noise level of the current latent, with these parameters being trainable and learned. This approach is orthogonal to ours, as it keeps a fixed "sampling schedule" while learning the "denoiser inputs", and integrating it with our optimized schedules could potentially improve the results even further. Xia et al. (2023) proposes using a schedule predictor, trained with reinforcement learning, that takes in the noisy latents and the current timestep as inputs, and predicts the optimal next step to denoise to. This results in a sampling schedule that adapts based on the

Figure 8. Side-by-side comparisons for Stable Video Diffusion (Blattmann et al., 2023a). We animate a meme (image-to-video). Using the optimized schedule results in a more stable video; note the temporal artifacts of the cup for the baseline. See supplementary material for full videos.

sample being generated. However, the authors experiment exclusively with the first-order DDIM solver and it remains unclear if their learnt schedule predictor generalizes to more commonly used higher-order solvers.

5. Experiments

We demonstrate how optimizing the sampling schedule can significantly boost generation quality using the same number of forward evaluations (NFEs). We show how upsampling an optimized schedule with a small number of steps generalizes to higher NFE regimes as well as how using a schedule optimized on one solver's KLUB can generalize to other solvers. We compare outputs of various SDE/ODE solvers while using different schedules and show that optimized schedules lead to improvements almost across the board. Popular heuristic schedules listed in App. B.2.

We evaluate our method on various datasets including 2D toy data, widely-used image datasets, and text-to-image and image-to-video models. As sample quality metric, for CIFAR10 (Krizhevsky et al., 2009), FFHQ (Karras et al., 2019), and ImageNet (Deng et al., 2009), we use FID scores (Heusel et al., 2017). For the text-to-image and text-to-video models, we show the benefits of our method both qualitatively and quantitatively using human evaluation scores.

5.1. Toy Experiments

In Fig. 4, we show the advantages of optimized sampling schedules using a 2D toy dataset. We used a continuous-time EDM-based DM to learn the score, which was used to optimize the schedule. The samples generated with the optimized schedule more closely resemble the original distribution and have less outliers. Additional 2D results in App. C.2.

5.2. CIFAR10, FFHQ, ImageNet

For CIFAR10 and FFHQ experiments, we use pretrained continuous-time DMs from Karras et al. (2022). For ImageNet, we use the pretrained latent DM from (Rombach et al., 2021) with classifier free guidance with a scale of 2.0.

We use 3 different classes of stochastic solvers: Stochastic DDIM (Song et al., 2020a), second-order SDE-DPM-Solver++ (Lu et al., 2022b), and the recently proposed 1st, 2nd, and 3rd order ER-SDE-Solvers (Cui et al., 2023). We also report FID scores for two popular deterministic solvers, namely DDIM (Song et al., 2020a) and DPM-Solver++ (2M) (Lu et al., 2022b). For simplicity, no dynamic thresholding is used (Saharia et al., 2022).

In Fig. 7, we compare FIDs of generated images using the AYS schedule versus the best baseline schedule across four different solvers, including two stochastic and two deterministic ones. The results clearly demonstrate the benefits of optimizing the schedule. In some cases, *e.g.* for SDE-DPM-Solver++(2M), images generated with an optimized 20 step schedule achieve FIDs comparable to those from a 30-step default schedule, achieving a 1.5x speedup. Additionally, the results indicate that as the number of steps increases, the impact of different schedules diminishes, which is due to the discretization error becoming small. For more comprehensive results, please see Tables 5 and 6 in App. C.3.

In Table 1, we compare the quality of images generated using the EDM, time-uniform, and AYS schedules on ImageNet. While the FID values occasionally exhibit untypical behavior, such as deterioration with an increased number of steps, we suspect this is due to the absence of thresholding, potentially causing instabilities with higher-order solvers for small NFE. Nevertheless, in most instances, the optimized schedule outperforms the other two in all three metrics.

5.3. Text-to-Image

We also used our method to optimize sampling schedules for popular open-source text-to-image models, including Stable Diffusion 1.5 (Rombach et al., 2021), SDXL (Podell et al., 2023), and DeepFloyd-IF (Dee, 2023). For models that rely on classifier-free guidance, each guidance value essentially creates a different score model, suggesting that the optimal schedule should be tailored to each specific value. However, our experiments show that schedules optimized with default guidance values are effective across a reasonable range of values. See App. C.5 for FID vs. CLIP score pareto curves.

The benefits of the optimized schedules are evident in Figs. 5 and 6, which present side-by-side comparisons for SD 1.5 and SDXL, respectively. The results demonstrate that optimized schedules yield superior images in low NFE regimes,

Table 1. Sample fidelity (FID ↓, sFID ↓, Inception Score ↑) on the ImageNet 256 × 256 dataset.

	Sampling method	Schedule	NFE=10			NFE=20			NFE=30		
			FID ↓	sFID ↓	IS ↑	FID ↓	sFID ↓	IS ↑	FID ↓	sFID ↓	IS ↑
Stochastic Samplers	Stochastic DDIM	EDM	66.71	126.92	25.04	17.42	49.89	152.74	9.85	26.15	242.81
		Time-uniform	24.48	67.96	112.53	9.32	22.65	256.27	8.41	13.67	299.44
		AYS	**23.13**	**64.37**	**118.61**	**8.96**	**19.78**	**264.98**	**8.29**	**11.65**	**304.37**
	SDE-DPM-Solver++ (2M)	EDM	8.48	21.83	214.49	7.05	8.17	307.41	7.55	6.58	325.78
		Time-uniform	8.47	13.36	243.09	7.63	11.02	282.77	**7.14**	8.59	305.57
		AYS	**6.11**	**8.48**	**281.44**	**6.79**	**5.93**	**322.92**	7.28	**5.48**	**330.01**
	ER-SDE-Solver 1	EDM	17.78	35.25	147.57	6.99	12.70	255.69	6.20	8.51	282.52
		Time-uniform	8.79	18.33	222.93	6.25	8.19	280.74	6.09	6.56	293.47
		AYS	**8.36**	**15.91**	**266.44**	**6.06**	**7.28**	**282.06**	**5.87**	**5.97**	**295.40**
	ER-SDE-Solver 2	EDM	7.36	14.19	231.46	5.58	6.33	290.80	5.85	5.69	299.12
		Time-uniform	**5.28**	**6.19**	**277.57**	5.56	5.55	295.69	5.72	5.50	300.25
		AYS	5.38	6.24	275.35	**5.45**	**5.19**	**297.78**	**5.71**	**5.16**	**301.79**
	ER-SDE-Solver 3	EDM	6.94	13.01	237.70	5.58	6.13	292.75	5.87	5.61	299.33
		Time-uniform	**5.13**	**6.08**	**277.65**	5.52	5.57	295.94	**5.71**	5.48	301.52
		AYS	5.28	6.10	275.80	**5.47**	**5.17**	**298.05**	5.73	**5.14**	**302.40**
Deterministic Solvers	DDIM	Time-uniform	7.57	14.53	224.50	5.39	7.08	273.33	5.23	5.87	283.27
		AYS	**6.96**	**12.21**	**226.25**	**5.09**	**12.21**	**273.94**	**4.99**	**5.53**	**283.37**
	DPM-Solver++ (2M)	LogSNR	4.82	6.83	252.71	4.81	5.41	287.20	4.98	5.22	288.81
		AYS	**4.31**	**6.64**	**260.32**	**4.70**	**5.34**	**284.17**	**4.96**	**5.15**	**290.65**

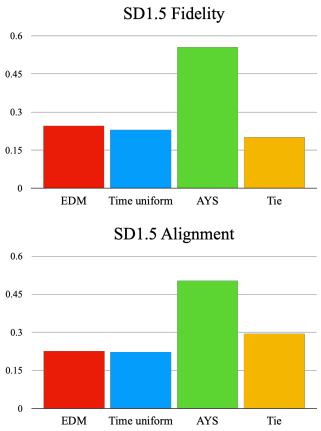


Figure 9. User study results on Stable Diffusion 1.5.

sometimes showing significant improvements. See App. C.5 for additional side-by-side comparisons.

To quantitatively evaluate the effectiveness of different schedules, we conducted a user study with 42 participants to assess *image fidelity* and *image-text alignment*. Each participant received a text prompt and three images generated with the EDM, time-uniform, and AYS schedules, respectively, using the same random seed. The SDE-DPM-Solver++(2M) (Lu et al., 2022b) was used to generate the images with 10 steps. The order of the images was randomly permuted to avoid any biases. The participants then select the superior option according to image-quality and image-text alignment, or a choice for a three-way tie. The results, summarized in Fig. 9, reveal a clear preference for our optimized schedule with respect to both metrics.

5.4. Video Generation Models

With the growing interest in video synthesis and open-source video diffusion models becoming available, it is important to look at efficient samplers in this area. However, few efficient samplers have been evaluated in this context. To address this gap, we also study the effect of our method in this domain, using the recent Stable Video Diffusion (SVD) (Blattmann et al., 2023a). We compare videos generated using DDIM with the default EDM schedule against our optimized schedule in Fig. 8. We find that the optimized schedule helps improve temporal color consistency and addresses the issue of over-saturation in later video frames. We also conduct a user-study on the generated videos, similar to the image-generation case. However, due to the continuous nature of SVD, the EDM schedule is used by default and serves as the baseline, and we compare it against our optimized schedule. The default DDIM (Song et al., 2020a) was used with 10 steps to generate the videos due to the instability of higher-order solvers. Once again the results, summarized in Table 2, reveal a clear preference for our optimized sampling schedule. More details about these experiments in App. C.6.

Table 2. Video generation user study results.

	EDM	AYS
SVD (Blattmann et al., 2023a)	42%	58%

6. Conclusions and Future Work

In summary, we present a novel framework for the optimization of sampling schedules in diffusion models aimed at enhancing the quality of generated samples in low NFE regimes. We successfully applied our method to several commonly used text-to-image and image-to-video models, and the schedules have been made publicly available[2]; see App. B.2 . Note that our framework is not strictly limited to diffusion models, and can also be integrated with recent, closely related generative techniques interpolating between data and noise, such as flow matching (Lipman et al., 2022; Esser et al., 2024) and stochastic interpolants (Albergo et al., 2023; Ma et al., 2024). When considering generative modeling with a Gaussian noise prior, these methods correspond to re-formulations of the same underlying generation framework and always allow us to form a generative SDE, necessary for the application of AYS. Looking forward, there are promising avenues for future research, including extending this framework to label- or text-conditional schedule optimization and applying it to single-step higher-order ODE solvers, such as Heun or Runge-Kutta methods.

[2]We also provide a colab notebook which shows how to use these schedules in practice on our project page.

Broader Impact

Diffusion models have evolved into a powerful and highly expressive generative modeling framework. Our novel method fundamentally advances diffusion models and accelerates their sampling. Faster synthesis can reduce diffusion models' inference compute demands, thereby decreasing their energy footprint, and it is also important for real-time applications. However, our approach is broadly applicable and its societal impact therefore depends on the specific domain and where and how the accelerated models are used. In our work, we validate the proposed techniques in the context of complex image and video synthesis, which have important content creation applications and can, for instance, improve the artistic workflow of digital artists and democratize creative expression. However, deep generative models like diffusion models can also be used to produce deceptive imagery and videos, as discussed, for instance, in Vaccari & Chadwick (2020); Nguyen et al. (2021); Mirsky & Lee (2021). Therefore, they need to be used with an abundance of caution.

References

Deepfloyd. if. https://github.com/deep-floyd/IF, 2023.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Atkinson, K., Han, W., and Stewart, D. E. *Numerical Solution of Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2009.

Bain, M., Nagrani, A., Varol, G., and Zisserman, A. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE International Conference on Computer Vision*, 2021.

Binkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., and Lorenz, D. Stable video diffusion: Scaling latent video diffusion models to large datasets. *ArXiv*, abs/2311.15127, 2023a.

Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., and Kreis, K. Align your latents: High-resolution video synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023b.

Brooks, T., Holynski, A., and Efros, A. A. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023.

Chen, S., Chewi, S., Li, J., Li, Y., Salim, A., and Zhang, A. R. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *ArXiv*, abs/2209.11215, 2022.

Cui, Q., Zhang, X., Lu, Z., and Liao, Q. Elucidating the solution space of extended reverse-time sde for diffusion models. *ArXiv*, abs/2309.06169, 2023.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis, 2021.

Dockhorn, T., Vahdat, A., and Kreis, K. GENIE: Higher-Order Denoising Diffusion Solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., and Rombach, R. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

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

Hoogeboom, E., Heek, J., and Salimans, T. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, 2023.

Huang, C.-W., Lim, J. H., and Courville, A. C. A variational perspective on diffusion-based generative models and score matching. *ArXiv*, abs/2106.02808, 2021.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Jolicoeur-Martineau, A., Li, K., Piche-Taillefer, R., Kachman, T., and Mitliagkas, I. Gotta go fast when generating data with score-based models. *ArXiv*, abs/2105.14080, 2021.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *ArXiv*, abs/2206.00364, 2022.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., and Lin, T.-Y. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, L., Ren, Y., Lin, Z., and Zhao, Z. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations (ICLR)*, 2022.

Liu, X., Zhang, X., Ma, J., Peng, J., and Liu, Q. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *ArXiv*, abs/2206.00927, 2022a.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *ArXiv*, abs/2211.01095, 2022b.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Gool, L. V. Repaint: Inpainting using denoising diffusion probabilistic models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11451–11461, 2022.

Luo, S., Tan, Y., Huang, L., Li, J., and Zhao, H. Latent consistency models: Synthesizing high-resolution images with few-step inference. *ArXiv*, abs/2310.04378, 2023.

Lyu, Z., Xudong, X., Yang, C., Lin, D., and Dai, B. Accelerating diffusion models via early stop of the diffusion process. *ArXiv*, abs/2205.12524, 2022.

Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

Meng, C., Gao, R., Kingma, D. P., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14297–14306, 2022.

Mirsky, Y. and Lee, W. The Creation and Detection of Deepfakes: A Survey. *ACM Comput. Surv.*, 54(1), 2021.

Nguyen, T. T., Nguyen, Q. V. H., Nguyen, C. M., Nguyen, D., Nguyen, D. T., and Nahavandi, S. Deep Learning for Deepfakes Creation and Detection: A Survey. *arXiv:1909.11573*, 2021.

Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021.

Oksendal, B. *Stochastic Differential Equations (3rd Ed.): An Introduction with Applications*. Springer-Verlag, Berlin, Heidelberg, 1992. ISBN 3387533354.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Muller, J., Penna, J., and Rombach, R. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *ArXiv*, abs/2307.01952, 2023.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021.

Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M. Palette: Image-to-image diffusion models. *ACM SIGGRAPH 2022 Conference Proceedings*, 2021a.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. Image super-resolution via iterative refinement. *arXiv:2104.07636*, 2021b.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet,

D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv*, abs/2205.11487, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models, 2022.

Sauer, A., Lorenz, D., Blattmann, A., and Rombach, R. Adversarial diffusion distillation. *ArXiv*, abs/2311.17042, 2023a.

Sauer, A., Lorenz, D., Blattmann, A., and Rombach, R. Adversarial diffusion distillation. *ArXiv*, abs/2311.17042, 2023b.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *ArXiv*, abs/2010.02502, 2020a.

Song, Y., Sohl-Dickstein, J. N., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models, 2023.

Vaccari, C. and Chadwick, A. Deepfakes and Disinformation: Exploring the Impact of Synthetic Political Video on Deception, Uncertainty, and Trust in News. *Social Media + Society*, 6(1):2056305120903408, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Wang, Y., Wang, X., Dinh, A.-D., Du, B., and Xu, C. Learning to schedule in diffusion probabilistic models. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2478–2488, 2023.

Watson, D., Ho, J., Norouzi, M., and Chan, W. Learning to efficiently sample from diffusion probabilistic models. *ArXiv*, abs/2106.03802, 2021.

Watson, D., Chan, W., Ho, J., and Norouzi, M. Learning fast samplers for diffusion models by differentiating through sample quality. *ArXiv*, abs/2202.05830, 2022.

Xia, M., Shen, Y., Lei, C., Zhou, Y., Yi, R., Zhao, D., Wang, W., and Liu, Y.-j. Towards more accurate diffusion model acceleration with a timestep aligner. *arXiv preprint arXiv:2310.09469*, 2023.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. In *International Conference on Learning Representations (ICLR)*, 2022.

Xu, Y., Deng, M., Cheng, X., Tian, Y., Liu, Z., and Jaakkola, T. Restart sampling for improving generative processes. *ArXiv*, abs/2306.14878, 2023a.

Xu, Y., Zhao, Y., Xiao, Z., and Hou, T. Ufogen: You forward once large scale text-to-image generation via diffusion gans. *ArXiv*, abs/2311.09257, 2023b.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. One-step diffusion with distribution matching distillation. *ArXiv*, abs/2311.18828, 2023.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *ArXiv*, abs/2204.13902, 2022.

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.

Zheng, H., He, P., Chen, W., and Zhou, M. Truncated diffusion probabilistic models. *ArXiv*, abs/2202.09671, 2022a.

Zheng, H., Nie, W., Vahdat, A., Azizzadenesheli, K., and Anandkumar, A. Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning*, 2022b.

Zheng, K., Lu, C., Chen, J., and Zhu, J. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36, 2024.

Appendix

A Theoretical Details **14**
A.1 Optimal Schedule for Isotropic Gaussian . . . 14
A.2 Deriving the KL-Divergence Upper Bound . . . 15
A.3 Early Stopping is a Necessity . . . 16
A.4 Exact Error Analysis . . . 18

B Experiment Details **19**
B.1 Practical Implementation Details . . . 19
B.2 Popular Sampling Schedules . . . 21
B.3 Optimized Schedules for Large Scale Models . . . 21

C Additional Results **21**
C.1 Gaussian Data Extras . . . 21
C.2 Extra 2D Experiments . . . 21
C.3 CIFAR10, FFHQ, and ImageNet Details . . . 23
C.4 Comparison with Watson et al. . . . 25
C.5 Text-to-Image Extras . . . 26
C.6 Stable Video Diffusion Details . . . 28

A. Theoretical Details

A.1. Optimal Schedule for Isotropic Gaussian

In the simple Gaussian setting where $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, c^2\mathbf{I}_{d\times d})$, the score after diffusion for time t can be analytically calculated as follows

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}, t) = -\frac{\mathbf{x}}{c^2 + t^2}. \tag{16}$$

Using this score, we can derive what 1 step of forward Euler will be, when going from noise level b to a using the probability flow ODE (see Eq. (4)) as follows

$$\hat{\mathbf{x}}_a = \mathbf{x}_b + (a - b) \times \frac{b}{b^2 + c^2}\mathbf{x}_b = (\frac{ab + c^2}{b^2 + c^2})\mathbf{x}_b. \tag{17}$$

Assuming that $\mathbf{x}_b \sim \mathcal{N}(\mathbf{0}, (c^2 + b^2)\mathbf{I}_{d\times d})$, the distribution obtained from forward Euler will be $\hat{\mathbf{x}}_a \sim \mathcal{N}(\mathbf{0}, (\frac{ab+c^2}{b^2+c^2})^2 \times (b^2 + c^2)\mathbf{I}_{d\times d})$.

This can be easily extended to n steps. Assuming that $a = t_0 < t_1 < \cdots < t_n = b$ is the schedule that is used to perform n steps from noise level b to a. Letting $\hat{\mathbf{x}}_a$ be the output, we have

$$\hat{\mathbf{x}}_a = \prod_{i=1}^{n} \left(\frac{t_{i-1}t_i + c^2}{t_i^2 + c^2} \right) \times \mathbf{x}_b \tag{18}$$

and similarly we will have $\hat{\mathbf{x}}_a \sim \mathcal{N}(\mathbf{0}, \prod_{i=1}^{n} (\frac{t_{i-1}t_i+c^2}{t_i^2+c^2})^2 \times (b^2 + c^2)\mathbf{I}_{d\times d})$.

We're looking to find the optimal values of t_i such that the KL-divergence between $\hat{\mathbf{x}}_a$ and $\mathbf{x}_a$ is minimized. Since both distributions are Gaussian, the KL-divergence has a closed form:

$$\begin{aligned}
D_{\mathrm{KL}}(p(x_a)\|p(\hat{\mathbf{x}}_a)) &= D_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, (t_0^2 + c^2)\mathbf{I}_{d\times d}),\ \mathcal{N}(\mathbf{0}, \prod_{i=1}^{n} (\frac{t_{i-1}t_i + c^2}{t_i^2 + c^2})^2 \times (t_n^2 + c^2)\mathbf{I}_{d\times d}) \right) \\
&= \frac{1}{2}\left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) \right] \\
&= \frac{1}{2}\left[d \times \log \frac{1}{f(t_0, \ldots, t_n)} - d + d \times f(t_0, \ldots, t_n) + 0 \right] \\
&= \frac{d}{2}\left(-\log f(t_0, \ldots, t_n) + f(t_0, \ldots, t_n) - 1 \right)
\end{aligned} \tag{19}$$

where

$$f(t_0, \ldots, t_n) = \frac{(t_0^2 + c^2)(t_1^2 + c^2)^2(t_2^2 + c^2)^2 \ldots (t_{n-1}^2 + c^2)^2(t_n^2 + c^2)}{(t_0t_1 + c^2)^2(t_1t_2 + c^2)^2 \ldots (t_{n-1}t_n + c^2)^2}.$$

To minimize this expression w.r.t. all t_i values, the partial derivatives of the expression must be zero. Formally, for $i \in [1, n-1]$ we have

$$\frac{\partial}{\partial t_i} D_{\mathrm{KL}}(p(\mathbf{x}_a)\|p(\hat{\mathbf{x}}_a)) = \frac{d}{2} \times \frac{\partial}{\partial t_i} f(t_0, \ldots, t_n) \times \left(1 - \frac{1}{f(t_0, \ldots, t_n)} \right). \tag{20}$$

Using the Cauchy–Schwarz inequalities we have

$$\left.\begin{aligned}
&(t_0^2 + c^2)(t_1^2 + c^2) > (t_0t_1 + c^2)^2 \\
&\ldots \\
&(t_{n-1}^2 + c^2)(t_n^2 + c^2) > (t_{n-1}t_n + c^2)^2
\end{aligned}\right\} \Rightarrow f(t_0, \ldots, t_n) > 1 \Rightarrow \left(1 - \frac{1}{f(t_0, \ldots, t_n)} \right) > 0, \tag{21}$$

where the inequalities are strict because of the assumption that all t_i are distinct. Therefore the partial derivative of f w.r.t. all t_i must be zero $\frac{\partial}{\partial t_i} f(t_0, \ldots, t_n) = 0$.

Computing this partial derivative gives

$$\frac{\partial}{\partial t_i} f(t_0, \ldots, t_n) = 0 \Rightarrow \frac{\partial}{\partial t_i}\left(\frac{(t_i^2 + c^2)^2}{(t_{i-1}t_i + c^2)^2 (t_i t_{i+1} + c^2)^2}\right) = 0$$
$$\Rightarrow \frac{\partial}{\partial t_i}\left(\frac{(t_i^2 + c^2)}{(t_{i-1}t_i + c^2)(t_i t_{i+1} + c^2)}\right) = 0$$

$$\Rightarrow (2t_i)(t_{i-1}t_i + c^2)(t_i t_{i+1} + c^2) = (t_i^2 + c^2)(2t_i t_{i-1} t_{i+1} + c^2(t_{i-1} + t_{i+1}))$$
$$\Rightarrow t_i^2(t_{i-1} + t_{i+1}) + 2m(c^2 - t_{i-1}t_{i+1}) - (t_{i-1} + t_{i+1})c^2 = 0$$
$$\Rightarrow t_i = \frac{(t_{i-1}t_{i+1} - c^2) + \sqrt{(t_{i-1}^2 + c^2)(t_{i+1}^2 + c^2)}}{t_{i-1} + t_{i+1}}. \tag{22}$$

This equation can be simplified and written as the following:

$$\alpha_{i-1} \coloneqq \arctan(t_{i-1}/c), \;\; \alpha_{i+1} \coloneqq \arctan(t_{i+1}/c) \Rightarrow t_i = c \tan\left(\frac{\alpha_{i-1} + \alpha_{i+1}}{2}\right). \tag{23}$$

Using this result, the values of $\arctan(t_i/c)$ must be linear in the optimal schedule, which concludes the proof.

A.2. Deriving the KL-Divergence Upper Bound

To derive Theorem 3.2, we borrowed the argument from Section 5.1 of (Chen et al., 2022).

As a reminder, the following two SDEs are considered

$$\begin{cases} \text{SDE 1}: & d\mathbf{x}_t = \mathbf{f}_1(\mathbf{x}_{0\to t}, t)dt + g(t)d\mathbf{w}_t \\ \text{SDE 2}: & d\mathbf{x}_t = \mathbf{f}_2(\mathbf{x}_{0\to t}, t)dt + g(t)d\mathbf{w}_t \end{cases}$$

where $\mathbf{x}_{0\to t}$ represents the entire path from the start ($t = 0$) to the current time t (this formulation is useful for multi-step methods that benefit from having access to the history).

We start with some notations. When applying Girsanov's theorem, it is convenient to think about a single stochastic process $(\mathbf{x}_t)_{t\in[0,T]}$ and to consider different measures over the path space $\mathcal{C}([0,T];\mathbb{R}^d)$. A stochastic process can be viewed as a function from sample space to path space, i.e. $\mathbf{x}(\omega) : \Omega \to \mathcal{C}([0,T];\mathbb{R}^d)$.

We define two measures over the path space:

- Q_{paths}, under which $(\mathbf{x}_t)_{t\in[0,T]}$ has the law of SDE 1,
- P_{paths}, under which $(\mathbf{x}_t)_{t\in[0,T]}$ has the law of SDE 2.

Assume that $\mathbf{b}_t = \frac{\mathbf{f}_2(\mathbf{x}_{0\to t},t) - \mathbf{f}_1(\mathbf{x}_{0\to t},t)}{g(t)}$ and let $\mathbf{B}_t$ be a Brownian motion under Q_{paths}. Let $\mathcal{E} \coloneqq \exp\left(\int_0^T \mathbf{b}_s d\mathbf{B}_s - \frac{1}{2}\int_0^T ||\mathbf{b}_s||^2 ds\right)$. According to (Chen et al., 2022), under mild regularity constraints, $\mathcal{E}$ is a random variable such that $\mathbb{E}_{Q_{paths}}[\mathcal{E}] = 1$. Therefore, we can define P'_{paths} to be a measure that satisfies $dP'_{paths} = \mathcal{E} dQ_{paths}$. According to theorem 8 of (Chen et al., 2022), under P'_{paths} the process $t \to B_t - \int_0^t b_s ds$ is a Brownian motion, which we will call $\boldsymbol{\beta}_t$. We can rewrite this as

$$d\mathbf{B}_t = \mathbf{b}_t dt + d\boldsymbol{\beta}_t. \tag{24}$$

Using the definition of Q_{paths}, we know that

$$d\mathbf{x}_t = \mathbf{f}_1(\mathbf{x}_{0\to t}, t)dt + g(t)d\mathbf{B}_t, \quad \mathbf{x}_0 \sim p(\mathbf{x}). \tag{25}$$

Since P'_{paths} is absolutely continuous with respect to Q_{paths}, i.e. $P'_{paths} \ll Q_{paths}$, this equation will also hold under P'_{paths}. Plugging Eq. (24) into the above, we can conclude that under P'_{paths} we have

$$d\mathbf{x}_t = \mathbf{f}_1(\mathbf{x}_{0\to t}, t)dt + g(t)\left(\mathbf{b}_t dt + d\boldsymbol{\beta}_t\right) \tag{26}$$

$$= \mathbf{f}_2(\mathbf{x}_{0\to t}, t)dt + g(t)d\boldsymbol{\beta}_t, \quad \mathbf{x}_0 \sim p(\mathbf{x}). \tag{27}$$

Since $\boldsymbol{\beta}_t$ is a Brownian motion under P'_{paths}, this becomes the exact same as SDE 2. Therefore $P'_{paths} = P_{paths}$. In other words

$$\frac{dP_{paths}}{dQ_{paths}} = \mathcal{E}. \tag{28}$$

Using this expression in the KL-divergence we have

$$\begin{aligned}
D_{\mathrm{KL}}(Q_{paths}||P_{paths}) &= \mathbb{E}_{Q_{paths}} \log \frac{dQ_{paths}}{dP_{paths}} \\
&= -\mathbb{E}_{Q_{paths}} \log \mathcal{E} \\
&= \mathbb{E}_{Q_{paths}} \left(-\int_0^T \mathbf{b}_s d\mathbf{B}_s + \frac{1}{2}\int_0^T ||\mathbf{b}_s||^2 ds \right) \\
&\overset{(i)}{=} \mathbb{E}_{Q_{paths}} \left(\frac{1}{2}\int_0^T ||\mathbf{b}_s||^2 ds \right) \\
&= \frac{1}{2}\mathbb{E}_{Q_{paths}} \left(\int_0^T \frac{||\mathbf{f}_1(\mathbf{x}_{0\to s}, s) - \mathbf{f}_2(\mathbf{x}_{0\to s}, s)||^2}{g(s)^2} ds \right),
\end{aligned}$$

where (i) is due to the martingale property of Ito integrals.

Since Q, P are marginals of Q_{paths} and P_{paths} at time $t = T$, by the data processing inequality, the KL-divergence $D_{\mathrm{KL}}(Q||P)$ is upper-bounded by the KL-divergence $D_{\mathrm{KL}}(Q_{paths}||P_{paths})$ which concludes the proof.

A.3. Early Stopping is a Necessity

In this section, we prove that the schedule that minimizes the KLUB isn't necessarily going to minimize the KL-divergence too. We will do this by once again considering the simple Gaussian case where $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I}_{d\times d})$.

The proof is by contradiction. Let's assume that the schedule minimizing the KLUB must also always minimize the KL as well. We'll consider the family of Extended Reverse-Time SDEs (ERSDEs introduced in Cui et al. (2023)) that have $h(t) = \lambda \times \sqrt{2t}$ for some constant λ. The SDE formulation for these will be

$$\begin{aligned}
d\mathbf{x}_t &= -(\lambda^2+1)t\nabla_x \log p(\mathbf{x}_t, t)\ \ dt + \lambda\sqrt{2t}\ \ d\mathbf{w}_t \\
\Rightarrow d\mathbf{x}_t &= (\lambda^2+1)\left(\frac{\mathbf{x}_t - \mathbf{x}_\theta(\mathbf{x}_t, t)}{t}\right)\ \ dt + \lambda\sqrt{2t}\ \ d\mathbf{w}_t \\
\Rightarrow d\mathbf{x}_t &= (\lambda^2+1)\frac{\mathbf{x}_t}{t} - (\lambda^2+1)\left(\frac{\mathbf{x}_\theta(\mathbf{x}_t, t)}{t}\right)\ \ dt + \lambda\sqrt{2t}\ \ d\mathbf{w}_t,
\end{aligned}$$

and Cui et al. (2023) have shown that these SDEs share the same marginals as the original reverse-time SDE and probability flow ODE.

Assuming we use a first-order approximation for $\mathbf{x}_\theta$ as our solver, the KLUB of this solver will be

$$\mathrm{KLUB} = \mathbb{E}_{\substack{\mathbf{x}_t\sim p_t \\ \mathbf{x}_{t_{next}}\sim p_{t_{next}|t}}} \int_{t_{min}}^{t_{max}} \frac{||(\lambda^2+1)/t \times \mathbf{x}_\theta(\mathbf{x}_t, t) - (\lambda^2+1)/t \times \mathbf{x}_\theta(\mathbf{x}_{t_{next}}, t_{next})||^2}{\lambda^2 \times 2t} dt \tag{29}$$

$$= \frac{(\lambda^2+1)^2}{2\lambda^2} \int_{t_{min}}^{t_{max}} \frac{1}{t^3} \mathbb{E}_{\substack{\mathbf{x}_t\sim p_t \\ \mathbf{x}_{t_{next}}\sim p_{t_{next}|t}}} ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_\theta(\mathbf{x}_{t_{next}}, t_{next})||^2 dt, \tag{30}$$

where t_{next} is the smallest value in the sampling schedule larger than t. Since all the terms containing λ only appear as a constant outside the integral, the schedule that minimizes the KLUB is the same for all $\lambda > 0$. As such, using our initial assumption, that schedule is KL-minimizing for all these solvers as well.

Since every term in the KLUB can be found analytically for this Gaussian example, we can derive exactly what the KLUB-optimal schedule will be. To do this, we start by noting that

$$-\frac{\mathbf{x}}{t^2+c^2} = \nabla_{\mathbf{x}} \log p(\mathbf{x}, t) = \frac{\mathbf{x}_\theta(\mathbf{x}, t) - \mathbf{x}}{t^2}$$
$$\Rightarrow \mathbf{x}_\theta(\mathbf{x}, t) = \frac{c^2}{t^2+c^2}\mathbf{x}$$

We can use this to calculate the expectations inside the integral of Eq. (30) as follows

$$\begin{aligned}
\mathbb{E}_{\substack{\mathbf{x}_t \sim p_t \\ \mathbf{x}_{t_{next}} \sim p_{t_{next}|t}}} ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_\theta(\mathbf{x}_{t_{next}}, t_{next})||^2 &= \mathbb{E}_{\substack{\mathbf{x}_t \sim \mathcal{N}(0,(t^2+c^2)\mathbf{I}) \\ \mathbf{x}_{t_{next}} \sim \mathcal{N}(\mathbf{x}_t, \sqrt{t_{next}^2 - t^2}\mathbf{I})}} ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_\theta(\mathbf{x}_{t_{next}}, t_{next})||^2 \\
&= \mathbb{E}_{\substack{\mathbf{x}_t \sim \mathcal{N}(0,(t^2+c^2)\mathbf{I}) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}} ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_\theta(\mathbf{x}_t + \sqrt{t_{next}^2 - t^2}\boldsymbol{\epsilon}, t_{next})||^2 \\
&= \mathbb{E}_{\substack{\mathbf{x}_t \sim \mathcal{N}(0,(t^2+c^2)\mathbf{I}) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}} \left\| \frac{c^2}{t^2+c^2}\mathbf{x}_t - \frac{c^2}{t_{next}^2+c^2}(\mathbf{x}_t + \sqrt{t_{next}^2 - t^2}\boldsymbol{\epsilon}) \right\|^2 \\
&= c^4 \mathbb{E}_{\substack{\mathbf{x}_t \sim N(0,(t^2+c^2)\mathbf{I}) \\ \boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}} \left\| \left(\frac{1}{t^2+c^2} - \frac{1}{t_{next}^2+c^2} \right) \mathbf{x}_t - \frac{\sqrt{t_{next}^2 - t^2}}{t_{next}^2+c^2}\boldsymbol{\epsilon} \right\|^2 \\
&= c^4 \times d \times \left[\left(\frac{1}{t^2+c^2} - \frac{1}{t_{next}^2+c^2} \right)^2 (t^2+c^2) + \frac{t_{next}^2 - t^2}{(t_{next}^2+c^2)^2} \right] \\
&= c^4 \times d \times \left(\frac{1}{t^2+c^2} - \frac{1}{t_{next}^2+c^2} \right),
\end{aligned}$$

where we used the fact that $\mathbf{x}_t, \boldsymbol{\epsilon}$ are independent random variables, and $\boldsymbol{\epsilon}$ has zero mean.

Assuming the sampling schedule is $a = t_0 < t_1 < \cdots < t_n = b$ and plugging this into Eq. (30) we have

$$\begin{aligned}
\text{KLUB} &\propto \int_{t_{min}}^{t_{max}} \frac{1}{t^3} \left(\frac{1}{t^2+c^2} - \frac{1}{t_{next}^2+c^2} \right) dt \\
&\propto \sum_{i=1}^{n} \left[-\frac{c^2 t_i^2}{c^2+t_i^2} \left(\frac{1}{t_i^2} - \frac{1}{t_{i-1}^2} \right) + \log\left(\frac{t_i^2+c^2}{t_{i-1}^2+c^2} \right) - \log\left(\frac{t_i^2}{t_{i-1}^2} \right) \right] \\
&= \log\left(\frac{b^2+c^2}{a^2+c^2} \right) - \log\frac{b^2}{a^2} + c^2 \sum_{i=1}^{n} \frac{t_i^2 - t_{i-1}^2}{(c^2+t_i^2) \times t_{i-1}^2}.
\end{aligned}$$

To derive the optimal t_i values, the partial derivative of the expression above w.r.t. each t_i must be zero. Writing the partial derivative w.r.t. t_i and setting it to zero and simplifying yields

$$\frac{c^2}{t_i^2} = \frac{\sqrt{(t_{i-1}^2+c^2)(t_{i+1}^2+c^2)} - t_{i-1}t_{i+1}}{t_{i-1}t_{i+1}}. \tag{31}$$

Now, we want to figure out what these solvers will look like when $\lambda \to 0$. Deriving what a single step of the solver from noise level b to a will be gives the following update rule

$$\mathbf{x}_a = \left(\frac{a}{b}\right)^{\lambda^2+1} \mathbf{x}_b + \mathbf{x}_\theta(\mathbf{x}_b, b) \left(1 - (\frac{a}{b})^{\lambda^2+1}\right) + a\sqrt{1 - (\frac{a}{b})^{2\lambda^2}}\, \mathbf{z}_b, \tag{32}$$

where $\mathbf{z}_b \sim \mathcal{N}(0, \mathbf{I})$. In the limit when $\lambda \to 0$, this reduces to one step of forward Euler on the probability flow ODE. Using our assumption that the KLUB-optimal schedule is KL-optimal, when $\lambda \to 0$ the same sampling schedule is KL-optimal

for all λ, and the update rule gets closer and closer to the forward Euler update rule. As such, this schedule must also be KL-optimal for forward Euler as well.

At this point, we have explicit expression for both the KLUB optimal schedule and forward Eulers's KL-optimal schedules from Eqs. (22) and (32).

$$\text{KL optimal schedule}: \; t_i = \frac{(t_{i-1}t_{i+1} - c^2) + \sqrt{(t_{i-1}^2 + c^2)(t_{i+1}^2 + c^2)}}{t_{i-1} + t_{i+1}}$$

$$\text{KLUB optimal schedule}: \; t_i = c \times \sqrt{\frac{t_{i-1}t_{i+1}}{\sqrt{(t_{i-1}^2 + c^2)(t_{i+1}^2 + c^2)} - t_{i-1}t_{i+1}}}$$

A simple comparison between these two equations makes it clear they are not the same, which is a contradiction, disproving our initial assumption.

Therefore, using the KLUB objective to optimize the schedule does not translate directly into minimizing the mismatch between final output distributions. This is expected, given that the objective measures the divergence in the path distributions of the two SDEs, which is an upper bound for the mismatch between output distributions. As a result, optimizing the schedule with this objective focuses on aligning the trajectories of the two SDEs, rather than their end states.

Empirically, we've found that the heuristic schedules commonly in use are extremely sub-optimal, affecting both output and path distributions. Optimizing these schedules using the KLUB objective aligns the solver's paths with the true paths of the exact SDE. Initially, this alignment process also brings the output distributions closer together. However, after a certain point, the process begins to favor the alignment of intermediate noised distributions at the expense of the final output distributions, leading to more closely aligned paths. Therefore, early stopping must be used to prevent this from happening.

A.4. Exact Error Analysis

In Sec. 3.2 we assumed the learnt model is very close to the ideal denoiser, which let us approximate $P^{\text{true paths}}$ with the path distribution from the forward noising process. In this section, we provide a detailed analysis without that assumption, deriving an accurate KLUB that includes the model's approximation error. We start with the following assumption:

Assumption A.1 (score estimation error)**.** Letting D^* be the ideal denoiser, for all $t \in [t_{min}, t_{max}]$ the score estimation error is bounded:

$$\mathbb{E}_{\mathbf{x}_t \sim p'(\mathbf{x},t)} \left\| \mathbf{D}^* \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) \right\|^2 \leq \epsilon^2.$$

Using this, we apply Theorem 3.2 to calculate the KLUB between the true reverse SDE, which contains the ideal denoiser D^*, and the discretized linear SDE. Let $P^{\text{exact}}_{t_i \to t_{i-1}}$ represent the path distributions of the exact revere-time SDE, which matches the paths of the forward noising process. Then we have:

$$\begin{aligned}
D_{\text{KL}}(P^{\text{exact}}_{t_i \to t_{i-1}} \| P^{\text{disc}}_{t_i \to t_{i-1}}) &\leq 2 \times \mathbb{E}_{P^{\text{exact}}_{t_i \to t_{i-1}}} \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \left\| \mathbf{D}^* \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i) \right) \right\|^2 dt \\
&= 2 \times \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \times \mathbb{E}_{\substack{\mathbf{x}_t \sim p'(\mathbf{x},t) \\ \mathbf{x}_{t_i} \sim p'(\mathbf{x}_{t_i}|\mathbf{x}_t)}} \left(\left\| \mathbf{D}^* \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i) \right) \right\|^2 \right) dt \\
\leq 4 \times \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \times \mathbb{E}_{\substack{\mathbf{x}_t \sim p'(\mathbf{x},t) \\ \mathbf{x}_{t_i} \sim p'(\mathbf{x}_{t_i}|\mathbf{x}_t)}} &\left(\left\| \mathbf{D}^* \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) \right\|^2 + \left\| \mathbf{D}_\theta \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i) \right) \right\|^2 \right) dt \\
&\leq 4 \times \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \times \mathbb{E}_{\substack{\mathbf{x}_t \sim p'(\mathbf{x},t) \\ \mathbf{x}_{t_i} \sim p'(\mathbf{x}_{t_i}|\mathbf{x}_t)}} \left(\epsilon^2 + \left\| \mathbf{D}_\theta \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i) \right) \right\|^2 \right) dt \\
= \underbrace{O(\epsilon^2)}_{\text{Approximation error}} &+ \underbrace{4 \times \int_{t_{i-1}}^{t_i} \frac{s(t)^2 \dot{\sigma}(t)}{\sigma(t)^3} \times \mathbb{E}_{\substack{\mathbf{x}_t \sim p'(\mathbf{x},t) \\ \mathbf{x}_{t_i} \sim p'(\mathbf{x}_{t_i}|\mathbf{x}_t)}} \left(\left\| \mathbf{D}_\theta \left(\frac{\mathbf{x}_t}{s(t)}, \sigma(t) \right) - \mathbf{D}_\theta \left(\frac{\mathbf{x}_{t_i}}{s(t_i)}, \sigma(t_i) \right) \right\|^2 \right) dt}_{\text{Discretization error}}.
\end{aligned}$$

This means in the exact case, the KLUB can be broken into 2 parts, namely an approximation error and a discretization error. The approximation error relies solely on the model, and can only be improved by training the model further. Therefore, we can ignore it and focus on minimizing the discretization error, which leads to the KLUB objective derived in Sec. 3.2.

B. Experiment Details

B.1. Practical Implementation Details

As mentioned in Sec. 3.3, in practice to optimize a schedule for a given model and dataset, first a 10-step schedule $(t_0, t_1, \ldots, t_{10})$ is initialized using one of the baseline hand-crafted schedules. For continuous-time models, we initialize the schedule according to the EDM scheme, and for discrete-time models, time-uniform initialization is used.

Afterwards, the schedule is optimized in a hierarchical manner. This is done by first optimizing all the 9 intermediate points $(t_1, t_2, \ldots, t_9)$ of the schedule iteratively and using an early-stopping mechanism to avoid over-fitting. Perceptual image quality metrics or even manual inspection can be used as proxies to determine the stopping point of the optimization. Next, for 2 rounds, a subdivision operation is done that doubles the number of steps of the schedule, and further fine-tuning is performed on the newly added intermediate points, keeping the initial previous-round points fixed. These fine-tuning stages do not need early stopping due to the fixed "shape" of the schedule from the first-round optimization. The main reason behind using hierarchical optimization is to speed up training which is due to two factors. First, when optimizing a specific point t_i of a schedule, the optimized value will always remain inside $[t_{i-1}, t_{i+1}]$. As a result, if instead of hierarchical optimization, we optimized a 40-step schedule directly, the changes of each point would be smaller (due to the tighter $[t_{i-1}, t_{i+1}]$ intervals), resulting in more iterations to converge. Secondly, after each subdivision, only half of the points of a schedule are being optimized and these points are non-adjacent, making each point's optimization independent of the others. This allows the entire process to be parallelized. Furthermore, since during the later stages each point being optimized lies in a fixed interval (its endpoints are frozen), a very small number of iterations is required for it to converge.

To optimize the i-th element t_i, a number of candidate values are chosen in a neighbourhood around t_i, and for each the KLUB value is estimated with Monte-Carlo integration. Finally, t_i is set to be the candidate with the minimum KLUB value. In practice, we select 11 candidates with the current t_i value being one of them to ensure the KLUB is always decreasing. A pseudocode for this is given in Algorithm 1. We also experimented with having t_i being learnable parameters that are differentially optimized with respect to the KLUB loss term. We tried two different scenarios where t_i's are all optimized simultaneously or iteratively. In our experiments, we found this approach to be extremely unstable, requiring heavy fine-tuning of hyperparameters as well as a large effective batch size to smooth the gradient estimates. The large batch size also resulted in very slow optimization. As a result, we opted for the zeroth-order optimization approach, which does not rely on noisy gradient estimates. This optimization is relatively low-dimensional, consisting of only a small set of time steps that need to be adjusted, and zeroth-order optimization can work well in such settings.

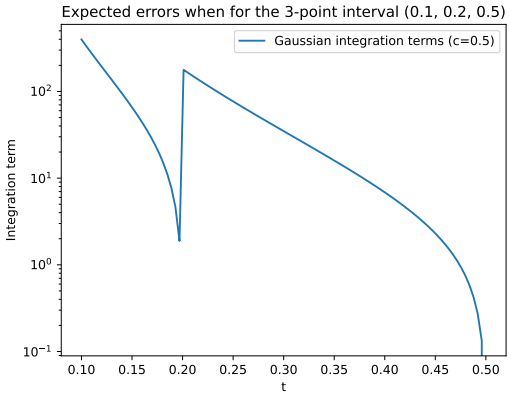


Figure 10. The figure illustrates the integration values of the KLUB in a Gaussian data setting within the interval $(0.1, 0.5)$ assuming a schedule of $(t_{i-1}, t_i, t_{i+1}) = (0.1, 0.2, 0.5)$, highlighting a large range of values and a discontinuity at 0.2.

To perform the Monte-Carlo integration, as discussed in Sec. 3.3, importance sampling is used. As mentioned previously, this is due to the integration values of the KLUB in Eq. (10) varying greatly in size. For example, we visualize these integration

values for $t \in [t_{i-1}, t_{i+1}]$ where $(t_{i-1}, t_i, t_{i+1}) = (0.1, 0.2, 0.5)$ in Fig. 10. The discontinuity at $t = 0.2$ highlights the point where the integrand values change from $(\frac{1}{t^2+c^2} - \frac{1}{0.2^2+c^2})$ for $t \in (0.1, 0.2]$ to $(\frac{1}{t^2+c^2} - \frac{1}{0.5^2+c^2})$ for $t \in (0.2, 0.5]$. As can be seen, in this example, the values range from $[0 - 1000]$ spanning roughly 3 orders of magnitude. A pseudocode is given in Algorithm 2.

In practice, we found using a subset of 8192 data samples with the time-based importance sampling to work well, and this configuration is used in all our experiments. Although the subset of samples used for estimating the KLUB isn't comprehensive, this is large enough to capture the essence of the data set and the optimized schedules generalize to the entire dataset. One interpretation of this could be that hand-designed schedules are so far from optimal that even optimizing them with respect to a small set of samples from the distribution gives substantial improvements. Moreover, note that we are optimizing a sampling schedule consisting of only a handful of timesteps. This represents a rather low-dimensional optimization problem, which may not require a large training datasets (in contrast, for instance, to the very high-dimensional optimization problem of training a large neural network from data).

Finally, the optimization time needed for different models depends heavily on how resource-heavy the model is because of the Monte-Carlo integration. However, in practice, each optimization only required at most 300 iterations. In our experiments, we used RTX6000 GPUs to carry out the optimization. The FFHQ and CIFAR10 experiments required 4 GPUs for 1.5 hours. The ImageNet 256x256 and text-to-image experiments were done with 8 GPUs and took roughly 3-4 hours. Lastly, the Stable Video Diffusion experiments were done with 16 GPUs and took 6 hours. It is worth noting that since the majority of training time was spent on Monte-Carlo integration and forward passing through the score network, increasing the number of GPUs linearly would almost linearly decrease the amount of time spent.

Algorithm 1 KLUB optimization with $\sigma(t) = t$ and $s(t) = 1$.

Input: denoiser $\mathbf{D}_\theta(\mathbf{x}, \sigma)$, schedule $t_{i \in \{0,1,\ldots,n\}}$
repeat
  Initialize $noChange =$ True
  for $i = 1$ **to** $n - 1$ **do**
    $candidates[0, \ldots, r-1] \leftarrow$ Neighbourhood around t_i
    for $j = 0$ **to** $r - 1$ **do**
      $KLUB[j] \leftarrow$ EstimateKLUB($\mathbf{D}_\theta$, $\{t_{i-1}, candidates_j, t_{i+1}\}$)
    end for
    minIdx $\leftarrow \arg\min KLUB[0, \ldots, r-1]$
    if $candidate_{\text{minIdx}} \neq t_i$ **then**
      $t_i \leftarrow candidate_{\text{minIdx}}$
      $noChange \leftarrow$ False
    end if
  end for
until $noChange$

Algorithm 2 Monte Carlo estimation of KLUB with $\sigma(t) = t$ and $s(t) = 1$.

Input: denoiser $\mathbf{D}_\theta(\mathbf{x}, \sigma)$, interval points t_{min}, t_{mid}, t_{max}, monte carlo samples n
for $i = 1$ **to** n **do**
  sample $\mathbf{x}_0 \sim p_{data}(\mathbf{x})$
  $t \leftarrow ImportanceSample(\pi, t_{min}, t_{mid}, t_{max})$
  $t_{upper} \leftarrow (t < t_{mid})\ ?\ t_{mid} : t_{max}$
  $\mathbf{x}_t \leftarrow \mathbf{x}_0 + t \times \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_{t_{upper}} \leftarrow \mathbf{x}_t + \sqrt{t_{upper}^2 - t^2} \times \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $KLUB[i] \leftarrow ||\mathbf{D}_\theta(\mathbf{x}_t, t) - \mathbf{D}_\theta(\mathbf{x}_{t_{upper}}, t_{upper})||^2 / (\frac{1}{t^2+c^2} - \frac{1}{t_{upper}^2+c^2})$
end for
$t_{upper} \leftarrow (t < t_{mid})\ ?\ t_{mid} :$
  t_{max}
return mean($KLUB[0, \ldots, n-1]$)

B.2. Popular Sampling Schedules

Currently, most diffusion models use one of a handful of different hand-designed sampling schedules at inference. Below we go over some of the most popular ones.

EDM Schedule: This schedule first introduced by (Karras et al., 2022) chooses the sampling schedule as follows:

$$\sigma(t_i) = (\sigma_{min}^{\frac{1}{\rho}} + (\sigma_{max}^{\frac{1}{\rho}} - \sigma_{min}^{\frac{1}{\rho}}) \times \frac{i}{n})^{\rho},$$

where $\rho = 7$ is usually used.

LogSNR schedule: This schedule is a special case of EDM's schedule where $\rho = 1$. Specifically:

$$\sigma(t_i) = (\sigma_{min} + (\sigma_{max} - \sigma_{min}) \times \frac{i}{n}).$$

Time-uniform schedule: This schedule is mainly used in discrete models. In these cases, the sampling schedule is simply:

$$t_i = \epsilon + \frac{i}{n}(1 - \epsilon).$$

In this case, the schedule will mimic the noise schedule with which the model was trained.

B.3. Optimized Schedules for Large Scale Models

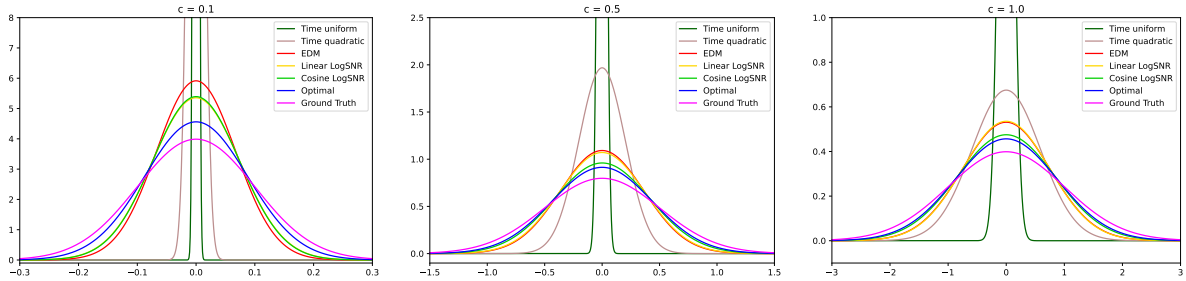
We provide our optimized schedules for Stable Diffusion 1.5, SDXL, DeepFloyd-IF, and Stable Video Diffusion in Table 3. The values in the table are the noise levels for the different steps i.e. $\sigma(t_i)$.

Table 3. Optimized schedules. The values represent the noise levels of the schedule $\sigma(t_n), \sigma(t_{n-1}), \ldots, \sigma(t_0)$.

	Optimized Schedules
Stable Diffusion 1.5 (Rombach et al., 2021)	[14.615, 6.475, 3.861, 2.697, 1.886, 1.396, 0.963, 0.652, 0.399, 0.152, 0.029]
SDXL (Podell et al., 2023)	[14.615, 6.315, 3.771, 2.181, 1.342, 0.862, 0.555, 0.380, 0.234, 0.113, 0.029]
DeepFloyd-IF / Stage 1 (Dee, 2023)	[160.41, 8.081, 3.315, 1.885, 1.207, 0.785, 0.553, 0.293, 0.186, 0.030, 0.006]
Stable Video Diffusion (Blattmann et al., 2023a)	[700.00, 54.5, 15.886, 7.977, 4.248, 1.789, 0.981, 0.403, 0.173, 0.034, 0.002]

C. Additional Results

C.1. Gaussian Data Extras



(a) $c = 0.1$ (b) $c = 0.5$ (c) $c = 1.0$

Figure 11. Comparing the output distributions of using various schedules for different initial c values in the Gaussian setting.

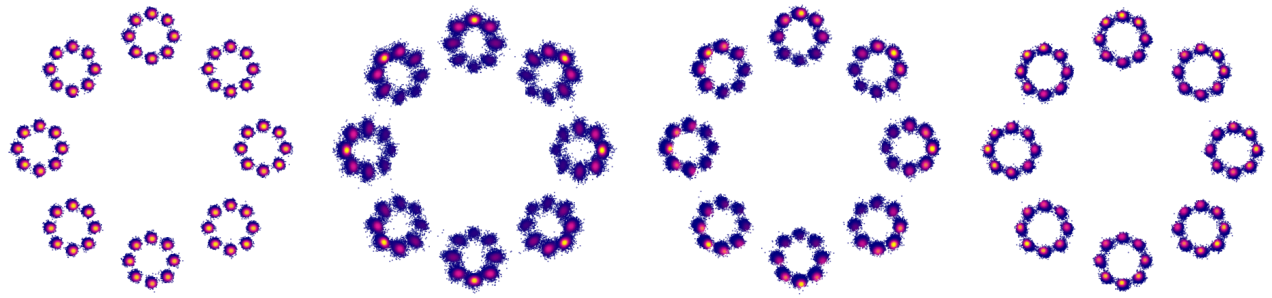
C.2. Extra 2D Experiments

In this section, we provide extra experiments for various 2D toy data. First we consider a set of datasets for which we know the ground truth score analytically, i.e. a mixture of gaussians. We consider 3 different variants of mixture of gaussians and

show samples generated with various samplers using the EDM, LogSNR, and optimized schedules in Figs. 12 to 14. We also report negative log-likelihoods (NLL) for these samples in Table 4.

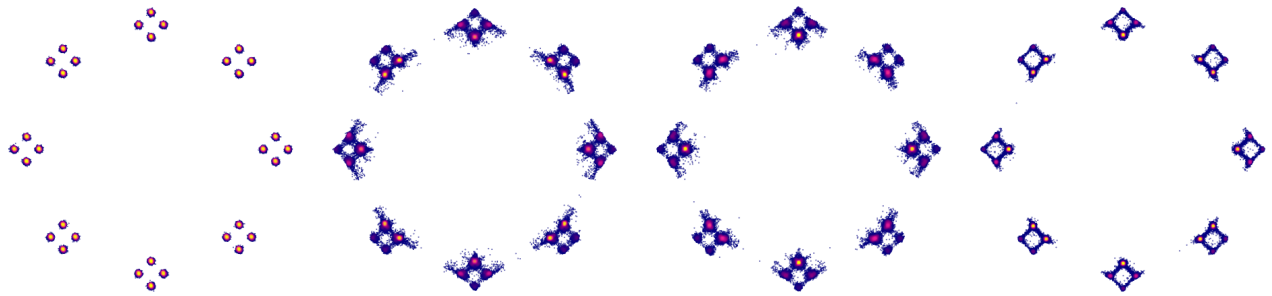
We further consider two more complex 2D distributions, and use a continuous-time EDM-based diffusion model to learn the score. For these datasets, we train the score model for 100,000 steps with a batch size of 8092. Figs. 15 and 16 showcase samples drawn from these models using different schedules.

For these 2D toy data experiments, the colors in Figs. 12 to 16 denote the local density of the samples where warmer colors correspond to higher density regions. The density is obtained with a 2D histogram with 50 bins on each axis. All experiments are done in the unconditional generation setting and do not involve any class labels.



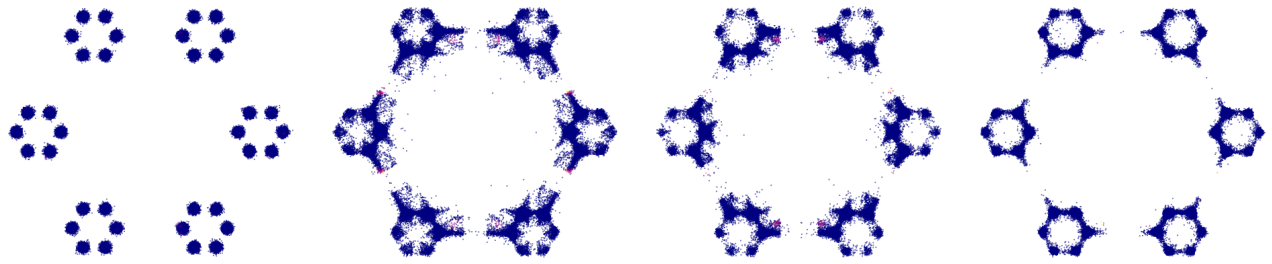
(a). Ground truth *(b).* EDM *(c).* LogSNR *(d).* AYS

Figure 12. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 10 steps of SDE-DPM-Solver++(2M) with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points.



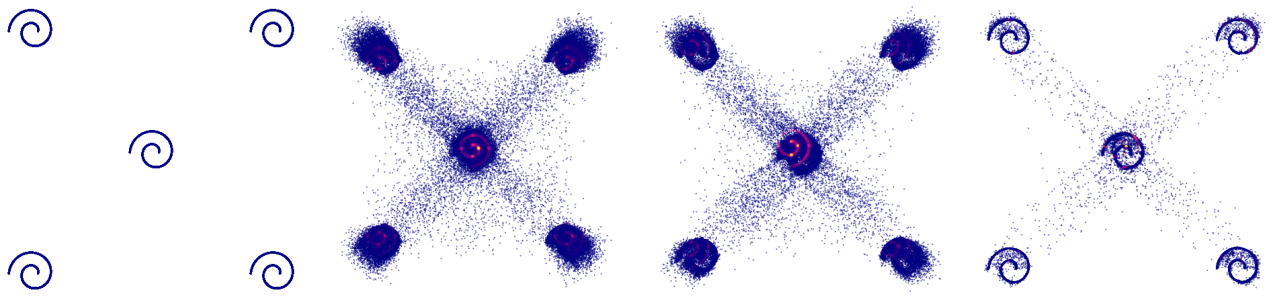
(a). Ground truth *(b).* EDM *(c).* LogSNR *(d).* AYS

Figure 13. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 7 steps of DDIM with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points.



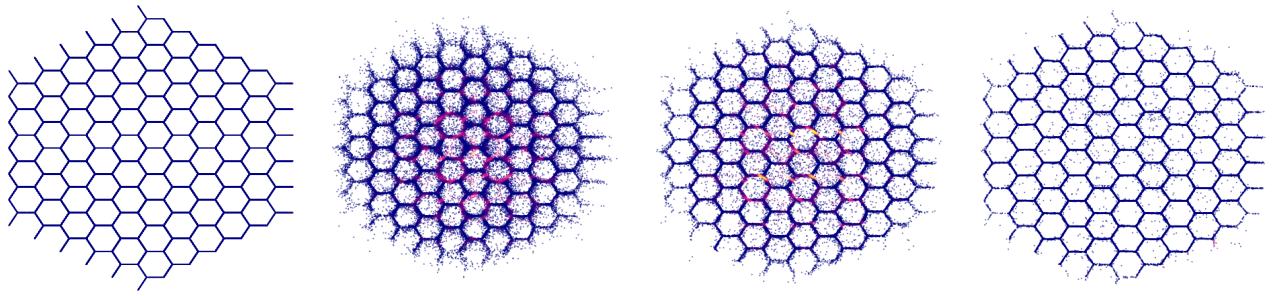
(a). Ground truth *(b).* EDM *(c).* LogSNR *(d).* AYS

Figure 14. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 6 steps of Stochastic-DDIM with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points.



(a). Ground truth *(b).* EDM *(c).* LogSNR *(d).* AYS

Figure 15. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 6 steps of SDE-DPM-Solver++(2M) with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points.



(a). Ground truth *(b).* EDM *(c).* LogSNR *(d).* AYS

Figure 16. **Modeling a 2D toy distribution**: (a) Ground truth samples; (b), (c), and (d) are samples generated using 8 steps of SDE-DPM-Solver++(2M) with EDM, LogSNR, and AYS schedules, respectively. Each image consists of 100,000 sampled points.

C.3. CIFAR10, FFHQ, and ImageNet Details

For these experiments, we generate 50,000 images to perform the evaluations. For the continuous-time models, i.e. the CIFAR10 and FFHQ experiments, we use the FID calculation script and reference statistics from (Karras et al., 2022). For the ImageNet results, we use the evaluation script from (Dhariwal & Nichol, 2021).

We provide more comprehensive results for CIFAR10 and FFHQ in Tables 5 and 6 respectively. As can be seen from the

Table 4. Performance (measures in negative log likelihood) for mixture of gaussian data and varying solvers, schedules, and number of steps.

Dataset	Solver	Schedule	NFE=6	NFE=8	NFE=10
Gaussian mixture 8x8	SDE-DPM-Solver++(2M)	EDM	9.018	4.029	1.522
		LogSNR	6.250	1.834	0.071
		AYS	**-0.143**	**-0.505**	**-0.574**
Gaussian mixture 8x4	DDIM	EDM	1.536	-0.144	-1.115
		LogSNR	1.446	-0.288	-1.091
		AYS	**-1.999**	**-2.260**	**-2.222**
Gaussian mixture 6x6	Stochastic-DDIM	EDM	1.166	-0.606	-0.996
		LogSNR	-0.012	-0.978	-1.231
		AYS	**-1.152**	**-1.376**	**-1.554**

results, the schedules optimized using the KLUB derived for Stochastic-DDIM generalize well to all stochastic solvers. This trend continues to ODE solvers as well, and KLUB-optimized schedules improve results on the first-order DDIM and the multi-step second-order DPM-Solver++(2M) as well.

Table 5. Sample fidelity measured by FID ↓ on the CIFAR10 32×32 unconditional dataset.

	Sampling method	Schedule	NFE=10	NFE=20	NFE=30	NFE=50
Stochastic Sampling	Stochastic DDIM	EDM	51.45	23.67	14.19	7.75
		AYS	**33.52**	**14.16**	**8.78**	**5.45**
	SDE-DPM-Solver++ (2M)	EDM	15.32	4.64	3.15	2.64
		AYS	**8.16**	**3.23**	**2.55**	**2.40**
	ER-SDE-Solver 1	EDM	17.97	6.70	4.31	3.02
		AYS	**12.93**	**5.09**	**3.50**	**2.66**
	ER-SDE-Solver 2	EDM	9.92	3.33	2.48	2.16
		AYS	**7.77**	**3.14**	**2.40**	**2.14**
	ER-SDE-Solver 3	EDM	9.47	3.15	2.39	2.13
		AYS	**7.55**	**3.07**	**2.36**	2.13
Deterministic Solvers	DDIM	LogSNR	16.44	6.01	3.97	2.82
		AYS	**10.73**	**4.67**	**3.30**	**2.56**
	DPM-Solver++ (2M)	LogSNR	5.07	2.37	2.12	2.04
		AYS	**2.98**	**2.10**	**2.02**	**2.01**

Table 6. Sample fidelity measured by FID ↓ on the FFHQ 64×64 dataset.

	Sampling method	Schedule	NFE=10	NFE=20	NFE=30	NFE=50
Stochastic Sampling	Stochastic DDIM	EDM	53.83	31.97	22.14	13.42
		AYS	**42.03**	**22.73**	**14.90**	**9.135**
	SDE-DPM-Solver++ (2M)	EDM	23.04	9.67	5.96	3.85
		AYS	**14.79**	**5.65**	**3.97**	**3.13**
	ER-SDE-Solver 1	EDM	21.25	9.29	6.24	4.28
		AYS	**15.27**	**7.09**	**4.88**	**3.68**
	ER-SDE-Solver 2	EDM	12.51	4.49	3.23	2.68
		AYS	**9.04**	**4.04**	**3.03**	2.68
	ER-SDE-Solver 3	EDM	11.97	4.18	3.06	**2.61**
		AYS	**8.71**	**3.92**	**2.97**	2.65
Deterministic Solvers	DDIM	EDM	18.37	8.19	5.60	3.96
		AYS	**12.83**	**6.05**	**4.41**	**3.38**
	DPM-Solver++ (2M)	LogSNR	7.07	3.41	2.87	2.62
		AYS	**5.43**	**3.29**	2.87	2.62

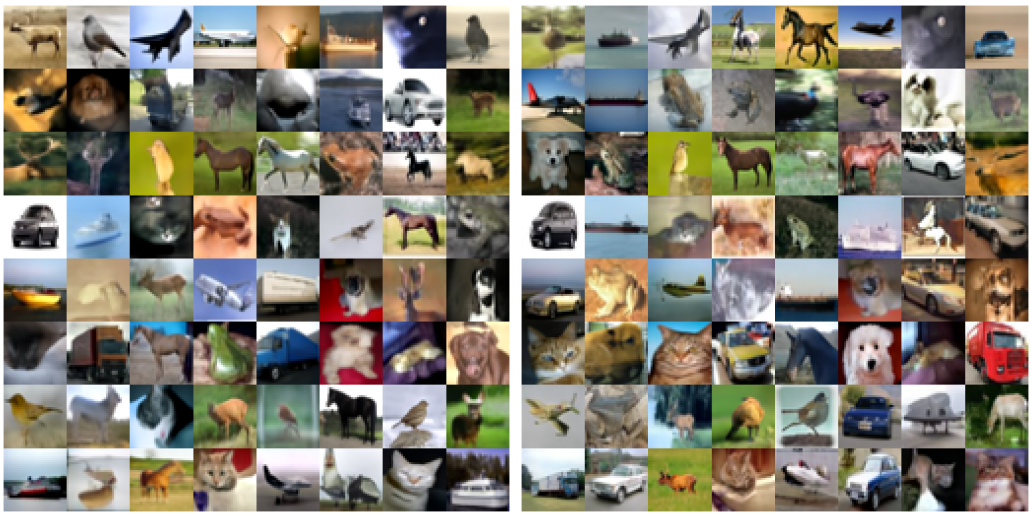


Figure 17. EDM Schedule

Figure 18. AYS Schedule

Figure 19. Side-by-side comparisons for CIFAR10 with EDM and AYS schedules. Samples are generated using 10 steps with the SDE-DPM-Solver++(2M) solver.

C.4. Comparison with Watson et al.

In this section, we compare our method against the one proposed by (Watson et al., 2022) on the unconditional ImageNet 64×64 dataset. Their approach works by formulating the weights of a multi-step solver and the sampling schedule as trainable parameters, and optimizing them by differentiating through Kernel Inception Distance (KID) as a perceptual loss. Note that this is only applicable to image diffusion models, and cannot be generally used for other data types. Furthermore, it is not clear how their method affects the diversity of samples, due to them directly optimizing the denoising variance to only increase the image quality.

The authors tested their method against DDIM with standard schedules (time-uniform and time-quadratic). For their experiments, they trained a DDPM following (Nichol & Dhariwal, 2021) with their L_{hybrid} objective for 3M steps. In contrast, we use the publicly available checkpoint, which was originally trained for 1.5M steps. For evaluation, the evaluation script from (Dhariwal & Nichol, 2021) is used.

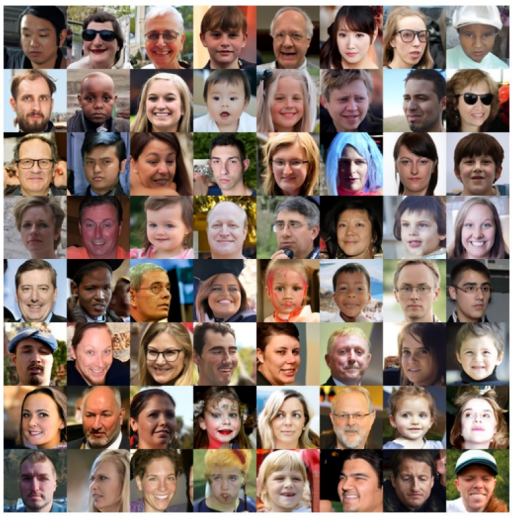


Figure 20. EDM Schedule

Figure 21. AYS Schedule

Figure 22. Side-by-side comparisons for FFHQ with EDM and AYS schedules.Samples are generated using 10 steps with the DPM-Solver++(2M) solver.

Table 7 summarizes the results. The numbers show that, despite using a better diffusion model trained for twice as many steps and optimizing the sampler itself, our optimized schedules alone outperform theirs in extremely low NFE regimes. However, as NFE increases, the influence of the schedules diminishes, causing the better diffusion model to gain the upper hand. Nevertheless, when comparing the improvements over the baseline time-uniform schedule, our performance, in terms of FID reduction, is on par with theirs.

Table 7. Image quality measured by FID ↓ / Inception Score ↑ on the unconditional ImageNet 64×64 dataset.

Model	Sampler	Schedule	NFE=5	NFE=10	NFE=15	NFE=20	NFE=25
3M steps	DDIM	Time-uniform	135.4 / 5.898	40.70 / 12.225	28.54 / 13.99	24.225 / 14.75	22.13 / 15.16
	DDIM	Time-quadratic	409.1 / 1.380	148.6 / 5.533	67.65 / 9.842	45.60 / 11.99	36.11 / 13.225
	GGDM +PRED	Optimized Schedule	**55.14 / 12.90**	**37.32 / 14.76**	**24.69 / 17.225**	**20.69 /17.92**	**18.40 / 18.12**
1.5M steps	DDIM	Time-uniform	145.01 / 5.45	42.51 / 11.25	30.32 / 12.89	26.60 / 13.57	24.77 / 14.00
	DDIM	AYS	**50.38 / 11.08**	**29.23 / 13.64**	**24.21 / 14.24**	**22.26 / 14.62**	**21.42 / 14.80**

C.5. Text-to-Image Extras

For these models that rely heavily on classifier-free guidance, each guidance value changes the models outputs, and can be seen as its own model. As such, it would be ideal to optimize the schedule for each guidance value. However, to keep things simple, we opt to only optimize the schedule using a default guidance value, and use the same schedule for all guidance weights in these results.

We made use of the COCO (Lin et al., 2014) dataset to optimize the schedule for the text-to-image models. We used a subset of 10,000 images for this task, and excluded these images during FID evaluation. Figs. 23 and 24 represent FID vs. CLIP score pareto curves for Stable Diffusion 1.5 and SDXL respectively.

Interestingly, our optimized SD 1.5 schedule also generalizes and improves images for several personalized text-to-image models based on Stable Diffusion 1.4/1.5. Figs. 25 to 27 show some side-by-side comparisons for these models. Please visit our project page for additional qualitative examples.

To quantitatively evaluate the effectiveness of different schedules we performed a user study. See results in Fig. 9. This study involved 42 participants and 600 distinct prompts. For each prompt, three images were generated using EDM, Time

Uniform, and AYS schedules using SDE-DPM-Solver++(2M) with 10 steps. Participants were asked to choose the best image in terms of fidelity and text alignment. The results, shown in Fig. 9, reveal a clear preference for the optimized schedule.

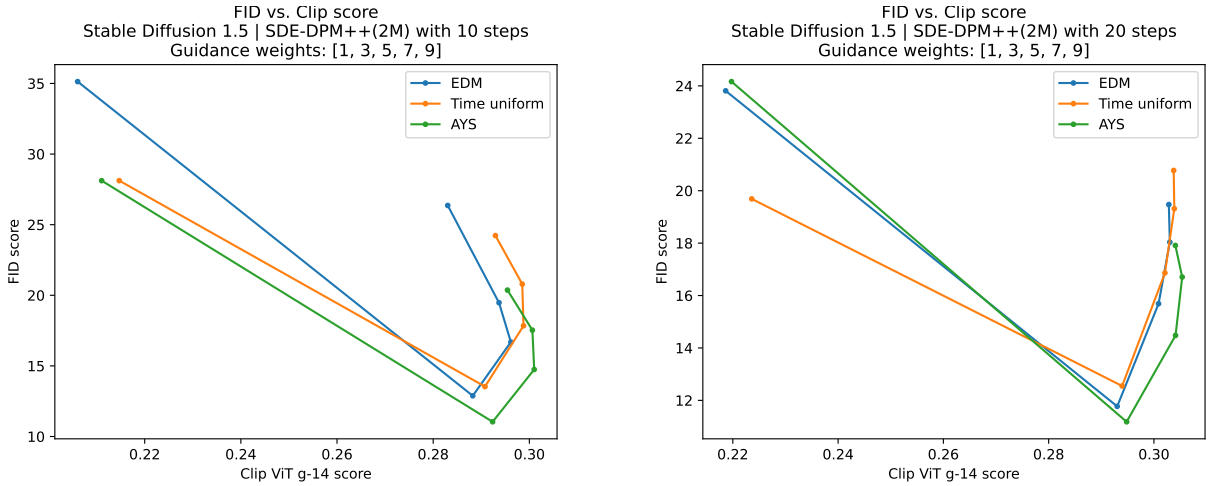


Figure 23. Plotting FID vs. CLIP scores for different classifier-free guidance weights for Stable Diffusion 1.5 using SDE-DPM-Solver++(2M) with 10 and 20 steps.

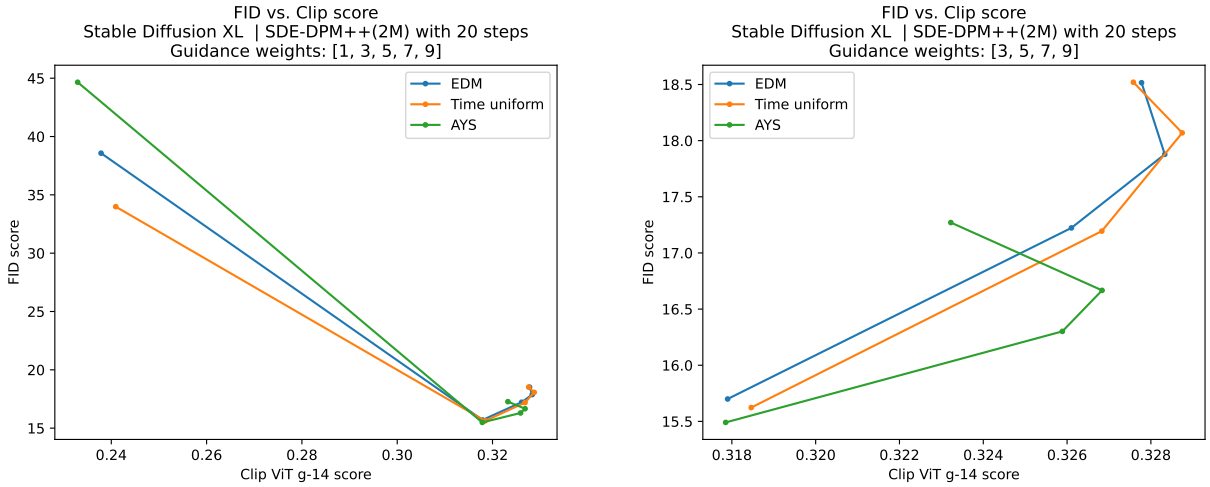


Figure 24. Plotting FID vs. CLIP scores for different classifier-free guidance weights for SDXL using SDE-DPM-Solver++(2M) with 20 steps. The image on the right is a zoomed in version of the left without the left most point corresponding to no guidance.

Figure 25. SD 1.5 + 10 steps + SDE-DPM-Solver++(2M)

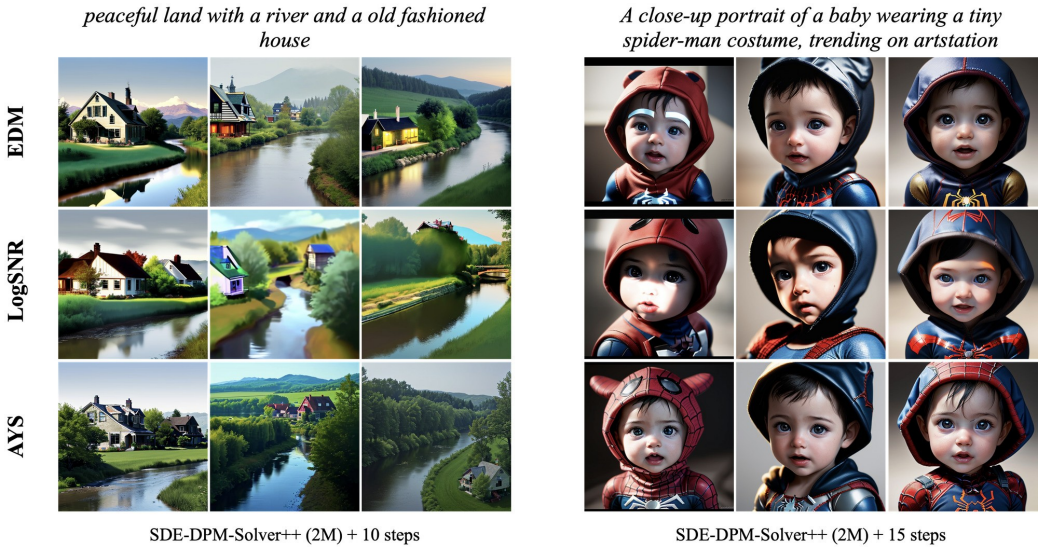


Figure 26. DreamShaper + SDE-DPM-Solver++(2M)

C.6. Stable Video Diffusion Details

For the video diffusion experiments, we used the validation subset of the WebVid10M dataset (Bain et al., 2021) to optimize the schedule. This subset contains 5,000 videos from the internet and we downsampled each to a resolution of 320×576. Given the unclear nature as to how other inputs to the model besides the first frame were obtained during the training of SVD, such as motion bucket id and noise augmentation strength, we simply set them to default values in our experiments. Note that this is extremely sub-optimal, as the model was not trained in this way, however it still produced visible benefits in our experiments.

We also do a user-study on the generated videos. For this, we asked ChatGPT for 150 visually interesting prompts. Afterwards, we used DALLE3 and SDXL to generate 150 images from these prompts. These images will act as the first frames of our generated videos. For each image and schedule, we generated 4 videos, resulting in 1200 videos, 600 using EDM and 600 using the optimized schedule. These were shown to users and asked to identify the best one. Results are summarized in Table 2.

Figure 27. RealisticVision 5.1 + 10 steps + SDE-DPM-Solver++(2M)

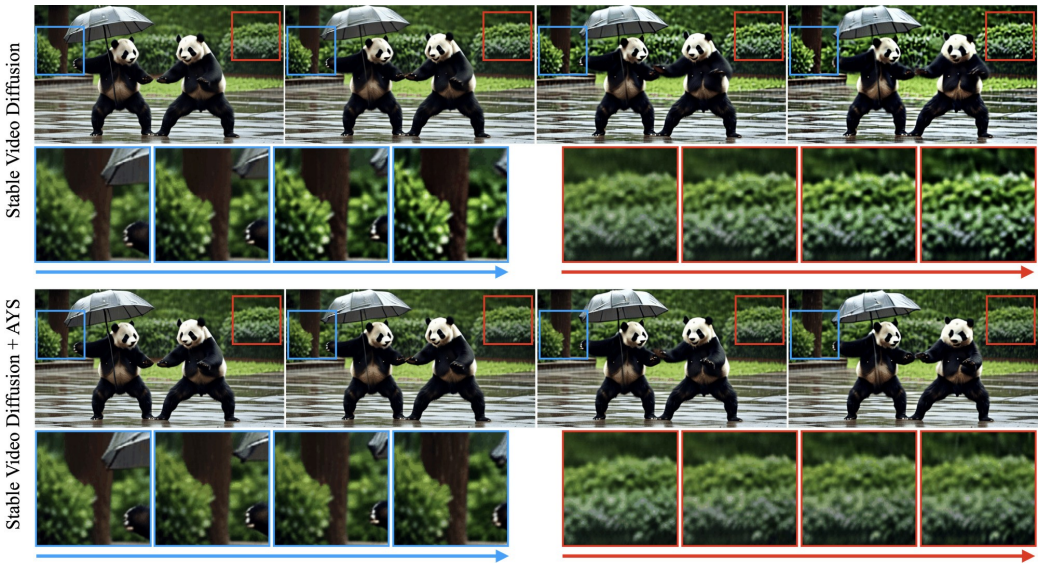


Figure 28. Side-by-side comparisons for Stable Video Diffusion (Blattmann et al., 2023a). Using the optimized schedule results in a more stable video; note the temporal color distortions of the background for the baseline.