Variational Trajectory Optimization of Anisotropic Diffusion Schedules

Pengxi Liu* **Zeyu Michael Li*** **Xiang Cheng**
Department of Electrical and Computer Engineering
Duke University
{pengxi.liu,zeyu.li030,xiang.cheng}@duke.edu

Abstract

We introduce a variational framework for diffusion models with anisotropic noise schedules parameterized by a matrix-valued path $M_t(\theta)$ that allocates noise across subspaces. Central to our framework is a trajectory-level objective that jointly trains the score network and *learns* $M_t(\theta)$, which encompasses general parameterization classes of matrix-valued noise schedules. We further derive an estimator for $\partial_\theta \nabla \log p_t$ that enables efficient optimization of the $M_t(\theta)$ schedule. For inference, we develop an efficiently-implementable reverse-ODE solver that is an anisotropic generalization of the second-order Heun discretization algorithm. Across CIFAR-10, AFHQv2, FFHQ, and ImageNet-64, our method consistently improves FID with largest gains at low NFE. Code is available at `https://github.com/kilonova801/anisotropic-diffusion`.

1 Introduction

Diffusion models generate samples by *reversing* a gradual Gaussian noising process. Starting from data $x_0 \sim p_0$, a forward process produces a family of noisy variables $(x_t)_{t\geq 0}$ with density p_t that smoothly interpolates between the data distribution and a tractable Gaussian. A neural *score* model (parameters ϕ) is trained to approximate the score $\nabla_x \log p_t(x)$; given this score, sampling integrates a reverse-time SDE/ODE that transports p_T back to p_0 [1–3]. Nearly all modern diffusion pipelines assume *isotropic* forward noise, meaning that at time t, the covariance of total injected Gaussian noise is a scalar multiple of identity. We ask whether learning separate noise schedules for different data subspaces, jointly with the score network, can improve generation, especially under limited sampling budgets.

Matrix-valued (anisotropic) schedules. In this paper, we replace the scalar noise level by a *matrix-valued* covariance trajectory M_t and study how to *learn* it. Concretely, in the standard variance-exploding (VE) / Brownian formulation, the isotropic process adds noise with covariance tI,

$$x_0 \sim p_0, \quad dx_t = dB_t \quad \Leftrightarrow \quad dx_t = -\tfrac{1}{2}\nabla \log p_t(x_t)dt, \tag{1}$$

where B_t is standard Brownian motion. We generalize this to anisotropic diffusion by instead considering a forward diffusion driven by a *matrix-valued diffusion coefficient*:

$$x_0 \sim p_0, \quad dx_t = (\partial_t M_t)^{1/2} dB_t \quad \Leftrightarrow \quad dx_t = -\tfrac{1}{2}\partial_t M_t \nabla \log p_t(x_t)dt, \tag{2}$$

where $M_t(\theta)$ is a matrix-valued noise-schedule, with arbitrary parameterization θ, satisfying

$$M_0 = 0, \qquad \partial_t M_t \succ 0 \qquad \text{for all } t. \tag{3}$$

For example, let P_{low} and P_{high} be orthogonal projectors onto complementary low- and high-frequency subspaces. Then

$$M_t = g_{\text{low}}(t)P_{\text{low}} + g_{\text{high}}(t)P_{\text{high}}$$

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

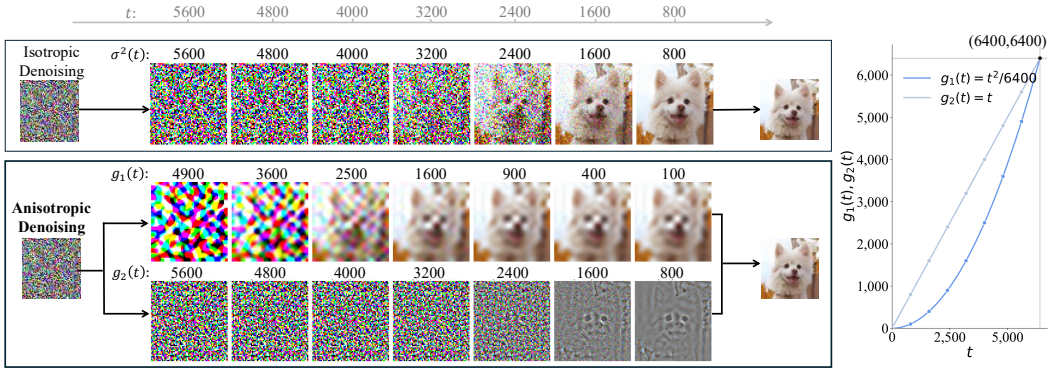

Figure 1: **Isotropic vs. anisotropic denoising.** Top: isotropic denoising. Bottom: anisotropic denoising with low- and high-frequency DCT subspaces (Section C). Columns schematically illustrate denoising as t decreases. Right: schematic schedules g_1, g_2 for the low- and high-frequency subspaces, respectively, illustrating how low-frequency structure can emerge before high-frequency detail. **Illustration only:** Both rows use the same dog image from AFHQv2 [9] to illustrate the different denoising patterns. The g_1–g_2 gap is exaggerated for clarity; the learned schedules exhibit smaller differences (cf. Fig. 2–5).

assigns separate noise variances to the two subspaces. When $g_{\text{low}}(t) = g_{\text{high}}(t)$, this reduces to an isotropic schedule. Our approach learns how these noise variances evolve over time.

The power of anisotropy. Replacing a scalar schedule by a matrix path allows noise (and thus denoising effort) to be allocated differently across directions and time, better matching data geometry. Natural images concentrate energy in low spatial frequencies [4]; latent diffusion effectively separates coarse structure from fine detail [5]; video models benefit from temporally structured priors or decomposed noise [6, 7]; and multi-resolution generation often proceeds coarse-to-fine [8]. Figure 1 illustrates one such effect: low-frequency content can be denoised earlier than high-frequency details when the schedule is allowed to differ across subspaces.

A variational formulation of matrix trajectory is necessary. The flexibility of M_t comes with a price: the design space is enormous. Instead of choosing a single scalar function, we must choose a trajectory in the cone of PSD matrices. In the isotropic VE case, the family of marginals is essentially one-dimensional: $p_t = p_0 * \mathcal{N}(0, \sigma(t)^2 I)$ (where $*$ denotes convolution) depends on t only through the scalar variance $\sigma(t)^2$, so changing the schedule mainly *reparameterizes time* along the same curve of distributions. In contrast, different anisotropic paths $M_t(\theta)$ generally produce genuinely different marginals $p_t(\cdot;\theta)$ (even up to time reparameterization), and hence different score targets and reverse-time dynamics. This makes hand-crafted anisotropy brittle and motivates a principled, data-driven approach.

1.1 Main Contributions

We develop a variational framework for *learning* anisotropic diffusion trajectories $M_t(\theta)$ jointly with a score model (parameters ϕ), rather than prescribing M_t by hand. We summarize our technical and empirical contributions below.

(1) A general variational framework for learning anisotropic diffusion models.

1. We introduce a variational framework for anisotropic diffusion with *matrix-valued* noise path $M_t(\theta)$. Our framework is general and supports broad classes of $M_t(\theta)$, including interpretable frequency-band schedules and data-adaptive (class-conditional) PCA schedules (Section C).
2. We propose a trajectory-level score-matching objective $L(\theta, \phi)$ (Section 3) that jointly trains the score network and learns $M_t(\theta)$ by minimizing mismatch between ideal and learned denoising dynamics along the reverse trajectory (with a pathwise change-of-measure interpretation).
3. We derive anisotropic reverse-ODE solvers by generalizing Euler and second-order Heun updates to matrix trajectories (Section 5), yielding closed-form steps expressed through increments of

$M_t^{1/2}$. The projector-based structure further lets any scalar diffusion solver be lifted to the anisotropic setting; instantiating this with DPM-Solver-v3 yields the best FID at very low NFE (Appendix D.4).

(2) An efficient schedule-gradient estimator for matrix-valued trajectories. Optimizing $M_t(\theta)$ is challenging because changing θ changes the entire forward family $p_t(\cdot;\theta)$, and thus requires $\partial_\theta \nabla \log p_t(x;\theta)$, which is not available from a network trained at a single θ. We derive a plug-in estimator of $\partial_\theta \nabla \log p_t(x;\theta)$ using only higher-order x-*directional* derivatives of the network, implementable in **three backward passes** and independent of $\dim(\theta)$ (Theorem 4.1). We further introduce a *flow parameterization* that stabilizes scale across noise levels and reduces estimator variance (Section 4.2).

(3) Empirical validation of anisotropic trajectory learning. On CIFAR-10, AFHQv2, FFHQ, and ImageNet-64 (Section 6), learned anisotropic trajectories match or improve upon the EDM baseline under the standard evaluation protocol, with the most pronounced gains in the low-NFE regime. We further benchmark against specialized few-step samplers (AYS [10], DPM-Solver-v3 [11]) and find that our method, despite operating at the *training* rather than *solver* level, achieves competitive or better FID. The improvements persist across solver budgets and schedule families, supporting the view that *learning the matrix trajectory improves score-matching and the resulting generative model*.

1.2 Related Work

Structured forward processes extend diffusion beyond isotropic noise through blurring, linear corruptions, and non-isotropic Gaussian covariances [12–16]. Whitened Score Diffusion [17] addresses covariance inversion in anisotropic score matching through a reparameterized learning target. Learned forward processes include scalar schedules in VDM [18] and per-pixel, input-adaptive schedules in MuLAN [19]. Our focus is joint optimization of a matrix covariance trajectory and the score network using a trajectory-level objective, with structured subspace parameterizations for efficient training and sampling. Sampler-side methods such as DPM-Solver and AYS [20, 11, 10] provide complementary choices for numerical integration. Appendix A discusses these connections in more detail.

2 Preliminaries

2.1 Anisotropic Diffusion: Process, Score, and Parameterizations

Recall the anisotropic diffusion process in (2). Let $M_t(\theta)$ denote the noise covariance at time t, parameterized by θ. p_t as defined in (2) has score given by

$$\nabla \log p_t(x;\theta) = M_t^{-1}(\theta)\mathbb{E}_{x_0|x_t=x}\left[x_0 - x_t\right], \tag{4}$$

where (x_0, x_t) are defined by the joint distribution $x_0 \sim p_0$ and $x_t = x_0 + \mathcal{N}(0, M_t)$. We provide a short proof in Lemma B.1 in Appendix B. In case of time-uniform isotropic diffusion (i.e. standard Brownian Motion), $M_t(\theta) = tI$, and the formula in (4) reduces to the standard score expression.

We parameterize a neural network $\mathrm{net}(x, t, \phi)$ to approximate the score, we also define flow, a transformation of net whose norm is approximately time-invariant:

$$\mathrm{net}(x, t, \phi) \approx \nabla \log p_t(x;\theta) \tag{5}$$

$$\mathrm{flow}(x, t, \phi) := M_t^{1/2}\mathrm{net}(x, t, \phi) \tag{6}$$

Continuous and discrete reverse ODE for anisotropic denoising. Recall the *forward ODE dynamics* as defined in (2): $dx_t = -\frac{1}{2}\partial_t M_t \nabla \log p_t(x_t)dt$. Given net in (5), the learned denoising ODE is given by the time-reversal of

$$d\bar{x}_t = -\frac{1}{2}\partial_t M_t(\theta)\mathrm{net}(\bar{x}_t, t, \phi)dt. \tag{7}$$

Variance-preserving anisotropic diffusion. It is often more useful to consider the *variance preserving* anisotropic diffusion, which is simply a time-dependent linear-transformation of (2).

Define

$$x_t^{VP} := (I + M_t(\theta))^{-1/2} x_t. \tag{8}$$

The choice of I above is based on the assumption that $Cov(x_0) \approx I$, which implies that $Cov(x_t^{VP}) \approx I$ for all t. Combining (8) with (2), and using that $M_t(\theta)$ commutes with $\partial_t M_t(\theta)$ (as for all families in Appendix C), gives the following dynamics:

$$\begin{aligned} &dx_t^{VP} = v(x_t, t; \theta)dt, \qquad \text{where} \\ &v(x, t; \theta) := -\frac{1}{2}(I + M_t(\theta))^{-1/2}\partial_t M_t(\theta)\nabla \log p_t(x; \theta) - \frac{1}{2}(I + M_t(\theta))^{-3/2}\partial_t M_t(\theta)x, \end{aligned} \tag{9}$$

where $v(x, t; \theta)$ is the *ideal* velocity field. Analogously, we define the *learned* velocity field

$$\begin{aligned} &d\bar{x}_t^{VP} = \bar{v}(x_t, t; \theta)dt, \qquad \text{where} \\ &\bar{v}(x, t; \theta, \phi) := -\frac{1}{2}(I + M_t(\theta))^{-1/2}\partial_t M_t(\theta)\text{net}(x, t, \phi) - \frac{1}{2}(I + M_t(\theta))^{-3/2}\partial_t M_t(\theta)x, \end{aligned} \tag{10}$$

3 Trajectory-Level Score Matching Loss

This section introduces our central training objective (11), which enables us to **jointly** learn a **matrix-valued** noise path $\{M_t(\theta)\}_{t\in[0,T]}$ together with a **score network** $\text{net}(x, t, \phi)$. At a high level, the loss measures a **trajectory-weighted score error** induced by the schedule $M_t(\theta)$, and it is designed so that (for any fixed θ) the optimal network recovers the true score (Lemma 3.1).

3.1 Defining the Trajectory Loss

Loss definition. We define our trajectory-level training objective as

$$L(\theta, \phi) := \mathbb{E}_{t\sim\text{Unif}[0,T],\, x_0\sim p_0,\, \epsilon\sim\mathcal{N}(0,I)} \left[\left\| W_t(\theta) \left(M_t(\theta)^{1/2} \text{net}\Big(x_0 + M_t(\theta)^{1/2}\epsilon,\ t,\ \phi\Big) + \epsilon \right) \right\|_2^2 \right]$$
$$\text{where} \qquad W_t(\theta) := (I + M_t(\theta))^{-1/2}(\partial_t M_t(\theta))M_t(\theta)^{-1/2}. \tag{11}$$

In words: we draw a noise level t, perturb data by $x_t := x_0 + M_t(\theta)^{1/2}\epsilon$, evaluate the score network at (x_t, t), and penalize its deviation from the *noise target* $-\epsilon$, under a *matrix-valued* weight $W_t(\theta)$ determined by the trajectory $M_t(\theta)$. Since $x_t = x_0 + M_t(\theta)^{1/2}\epsilon$, we have $M_t(\theta)^{-1/2}\epsilon = -M_t(\theta)^{-1}(x_0 - x_t)$, so the network term inside (11) is trained to match the conditional denoising direction. The operator $W_t(\theta)$ serves as a *matrix-valued* weighting schedule that can emphasize different subspaces at different times, and it is exactly this dependence on $M_t(\theta)$ that allows us to *optimize the trajectory* θ.

Equivalence of loss to velocity mismatch norm. Recall from Section 2 that under the variance-preserving (VP) reparameterization (8), the *ideal* and *learned* VP velocity fields are $v(\cdot, t; \theta)$ and $\bar{v}(\cdot, t; \theta, \phi)$ as defined in (9) and (10). A natural trajectory-level principle is to choose (θ, ϕ) so that the learned denoising dynamics match the ideal ones along the reverse trajectory. The continuous-time form of this principle is the path integral

$$\int_0^T \mathbb{E}_{x_0\sim p_0,\, \epsilon\sim\mathcal{N}(0,I)} \left[\|\bar{v}(x_t, t; \theta, \phi) - \tilde{v}(x_t, x_0, t; \theta)\|_2^2 \right] dt, \qquad x_t = x_0 + M_t(\theta)^{1/2}\epsilon, \tag{12}$$

where $\tilde{v}(\cdot, x_0, t; \theta)$ is obtained from the ideal velocity $v(\cdot, t; \theta)$ by replacing the intractable score $\nabla \log p_t(\cdot; \theta)$ with the single-sample proxy $M_t(\theta)^{-1}(x_0 - x_t)$ (see the score identity (4)):

$$\tilde{v}(x_t, x_0, t; \theta) := -\frac{1}{2}(I + M_t(\theta))^{-1/2}\partial_t M_t(\theta)\, M_t(\theta)^{-1}(x_0 - x_t) - \frac{1}{2}(I + M_t(\theta))^{-3/2}\partial_t M_t(\theta)\, x_t. \tag{13}$$

Expanding $\bar{v}$ and $\tilde{v}$, the common contraction term $-\frac{1}{2}(I+M_t)^{-3/2}\partial_t M_t\, x_t$ cancels, and (12) reduces to the compact matrix-weighted score-matching form (11) (up to an overall constant factor and the standard Monte Carlo approximation of the time integral by sampling t).

3.2 Key Properties and Interpretation

Exact score at optimality. As standard in the literature [21, 2], we show that for any fixed schedule $M_t(\theta)$ satisfying (3), the optimal network for $L(\theta, \phi)$ recovers the true score in the infinite-data/infinite-capacity limit:

Lemma 3.1 (Exact score at optimality)**.** *Fix any schedule $M_t(\theta)$ satisfying* (3)*. In the limit of infinite data and model capacity, the minimizer $\phi^*(\theta) = \arg\min_\phi L(\theta, \phi)$ satisfies*

$$\mathrm{net}(x, t, \phi^*(\theta)) = \nabla_x \log p_t(x; \theta). \tag{14}$$

We defer the proof of Lemma 3.1 to Appendix B.4.1.

Connection to pathwise change-of-measure (Girsanov). The objective (12) is a pathwise integral of a velocity mismatch, aligning with the classical change-of-measure control suggested by Girsanov's theorem. Concretely, for two processes $dx_t = v(x_t, t)\,dt + dB_t$ and $d\bar{x}_t = \bar{v}(\bar{x}_t, t)\,dt + dB_t$, the path-space KL is bounded by a constant times $\int_0^T \mathbb{E}\left[\|\bar{v}(x_t, t) - v(x_t, t)\|_2^2\right]\,dt$ (under standard regularity, e.g. Novikov's condition). Related bounds have been used to control reverse *SDE* discretization error [22]. Our loss can be viewed as a tractable surrogate in this spirit: it replaces the intractable score-dependent v with the proxy velocity $\tilde{v}$ in (13).

Consistency check: isotropic trajectory loss reduces to weighted score matching. When $M_t(\theta) = g(t; \theta)I$ for a scalar function g, the weighting operator $W_t(\theta)$ becomes the scalar function $W_t(\theta) = \frac{\partial_t g(t;\theta)}{\sqrt{g(t;\theta)}\sqrt{1+g(t;\theta)}}\, I$. In Lemma B.2 (Appendix B), we formally prove the equivalence to a standard weighted score-matching objective. Thus, in the isotropic case, choosing $g(t; \theta)$ is essentially choosing a weighting over noise levels. Our main focus, however, is that (11) enables **optimization over matrix-valued trajectories**.

4 Optimization of the Trajectory Loss

This section explains how we *learn* the matrix-valued trajectory $M_t(\theta)$. We first emphasize the subtle but crucial relationship between θ and the learned network: the parametric model $\mathrm{net}(x, t, \phi)$ has *no explicit* dependence on θ, but the *optimal* network for a given schedule *does* depend on θ, because it approximates the score of the schedule-dependent marginal $p_t(\cdot; \theta) = p_0 * \mathcal{N}(0, M_t(\theta))$. We therefore present schedule learning as the following bilevel problem:

Let $L(\theta, \phi)$ be as defined in (11). Define

$$\phi^*(\theta) \in \arg\min_\phi L(\theta, \phi), \qquad \Leftrightarrow \qquad H(\theta) \;\; := L(\theta, \phi^*(\theta)) = \min_\phi L(\theta, \phi). \tag{15}$$

From (15), the joint minimization over (θ, ϕ) is equivalent to

$$\min_{\theta,\phi} L(\theta, \phi) = \min_\theta H(\theta).$$

Lemma 3.1 implies that in the infinite-data limit, $\mathrm{net}(x, t, \phi^*(\theta)) = \nabla_x \log p_t(x; \theta)$ for any θ.

The challenge of computing $\partial_\theta H(\theta)$: When optimizing θ in (15), we need $\partial_\theta H(\theta)$. Because $H(\theta)$ evaluates the loss at the *θ-dependent* optimum $\phi^*(\theta)$, differentiating H involves computing the derivative of the optimum score field wrt θ, i.e. $\partial_\theta \mathrm{net}(x, t, \phi^*(\theta)) \approx \partial_\theta \nabla \log p_t(x, \theta)$. Even though $\mathrm{net}(x, t, \phi)$ does not depend on θ, an implicit dependency on θ is introduced via $\phi^*(\theta)$.

The difficulty arises because at any point during training, **we only observe the current value of ϕ.** There is no simple way to directly compute $\partial_\theta \mathrm{net}(x, t, \phi^*(\theta))$ or even $\partial_\theta \phi^*(\theta)$. However, there is remarkable structure in $p_t(x, \theta) = p_0 * \mathcal{N}(0, M_t(\theta))$. By exploiting this fact, we show that $\partial_\theta \nabla \log p_t(x, \theta)$ can be written using **only derivatives wrt x, without explicit derivatives wrt θ.** In turn, this enables us to approximate $\partial_\theta \mathrm{net}(x, t, \phi^*(\theta))$ using only x-derivatives of the network.

4.1 A Plug-in Approximation for $\partial_\theta \nabla \log p_t$ via Stochastic Calculus

Below we provide an identity for expressing the θ-derivative of the true score $\partial_\theta \nabla \log p_t(x, \theta)$ using terms that only depend on entries of $\nabla^2 \log p_t(x, \theta)$.

Theorem 4.1. *Let $\theta \in \mathbb{R}^c$ and let $j \in \{1, \dots, c\}$ be any coordinate. Then for all $x \in \mathbb{R}^d$,*

$$\begin{aligned}\partial_{\theta_j} \nabla \log p_t(x;\theta) = \frac{1}{2}\sum_{i=1}^{d} \partial_r \partial_s \, \nabla \log p_t\Big(x + re_i + s\,\partial_{\theta_j} M_t(\theta)\, e_i; \theta\Big)\Big|_{r=s=0} \\ +\ \partial_s \, \nabla \log p_t\Big(x + s\,\partial_{\theta_j} M_t(\theta)\, \nabla \log p_t(x;\theta); \theta\Big)\Big|_{s=0}.\end{aligned} \tag{16}$$

Let $\phi^*(\theta)$ be defined as in (15)*. Under sufficient model capacity and data,*

$$\begin{aligned}\partial_{\theta_j} \mathrm{net}(x,t,\phi^*(\theta)) = \frac{1}{2}\sum_{i=1}^{d} \partial_r \partial_s \, \mathrm{net}\Big(x + re_i + s\,\partial_{\theta_j} M_t(\theta)\, e_i,\ t, \phi^*(\theta)\Big)\Big|_{r=s=0} \\ +\ \partial_s \, \mathrm{net}\Big(x + s\,\partial_{\theta_j} M_t(\theta)\, \mathrm{net}(x,t,\phi^*(\theta)),\ t, \phi^*(\theta)\Big)\Big|_{s=0}.\end{aligned} \tag{17}$$

We defer the proof to Appendix B.4.2.

4.2 Flow Parameterization and Variance Reduction

A practical concern is that $\|\mathrm{net}(x,t,\phi)\|_2$ typically scales like $\|M_t(\theta)^{-1/2}\|_2$, which can vary significantly across noise levels and inflate gradient variance. We therefore work with the normalized vector field

$$\mathrm{flow}(x,t,\phi) := M_t(\theta)^{1/2}\, \mathrm{net}(x,t,\phi), \tag{18}$$

whose scale is approximately time-invariant. Using Theorem 4.1, we can also express $\partial_\theta \mathrm{flow}$ in terms of x-directional derivatives:

Corollary 4.2. *Under the same setup as Theorem 4.1,*

$$\begin{aligned}\partial_\theta \mathrm{flow}(x,t,\phi^*(\theta)) = \frac{1}{2}\sum_{i=1}^{d} \partial_r \partial_s \, \mathrm{flow}\Big(x + re_i + s\,\partial_\theta M_t(\theta)\, e_i,\ t, \phi^*(\theta)\Big)\Big|_{r=s=0} \\ +\ \partial_s \, \mathrm{flow}\Big(x + s\, M_t(\theta)^{-1/2}\,\partial_\theta M_t(\theta)\, \mathrm{flow}(x,t,\phi^*(\theta)),\ t, \phi^*(\theta)\Big)\Big|_{s=0} \\ +\ \frac{1}{2}\, M_t(\theta)^{-1}\big(\partial_\theta M_t(\theta)\big)\, \mathrm{flow}(x,t,\phi^*(\theta)).\end{aligned}$$

We defer the proof of Corollary 4.2 to Appendix B.4.3.

4.3 Final Optimization Formulation for $H(\theta) = \min_\phi L(\theta,\phi)$

We now clarify how the schedule gradient uses (17) (or equivalently Corollary 4.2). We emphasize that the additional $\partial_\theta \mathrm{net}(x,t,\phi^*(\theta))$ term appears when differentiating the **outer value function** $H(\theta) = \min_\phi L(\theta,\phi)$, not when differentiating $L(\theta,\phi)$ with ϕ held fixed.

For notational clarity we treat θ as a scalar below; the vector case follows by applying the same derivation to each coordinate θ_j. Let $x_0 \sim p_0$, $\epsilon \sim \mathcal{N}(0, I)$, sample t as in Section 3, and set $x_t := x_0 + M_t(\theta)^{1/2}\epsilon$. Using the flow form of the loss, we verify that $L(\theta,\phi)$ of (11) is equivalent to

$$L(\theta,\phi) = \mathbb{E}_{x_0,\epsilon,t}\left[\Big\|W_t(\theta)\big(\mathrm{flow}(x_t,t,\phi) + \epsilon\big)\Big\|_2^2\right] \qquad W_t(\theta) := (I + M_t(\theta))^{-1/2}\,\partial_t M_t(\theta)\, M_t(\theta)^{-1/2}. \tag{19}$$

We define the convenient notation $\ell(v,\epsilon,W) := \|W(\theta)(v+\epsilon)\|_2^2$, so that $L(\theta,\phi) = \mathbb{E}_{x_0,\epsilon,t}\left[\ell(\mathrm{flow}(x_t,t,\phi),\epsilon,W_t(\theta))\right]$.

Chain rule for the outer objective. Define $\mathrm{flow}^*_\theta(x,t) := \mathrm{flow}(x,t,\phi^*(\theta))$ and recall $H(\theta) = L(\theta,\phi^*(\theta))$. Differentiating gives

$$\partial_\theta H(\theta) = \underbrace{\partial_\theta L(\theta,\phi)\Big|_{\phi=\phi^*(\theta)}}_{\text{explicit dependence through } M_t(\theta)} + \underbrace{\mathbb{E}_{x_0,\epsilon,t}\left[\Big\langle \nabla_v \ell(\mathrm{flow}(x_t,t,\phi^*(\theta)),\epsilon,W_t(\theta)),\ \partial_\theta \mathrm{flow}(x_t,t,\phi^*(\theta))\Big\rangle\right]}_{\text{implicit dependence through the optimal score/flow field}}, \tag{20}$$

The second term on the RHS above uses the (per-sample) gradient of $\ell(v, \epsilon, t)$ with respect to v given by $\nabla_{\text{flow}}\ell_t = 2\, W_t(\theta)^\top W_t(\theta)\,\big(v + \epsilon\big)$, this term isolates the dependence of $H(\theta)$ on $\phi^*(\theta)$. We estimate $\partial_\theta \text{flow}^*_\theta(x_t, t)$ using Corollary 4.2 (equivalently, estimate $\partial_\theta \text{net}^*_\theta$ using (17)). In practice, the expectation in (20) is approximated by minibatch averages over samples $(x_0^{(j)}, \epsilon^{(j)}, t^{(j)})$, for $j = 1...\text{BatchSize}$.

5 Denoising Algorithms for Matrix Trajectories

This section describes sampling (denoising) given (i) a learned PSD matrix trajectory $M_t(\theta)$ and (ii) a trained network $\text{flow}(x, t, \phi)$, where

$$\text{flow}(x, t, \phi) \approx M_t(\theta)^{1/2}\, \nabla_x \log p_t(x; \theta).$$

We focus on reverse-time ODE sampling, which is the inference method used throughout our experiments. We remark that our method generalizes easily to the reverse-time SDE setting. Via the projector-based parameterization of $M_t(\theta)$, scalar diffusion solvers (e.g., DPM-Solver-v3) can also be lifted to the anisotropic setting through subspace decomposition (Appendix D.4).

5.1 Reverse-time ODE in Flow Form and Initialization

Using $\text{net}(x, t, \phi) = M_t(\theta)^{-1/2}\text{flow}(x, t, \phi)$, and the learned ODE from (7), the reverse-time ODE is equivalent to the time-reversal of the following:

$$d\bar{x}_t = -\frac{1}{2}\, \partial_t M_t(\theta)\, M_t(\theta)^{-1/2}\, \text{flow}(\bar{x}_t, t, \phi)\, dt, \tag{21}$$

For initialization we sample $\xi \sim \mathcal{N}(0, I)$ and set

$$\bar{x}_T := M_T(\theta)^{1/2}\xi, \tag{22}$$

with T chosen large enough that p_T is close to a centered Gaussian.

Why the solver uses increments of $M_t^{1/2}$. In this section, we will assume that the matrices $M_t(\theta)$ share eigenvectors and therefore commute with $\partial_t M_t(\theta)$. This assumption is not necessary, but helps significantly simplify the algebra. In Section C, we discuss numerous examples of families of $M_t(\theta)$ which satisfy this assumption. Under this commutation, $\frac{1}{2}\, \partial_t M_t(\theta)\, M_t(\theta)^{-1/2} = \partial_t\big(M_t(\theta)^{1/2}\big)$, so (21) becomes

$$d\bar{x}_t = -\partial_t\big(M_t(\theta)^{1/2}\big)\, \text{flow}(\bar{x}_t, t, \phi)\, dt. \tag{23}$$

This reveals that the natural step size for anisotropic denoising is the *matrix increment* $\Delta M_t^{1/2}$, directly generalizing the scalar EDM viewpoint where $M_t = \sigma(t)^2 I$ and $\Delta M_k^{1/2} = (\sigma_{k-1} - \sigma_k)I$.

5.2 Reverse Euler Discretization

Let K be the number of solver steps and choose a discretization grid

$$0 = t_0 < t_1 < \cdots < t_K = T. \tag{24}$$

Reverse-time integration proceeds from t_K down to t_0. For convenience, define

$$U_k := M_{t_k}(\theta)^{1/2}, \qquad \Delta U_k := U_k - U_{k-1} \quad (k = 1, \ldots, K). \tag{25}$$

A first-order explicit Euler discretization of the *time-reversal* of (23) yields

$$\bar{x}_{t_{k-1}} = \bar{x}_{t_k} + \Delta U_k\, \text{flow}(\bar{x}_{t_k}, t_k, \phi), \qquad k = 1, \ldots, K. \tag{26}$$

We present the above Euler-Discretization scheme (26) mainly as a baseline derivation; in our experiments, second-order Heun consistently dominates Euler in FID per NFE.

5.3 Heun's Second-order Method for Matrix Trajectories

We now present the matrix-valued generalization of Heun's second-order algorithm. We inherit the notation from Section 5.2 above. Fix a secondary evaluation time $\hat{t}_k \in [t_{k-1}, t_k]$ and denote $\hat{U}_k := M_{\hat{t}_k}(\theta)^{1/2}$. Let $u_k := -\text{flow}(\bar{x}_{t_k}, t_k, \phi)$. We form a predictor at $\hat{t}_k$ by

$$\hat{x}_{\hat{t}_k} = \bar{x}_{t_k} + (\hat{U}_k - U_k)\, u_k, \qquad \hat{u}_k := -\text{flow}(\hat{x}_{\hat{t}_k}, \hat{t}_k, \phi). \tag{27}$$

At any time $s \in [t_{k-1}, t_k]$, we approximate the intermediate flow field at s by the affine model $\tilde{u}(s) := u_k + (U(s) - U_k)\,(\hat{U}_k - U_k)^{-1}\,(\hat{u}_k - u_k)$, with $U(s) := M_s(\theta)^{1/2}$. Substituting this approximation into (23) and integrating yields the closed-form second-order update

$$\bar{x}_{t_{k-1}} = \bar{x}_{t_k} - \Delta U_k\, u_k + \frac{1}{2}\,(\Delta U_k)^2 (\hat{U}_k - U_k)^{-1} (\hat{u}_k - u_k). \tag{28}$$

We defer the proof to Lemma B.3 in the appendix. When $\hat{t}_k = t_{k-1}$ (endpoint choice), we have $\hat{U}_k = U_{k-1}$ and (28) reduces to the familiar Heun form $\bar{x}_{t_{k-1}} = \bar{x}_{t_k} - \Delta U_k\, \frac{u_k + \hat{u}_k}{2}$, which matches the standard scalar EDM/Heun update when $M_t = \sigma(t)^2 I$. In experiments we consider two choices of $\hat{t}_k$, i.e. $\hat{t}_k = t_{k-1}$ (endpoint) and $\hat{t}_k = (t_{k-1} + t_k)/2$ (midpoint).

The solver grid $\{t_k\}_{k=0}^K$ is treated as an orthogonal design choice. In our experiments, the grid is uniform over $[0, T]$, i.e., $t_k = kT/K$.

Efficient implementation under structured parameterizations. All updates require only (i) matrix–vector products with $M_t(\theta)^{1/2}$ and (ii) product of $(\hat{U}_k - U_k)^{-1}$ with vectors. Under the projector-based families in Section C, these operations reduce to inexpensive subspace-wise scalings (and subspace-wise pseudoinverses), avoiding explicit $d \times d$ matrix square roots or dense linear algebra.

6 Experiments

We evaluate learned anisotropic diffusion schedules on four standard image generation benchmarks: CIFAR-10 (32×32) [23], AFHQv2 (64×64) [9], FFHQ (64×64) [24], and ImageNet-64 (64×64) [25].

Baselines. We compare against two groups:

- **(a) EDM** [3]: matched-architecture isotropic-diffusion reference with Heun sampling.
- **(b) Few-step solver-side baselines**: AYS [10] (learned time discretization) and DPM-Solver-v3 [11], the latest DPM-Solver variant (after DPM-Solver and DPM-Solver++) and a state-of-the-art training-free solver for few-NFE.

Our framework acts at the *training* level and is complementary to (b); we therefore also report ***Ours + DPM-v3***, where the orthogonal-projector structure of $M_t(\theta)$ decouples dynamics into per-subspace scalar diffusions to which DPM-Solver-v3 applies directly (Appendix D.4).

Training protocol. Our models are initialized from the corresponding pretrained EDM networks and then fine-tuned while learning the schedule parameters θ (and continuing to update network parameters ϕ) as in (19). Across all datasets, fine-tuning consumes the equivalent of 1.2M image passes. Full optimization, distributed-training, and EMA details are given in Appendix D.2. To isolate the effect of schedule learning from raw fine-tuning compute, we further fine-tune EDM on ImageNet-64 (the largest benchmark) under both EDM's own recipe and our hyperparameters, and find FID within run-to-run variance of the released checkpoint (Appendix E.4).

Evaluation protocol. We generate 50k samples per configuration and compute Fréchet Inception Distance (FID↓). For both EDM and our method we repeat sampling three times with independent random seeds and report the *minimum* FID, matching the EDM evaluation protocol [3]. Solver budgets are reported in number of function evaluations (NFE). EDM's Heun sampler with N steps uses NFE $= 2N - 1$, since its second-order correction is skipped on the final step to $\sigma = 0$ [3]. At

each paired budget, our midpoint matrix Heun sampler (Section 5) uses NFE $= 2N - 2$, one fewer than EDM (see Appendix D.3 for the breakdown), so our reported FID is conservative against EDM. Comparisons against AYS and DPM-Solver-v3 in Table 2 use the corresponding solver-side methods.

Schedule variants. For each dataset we train and evaluate a subset of the following instantiations of $M_t(\theta) = \sum_j g_j(t;\theta)P_j$, which span a range of expressivity: global isotropic schedules g, class-conditional isotropic schedules ($\{g_y\}_{y=1}^C$), two-subspace anisotropic schedules over a fixed DCT basis ($g_1^{\mathrm{DCT}}, g_2^{\mathrm{DCT}}$), two-subspace anisotropic schedules over class-conditional PCA bases ($g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}}$), and the class-conditional analogues $\{(g_{1,y}^{\mathrm{DCT}}, g_{2,y}^{\mathrm{DCT}})\}_{y=1}^C$ and $\{(g_{1,y}^{\mathrm{PCA}}, g_{2,y}^{\mathrm{PCA}})\}_{y=1}^C$. Class conditioning can enter through the subspace basis, the scalar schedules, or both. In particular, ($g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}}$) uses class-conditional PCA bases with schedules shared across classes, whereas $\{(g_{1,y}^{\mathrm{PCA}}, g_{2,y}^{\mathrm{PCA}})\}_{y=1}^C$ makes both the bases and schedules class-dependent. Detailed definitions appear in Appendix C.2 and C.3.

CIFAR-10								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	17/16	35/34	59/58	79/78
EDM	35.52	14.37	6.694	4.231	3.027	**1.829**	1.868	1.890
Ours	**3.839**	**2.900**	**2.538**	**2.332**	**2.213**	1.851	**1.803**	**1.803**
AFHQv2								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	27.98	13.66	7.587	4.746	2.986	2.075	2.042	2.046
Ours	**4.524**	**2.977**	**2.513**	**2.401**	**2.265**	**2.070**	**2.015**	**2.010**
FFHQ								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	57.14	29.39	15.81	9.769	5.169	2.575	2.391	2.374
Ours	**4.967**	**3.776**	**3.304**	**3.011**	**2.711**	**2.350**	**2.297**	**2.318**
ImageNet-64								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	35.32	15.19	8.160	5.458	3.586	2.428	2.294	2.276
Ours	**8.366**	**5.823**	**4.626**	**3.992**	**3.282**	**2.378**	**2.259**	**2.239**

Table 1: FID ↓ vs. NFE across datasets (50k samples); NFE columns are paired (EDM / Ours), with Ours using one fewer evaluation. "Ours" reports the class-conditional PCA-basis on CIFAR-10, the class-conditional PCA-basis with conditional schedule on ImageNet-64, and the DCT-basis on AFHQv2 and FFHQ (full per-variant breakdown with standard deviations in Table 3, Appendix E.1). Each entry is the minimum FID across 3 independent generations of 50k images.

FFHQ				
NFE	10	20	30	50
AYS	5.430	3.290	2.870	2.620
Ours	**3.776**	**2.625**	**2.407**	**2.356**

CIFAR-10				
NFE	6	8	34	78
DPM-Solver-v3	6.314	3.845	1.913	1.914
Ours (Heun)	11.03	3.618	**1.815**	**1.803**
Ours + DPM-Solver-v3	**4.208**	**3.036**	2.383	2.384

Table 2: FID ↓ vs. NFE in the few-step regime (50k samples). **Left:** our method versus AYS on FFHQ. **Right:** composing our method with DPM-Solver-v3 on CIFAR-10; the combination achieves the best FID at low NFE (Appendix D.4), while our matrix Heun solver alone wins at higher NFE.

6.1 Discussion of Results

Table 1 reports FID across solver budgets against the EDM baseline; Table 2 compares against solver-side baselines (AYS, DPM-Solver-v3) and the composed sampler. Figures 2–5 (Appendix E.2) visualize the learned schedules and anisotropy ratios.

(i) Consistent FID improvement over the EDM baseline. Our method consistently outperforms EDM in Table 1, with the most pronounced gains at low NFE: at NFE ≈ 9, FID improves from $35.52 \to 3.84$ on CIFAR-10, $27.98 \to 4.52$ on AFHQv2, $57.14 \to 4.97$ on FFHQ, and $35.32 \to 8.37$ on ImageNet-64. At converged-NFE budgets the absolute gap is smaller but still consistent: CIFAR-10 $1.829 \to 1.803$, AFHQv2 $2.042 \to 2.010$, FFHQ $2.374 \to 2.297$, ImageNet-64 $2.276 \to 2.239$. The single exception is CIFAR-10 at NFE 35, where EDM's 1.829 slightly beats our 1.851; nevertheless, at all other CIFAR-10 NFE budgets (including the higher-NFE points 59 and 79) our method dominates, and our overall best FID across NFEs (1.803) is still better than EDM's overall best (1.829).

(ii) Matrix-valued anisotropy demonstrates benefits. Beyond the isotropic warm-up, matrix-valued schedules are often beneficial. As shown in Table 3 (Appendix E.1), on CIFAR-10 the $(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$ schedule improves upon both EDM and the learned isotropic schedule g at every budget except NFE 34, and on AFHQv2 and FFHQ the DCT anisotropic schedule improves upon EDM at every budget. On ImageNet-64, class-conditional anisotropic variants (over both DCT and PCA bases) are the strongest, with FID approaching 2.24 at the highest NFE budgets.

(iii) Class-conditional variants matter most on complex conditional data. On ImageNet-64, class-conditional schedules improve substantially over shared schedules at low NFE (PCA basis: 8.366 vs. 29.47 at NFE 8), and the class-conditional anisotropic variants over DCT and PCA bases perform comparably (within run-to-run variability of each other; see Table 3). On CIFAR-10, the best FID (1.803) comes from $(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$, which uses class-conditional PCA bases with shared schedules, whereas class-conditional isotropic schedules improve over the global isotropic schedule at the lowest budget (3.691 vs. 4.241 at NFE 8).

(iv) Competitive with solver-side baselines, and best when composed. Our method outperforms AYS at all NFE budgets on FFHQ (Table 2, left): e.g. FID 3.776 vs. AYS's 5.430 at NFE $= 10$ and 2.356 vs. 2.620 at NFE $= 50$. A further practical advantage of our framework is that the projector-based parameterization $M_t(\theta) = \sum_j g_j(t;\theta) P_j$ (Section C.1) decouples the diffusion into independent scalar VE processes per subspace, so DPM-Solver-v3 can be applied within each subspace using the corresponding $g_j(t;\theta)$ (Appendix D.4). The composition (***Ours + DPM-v3***) substantially outperforms either method alone at very low NFE: FID 4.208 vs. DPM-v3's 6.314 and our anisotropic Heun's 11.03 at NFE $= 6$, and 3.036 vs. 3.845 and 3.618 at NFE $= 8$ (Table 2). At higher NFE budgets (≥ 34), our matrix Heun solver alone is strongest. The regime split suggests a practical recipe: use ***Ours + DPM-v3*** for fast inference (NFE ≤ 10) and the matrix Heun solver of Section 5 when fidelity matters more than speed.

7 Conclusion

We introduced a variational framework for anisotropic diffusion that replaces the scalar noise level with a monotone PSD matrix trajectory $M_t(\theta)$ learned jointly with the score network. Three pieces enable this: a trajectory-level objective with a pathwise change-of-measure interpretation, a plug-in identity for $\partial_\theta \nabla \log p_t$ via x-directional derivatives only, and a matrix-valued Heun reverse-time ODE solver. Empirically, learned schedules consistently improve over EDM on CIFAR-10, AFHQv2, FFHQ, and ImageNet-64, with largest gains at low NFE and strongest results from class-conditional schedules or bases. Composing with DPM-Solver-v3 further yields the best FID at very low NFE.

Acknowledgments and Disclosure of Funding

We thank the Duke Compute Cluster (DCC) for providing the computational resources used in this work. This work received no third-party funding. The authors declare no competing interests.

References

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

[2] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

[4] Daniel Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. *Advances in neural information processing systems*, 6, 1993.

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

[6] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22930–22941, 2023.

[7] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. *arXiv preprint arXiv:2303.08320*, 2023.

[8] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[10] Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. *arXiv preprint arXiv:2404.14507*, 2024.

[11] Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36: 55502–55542, 2023.

[12] Emiel Hoogeboom and Tim Salimans. Blurring diffusion models. *arXiv preprint arXiv:2209.05557*, 2022.

[13] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

[14] Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022.

[15] Vikram Voleti, Christopher Pal, and Adam Oberman. Score-based denoising diffusion with non-isotropic gaussian noise models. *arXiv preprint arXiv:2210.12254*, 2022.

[16] Luca Scimeca, Thomas Jiralerspong, Berton Earnshaw, Jason Hartford, and Yoshua Bengio. Learning what matters: Steering diffusion via spectrally anisotropic forward noise. *arXiv preprint arXiv:2510.09660*, 2025.

[17] Jeffrey Alido, Tongyu Li, Yu Sun, and Lei Tian. Whitened score diffusion: A structured prior for imaging inverse problems. *Advances in Neural Information Processing Systems*, 38: 65168–65195, 2026.

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

[19] Subham Sahoo, Aaron Gokaslan, Christopher M De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. *Advances in Neural Information Processing Systems*, 37: 105730–105779, 2024.

[20] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

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

[22] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[26] Jente Vandersanden, Sascha Holl, Xingchang Huang, and Gurprit Singh. Edge-preserving noise for diffusion models. In *Computer Graphics Forum*, page e70383. Wiley Online Library, 2025.

[27] Xingchang Huang, Corentin Salaun, Cristina Vasconcelos, Christian Theobalt, Cengiz Oztireli, and Gurprit Singh. Blue noise for diffusion models. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[28] Bowen Jing, Gabriele Corso, Renato Berlinghieri, and Tommi Jaakkola. Subspace diffusion generative models. In *European conference on computer vision*, pages 274–289. Springer, 2022.

[29] Pascal Chang, Jingwei Tang, Markus Gross, and Vinicius C Azevedo. How i warped your noise: a temporally-correlated noise prior for diffusion models. *arXiv preprint arXiv:2504.03072*, 2025.

[30] Kushagra Pandey and Stephan Mandt. A complete recipe for diffusion generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4261–4272, 2023.

[31] Raghav Singhal, Mark Goldstein, and Rajesh Ranganath. Where to diffuse, how to diffuse, and how to get back: Automated learning for multivariate diffusions. *arXiv preprint arXiv:2302.07261*, 2023.

[32] Weitao Du, He Zhang, Tao Yang, and Yuanqi Du. A flexible diffusion model. In *International Conference on Machine Learning*, pages 8678–8696. PMLR, 2023.

[33] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023.

Appendix

Table of Contents

A Related Work 14

B Theoretical Foundations 14

B.1 Anisotropic Score as a Conditional Expectation 14

B.2 Equivalence of Isotropic Schedules and Weighted Score-Matching 15

B.3 Closed-Form Integral for the Matrix Heun Step 16

B.4 Proofs of Main Results 16

B.4.1 Proof of Lemma 3.1 16

B.4.2 Proof of Theorem 4.1 17

B.4.3 Proof of Corollary 4.2 17

C Practical Choices of Matrix Schedule $M_t(\theta)$ 18

C.1 A Subspace-Schedule Framework 18

C.2 Schedule Families Used in Experiments 18

C.3 Schedule Variants Evaluated 19

D Experiment Details 20

D.1 Subspace Schedule Parameterization g_i 20

D.2 Training Setup 21

D.3 Evaluation Protocol Details 21

D.4 Composing Anisotropic Schedules with DPM-Solver-v3 21

E Additional Experimental Analysis 23

E.1 Per-Variant FID Breakdown 23

E.2 Learned Schedules across Datasets 24

E.3 Training and Inference Speed 24

E.4 Fine-Tuning Controls with Matched Image Budgets 26

E.5 Subspace Bases 27

E.5.1 2D-DCT bases 27

E.5.2 PCA bases 27

F Broader Impacts 28

A Related Work

Structured noising and anisotropy. Prior work introduces non-isotropic structure into diffusion via edge-aware perturbations [26], correlated or blue-noise masks [27], frequency- or subspace-restricted noise [28], and temporally structured priors for video models [7, 6, 29]. These approaches typically fix the structure of the forward process by hand (selecting a particular spatial pattern, frequency mask, or temporal correlation) rather than learning the matrix trajectory from data. In contrast, we learn a *global*, monotone PSD *matrix* trajectory $M_t(\theta)$ end-to-end (e.g., DCT- or PCA-parameterized) and optimize it *jointly* with the score network via a trajectory-level loss, allocating noise (and denoising effort) across *directions and subspaces* in a data-driven way.

Multivariate diffusions and learned matrix parameters. A closer line of work jointly considers matrix-valued forward-process parameters alongside a score model, but each restricts the matrix family for tractability and stops short of optimizing a general $d \times d$ covariance trajectory. Pandey and Mandt [30] provide a general recipe for forward-process design in augmented phase space; for tractability they restrict the diffusion and drift matrices $D(z), Q(z)$ to constants and instantiate the resulting PSLD model with a small set of scalar hyperparameters (Γ, M, ν, β). These matrices are therefore treated as design choices/hyperparameters set by hand rather than a jointly learned matrix trajectory. Singhal et al. [31] optimize an ELBO for multivariate diffusions parameterized through low-dimensional auxiliary-variable dynamics: with $K-1$ auxiliary variables, the forward matrices are $K \times K$ (with $K \in \{2, 3\}$ in their experiments) and shared across data coordinates; their Appendix C further parameterizes $Q_\phi(s), D_\phi(s)$ as scalar time-functions multiplying time-independent matrices, with D taken diagonal in their experiments. The matrix family is thus low-dimensional and coordinate-shared rather than a general $d \times d$ data-space trajectory. Du et al. [32], motivated by Riemannian and symplectic geometry, derive a flexible class of multivariate flow-based models, but for image generation explicitly use a simplified linear FP-Diffusion with $R^{-1}(x)$ taken as a constant SPD matrix independent of x; the practical image model thus uses a global constant metric rather than a learned data-dependent manifold.

We instead learn a general data-space covariance path $M_t(\theta) \in \mathbb{R}^{d\times d}$ end-to-end via a trajectory-level loss. This generality introduces a key technical challenge: changing θ changes the target score $\nabla \log p_t(x; \theta)$, which is not available from a network trained at a single θ. Estimating $\partial_\theta \nabla \log p_t(x; \theta)$ efficiently is exactly the purpose of Theorem 4.1.

ELBO views and schedule learning. VDM [18] and the ELBO reinterpretation of common diffusion losses [33] relate *scalar* schedule learning to reweighting *noise levels*. MuLAN [19] learns (possibly input-conditional) *per-pixel* noise rates to tighten an ELBO for density estimation. Our formulation differs in that (i) $M_t(\theta)$ is *anisotropic* (reweights *directions*, not just levels), (ii) we derive a practical schedule-gradient estimator that accounts for the schedule-dependent score target, and (iii) we optimize the schedule to improve *sample quality*, yielding consistent FID gains.

Few-step samplers and solver design. A complementary line of work targets the *sampler* rather than the model, including DPM-Solver and its higher-order extensions [20, 11] and learned step schedules such as Align-Your-Steps (AYS) [10]. Our approach acts on the *training objective* and the *forward schedule* $M_t(\theta)$, and is therefore largely orthogonal to these solver-side methods. In Section 6, we show that our learned anisotropic trajectories match or outperform AYS and DPM-Solver-v3 across a range of NFE budgets, and compose with DPM-Solver-v3 (Appendix D.4) to yield further gains in the very-low-NFE regime.

B Theoretical Foundations

B.1 Anisotropic Score as a Conditional Expectation

Lemma B.1. *If x_t evolves as either one of the two processes:*

$$1.\ (x_t - x_0) \sim \mathcal{N}(0, M_t(\theta))$$

$$2.\ dx_t = -\frac{1}{2}\partial_t M_t(\theta) \nabla \log p_t(x_t; \theta) dt,$$

where $p_t(x;\theta) := p_0 * \mathcal{N}(0, M_t(\theta))$, then the score above is the conditional expectation

$$\nabla \log p_t(x;\theta) = M_t^{-1}(\theta)\mathbb{E}_{x_0|x_t=x}\left[x_0 - x_t\right],$$

where (x_0, x_t) are defined by the joint distribution $x_0 \sim p_0$ and $x_t = x_0 + \mathcal{N}(0, M_t)$.

Proof of Lemma B.1. Fix t and θ, and write $\Sigma := M_t(\theta)$. For notational simplicity we suppress the dependence on (t, θ) when clear. The marginal $p_t(\cdot;\theta)$ is the Gaussian convolution

$$p_t(x;\theta) = \int_{\mathbb{R}^d} p_0(x_0)\,\varphi_\Sigma(x - x_0)\,dx_0, \qquad \varphi_\Sigma(z) := (2\pi)^{-d/2}|\Sigma|^{-1/2}\exp\Big(-\tfrac{1}{2}z^\top\Sigma^{-1}z\Big).$$

Assuming $\Sigma \succ 0$, we may differentiate under the integral sign to obtain

$$\nabla_x p_t(x;\theta) = \int p_0(x_0)\,\nabla_x\varphi_\Sigma(x - x_0)\,dx_0.$$

Since $\nabla_x\varphi_\Sigma(x - x_0) = \Sigma^{-1}(x_0 - x)\,\varphi_\Sigma(x - x_0)$, we get

$$\nabla_x p_t(x;\theta) = \Sigma^{-1}\int p_0(x_0)\,(x_0 - x)\,\varphi_\Sigma(x - x_0)\,dx_0.$$

Dividing by $p_t(x;\theta)$ yields

$$\nabla_x \log p_t(x;\theta) = \Sigma^{-1}\,\frac{\int p_0(x_0)\,(x_0 - x)\,\varphi_\Sigma(x - x_0)\,dx_0}{\int p_0(x_0)\,\varphi_\Sigma(x - x_0)\,dx_0} = \Sigma^{-1}\,\mathbb{E}\left[x_0 - x \mid x_t = x\right],$$

where the conditional distribution is $p(x_0 \mid x_t = x) \propto p_0(x_0)\varphi_\Sigma(x - x_0)$. Finally, since $x_t = x$ on the conditioning event, $x_0 - x = x_0 - x_t$, giving the stated identity. □

B.2 Equivalence of Isotropic Schedules and Weighted Score-Matching

Lemma B.2. *For any $w(t)$, there exists a $g_t(\theta)$ and constant c, such that for any $h(t)$*

$$\int_0^T \frac{(\partial_t g_t(\theta))^2}{1 + g_t(\theta)} h(g_t(\theta))dt = c\int_0^T w(t)h(t)dt.$$

Proof of Lemma B.2. Let $w : [0, T] \to (0, \infty)$ be measurable and define

$$\Phi(x) := \int_0^x \frac{ds}{(1 + s)w(s)} < \infty.$$

Then, to prove this Lemma, it is sufficient to show there exist a $c = \frac{\Phi(T)}{T} > 0$ and a strictly increasing continuous function $g : [0, T] \to [0, T]$ with $g(0) = 0,\ g(T) = T$ such that

$$\frac{g'\big(g^{-1}(t)\big)}{1 + t} = cw(t) \text{ for all } t.$$

where $c = \Phi(T)/T$.

Define g implicitly by $\Phi\big(g(t)\big) = ct$ for $(0 \le t \le T)$, i.e. $g(t) = \Phi^{-1}(ct)$. Differentiating $\Phi(g(t)) = ct$ with respect to t yields the separable ODE $g'(t) = (1 + g(t))cw(g(t))$, and with $r = g(t)$ this is $\frac{g'(g^{-1}(r))}{1+r} = cw(r)$. Hence, we derive an expression for g involving a constant c and any w. Monotonicity of $g(t)$ follows since $w > 0$ and $c > 0$ imply $g'(t) > 0$.

Substituting into $c\int_0^T w(t)h(t)dt$, we get

$$c\int_0^T w(r)h(r)dr = \int_0^T \frac{g'(t)}{1 + g(t)} h(g(t))g'(t)dt.$$

□

B.3 Closed-Form Integral for the Matrix Heun Step

Lemma B.3. *Assume M_t commutes with $\partial_t M_t$ for all t. Let $\tilde{u}$ be as defined in Section 5. Then*

$$\int_{t_k}^{t_{k-1}} (\partial_s M_s^{1/2})\tilde{u}(s)\,ds$$
$$= (M_{t_k}^{1/2} - M_{t_{k-1}}^{1/2})(\mathrm{flow}(\bar{x}_{t_k}, t_k)) - \frac{1}{2}(M_{t_{k-1}}^{1/2} - M_{t_k}^{1/2})^2\big(M_{\hat{t}_k}^{1/2} - M_{t_k}^{1/2}\big)^{-1} \cdot \big(\mathrm{flow}(\hat{x}_{\hat{t}_k}, \hat{t}_k) - \mathrm{flow}(\bar{x}_{t_k}, t_k)\big).$$

Proof. Recall our notation that

$$u_k := -\mathrm{flow}(\bar{x}_{t_k}, t_k, \phi), \qquad \hat{u}_k := -\mathrm{flow}(\hat{x}_{\hat{t}_k}, \hat{t}_k, \phi).$$

$$\begin{aligned}
&\int_{t_k}^{t_{k-1}} (\partial_s M_s^{1/2}(\theta))\tilde{u}(s)\,ds \\
&= \int_{t_k}^{t_{k-1}} (\partial_s M_s^{1/2}(\theta))\Big(u_k + (M_s^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))\,(M_{\hat{t}_k}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))^{-1}\,(\hat{u}_k - u_k)\Big)ds \\
&= \Big(M_{t_{k-1}}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta)\Big)u_k + \int_{t_k}^{t_{k-1}} (\partial_s M_s^{1/2}(\theta))(M_s^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))\,ds\;\; (M_{\hat{t}_k}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))^{-1}(\hat{u}_k - u_k) \\
&= \Big(M_{t_{k-1}}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta)\Big)u_k + \frac{1}{2}(M_{t_{k-1}}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))^2(M_{\hat{t}_k}^{1/2}(\theta) - M_{t_k}^{1/2}(\theta))^{-1}(\hat{u}_k - u_k).
\end{aligned}$$

□

B.4 Proofs of Main Results

B.4.1 Proof of Lemma 3.1

We simplify the term inside the Euclidean norm in (12):

$$\begin{aligned}
&\bar{v}(x_t, t; \theta, \phi) - \tilde{v}(x_t, x_0, t; \theta) \\
&= \frac{1}{2}(I + M_t(\theta))^{-1/2}\partial_t M_t(\theta)(M_t^{-1}(\theta)(x_0 - x_t) - \mathrm{net}(x_t, t, \phi)).
\end{aligned}$$

Let M_{all} denote $(I + M_t(\theta))^{-1/2}\partial_t M_t(\theta)$. Let $V : \mathbb{R}^d \to \mathbb{R}^d$ be an arbitrary vector field.

We can minimize the expectation pointwise at each x_t. We can rewrite the expectation

$$\mathbb{E}_{x_0,\epsilon}\left[\left\|M_{\text{all}}(\mathrm{net}(x_t, t, \phi) - M_t^{-1}(\theta)(x_0 - x_t))\right\|_2^2\right]$$

as

$$\begin{aligned}
&\mathbb{E}_{x_0,\epsilon}\left[\left\|M_{\text{all}}(V(x_t) - M_t^{-1}(\theta)(x_0 - x_t))\right\|_2^2\right] \\
&= \mathbb{E}_{x_t}\left[\mathbb{E}_{x_0,\epsilon|x_t}\left[\left\|M_{\text{all}}(V(x_t) - M_t^{-1}(\theta)(x_0 - x_t))\right\|_2^2\right]\right]
\end{aligned}$$

by the law of iterated expectation.

$$\begin{aligned}
&\arg\min_{v\in\mathbb{R}^d} \mathbb{E}_{x_0,\epsilon|x_t}\Big[(v - M_t^{-1}(\theta)(x_0 - x_t))^\top \\
&\qquad M_{\text{all}}^\top M_{\text{all}}(v - M_t^{-1}(\theta)(x_0 - x_t))\Big] \\
&= M_t^{-1}(\theta)\mathbb{E}_{x_0,\epsilon|x_t}\left[x_0 - x_t\right] = \nabla \log p_t(x_t; \theta).
\end{aligned}$$

Let $Q = M_{\text{all}}^\top M_{\text{all}}$. Under (3), M_{all} is invertible, so Q is positive definite. The conditional quadratic loss is therefore uniquely minimized at $M_t(\theta)^{-1}\mathbb{E}[x_0 - x_t \mid x_t]$, which equals the score by (4).

B.4.2 Proof of Theorem 4.1

To simplify notation, we will drop the index j and treat θ as a scalar. The general proof for $\theta \in \mathbb{R}^c$ follows by repeating the proof for each θ_j, while holding all other $\theta_k's$ fixed. Recall that $p_t(\cdot;\theta) = p_0 * \mathcal{N}(0, M_t(\theta))$. By the heat equation (applied to $\partial_\theta M_t(\theta)$ instead of $\partial_t M_t(\theta)$),

$$\begin{aligned}
&\partial_\theta p_t(x;\theta) \\
=&\frac{1}{2}\mathbf{div}(p_t(x;\theta)\partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)) \\
=&\frac{1}{2}p_t(x;\theta)\mathbf{div}(\partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)) \\
&+ \frac{1}{2}p_t(x;\theta)\left\langle \nabla \log p_t(x;\theta), \partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)\right\rangle .
\end{aligned}$$

Dividing both sides by $p_t(x;\theta)$ gives

$$\begin{aligned}
&\partial_\theta \log p_t(x;\theta) \\
=&\frac{1}{2}\mathbf{div}(\partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)) \\
&+ \frac{1}{2}\left\langle \nabla \log p_t(x;\theta), \partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)\right\rangle \\
=&\frac{1}{2}\sum_i \left\langle \partial_\theta M_t(\theta)e_i, \nabla^2 \log p_t(x;\theta)e_i\right\rangle \\
&+ \frac{1}{2}\left\langle \nabla \log p_t(x;\theta), \partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)\right\rangle \\
=&\frac{1}{2}\sum_i \left\langle \partial_\theta M_t(\theta)e_i, \partial_c \nabla \log p_t(x+ce_i;\theta)\right\rangle \\
&+ \frac{1}{2}\left\langle \nabla \log p_t(x;\theta), \partial_\theta M_t(\theta)\nabla \log p_t(x;\theta)\right\rangle .
\end{aligned}$$

Taking a derivative wrt x gives

$$\begin{aligned}
&\partial_\theta \nabla \log p_t(x;\theta) \\
=&\frac{1}{2}\sum_i \left\langle \partial_\theta M_t(\theta)e_i, \partial_c \nabla^2 \log p_t(x+ce_i;\theta)\right\rangle \\
&+ \left\langle \partial_\theta M_t(\theta)\nabla \log p_t(x;\theta), \nabla^2 \log p_t(x;\theta)\right\rangle \\
=&\frac{1}{2}\sum_i \partial_r\partial_s \nabla \log p_t(x+re_i+s\partial_\theta M_t(\theta)e_i;\theta) \\
&+ \partial_s \nabla \log p_t(x+s\partial_\theta M_t(\theta)\nabla \log p_t(x;\theta);\theta).
\end{aligned}$$

This concludes the proof of (16). To prove (17), we combine (16) with the guarantee that $\mathrm{net}(x,t,\phi^*(\theta)) = \nabla \log p_t(x;\theta)$ from Lemma 3.1.

B.4.3 Proof of Corollary 4.2

Recall that

$$\mathrm{flow}(x;\theta) = M_t(\theta)^{1/2}\mathrm{net}(x;\theta) = M_t(\theta)^{1/2}\nabla \log p_t(x;\theta).$$

Recall also from (17) that

$$\begin{aligned}
&\partial_\theta \mathrm{net}(x,t,\phi^*(\theta)) \\
=&\frac{1}{2}\sum_{i=1}^d \partial_r\partial_s \mathrm{net}(x+re_i+s\partial_\theta M_t(\theta)e_i,t,\phi^*(\theta)) \\
&+ \partial_s \mathrm{net}(x+s\partial_\theta M_t(\theta)\mathrm{net}(x,t,\phi^*(\theta)),t,\phi^*(\theta)).
\end{aligned}$$

Combining the two equations above:

$$
\begin{aligned}
&\partial_\theta \mathrm{flow}(x;\phi^*(\theta)) \\
=&M_t(\theta)^{1/2}\partial_\theta \mathrm{net}(x;\phi^*(\theta)) + \frac{1}{2}M_t(\theta)^{-1/2}(\partial_\theta M_t(\theta))\mathrm{net}(x;\phi^*(\theta)) \\
=&\frac{1}{2}\sum_i M_t(\theta)^{1/2}\partial_r\partial_s \mathrm{net}(x + re_i + s\partial_\theta M_t(\theta)e_i;\phi^*(\theta)) \\
&+ M_t(\theta)^{1/2}\partial_s \mathrm{net}(x + s\partial_\theta M_t(\theta)\mathrm{net}(x;\phi^*(\theta));\phi^*(\theta)) + \frac{1}{2}M_t(\theta)^{-1/2}(\partial_\theta M_t(\theta))\mathrm{net}(x;\phi^*(\theta)) \\
=&\frac{1}{2}\sum_i \partial_r\partial_s \mathrm{flow}(x + re_i + s\partial_\theta M_t(\theta)e_i;\phi^*(\theta)) \\
&+ \partial_s \mathrm{flow}(x + sM_t(\theta)^{-1/2}\partial_\theta M_t(\theta)\mathrm{flow}(x;\phi^*(\theta));\phi^*(\theta)) + \frac{1}{2}M_t(\theta)^{-1}(\partial_\theta M_t(\theta))\mathrm{flow}(x;\phi^*(\theta)).
\end{aligned}
$$

C Practical Choices of Matrix Schedule $M_t(\theta)$

The framework in Sections 3–5 applies to arbitrary monotone PSD matrix trajectories $M_t(\theta)$. Below, we demonstrate a number of specific parametric families of $M_t(\theta)$. In each case, we verify that the parameterization satisfies: (i) **monotonicity** ($\partial_t M_t(\theta) \succ 0$), (ii) **efficient evaluation** of matrix functions appearing in the loss and sampler (e.g. $M_t^{\pm 1/2}$ and $(I + M_t)^{-1/2}$), and (iii) **commutation of** $\partial_t M_t(\theta)$ and $M_t(\theta)$.

C.1 A Subspace-Schedule Framework

We use a broad family based on an orthogonal decomposition of $\mathbb{R}^d$ into subspaces. Let $\{P_j\}_{j=1}^J$ be *orthogonal projection matrices* such that $P_j = P_j^\top \succeq 0$, $P_j^2 = P_j$, $P_j P_k = 0$ for $j \neq k$, and $\sum_{j=1}^J P_j = I$. Let $\{g_j(\cdot;\theta)\}_{j=1}^J$ be strictly increasing scalar functions with $g_j(0;\theta) = g_0$ and $g_j(T;\theta) = T$, where $g_0 > 0$ is a small constant, so that $M_0 = g_0 I \approx 0$. In our experiments, each $g_j(\cdot;\theta)$ is parameterized by a monotone knot-based family (Appendix D.1), which ensures $\dot{g}_j > 0$ and differentiability in θ. We define the matrix-valued schedule

$$
M_t(\theta) := \sum_{j=1}^J g_j(t;\theta)\, P_j, \tag{29}
$$

so that $\partial_t M_t(\theta) = \sum_{j=1}^J \dot{g}_j(t;\theta) P_j \succ 0$ whenever $\dot{g}_j > 0$ for all j. This construction therefore satisfies the monotonicity constraint in (3) automatically. Because the P_j are mutually orthogonal projectors, any (scalar) matrix function f acts *subspace-wise*:

$$
f(M_t(\theta)) = \sum_{j=1}^J f(g_j(t;\theta))\, P_j. \tag{30}
$$

In particular, the quantities needed by our loss and sampler ($M_t^{\pm 1/2}$, M_t^{-1}, and $(I + M_t)^{-1/2}$) reduce to inexpensive scalings on each subspace. This also matches the solver structure in Section 5, where updates depend on increments of $M_t^{1/2}$; under (29), these increments are simply subspace-wise scalar increments.

The form of (29) separates:

- **Directions / subspaces** (the projectors $\{P_j\}$, determined by a basis choice and a grouping), and
- **Time allocation** (the scalar schedules $\{g_j(\cdot;\theta)\}$).

This separation lets us trade off interpretability and expressivity while keeping training and sampling efficient.

C.2 Schedule Families Used in Experiments

We now list the concrete instances of (29) used in our experiments in Section 6.

(a) Isotropic schedule (warm-up / special case). Standard isotropic diffusion corresponds to $J = 1$ with $P_1 = I$, so $M_t(\theta) = g(t;\theta)I$. This serves as a sanity check and warm-up baseline; it also emphasizes that our framework strictly generalizes scalar schedule learning.

(b) Fixed structured basis (DCT subspaces). For images, we often choose a fixed orthonormal transform $U \in \mathbb{R}^{d\times d}$ exposing interpretable directions (e.g. a 2D-DCT basis; see Appendix E.5.1). Partition the basis indices into J groups defining subspaces $S_1, \dots, S_J$ (ordered by increasing frequency), and let P_j be the projector onto S_j. A particularly simple instance is $J = 2$: S_1 is a low-frequency subspace and $S_2 = S_1^{\perp}$ its complement, giving

$$M_t(\theta) = g_1^{\mathrm{DCT}}(t;\theta)P_1 + g_2^{\mathrm{DCT}}(t;\theta)P_2.$$

This parameterization yields an interpretable notion of anisotropy (coarse-to-fine denoising) while remaining cheap.

(c) Class-conditional PCA bases (data-dependent subspaces, shared schedules). To incorporate data geometry, we can make the *basis* depend on a class label $y \in \{1, \dots, C\}$ (in the case of ImageNet and CIFAR10). For each class y, compute a PCA basis from training samples in that class and form mutually orthogonal subspaces $\{S_j^{(y)}\}_{j=1}^{J}$ that partition $\mathbb{R}^d$ (e.g. $S_1^{(y)}$ spans the top-k PCs and the remaining subspaces capture residual directions). Let $P_j^{(y)}$ be the projector onto $S_j^{(y)}$. We then define

$$M_t(\theta; y) = \sum_{j=1}^{J} g_j^{\mathrm{PCA}}(t;\theta)\, P_j^{(y)}.$$

Here the *subspaces* vary with y while the scalar schedules are shared across classes, isolating the effect of a class-dependent geometry.

(d) Shared structured basis + class-conditional schedules. For class-conditional generation with a score network $\mathrm{net}(x, t, y; \phi)$, we can instead keep the projectors fixed (e.g. DCT projectors $\{P_j\}$) and make the schedules class-specific:

$$M_t(\theta; y) = \sum_{j=1}^{J} g_{j,y}^{\mathrm{DCT}}(t;\theta)\, P_j.$$

Even in the simple $J = 2$ case this yields $2C$ coupled scalar trajectories, making black-box search over θ impractical and motivating our gradient-based optimization of θ.

(e) Class-conditional PCA bases + class-conditional schedules (most expressive). Finally, we can let *both* the subspaces and schedules depend on class:

$$M_t(\theta; y) = \sum_{j=1}^{J} g_{j,y}^{\mathrm{PCA}}(t;\theta)\, P_j^{(y)}.$$

This is the largest design space we consider. It can align anisotropic diffusion dynamics with class-dependent data geometry, but it also highlights why an efficient variational optimization scheme is needed.

Discussion: expressivity vs. efficiency. Across (a)–(e), the template (29) ensures that all matrix operations required by the trajectory loss and the anisotropic solvers reduce to subspace-wise scalings via (30). Increasing expressivity corresponds to increasing J, allowing class dependence, or making the basis data-dependent (PCA), while retaining tractable computation.

C.3 Schedule Variants Evaluated

We compare the following schedule families (see Section C for the corresponding parameterizations):

1. **Class-conditional isotropic schedule $\{g_y\}_{y=1}^{C}$.** A separate scalar schedule per class: $M_t(y) = g_y(t;\theta)I$.

2. **DCT anisotropic schedule** $(g_1^{\mathrm{DCT}}, g_2^{\mathrm{DCT}})$**.** A two-subspace matrix schedule with $J = 2$ in a fixed 2D-DCT basis (Appendix E.5.1), where the "low-frequency" subspace consists of the $(H/2) \times (H/2)$ lowest-frequency DCT modes (i.e., $H^2/4$ basis vectors) and the second subspace is its orthogonal complement: $M_t = g_1^{\mathrm{DCT}}(t;\theta)P_1 + g_2^{\mathrm{DCT}}(t;\theta)P_2$.
3. **PCA anisotropic schedule** $(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$**.** A two-subspace matrix schedule with $J = 2$ using class-conditional PCA bases (subspaces depend on class, schedules are shared), as described in Section C.
4. **Class-conditional DCT anisotropic schedules** $\{(g_{1,y}^{\mathrm{DCT}}, g_{2,y}^{\mathrm{DCT}})\}_{y=1}^{C}$**.** Fixed DCT subspaces but class-specific scalar schedules.
5. **Class-conditional PCA anisotropic schedules** $\{(g_{1,y}^{\mathrm{PCA}}, g_{2,y}^{\mathrm{PCA}})\}_{y=1}^{C}$**.** The most expressive conditional variant we consider: both PCA subspaces and schedules depend on class.

Conditional vs. unconditional datasets. We consider CIFAR-10 and ImageNet-64 as class-conditional datasets. For CIFAR-10, which is relatively simple, we first train isotropic schedules, including both a global isotropic schedule and class-conditional isotropic schedules, as warm-up ablations. Building on this, we separately study the effects of class-conditional schedules and class-conditional bases. For ImageNet-64, due to its higher visual complexity, we directly adopt class-conditional isotropic schedules during warm-up. We then evaluate more expressive variants, including class-conditional schedules, class-conditional bases, and their combination. For AFHQv2 and FFHQ, which do not provide class labels, we primarily train structured-basis anisotropic schedules based on a fixed DCT decomposition with a global isotropic schedule g as an ablation for warm-up.

D Experiment Details

D.1 Subspace Schedule Parameterization g_i

For a fixed set of m node locations $0 = \tau_0 < \tau_1 < \cdots < \tau_{m-1} = T$, the subspace noise schedule is defined in log–space between the smallest and largest variance values $g(0) = g_0$ and $g(T) = T$. The trainable parameters $\theta = (\theta_1, \ldots, \theta_{m-1})$ define a collection of strictly positive increments

$$s_i = \mathrm{softplus}(\theta_i), \qquad i = 1, \ldots, m-1.$$

These increments are then rescaled by α so that their sum exactly matches the total log–gap between the endpoints,

$$\alpha = \frac{\log T - \log g_0}{\sum_{i=1}^{m-1} s_i}.$$

The log–values at the nodes are constructed by cumulative summation,

$$\ell_0 = \log g_0, \qquad \ell_j = \ell_0 + \sum_{i=1}^{j} \alpha s_i, \quad j = 1, \ldots, m-1,$$

so that $\ell_{m-1} = \log T$.

Given a time $t \in [0, T]$, one locates the enclosing interval $[\tau_{j-1}, \tau_j]$ and computes the normalized position

$$p(t) = \frac{t - \tau_{j-1}}{\tau_j - \tau_{j-1}}.$$

The value of $g(t)$ is then obtained by linearly interpolating between successive log–nodes and exponentiating:

$$\log g(t) = (1 - p(t))\ell_{j-1} + p(t)\ell_j, \qquad g(t) = \exp\big(\log g(t)\big).$$

Within each interval the derivative takes the simple form

$$g'(t) = g(t)\frac{\ell_j - \ell_{j-1}}{\tau_j - \tau_{j-1}}.$$

D.2 Training Setup

Initialization. All experiments are initialized from the corresponding pretrained EDM checkpoints. We load both raw network weights and their exponential moving average (EMA) counterparts. When learning diffusion schedules, we initialize schedule parameters from a pretrained isotropic schedule model. For conditional variants, schedule parameters are instantiated per class. If class-wise isotropic schedules are available, they are used for initialization; otherwise, a shared global schedule is copied across classes.

Joint optimization of network and schedule. We jointly optimize network parameters ϕ and schedule parameters θ using Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$). Network and schedule parameters use separate learning rates. Gradients are accumulated over multiple micro-batches to achieve the target effective batch size. After backpropagation, gradients are explicitly sanitized by replacing NaNs and infinities with finite values to ensure numerical stability.

For class-conditional schedules, schedule updates are applied only to classes present in the current batch; in distributed settings, we synchronize the set of active classes across workers before applying optimizer steps to maintain consistent parameter updates.

Learning-rate schedule. We employ a linear warm-up for the first K_{ramp} kimg. After warm-up, the learning rate follows a mild decay schedule, with a decay factor applied after the midpoint of training. The learning rate of schedule parameters scales proportionally to the network learning rate.

EMA update. We maintain an exponential moving average of network parameters throughout training. The EMA half-life is specified in kimg units. Optionally, we apply an EMA ramp-up, where the effective half-life is bounded by the number of images seen so far during early training. EMA parameters are synchronized across workers in distributed settings.

Distributed training. For large-scale experiments (e.g., ImageNet-64), we use PyTorch Distributed Data Parallel (DDP) with NCCL backend. The global batch is evenly partitioned across workers. All parameters are broadcast from rank 0 at initialization. Gradients of network parameters are synchronized via DDP. Schedule parameters are explicitly all-reduced across workers, including zero-filled gradients for classes absent on a given worker, to guarantee consistent updates across GPUs.

Training budget and checkpoints. Across all datasets, fine-tuning processes approximately 1.2M images in total. We periodically save raw network weights, EMA weights, and schedule parameters. Unless otherwise specified, evaluation is performed using the final EMA checkpoint.

D.3 Evaluation protocol details

NFE counting convention. Both Heun variants evaluate the network twice per step. EDM uses the *endpoint variant* $\hat{t}_k = t_{k-1}$ and skips the second-order correction on its final step to $\sigma = 0$, so N steps cost NFE $= 2N - 1$ [3]. Our matrix Heun solver uses the *midpoint variant* $\hat{t}_k = (t_{k-1} + t_k)/2$; since $\hat{t}_k > 0$ on every step, all steps, including the last, use two evaluations. At each paired budget in Table 1, we run $N - 1$ midpoint steps, giving NFE $= 2N - 2$, one fewer than EDM. Throughout the paper, we report each method under its own counting convention.

D.4 Composing Anisotropic Schedules with DPM-Solver-v3

This appendix explains how our learned anisotropic schedule $M_t(\theta)$ composes with off-the-shelf scalar diffusion solvers such as DPM-Solver and DPM-Solver-v3 [20, 11], yielding the *Ours + DPM-v3* sampler reported in Table 2 (right).

D.4.1 Subspace decoupling of the anisotropic dynamics

Recall the projector-based parameterization (Section C):

$$M_t(\theta) = \sum_{j=1}^{J} g_j(t;\theta)\, P_j, \qquad P_i P_j = \delta_{ij} P_j, \qquad \sum_j P_j = I. \tag{31}$$

Let V_j be an orthonormal basis spanning $\mathrm{range}(P_j)$, so that $P_j = V_j V_j^\top$, and let $V := [V_1 \cdots V_J]$ collect the bases. The forward dynamics $dx_t = (\partial_t M_t)^{1/2}\, dB_t$ commutes with V in the sense that the projected coordinate $z_t^{(j)} := V_j^\top x_t$ satisfies a *scalar* variance-exploding diffusion within subspace j:

$$dz_t^{(j)} = \big(\partial_t g_j(t;\theta)\big)^{1/2} dB_t^{(j)}, \qquad z_t^{(j)} \mid z_0^{(j)} \sim \mathcal{N}\big(z_0^{(j)},\, g_j(t;\theta)\, I_{d_j}\big), \tag{32}$$

where $d_j = \mathrm{rank}(P_j)$ and $B^{(j)}$ are independent Brownian motions. The effective scalar noise level in subspace j is therefore $\sigma^{(j)}(t)^2 = g_j(t;\theta)$.

D.4.2 Per-subspace solver application

Equation (32) states that, in the V-frame, our anisotropic process is exactly J *independent* scalar VE diffusions sharing a common time variable. Consequently, any scalar diffusion solver (e.g., DPM-Solver, DPM-Solver++, DPM-Solver-v3, or EDM Heun) applies within each subspace using the corresponding scalar schedule $g_j(t;\theta)$. Concretely, given a learned score field $\nabla_x \log p_t(x;\theta)$ approximated by the trained network, one DPM-Solver-v3 step within subspace j uses:

- the scalar schedule $\sigma^{(j)}(t) = g_j(t;\theta)^{1/2}$ to construct the solver's internal log-SNR grid and update coefficients;
- the projected score $V_j^\top \nabla_x \log p_t(x;\theta)$ as the solver's denoiser output;
- the projected iterate $V_j^\top x$ as the state being updated.

The composed sampler OURS + DPM-V3 is then simply: (i) project the iterate into the V-frame, (ii) take a DPM-Solver-v3 step in each subspace using $g_j(t;\theta)$, (iii) project back. The procedure inherits the per-step cost of one network call (which produces the score field $\nabla_x \log p_t$) plus inexpensive projections onto $\{V_j\}$.

D.4.3 Implementation note: *scalarized* mode

Our implementation realizes the per-subspace step in a particularly lightweight way that we refer to as the *scalarized* mode (used in our codebase as `dpmv3_official_scalar_eff`):

1. evaluate the score network on the un-projected iterate x, yielding $s := \nabla_x \log p_t(x;\theta)$;
2. form the DPM-Solver-v3 update independently for each subspace j using the scalar schedule $\sigma^{(j)}(t) = g_j(t;\theta)^{1/2}$ and the projected score $V_j^\top s$;
3. assemble the next-step iterate by summing contributions across subspaces, $x' = \sum_j V_j (V_j^\top x'_j)$, where x'_j is the per-subspace update.

This avoids modifying the score network's input space and allows direct reuse of the EDM-style network used by our matrix Heun solver. The reduction is exact under (31) because the dynamics, the marginals, and the score all decouple along V; "scalarized" refers only to the fact that the solver's internal step computation is performed using the scalar effective $\sigma^{(j)}(t)$ rather than the full matrix $M_t^{1/2}$.

D.4.4 When to use which solver

In Table 2 (right), *Ours + DPM-v3* dominates at NFE ≤ 8, while our matrix Heun solver of Section 5 dominates at NFE ≥ 34. The crossover reflects standard tradeoffs in diffusion sampling: DPM-Solver-v3 uses higher-order extrapolation tuned for very small step counts, whereas the second-order Heun solver becomes more accurate as the discretization becomes finer. For practitioners, we recommend the composed sampler for fast inference and the matrix Heun solver when fidelity is the priority.

E Additional Experimental Analysis

E.1 Per-Variant FID Breakdown

Table 3 provides the full per-variant breakdown across all evaluated NFE budgets. Rows are labeled by schedule family (Appendix C.3):

- g: global isotropic schedule;
- $\{g_y\}_{y=1}^{C}$: class-conditional isotropic schedules;
- $(g_1^{\mathrm{DCT}}, g_2^{\mathrm{DCT}})$: two-subspace DCT anisotropic schedule;
- $(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$: two-subspace PCA anisotropic schedule with class-conditional bases and shared schedules;
- $\{(g_{1,y}^{\mathrm{DCT}}, g_{2,y}^{\mathrm{DCT}})\}_{y=1}^{C}$: class-conditional DCT schedules;
- $\{(g_{1,y}^{\mathrm{PCA}}, g_{2,y}^{\mathrm{PCA}})\}_{y=1}^{C}$: class-conditional PCA bases and schedules.

CIFAR-10								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	17/16	35/34	59/58	79/78
EDM	35.52 (0.0448)	14.37 (0.0387)	6.694 (0.0563)	4.231 (0.0462)	3.027 (0.0436)	1.829 (0.0162)	1.868 (0.0127)	1.890 (0.0112)
g	4.241 (0.0590)	3.087 (0.0163)	2.650 (0.0120)	2.431 (0.0177)	2.328 (0.0166)	**1.815** (0.0305)	1.820 (0.0299)	1.835 (0.0302)
$\{g_y\}_{y=1}^{C}$	3.691 (0.0165)	3.090 (0.0159)	2.735 (0.0112)	2.582 (0.0069)	2.480 (0.0096)	1.831 (0.0145)	1.829 (0.0081)	1.841 (0.0080)
$\{(g_{1,y}^{\mathrm{DCT}}, g_{2,y}^{\mathrm{DCT}})\}_{y=1}^{C}$	**3.618** (0.0202)	3.056 (0.0152)	2.681 (0.0118)	2.535 (0.0061)	2.431 (0.0065)	1.829 (0.0120)	1.810 (0.0072)	1.818 (0.0078)
$(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$	3.839 (0.0661)	**2.900** (0.0243)	**2.538** (0.0112)	**2.332** (0.0029)	**2.213** (0.0062)	1.851 (0.0113)	**1.803** (0.0103)	**1.803** (0.0138)
AFHQv2								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	27.98 (0.0544)	13.66 (0.0778)	7.587 (0.0729)	4.746 (0.0486)	2.986 (0.0359)	2.075 (0.0167)	2.042 (0.0175)	2.046 (0.0173)
$(g_1^{\mathrm{DCT}}, g_2^{\mathrm{DCT}})$	**4.524** (0.0210)	**2.977** (0.0195)	**2.513** (0.0108)	**2.401** (0.0026)	**2.265** (0.0134)	**2.070** (0.0094)	**2.015** (0.0095)	**2.010** (0.0255)
FFHQ								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	57.14 (0.0774)	29.39 (0.0497)	15.81 (0.0857)	9.769 (0.0853)	5.169 (0.0480)	2.575 (0.0352)	2.391 (0.0374)	2.374 (0.0381)
$(g_1^{\mathrm{DCT}}, g_2^{\mathrm{DCT}})$	**4.967** (0.0400)	**3.776** (0.0321)	**3.304** (0.0263)	**3.011** (0.0351)	**2.711** (0.0272)	**2.350** (0.0282)	**2.297** (0.0331)	**2.318** (0.0336)
ImageNet-64								
NFE (EDM/Ours)	9/8	11/10	13/12	15/14	19/18	39/38	79/78	119/118
EDM	35.32 (0.2144)	15.19 (0.0894)	8.160 (0.0191)	5.458 (0.0468)	3.586 (0.0533)	2.428 (0.0453)	2.294 (0.0350)	2.276 (0.0326)
$\{g_y\}_{y=1}^{C}$	8.416 (0.0574)	5.907 (0.0224)	4.749 (0.0364)	4.079 (0.0347)	3.366 (0.0327)	2.455 (0.0090)	2.314 (0.0116)	2.296 (0.0095)
$(g_1^{\mathrm{PCA}}, g_2^{\mathrm{PCA}})$	29.47 (0.1691)	15.55 (0.1727)	8.558 (0.1230)	5.666 (0.0867)	3.409 (0.0619)	**2.362** (0.0498)	2.263 (0.0446)	2.240 (0.0464)
$\{(g_{1,y}^{\mathrm{PCA}}, g_{2,y}^{\mathrm{PCA}})\}_{y=1}^{C}$	8.366 (0.0492)	5.823 (0.0545)	4.626 (0.0786)	3.992 (0.0564)	3.282 (0.0575)	2.378 (0.0539)	**2.259** (0.0416)	2.239 (0.0396)
$\{(g_{1,y}^{\mathrm{DCT}}, g_{2,y}^{\mathrm{DCT}})\}_{y=1}^{C}$	**8.340** (0.0723)	**5.808** (0.0601)	**4.615** (0.0833)	**3.983** (0.0592)	**3.275** (0.0586)	2.377 (0.0546)	**2.259** (0.0414)	**2.238** (0.0412)

Table 3: FID ↓ vs. NFE across datasets (50k samples). See Appendix C.3 for a description of each schedule variant. Each entry reports the minimum FID across 3 independent random generations of 50k images, with the standard deviation shown in parentheses.

E.2 Learned Schedules Across Datasets

Figures 2–5 visualize the learned schedules across datasets, subspace parameterizations, and conditioning schemes.

First, across all datasets (Figures 2–4), the learned anisotropic ratios deviate systematically from one over a substantial portion of the diffusion horizon, confirming that the two subspaces are not treated symmetrically by the learned schedule. However, the magnitude and shape of this deviation depend strongly on the dataset. For CIFAR-10 (Figure 2(b)) and AFHQv2 (Figure 3), the anisotropy ratio exhibits a smooth mid-horizon amplification before returning to one near the endpoints, suggesting that anisotropy is primarily active during intermediate diffusion times. In contrast, for FFHQ (Figure 4) the ratio shows a more pronounced early amplification followed by a gradual decay, indicating stronger low-frequency emphasis at early noise levels.

Second, on the class-conditional datasets (CIFAR-10 and ImageNet-64; Figures 2 and 5), class-conditional training induces clear inter-class variability in both isotropic and anisotropic schedules. In CIFAR-10 (Figure 2(c,d)), the class-conditional isotropic schedules display substantial dispersion around the geometric mean, and the anisotropic schedules further amplify these differences. On ImageNet-64 (Figure 5(a–f)), this effect becomes more structured: while class-conditional isotropic schedules vary moderately, the anisotropic variants reveal consistent class-dependent shifts in both geometric means and anisotropy ratios. Notably, the PCA-based conditional schedules in Figure 5(d–f) exhibit tighter concentration than their DCT counterparts (Figure 5(b,c)), suggesting that class-adaptive subspaces reduce unnecessary variability.

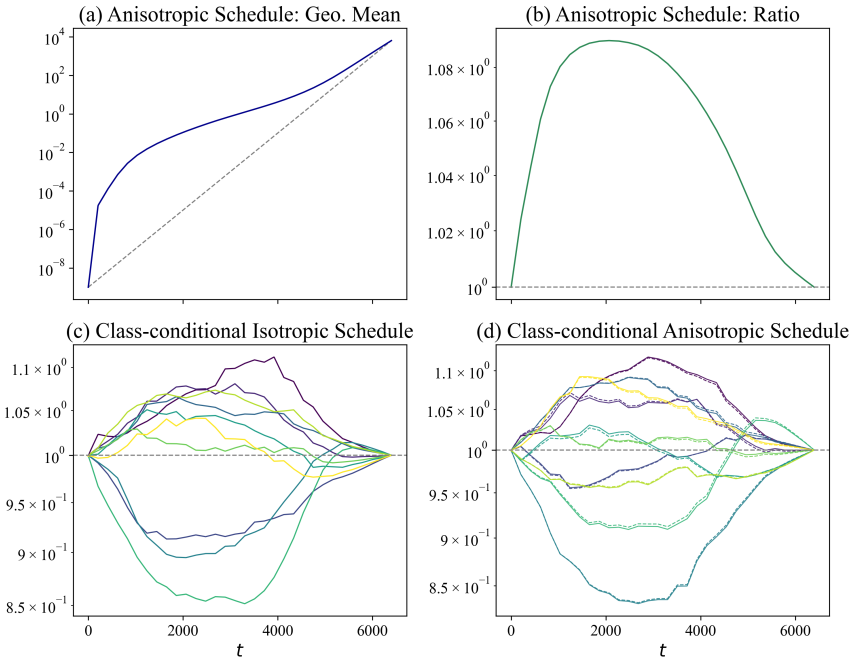

Figure 2: **CIFAR-10 learned schedule analysis.** (a) PCA-based geometric mean $\sqrt{g_1^{\mathrm{PCA}}(t) g_2^{\mathrm{PCA}}(t)}$ with a log-linear reference (gray dashed; log y-axis). (b) PCA anisotropy ratio $g_1^{\mathrm{PCA}}(t)/g_2^{\mathrm{PCA}}(t)$. (c) Class-conditional isotropic schedules $g_y(t)/\bar{g}(t)$, where $\bar{g}(t) = \left(\prod_{y=1}^{C} g_y(t)\right)^{1/C}$ with $C = \#\mathrm{class} = 10$. (d) Class-conditional anisotropic schedules over DCT subspaces: $g_{k,y}^{\mathrm{DCT}}(t)/\bar{g}_k(t)$ for $k \in \{1, 2\}$ (solid/dashed), where y indexes classes and $\bar{g}_k(t)$ is the geometric mean across classes.

E.3 Training and Inference Speed

We report wall-clock training and inference times for our method versus the EDM baseline on a single NVIDIA H200 GPU (141 GB HBM3e memory, ∼4.8 TB/s memory bandwidth). The large memory capacity is useful here because our schedule-gradient estimator (Theorem 4.1) materializes

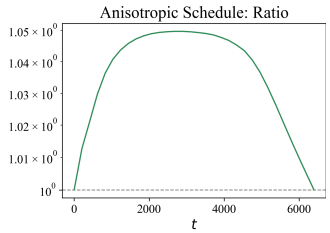

Figure 3: **AFHQv2 learned schedule analysis.** Ratio between the two learned DCT-based schedules $g_1^{\mathrm{DCT}}(t)$ and $g_2^{\mathrm{DCT}}(t)$ over time.

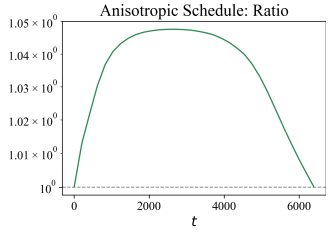

Figure 4: **FFHQ learned schedule analysis.** Ratio between the two learned DCT-based schedules $g_1^{\mathrm{DCT}}(t)$ and $g_2^{\mathrm{DCT}}(t)$ over time.

higher-order directional derivatives on top of the standard score-network forward/backward, which increases peak memory pressure relative to vanilla EDM training.

Training. Our method increases wall-clock training time relative to EDM under the matched 1.2M-image fine-tuning protocol; the slowdown comes mainly from the schedule optimization step, which adds higher-order directional-derivative computations and the corresponding backward passes. Per-dataset numbers are reported in Table 4 (top).

Inference. At inference time the per-step overhead is small. The dominant cost remains score-network evaluations; the additional per-step work is structured matrix operations involving $M_t(\theta)$ and related terms. Under the projector-based parameterizations of Section C, these reduce to inexpensive subspace-wise scalings and avoid explicit dense $d\times d$ matrix square roots. Per-dataset numbers (50k samples, batch size 64, NFE $= 35$) are reported in Table 4 (bottom).

Training (1.2M-image fine-tuning, single H200)		
Dataset	EDM fine-tuning time	Our fine-tuning time
CIFAR-10	58m	2h 20m
AFHQv2	1h 53m	8h
FFHQ	1h 53m	8h
ImageNet-64	5h 04m	35h
Inference (50k samples, batch size 64, NFE $= 35$, single H200)		
Dataset	EDM	Ours
CIFAR-10	14m	14m
AFHQv2	25m	27m
FFHQ	25m	27m
ImageNet-64	49m	58m

Table 4: Wall-clock training and inference times on a single NVIDIA H200 GPU. **Top:** fine-tuning from the released EDM checkpoint for 1.2M additional images. **Bottom:** sampling 50k images at NFE $= 35$ with batch size 64. Inference cost is essentially unchanged from EDM, while training cost is 2–7$\times$ higher due to the schedule-gradient estimator.

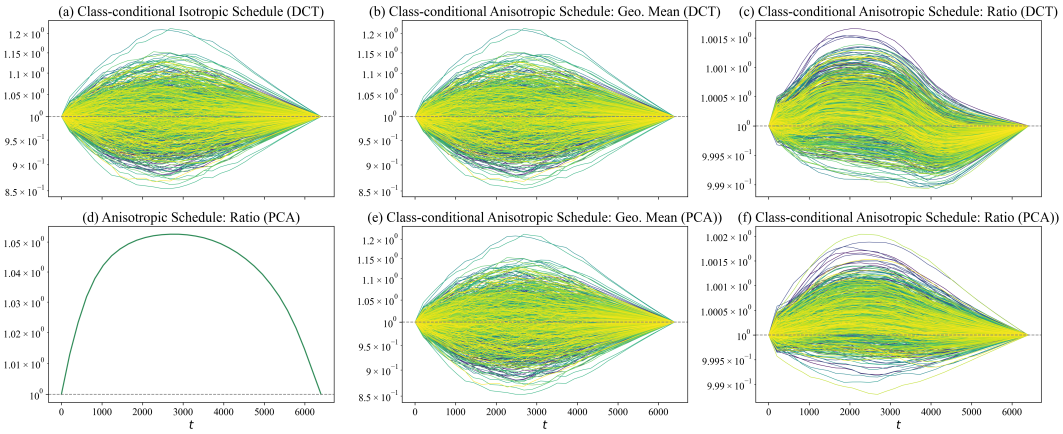

Figure 5: **ImageNet-64 learned schedule analysis.** (a) Class-conditional isotropic schedules shown as $g_y(t)/\bar{g}(t)$, where $\bar{g}(t)$ denotes the geometric mean across classes. (b) Class-conditional anisotropic schedules summarized by the geometric mean $\sqrt{g_{1,y}^{\mathrm{DCT}}(t)g_{2,y}^{\mathrm{DCT}}(t)}$ and normalized by its class-wise geometric mean. (c) Class-conditional anisotropy ratios $g_{1,y}^{\mathrm{DCT}}/g_{2,y}^{\mathrm{DCT}}$. (d) PCA-based (class-conditional basis) anisotropic schedule ratio $g_{1,y}^{\mathrm{PCA}}(t)/g_{2,y}^{\mathrm{PCA}}(t)$. (e) PCA-based (class-conditional basis) class-conditional anisotropic schedules shown as $\sqrt{g_{1,y}^{\mathrm{PCA}}(t)g_{2,y}^{\mathrm{PCA}}(t)}$ normalized by the geometric mean across classes. (f) PCA-based (class-conditional basis) class-conditional anisotropy ratios $g_{1,y}^{\mathrm{PCA}}(t)/g_{2,y}^{\mathrm{PCA}}(t)$. All ratios are plotted on a logarithmic y-axis, and dashed horizontal lines indicate the reference value 1.

E.4 Fine-Tuning Controls with Matched Image Budgets

A natural concern is whether our consistent FID improvement over EDM in Table 1 could be partially attributable to the extra 1.2M-image fine-tuning consumed by our schedule-learning protocol (Section D.2), rather than to the learned anisotropic schedule itself. To rule this out, we run two complementary checks on **ImageNet-64**, the largest and most challenging of our benchmarks, and therefore the dataset where additional training has the greatest a priori chance of further reducing EDM's FID. The other datasets are smaller and closer to the convergence regime, so the conclusions transfer *a fortiori*.

(a) Fine-tuning EDM with its own recipe. We fine-tune the released EDM checkpoint for 1.2M further images using EDM's own intended training recipe (isotropic schedule, original optimizer, original loss weighting), and re-evaluate FID under the standard EDM sampling protocol. As shown in Table 5, the extended-training EDM achieves FID values within run-to-run variance of the original EDM numbers reproduced in Table 3, confirming that the released checkpoint is already well converged.

ImageNet-64 (EDM + 1.2M extra fine-tuning images)								
NFE	9	11	13	15	19	39	79	119
FID	35.39 (0.1752)	15.22 (0.0715)	8.182 (0.0149)	5.473 (0.0373)	3.594 (0.0427)	2.430 (0.0365)	2.297 (0.0286)	2.278 (0.0264)

Table 5: EDM fine-tuning control on ImageNet-64 with a matched image budget. We fine-tune the released EDM checkpoint for an additional 1.2M images using EDM's own training recipe and re-evaluate FID across the same NFE budgets as Table 1. The values are within run-to-run variance of the original EDM numbers (cf. Table 3, ImageNet-64 row), indicating that the released checkpoint is already well converged and that the improvements reported in our main results stem from the learned anisotropic schedule rather than from additional compute. Each FID is the minimum across 3 independent generations of 50k images; standard deviations are shown in parentheses.

(b) Fine-tuning EDM with our hyperparameters. A complementary concern is that our specific training hyperparameters (optimizer state, learning-rate schedule, EMA decay) might themselves be responsible for the improvement. We therefore fine-tune EDM for 1.2M images using *our* hyperparameters but with the schedule held fixed at EDM's isotropic baseline, and re-evaluate FID. As shown in Table 6, the values are again within run-to-run variance of the original EDM, ruling out hyperparameter-induced gains.

ImageNet-64 (EDM + 1.2M images, our hyperparameters, isotropic schedule)								
NFE	9	11	13	15	19	39	79	119
FID	35.35 (0.1767)	15.19 (0.0714)	8.164 (0.0139)	5.456 (0.0383)	3.585 (0.0433)	2.428 (0.0365)	2.294 (0.0289)	2.276 (0.0261)

Table 6: Hyperparameter-matched EDM sanity check on ImageNet-64. We fine-tune the released EDM checkpoint for 1.2M images using *our* training hyperparameters (optimizer, learning-rate schedule, EMA, batch size) while keeping the schedule isotropic, and re-evaluate FID. Values are within run-to-run variance of both the original EDM numbers (Table 3, ImageNet-64 row) and the EDM-recipe sanity check (Table 5), ruling out hyperparameter-induced gains. Each FID is the minimum across 3 independent generations of 50k images; standard deviations are shown in parentheses.

E.5 Subspace Bases

We use two families of orthonormal subspace bases throughout the paper: a fixed structured basis (2D-DCT) and a data-adaptive class-conditional basis obtained from PCA. This section defines the 2D-DCT basis (Section E.5.1) and analyzes the geometric structure of the learned PCA bases (Section E.5.2).

E.5.1 2D-DCT Bases

Let H be the image side length and $d = H^2$. The two-dimensional DCT (type-II) basis over $\mathbb{R}^{H\times H}$ is defined as follows. For each pair $(p, q) \in \{0, \dots, H-1\}^2$, the basis is

$$\Phi_{p,q}(x, y) = \gamma_p \gamma_q \cos\Big(\tfrac{(2x+1)p\pi}{2H}\Big) \cos\Big(\tfrac{(2y+1)q\pi}{2H}\Big),$$
$$x, y = 0, \dots, H-1.$$

with normalization factors

$$\gamma_p = \begin{cases} H^{-1/2}, & p = 0, \\ \sqrt{2}H^{-1/2}, & p > 0, \end{cases} \qquad \gamma_q = \begin{cases} H^{-1/2}, & q = 0, \\ \sqrt{2}H^{-1/2}, & q > 0. \end{cases}$$

Vectorizing each $\Phi_{p,q}$ into $\mathbb{R}^d$ and enumerating them yields the orthonormal basis $\{v_1, \dots, v_{H^2}\}$, which are the 2D-DCT basis of $\mathbb{R}^d$. Please refer to Figure 6 for example 2D-DCT bases.

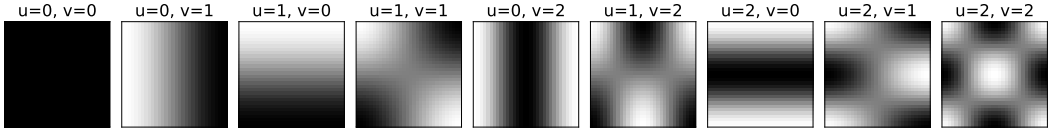

Figure 6: The first nine 2D-DCT bases ordered by increasing frequency.

E.5.2 PCA Bases

To analyze the geometry of the learned PCA subspaces, we first visualize the leading principal components for representative classes (cat, dog, airplane) in Figure 7. Each row shows the top principal directions reshaped into spatial patterns. We observe that semantically related classes (e.g., cat and dog) exhibit qualitatively similar dominant spatial structures, whereas structurally distinct classes (e.g., airplane) show noticeably different patterns.

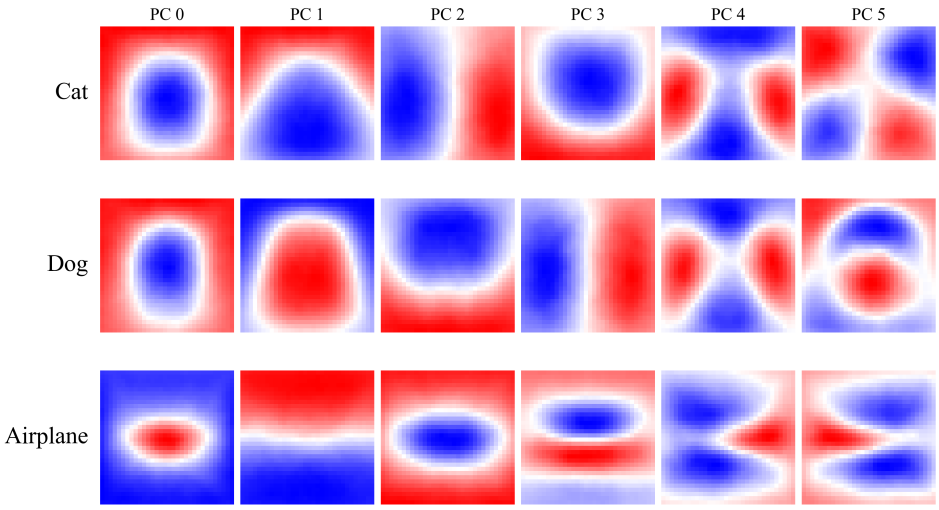

Figure 7: **Class-specific PCA subspaces.** Leading principal components for representative CIFAR-10 classes (cat, dog, airplane), reshaped as spatial patterns. Semantically related classes (e.g., cat and dog) exhibit qualitatively similar dominant structures, whereas visually distinct classes (e.g., airplane) display different principal directions.

To quantify cross-class similarity, we compute pairwise distances between subspaces using the Frobenius norm of projector differences, i.e.,

$$d\left(Q_1, Q_2\right) = \frac{\left\|Q_1 Q_1^\top - Q_2 Q_2^\top\right\|_F}{\sqrt{2k}},$$

where Q_1, Q_2 are orthonormal bases. We then apply classical multidimensional scaling (MDS) to embed this distance matrix into two dimensions for visualization. MDS constructs a low-dimensional embedding whose pairwise Euclidean distances best approximate the original subspace distances. Figure 8 reveals clear semantic clustering, with animal classes grouped together and separated from vehicle classes.

These observations suggest that the learned PCA subspaces capture semantically meaningful geometric structure.

F Broader Impacts

This work is methodological: it improves sample quality and few-step generation in diffusion models. Positive impacts include reduced compute and energy cost for high-quality generation, particularly in the low-NFE regime. Dual-use concerns common to all generative-model improvements apply, i.e., higher-fidelity samplers can be misused for deepfakes, non-consensual imagery, or disinformation. Our experiments use only publicly available datasets and fine-tune released EDM checkpoints; we release no new datasets or models trained on novel sources.

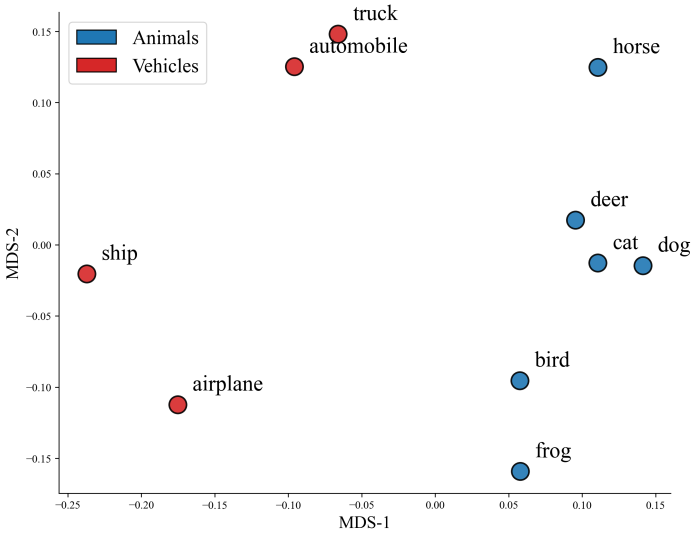

Figure 8: **Subspace distance embedding via classical MDS.** Pairwise distances between class-specific PCA subspaces are computed using the Frobenius norm between orthogonal projectors. Classical multidimensional scaling (MDS) embeds the resulting distance matrix into two dimensions. Animal classes cluster together and are separated from vehicle classes, indicating that the learned subspaces capture semantically structured geometry.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are stated in the abstract and introduction, and are supported by the theory in Sections 3–5 and experiments in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the additional fine-tuning compute cost in Appendix E.3 and dual-use concerns in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The main theoretical results are stated with assumptions, and proofs are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the training and evaluation protocol, including fine-tuning budget, sample size, FID computation, repeated sampling, and NFE reporting.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We evaluate on public image-generation benchmarks: CIFAR-10, AFHQv2, FFHQ, and ImageNet-64. No new dataset is introduced. An code repository is provided for review and will be replaced by the public repository after acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We describe the datasets, training protocol, evaluation protocol, fine-tuning budget, sampling setup, and NFE definition in the paper, with additional details provided in the appendix and code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We repeat sampling three times with independent random seeds and report the minimum FID throughout, matching the EDM evaluation protocol. The appendix table additionally reports the standard deviation across the three runs to quantify run-to-run variability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the fine-tuning budget in image passes and describe the main evaluation cost from repeated 50k-sample generation. Additional hardware and implementation details are provided in the appendix/code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses standard public datasets and pretrained models, preserves anonymity, and does not involve private data or human-subject experiments.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive impacts (lower inference compute) and dual-use concerns are discussed in Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original datasets, pretrained EDM models, and baseline methods used in the experiments, and use them according to their intended research settings.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release code implementing the proposed schedules, training objective, and sampling/evaluation pipeline, with documentation provided in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use LLMs as an important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.