Learning When to Denoise: Asynchronous Schedules for Faster Multi-Representation Diffusion Training

Bingshuo Qian **Xiang Cheng**
Department of Electrical and Computer Engineering
Duke University
{bingshuo.qian, xiang.cheng}@duke.edu

Abstract

Multi-representation diffusion models can improve visual synthesis by denoising complementary views of an image, but their performance depends critically on the asynchronous schedule that determines when each representation is denoised. We propose to learn this schedule. Our method formulates asynchronous flow matching over multiple representation spaces and uses a schedule-corrected objective that keeps each representation's local noising-time weights fixed as the schedule changes. We instantiate the schedule with a flexible parametric class that is convex and monotone by construction, and learn it using a fast joint probe with less than 1% additional training compute. On ImageNet 256×256, the learned schedule substantially improves both convergence speed and final quality under a matched 675M-parameter XL backbone. With AutoGuidance, our 200-epoch model reaches FID 1.05, matching the 800-epoch SFD-XL baseline with $4\times$ less training. Training to 600 epochs achieves **FID 1.02, below the 1.04 of the 1B-parameter SFD-XXL while using fewer parameters.** In the unguided setting, our 200-epoch model reaches FID 2.37, already below the best 800-epoch SFD-XL result (2.54) at $4\times$ less training, and improves to FID 2.14 at 600 epochs. Code is available at https://github.com/bsq532087/LWD.

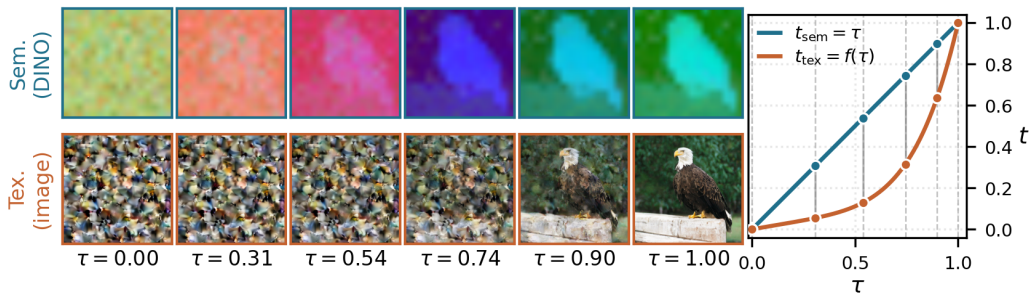


Figure 1: **Learning the relative denoising schedule between semantic and texture latents.** Along a shared global time τ, the semantic branch follows $t_{\text{sem}} = \tau$, while the texture branch follows a learned schedule $t_{\text{tex}} = f^\star(\tau) \leq \tau$. Since larger t denotes a cleaner latent, semantic structure is formed earlier and guides subsequent texture refinement. The dashed lines mark the trajectory states visualized on the left.

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

1 Introduction

Diffusion and flow-based generative models have become a standard recipe for high-fidelity visual synthesis. Much of their practical success comes from choosing a representation in which denoising is both expressive and efficient: latent diffusion reduces pixel-space cost by denoising compressed VAE latents [21], while diffusion transformers and flow-matching variants scale this recipe with large Transformer backbones [19, 15, 13].

A recent and complementary trend is to inject additional representational information into the generative process. For example, REPA aligns diffusion hidden states with clean features from pretrained visual encoders [27], VA-VAE aligns the autoencoder latent space with vision-foundation-model features [26], and FAE compresses pretrained visual features into compact latents for generation [7]. SFD [18] and Latent Forcing [1] improved generation quality by directly denoising over multiple different representations.

The success of multi-representation denoising motivates a broader question: at what rate should each representation be denoised? A synchronous schedule ties all representations to the same noise level at each sampling step. An asynchronous schedule instead allows different representations to occupy different noise levels at the same step. This flexibility is useful because different representations need not carry the same kind of information. A higher-level semantic representation may encode global structure useful for conditioning, while an image-decoding texture representation may be responsible for recovering fine visual detail.

Choosing the asynchronous schedule is therefore a central design problem. A good schedule must coordinate how information is revealed across representation spaces, while balancing how easily the flow can be learned against how easily the resulting ODE can be integrated. These requirements depend on the representation pair, model architecture, training objective, and sampler. SFD sets the schedule by hand: the semantic latent leads the texture latent by a fixed offset chosen by grid search, which freezes the texture latent at the start of sampling and the semantic latent at the end. Such a search does not extend to more flexible schedule classes or more representations, where selecting a good schedule becomes a difficult modeling problem rather than a minor hyperparameter choice.

In this paper, we propose a framework for jointly learning the asynchronous schedule alongside the flow network. We first formulate a unified loss for both the flow network and the asynchronous schedule, and then propose an efficient two-stage algorithm that optimizes the schedule while learning to match the flow. Our key contributions are summarized below:

Contributions.

1. **Asynchronous flow matching with learnable semantic–texture schedules.** We formulate flow matching for composite latents with separate semantic and texture representation groups, where each space follows its own local noising time while a single global time indexes the sampling trajectory. This allows general differentiable semantic–texture schedules. We theoretically characterize the ideal asynchronous flow as the conditional-mean vector field, which satisfies the continuity equation of the asynchronous interpolation, and equivalently as a group-wise transformation of the score of that interpolation (Theorem 3.1). This formulation makes the schedule itself an optimizable object, enabling flexible parameterizations beyond settings that can be selected by grid search.
2. **A schedule-corrected objective for flow and schedule learning.** We define a component-wise flow objective that plays two roles. With the schedule fixed, it learns the local asynchronous flow. With the denoiser adapting, it provides the signal for learning the schedule. We show that its population minimizer recovers the ideal local flows along the schedule (Theorem E.1). For schedule learning, we identify a confound: changing the schedule also changes the effective distribution over local noising times at which the flow-matching losses are evaluated. We prove a necessary and sufficient change-of-variables correction that exactly counteracts this effect in the objective, keeping the marginal weighting over each group's local noising time fixed as the schedule changes (Lemma 3.2). Finally, we add a kinetic-energy regularizer that favors discretization-friendly trajectories, which completes the schedule and denoiser objectives (Eqs. (17) and (18)).
3. **Constrained schedule class and efficient joint optimization.** We propose a parametric schedule class that enforces monotonicity, convexity, and semantic-leading behavior by construction, while allowing efficient evaluation of both the schedule and its derivatives. We also identify the need for

joint schedule–denoiser optimization: with a frozen denoiser, schedule updates are dominated by model–schedule mismatch (Section 3.5). We therefore introduce an efficient two-stage procedure (Algorithm 1) that jointly learns the schedule and the denoiser with less than 1% overhead in compute budget.

4. **Faster convergence and lower final FID.** On class-conditional ImageNet 256×256, with SFD's architecture, latent representation, auxiliary losses, sampler, and weak-model architecture unchanged, the learned schedule reaches the final quality of SFD-XL with about $4\times$ less training: unguided FID 2.37 at 1M iterations versus 2.54 at 4M, and 1.05 at 200 epochs versus 1.06 at 800 with AutoGuidance. It improves FID at every matched training budget, consistent with its smooth coupling being easier to learn than SFD's offset path with frozen intervals. At 3M iterations (600 epochs) it reaches FID 2.14 unguided and $\mathbf{1.02}$ with AutoGuidance, the lowest among all 675M-parameter models we compare against and below the 1.0B-parameter SFD-XXL (1.04). The same schedule-learning procedure also improves over SFD when the SemVAE semantic latent is replaced by DINO-PCA or CLIP-PCA features, suggesting that the benefit comes from learning the semantic–texture schedule rather than from a representation-specific trick.

2 Related Work

Semantic-First Diffusion. The most directly comparable prior work is Semantic-First Diffusion (SFD) [18], which also forms the basis of our experimental setup. SFD encodes each image into two latent groups, a texture latent from an image VAE and a semantic latent from a SemVAE compressing pretrained DINOv2 features, and trains a single diffusion transformer that denoises both groups asynchronously, with the semantic group leading by a fixed offset. We retain SFD's architecture, latent representation, weak-model architecture for AutoGuidance, and sampler, but replace this offset with a learned semantic-leading schedule.

Asynchronous and multi-representation denoising. Several other generative systems exploit the fact that different parts of a sample need not be denoised at the same rate. Diffusion Forcing assigns independent noise levels to different tokens, bridging next-token prediction and full-sequence diffusion in sequential domains [2]. Latent Forcing reorders the diffusion trajectory across latents and pixels with separately tuned schedules, letting latents form before high-frequency pixel content [1]. Like SFD, both fix the cross-representation schedule by hand or by a small low-dimensional sweep. Our work shares their goal of decoupled denoising rates, but learns a flexible semantic-leading schedule from data while keeping the representation setup and sampler fixed. Recent work also studies learned anisotropic schedules more broadly: Liu et al. [14] optimize matrix-valued noise schedules that allocate noise across subspaces. Our focus is the representation-aware latent setting: we learn a semantic-leading schedule for explicit semantic and texture groups, with a corrected objective that keeps local-time weighting invariant.

Representation-aware diffusion. A complementary line of work changes *which* representations the denoiser operates on. REPA aligns hidden states with pretrained visual features [27]; REPA-E extends this to end-to-end joint training of the VAE and diffusion model [12]; VA-VAE aligns the autoencoder latent space with vision-foundation-model features [26]; FAE compresses pretrained visual features into compact generative latents [7]; and further work incorporates semantic features through representation entanglement, joint image–feature synthesis, or representation autoencoding [24, 10, 28]. These methods improve *what* the denoiser sees under a single global denoising schedule. We are concerned instead with how different representations should evolve over denoising time once a multi-representation latent is given.

3 Method

3.1 Asynchronous Flow in Texture and Semantic Spaces

Semantic and texture spaces. We now consider a composite latent with two representation groups. The *texture* group is the image VAE latent: it is the latent that is decoded directly to pixels at the end of sampling. The *semantic* group contains higher-level visual information, such as features derived from DINOv2. In our main experiments, we follow Semantic-First Diffusion (SFD) [18]: the texture

group is the SFD image-VAE latent, and the semantic group is the SemVAE latent trained to compress pretrained DINOv2 features [17]. We also evaluate alternative semantic representations, including DINO-PCA and CLIP-PCA [20], in Section 4.4. We refer to SFD [18] for additional details on the representation construction. Let

$$x_1 = \left[x_1^{\text{tex}}, x_1^{\text{sem}}\right] \in \mathbb{R}^{d_{\text{tex}}+d_{\text{sem}}}, \qquad x_0 = \left[x_0^{\text{tex}}, x_0^{\text{sem}}\right] \in \mathbb{R}^{d_{\text{tex}}+d_{\text{sem}}}, \tag{1}$$

where x_1 is the encoded data latent and $x_0 \sim \mathcal{N}(0, I)$ is Gaussian noise with the same shape, drawn independently of x_1 and the class label c. We flatten spatial dimensions for notation, so $x_1^{\text{tex}}, x_0^{\text{tex}} \in \mathbb{R}^{d_{\text{tex}}}$ and $x_1^{\text{sem}}, x_0^{\text{sem}} \in \mathbb{R}^{d_{\text{sem}}}$.

Let $\mathcal{G} = \{\text{tex}, \text{sem}\}$ denote the set of representation groups. For each group $g \in \mathcal{G}$, let $t_g \in [0, 1]$ denote the *local time* for that group. We apply the standard linear flow-matching path (Eq. (21) in Appendix C.1) separately in each representation space; in our convention, $t_g = 0$ corresponds to $\mathcal{N}(0, I)$ noise and $t_g = 1$ corresponds to clean data:

$$x^g(t_g) = (1 - t_g)x_0^g + t_g x_1^g, \qquad u^g(t_g) = \frac{dx^g(t_g)}{dt_g} = x_1^g - x_0^g. \tag{2}$$

The velocity $u^g(t_g)$ is a sample-wise local-time velocity: it depends only on the endpoint pair (x_0^g, x_1^g) in group g, and measures motion per unit change in that group's local time t_g. For the linear path above, this velocity is constant in t_g, but we keep the time argument to emphasize which local time it is associated with.

For fixed local times $(t_{\text{tex}}, t_{\text{sem}})$, define the asynchronous composite state $z(t_{\text{tex}}, t_{\text{sem}}) = [x^{\text{tex}}(t_{\text{tex}}), x^{\text{sem}}(t_{\text{sem}})]$, and the corresponding asynchronous noised distribution $p_{t_{\text{tex}},t_{\text{sem}}}(\cdot \mid c) = \text{Law}(z(t_{\text{tex}}, t_{\text{sem}}) \mid c)$. The ideal local flow for group g is the conditional mean of the local velocity:

$$v_g^\star(z, t_{\text{tex}}, t_{\text{sem}}, c) = \mathbb{E}\left[u^g(t_g) \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right], \qquad g \in \mathcal{G}, \tag{3}$$

where $t_g = t_{\text{tex}}$ for $g = \text{tex}$ and $t_g = t_{\text{sem}}$ for $g = \text{sem}$.

Unifying component flows under a single global time. An asynchronous denoising trajectory allows the two groups to occupy different local times. However, the actual sampler still performs **one sequence of model evaluations**, so we need a single global time $\tau \in [0, 1]$ indexing the steps of the denoising algorithm. A schedule specifies how each local time moves as a function of this global time: $t_{\text{tex}}(\tau),\ t_{\text{sem}}(\tau),\ \tau \in [0, 1]$, with endpoint constraints $t_{\text{tex}}(0) = t_{\text{sem}}(0) = 0,\ t_{\text{tex}}(1) = t_{\text{sem}}(1) = 1$. The scheduled asynchronous state and its distribution are

$$z(\tau) = z(t_{\text{tex}}(\tau), t_{\text{sem}}(\tau)) = \left[x^{\text{tex}}(t_{\text{tex}}(\tau)), x^{\text{sem}}(t_{\text{sem}}(\tau))\right], \tag{4}$$

and $p_\tau(\cdot \mid c) = p_{t_{\text{tex}}(\tau),t_{\text{sem}}(\tau)}(\cdot \mid c) = \text{Law}(z(\tau) \mid c)$. Differentiating the scheduled state gives

$$\dot{z}(\tau) = \frac{dz(\tau)}{d\tau} = \left[t'_{\text{tex}}(\tau)u^{\text{tex}}(t_{\text{tex}}(\tau)),\ t'_{\text{sem}}(\tau)u^{\text{sem}}(t_{\text{sem}}(\tau))\right]. \tag{5}$$

The schedule derivatives in Eq. (5) convert local-time velocities into the global-time velocity of the full state used by the sampler.

The next theorem formalizes these relationships.

Theorem 3.1 (Ideal asynchronous flow)**.** *Let $p_1(\cdot \mid c)$ denote the distribution of clean composite latents $x_1 = [x_1^{\text{tex}}, x_1^{\text{sem}}]$ conditioned on class c, and let $p_0 = \mathcal{N}(0, I_{d_{\text{tex}}+d_{\text{sem}}})$ denote the base Gaussian distribution, with $x_0 \sim p_0$ independent of (x_1, c) and $\mathbb{E}\|x_1\|^2 < \infty$. Let $t_{\text{tex}}(\tau)$ and $t_{\text{sem}}(\tau)$ be continuously differentiable, nondecreasing local-time schedules with $t_g(\tau) < 1$ for $\tau < 1$, satisfying $t_{\text{tex}}(0) = t_{\text{sem}}(0) = 0,\ t_{\text{tex}}(1) = t_{\text{sem}}(1) = 1$. Then the following hold:*

(I) Global-time flow. *The global-time vector field*

$$V^\star(z, \tau, c) = \left[t'_{\text{tex}}(\tau)v^\star_{\text{tex}}\big(z, t_{\text{tex}}(\tau), t_{\text{sem}}(\tau), c\big),\ t'_{\text{sem}}(\tau)v^\star_{\text{sem}}\big(z, t_{\text{tex}}(\tau), t_{\text{sem}}(\tau), c\big)\right] \tag{6}$$

drives the scheduled distributional path p_τ. Formally, $V^\star$ and p_τ satisfy the continuity equation $\partial_\tau p_\tau(z \mid c) + \nabla_z \cdot (p_\tau(z \mid c)V^\star(z, \tau, c)) = 0$. Consequently, under the usual regularity conditions for the probability-flow ODE, the solution of $\frac{dZ_\tau}{d\tau} = V^\star(Z_\tau, \tau, c)$ has marginals $p_\tau(\cdot \mid c)$ and transports Gaussian noise at $\tau = 0$ to the data-latent distribution at $\tau = 1$.

(II) Score characterization. *For the Gaussian linear noising path in Eq. (2), the ideal flow of each group $g \in \{\text{tex}, \text{sem}\}$ satisfies*

$$v_g^\star(z, t_{\text{tex}}, t_{\text{sem}}, c) = \frac{1}{t_g} z^g + \frac{1 - t_g}{t_g} \nabla_{z^g} \log p_{t_{\text{tex}}, t_{\text{sem}}}(z \mid c), \qquad 0 < t_{\text{tex}}, t_{\text{sem}} < 1. \tag{7}$$

The explicit form of p_τ and the resulting score form of $V^\star$ are given with the proof in Appendix E.1.

3.2 Flow Matching under a Fixed Asynchronous Schedule

A simplifying assumption. For the ideal flow, only the relative speed of the two local-time schedules matters. Assuming the semantic local-time schedule is strictly increasing, we can reparameterize global time so that semantic time itself is the global time: $t_{\text{sem}}(\tau) = \tau$. We write the texture time as $t_{\text{tex}}(\tau; \rho)$, where ρ parameterizes the schedule. The scheduled asynchronous state and its global-time velocity are

$$z_\rho(\tau) = \left[x^{\text{tex}}(t_{\text{tex}}(\tau; \rho)), x^{\text{sem}}(\tau)\right], \qquad \dot{z}_\rho(\tau) = \left[t'_{\text{tex}}(\tau; \rho) u^{\text{tex}}(t_{\text{tex}}(\tau; \rho)),\ u^{\text{sem}}(\tau)\right]. \tag{8}$$

Since larger local time means less noise, the constraint $t_{\text{tex}}(\tau; \rho) \le \tau\ \forall \tau \in [0, 1]$ makes the semantic component cleaner than the texture component along the trajectory. We refer to this as a semantic-leading schedule.

Component flow-matching loss. The ideal local flows in Eq. (3) are not available in closed form, so we learn them by flow matching. For a texture schedule $t_{\text{tex}}(\tau; \rho)$, the scheduled state $z_\rho(\tau)$ has distribution $p_\tau^\rho(\cdot \mid c) = p_{t_{\text{tex}}(\tau;\rho),\tau}(\cdot \mid c) = \text{Law}(z_\rho(\tau) \mid c)$, the scheduled version of $p_{t_{\text{tex}}, t_{\text{sem}}}$ from Section 3.1.

A θ-parameterized denoising network receives the asynchronous state, the two local times, and the class condition, and predicts group-wise local velocities $\hat{v}_\theta^{\text{tex}}(z, t_{\text{tex}}, t_{\text{sem}}, c)$ and $\hat{v}_\theta^{\text{sem}}(z, t_{\text{tex}}, t_{\text{sem}}, c)$.

For $g \in \{\text{tex}, \text{sem}\}$, define the component-wise norm $\|a^g\|_g^2 := \frac{1}{d_g}\|a^g\|_2^2$. We define the **component flow-matching losses** as

$$\ell_g(\theta; t_{\text{tex}}, t_{\text{sem}}) = \mathbb{E}_{x_0, x_1, c}\left[\left\|\hat{v}_\theta^g\big(z(t_{\text{tex}}, t_{\text{sem}}), t_{\text{tex}}, t_{\text{sem}}, c\big) - u^g(t_g)\right\|_g^2\right], \qquad g \in \{\text{tex}, \text{sem}\}. \tag{9}$$

The targets u^g, as defined in (2), are *velocities with respect to local time t_g*. For any fixed parametric schedule $t_{\text{tex}}(\tau; \rho)$, the **global flow-matching objective** is the sum of the texture and semantic component losses:

$$\mathcal{L}_{\text{flow}}(\theta, \rho; \omega) = \mathbb{E}_{\tau \sim \mathcal{U}(0,1)}\left[\omega_{\text{tex}}(\tau, \rho)\ell_{\text{tex}}\big(\theta; t_{\text{tex}}(\tau; \rho), \tau\big) + \omega_{\text{sem}}(\tau, \rho)\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(\tau; \rho), \tau\big)\right], \tag{10}$$

where $\omega_{\text{tex}}(\tau, \rho) > 0$ and $\omega_{\text{sem}}(\tau, \rho) > 0$ are generic time weights. These weights determine how different portions of the trajectory are emphasized during training. We discuss the proper choice of ω_{tex} and ω_{sem} in Section 3.3; for now, we treat them as arbitrary positive weight functions.

Theorem E.1 in Appendix E.2 states the population target learned by Eq. (10). For fixed ρ, minimizing $\mathcal{L}_{\text{flow}}(\theta, \rho; \omega)$ over θ recovers the ideal local flows defined in (3) along the schedule.

Inference. We freeze the learned schedule ρ, and write $t_{\text{tex}}^\star(\tau) = t_{\text{tex}}(\tau; \rho^\star)$, written $f^\star(\tau)$ outside this section. We sample $z_0 = [z_0^{\text{tex}}, z_0^{\text{sem}}]$ from Gaussian noise and evolve a single global-time trajectory from $\tau = 0$ to $\tau = 1$. Since the denoiser predicts local velocities, the texture branch is converted to global time by the chain rule:

$$\frac{dz_\tau^{\text{tex}}}{d\tau} = t_{\text{tex}}^\star{}'(\tau)\, \hat{v}_\theta^{\text{tex}}\big(z_\tau, t_{\text{tex}}^\star(\tau), \tau, c\big), \qquad \frac{dz_\tau^{\text{sem}}}{d\tau} = \hat{v}_\theta^{\text{sem}}\big(z_\tau, t_{\text{tex}}^\star(\tau), \tau, c\big). \tag{11}$$

Because $t_{\text{tex}}^\star(0) = 0$, $t_{\text{tex}}^\star(1) = 1$, and $t_{\text{sem}}(\tau) = \tau$, the local times of both representation groups run from noise ($t = 0$) to clean data ($t = 1$). The final image is decoded from the texture latent z_1^{tex}; the semantic latent is generated jointly and used only as an auxiliary representation along the denoising trajectory.

3.3 Objective for Learning the Schedule

Section 3.2 defines the weighted flow objective $\mathcal{L}_{\text{flow}}(\theta, \rho; \omega)$ for an arbitrary choice of time weights $\omega = (\omega_{\text{tex}}, \omega_{\text{sem}})$. We now choose these weights so that the same objective is useful for learning the schedule $t_{\text{tex}}(\tau; \rho)$.

Invariant local-time weighting criteria. The schedule should decide which semantic time is paired with each texture time, but it should not change how much training weight each local noise level receives. Since we set $t_{\text{sem}}(\tau) = \tau$, a schedule pairs semantic local time s with texture local time $t_{\text{tex}}(s; \rho)$. Conversely, a texture local time s is paired with semantic time $t_{\text{tex}}^{-1}(s; \rho)$, where $t_{\text{tex}}^{-1}(\cdot; \rho)$ is the inverse of $t_{\text{tex}}(\cdot; \rho)$ in τ.

When learning the ρ-parameterized schedule, we want the weighted flow objective to have the following local-time form *for all* ρ:

$$\begin{aligned}\mathcal{L}_{\text{flow}}(\theta, \rho; \omega) = \mathbb{E}_{s_{\text{tex}} \sim \mathcal{U}(0,1)} \left[\ell_{\text{tex}}\big(\theta; s_{\text{tex}}, t_{\text{tex}}^{-1}(s_{\text{tex}}; \rho)\big) \right] \\ + \mathbb{E}_{s_{\text{sem}} \sim \mathcal{U}(0,1)} \left[\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(s_{\text{sem}}; \rho), s_{\text{sem}}\big) \right].\end{aligned} \tag{12}$$

This condition keeps the marginal weighting over texture local time uniform and the marginal weighting over semantic local time uniform. Changing ρ still changes the cross-representation pairing inside the loss arguments, but it does not change how much total loss weight is assigned to each local time.

The subtlety is that minibatch training samples the global time τ, not the local texture time. Therefore, a fixed global-time weight need not correspond to a fixed local-time weight. For example, if one uses the uncorrected texture weight $\omega_{\text{tex}}(\tau, \rho) = 1$, then the effective distribution on local texture time s_{tex} is no longer uniform, but is instead $1/t'_{\text{tex}}(t_{\text{tex}}^{-1}(s_{\text{tex}}; \rho); \rho)$. Thus the schedule can change the objective in two ways at once: it changes the semantic–texture ordering, and it changes the amount of training weight assigned to different texture noise levels. The second effect is a confound. In particular, the optimizer could reduce the contribution of a difficult texture-time region by making $t_{\text{tex}}(\tau; \rho)$ pass through that region quickly, rather than by finding a better denoising order.

The following lemma states a *necessary and sufficient* ω choice that removes this confound.

Lemma 3.2 (Local-time invariant weighting)**.** *Assume $t_{\text{tex}}(\cdot; \rho)$ is continuously differentiable with $t_{\text{tex}}(0; \rho) = 0$, $t_{\text{tex}}(1; \rho) = 1$, and $t'_{\text{tex}}(\tau; \rho) > 0$ for $\tau \in (0, 1)$. As an identity for arbitrary component losses, the local-time invariance condition in Eq. (12) is satisfied if and only if*

$$\omega_{\text{tex}}(\tau, \rho) = t'_{\text{tex}}(\tau; \rho), \qquad \omega_{\text{sem}}(\tau, \rho) = 1 \tag{13}$$

almost everywhere.

We defer the proof of Lemma 3.2 to Appendix E.3.

Finally, we add fixed group weights w_{tex} and w_{sem} to balance the two representation groups; as constants, they rescale each group's local-time weighting without changing its shape. The corrected weights used by our method are

$$\omega_{\text{tex}}^{\text{corr}}(\tau, \rho) = w_{\text{tex}} \ \text{sg}\left(t'_{\text{tex}}(\tau; \rho)\right), \qquad \omega_{\text{sem}}^{\text{corr}}(\tau, \rho) = w_{\text{sem}}, \tag{14}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. The forward value of $t'_{\text{tex}}(\tau; \rho)$ performs the change-of-variables correction. We stop gradients through this factor because it is an importance-weighting correction, not a schedule objective: the schedule update treats it as a fixed weight and differentiates only through the texture time at which the loss is evaluated.

We plug (14) into (10) and define (with slight abuse of notation) the corrected flow objective as

$$\mathcal{L}_{\text{flow}}(\theta, \rho) := \mathcal{L}_{\text{flow}}\big(\theta, \rho; \omega^{\text{corr}}\big). \tag{15}$$

The explicit global-time and local-time forms of Eq. (15) are given in Appendix E.4. Thus, for every schedule, the corrected objective keeps the marginal local-time weighting fixed, and its value depends on ρ only through the cross-representation ordering.

Kinetic regularization. The corrected flow objective controls the training-time velocity regression problem, but it does not by itself control the geometry of the sampling trajectory. A schedule can

fit local velocities well while still being poor for finite-step ODE sampling; for example, it may compress most texture denoising into a short interval of global time.

Motivated by the global-time flow in Eq. (6), we penalize the squared speed of the predicted global-time velocity along the training interpolation:

$$\mathcal{R}_{\mathrm{kin}}(\theta,\rho)=\mathbb{E}_{\tau\sim\mathcal{U}(0,1),x_0,x_1,c}\Big[t'_{\mathrm{tex}}(\tau;\rho)^2\,\big\|\hat{v}_\theta^{\mathrm{tex}}\big(z_\rho(\tau),t_{\mathrm{tex}}(\tau;\rho),\tau,c\big)\big\|_{\mathrm{tex}}^2 + \big\|\hat{v}_\theta^{\mathrm{sem}}\big(z_\rho(\tau),t_{\mathrm{tex}}(\tau;\rho),\tau,c\big)\big\|_{\mathrm{sem}}^2\Big]. \tag{16}$$

The factor $t'_{\mathrm{tex}}(\tau;\rho)^2$ appears because texture velocities are predicted per unit local texture time but are used per unit global sampling time. Penalizing this quantity discourages schedules that move texture too quickly over a small number of sampling steps.

Schedule and denoiser objectives. The two objectives used in the remainder of the method are

$$\mathcal{L}_\rho(\theta,\rho)=\mathcal{L}_{\mathrm{flow}}(\theta,\rho)+\lambda\mathcal{R}_{\mathrm{kin}}(\theta,\rho), \tag{17}$$

$$\mathcal{L}_\theta(\theta,\rho)=\mathcal{L}_{\mathrm{flow}}(\theta,\rho)+\mathcal{L}_{\mathrm{aux}}(\theta,\rho). \tag{18}$$

The schedule objective $\mathcal{L}_\rho$ is used during the probe stage to select a schedule that preserves local-time weighting and avoids texture motion concentrated in a few sampling steps. The denoiser objective $\mathcal{L}_\theta$ is used after the schedule is fixed; its auxiliary term $\mathcal{L}_{\mathrm{aux}}$ keeps the remaining SFD [18] training recipe unchanged, including the LightningDiT auxiliary losses and the REPA alignment loss [27]. Section 3.5 describes the two-stage optimization procedure.

3.4 Semantic-Leading Schedule Parameterization

After setting semantic time as the global time, $t_{\mathrm{sem}}(\tau)=\tau$, the only schedule to learn is the texture schedule $t_{\mathrm{tex}}(\tau;\rho)$.

Convex monotone schedule family. We parameterize the derivative of the texture schedule as a normalized polynomial with positive coefficients:

$$t'_{\mathrm{tex}}(\tau;\rho)=\frac{1}{Z_\rho}\sum_{m=0}^{M}a_m\tau^m,\qquad a_m=\mathrm{softplus}(\rho_m)>0,\qquad Z_\rho=\sum_{m=0}^{M}\frac{a_m}{m+1}. \tag{19}$$

The texture schedule is obtained by closed-form integration:

$$t_{\mathrm{tex}}(\tau;\rho)=\int_0^\tau t'_{\mathrm{tex}}(s;\rho)\,ds=\frac{1}{Z_\rho}\sum_{m=0}^{M}\frac{a_m}{m+1}\tau^{m+1}. \tag{20}$$

The normalization Z_ρ enforces $t_{\mathrm{tex}}(1;\rho)=1$, while the integral form gives $t_{\mathrm{tex}}(0;\rho)=0$. Since the coefficients are positive, $t'_{\mathrm{tex}}(\tau;\rho)>0$ on $[0,1]$, so $t_{\mathrm{tex}}(\cdot;\rho)$ is strictly increasing and has a well-defined inverse. This inverse is used in the local-time change-of-variables analysis in Section 3.3; the implementation itself samples τ and evaluates $t_{\mathrm{tex}}(\tau;\rho)$ and $t'_{\mathrm{tex}}(\tau;\rho)$ directly.

Semantic-leading property. The same parameterization enforces semantic-leading schedules by convexity: $t_{\mathrm{tex}}(\tau;\rho)\le\tau$ for all $\tau\in[0,1]$ (Appendix C.4). Because larger local time means less noise, this inequality ensures that the semantic component is always at least as clean as the texture component along the trajectory. Thus the semantic-leading constraint is enforced by construction, without an auxiliary penalty.

We use $M=4$ in all experiments. The schedule parameters are initialized so that $t_{\mathrm{tex}}(\tau;\rho)$ is close to the identity schedule $t_{\mathrm{tex}}(\tau;\rho)\approx\tau$, and the schedule probe in Section 3.5 then learns the semantic-leading deviation from this initialization.

3.5 Bilevel Schedule Probe

The schedule objective in Eq. (17) defines what makes a schedule useful, but it cannot be optimized over ρ *in isolation*. The quality of a schedule depends on the denoiser obtained after adapting to that schedule, so schedule learning is naturally a bilevel problem (Eq. (23) in Appendix C.5).

This bilevel view has an important practical consequence: schedule gradients are only meaningful when the denoiser tracks the schedule. If θ is frozen, changing ρ changes the asynchronous state $z_\rho(\tau)$ and the time inputs $(t_{\text{tex}}(\tau;\rho),\tau)$ on which the denoiser is evaluated. The resulting flow-matching error reflects a model–schedule mismatch, rather than the intrinsic quality of the new schedule. In practice, this mismatch makes the loss rise sharply under schedule perturbations, so ρ barely moves. We therefore learn ρ using a temporary denoiser that is optimized jointly with the schedule.

We do not solve Eq. (23) exactly. Full denoiser training is expensive, but the schedule parameters are low-dimensional and converge quickly. A short joint optimization of the denoiser and the schedule is sufficient to identify a good schedule. We therefore use a two-stage recipe. In Stage I, we run a short joint probe over a temporary denoiser θ_{probe} and the schedule parameters ρ, using only about 1% of the main training budget. In Stage II, we freeze the learned schedule, discard θ_{probe}, and train the final denoiser θ from scratch. No schedule parameters are updated during the main training run.

During the probe, we optimize $\mathcal{L}_{\text{probe}}(\theta_{\text{probe}},\rho) = \mathcal{L}_\rho(\theta_{\text{probe}},\rho)$. Equivalently, using Eq. (17), this is $\mathcal{L}_{\text{flow}}(\theta_{\text{probe}},\rho) + \lambda\mathcal{R}_{\text{kin}}(\theta_{\text{probe}},\rho)$. The probe denoiser is not intended to be a final generative model; it only tracks the changing flow-matching problem well enough to supply useful gradients for the schedule. We choose λ from a one-dimensional sweep as the weakest value that prevents collapse toward the extremal schedule. This transition is visualized in Section 4.2.

After the probe, we average the schedule parameters over the stable post-burn-in window and freeze the resulting schedule $t^\star_{\text{tex}}(\tau)$. The final denoiser is then trained from scratch with this fixed schedule using $\mathcal{L}_\theta$, including the unchanged SFD auxiliary losses.

Algorithm 1 in Appendix C.6 summarizes the procedure.

4 Experiments

4.1 Experimental Setup

Task and protocol. We evaluate on class-conditional ImageNet-256×256 generation. Our goal is to isolate the effect of schedule discovery. Therefore, all controlled comparisons use the SFD framework [18]. We keep the architecture, latent representation, auxiliary losses, weak-model architecture for guidance, sampler, and evaluation protocol unchanged; our changes are confined to the learned fixed schedule $f^\star$ and the schedule-learning procedure used to obtain it.

Baselines. Our most direct baseline is SFD [18], which shares with us the asynchronous semantic–texture latent representation, the LightningDiT-XL/1 backbone, the weak-model architecture and training budget for AutoGuidance, and the sampler; we report the released SFD-XL and SFD-XXL checkpoints. For broader context in Table 2, we also include representative class-conditional ImageNet 256×256 generators (listed in Appendix B). These broader comparisons are not fully controlled, but contextualize the system-level performance of our learned schedule.

Metrics. Following standard ImageNet generation practice, we generate 50K class-balanced samples and report FID [8], sFID [16], Inception Score (IS) [22], Precision, and Recall [11]. Unless otherwise stated, samples are evaluated against the ADM reference statistics [4].

Implementation. We train on the SFD-released ImageNet latent dataset with a LightningDiT-XL/1 backbone (675M parameters) and batch size 256. The schedule probe (Section 3.5) is run for $S_{\text{probe}} = 10\text{K}$ steps with $S_{\text{burn}} = 5\text{K}$ burn-in, approximately 1% of our 1M-iteration main training runs; schedule discovery therefore adds negligible overhead on top of main training. We report results across training budgets up to 3M iterations. We index unguided convergence (Table 1) by iteration count, following the LightningDiT and SFD baselines, and the system-level comparison (Table 2) by epoch, following the methods we compare against; at batch size 256 on ImageNet-1k one epoch is roughly 5K iterations, so 80, 200, and 600 epochs correspond to about 0.4M, 1M, and 3M iterations. Full training and hardware details are given in Appendix B. Sampling integrates the ODE with the adaptive dopri5 solver on a grid of 250 time points, and AutoGuidance [9] uses a weak model with the same configuration as SFD-XL's (LightningDiT-B trained for 70K steps), but trained under our learned schedule, so that the weak and main models share the same asynchronous denoising path. Full hyperparameter settings are listed in Tables 4 and 5 of Appendix B.

Table 1: **Unguided FID convergence.** All entries use 675M-parameter XL backbones. Baseline numbers are from Peebles and Xie [19], Yao et al. [26], Yu et al. [27], Pan et al. [18].

Model	Iter.	FID↓
DiT-XL/2	400K	19.47
DiT-XL/2	7M	9.62
LightningDiT	400K	9.29
LightningDiT	1M	7.48
LightningDiT	2M	6.88
LightningDiT	4M	6.50
+ REPA	400K	6.94
+ REPA	1M	6.17
+ REPA	2M	5.87
+ REPA	4M	5.84
+ SFD	70K	8.79
+ SFD	120K	6.22
+ SFD	400K	3.53
+ SFD	1M	2.82
+ SFD	2M	2.74
+ SFD	4M	2.54
+ Ours	70K	**6.89**
+ Ours	120K	**4.93**
+ Ours	400K	**2.87**
+ Ours	800K	**2.53**
+ Ours	1M	**2.37**
+ Ours	2M	**2.21**
+ Ours	3M	**2.14**

Table 2: **System-level comparison with AutoGuidance.** Class-conditional ImageNet 256×256 results under AutoGuidance; baseline numbers are from the corresponding papers.

Method	Ep.	Par.	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑
Latent Diffusion Models							
DiT-XL	1400	675M	2.27	4.60	278.2	**0.83**	0.57
MaskDiT	1600	675M	2.28	5.67	276.6	0.80	0.61
SiT-XL	1400	675M	2.06	4.50	270.3	0.82	0.59
FasterDiT	400	675M	2.03	4.63	264.0	0.81	0.60
MDT	1300	675M	1.79	4.57	283.0	0.81	0.61
MDTv2	1080	675M	1.58	4.52	**314.7**	0.79	0.65
DDT	400	675M	1.26	–	310.6	0.79	0.65
Leveraging Visual Representations							
VA-VAE	800	675M	1.35	4.15	295.3	0.79	0.65
REPA	800	675M	1.42	4.70	305.7	0.80	0.65
REPA-E	800	675M	1.12	4.09	302.9	0.79	0.66
ReDi	800	675M	1.61	4.66	295.1	0.78	0.64
REG	800	677M	1.36	4.25	299.4	0.77	0.66
RAE-DiT	800	676M	1.41	–	309.4	0.80	0.63
RAE-DiTDH	800	839M	1.13	–	262.6	0.78	0.67
SFD-XL	80	675M	1.30	3.87	233.4	0.78	0.64
SFD-XL	800	675M	1.06	3.89	267.0	0.78	0.67
SFD-XXL	80	1.0B	1.19	4.00	240.4	0.78	0.65
SFD-XXL	800	1.0B	1.04	**3.75**	264.2	0.78	0.66
Ours-XL	80	675M	1.14	3.79	248.4	0.78	0.71
Ours-XL	200	675M	1.05	3.79	273.0	0.78	**0.72**
Ours-XL	600	675M	**1.02**	3.78	270.8	0.78	0.66

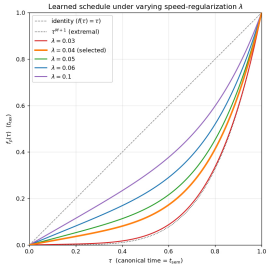


Figure 2: λ-sweep of the schedule probe.

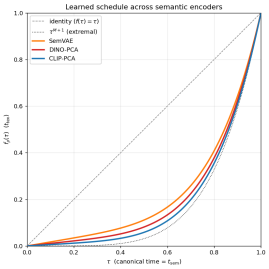


Figure 3: Learned schedules across semantic representations.

Table 3: **Robustness to semantic representations.** Unguided FID at 400K iterations; the SemVAE entry matches the 400K row of Table 1. SFD numbers are from Pan et al. [18]. †SFD trains a CLIP-VAE on top of CLIP features; ours uses CLIP-PCA without that intermediate VAE.

Feature source	Semantic latent	SFD [18]	Ours
DINOv2-B [17]	SemVAE	3.53	**2.87**
DINOv2-B [17]	PCA	4.06	**2.97**
CLIP [20]	VAE / PCA†	4.89	**4.54**

4.2 Schedule Probe Diagnostics

We sweep the kinetic-energy regularization strength $\lambda \in \{1, 2, 3, 4, 5, 6, 10\} \times 10^{-2}$ during the probe and observe two regimes (Figure 2). For $\lambda \leq 3 \times 10^{-2}$ the regularizer is too weak to prevent collapse, and the schedule converges to the same extremal semantic-leading curve regardless of the exact value, so we plot a single representative curve for this collapsed regime. For $\lambda \geq 4 \times 10^{-2}$ the schedule stabilizes into a smooth curve that becomes progressively closer to the identity as λ grows. We use the weakest stable value, $\lambda = 4 \times 10^{-2}$, for all main experiments.

4.3 Main Results

We evaluate the learned schedule in two regimes. First, we compare unguided FID across training budgets to measure convergence speed. Second, we evaluate the final system with AutoGuidance and compare against state-of-the-art class-conditional ImageNet generators. In both regimes, the most direct comparison is SFD-XL, since Ours and SFD-XL use the same backbone, latent representation, weak-model architecture, sampler, and evaluation protocol. They differ in the fixed schedule used for asynchronous denoising and in the associated corrected flow loss from Eq. (15).

Faster unguided convergence. Table 1 shows that the learned schedule improves sample-efficiency over the fixed SFD schedule at every matched training budget. The effect is largest in the low- and mid-compute regimes: at 400K iterations, it improves FID from 3.53 to 2.87, and it reaches FID 2.53 at 800K iterations, matching the 4M-iteration SFD-XL checkpoint with roughly $5\times$ fewer diffusion-model updates.

Better final unguided quality. The gain is not only an early-training effect. At 1M iterations it reaches FID 2.37, improving over the best reported SFD-XL unguided result of 2.54 at 4M iterations; extending training to 3M lowers the unguided FID further to 2.14. Thus the learned schedule both accelerates convergence and improves the final unguided model under the same architecture and latent representation.

Both gains transfer under AutoGuidance. Table 2 shows the same pattern under AutoGuidance [9]. With the same weak-model architecture and guidance scale as SFD-XL, our model cuts SFD-XL's FID from 1.30 to **1.14** at 80 epochs and reaches **FID 1.047** (1.05 in Table 2) at 200 epochs, on par with the 1.0B-parameter SFD-XXL (1.04) despite a smaller backbone and one quarter of the training budget. Training longer to 600 epochs reaches **FID 1.02**, the lowest among all 675M-parameter entries in Table 2 and below the 1.0B-parameter SFD-XXL. On Recall, our 200-epoch model reaches **0.72**, the highest among all entries in Table 2, indicating that the gains include broader coverage of the data distribution rather than sharper modes alone. Figure 5 in Appendix F shows class-conditional samples from the same guided sampling setup.

4.4 Robustness to Semantic Representations

Alternative semantic representations. The main experiments use the SFD SemVAE semantic latent, itself trained as a VAE on top of DINOv2-B [17] features. To test whether the schedule-learning procedure depends on this encoder, we evaluate two simpler alternatives that skip the SemVAE training step: *DINO-PCA*, a direct PCA projection of the same DINOv2-B feature dataset that SemVAE was trained on, and *CLIP-PCA*, a PCA projection of CLIP [20] features on ImageNet, both to the same channel count as SemVAE. The only changes from the main setup are the semantic latent and the per-encoder schedule. The kinetic-regularizer strength is set per semantic representation: SemVAE uses the main value $\lambda = 4 \times 10^{-2}$ (Section 4.2), and DINO-PCA and CLIP-PCA share $\lambda = 2 \times 10^{-2}$.

Schedule shape varies with the semantic encoder. The frozen schedules differ across encoders (Figure 3): CLIP-PCA produces the most aggressive semantic-leading curve, DINO-PCA sits in the middle, and SemVAE stays closest to the identity. Roughly, the richer the semantic latent's information about the image, the less the texture branch is delayed.

Results. At 400K iterations (Table 3), the learned schedule reaches FID 2.97 with DINO-PCA and 4.54 with CLIP-PCA. Under the same two encoders, Pan et al. [18] report 4.06 with DINO-PCA and 4.89 with a CLIP-VAE trained on CLIP features. Schedule learning thus improves over the hand-tuned offset at every encoder, and in the CLIP case our PCA projection already beats SFD's CLIP-VAE recipe while skipping the VAE-training step entirely.

5 Conclusion

We replace the hand-tuned semantic-leading offset of SFD with a schedule $f^\star$ learned as a convex monotone bijection of the global time, so the leading property holds without an auxiliary constraint. A short regularized probe discovers $f^\star$ jointly with a temporary denoiser, while a kinetic-energy regularizer keeps the resulting curve away from the extremal collapse that fits the training loss but is too stiff to integrate in finite steps. The averaged curve is frozen for full training, leaving the final inference interface identical to a fixed-schedule asynchronous model. On ImageNet-256×256, this gives faster unguided convergence and better final FID than the hand-tuned schedule under the same backbone, and under AutoGuidance reaches FID 1.02, the lowest among the 675M-parameter systems in our comparison and below the 1.0B-parameter SFD-XXL. The gain also transfers across DINO- and CLIP-based semantic encoders. We discuss limitations and directions for future work in Appendix A.

Acknowledgments and Disclosure of Funding

Part of the computation was performed on the Duke Compute Cluster. This work received no third-party funding. The authors declare no competing interests.

References

[1] Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Fei-Fei Li. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation, 2026.

[2] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, 2024.

[3] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

[4] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

[5] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[6] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[7] Yuan Gao, Chen Chen, Tianrong Chen, and Jiatao Gu. One layer is enough: Adapting pretrained visual encoders for image generation, 2025.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[9] Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself, 2024.

[10] Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. In *Advances in Neural Information Processing Systems*, 2025.

[11] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, 2019.

[12] Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning with latent diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[13] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[14] Pengxi Liu, Zeyu Michael Li, and Xiang Cheng. Variational trajectory optimization of anisotropic diffusion schedules, 2026.

[15] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, 2024.

[16] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7958–7968, 2021.

[17] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[18] Yueming Pan, Ruoyu Feng, Qi Dai, Yuqi Wang, Wenfeng Lin, Mingyu Guo, Chong Luo, and Nanning Zheng. Semantics lead the way: Harmonizing semantic and texture modeling with asynchronous latent diffusion, 2025.

[19] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763, 2021.

[21] 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*, 2022.

[22] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, 2016.

[23] Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: Decoupled diffusion transformer, 2025.

[24] Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *Advances in Neural Information Processing Systems*, 2025.

[25] Jingfeng Yao, Wang Cheng, Wenyu Liu, and Xinggang Wang. FasterDiT: Towards faster diffusion transformers training without architecture modification. In *Advances in Neural Information Processing Systems*, 2024.

[26] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[27] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

[28] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders, 2025.

[29] Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research*, 2024.

A Limitations and Future Work

Our experiments cover a single visual modality at a single resolution and backbone size: class-conditional ImageNet-256 $\times$ 256 with the LightningDiT-XL/1 backbone. While the gains transfer across the three semantic encoders we evaluate (Section 4.4), we do not study text-to-image, video, or audio diffusion, and we do not scale beyond 675M parameters. The interaction between $f^\star$ and other guidance methods is also limited in scope: we use AutoGuidance with the same weak-model configuration as SFD-XL, and we have not separately characterized how the learned schedule interacts with classifier-free guidance or with guidance weights that themselves vary along τ.

We parameterize $t'_{\text{tex}}(\tau;\rho)$ as a degree-$M = 4$ polynomial. This is the minimal expressive degree that gave a non-collapsed curve under the kinetic regularizer in preliminary tests, and we have not compared richer monotone families such as splines or input-convex networks. The probe also requires a per-encoder sweep of λ, since the value that prevents collapse shifts with the semantic encoder, and transferring the procedure to a new encoder involves a small one-dimensional search.

A natural next direction is to extend the framework beyond two representation groups, learning a joint schedule among a texture latent, a semantic latent, and additional auxiliary representations such as depth or segmentation. The convex-monotone parameterization extends directly to a collection of pairwise schedules, but the kinetic regularizer and probe budget would need to be revisited at this larger scale.

B Additional Experimental Details

Baselines. The representative class-conditional ImageNet 256 $\times$ 256 generators in Table 2 fall into two groups. *Latent diffusion transformers*: DiT [19], SiT [15], MaskDiT [29], FasterDiT [25], MDT [5], MDTv2 [6], and DDT [23]. *Methods leveraging visual representations*: VA-VAE [26], REPA [27], REPA-E [12], ReDi [10], REG [24], and RAE [28].

Metrics. FID measures distributional visual quality, sFID emphasizes spatial statistics, IS measures class-conditional sample quality and diversity, while Precision and Recall separate fidelity from coverage.

Setup details. We use the SFD [18] implementation and its released ImageNet-256 $\times$ 256 latent dataset. Texture latents come from the SD-VAE f16-d32 of LightningDiT [26]; semantic latents come from the SemVAE encoder, which compresses DINOv2-B [17] patch features with registers [3] into a compact semantic latent. The diffusion backbone is LightningDiT-XL/1, with REPA [27] alignment between an internal DiT block and DINOv2-B final-layer features. The implementation averages the squared errors over all 48 channels and multiplies the 16 semantic channels by 2; since the texture group has 32 channels, this equals $w_{\text{tex}} = w_{\text{sem}} = 2/3$ on the per-group normalized losses of Eq. (9), i.e., equal weights for the two groups. FID is computed against the ADM [4] reference statistics. All remaining numerical settings are listed in Tables 4 (schedule learning) and 5 (inherited).

Hardware and runtime. Training budgets up to 1M iterations were run on NVIDIA B200 and longer budgets (2M and 3M iterations) on NVIDIA H200, under an identical recipe, sampler, and evaluation protocol. On B200 the main training run uses 2 GPUs at a throughput of 4.6 steps per second, giving approximately 60 hours of wall-clock time for the 1M-iteration budget; the schedule probe runs for $S_{\text{probe}} = 10\text{K}$ steps and takes approximately 36 minutes, consistent with the under-1% overhead reported in Section 4.1.

Guidance scale. For AutoGuidance we use guidance scale $w = 1.5$ (matching SFD-XL) for the 80- and 200-epoch models, and $w = 1.3$ for the more-converged 600-epoch model.

C Additional Method Details

Overview of Section 3. Our method learns an asynchronous denoising schedule for a composite latent containing texture and semantic representations. Section 3.1 first defines the asynchronous flow induced by separate local-time schedules for the two representation groups, and characterizes

Table 4: Hyperparameters introduced or chosen in this work for schedule learning. All other settings are inherited from SFD [18]/LightningDiT [26] and listed in Table 5.

Setting	Value
Schedule parameterization	
Polynomial degree M	4
ρ initialization	near identity ($t_{\text{tex}}(\tau;\rho) \approx \tau$)
Group weight w_{tex}	$2/3$
Group weight w_{sem}	$2/3$
Stop-grad on $t'_{\text{tex}}(\tau;\rho)$ factor	yes
Schedule probe	
Probe steps S_{probe}	10K
Burn-in steps S_{burn}	5K
Kinetic regularizer λ	4×10^{-2}
Probe optimizer	AdamW
Probe learning rate	10^{-4} (denoiser), 10^{-2} (ρ)
Probe batch size	256

the corresponding ideal local and global velocity fields. Section 3.2 then shows how these local velocities are learned by flow matching for a fixed schedule, and how the learned local velocities are converted into the global-time ODE used at inference. Section 3.3 defines the objective used to learn the schedule: a Jacobian-corrected flow loss keeps the marginal weighting over local times invariant, while a kinetic regularizer discourages schedules that are difficult to discretize. Finally, Section 3.4 states our explicit parameterization of the learnable schedule: a construction which restricts the learnable texture schedule $t_{\text{tex}}(\tau;\rho)$ to a convex monotone family and enforces semantic-leading behavior. Section 3.5 describes the efficient two-stage optimization procedure: a short joint probe learns the schedule, after which the schedule is frozen and the final denoiser is trained from scratch.

C.1 Standard flow matching

Background: standard flow matching. We first recall the standard flow-matching construction. Let y_0 denote a sample from a base distribution, let y_1 denote a data sample given condition c, drawn independently of y_0, and define the linear path

$$y(t) = (1-t)y_0 + ty_1, \qquad u = \frac{dy(t)}{dt} = y_1 - y_0, \qquad t \in [0,1]. \tag{21}$$

In our convention, $t = 0$ corresponds to $\mathcal{N}(0, I)$ noise and $t = 1$ corresponds to clean data. Let $p_t(\cdot \mid c) := \text{Law}(y(t) \mid c)$ denote the interpolating distribution. The ideal flow field along this path is the conditional mean velocity

$$v^\star(y, t, c) = \mathbb{E}\left[u \mid y(t) = y,\ c\right]. \tag{22}$$

This field drives the curve of distributions $\{p_t\}_{t\in[0,1]}$: under the usual regularity conditions, the ODE $dY_t/dt = v^\star(Y_t, t, c)$ transports the base distribution p_0 to the data distribution p_1.

The component interpolant in Eq. (2) has exactly the same form as Eq. (21), with $y(t) = x^g(t_g)$, $t = t_g$, and $u = u^g$.

Eq. (5) is the main difference from the standard path in Eq. (21). The standard interpolant $y(t)$ has a single time variable, so every coordinate is evaluated at the same noise level. The asynchronous state $z(\tau)$ has a single global sampler time τ, but its texture and semantic components may be evaluated at different local times $t_{\text{tex}}(\tau)$ and $t_{\text{sem}}(\tau)$; importantly, the noise levels at $z(\tau)$ differ in the tex and sem components.

C.2 Velocity objects

Our method centers around three subtly different velocity objects:

Table 5: Hyperparameters inherited from SFD [18] and LightningDiT [26]. Schedule-learning hyperparameters introduced by this work are listed separately in Table 4.

Setting	**Value**
Architecture	
Backbone	LightningDiT-XL/1
Parameters	675M
DiT blocks	28
Hidden dim	1152
Attention heads	16
MLP ratio	4.0
Patch size	1
Texture latent shape	$32 \times 16 \times 16$
Semantic latent shape	$16 \times 16 \times 16$
SemVAE parameters	29M
Main training	
Batch size	256
Optimizer	AdamW
Learning rate	10^{-4}
β_1	0.9
β_2	0.999
Weight decay	0
LR warmup	none
LR schedule	constant
Gradient clipping	none
EMA decay	0.9999
Precision	bf16 mixed
Auxiliary losses	
REPA alignment from / to	DiT block 2 / DINOv2 final layer
REPA loss weight	1.0
Cosine-direction loss weight	1.0
Sampling	
ODE solver	dopri5
Output time points	250
atol	10^{-6}
rtol	10^{-3}
AutoGuidance scale w	1.5 (80/200 ep.), 1.3 (600 ep.)
Weak model	LightningDiT-B
Weak model training steps	70K
Evaluation	
Sample count	50K, class-balanced
Reference batch	`VIRTUAL_imagenet256_labeled.npz`
Metrics	FID, sFID, IS, Precision, Recall

1. The endpoint velocity $u^g(t_g)$ in (2) is a sample-wise velocity **determined by its own group's endpoint pair** (x_0^g, x_1^g).
2. The ideal local flow $v_g^\star$ in (3) is a population velocity field obtained by **conditioning on the full composite state** z, so $v_{\text{tex}}^\star$ may depend on both the texture and semantic components, and likewise for $v_{\text{sem}}^\star$.
3. The global flow $V^\star$ in (6), is the global-time vector field obtained by concatenating the local flows and multiplying by the corresponding local-time schedule derivatives.

Theorem 3.1 formalizes these relationships.

C.3 Requirements for the schedule objective and parameterization

A good schedule objective should satisfy two requirements. First, it should optimize the semantic–texture denoising order **without changing the marginal weighting** over local noising times. Second, it should prefer schedules whose global-time trajectories are **stable under finite-step ODE sampling**. We address these requirements with a change-of-variables reweighting and a kinetic regularizer (Section 3.3).

For the schedule itself (Section 3.4), we use a compact parameterization that enforces the structural constraints required by Sections 3.2–3.3. The texture schedule should satisfy the endpoint constraints $t_{\text{tex}}(0;\rho) = 0$ and $t_{\text{tex}}(1;\rho) = 1$, be monotone so that local texture time never runs backward, be semantic-leading so that $t_{\text{tex}}(\tau;\rho) \le \tau$, and provide an easily computed derivative $t'_{\text{tex}}(\tau;\rho)$. The derivative is needed for the inference ODE, the Jacobian-corrected flow loss, and the kinetic regularizer.

C.4 Semantic-leading property

Differentiating Eq. (19) gives $t''_{\text{tex}}(\tau;\rho) = Z_\rho^{-1} \sum_{m=1}^{M} m a_m \tau^{m-1} \ge 0,\ \tau \in [0,1]$, so $t_{\text{tex}}(\cdot;\rho)$ is convex. Since $t_{\text{tex}}(0;\rho) = 0$ and $t_{\text{tex}}(1;\rho) = 1$, convexity implies that the schedule lies below the chord connecting its endpoints: $t_{\text{tex}}(\tau;\rho) \le \tau,\ \forall \tau \in [0,1]$.

C.5 Bilevel formulation of schedule learning

Schedule learning is naturally a bilevel problem:

$$\rho^\star \in \arg\min_\rho \mathcal{L}_\rho\big(\theta^\star(\rho), \rho\big), \qquad \theta^\star(\rho) \in \arg\min_\theta \mathcal{L}_\theta(\theta, \rho). \tag{23}$$

The inner problem adapts the denoiser to a fixed schedule. The outer problem then chooses the schedule using the adapted denoiser.

C.6 Two-stage schedule probe

Algorithm 1 summarizes the two-stage procedure of Section 3.5.

D Additional Experiments and Analyses

Unless stated otherwise, the experiments in this section train the final model from scratch at the 70K and 120K budgets of Table 1 and report unguided FID-50K. These budgets count only the updates of the final denoiser; the separate 10K-step probe is not included, and its denoiser is discarded rather than used to warm-start the final model.

D.1 Schedule learning and the corrected objective

Ours differs from SFD both in the learned schedule and in the corrected texture weight of Eq. (14). To separate the two, we apply the equivalent correction to SFD's fixed offset and train the learned $f^\star$ without it. Eq. (14) cannot be copied literally to SFD, because SFD's path contains clamped intervals at both ends. We therefore write the equivalent correction in terms of SFD's global clock:

Algorithm 1 Two-stage schedule probe and fixed-schedule training

Require: Main training budget S_{train}, probe fraction $\eta_{\text{probe}} \approx 0.01$, burn-in steps S_{burn}

Stage I: joint schedule probe

Train: temporary denoiser θ_{probe}, schedule parameters ρ

Objective:

$$\mathcal{L}_{\text{probe}}\big(\theta_{\text{probe}}, \rho\big) = \mathcal{L}_{\rho}\big(\theta_{\text{probe}}, \rho\big)$$

Set

$$S_{\text{probe}} \leftarrow \lceil \eta_{\text{probe}} S_{\text{train}} \rceil$$

Initialize θ_{probe}

Initialize ρ with $t_{\text{tex}}(\tau; \rho) \approx \tau$

Initialize schedule buffer $\mathcal{B} \leftarrow \emptyset$

for $s = 1, \ldots, S_{\text{probe}}$ **do**

  Sample (x_1, c), Gaussian noise x_0, and $\tau \sim \mathcal{U}(0, 1)$

  Update both θ_{probe} and ρ using

$$\nabla_{\theta_{\text{probe}}, \rho} \mathcal{L}_{\text{probe}}\big(\theta_{\text{probe}}, \rho\big)$$

  if $s > S_{\text{burn}}$ **then**

    Append ρ to $\mathcal{B}$

  end if

end for

Average stable schedule parameters:

$$\bar{\rho} \leftarrow |\mathcal{B}|^{-1} \sum_{\rho_s \in \mathcal{B}} \rho_s$$

Define and freeze the learned texture schedule

$$t^{\star}_{\text{tex}}(\tau) \leftarrow t_{\text{tex}}(\tau; \bar{\rho})$$

Discard θ_{probe}

Stage II: fixed-schedule final training

Train: final denoiser θ

Freeze: schedule parameters $\bar{\rho}$

Objective:

$$\mathcal{L}_{\theta}(\theta, \bar{\rho})$$

Initialize final denoiser θ from scratch

for $s = 1, \ldots, S_{\text{train}}$ **do**

  Sample (x_1, c), Gaussian noise x_0, and $\tau \sim \mathcal{U}(0, 1)$

  Update only θ using

$$\nabla_{\theta} \mathcal{L}_{\theta}(\theta, \bar{\rho})$$

end for

Table 6: **Ablations.** Unguided FID-50K of the final model at 70K and 120K iterations.

Schedule and training objective	70K	120K
Schedule learning and the corrected objective		
SFD fixed offset	8.79	6.22
SFD fixed offset + uniform local-time weighting	8.20	5.51
Learned $f^{\star}$ without Eq. (14)	8.87	6.01
Learned $f^{\star}$ (Ours)	**6.89**	4.93
Learned schedule family		
Non-convex (Bernstein, degree 9)	9.74	6.50
Convex, $M = 4$ (Ours)	**6.89**	4.93
Convex, $M = 5$	7.33	**4.80**
Convex, $M = 6$	7.66	5.13

each branch is weighted only while its own local time is moving, so that its active local-time range remains uniformly weighted. With normalized global clock $u \in [0,1]$ and offset $\delta = 0.3$,

$$\begin{aligned} t_{\text{sem}}(u) &= \min((1+\delta)u, 1), \qquad t_{\text{tex}}(u) = \max((1+\delta)u - \delta, 0), \\ J_{\text{sem}}(u) &= (1+\delta)\mathbf{1}[u < \tfrac{1}{1+\delta}], \quad J_{\text{tex}}(u) = (1+\delta)\mathbf{1}[u > \tfrac{\delta}{1+\delta}], \end{aligned} \tag{24}$$

and J_{sem}, J_{tex} multiply the semantic and texture losses. While a branch is active, its loss is multiplied by $1 + \delta = 1.3$; while it is clamped, its duplicated endpoint loss receives zero weight. The correction is applied to both branches because SFD freezes texture at the beginning and semantics at the end.

The correction improves fixed SFD from 6.22 to 5.51 at 120K, a general benefit of the reweighting. The learned $f^\star$ without the correction also improves over SFD (6.01 vs. 6.22), but this comparison changes the semantic–texture pairing and the implicitly induced texture-time training density at once. Holding the density treatment fixed, schedule learning improves FID from 5.51 to 4.93. The correction is especially well matched to the curved learned schedule: because $f^{\star\prime}(\tau)$ varies over time, uniform-τ sampling would otherwise change the texture-time density and under-train the late texture region, giving 8.87 instead of 6.89 at 70K. Schedule learning and the corrected objective are therefore complementary: learning changes which semantic state is paired with each texture state, and the correction ensures that this change in ordering does not also change texture-time coverage.

D.2 Schedule family and polynomial degree

The non-convex comparison replaces the convex degree-4 map with a degree-9 Bernstein parameterization,

$$\begin{aligned} f'(\tau) &= \frac{1}{Z}\sum_{m=0}^{9} a_m B_{m,9}(\tau), \quad B_{m,9}(\tau) = \binom{9}{m}\tau^m(1-\tau)^{9-m}, \\ a_m &= \text{softplus}(r_m) > 0, \qquad\qquad Z = \frac{1}{10}\sum_{m=0}^{9} a_m, \end{aligned} \tag{25}$$

with $f(\tau) = \int_0^\tau f'(s)\,ds$ in closed form. This guarantees $f(0) = 0$, $f(1) = 1$, and $f' > 0$, while the free Bernstein profile allows f' to increase or decrease, so f need not be convex. The semantic-leading condition $f(\tau) \le \tau$ is enforced separately during the probe. The probe, kinetic regularization, schedule learning rate, stable-window averaging, final training, and evaluation protocol are unchanged. The learned schedule is genuinely non-convex: it stays nearly synchronous early, delays texture mainly in the middle, and catches up near the endpoint (Figure 4). It reaches 9.74 and 6.50, against 6.89 and 4.93 for the convex schedule, which supports convexity as a useful SFD-informed inductive bias in our setting.

Repeating the complete procedure with $M = 5$ and $M = 6$, each with a new probe and a final model trained from scratch, gives 7.33/4.80 and 7.66/5.13 at 70K/120K. The early 70K results vary with M, while at 120K all three degrees lie between 4.80 and 5.13, and every degree improves over SFD at both budgets. We use $M = 4$, the smallest degree that produced a stable, non-collapsed schedule.

D.3 Interpretation of the learned schedule

Writing the semantic lead as $g(\tau) = \tau - f(\tau)$, convexity of f with fixed endpoints makes g nonnegative and concave, with $g(0) = g(1) = 0$: the lead is built and then removed once. This is the coarse geometry of SFD's offset, whose semantic lead is introduced, maintained, and removed once rather than repeatedly expanding and contracting; convexity is a compact proxy for these semantic-first dynamics. What the probe learns is therefore not whether semantics should lead, but how the lead is distributed over time. The final SemVAE schedule builds a lead of about 0.44 around two-thirds of the trajectory ($\tau \approx 0.67$) and then closes it smoothly, rather than holding SFD's fixed 0.3 plateau between piecewise-linear transitions.

The same smoothness offers a reading of the early-training advantage in Table 1. At the matched 70K and 120K budgets, Ours and SFD train the same architecture for the same number of updates under two frozen schedules. SFD's clamped offset is piecewise linear, with kinks at the clamp boundaries and intervals in which one representation's local time is frozen, whereas $f^\star$ is continuously differentiable and yields a smoothly varying semantic–texture coupling; the early advantage is consistent with a conditional vector field that is easier to fit.

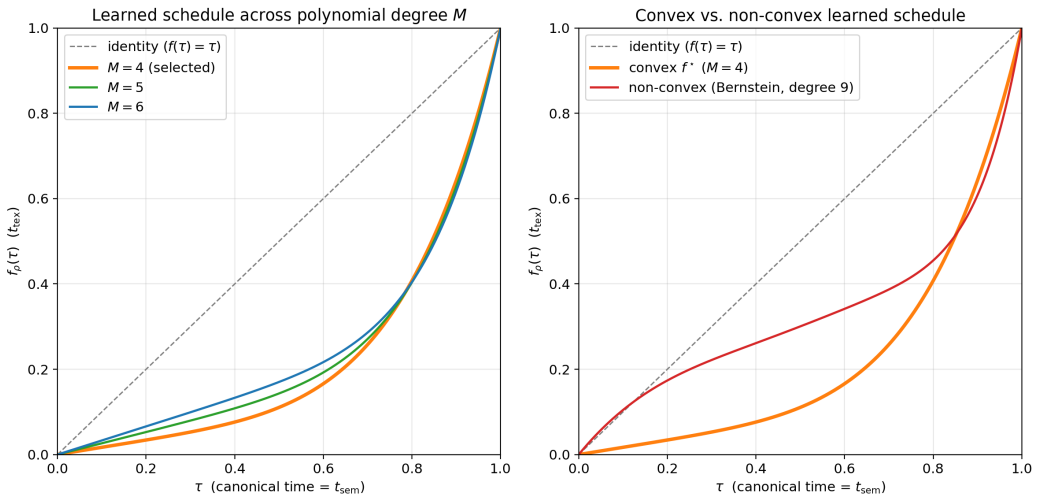


Figure 4: **Learned schedules in the ablations.** Left: probe-learned convex schedules for polynomial degrees $M \in \{4, 5, 6\}$. Right: the convex $f^\star$ and the probe-learned non-convex Bernstein schedule.

D.4 Spatial statistics of the semantic representations

An aggressive schedule means a smaller $f(\tau)$: semantics is farther denoised while texture remains at a noisier local time, so the probe learns how much semantic head start is needed before semantic conditioning helps texture prediction. The relevant distinction is local, position-aligned information useful for texture prediction, not global semantic strength. SemVAE retains some such detail, so a partially resolved semantic state already provides local cues and texture need not wait long. The PCA representations are smoother and more global; their local values carry little direct texture information, so texture benefits mainly once the semantic field is cleaner and more coherent, inducing a larger delay.

Table 7 quantifies this with two statistics of the standardized 16×16 semantic maps: the lag-1 spatial autocorrelation, which is high for smooth maps, and the low-pass R^2, the fraction of variance that survives $16 \to 8 \to 16$ down/up-sampling. Both increase monotonically with schedule aggressiveness: representations richer in local, texture-relevant information require less semantic lead.

Table 7: **Spatial statistics of the semantic representations** and the shape of the learned schedule (Figure 3).

Semantic latent	Lag-1 autocorr.	Low-pass R^2	Learned schedule
SemVAE	0.59	0.67	closest to identity
DINO-PCA	0.75	0.79	intermediate
CLIP-PCA	0.80	0.83	most aggressive

D.5 Reading of the schedule objective

The schedule objective $\mathcal{L}_\rho$ in Eq. (17) can be read as the Lagrangian form of a constrained problem: find a coupling with low velocity-prediction error while keeping the global-time trajectory energy within a stable range. Under squared loss, the population-optimal predictor is the conditional mean, and its minimum risk is the total conditional variance,

$$\inf_h \mathbb{E}\big[\|v - h(z_\tau, \tau, c)\|^2\big] = \mathbb{E}\big[\operatorname{tr} \operatorname{Cov}(v \mid z_\tau, \tau, c)\big], \tag{26}$$

where v is the group-velocity target, z_τ is the joint noisy semantic–texture state, and the infimum runs over all predictors h. Because the corrected weights hold the local-time density fixed, a lower minimum of $\mathcal{L}_{\text{flow}}$ identifies a coupling under which the required velocities are less ambiguous from the model's input, rather than a schedule that merely reweights noise levels.

Predictability alone does not account for the numerical cost of traversing the path with finite NFEs. An extreme semantic-first schedule may make texture prediction easier while postponing nearly all texture motion to a narrow interval, where $t'_{\mathrm{tex}}(\tau;\rho)$ and the global-time speed become large. $\mathcal{R}_{\mathrm{kin}}$ is the squared-speed energy of the predicted global-time velocity along the training interpolation and directly penalizes such concentration. This matters because both branches share one global clock and one NFE budget, so the solver cannot refine the semantic and texture time grids independently. We chose the kinetic energy because it targets the observed collapse; schedule-only alternatives such as an L_2 penalty on the deviation of t'_{tex} from one or total variation do not account for the denoiser-dependent latent velocity, and a penalty on the ODE condition number would be substantially more expensive to estimate.

D.6 Sampling variability of the headline results

We generate three independent 50K class-balanced sample sets from each fixed checkpoint, varying only the sampling seed. For every run, we generate 50 samples per ImageNet class using sampling seeds $(0, 1, 2)$, the same ADM reference statistics, a byte-identical FID implementation, the same solver, and the same 2-GPU layout, with each method's released sampling implementation and checkpoints and the corresponding weak checkpoint for AutoGuidance. Ours is evaluated with AutoGuidance scale $w = 1.4$ in this table and SFD-XL with its released sampling configuration. The standard deviations in Table 8 are sampling-seed variation for fixed checkpoints.

Table 8: **FID-50K under three sampling seeds** for the 1M-iteration Ours-XL checkpoint and the released 4M-iteration SFD-XL checkpoint. Ours uses AutoGuidance scale $w = 1.4$.

	seed 0	seed 1	seed 2	mean $\pm$ std
Ours-XL 1M, unguided	2.364	2.407	2.439	2.404 ± 0.038
SFD-XL 4M, unguided	2.607	2.630	2.606	2.614 ± 0.014
Ours-XL 1M, AutoGuidance ($w = 1.4$)	1.035	1.013	1.020	1.023 ± 0.011
SFD-XL 4M, AutoGuidance	1.065	1.046	1.059	1.056 ± 0.010

All three evaluations of Ours (2.364–2.439) are lower than all three evaluations of the released SFD-XL checkpoint (2.606–2.630), with a minimum separation of 0.167. With AutoGuidance, the highest FID among our three runs (1.035) remains below the lowest among the three SFD-XL runs (1.046). Our seed-0 unguided value is within 0.01 of the 2.37 in Table 1. The released SFD-XL checkpoint gives 2.614 ± 0.014 in this re-evaluation, compared with the 2.54 reported by Pan et al. [18] and listed in Table 1.

E Proofs for Main Results

E.1 Proof of Theorem 3.1

In addition to claims (I) and (II) of Theorem 3.1, we state and prove the explicit form of the scheduled distribution and the score form of the global flow:

Interpolating distribution. The scheduled distribution $p_\tau(\cdot \mid c)$ has the explicit form

$$p_\tau(\cdot \mid c) = \big((A_\tau)_{\#} p_1(\cdot \mid c)\big) * \mathcal{N}(0, \Sigma_\tau), \tag{27}$$

where $A_\tau = \mathrm{diag}\big(t_{\mathrm{tex}}(\tau) I_{d_{\mathrm{tex}}}, t_{\mathrm{sem}}(\tau) I_{d_{\mathrm{sem}}}\big)$, and $\Sigma_\tau = \mathrm{diag}\big((1 - t_{\mathrm{tex}}(\tau))^2 I_{d_{\mathrm{tex}}}, (1 - t_{\mathrm{sem}}(\tau))^2 I_{d_{\mathrm{sem}}}\big)$, and $(A_\tau)_{\#} p_1$ denotes the pushforward of p_1 under $x \mapsto A_\tau x$. Thus p_τ starts at the base Gaussian distribution when $\tau = 0$ and ends at the clean data-latent distribution when $\tau = 1$, with endpoint identities understood as weak limits.

Consequently, the ideal global flow can be written in terms of the overall score of p_τ as

$$\begin{aligned} V^\star(z, \tau, c) = \Bigg[& \frac{t'_{\mathrm{tex}}(\tau)}{t_{\mathrm{tex}}(\tau)} z^{\mathrm{tex}} + \frac{t'_{\mathrm{tex}}(\tau)(1 - t_{\mathrm{tex}}(\tau))}{t_{\mathrm{tex}}(\tau)} \nabla_{z^{\mathrm{tex}}} \log p_\tau(z \mid c), \\ & \frac{t'_{\mathrm{sem}}(\tau)}{t_{\mathrm{sem}}(\tau)} z^{\mathrm{sem}} + \frac{t'_{\mathrm{sem}}(\tau)(1 - t_{\mathrm{sem}}(\tau))}{t_{\mathrm{sem}}(\tau)} \nabla_{z^{\mathrm{sem}}} \log p_\tau(z \mid c) \Bigg]. \end{aligned} \tag{28}$$

Proof sketch. The interpolating distribution follows by writing the asynchronous state as an anisotropically scaled data latent plus anisotropic Gaussian noise. The continuity equation follows by differentiating expectations of smooth test functions along the asynchronous path and conditioning the sample-wise velocity on the observed state. The score identities follow from Tweedie's identity applied separately to the texture and semantic Gaussian channels.

We prove the three claims in order. Throughout the proof, all distributions are conditioned on the class label c, and we suppress this conditioning when it is clear from context.

Interpolating distribution. For a fixed global time τ, define

$$A_\tau = \operatorname{diag}\big(t_{\text{tex}}(\tau) I_{d_{\text{tex}}}, t_{\text{sem}}(\tau) I_{d_{\text{sem}}}\big), \qquad B_\tau = \operatorname{diag}\big((1 - t_{\text{tex}}(\tau)) I_{d_{\text{tex}}}, (1 - t_{\text{sem}}(\tau)) I_{d_{\text{sem}}}\big).$$

By the component-wise interpolation in Eq. (2), the asynchronous state can be written as

$$z(\tau) = A_\tau x_1 + B_\tau x_0, \tag{29}$$

where $x_1 \sim p_1(\cdot \mid c)$ and $x_0 \sim \mathcal{N}(0, I_{d_{\text{tex}}+d_{\text{sem}}})$. Therefore, conditional on x_1, the random variable $z(\tau)$ is Gaussian:

$$z(\tau) \mid x_1 \sim \mathcal{N}\left(A_\tau x_1, B_\tau B_\tau^\top\right).$$

Since

$$B_\tau B_\tau^\top = \operatorname{diag}\big((1 - t_{\text{tex}}(\tau))^2 I_{d_{\text{tex}}}, (1 - t_{\text{sem}}(\tau))^2 I_{d_{\text{sem}}}\big) = \Sigma_\tau,$$

marginalizing over $x_1 \sim p_1(\cdot \mid c)$ gives

$$p_\tau(\cdot \mid c) = \big((A_\tau)_\# p_1(\cdot \mid c)\big) * \mathcal{N}(0, \Sigma_\tau),$$

which is Eq. (27). At $\tau = 0$, $A_\tau = 0$ and $\Sigma_\tau = I$, so $p_\tau = p_0$. At $\tau = 1$, $A_\tau = I$ and $\Sigma_\tau = 0$, so $p_\tau = p_1$, with the latter identity understood as a weak limit.

Continuity equation. For each endpoint pair (x_0, x_1), differentiating Eq. (4) with respect to global time gives

$$\dot{z}(\tau) = \left[t'_{\text{tex}}(\tau) u^{\text{tex}}(t_{\text{tex}}(\tau)),\ t'_{\text{sem}}(\tau) u^{\text{sem}}(t_{\text{sem}}(\tau))\right]. \tag{30}$$

Let $\varphi \in C_c^\infty(\mathbb{R}^{d_{\text{tex}}+d_{\text{sem}}})$ be a smooth test function. Then

$$\frac{d}{d\tau}\mathbb{E}\left[\varphi(z(\tau)) \mid c\right] = \mathbb{E}\left[\nabla_z \varphi(z(\tau))^\top \dot{z}(\tau) \mid c\right]. \tag{31}$$

Taking conditional expectation of the sample-wise global-time velocity given $z(\tau) = z$ yields

$$\begin{aligned}
\mathbb{E}\left[\dot{z}(\tau) \mid z(\tau) = z,\ c\right] &= \left[t'_{\text{tex}}(\tau)\mathbb{E}\left[u^{\text{tex}}(t_{\text{tex}}(\tau)) \mid z(\tau) = z,\ c\right],\right. && (32)\\
&\qquad \left. t'_{\text{sem}}(\tau)\mathbb{E}\left[u^{\text{sem}}(t_{\text{sem}}(\tau)) \mid z(\tau) = z,\ c\right]\right] && (33)\\
&= V^\star(z, \tau, c), && (34)
\end{aligned}$$

where the last equality uses Eq. (3). Therefore,

$$\frac{d}{d\tau}\int \varphi(z) p_\tau(z \mid c)\, dz = \int \nabla_z \varphi(z)^\top V^\star(z, \tau, c) p_\tau(z \mid c)\, dz.$$

Integrating by parts gives

$$\frac{d}{d\tau}\int \varphi(z) p_\tau(z \mid c)\, dz = -\int \varphi(z) \nabla_z \cdot \left(p_\tau(z \mid c) V^\star(z, \tau, c)\right)\, dz.$$

Since this holds for all smooth compactly supported test functions φ, p_τ satisfies the continuity equation in the weak sense:

$$\partial_\tau p_\tau(z \mid c) + \nabla_z \cdot \left(p_\tau(z \mid c) V^\star(z, \tau, c)\right) = 0.$$

Under the usual regularity assumptions ensuring existence and uniqueness of the ODE flow, the probability-flow ODE

$$\frac{dZ_\tau}{d\tau} = V^\star(Z_\tau, \tau, c)$$

has marginals $p_\tau(\cdot \mid c)$. Since p_0 is Gaussian and p_1 is the clean data-latent distribution, this ODE transports noise to data.

Score characterization. Fix local times $(t_{\text{tex}}, t_{\text{sem}})$ and a group $g \in \{\text{tex}, \text{sem}\}$. Write $t_g = t_{\text{tex}}$ if $g = \text{tex}$ and $t_g = t_{\text{sem}}$ if $g = \text{sem}$. For the linear Gaussian noising channel,

$$z^g = t_g x_1^g + (1 - t_g) x_0^g, \qquad x_0^g \sim \mathcal{N}(0, I).$$

Let

$$s_g(z, t_{\text{tex}}, t_{\text{sem}}, c) = \nabla_{z^g} \log p_{t_{\text{tex}}, t_{\text{sem}}}(z \mid c).$$

Differentiating the conditional Gaussian density with respect to z^g and then averaging over the posterior of x_1 gives the group-wise Tweedie identity

$$s_g(z, t_{\text{tex}}, t_{\text{sem}}, c) = -\frac{\mathbb{E}\left[x_0^g \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right]}{1 - t_g}. \tag{35}$$

Equivalently,

$$\mathbb{E}\left[x_0^g \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right] = -(1 - t_g) s_g(z, t_{\text{tex}}, t_{\text{sem}}, c). \tag{36}$$

Using $z^g = t_g x_1^g + (1 - t_g) x_0^g$, we also have

$$\mathbb{E}\left[x_1^g \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right] = \frac{z^g - (1 - t_g)\mathbb{E}\left[x_0^g \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right]}{t_g} \tag{37}$$

$$= \frac{z^g + (1 - t_g)^2 s_g(z, t_{\text{tex}}, t_{\text{sem}}, c)}{t_g}. \tag{38}$$

Therefore the ideal local flow is

$$v_g^\star(z, t_{\text{tex}}, t_{\text{sem}}, c) = \mathbb{E}\left[x_1^g - x_0^g \mid z(t_{\text{tex}}, t_{\text{sem}}) = z,\ c\right] \tag{39}$$

$$= \frac{z^g}{t_g} + \frac{(1 - t_g)^2}{t_g} s_g(z, t_{\text{tex}}, t_{\text{sem}}, c) + (1 - t_g) s_g(z, t_{\text{tex}}, t_{\text{sem}}, c) \tag{40}$$

$$= \frac{z^g}{t_g} + \frac{1 - t_g}{t_g} s_g(z, t_{\text{tex}}, t_{\text{sem}}, c). \tag{41}$$

Specializing this identity to $g = \text{tex}$ and to $g = \text{sem}$ gives Eq. (7). Finally, along the scheduled path, $p_\tau = p_{t_{\text{tex}}(\tau), t_{\text{sem}}(\tau)}$. Substituting the two local score forms into Eq. (6) gives Eq. (28).

E.2 Proof of Theorem E.1

Theorem E.1 (Population optimum of asynchronous flow matching)**.** *Fix a continuously differentiable, nondecreasing schedule $t_{\text{tex}}(\cdot; \rho)$. Assume $\omega_{\text{tex}}(\tau, \rho) > 0$ and $\omega_{\text{sem}}(\tau, \rho) > 0$ depend only on the sampled local times, not on the endpoint pair (x_0, x_1), and that the weighted risk $\mathbb{E}\big[\sum_g \omega_g(\tau, \rho) \|u^g\|_g^2\big]$ is finite. If each $\hat{v}^g$ ranges over all measurable functions of $(z, t_{\text{tex}}, t_{\text{sem}}, c)$, then any minimizer over θ of Eq. (10) satisfies*

$$\hat{v}_{\theta^\star}^g\big(z, t_{\text{tex}}(\tau; \rho), \tau, c\big) = v_g^\star\big(z, t_{\text{tex}}(\tau; \rho), \tau, c\big), \qquad g \in \{\text{tex}, \text{sem}\}, \tag{42}$$

for $p_\tau^\rho(z \mid c)$-almost every z and almost every τ.

Proof sketch. For each fixed pair of local times, the squared-error minimizer is the conditional mean of the corresponding local-time velocity. Positive time weights change how local-time pairs are averaged, but not this pointwise conditional mean.

We prove the statement for the texture component; the semantic component is identical. Fix a schedule $t_{\text{tex}}(\tau; \rho)$. For a fixed global time τ, write

$$T_{\text{tex}} = t_{\text{tex}}(\tau; \rho), \qquad T_{\text{sem}} = \tau, \qquad Z = z(T_{\text{tex}}, T_{\text{sem}}).$$

The texture component of the weighted flow objective at this time is

$$\omega_{\text{tex}}(\tau, \rho)\mathbb{E}\left[\left\|\hat{v}_\theta^{\text{tex}}(Z, T_{\text{tex}}, T_{\text{sem}}, c) - u^{\text{tex}}(T_{\text{tex}})\right\|_{\text{tex}}^2\right].$$

Since $\omega_{\text{tex}}(\tau, \rho) > 0$ and depends only on the sampled local times, it does not change the pointwise minimizer with respect to the predicted function. Thus it suffices to minimize

$$\mathbb{E}\left[\left\|h(Z, T_{\text{tex}}, T_{\text{sem}}, c) - u^{\text{tex}}(T_{\text{tex}})\right\|_{\text{tex}}^2\right]$$

over measurable functions h. By the standard L_2-projection identity, the minimizer is the conditional mean

$$h^\star(z, T_{\text{tex}}, T_{\text{sem}}, c) = \mathbb{E}\left[u^{\text{tex}}(T_{\text{tex}}) \mid Z = z,\ T_{\text{tex}},\ T_{\text{sem}},\ c\right].$$

Because T_{tex} and T_{sem} are fixed by the sampled global time τ, this conditional mean is exactly

$$v^\star_{\text{tex}}(z, T_{\text{tex}}, T_{\text{sem}}, c) = v^\star_{\text{tex}}\big(z, t_{\text{tex}}(\tau;\rho), \tau, c\big),$$

as defined in Eq. (3). Therefore, in the infinite-data and infinite-capacity limit,

$$\hat{v}^{\text{tex}}_{\theta^\star}\big(z, t_{\text{tex}}(\tau;\rho), \tau, c\big) = v^\star_{\text{tex}}\big(z, t_{\text{tex}}(\tau;\rho), \tau, c\big)$$

for $p^\rho_\tau(z \mid c)$-almost every z and almost every τ.

The same argument applied to the semantic component gives

$$\hat{v}^{\text{sem}}_{\theta^\star}\big(z, t_{\text{tex}}(\tau;\rho), \tau, c\big) = v^\star_{\text{sem}}\big(z, t_{\text{tex}}(\tau;\rho), \tau, c\big).$$

Because the model is assumed to have infinite capacity, the two component predictions can simultaneously realize their respective conditional means. Positive time weights only determine how local-time pairs are averaged in the global objective; they do not alter the pointwise conditional-mean target at any fixed local-time pair.

E.3 Proof of Lemma 3.2

By definition,

$$\mathbb{E}_{\tau\sim\mathcal{U}(0,1)}\left[\omega_{\text{tex}}(\tau,\rho)\ell_{\text{tex}}\big(\theta; t_{\text{tex}}(\tau;\rho), \tau\big)\right] = \int_0^1 \omega_{\text{tex}}(\tau,\rho)\ell_{\text{tex}}\big(\theta; t_{\text{tex}}(\tau;\rho), \tau\big)\, d\tau. \tag{43}$$

Apply the change of variables $s = t_{\text{tex}}(\tau;\rho)$. Since $t_{\text{tex}}(\cdot;\rho)$ is a C^1 bijection of $[0,1]$ with $t'_{\text{tex}} > 0$ on $(0,1)$,

$$\tau = t^{-1}_{\text{tex}}(s;\rho), \qquad d\tau = \frac{ds}{t'_{\text{tex}}\big(t^{-1}_{\text{tex}}(s;\rho);\rho\big)}.$$

Therefore the texture term becomes

$$\int_0^1 \frac{\omega_{\text{tex}}\big(t^{-1}_{\text{tex}}(s;\rho),\rho\big)}{t'_{\text{tex}}\big(t^{-1}_{\text{tex}}(s;\rho);\rho\big)}\ell_{\text{tex}}\big(\theta; s, t^{-1}_{\text{tex}}(s;\rho)\big)\, ds. \tag{44}$$

For this to equal the first term of Eq. (12) for arbitrary ℓ_{tex}, the ratio multiplying ℓ_{tex} must equal one almost everywhere. This gives $\omega_{\text{tex}}(\tau,\rho) = t'_{\text{tex}}(\tau;\rho)$. The semantic term requires no change of variables because $t_{\text{sem}}(\tau) = \tau$:

$$\mathbb{E}_{\tau\sim\mathcal{U}(0,1)}\left[\omega_{\text{sem}}(\tau,\rho)\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(\tau;\rho), \tau\big)\right] = \int_0^1 \omega_{\text{sem}}(s,\rho)\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(s;\rho), s\big)\, ds. \tag{45}$$

Matching the second term of Eq. (12) for arbitrary ℓ_{sem} therefore requires $\omega_{\text{sem}}(\tau,\rho) = 1$ almost everywhere.

E.4 Explicit forms of the corrected objective

Plugging Eq. (14) into Eq. (10), the corrected flow objective of Eq. (15) reads, equivalently,

$$\mathcal{L}_{\text{flow}}(\theta,\rho) = \mathbb{E}_{\tau\sim\mathcal{U}(0,1)}\left[w_{\text{tex}}\ \text{sg}\left(t'_{\text{tex}}(\tau;\rho)\right)\ell_{\text{tex}}\big(\theta; t_{\text{tex}}(\tau;\rho),\tau\big) + w_{\text{sem}}\,\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(\tau;\rho),\tau\big)\right]. \tag{46}$$

Ignoring the stop-gradient annotation, Eq. (46) has the local-time form

$$\mathcal{L}_{\text{flow}}(\theta,\rho) = w_{\text{tex}}\mathbb{E}_{s\sim\mathcal{U}(0,1)}\left[\ell_{\text{tex}}\big(\theta; s, t^{-1}_{\text{tex}}(s;\rho)\big)\right] + w_{\text{sem}}\mathbb{E}_{s\sim\mathcal{U}(0,1)}\left[\ell_{\text{sem}}\big(\theta; t_{\text{tex}}(s;\rho), s\big)\right]. \tag{47}$$

The two forms have the same value for every ρ. With the stop-gradient, the schedule gradient is that of the global-time form in Eq. (46) with t'_{tex} held fixed.

F Additional Qualitative Results

We show additional class-conditional samples from the final Ours-XL model at 256×256, one figure per ImageNet class. All samples are generated with AutoGuidance ($w = 1.5$), matching the setting of Table 2.



Figure 5: **Qualitative samples from Ours-XL trained at** 256×256 **resolution.** We show selected class-conditional samples generated by the final Ours-XL model using the same AutoGuidance setting as in Table 2.



Figure 6: Cockatoo (class 89). Samples from Ours-XL with AutoGuidance, $w = 1.5$.



Figure 7: Husky (class 250). Samples from Ours-XL with AutoGuidance, $w = 1.5$.



Figure 8: Lion (class 291). Samples from Ours-XL with AutoGuidance, $w = 1.5$.



Figure 9: Balloon (class 417). Samples from Ours-XL with AutoGuidance, $w = 1.5$.



Figure 10: Coral reef (class 973). Samples from Ours-XL with AutoGuidance, $w = 1.5$.



Figure 11: Volcano (class 980). Samples from Ours-XL with AutoGuidance, $w = 1.5$.

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 in the abstract and introduction accurately reflect the paper's contributions and scope. They are supported by the experimental results, and we avoid claims beyond the evaluated settings.

   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: The paper includes a dedicated Limitations section discussing the experimental scope, assumptions, computational considerations, and scenarios where the method may require further validation.

   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: Theorems 3.1 and E.1 and Lemma 3.2 (Section 3) state their assumptions in full; complete proofs are given in Appendix E.

   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: The paper provides the information needed to reproduce the main experimental results, including datasets, model configurations, training details, evaluation protocols, and implementation details.

   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: The code is released at https://github.com/bsq532087/LWD, together with instructions for environment setup, data preparation, training, and evaluation to reproduce the main experimental results.

   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: The paper specifies the experimental settings needed to understand the results, including data splits, model configurations, hyperparameters, optimizer choices, training details, and evaluation protocols.

   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: Appendix D.6 reports the mean and standard deviation of FID-50K over three sampling seeds for the headline checkpoints of Ours-XL and SFD-XL, and states that this captures sampling variation for fixed checkpoints. Other entries follow the single-evaluation convention of prior ImageNet 256×256 work (DiT, SiT, REPA, SFD).

   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: Hardware and total runtime for the schedule probe and the main training run are reported in Appendix B.

   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 research conforms with the NeurIPS Code of Ethics. The paper proposes a methodological improvement to class-conditional ImageNet generation; no human subjects, sensitive data, or restricted assets are involved.

   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: [N/A]

    Justification: The work is a methodological improvement to existing class-conditional latent diffusion training. It does not introduce qualitatively new generative capabilities and inherits the broader-impact considerations of the underlying class-conditional ImageNet generation literature, which it does not change.

    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: We do not release a new high-risk pretrained model or scraped dataset; the paper improves the training procedure for an existing class-conditional generation paradigm.

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: All existing assets used in the paper (the SFD codebase and released ImageNet latent dataset, the LightningDiT architecture, the DINOv2 and CLIP feature encoders, and the ImageNet classification dataset) are credited via citation in the relevant sections, and we follow the original licenses and terms of use of each asset.

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 the training and evaluation code together with the final Ours-XL model checkpoint via `https://github.com/bsq532087/LWD`, with README documentation covering environment setup, dataset preparation, training, and evaluation.

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 research with human subjects.

    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 research with human subjects.

    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: LLMs are not a component of the core method. The proposed method consists of a learnable convex schedule and a denoising transformer trained with flow matching, neither of which involves an LLM.

    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.