Stable Continuous-Time Consistency Distillation: An Empirical Study with a Multistep Extension

Anonymous authors
Paper under double-blind review

Abstract

Continuous-time consistency models are an attractive route to few-step generation: they remove the discretization error inherent to the timestep grid that discrete-time consistency models depend on. However, until recently, training them was unstable, and Lu & Song (2025) address this with the TrigFlow formulation and a coordinated set of architectural and training-objective changes. We conduct an independent empirical study of this framework on CIFAR-10 and ImageNet-64. No open-weight TrigFlow teachers exist at these scales, so we train them from scratch following Lu & Song (2025)'s prescription and then verify the consistency distillation (sCD) and consistency training (sCT) behavior of students initialized from these teachers. On CIFAR-10 our students reproduce the reported FID at one and two steps. On ImageNet-64 they fall short, a gap we trace to our teacher and to architectural choices the recipe leaves unspecified. We also report a parameter-count gap of about 5% between the model size stated in the paper and a faithful re-implementation, which we attribute to the pixel-normalization in Adaptive Double Normalization.

We then propose *Multistep sCD (MS-sCD)*, a continuous-time, segment-conditioned multistep generalization of sCD. The segment-conditioning idea is borrowed from the discrete-time multistep consistency distillation of Heek et al. (2024). The rest of the algorithm follows sCD. MS-sCD replaces sCD's fixed two-step sampling schedule with a trained segment boundary and recovers sCD at $M = 1$. We report FID at $M = 2, 4, 8$ and compare against the discrete-time MSCD baseline. Code, training scripts, and all teacher and student checkpoints are provided as anonymized supplementary material with this submission, and will be released publicly upon acceptance.

1 Introduction

1.1 Motivation

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020; Karras et al., 2022) produce high-quality samples but require many sequential network evaluations, motivating an active line of work on reducing the sampling cost. Progressive distillation (Salimans & Ho, 2022) repeatedly halves the sampling steps, and each new student learns to match two steps of the one before it. Consistency models (Song & Dhariwal, 2024a) define a network $f_\theta(x_t, t)$ that maps any point on a probability-flow ODE trajectory to its origin x_0, enforced by the boundary condition $f_\theta(x, 0) = x$ together with a consistency loss between predictions at adjacent timesteps. The trained network supports one or few-step generation. Subsequent work on discrete-time CMs (Song & Dhariwal, 2024b; Geng et al., 2025) refines the training methodology through the pseudo-Huber loss, manually designed timestep and weighting schedules, and related changes. Multistep consistency distillation (MSCD) (Heek et al., 2024) trains a segment-conditioned network for M-step generation and runs a DDIM (Song et al., 2021) teacher step inside each segment. To correct the variance shortfall of finite DDIM integration, Heek et al. add the aDDIM sampler. A separate family of distillation methods matches the distribution of the student's samples to the teacher's, including variational score distillation (Wang et al., 2023; Yin et al., 2024) and the moment-matching approach of Salimans et al. (2024).

The CM literature above largely operates in discrete time, where the training loss depends on a chosen timestep grid. Song & Dhariwal (2024a) also propose a continuous-time variant (their Remark 10) that removes the grid, but report training instability. Song & Dhariwal (2024b); Geng et al. (2025) likewise restrict their experiments to the discrete case. sCM (Lu & Song, 2025) addresses this instability through changes to the model parameterization (TrigFlow, identity time conditioning, positional time embeddings, adaptive double normalization) and to the training objective (tangent normalization, adaptive weighting, tangent warmup), and reports stable continuous-time CMs that scale to ImageNet-512 at 1.5B parameters.

1.2 Contributions

Our contributions are:

1. **Reproduction with implementation clarifications.** We reproduce sCM's TrigFlow teacher and sCD/sCT students on CIFAR-10 (unconditional, DDPM++ backbone) and ImageNet-64 (class-conditional, EDM2-S backbone), following the two configurations sCM uses. We make explicit the implementation details required to run the continuous-time CM training loop.
2. **Continuous-time MS-sCD (extension).** We derive and implement a continuous-time, segment-conditioned multistep generalization of sCD. The derivation follows directly from sCM's TrigFlow algebra with $\Delta t := t - t_k$ replacing t, the segment boundary t_k entering as a zero-initialized conditioning input, and t_k receiving a zero tangent in the JVP call.
3. **Matched baselines.** We reproduce MSCD (Heek et al., 2024) and moment matching (Salimans et al., 2024) on the same EDM2-S backbone and teacher as our sCM and MS-sCD runs, so multistep comparisons isolate the continuous-time formulation.

2 Background

2.1 Diffusion process and probability flow ODE

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) define a forward process that gradually corrupts a data sample x_0 with Gaussian noise according to a schedule (α_t, σ_t), producing $x_t = \alpha_t\, x_0 + \sigma_t\, z$ with $z \sim \mathcal{N}(0, I)$. Song et al. (2020) show that the associated marginals $p_t(x_t)$ are also generated by a deterministic probability flow ODE (PF-ODE), $dx_t/dt = f(t)\, x_t - \frac{1}{2} g(t)^2\, \nabla_{x_t} \log p_t(x_t)$, whose vector field is determined by the score. Sampling reduces to integrating this ODE from $t = T$ (approximately Gaussian) back to $t = 0$ (data). The EDM framework (Karras et al., 2022) adopts a variance-exploding schedule and parameterizes the denoiser as $D_\theta(x_t) = c_{\text{skip}}(t)\, x_t + c_{\text{out}}(t)\, F_\theta(c_{\text{in}}(t)\, x_t,\, c_{\text{noise}}(t))$, with F_θ a neural network and $(c_{\text{skip}}, c_{\text{out}}, c_{\text{in}}, c_{\text{noise}})$ preconditioners chosen so that the training objective has unit variance across timesteps.

2.2 Flow matching

Flow matching (Lipman et al., 2022; Liu et al., 2022; Albergo et al., 2025) fixes an interpolation $x_t = \alpha_t\, x_0 + \sigma_t\, z$ between data and noise and trains a velocity field $v_\theta(x_t, t)$ to match $\dot{x}_t = \alpha'_t\, x_0 + \sigma'_t\, z$, so the PF-ODE becomes $dx_t/dt = v_\theta(x_t, t)$. Different choices of (α_t, σ_t) recover variance-preserving diffusion, rectified flow (Liu et al., 2022), or stochastic interpolants (Albergo et al., 2025).

2.3 TrigFlow

TrigFlow (Lu & Song, 2025) unifies EDM and flow matching by parameterizing noisy samples as

$$x_t = \cos(t)\, x_0 + \sin(t)\, z, \quad z \sim \mathcal{N}(0, \sigma_d^2 I), \quad t \in [0, \pi/2]. \tag{1}$$

The ideal velocity along this trajectory is $\dot{x}_t^* = \cos(t)\, z - \sin(t)\, x_0$. A TrigFlow network F_θ approximates this velocity, trained via:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{x_0, z, t}\big[\|\sigma_d\, F_\theta(x_t/\sigma_d,\, t) - \dot{x}_t^*\|^2\big]. \tag{2}$$

The PF-ODE is $dx_t/dt = \sigma_d\, F_\theta(x_t/\sigma_d, t)$. TrigFlow satisfies EDM's preconditioner requirements with $c_{\text{skip}}(t) = \cos(t)$, $c_{\text{out}}(t) = -\sigma_d \sin(t)$, $c_{\text{in}}(t) = 1/\sigma_d$, and $c_{\text{noise}}(t) = t$ (identity time conditioning, see Lu & Song 2025 for the derivation).

CM parameterization. The TrigFlow CM (Lu & Song, 2025) parameterizes the consistency function as a one-step TrigFlow-DDIM step from t to 0:

$$f_\theta(x_t, t) = \cos(t)\, x_t - \sin(t)\, \sigma_d\, F_\theta(x_t/\sigma_d,\, t). \tag{3}$$

This automatically satisfies the boundary condition $f_\theta(x_0, 0) = x_0$ because $\sin(0) = 0$.

2.4 Continuous-time CM training

Taking the discrete consistency loss to the continuous-time limit (Song's Remark 10 in Song & Dhariwal 2024a), the training gradient becomes:

$$\nabla_\theta\, \mathbb{E}_{x_t,t}\left[w(t)\; f_\theta^\top(x_t, t)\; \frac{df_{\theta^-}(x_t, t)}{dt} \right], \tag{4}$$

where θ^- denotes a stop-gradient on θ (no separate EMA target network is used, following Song & Dhariwal 2024b; Lu & Song 2025). Expanding the time derivative along the PF-ODE gives

$$\frac{df_{\theta^-}}{dt} = \cos(t)\left(\frac{dx_t}{dt} - \sigma_d\, F_{\theta^-} \right) - \sin(t)\left(x_t + \sigma_d\, \frac{dF_{\theta^-}}{dt} \right), \tag{5}$$

where the second term carries dF_{θ^-}/dt, computed by a forward-mode JVP and the source of training instability.

2.5 sCM's six stabilization fixes

sCM identifies the instability in equation 5 as:

$$\sin(t)\, \partial_t F_{\theta^-} = \sin(t) \cdot \frac{\partial c_{\text{noise}}}{\partial t} \cdot \frac{\partial\, \text{emb}(c_{\text{noise}})}{\partial c_{\text{noise}}} \cdot \frac{\partial F_{\theta^-}}{\partial\, \text{emb}}, \tag{6}$$

and attacks each factor in turn. Three of its six fixes change the architecture, and the other three change the training objective.

The architectural fixes keep the product in equation 6 bounded as $t \to \pi/2$. The first culprit is EDM's time conditioning $c_{\text{noise}} = \log(\sigma_d \tan t)$, which makes $\sin(t)\, \partial_t c_{\text{noise}} = 1/\cos(t)$ blow up at the boundary, so sCM switches to identity conditioning $c_{\text{noise}}(t) = t$ and is left with the bounded $\sin(t)$. The second is the Fourier time embedding, whose large effective scale magnifies $\partial\, \text{emb}/\partial c_{\text{noise}}$. A positional embedding drops that scale to about 0.02. The third change is Adaptive Double Normalization (ADN), which replaces AdaGN (Dhariwal & Nichol, 2021) with

$$y = \text{norm}(y) \odot \text{pnorm}(s(t)) + \text{pnorm}(b(t)), \tag{7}$$

where norm is a group norm and pnorm is pixel normalization (Section 3.4). Capping the scale and shift vectors this way holds $\partial F_{\theta^-}/\partial\, \text{emb}$ down throughout training.

The other three fixes act on the gradient itself. Tangent normalization divides the raw tangent by its own Euclidean norm plus a constant, $g \to g/(\|g\|_2 + c)$ with $c = 0.1$ (Section 4.2). Adaptive weighting rewrites the inner-product gradient equation 4 as an MSE, $\nabla_\theta \mathbb{E}[F_\theta^\top y] = \frac{1}{2} \nabla_\theta \mathbb{E}[\|F_\theta - F_{\theta^-} + y\|^2]$, and trains a log-variance head $w_\phi(t)$ next to F_θ. Finally, the tangent warmup ramps the coefficient $r = \min(1,\, \text{iter}/H)$ on the $\sin(t)$ term from 0 to 1 over the first $H = 10{,}000$ iterations.

2.6 Multistep Consistency Distillation (Heek et al., 2024)

Heek et al. discretize the generation trajectory into M segments with boundaries $0 = t_0 < t_1 < \cdots < t_M$, condition the network on the segment index, and use DDIM teacher steps within each segment. They propose the aDDIM correction to compensate for the variance shortfall of finite DDIM steps. The paper trains on the discrete grid. It also notes that continuous-time training is possible provided the model takes an extra conditioning signal that identifies the current segment (so the prediction can change discontinuously across segment boundaries), and reports that early experiments with this variant worked comparably (Heek et al., 2024, §3). The continuous-time training is not pursued or reported. Our MS-sCD (Section 6) follows this remark by adding a segment-boundary input t_k to the network and training in continuous time within the TrigFlow CM framework.

3 TrigFlow Teacher: Architecture and Implementation

3.1 Backbones

We train at two settings, matching sCM's own backbone choices. CIFAR-10 (unconditional, 32×32) uses the Score SDE DDPM++ network of Song et al. (2020) with the EDM 2022 (Karras et al., 2022) recipe. ImageNet-64 (class-conditional, 64×64) uses the magnitude-preserving EDM2 UNet (Karras et al., 2024). Both carry the TrigFlow parameterization, identity time conditioning, positional time embeddings, and ADN (Section 2.5). What separates them is the normalization machinery. The EDM2 backbone is built from magnitude-preserving convolutions and uses EDM2's `normalize` ($\varepsilon = 10^{-4}$) for the pixel normalization inside ADN. DDPM++ has no weight-normalized layers and instead applies the PGGAN pixel norm (Karras et al., 2018) ($\varepsilon = 10^{-8}$) that sCM cites. The rest of this section covers the two implementation choices the paper does not pin down: which ADN block variant we run, and how the adaptive-weighting head and its loss are realized.

3.2 ADN block and parameter count

ADN modulates every residual block as $y \cdot \mathrm{pnorm}(s) + \mathrm{pnorm}(b)$, with the scale s and shift b produced by a single linear projection of the time embedding. Relative to EDM2's scale-only modulation $y \cdot (c + 1)$, this scale-and-shift form requires the time-conditioning projection in each block to emit $2C$ channels instead of C, doubling the projection's weight tensor and adding roughly $C \cdot d_{\mathrm{emb}}$ parameters per block, where d_{emb} is the time-embedding dimension. Summed over the encoder and decoder blocks of the UNet, this accounts for the bulk of a parameter-count gap we observe between the size reported by sCM (280.2M for size S, the EDM2-S baseline figure) and our re-implementation, which has 293,922,753 parameters (≈294M, about 5% larger). The same offset is expected at the larger model sizes.

3.3 Adaptive-weighting head and loss form

The paper does not specify the architecture of $w_\phi(t)$. We use a fixed positional embedding (128 frequencies on t) followed by a single magnitude-preserving linear map to a scalar, giving one $w_\phi(t)$ per image. The head is part of the teacher checkpoint and initializes the student's weighting head.

For the loss, we minimize the EDM2 form

$$\ell_{\mathrm{ours}}(x_0, z, t) = \frac{\|\sigma_d F_\theta(x_t/\sigma_d, t) - v_t\|^2}{e^{w_\phi(t)}} + w_\phi(t), \tag{8}$$

computed elementwise (one $w_\phi(t)$ broadcast over the $D = C \times H \times W$ pixels of each sample) and reduced by summation over batch and pixels. The sCM appendix writes the corresponding scalar-per-image objective as

$$\ell_{\mathrm{sCM}}(x_0, z, t) = \frac{e^{w_\phi(t)}}{D} \|\sigma_d F_\theta(x_t/\sigma_d, t) - v_t\|^2 - w_\phi(t). \tag{9}$$

Both forms implement the EDM2 variance-weighting principle, and we use equation 8 throughout consistent with the EDM2 training our code is built on.

3.4 Pixel-normalization

We implement pnorm with EDM2's `normalize` (pixel normalization in the sense of Karras et al. 2018), applied along the channel dimension:

$$\mathrm{pnorm}(v) = \frac{v}{\mathrm{RMS}(v) + \varepsilon}, \quad \varepsilon = 10^{-4}, \tag{10}$$

for a length-C channel vector v. Appendix A.2 restates the full routine. The teacher training recipe and hyperparameters are deferred to Appendix F.

4 sCM Student: sCD and sCT

4.1 Loss formulation

The student is initialized from the teacher checkpoint. The loss minimized during sCD training is:

$$\mathcal{L}_{\mathrm{sCD}}(\theta, \phi) = \mathbb{E}_{x_0, z, t}\left[\frac{\|F_\theta - F_{\theta^-} - g\|_2^2}{D\, e^{w_\phi(t)}} + w_\phi(t)\right], \tag{11}$$

where D is the number of pixels, F_{θ^-} denotes the student forward pass with stop-gradient (see Section 2.4, with no separate EMA target maintained), and g is the normalized tangent (Section 4.2). The sCD teacher provides $dx_t/dt = \sigma_d\, F_{\mathrm{teacher}}(x_t/\sigma_d, t)$. For sCT, the "teacher" is the ground-truth ODE tangent: $dx_t/dt = \cos(t)\, z - \sin(t)\, x_0$ (no separate teacher network needed).

The raw tangent vector before normalization is:

$$\tilde{g} = -\cos^2(t)\left(\sigma_d\, F_{\theta^-} - \frac{dx_t}{dt}\right) - r \cdot \cos(t)\sin(t)\left(x_t + \sigma_d\, \frac{dF_{\theta^-}}{dt}\right), \tag{12}$$

where $r = \min(1, \mathrm{iter}/H)$ is the tangent warmup ramp and dF_{θ^-}/dt is computed via JVP (Section 4.2).

4.2 Computing the tangent

(a) JVP input tangents. The naive tangent $(dx_t/dt,\, 1)$ overflows in FP16 near $t \to 0$ and $t \to \pi/2$, so sCM rescales it by $\cos(t)\sin(t)$. We feed the network the unscaled x_t and let it apply its own $c_{\mathrm{in}} = 1/\sigma_d$, with tangents

$$v_x = \cos(t)\sin(t)\, \frac{dx_t}{dt}, \qquad v_t = \cos(t)\sin(t) \tag{13}$$

at primals $(x_t,\, t)$. The network's internal $1/\sigma_d$ supplies the factor that sCM writes explicitly into v_t, so the forward-mode JVP returns $\cos(t)\sin(t)\, dF_{\theta^-}/dt$ directly.

(b) Tangent normalization. We normalize the tangent exactly as in sCM's Algorithm 1, $g \to g/(\|g\|_2 + c)$ with $c = 0.1$, where $\|g\|_2$ is the Euclidean norm over each sample's $C \times H \times W$ entries. We do not rescale it by $\sqrt{C\,H\,W}$; the pixel normalization inside ADN (Section 3.4) is a separate operation.

(c) In-place weight renormalization and the JVP. EDM2's magnitude-preserving convolutions write the normalized weight back into the parameter slot on each training-mode forward pass. The forward output is unaffected, since the normalization is also applied functionally to the local tensor used in the convolution, but the side effect violates the functional-purity contract that forward-mode autodiff operates under. We therefore suspend the in-place write for the duration of the JVP call. As a consequence, the stored parameter is not pinned to the unit sphere during student training. It drifts under the optimizer while the forward remains magnitude-preserving via the functional normalization. We have not observed this drift to destabilize training in our runs. For the attention layers, we follow the EDM2 release, which notes that an optimized fused scaled-dot-product implementation had not been verified to produce identical results, and use a plain einsum implementation. The JVP rule for the optimized path in Lu & Song (2025)'s appendix is therefore not used in our setup.

Table 1: **TrigFlow teacher FID, 50k samples, standard ADM evaluation.** ImageNet-64 is class-conditional with the EDM2-S backbone. CIFAR-10 is unconditional with the DDPM++ backbone. Sampler is DPM-Solver-2S with Heun's intermediate step (NFE=63 for ImageNet-64, NFE=35 for CIFAR-10)

Dataset	Method	NFE	FID
ImageNet-64	EDM2-S (Karras et al., 2024)	63	1.58
ImageNet-64	TrigFlow teacher (Lu & Song, 2025)	63	1.70
ImageNet-64	TrigFlow teacher (ours)	63	**1.83**
CIFAR-10	TrigFlow teacher (Lu & Song, 2025)	35	2.15
CIFAR-10	TrigFlow teacher (ours)	35	**2.08**

4.3 sCD vs. sCT

The two variants share the loss in equation 11 and differ only in where the tangent dx_t/dt comes from, which we gave in Section 4.1. sCD reads it from the frozen teacher and pays for a second forward pass each step, but needs no teacher once trained. sCT uses the ground-truth flow tangent and trains with a single pass.

We initialize the student from the teacher's weights in both cases. This is easy to overlook for sCT, which is usually described as teacher-free. In sCM that label applies only to the loss. Algorithm 1 sets $\theta \leftarrow \theta_{\text{pretrain}}$, and the appendix says so directly, so the weights themselves still come from the diffusion teacher. That is the real difference from the train-from-scratch consistency training of Song & Dhariwal (2024a).

5 Reproduction Results

5.1 Teacher

Table 1 reports the TrigFlow teacher FIDs, and the two datasets part ways already at this stage. On CIFAR-10 our teacher reaches 2.08, slightly better than the paper's 2.15. On ImageNet-64 it reaches 1.83 against the reported 1.70. We followed the appendix schedule in full and swept the post-hoc phEMA σ_{rel}, settling on 0.115 rather than the prescribed 0.075, which narrowed the ImageNet gap without closing it. The likeliest cause is the adaptive-weighting loss of Section 3.3. sCM's explicit $1/D$ factor on $\|r\|^2$ equation 9 is unstable in our FP16 setup, so we keep the EDM2 sum convention equation 8, which changes the per-step gradient on the weighting head. The teacher also has to be retrained from scratch under TrigFlow and identity time conditioning, since a stock EDM2-S checkpoint cannot be reused. The 0.13 shortfall is small on its own, but it sets a floor that the distilled students inherit, a point we return to in Section 8.1.

5.2 sCD and sCT results

Tables 2 and 3 report our sCD and sCT results on ImageNet-64 and CIFAR-10. CIFAR-10 is a clean reproduction. sCD reaches 3.59/2.39 at one and two steps, slightly ahead of the paper's 3.66/2.52, and sCT lands at 2.88/2.09 against its 2.85/2.06. Both students essentially match the published numbers, which confirms that our training loop and 50k-sample FID protocol are faithful wherever the recipe is fully specified. ImageNet-64 is the harder case. Despite training for the full prescribed schedule, our sCD-S trails the reported size-S numbers by about 0.6 FID at both step counts (3.59 vs. 2.97 at NFE=1, and 2.66 vs. 2.07 at NFE=2), and sCT-S trails by more. We take up this asymmetry in Section 8.3.

6 Extension: Continuous-Time Multistep sCD (MS-sCD)

6.1 Motivation

sCM's two-step sampling uses a fixed intermediate time $t_{\text{mid}} = 1.1$, whose optimal value varies with model size and training duration. More broadly, sCM's multi-step generation applies a single-step CM repeatedly, whereas MSCD trains a segment-conditioned network and runs one step per segment (Figure 1). The sCM

Table 2: **sCD and sCT results on ImageNet-64 (EDM2-S), without classifier-free guidance.**

Method	Backbone	NFE=1 FID	NFE=2 FID	Framework
iCT (Song & Dhariwal, 2024b)	ADM-style	3.25	2.83	discrete, EDM
ECT (Geng et al., 2025)	ADM-style	2.73	2.04	discrete, EDM
MSCD[†] (Heek et al., 2024)	UViT-small	3.2	1.9	discrete
sCM sCD-S[‡] (Lu & Song, 2025)	EDM2-S/TrigFlow	2.97	2.07	TrigFlow
sCM sCT-S[‡] (Lu & Song, 2025)	EDM2-S/TrigFlow	3.23	2.93	TrigFlow
Ours sCD	EDM2-S/TrigFlow	3.59	2.66	TrigFlow
Ours sCT	EDM2-S/TrigFlow	4.34	3.98	TrigFlow

[†] Heek et al.'s MSCD uses a UViT backbone (transformer UNet with 16×1024-channel mid-blocks at the middle stage), structurally different from the magnitude-preserving EDM2 UNet we use. That model is also substantially larger in parameter count than the EDM2-S/TrigFlow models compared here, so its FID values are given for context rather than as an architecture-matched baseline. [‡] Numbers from sCM Table 12 (appendix), at the EDM2-S size matched to our setup. The main-paper sCM table reports the XL/XXL variants.

Table 3: **sCD and sCT results on unconditional CIFAR-10, DDPM++ backbone, 50k samples.** sCM paper numbers from Lu & Song (2025), Table 1.

Method	NFE=1 FID	NFE=2 FID	Framework
iCT (Song & Dhariwal, 2024b)	2.83	2.46	discrete, EDM
ECT (Geng et al., 2025)	3.60	2.11	discrete, EDM
CD-LPIPS (Song & Dhariwal, 2024a)	3.55	2.93	discrete, EDM
sCM sCD (Lu & Song, 2025)	3.66	2.52	TrigFlow
sCM sCT (Lu & Song, 2025)	2.85	2.06	TrigFlow
Ours sCD	**3.59**	**2.39**	TrigFlow
Ours sCT	2.88	2.09	TrigFlow

authors list "incorporating our proposed techniques into multi-step consistency models" as future work, and Heek et al. (2024) note that continuous-time training of an MSCD-style model is possible if the network takes an explicit conditioning signal identifying the current segment, so that the prediction can change discontinuously across segment boundaries.

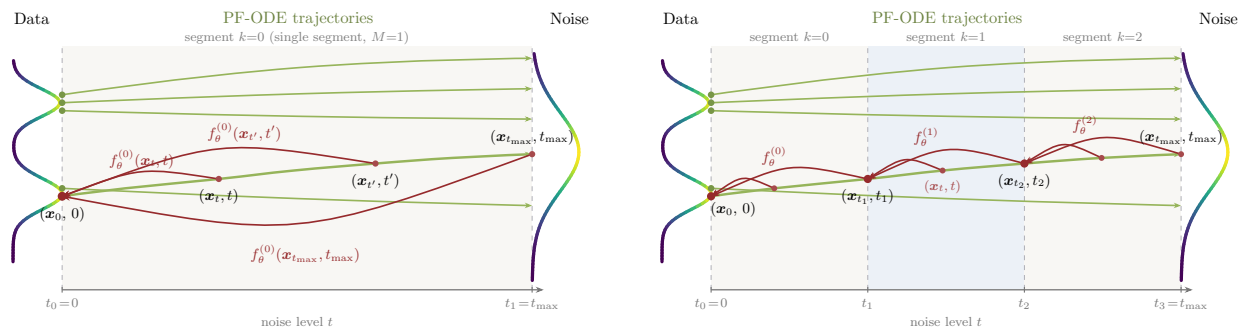
MS-sCD combines these two ideas. We add a segment-boundary input t_k to the sCM network through a zero-initialized embedding, so the model reduces to sCM at $M = 1$ and becomes a segment-conditioned multistep CM otherwise. Three things change as a result: the segment CM parameterization (Section 6.2), its JVP target (Section 6.3), and the training procedure (Section 6.4).

6.2 Segment-conditioned TrigFlow CM

We discretize $[0, t_{\max}]$ into M segments with boundaries $0 = t_0 < t_1 < \cdots < t_M = t_{\max}$, $t_{\max} = \arctan(\sigma_{\max}/\sigma_d)$. Segment k covers $(t_k, t_{k+1}]$. For a point x_t in segment k, define $\Delta t := t - t_k$. The *segment CM* is:

$$f_\theta^{(k)}(x_t, t) := \cos(\Delta t)\, x_t - \sin(\Delta t)\, \sigma_d\, F_\theta\left(\frac{x_t}{\sigma_d}, t, t_k\right). \tag{14}$$

This is a TrigFlow-DDIM step from t to t_k (not to 0). It satisfies the *segment boundary condition*: $f_\theta^{(k)}(x_{t_k}, t_k) = x_{t_k}$ because $\sin(0) = 0$, so the CM is the identity at the bottom of each segment. Figure 1 illustrates the parameterization.



(a) $M = 1$ (sCM): a single segment covers $[0, t_{\max}]$. $f_\theta^{(0)}$ maps any point on a PF-ODE trajectory to the data origin $(x_0, 0)$.

(b) $M = 3$ (MS-sCD): three segments. $f_\theta^{(k)}$ maps any point (x_t, t) inside segment k to that segment's lower boundary (x_{t_k}, t_k).

Figure 1: Green arrows are PF-ODE trajectories (data $\leftrightarrow$ noise) and red arrows show the consistency map. **(a)** For $M = 1$ the segment CM equation 14 reduces to sCM: $f_\theta^{(0)}$ collapses every trajectory point to the data origin $(x_0, 0)$. **(b)** For $M > 1$, MS-sCD splits the time axis at $0 = t_0 < t_1 < \cdots < t_M = t_{\max}$ and trains one segment-conditioned map $f_\theta^{(k)}$ per segment k. Each $f_\theta^{(k)}$ collapses trajectory points inside segment k to the segment's lower boundary (x_{t_k}, t_k). The boundary points themselves are fixed: $f_\theta^{(k)}(x_{t_k}, t_k) = x_{t_k}$.

The network F_θ takes t_k as an additional conditioning input, implemented as:

$$\text{emb} \;\; += \;\; \text{linear}_{\text{init}=0}(\text{PositionalEmbed}(t_k))\,, \tag{15}$$

with a zero-initialized linear projection.

6.3 MS-sCD loss

We apply Song's Remark 10 (Song & Dhariwal, 2024a) to the segment CM. All algebra is identical to sCM, with Δt replacing t everywhere. Since t_k is constant along any PF-ODE trajectory (it is a conditioning input, not a trajectory variable), it does not enter the JVP tangent.

The total time derivative of $f_{\theta^-}^{(k)}$ along the ODE is:

$$\frac{df_{\theta^-}^{(k)}}{dt} = \cos(\Delta t)\left(\frac{dx_t}{dt} - \sigma_d\, F_{\theta^-}\right) - \sin(\Delta t)\left(x_t + \sigma_d\, \frac{dF_{\theta^-}}{dt}\right). \tag{16}$$

Note $\partial \Delta t / \partial t = 1$ (since t_k is constant), so Δt plays the same role as t in the sCM derivation.

After the inner-product-to-MSE conversion and prior weighting $\lambda(t, k) = 1/(\sigma_d \tan \Delta t)$:

$$\mathcal{L}_{\text{MS-sCD}}(\theta, \phi) = \mathbb{E}\left[\frac{1}{D\, e^{w_\phi(t)}}\, \|F_\theta - F_{\theta^-} - \tilde{g}_k\|_2^2 + w_\phi(t)\right], \tag{17}$$

where the raw target vector $\tilde{g}_k$ replaces t with Δt in equation 12:

$$\tilde{g}_k = -\cos^2(\Delta t)\left(\sigma_d\, F_{\theta^-} - \frac{dx_t}{dt}\right) - r \cdot \cos(\Delta t)\sin(\Delta t)\left(x_t + \sigma_d\, \frac{dF_{\theta^-}}{dt}\right). \tag{18}$$

The JVP tangent vectors carry over with Δt replacing t:

$$v_x = \cos(\Delta t)\sin(\Delta t) \cdot dx_t/dt, \qquad v_t = \cos(\Delta t)\sin(\Delta t), \tag{19}$$

and t_k is passed as a primal input with zero tangent (since $dt_k/dt = 0$ along the PF-ODE).

Algorithm 1 Continuous-time Multistep sCD (MS-sCD) training, sCD variant.

Input: data $\mathcal{D}$ (std. σ_d), teacher F_{pretrain}, student F_θ (init. from F_{pretrain}, t_k-embedding init. to zero), weighting w_ϕ, segments $0 = t_0 < \cdots < t_M = t_{\max}$, proposal $(P_{\text{mean}}, P_{\text{std}})$, norm const $c = 0.1$, warmup H, lr η.
Init: $\theta \leftarrow \theta_{\text{pretrain}}$, iter $\leftarrow 0$.
repeat
  $x_0 \sim \mathcal{D},\ \ z \sim \mathcal{N}(0, \sigma_d^2 I)$
  $\tau \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2),\ \ t \leftarrow \arctan(e^\tau/\sigma_d)$
  find k s.t. $t \in (t_k, t_{k+1}]$; $\ \Delta t \leftarrow t - t_k$ {segment lookup}
  $x_t \leftarrow \cos(t)\, x_0 + \sin(t)\, z$
  $dx_t/dt \leftarrow \sigma_d\, F_{\text{pretrain}}(x_t/\sigma_d,\, t)$ {sCT: $\cos(t)z - \sin(t)x_0$}
  $r \leftarrow \min(1, \text{iter}/H)$ {tangent warmup}
  $Y \leftarrow \text{JVP}\big[F_{\theta^-}(\cdot,\cdot,t_k)\big]$ at $(x_t,\, t)$ with tangent $\big(\cos\Delta t \sin\Delta t\,(dx_t/dt),\ \cos\Delta t \sin\Delta t\big)$ {network applies $c_{\text{in}}{=}1/\sigma_d$; $\Delta t = t - t_k$}
  $\tilde{g} \leftarrow -\cos^2\Delta t\,(\sigma_d F_{\theta^-} - dx_t/dt) - r\,(\cos\Delta t \sin\Delta t\, x_t + \sigma_d\, Y)$
  $\tilde{g} \leftarrow \tilde{g}/(\|\tilde{g}\|_2 + c)$ {L2 tangent norm}
  $\mathcal{L} \leftarrow \frac{1}{D\, e^{w_\phi(t)}} \big\|F_\theta(x_t/\sigma_d, t, t_k) - F_{\theta^-} - \tilde{g}\big\|_2^2 + w_\phi(t)$
  $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{\theta,\phi} \mathcal{L};\ \ \text{iter} \leftarrow \text{iter} + 1$
until convergence

Algorithm 2 MS-sCD sampling (M steps, M NFE)

Draw $x_{t_M} \sim \mathcal{N}(0, \sigma_d^2 I)$
for $k = M-1, M-2, \ldots, 0$ **do**
  $\Delta_k \leftarrow t_{k+1} - t_k$
  $x_{t_k} \leftarrow \cos(\Delta_k)\, x_{t_{k+1}} - \sin(\Delta_k)\, \sigma_d\, F_\theta\big(x_{t_{k+1}}/\sigma_d,\ t_{k+1},\ t_k\big)$
end for
return x_{t_0} [approximation to x_0]

6.4 Training algorithm

Algorithm 1 is sCM's Algorithm with three changes. Namely, a segment lookup that sets t_k from the segment t falls into, the substitution $\Delta t = t - t_k$ in the tangent, and the extra conditioning input t_k passed to the network on both the differentiated and the stop-gradient branches of the loss. At $M = 1$, we set $t_0 = 0$, $t_1 = t_{\max}$, so $\Delta t = t$ for all $t \in (0, t_{\max}]$. The zero-initialized t_k linear produces no output. Then equation 17 reduces exactly to $\mathcal{L}_{\text{sCD}}$ equation 11.

6.5 Sampling

Algorithm 2 gives the M-step sampling procedure. Each step applies the segment CM equation 14 from t_{k+1} to t_k, for a total cost of M network evaluations. At $M = 2$, the NFE budget matches sCM's two-step sampling, with σ_{mid} replaced by the trained segment boundary t_1.

6.6 Results

Table 4 reports MS-sCD FID at $M = 2, 4, 8$ against MSCD and the sCM baseline. At $M{=}2$, MS-sCD outperforms our discrete-time MSCD reproduction by 0.91 FID (2.53 vs. 3.44), consistent with the continuous-time formulation avoiding the discretization error that finite DDIM steps incur in the low-NFE regime. MSCD closes the gap at larger M and slightly surpasses MS-sCD at $M{=}8$ (1.79 vs. 2.08), approaching the teacher FID. Figure 2 shows FID vs. NFE, with MS-sCD steadily decreasing toward the teacher FID as M grows.

Table 4: **MS-sCD vs. baselines on ImageNet-64 (EDM2-S, class-conditional, no CFG, 50k samples).** sCM-S baselines are from Lu & Song (2025), appendix Table 12, where the main-paper sCM table reports the XL/XXL variants. The MSCD baselines on the EDM2-S backbone are *our* reproductions of Heek et al. (2024)'s method on this architecture (not Heek et al.'s reported numbers, which use a larger backbone, see Section 7)

Method	M	NFE	FID
sCM sCD-S (Lu & Song, 2025) (appendix Table 12)	1	1	2.97
sCM sCD-S (Lu & Song, 2025) (appendix, 2-step)	1	2	2.07
MSCD on EDM2-S (ours)	2	2	3.44
MSCD on EDM2-S (ours)	4	4	2.32
MSCD on EDM2-S (ours)	8	8	1.79
MS-sCD (ours)	2	2	2.53
MS-sCD (ours)	4	4	2.26
MS-sCD (ours)	8	8	2.08
TrigFlow teacher (ours)	—	63	1.83

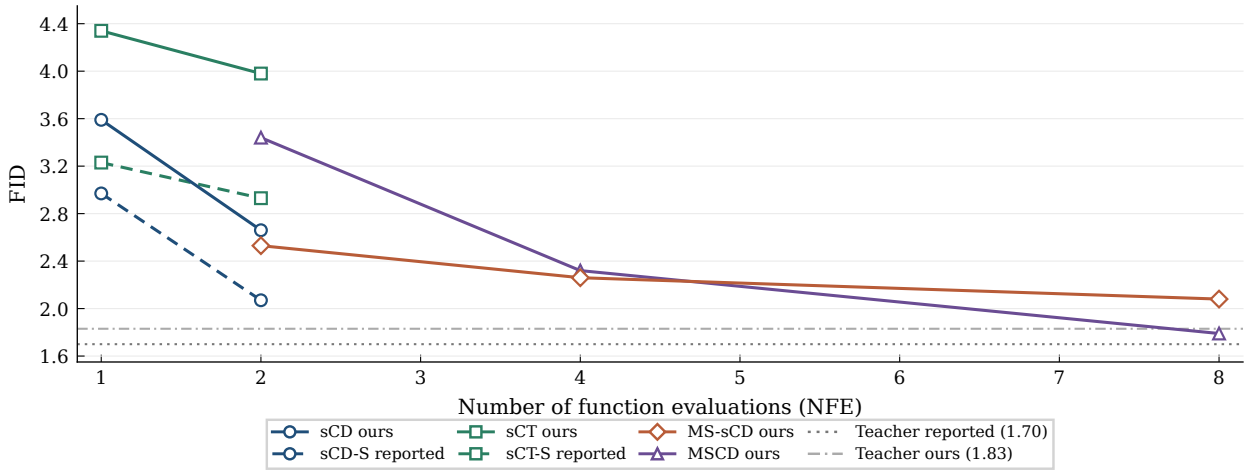


Figure 2: FID vs. NFE on ImageNet-64 (EDM2-S, class-conditional, no CFG, 50k samples). Solid lines: our EDM2-S/TrigFlow reproduction. Dashed lines: sCM's reported sCD-S and sCT-S numbers (Lu & Song, 2025, appendix Table 12). The two multistep students extend past the NFE$\leq$2 regime of the single-step CMs by varying the number of segments $M \in \{2, 4, 8\}$: continuous MS-sCD (orange) leads at M=2, while our discrete MSCD reproduction (purple) overtakes it by M=8, both approaching the teacher reference (horizontal lines: ours 1.83, sCM-reported 1.70).

7 Supporting Context: MSCD and Moment Matching

Before training the TrigFlow models we reproduced two existing distillation methods on class-conditional ImageNet-64, using the EDM framework (Karras et al., 2022) with the ADM-style UNet of Dhariwal & Nichol (2021), to validate our 50k-sample FID pipeline against published numbers. Our Multistep Consistency Distillation (MSCD, Heek et al. 2024) student reaches FID $\approx$2.3 at 8 steps, matching the deterministic Heun teacher at 511 NFE. This is the same method we later port to the EDM2-S backbone as the discrete-time baseline for MS-sCD (Table 4). Because MSCD trains the student to follow the deterministic teacher ODE within each segment, its attainable FID is bounded by that sampler. We separately reproduced moment matching (Salimans et al., 2024), a distribution-matching distillation, on the same backbone, reaching FID $\approx$1.4 with an 8-step student. We do not pursue this family further. The contrast is worth noting all the

same. On images, distribution-matching distillation reaches much lower few-step FID than the ODE-bounded consistency objectives we focus on.

8 Discussion

8.1 The teacher gap propagates to the distilled students

Our TrigFlow teacher reaches FID 1.83 on ImageNet-64 against the reported 1.70 (Table 1). Since sCD distills the teacher ODE, the student's attainable FID is bounded by the teacher's, so part of our sCD-S shortfall is inherited rather than a distillation failure. The sCT-S gap, which does not distill the teacher, must have a separate cause (Section 8.2). The teacher deficit also reframes the multistep route: a one-step student must absorb the entire teacher-to-data map in a single evaluation, whereas MSCD and MS-sCD divide it across M segments and lower the per-step burden, tracking the teacher more closely at the same total compute. We therefore expect that, with the residual teacher gap closed, the multistep budget–quality trade-off is the more dependable use of a fixed sampling budget at small model scale. Consistent with this, our MSCD reproduction also appears less sensitive to the undisclosed architectural choices of Section 8.3, as its per-segment targets are supervised by explicit teacher steps rather than a continuous-time tangent.

8.2 sCT vs. sCD ordering depends on model capacity

Lu & Song (2025) state in their Conclusion that "sCT consistently outperforms sCD on CIFAR-10 and ImageNet 64×64 but is less effective than sCD across different model scales on ImageNet 512×512," attributing the ImageNet-512 exception to the training variance of CT in latent spaces with imperfect encoders. The same paper's appendix table at ImageNet-64 (sCM Table 12) reports sCT-S at FID 3.23/2.93 (1-/2-step) versus sCD-S at 2.97/2.07, so sCT is worse than sCD at the smallest model size by ≈0.26 at NFE=1 and ≈0.86 at NFE=2. From size M upward the ordering reverses (sCT-M 2.25/1.81 vs. sCD-M 2.79/1.89), and sCT wins through XL as the paper's claim suggests. The blanket statement therefore holds only at the larger scales. At EDM2-S the ordering is reversed even in a pixel-space setting where the latent-space hypothesis does not apply. The sCT/sCD ordering is therefore capacity-dependent, not purely dataset-dependent.

8.3 Backbone-dependent reproduction quality on CIFAR-10 versus ImageNet-64

On CIFAR-10 our sCD matches and slightly exceeds the reported FID at both step counts. On ImageNet-64, despite following the full prescribed schedule, sCD-S lags by ≈0.6 FID and sCT-S by more. The gap is therefore backbone-conditional rather than caused by under-training, and it concentrates where sCM's prescription is inherited from the EDM2 code base or left undisclosed. Two such choices arise only on ImageNet-64.

Adaptive-weighting head $w_\phi(t)$. sCM specifies the weighting objective but not the architecture of w_ϕ, only that it follows EDM2 (a Fourier embedding into a magnitude-preserving linear map). We use a positional embedding of t instead, matching sCM's documented Fourier-to-positional substitution for the main network, which the paper does not extend explicitly to the weighting head.

Attention implementation. sCM states that the ImageNet-64 students use FlashAttention with a custom JVP rule, but the released EDM2 backbone uses an einsum attention by default (its notes flag the fused kernel as insufficiently tested). We follow that default. The FlashAttention path and its JVP rule cannot be reconstructed from the available artifacts, and the FP16 reduction-order difference could accumulate over the 400k-iteration student schedule. That two architectural details are already missing from the prescription also suggests there may be others we cannot see in the released artifacts, any of which could contribute to the gap.

9 Conclusion and Future Work

We set out to reproduce the TrigFlow/sCM formulation and to see how far it extends. The formulation itself holds up. Each of the six stabilization fixes maps to an identifiable term in the instability decomposition

equation 6, and on CIFAR-10 we recover the reported sCD FID at both step counts and slightly improve on it. ImageNet-64 is where we fall short. Our teacher never reaches the reported 1.70, and two architectural choices are left unspecified at that backbone, so a faithful re-implementation settles at a different FID. We do not read this as evidence against the method. It shows how much of the recipe stays implicit once the backbone grows.

That teacher gap shapes what we take away. A student that matches the teacher's trajectory, as sCD does, cannot do better than that teacher, and a one-step student has to absorb the whole teacher-to-data map at once. Spreading that map over several segments uses a fixed sampling budget more reliably at small model scale, which is exactly what MS-sCD does. This trajectory bound is not universal. Distribution-matching distillation such as moment matching (Salimans et al., 2024) can surpass its teacher at low NFE, because the teacher's own per-timestep prediction errors leave room to improve on it. We therefore view MS-sCD, our continuous-time multistep generalization of sCD, as the part of this work most likely to carry over to other settings. The formulation, training scripts, and all teacher and student checkpoints accompany this submission.

References

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

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

Zhengyang Geng, Ashwini Pokle, Weijian Luo, Justin Lin, and Zico Kolter. Consistency models made easy. In *International Conference on Learning Representations*, volume 2025, pp. 75158–75185, 2025.

Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

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

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

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

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the training dynamics of diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *International Conference on Learning Representations*, volume 2025, pp. 50611–50649, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. *Advances in Neural Information Processing Systems*, 37:36046–36070, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, volume 2024, pp. 15078–15097, 2024a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, volume 2024, pp. 15078–15097, 2024b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *Advances in Neural Information Processing Systems*, 2023.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

A Derivations

A.1 CM parameterization as a TrigFlow-DDIM step

Holding the network output F_θ constant over $[s, t]$ and integrating the TrigFlow PF-ODE gives the TrigFlow analog of a DDIM step (Song et al., 2021; Lu et al., 2025),

$$x_s = \cos(t-s)\, x_t - \sin(t-s)\, \sigma_d\, F_\theta(x_t/\sigma_d, t), \tag{20}$$

the first-order member of the DPM-Solver family. Setting $s = 0$ recovers the CM parameterization equation 3, $f_\theta(x_t, t) = \cos(t)\, x_t - \sin(t)\, \sigma_d\, F_\theta(x_t/\sigma_d, t)$, so the CM is exactly the single-step jump from (x_t, t) to the data origin.

A.2 ADN normalization

We restate EDM2's `normalize` function and derive its properties.

Function:

$$\begin{aligned} \mathrm{norm}(x, \mathrm{dim}) &= x \,/\, (N(x) + \varepsilon), \\ N(x) &:= \|x\|_{\ell_2,\,\mathrm{dim}} \cdot \sqrt{\frac{|\mathrm{norm}(x)|}{|x|}}, \quad \varepsilon = 10^{-4}, \end{aligned} \tag{21}$$

where $\|\cdot\|_{\ell_2,\mathrm{dim}}$ denotes the ℓ_2 norm along the specified axes (with `keepdim=True`), $|\mathrm{norm}(x)|$ is the number of elements in the norm output tensor, and $|x|$ is the total number of elements in x.

Applied along `dim=1` to shape $[B, C, H, W]$: The norm output has shape $[B, 1, H, W]$, so $|\mathrm{norm}| = B{\cdot}H{\cdot}W$ and $|x| = B{\cdot}C{\cdot}H{\cdot}W$, giving $\sqrt{|\mathrm{norm}|/|x|} = 1/\sqrt{C}$. Therefore $N(x) = \|v\|_2/\sqrt{C} + \varepsilon = \mathrm{RMS}(v) + \varepsilon$ (where $v = x_{b,:,h,w}$ is the channel vector at spatial location (b, h, w)).

The $1/\sqrt{C}$ factor. Pixel normalization is RMS normalization by construction, so the scale and shift entries stay $O(1)$ regardless of C. The reproducibility trap is to write a plain ℓ_2 normalization instead (dropping

the $1/\sqrt{C}$): the entries then shrink to $O(1/\sqrt{C})$ and the affine modulation vanishes at large C. sCM cites Karras et al. (2018) for pnorm but does not fix ε or its placement. We use EDM2's `normalize` ($\varepsilon = 10^{-4}$, added outside the norm) for the EDM2 backbone and the original PGGAN form ($\varepsilon = 10^{-8}$, inside the square root) for DDPM++.

ADN in the residual block:

$$[s_{\text{pre}}, b_{\text{pre}}] = \text{split}(\texttt{emb_linear}(\text{emb})), \tag{22}$$

$$s = \text{norm}(s_{\text{pre}}, \text{dim} = 1), \quad b = \text{norm}(b_{\text{pre}}, \text{dim} = 1), \tag{23}$$

$$y = \text{norm}_{\text{group}}(y) \odot s[\ldots] + b[\ldots], \tag{24}$$

where $[\ldots]$ denotes broadcasting over the H, W dimensions.

B Forward-Mode Autodiff of Magnitude-Preserving Convolutions

The magnitude-preserving convolutions used in the EDM2 backbone normalize their weight tensors in two complementary ways on every training-mode forward pass. The output is computed using the normalized weight (a pure functional operation), and the underlying parameter slot is also overwritten with the normalized value as a side effect. The first keeps the convolution's output magnitude-preserving. The second keeps the stored parameter itself on the unit sphere across optimizer steps.

The functional purity required by forward-mode autodiff is at odds with the side effect. PyTorch documents in-place writes to leaf tensors inside transformed functions as undefined behavior. We therefore disable the in-place write for the duration of any JVP call. The functional normalization is unaffected, so the conv output during the JVP is bit-for-bit the same as it would be otherwise.

This has a cost. Every student step is a JVP call with the in-place write suppressed, so the stored parameter is no longer pinned to the unit sphere and only the optimizer updates it. The parameter norm can then drift over training, while the conv output stays magnitude-preserving through the functional normalization. We rely on the optimizer's adaptive scaling to handle the drift. We have not isolated training instabilities attributable to it in our testing.

C Dropout in Multistep Consistency Models

A discrete-time consistency-distillation loss evaluates the student at two inputs each step, in two separate forward passes. The first, (x_t, t), gives the prediction whose gradient we take. The second, (x_s, s), is the teacher-stepped point that supplies the stop-gradient target. When the network contains dropout layers, the two forward passes sample independent random masks, and the squared error between the two predictions mixes the consistency residual. The fix, introduced for consistency models by Song & Dhariwal (2024b), is to share dropout masks across the two passes. We record the random state immediately before the first pass and restore it before the second. With the same state, the same masks are sampled, so any difference between the two predictions reflects only the change in inputs and not the change in dropout pattern.

In our MSCD reproductions on the EDM (Karras et al., 2022) and EDM2 (Karras et al., 2024) backbones we see that without dropout the segment-conditioned student overfits within the training budget. This matches the observation of Song & Dhariwal (2024b) that the dropout rate is among the hyperparameters most affecting the convergence of consistency training. Once dropout is used, mask synchronization is required for the consistency loss to remain an unbiased estimator of the prediction discrepancy.

sCM sidesteps the issue for its single-step students by prescribing zero dropout for sCD on CIFAR-10 and ImageNet-64 (Lu & Song, 2025, Appendix C), and our sCD runs follow that. Synchronization then only matters for the discrete MSCD reproductions, which take two separate forward passes per step. Continuous MS-sCD computes the prediction and its time-derivative in a single dual-number JVP forward, so both share the same dropout mask automatically and need no synchronization.

The dropout rate matters as much as the synchronization. We tried a few ablations for MS-sCD. With no dropout the segment-conditioned student overfits and its FID stalls around 2.3 at eight steps. At 0.2 it trains

too slowly and shows signs of instability. A rate of 0.1, applied only at resolutions ≤ 16, worked best, and it is probably why our two-step FID (2.53) beats the discrete MSCD baseline. We use 0.1 for all reported MS-sCD numbers.

D Live Weights versus phEMA for MS-sCD

For the teacher and the single-step students we evaluate with post-hoc EMA (phEMA), sweeping σ_{rel} and keeping the best by validation FID. For MS-sCD we also try the live (non-EMA) weights. We save them every 4 Mi (≈4.2M) training images, the same cadence as our moment-matching runs, and evaluate each save. We pick between the live weights and phEMA by the same validation FID we use for the σ_{rel} sweep. The live weights won on some MS-sCD runs and phEMA on others, so we report whichever was better for each. We did not see a pattern that says in advance which one to use.

E Noise Sampling for the Multistep Students

Heek et al. (2024) split diffusion time into equal segments and sample the training noise uniformly over a discrete, cosine-parameterized step grid that anneals from 64 to 1280 steps. We do not follow that recipe. Instead we sample noise the way each teacher was trained. For the discrete MSCD on EDM2-S we use the teacher's EDM (Karras) σ grid, annealing the step count from 64 to 1280, not a cosine grid. For MS-sCD we keep the segment boundaries uniform in TrigFlow time but draw the noise level t from the teacher's own log-normal distribution ($P_{\text{mean}} = -1.0$, $P_{\text{std}} = 1.6$) instead of spreading samples evenly across segments. Matching the teacher's noise distribution worked better in our runs. Spreading samples evenly across segments puts too much weight on noise levels the teacher rarely saw in training.

For the two-step (M=2) discrete MSCD on EDM2 we set the interior knot of the student grid to $\sigma = 1.1$, giving $[\sigma_{\max}, 1.1, 0]$, since the default Karras schedule places it near $\sigma_{\min} = 0.002$ and collapses the first segment. The M=4 runs keep the default Karras grid.

F Experimental Details

F.1 Teacher training

Both teachers are trained with the TrigFlow diffusion loss equation 2 using log-normal noise-level sampling $t = \arctan(e^{\tau}/\sigma_d)$, $\tau \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, $\sigma_d = 0.5$.

ImageNet-64 (EDM2-S backbone). $P_{\text{mean}} = -0.8$, $P_{\text{std}} = 1.6$. Optimizer: Adam, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\varepsilon = 10^{-11}$, reference learning rate $\eta_{\text{ref}} = 0.01$ (EDM2-style auto-scaling via reference batch count 35,000). Batch size 2048 on 32 H100 GPUs. We use the post-hoc power-function EMA framework of EDM2 (phEMA). The training run records phEMA snapshots, and at evaluation we synthesize the EMA at a chosen σ_{rel} (Karras et al., 2024). The sCM paper's appendix uses $\sigma_{\text{rel}} = 0.075$. In our run we swept σ_{rel} and found the optimal value to be σ_{rel}=0.115, which we use for the reported FID. The rest of the training recipe matches the sCM appendix.

CIFAR-10 (DDPM++ backbone, unconditional). We follow sCM's Appendix C.2 settings, which themselves follow the EDM 2022 CIFAR-10 recipe (Karras et al., 2022): $P_{\text{mean}} = -1.2$, $P_{\text{std}} = 1.2$, dropout 0.13, batch size 512, learning rate 10^{-3}, Adam $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$, no learning-rate decay, 400k training iterations. Backbone: Score SDE DDPM++ (Song et al., 2020) with channel multiplier $[2, 2, 2]$, four residual blocks per resolution, and attention at 16×16.

For CIFAR-10 we use EDM's augmentation pipeline (horizontal flip, isotropic scale, rotation, anisotropic scale, fractional translation, all at $p = 0.12$) with the augmentation labels passed to the network. ImageNet-64 uses no augmentation.

FID evaluation. All FID values use 50,000 samples with the standard ADM evaluation protocol (Dhariwal & Nichol, 2021), compared against the ILSVRC 2012 training set statistics. Generation is class-conditional

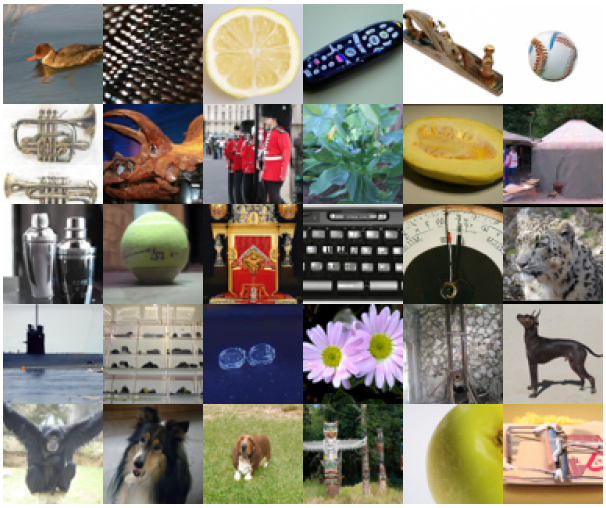
Table 5: Hyperparameters for all experiments.

	Teacher	sCD	sCT	MS-sCD	MSCD-EDM	MM
Framework	TrigFlow	TrigFlow	TrigFlow	TrigFlow	EDM	EDM
Model	EDM2-S	EDM2-S	EDM2-S	EDM2-S	EDM-S	EDM-S
Batch size	2048	2048	2048	2048	2048	2048
GPUs	32	32	32	32	32	16
P_{mean}	-0.8	-1.0	-1.0	-1.0	-0.8	—
P_{std}	1.6	1.6	1.6	1.6	1.6	—
σ_d	0.5	0.5	0.5	0.5	0.5	0.5
Dropout	0	0	0.45	0.10	0.10	0
LR ref	0.01	0.01	0.01	0.01	5×10^{-4}	2×10^{-6}
Adam β	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)	(0.0, 0.99)
Adam ε	10^{-11}	10^{-11}	10^{-11}	10^{-11}	10^{-8}	10^{-12}
phEMA std	0.075	0.075	0.075	—	—	—
Eval. weights	phEMA	phEMA	phEMA	both	EMA	live
Tangent c	—	0.1	0.1	0.1	—	—
Warmup H	—	10k	10k	10k	—	—
Params	293.9M	293.9M	293.9M	293.9M	280.2M	295.9M

for ImageNet-64, with no classifier-free guidance. Following Karras et al. (2024), we compute FID over a sweep of five generator seeds and report the best.

G Uncurated Samples on ImageNet-64

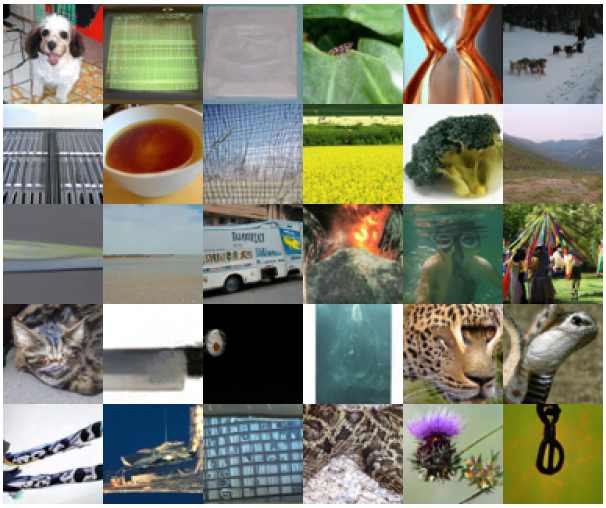
Each panel below shows 30 uncurated class-conditional ImageNet-64×64 samples (arranged as a 6×5 grid) from one of the four models discussed in the paper.



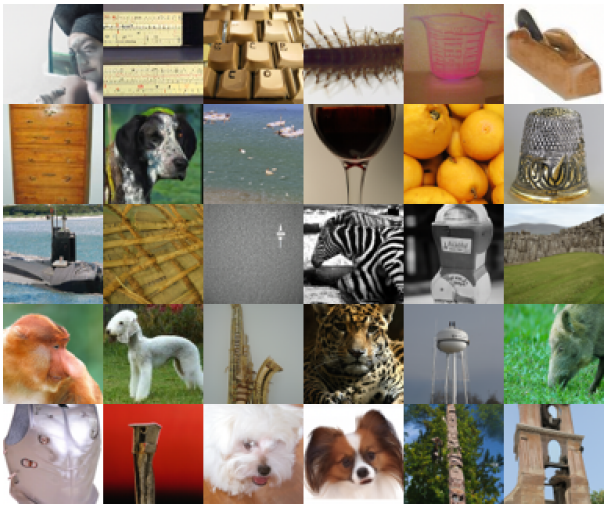
(a) TrigFlow teacher (EDM2-S backbone), 63-NFE DPM-Solver-2S samples.

(b) sCD student (TrigFlow, EDM2-S, ours), 2-NFE samples with intermediate time $t_{\text{mid}} = 1.1$ (sCM Algorithm 2).

Figure 3: Teacher and sCD student samples.



(a) sCT student (TrigFlow, EDM2-S, ours), 2-NFE samples with intermediate time $t_{\text{mid}} = 1.1$ (sCM Algorithm 2).



(b) MSCD on EDM2-S (ours), $M = 8$ (8-NFE) samples.

Figure 4: sCT student and 8-step MSCD samples.