SELF-DRIFT: ONE-STEP GENERATIVE FLOWS LEARN FROM THEIR OWN REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

One-step flow models promise low-latency generation by compressing transport into a single network evaluation, but trajectory supervision alone can leave a mismatch between predicted endpoints and the data distribution. Existing distribution-correction methods commonly obtain their comparison geometry from an external teacher, encoder, or discriminator, so the feedback comes from a model separate from the flow being optimized. We introduce **Self-Drift**, a joint trajectory–distribution training scheme that closes this loop inside the flow itself. Our dual-horizon trajectory formulation pairs local transport learning with long-horizon prediction to a fixed clean endpoint, enabling the flow to compare predicted and real endpoints through its own representation and class-conditional feature memories. A frozen-lens semi-gradient routes a detached attraction–repulsion target through the endpoint, providing distribution feedback across the trajectory while preserving path-wise transport learning. Real-data-anchored, condition-number-capped whitening stabilizes the evolving feature space. Self-Drift learns this correction geometry in the same from-scratch process. The correction is self-contained and training-only: it needs no separately trained distribution-correction model and leaves the original one-step inference path unchanged. Self-Drift achieves a leading FID of **1.36** at NFE=1 among the listed from-scratch diffusion/flow methods on class-conditional ImageNet 256×256.

1 INTRODUCTION

Flow-based generative models learn a transport from a simple prior to the data distribution (Lipman et al., 2023; Liu et al., 2023). Recent objectives compress this transport into a single network evaluation (Geng et al., 2025), enabling high-quality generation with minimal sampling latency. For a one-step flow, the clean endpoint predicted from a randomly sampled trajectory state is where two requirements meet: each prediction should be consistent with its transport path, while the aggregate distribution of predictions should match the data distribution. Trajectory objectives directly supervise the former, but finite model capacity and optimization can leave residual marginal mismatch because they do not explicitly compare the predicted endpoint distribution with the data marginal. The predicted endpoint therefore provides a natural interface for adding distribution-level correction without replacing path-wise learning.

Distribution correction, however, requires a geometry in which real and generated samples can be compared. Existing approaches commonly obtain this geometry from outside the generator: distribution-matching distillation transfers a score field from a pretrained diffusion teacher (Yin et al., 2024), adversarial distillation learns an auxiliary discriminator (Sauer et al., 2024), and drifting-based methods use representations supplied by a pretrained encoder or diffusion teacher (Deng et al., 2026; Zhang et al., 2026). Although effective, these approaches separate the source of distribution feedback from the flow that learns the transport. This raises a basic question: *can the one-step flow itself provide the representation geometry needed to correct its endpoint distribution?*

We answer this question with **Self-Drift**, a joint trajectory–distribution training scheme that keeps both signals inside the same online flow. To enable this feedback throughout the trajectory, we design a dual-horizon formulation whose long-horizon branch reconstructs a clean image latent in one evaluation, while its local branch supplies transport supervision. These endpoints share the input domain of real data, allowing both to be compared through the flow's own feature lens (Figure 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

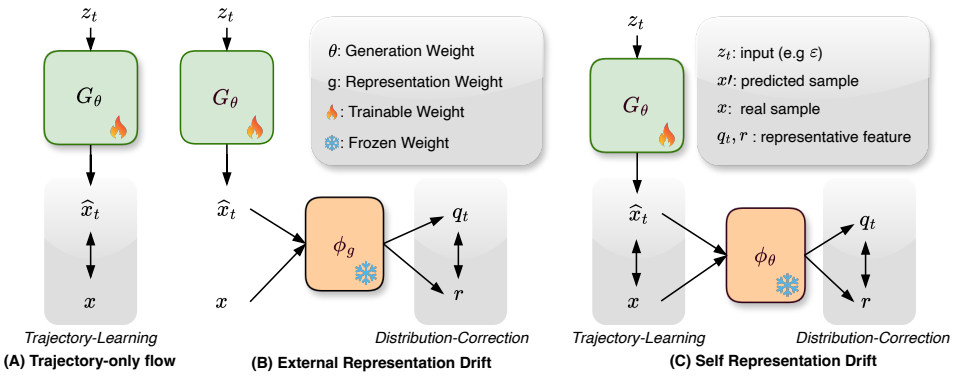


Figure 1: Overview of Self-Drift. (A) Our dual-horizon flow learns clean-endpoint predictions through trajectory supervision. (B) Existing drifting approaches compare generated and real samples in an externally supplied representation space. (C) Self-Drift reuses an intermediate representation of the same online flow to compare its predicted clean endpoints with real data.

Class-conditional memories of real and generated features define a multi-scale, nonparametric attraction–repulsion field, which produces a detached feature target for the endpoint. Regressing this target propagates distribution feedback through the endpoint prediction and updates the same flow that produced it. Thus, Self-Drift turns the flow's own evolving representation into the geometry that corrects its predicted distribution, without introducing a separately trainable drift model.

Sharing the endpoint predictor and feature lens creates an important optimization requirement: the drift objective should move the endpoint rather than reduce itself by directly changing the current feature coordinates. During each feature forward, we therefore treat the representation parameters as a frozen lens while keeping the representation differentiable with respect to its endpoint input. This **frozen-lens semi-gradient** directs the correction through the endpoint to the flow parameters, while the online representation continues to evolve naturally across optimization steps. Because trajectory time is sampled throughout training, the same mechanism performs **trajectory-wide endpoint drifting** rather than correcting only samples generated from the pure-noise boundary. Persistent class-conditional memories and real-data-anchored, condition-number-capped whitening stabilize the resulting co-evolving drift space.

Self-Drift learns both the one-step transport and the representation geometry used to correct it from the same random initialization and target data. Distribution feedback therefore emerges within the same from-scratch learning system, without an external generative teacher, separately pretrained representation encoder for distribution correction, or adversarial discriminator. This makes the correction self-contained and keeps the deployed sampler unchanged: memories, whitening, and drift computation are used only during training, so one-step inference incurs no additional computation. On class-conditional ImageNet 256×256, Self-Drift-XL/2 reaches an FID of **1.36** at NFE=1, a leading result among the listed from-scratch diffusion/flow methods, without a trainable auxiliary network or inference-time overhead.

Our contributions connect trajectory design, distribution feedback, and practical training:

1. We introduce **Self-Drift**, which couples path-wise learning with **trajectory-wide distribution correction** in the same flow's online representation. A frozen-lens semi-gradient routes feedback through predicted endpoints while the representation evolves with the generator.

2. We design a **dual-horizon clean-endpoint flow** for this coupling. Local transport supervision and long-horizon endpoint prediction make clean image latents available for comparison with real data from every sampled trajectory state, without an additional generator rollout.

3. We make this co-evolving drift space practical with class-conditional feature memories and real-data-anchored, condition-number-capped whitening, stabilizing training while keeping the deployed sampler unchanged.

2 Related Work

Trajectory learning. Diffusion and flow models learn transport from a simple prior to data through path supervision (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a;b; Lipman et al., 2023; Liu et al., 2023; Albergo et al., 2023; Song & Dhariwal, 2024; Song et al., 2023; Geng et al., 2024; Lu & Song, 2025; Yang et al., 2024; Kim et al., 2024; Salimans & Ho, 2022; Luo et al., 2023). Fast variants such as MeanFlow, Improved Mean Flows, and Shortcut Models reduce sampling cost by learning average velocities or flow maps (Geng et al., 2025; 2026; Frans et al., 2025). These trajectory objectives constrain endpoint marginals indirectly. Inductive Moment Matching instead explicitly matches intermediate marginal distributions through inductive bootstrapping (Zhou et al., 2025). Noise-unconditional denoising models study generation without explicit noise-level conditioning (Sun et al., 2025).

Generation with internal representations. Self-perceptual loss uses a frozen copy of a diffusion model for paired feature supervision (Lin & Yang, 2024); SRA, LayerSync, and Self-Flow improve generation through internal representation alignment (Jiang et al., 2025; Haghighi et al., 2026; Chefer et al., 2026). Self-Drift instead compares real and generated distributions through predicted endpoints.

Distribution correction. GANs and adversarial flow models impose realism through adversarial feedback (Goodfellow et al., 2020; Brock et al., 2019; Sauer et al., 2022; Lin et al., 2025), while Distribution Matching Distillation transfers a score correction from a pretrained teacher, Adversarial Diffusion Distillation uses a discriminator, and Drifting Model explicitly constructs a kernel-weighted sample drift (Yin et al., 2024; Sauer et al., 2024; Deng et al., 2026; Zhang et al., 2026). W-Flow uses Sinkhorn gradient-flow updates (Han et al., 2026), while Li & Zhu (2026) relate drifting to long–short flow-map composition. Related objectives include Sinkhorn divergences, adversarial kernel learning in MMD-GAN, and attraction–repulsion for representation learning in CACR (Genevay et al., 2018; Li et al., 2017; Zheng et al., 2023). Self-Drift couples clean-endpoint trajectory learning with sample-level distribution correction in the same online flow, using a frozen-lens semi-gradient without an external correction model or a global Sinkhorn/MMD objective. With Gaussian kernels, drifting relates to smoothed score differences (Lai et al., 2026); our estimator shares this intuition with DMD but constructs its field from samples rather than teacher scores.

3 Method

3.1 A Shared Clean Endpoint

Self-Drift couples trajectory learning and distribution correction through a shared clean endpoint. The same online flow predicts endpoints and provides features for comparison with real data.

Let $(x, y) \sim p_{\text{data}}$ be a labeled example, where x is a clean image latent, and let $\epsilon \sim \mathcal{N}(0, I)$. We interpolate between data and noise as

$$z_t = (1-t)x + t\epsilon, \qquad t \in [0,1]. \tag{1}$$

Here $t = 0$ corresponds to data and $t = 1$ to noise. For a sampled state z_t, the flow's long-horizon prediction $u_\theta(z_t, y, \omega)$ defines the clean-endpoint estimate

$$\hat{x}_t = z_t + t\, u_\theta(z_t, y, \omega), \tag{2}$$

where ω is the guidance strength. We optimize the joint trajectory–distribution objective

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{traj}}(\theta) + \lambda_{\text{SD}} \mathcal{L}_{\text{SD}}(\theta). \tag{3}$$

The trajectory term learns the transport that produces $\hat{x}_t$, while Self-Drift corrects its distribution using the flow's own features. The coefficient λ_{SD} weights the distribution correction.

3.2 Dual-Horizon Trajectory Learning

MeanFlow (Geng et al., 2025) and iMF (Geng et al., 2026) predict average velocities over source–destination intervals. To supply clean endpoints for Self-Drift at every sampled state, we design a

dual-horizon formulation: the local task learns transport, while the long-horizon task always targets the data endpoint. A task condition h selects the prediction:

$$v_\theta = G_\theta(z_t; h = 0, \omega, y), \qquad u_\theta = G_\theta(z_t; h = 1, \omega, y). \tag{4}$$

The network receives no explicit source or destination time; t determines the training state, endpoint reconstruction, and losses. We sample $t = \max(\mathrm{sigmoid}(\xi), \delta)$ with $\xi \sim \mathcal{N}(0, 1)$, where $\delta = 10^{-2}$ is the default local interval.

Local supervision. Moving toward data along Eq. (1) decreases t, with displacement $x - \epsilon$ over the full interval. To incorporate classifier-free guidance, let $v_\theta^{\mathrm{uc}} = G_\theta(z_t; h = 0, 0, \emptyset)$ and define

$$\widetilde{v}_t = x - \epsilon + \omega \sin^3(\pi t)\, \mathrm{sg}[v_\theta - v_\theta^{\mathrm{uc}}], \tag{5}$$

where $\mathrm{sg}[\cdot]$ denotes stop-gradient. Both model predictions inside the target are detached, and $\omega = 0$ recovers the unguided target. The local loss is $\ell_{\mathrm{short}} = \|\delta(v_\theta - \widetilde{v}_t)\|_2^2$.

Endpoint consistency. Let $t^- = t - \delta$ and take a local step $\widetilde{z}_{t^-} = z_t + \delta\, \mathrm{sg}[v_\theta]$. We evaluate the long-horizon branch at this state to obtain $u_\theta^- = \mathrm{sg}[G_\theta(\widetilde{z}_{t^-}; h = 1, \omega, y)]$. Matching the direct displacement to the local target step followed by the remaining displacement gives

$$\ell_{\mathrm{long}} = \left\| \delta\, \widetilde{v}_t + t^- u_\theta^- - t\, u_\theta \right\|_2^2. \tag{6}$$

We normalize each per-example loss by its detached magnitude, $\mathcal{A}(\ell) = \ell / \mathrm{sg}[\ell]$, leaving numerically vanishing losses unscaled. The trajectory objective is

$$\mathcal{L}_{\mathrm{traj}} = \mathbb{E}\left[\frac{\delta}{t} \mathcal{A}(\ell_{\mathrm{long}}) + w_{\mathrm{short}}\, \mathcal{A}(\ell_{\mathrm{short}}) \right]. \tag{7}$$

The expectation is over data, noise, times, and guidance strengths. At each update, the weight δ/t compensates for the explicit t/δ factor in the long-horizon semi-gradient with detached targets (Appendix B.2). The local weight w_{short} is 5 by default. Setting $\lambda_{\mathrm{SD}} = 0$ gives the trajectory-only version of our dual-horizon flow.

3.3 Self-Drift in the Flow's Own Representation

Clean endpoints and real data share the same input domain. We compare them using an intermediate feature map $\phi_\theta(\cdot, y) : \mathcal{X} \to \mathbb{R}^d$ of the current online flow. Both inputs use $h = 1$, a fixed guidance strength, and their class label, without explicit time conditioning. For our transformer, features are obtained by averaging image tokens at the selected layer, excluding conditioning tokens. Layer indices refer to the full shared transformer backbone. The raw query feature is

$$\bar{q}_t = \phi_\theta^{\mathrm{fr}}(\hat{x}_t, y), \tag{8}$$

where fr indicates that the feature parameters are held fixed during feature extraction, while derivatives with respect to $\hat{x}_t$ are retained. Real features are computed with the same map without gradients.

Class-conditional memories. For each class, two banks retain up to K detached, unwhitened features of real examples and predicted endpoints, respectively. We apply drift to queries whose class labels have not been dropped and whose two class banks are nonempty. We index these B eligible queries by b, suppressing their sampled trajectory times in the feature notation below. For each query, we uniformly sample M real features $\{\bar{p}_{bi}\}_{i=1}^M$ and N generated features $\{\bar{n}_{bj}\}_{j=1}^N$ from its class banks. The default configuration uses $M = N = 64$ and $K = 128$.

Real-anchored whitening. Intermediate features evolve during training and can be anisotropic. We estimate a shared whitening map from real features across classes. Exponential averages of their mean μ_k and second moment S_k at iteration k, with decay $\beta = 0.99$, define the covariance $\Sigma_k = S_k - \mu_k \mu_k^\mathsf{T}$. Given $\Sigma_k = U\, \mathrm{diag}(\sigma_i) U^\mathsf{T}$, we cap its condition number at κ by flooring small eigenvalues:

$$\widetilde{\sigma}_i = \max\left(\sigma_i, \frac{\sigma_{\max}}{\kappa}\right), \qquad W_k = U\, \mathrm{diag}\left(\widetilde{\sigma}_i^{-1/2}\right) U^\mathsf{T}. \tag{9}$$

We refresh W_k every 100 iterations and use $\kappa = 25$ for XL/2. The query becomes $q_b = W_k(\bar{q}_b - \mu_k)$; applying the same map to sampled references gives p_{bi} and n_{bj}. The banks store raw features, so all references are transformed using the current whitening map. The eigenvalue floor limits amplification of low-variance directions while keeping the correction anchored to real data.

Attraction–repulsion field. For each query, let $d^+_{bi} = \|q_b - p_{bi}\|_2$ and $d^-_{bj} = \|q_b - n_{bj}\|_2$. Set s to the mean of all query–reference distances, floored at ε_s, and define the coordinate scale $a = \max(s/\sqrt{d}, \varepsilon_s)$, with $\varepsilon_s = 10^{-3}$. At each neighborhood scale $R \in \mathcal{R}$, we weight references by distance:

$$\alpha^{+,R}_{bi} = \mathrm{softmax}_i\left(-\frac{d^+_{bi}}{2Rs}\right), \qquad \alpha^{-,R}_{bj} = \mathrm{softmax}_j\left(-\frac{d^-_{bj}}{2Rs}\right). \tag{10}$$

The difference between the real and generated barycenters defines the field

$$f^R_b = \frac{1}{a}\left(\sum_{i=1}^{M} \alpha^{+,R}_{bi} p_{bi} - \sum_{j=1}^{N} \alpha^{-,R}_{bj} n_{bj}\right). \tag{11}$$

The first term attracts the query toward nearby real features; the second repels it from regions occupied by generated references. We normalize each field by its root-mean-square magnitude across queries and coordinates, then sum over scales:

$$c_R = \sqrt{\max\left(\varepsilon_f, \frac{1}{Bd}\sum_{b=1}^{B} \|f^R_b\|_2^2\right)}, \qquad f_b = \sum_{R\in\mathcal{R}} \frac{f^R_b}{c_R}. \tag{12}$$

We use $\varepsilon_f = 10^{-8}$ for numerical stability and $\mathcal{R} = \{0.02, 0.05, 0.2\}$ to cover multiple neighborhood scales, following the practice of Drift Model (Deng et al., 2026).

This has the attraction–repulsion intuition of a kernel particle or MMD gradient update, but it is not the gradient of a standard global MMD scalar: the weights are query-dependent, the positive and negative banks are normalized separately, and the multiscale fields are detached and renormalized. We likewise do not optimize a Sinkhorn or other optimal transport objective; the field is used directly as a sample-level transport target.

Frozen-lens semi-gradient. All reference features, the whitening map, distance weights, and scale normalizers are detached. The drift defines a target in scaled feature coordinates and a regression loss:

$$q^\star_b = \mathrm{sg}[q_b/a + f_b], \tag{13}$$

$$\mathcal{L}_{\mathrm{SD}} = \frac{1}{Bd}\sum_{b=1}^{B} \|q_b/a - q^\star_b\|_2^2. \tag{14}$$

We set $\mathcal{L}_{\mathrm{SD}} = 0$ when no query is eligible. Freezing the feature parameters preserves the gradient path through the predicted endpoint $\hat{x}_{t_b}$ of each query:

$$\nabla_\theta \mathcal{L}_{\mathrm{SD}} = \sum_{b=1}^{B} \left(\frac{\partial \hat{x}_{t_b}}{\partial \theta}\right)^{\mathsf{T}} \left(\frac{\partial q_b}{\partial \hat{x}_{t_b}}\right)^{\mathsf{T}} \nabla_{q_b} \mathcal{L}_{\mathrm{SD}}. \tag{15}$$

The input Jacobian includes the frozen feature map and whitening transform, so Self-Drift updates the flow through endpoint prediction without a direct gradient through the feature parameters. The representation still evolves with the shared flow after each update. Applying the loss at sampled times provides trajectory-wide distribution correction.

3.4 Joint Training and One-Step Inference

Training. Each iteration forms clean-endpoint predictions and evaluates the trajectory loss. We extract query and real features, update the real-feature statistics, and compute Self-Drift against the existing memories. The current features are then detached and inserted into their class banks, and the joint objective in Eq. (3) updates the flow. This order keeps the current query out of its reference set.

We train the flow and its online feature lens together from random initialization. Feature memories and real-feature statistics are accumulated throughout training, and we use a fixed weight $\lambda_{\mathrm{SD}} = 0.1$ on drift-active iterations. The training schedule is detailed in Appendix C.1; the correction requires no separately trained model.

Method	# Params	NFE	FID
1-NFE diffusion/flow from scratch			
iCT-XL/2 [54]	675M	1	34.24
Shortcut-XL/2 [10]	675M	1	10.60
MeanFlow-XL/2 [13]	676M	1	3.43
TiM-XL/2 [62]	664M	1	3.26
α-Flow-XL/2+ [68]	676M	1	2.58
DuoFlow-XL/2 [34]	676M	1	2.46
iMF-XL/2 [14]	610M	1	1.72
Self-Drift-B/2	87M	1	2.66
Self-Drift-M/2	172M	1	1.78
Self-Drift-L/2	406M	1	1.47
Self-Drift-XL/2	608M	1	**1.36**
2-NFE diffusion/flow from scratch			
iCT-XL/2 [54]	675M	2	20.30
IMM-XL/2 [72]	675M	1×2	7.77
MeanFlow-XL/2+ [13]	676M	2	2.20
α-Flow-XL/2+ [68]	676M	2	1.95
iMF-XL/2 [14]	610M	2	1.54
Self-Drift-XL/2	608M	2	**1.30**

Method	# Params	NFE	FID
1-NFE diffusion/flow (distillation)			
π-Flow-XL/2 [5]	675M	1	2.85
DMF-XL/2+ [31]	675M	1	2.16
FACM-XL/2 [45]	675M	1	1.76
Multi-NFE diffusion/flow			
ADM-G [8]	554M	250×2	4.59
LDM-4-G [48]	400M	250×2	3.60
SimDiff [22]	2B	1000×2	2.77
DiT-XL/2 [44]	675M	250×2	2.27
SiT-XL/2 [43]	675M	250×2	2.06
SiT-XL/2 + REPA [67]	675M	250×2	1.42
SiD2 [23]	–	512×2	1.38
LightningDiT-XL/2 [64]	675M	250×2	1.35
DDT-XL/2 [61]	675M	250×2	1.26
RAE [70] + DiT^{DH}-XL	839M	50×2	1.13
Distribution Correction (with Extra Model)			
DriftModel-L/2 [7]	463M	1	1.54
W-Flow-XL/2 [18]	679M	1	1.29

Method	# Params	NFE	FID
GANs			
BigGAN [2]	112M	1	6.95
GigaGAN [26]	569M	1	3.45
StyleGAN-XL [50]	166M	1	2.30
autoregressive/masking			
JetFormer-L [60]	2.75B	256×2	6.64
MaskGIT [3]	227M	8	6.18
RQ-Transformer [30]	3.8B	256×2	3.80
DetailFlow-64 [40]	326M	71	2.62
STARFlow [16]	1.4B	1024×2	2.40
LLamaGen [57]	3.1B	256×2	2.18
FlexVAR [25]	1B	13	2.08
VAR-$d30$ [59]	2B	10×2	1.92
FlowAR-H [46]	1.9B	125	1.65
MAR-H [33]	943M	256×2	1.55
RAR-XXL [66]	1.5B	256×2	1.48
xAR-H [47]	1.1B	50×2	1.24

Table 1: **Class-conditional ImageNet** 256×256 **results.** The three panels separate randomly initialized one- or two-NFE diffusion/flow models, distilled or multi-step methods (including methods with an additional distribution-correction model), and GAN or autoregressive/masked baselines. Each row reports parameter count, number of function evaluations, and FID (lower is better). Self-Drift results use direct from-scratch training. An NFE factor of $\times2$ denotes the paired conditional and unconditional evaluations used for classifier-free guidance.

Inference. Given $\epsilon \sim \mathcal{N}(0, I)$ and class label y, the flow with trained parameters $\theta^\star$ generates a sample in one long-horizon evaluation:

$$\hat{x} = \epsilon + u_{\theta^\star}(\epsilon, y, \omega). \tag{16}$$

Feature extraction, memories, whitening, and drift targets are used only during training, leaving the one-step inference path unchanged.

4 Experiments

4.1 Experimental Setup

Dataset. We use ImageNet images at a resolution of 256×256 (Deng et al., 2009). In line with established latent-space generative modeling practice (Zhou et al., 2025; Frans et al., 2025; Peebles & Xie, 2023; Geng et al., 2025; 2026), our experiments are performed in the representation space of a pre-trained Variational Autoencoder (VAE) tokenizer (Rombach et al., 2022). The tokenizer maps each image to a $32\times32\times4$ latent tensor. Self-Drift models this latent distribution, and its generated representations are decoded by the tokenizer before evaluation in pixel space. The VAE defines the latent input/output interface; the drift geometry is supplied separately by the flow's online intermediate features.

Training. Main results use direct from-scratch training, with shared training settings across model sizes. Unless noted otherwise, we use 8 NVIDIA H800 GPUs, global batch size 1024, and Adam (Kingma & Ba, 2015) with learning rate $2\text{e}{-4}$ (Yao et al., 2025), $(\beta_1, \beta_2) = (0.9, 0.95)$, and zero weight decay. The learning rate increases linearly over 10 epochs and then remains constant. Weight EMA decay increases from 0.9996 to 0.9999 following a cosine schedule. We sample t from a logit-normal distribution (Esser et al., 2024) and use classifier-free guidance (CFG) (Ho & Salimans, 2022) with class dropout probability 0.1. The online drift lens is part of this jointly trained flow and has no separate ImageNet classification pretraining.

Evaluation. Following MeanFlow and iMF (Geng et al., 2025; 2026), we use Fréchet Inception Distance (FID) (Heusel et al., 2017) on 50,000 generated samples as our primary metric, comparing generated and real feature statistics. We report Inception Score (IS) as a complementary metric, noting the FID–IS trade-off observed in iMF. Unless stated otherwise, evaluation uses NFE=1.

4.2 Main Results: From-Scratch One-Step Generation

Table 1 evaluates Self-Drift, which combines our dual-horizon clean-endpoint flow with distribution feedback from its own representations. Our trajectory-only version is evaluated in Table 2b. Self-Drift-



Figure 2: **1-NFE Generation.** Curated generated samples from our Self-Drift-XL/2 model (1.36 FID) with class-conditional guidance on ImageNet 256×256.

XL/2 achieves an FID of 1.36, outperforming the listed from-scratch one-step baselines, including iMF-XL/2 (1.72). The gain is obtained without an external distribution model or a trainable auxiliary network. Figure 2 shows representative 1-NFE samples from Self-Drift-XL/2, with coherent class-conditional content and visual diversity.

Model Scaling and Few-Step Quality. At NFE=1, Self-Drift improves from 2.66 FID for B/2 to 1.47 for L/2 and 1.36 for XL/2, showing that the online drift objective scales with model capacity. Allowing one additional function evaluation further reduces the XL/2 FID to 1.30. Thus, the method maintains strong few-step quality while leaving its one-step inference path unchanged.

4.3 Ablation Studies

Unless stated otherwise, ablations use B/2 at NFE=1 with CFG settings held fixed. For efficient B/2 ablations, all variants start from the same 240-epoch trajectory-only checkpoint (FID 3.63) and continue to epoch 480. Trajectory-only continuation reaches 3.57 FID, while Self-Drift reaches 2.74, close to the from-scratch result of 2.66. Continuation settings are given in Appendix C.1.

Table 2: **Self-Drift ablations and scaling on ImageNet** 256×256 **at NFE=1.** (a) Endpoint and feedback-design variants for B/2. (b) FID results for our dual-horizon flow with and without the Self-Drift loss, with iMF as an interval-based trajectory reference.

Variant	FID ↓	IS ↑
Our flow w/o drift	3.57	259.3
Self-Drift	**2.74**	230.1
Endpoint design		
w/ t conditioning	2.85	226.2
Short-step endpoint	29.25	54.6
Boundary-only	3.01	224.9
Feedback design		
Frozen-self	3.19	236.9
MAE-Drift	2.84	248.2
w/o whitening	2.92	231.7
Direct lens grad.	193.68	5.3

(a) B/2 variants continued to epoch 480 from a shared checkpoint. Direct lens gradients also update ϕ_θ through feature extraction.

Size	Params	GFLOPs	w/o Drift	w/ Drift
iMF (Geng et al., 2026) (640 epochs; XL/2: 800)				
B/2	89M	24.9	3.39	N/A
M/2	174M	49.9	2.27	N/A
L/2	409M	116.4	1.86	N/A
XL/2	610M	174.6	1.72	N/A
Our dual-horizon flow (480 epochs)				
B/2	87M	24.4	3.57	**2.74**
M/2	172M	48.9	2.98	**1.78**
L/2	406M	114.4	2.64	**1.47**
XL/2	608M	171.5	2.46	**1.36**

(b) Scaling and inference cost. B/2 uses the ablation protocol. Each pair uses the same trajectory design and training budget. N/A: no iMF + Self-Drift result is reported. Params and GFLOPs describe the one-step generator and are unchanged by drift.

4.3.1 From Clean Endpoints to Distribution Feedback

Clean-endpoint design. Our dual-horizon formulation (Sec. 3.2) reconstructs clean endpoints from sampled trajectory states, providing the shared input domain needed to compare generated

queries with real references. In Table 2a, adding t conditioning slightly worsens FID from 2.74 to 2.85. Dropping the long-horizon loss and constructing endpoints as $\hat{x}_t = z_t + t\, v_\theta(z_t, y, \omega)$ gives 29.25, supporting dedicated long-horizon learning (Appendix C.1). Boundary-only retains trajectory learning but applies drift to fresh one-step samples from pure noise, giving 3.01 (Appendix C.1). This supports applying endpoint drift throughout the trajectory under the evaluated settings.

Feedback design. Table 2a shows that Self-Drift reduces B/2 FID from 3.57 for trajectory-only continuation to 2.74. Replacing the online lens with Drifting Models' pretrained MAE (Deng et al., 2026) yields 2.84 (Appendix C.1). Allowing direct parameter gradients through the feature lens severely degrades generation (FID 193.68), supporting the frozen-lens semi-gradient that routes feedback through the predicted endpoint. Removing real-anchored whitening raises FID to 2.92, compared with 2.74 with whitening, supporting its role in the correction geometry. DMD (Yin et al., 2024), using a frozen trajectory-only teacher and an online fake-score model, does not improve on the 240-epoch initialization in our setting (Appendix C.1).

Online versus frozen self-representation. Frozen-self keeps a copy of the shared checkpoint's feature lens fixed throughout continuation, retaining input gradients to train the generator. It reaches 3.19 FID and 236.9 IS, compared with 2.74 and 230.1 for Self-Drift (Table 2a). Under the same continuation budget, online features lower FID by 0.45, supporting the value of updating the comparison geometry as the generator learns.

Improvement across model sizes. Table 2b compares our dual-horizon flow with and without the Self-Drift loss. Adding the loss reduces FID by 0.83–1.20 across B/2, M/2, L/2, and XL/2, including a reduction from 2.46 to 1.36 for XL/2. Drift leaves each generator's parameters and inference GFLOPs unchanged. iMF provides a separate trajectory-learning reference; training protocols differ, while Self-Drift achieves lower FID at all four scales.

4.3.2 Sensitivity to the Drift Configuration

Having established the correction design, Table 3 examines the choices that determine its geometry, strength, and reference set.

Feature layer. Layers 9 and 10 give similar FID (2.74 and 2.78), compared with 3.11 at layer 7 and 3.16 at the final layer 12. We use layer 9 for B/2. The independent probe below examines why the correction direction varies with feature depth.

Correction strength. Weights $\lambda_{\mathrm{SD}} = 0.05$ and 0.1 yield similar FID (2.77 and 2.74), while increasing the weight to 0.5 or 1.0 gives 3.12 or 4.25. This favors a modest drift weight alongside trajectory supervision.

Reference and memory sizes. Reducing real references from 64 to 32 gives FID 2.87, while reducing generated references from 64 to 32 gives 2.73, close to the default 2.74. Doubling each per-class bank from 128 to 256 entries gives 2.76, with no observed FID gain from the larger memory. We retain 64 real and 64 generated references with 128 entries per bank as the default.

Table 3: **Sensitivity to Self-Drift design choices.** All results use B/2 at NFE=1; bold settings mark the default configuration.

Drift Layer	FID ↓	IS↑
Layer 7	3.11	222.7
Layer 9	2.74	230.1
Layer 10	2.78	231.0
Layer 12	3.16	231.5

(a) Feature-extraction layer ℓ (one-based index).

λ_{SD}	FID ↓	IS↑
0.05	2.77	234.4
0.1	2.74	230.1
0.5	3.12	225.2
1.0	4.25	191.8

(b) Weight λ_{SD} of the Self-Drift loss.

$M/N/K$	Mem. (MiB)	FID ↓	IS↑
32/64/128	397.6	2.87	229.8
64/32/128	398.6	2.73	229.1
64/64/128	409.5	2.74	230.1
64/64/256	775.5	2.76	227.1

(c) M/N: real/generated references per query; K: capacity of each per-class bank.

4.3.3 Independent Assessment of the Drift Direction

The FID gains and layer dependence motivate a more direct question: does drifting in the flow's own feature space move samples toward the data in an independent representation? We evaluate this direction in Inception space at the shared trajectory-only checkpoint, before any Self-Drift training.

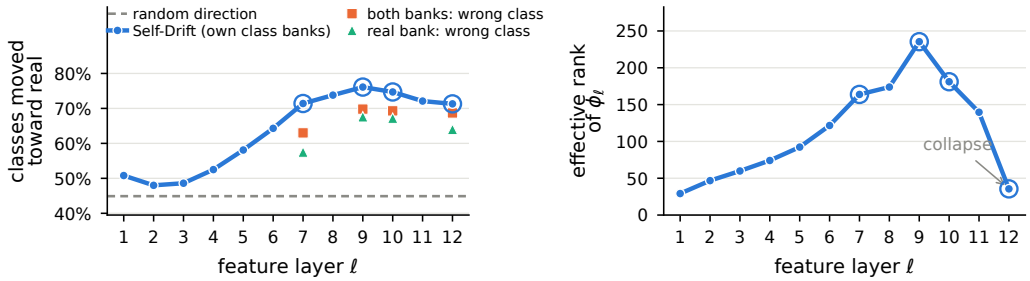


Figure 3: All measurements are on the four ablations' *shared* initialisation. **(a)** Fraction of the 1000 classes with positive alignment to their real-minus-generated centroid direction (binomial SE 1.4 percentage points). Circled markers identify the four layers whose final training FIDs are reported in Table 3a; those FIDs are not used by the probe. **(b)** Effective rank of the whitened ϕ_ℓ: exp of its covariance spectrum's entropy, i.e. how many of the 768 directions carry variance.

Protocol. The four ablations differ only in ℓ and share one trajectory-only initialisation (FID 3.63); we measure there, on the online weights the drift lens itself uses. Per class we build the training banks (64 real and 64 generated one-step endpoints, the trained whitening and R schedule), take the drift gradient $g = \nabla_{\hat{x}} \mathcal{L}_{\mathrm{SD}}$ at a held-out endpoint, and step $\hat{x}' = \hat{x} - \rho \|\hat{x}\|_2 \, g / \|g\|_2$ at matched relative size $\rho = 0.02$. Let ψ denote VAE decoding followed by Inception-2048 feature extraction. We assess whether the displacement $\Delta = \psi(\hat{x}') - \psi(\hat{x})$ points from generated toward real *for that class*: $\cos(\Delta,\ \mu_c^{\mathrm{real}} - \mu_c^{\mathrm{gen}})$, with centroids from 16 held-out real and 16 held-out generated samples each.

The direction is useful, and it is class-conditional. One step at $\ell = 9$ gives positive centroid alignment for 76.1% of the 1000 classes, against 44.9% for an isotropic step of equal norm (Fig. 3a); the same step lowers 8k-sample FID from 8.64 by 0.29 where the random step raises it. Mismatching the class identity of the banks reduces alignment to 69.8% with both banks drawn from a wrong class and 67.6% with only the real bank mismatched (the orange and green markers in Fig. 3a) — and it survives deleting the global real/generated direction outright (72.6% on its orthogonal complement). These controls support a role for class-conditional geometry beyond a global real/generated separation.

The optimum is intermediate, and precedes any drift training. Alignment remains around 50% for $\ell \leq 4$ (48–53%), peaks at $\ell = 9$, then falls (Fig. 3a). Measured wholly at the shared initialisation, it still ranks the four trained ablations in their exact final FID order (Spearman $\rho = 1.0$ with negative FID, across two probe runs), so the association is already present before Self-Drift training. Effective rank (Fig. 3b) provides a complementary observation: it rises through the early layers despite their weak alignment, but falls to 36 at the deepest layer, compared with 236 at $\ell = 9$. In the step-size probe, the deepest layer is more sensitive: alignment falls from 71.0% to 45.7% as ρ increases from 0.02 to 0.1, compared with 75.6% to 59.7% at $\ell = 9$. Together, these observations favor an intermediate representation with both useful directional alignment and higher effective rank. This local probe does not establish that training dynamics follow it; at $\rho = 0.05$ the $\ell = 7/10$ ranks swap.

5 Conclusion

We presented Self-Drift, a joint trajectory–distribution training scheme for one-step flow models. Its dual-horizon flow maps each sampled trajectory state to a shared clean endpoint, while an intermediate representation of the same flow provides class-conditional, multiscale attraction–repulsion feedback. The frozen-lens semi-gradient routes this correction through the endpoint, and real-data-anchored whitening stabilizes the co-evolving feature space. Learned from scratch without an auxiliary correction model, Self-Drift-XL/2 reaches an FID of 1.36 at NFE=1 on class-conditional ImageNet 256×256 while leaving the one-step inference path unchanged. W-Flow (Han et al., 2026) also trains its generator from scratch and achieves a lower one-step FID of 1.29, using an external pretrained feature encoder. Closing this quality gap while retaining the flow's own representation for distribution feedback is a direction for future work.

REFERENCES

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

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022.

Hila Chefer, Patrick Esser, Dominik Lorenz, Dustin Podell, Vikash Raja, Vinh Tong, Antonio Torralba, and Robin Rombach. Self-supervised flow matching for scalable multi-modal synthesis. *arXiv preprint arXiv:2603.06507*, 2026.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation. *arXiv preprint arXiv:2510.14974*, 2025.

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

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *ICLR*, 2025.

Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84, pp. 1608–1617, 2018.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Jiatao Gu, Tianrong Chen, David Berthelot, Huangjie Zheng, Yuyang Wang, Ruixiang Zhang, Laurent Dinh, Miguel Angel Bautista, Josh Susskind, and Shuangfei Zhai. Starflow: Scaling latent normalizing flows for high-resolution image synthesis. *NeurIPS*, 2025.

Yasaman Haghighi, Bastien Van Delft, Mariam Hassan, and Alexandre Alahi. Layersync: Self-aligning intermediate layers. In *International Conference on Learning Representations*, volume 2026, pp. 29918–29950, 2026.

Jiaqi Han, Puheng Li, Qiushan Guo, Renyuan Xu, Stefano Ermon, and Emmanuel J Candès. One-step generative modeling via wasserstein gradient flows. *arXiv preprint arXiv:2605.11755*, 2026.

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. *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. In *CVPR*, 2025.

Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. *arXiv preprint arXiv:2505.02831*, 2025.

Siyu Jiao, Gengwei Zhang, Yinlong Qian, Jiancheng Huang, Yao Zhao, Humphrey Shi, Lin Ma, Yunchao Wei, and Zequn Jie. Flexvar: Flexible visual autoregressive modeling without residual prediction. *Advances in Neural Information Processing Systems*, 38:8955–8978, 2026.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *ICLR*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Chieh-Hsin Lai, Bac Nguyen, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yuki Mitsufuji, Stefano Ermon, and Molei Tao. A unified view of score-based and drifting models. *arXiv preprint arXiv:2603.07514*, 2026.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022.

Kyungmin Lee, Sihyun Yu, and Jinwoo Shin. Decoupled meanflow: Turning flow models into flow maps for accelerated sampling. *arXiv preprint arXiv:2510.24474*, 2025.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, 2017.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

Zeming Li, Xiangyu Zhang, Ping Tan, and Heung-Yeung Shum. Duoflow: Jvp-free finite-difference mean flows for one-step image generation. In *European Conference on Computer Vision (ECCV)*, 2026.

Zhiqi Li and Bo Zhu. A long-short flow-map perspective for drifting models. *arXiv preprint arXiv:2602.20463*, 2026.

Shanchuan Lin and Xiao Yang. Diffusion model with perceptual loss. *arXiv preprint arXiv:2401.00110*, 2024.

Shanchuan Lin, Ceyuan Yang, Zhijie Lin, Hao Chen, and Haoqi Fan. Adversarial flow models. *arXiv preprint arXiv:2511.22475*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Yiheng Liu, Liao Qu, Huichao Zhang, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Xian Li, Shuai Wang, Daniel K Du, et al. Detailflow: 1d coarse-to-fine autoregressive image generation via next-detail prediction. *arXiv preprint arXiv:2505.21473*, 2025.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. In *ICLR*, 2025.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023.

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 *ECCV*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *CVPR*, 2023.

Yansong Peng, Kai Zhu, Yu Liu, Pingyu Wu, Hebei Li, Xiaoyan Sun, and Feng Wu. Flow-anchored consistency models. *arXiv preprint arXiv:2507.03738*, 2025.

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Flowar: Scale-wise autoregressive image generation meets flow matching. *arXiv preprint arXiv:2412.15205*, 2024.

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. In *ICCV*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *ICLR*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Is noise conditioning necessary for denoising generative models? *arXiv preprint arXiv:2502.13129*, 2025.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text. In *ICLR*, 2025.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025a.

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. *arXiv preprint arXiv:2509.04394*, 2025b.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15703–15712, 2025.

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

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *ICCV*, 2025a.

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 *ICLR*, 2025b.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. *arXiv preprint arXiv:2510.20771*, 2025.

Yuan Zhang, Chenyi Li, Haodong Yu, Guoqing Ma, Jiajun Zha, Yuanming Yang, Bo Wang, Wei Tang, Wenbo Li, Haoyang Huang, and Nan Duan. Teacher-feature drifting: One-step diffusion distillation with pretrained diffusion representations. *arXiv preprint arXiv:2605.07327*, 2026.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Huangjie Zheng, Xu Chen, Jiangchao Yao, Hongxia Yang, Chunyuan Li, Ya Zhang, Hao Zhang, Ivor Tsang, Jingren Zhou, and Mingyuan Zhou. Contrastive attraction and contrastive repulsion for representation learning. *Transactions on Machine Learning Research*, 2023.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. In *ICML*, 2025.

A Declaration of LLM usage

LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology or originality of the research.

B Method Supplement

B.1 Pathwise Motivation for Endpoint Consistency

We motivate the two prediction tasks using a continuously differentiable path ζ_t, $t \in [0, 1]$, with endpoint ζ_0 in the clean-data domain. The quantities below describe this specified path, independently of the learned predictions. Define its velocity toward decreasing t and its remaining displacement per unit time by

$$v_{\mathrm{path}}(t) = -\partial_t \zeta_t, \tag{17}$$

$$u_{\mathrm{path}}(t) = \frac{\zeta_0 - \zeta_t}{t}, \qquad t > 0. \tag{18}$$

The fundamental theorem of calculus gives

$$t\, u_{\mathrm{path}}(t) = \zeta_0 - \zeta_t = \int_0^t v_{\mathrm{path}}(\tau)\, \mathrm{d}\tau. \tag{19}$$

Thus $u_{\mathrm{path}}(t)$ is the average velocity over the remaining path. For $0 < \delta \le t$, let $\bar{v}_\delta(t) = (\zeta_{t-\delta} - \zeta_t)/\delta$ be the exact average velocity over the short interval. Splitting the displacement yields

$$t\, u_{\mathrm{path}}(t) = \delta \bar{v}_\delta(t) + (t - \delta) u_{\mathrm{path}}(t - \delta), \tag{20}$$

where the remaining displacement is defined as zero when $t = \delta$. This identity is exact for the path increments, and $\bar{v}_\delta(t) \to v_{\mathrm{path}}(t)$ as $\delta \to 0$.

Connection to the training loss. Equation (6) uses this displacement decomposition as a consistency target. It substitutes $\delta \widetilde{v}_t$ for the short displacement and evaluates the remaining displacement at the predicted local step $\widetilde{z}_{t^-} = z_t + \delta\, \mathrm{sg}[v_\theta]$. The local displacement used to select this state differs from the target by $\delta(\mathrm{sg}[v_\theta] - \widetilde{v}_t)$, whose forward value is the short-loss residual. The guided target is held fixed within each update, but changes with the model across updates. It need not be the velocity of the original linear interpolation path.

Scope. These pathwise identities motivate the objective rather than establish a population-level equivalence. With stochastic data–noise pairs, the local target need not be exactly predictable from the network inputs, so joint zero residuals are an idealized condition. In particular, this argument does not show that omitting time conditioning preserves the optimum of a time-conditioned objective, or that the joint training procedure converges to the data distribution. Positive loss weights and detached normalization preserve attainable zero residuals, but can change stationary points when residuals remain nonzero.

B.2 Finite-Difference Correction and Loss Weighting

For a fixed training sample and the current parameters, hold t, δ, the target evaluations, and the adaptive normalization denominators fixed during differentiation, as in Section 3.2. Using the detached u_θ^- and $\widetilde{v}_t$, define the finite-difference quantity

$$V_\delta = \frac{t u_\theta - (t - \delta) u_\theta^-}{\delta} = u_\theta + (t - \delta)\frac{u_\theta - u_\theta^-}{\delta}. \tag{21}$$

The long-horizon loss is then $\ell_{\mathrm{long}} = \|\delta(V_\delta - \widetilde{v}_t)\|_2^2$. Writing $J_u = \partial u_\theta / \partial \theta$, stop-gradient on u_θ^- gives $\partial V_\delta / \partial \theta = (t/\delta) J_u$. To isolate this factor, consider the same forward estimate with its finite-difference correction detached:

$$\overline{V}_\delta = u_\theta + (t - \delta)\, \mathrm{sg}\left[\frac{u_\theta - u_\theta^-}{\delta}\right]. \tag{22}$$

Now $\overline{V}_\delta = V_\delta$ in value, but $\partial \overline{V}_\delta / \partial \theta = J_u$. Both losses below therefore have the same residual values and detached normalization denominators, yielding the exact first-order identity

$$\nabla_\theta \left[\frac{\delta}{t} \mathcal{A}(\ell_{\text{long}}) \right] = \nabla_\theta \mathcal{A}\big(\|\delta(\overline{V}_\delta - \widetilde{v}_t)\|_2^2 \big) . \tag{23}$$

Keeping the displacement scale δ on both sides also covers the unscaled branch of $\mathcal{A}$ for vanishing losses. Thus, δ/t compensates for the explicit finite-difference Jacobian factor without computing a JVP. The equality concerns the first-order gradient of the long-horizon loss term at the current update, with its targets and normalization held fixed. It does not imply statistical consistency of the joint objective; residuals and network Jacobians can still vary with t.

B.3 Training Algorithm and Implementation Advantages

Algorithm 1 summarizes the core training objective. For each interpolated state, the flow predicts a local displacement and a direct displacement to the clean endpoint. The trajectory loss combines local supervision with consistency between the direct displacement and a local step followed by the remaining displacement. The resulting endpoint is then compared with real and generated references through the flow's own features. Current features enter the banks only after the drift loss is computed, keeping each query out of its reference set. The pseudocode shows the conditional training path with classifier-free guidance: `w` denotes ω in Eq. (5), and `c` denotes the class labels. The unconditional local prediction uses zero guidance, and the detached guided target supervises both trajectory losses. Class dropout and the drift-activation schedule are omitted; dropped-label samples use $\omega = 0$ for trajectory supervision and are excluded from drift queries. The helper `drift_loss` encapsulates class-conditional reference sampling, whitening, and the detached attraction–repulsion target.

Endpoint-consistency learning without JVPs. The long-horizon branch parameterizes the remaining displacement as $t\, u_\theta$, motivated by the pathwise relation in Section B.1. Its consistency target uses predictions at the current state and a locally shifted state. This requires ordinary network evaluations and a detached target, with no Jacobian–vector products (JVPs), explicit time derivatives, or higher-order differentiation. The local branch supplies the short step, avoiding a multi-step rollout when constructing the target. The two regression losses couple local transport supervision with endpoint prediction.

Standard first-order backpropagation. Self-Drift preserves this simple differentiation structure. Its reference features, whitening transform, and drift target are detached. The query feature retains an input gradient through the frozen feature map, which ordinary reverse-mode backpropagation passes to the predicted endpoint and then to the flow parameters. Freezing the feature parameters avoids direct parameter-gradient computation along the feature-extraction path. No Jacobian is materialized, and there is no differentiation through a gradient computation. The trajectory and drift losses can therefore be combined in a single backward pass.

Compatibility with compilation and fused attention. The implementation provides `torch.compile` paths for both the generator and feature-extraction forwards. Predictions at the two horizons are batched together, and target evaluations are similarly grouped, reducing separate model invocations. The backbone uses PyTorch's scaled dot-product attention interface, which can dispatch to FlashAttention on supported hardware and data types. Since the objective requires only standard forward and backward operations, these attention kernels need no forward-mode autodifferentiation or double-backward support. Mixed-precision execution is also compatible: the feature path uses bfloat16 autocasting while drift distances and losses are evaluated in float32.

Simple tensor operations and reusable references. Outside the network, the core losses use additions, scalar multiplication, squared residuals, matrix products, distances, softmax weights, and reductions. The drift target is computed directly from weighted reference features, without an iterative transport solver or an inner optimization loop. Whitening uses running real-feature statistics and a periodically refreshed eigendecomposition, both outside the gradient graph. These bank and statistics updates can remain outside the compiled network forwards. The feature banks reuse past computations to provide references beyond the current minibatch, with fixed capacity and a fixed number of sampled references per query. This bounds storage and reference-processing cost independently of training duration.

Algorithm 1 Self-Drift training and one-step inference with classifier-free guidance (CFG).

```
# fn(z, h, w, c): velocity toward data; h=0 (local), h=1 (endpoint)
# phi(z, c): pooled fn features at h=1 and fixed guidance
# phi freezes weights and retains input gradients
# norm_mse: per- example MSE divided by its stop- gradient value
# x, c: data and class batches; dt: local time interval (0.01)

t, w = sample_t_cfg()
t = t.clamp(min=dt)
e = randn_like(x)
z = (1 - t) * x + t * e

# Conditional and unconditional local velocities
v_c = fn(z, 0, w, c)
v_u = fn(z, 0, 0, None)

# CFG target
v_tgt = (x - e) + w * sin(pi * t)**3 * (v_c - v_u)

# Dual- horizon trajectory loss
short_step = v_c * dt
long_step = fn(z, 1, w, c) * t
next_z = z + stopgrad(short_step)
next_step = stopgrad(fn(next_z, 1, w, c)) * (t - dt)
target_step = stopgrad(v_tgt) * dt

loss_short = norm_mse(short_step - target_step)
loss_long = norm_mse(long_step - target_step - next_step)
loss_traj = (w_short * loss_short + (dt / t) * loss_long).mean()

# Self- Drift on an active iteration
x_hat = z + long_step
q, p = phi(x_hat, c), stopgrad(phi(x, c))
update_real_stats(p)
W = detached_whitener() # periodically refreshed from real statistics
loss_drift = drift_loss(q, real_bank, gen_bank, c, W)

gen_bank.add(stopgrad(q), c)
real_bank.add(p, c)

loss = loss_traj + w_drift * loss_drift

# One- step inference with trained fn (given shape, w, c)
e = randn(shape)
x_gen = e + fn(e, 1, w, c)
```

Shared computation and unchanged one-step inference. The drift query reuses the endpoint already produced for trajectory learning, so it requires no additional full-generator prediction. Feature extraction evaluates only the required prefix of the same flow, with no separately trained encoder, teacher, or discriminator and no additional learned parameters for distribution correction. Both objectives update the same model in one training process. At inference, all feature extraction, banks, whitening, and drift computations are removed; sampling remains a single long-horizon evaluation. Section C.2 quantifies the additional training computation separately.

C Additional Ablations

C.1 Training Protocols

Training configuration. Alongside the settings in Section 4.1, we use $\delta = 0.01$, $w_{\text{short}} = 5$, and $\lambda_{\text{SD}} = 0.1$. Trajectory supervision, feature-memory updates, and real-feature statistics run

throughout training. Following the initial learning-rate warm-up, drift is active every epoch or every other epoch (`drift_freq`=1 or 2), at every iteration within an active epoch. The feature lens uses guidance condition $10/11$ and, for B/2, layer 9 of the full shared backbone. For conditional training samples, we draw ω uniformly from $[15/16 - 10/11,\ 15/16]$ and use $\omega = 0.85$ for evaluation. Full experiment configurations will be released with the code.

Continuation for efficient ablations. The B/2 ablations reuse the online weights of a 240-epoch trajectory-only checkpoint (FID 3.63) and continue to epoch 480, using $\delta = 0.005$, $w_{\text{short}} = 1$, a learning rate of $2\text{e}{-4}/3$ at global batch size 1024, no learning-rate warm-up, and a fixed EMA decay of 0.9999. The first 1000 continuation steps populate the feature banks and real-feature statistics; thereafter drift is active at every iteration with weight 0.1. The guidance settings, feature layer, and memory configuration are unchanged.

MAE-Drift. We use the pretrained latent-space MAE released by Drifting Models (Deng et al., 2026), with a ResNet-style encoder of base width 256. This ablation replaces the flow's feature extractor in the drift path with the frozen MAE, retaining gradients with respect to generated endpoints. The trajectory objective, feature-bank procedure, whitening, and drift loss are unchanged. The result is 2.84 FID and 248.2 IS.

DMD comparison. We evaluate the distribution-matching objective of DMD (Yin et al., 2024) using the short-step branch of a frozen trajectory-only checkpoint as the teacher. We retain trajectory supervision and add the DMD loss with weight 0.1, giving $\mathcal{L}_{\text{traj}} + 0.1\mathcal{L}_{\text{DMD}}$. The DMD term uses a fresh one-step sample x_g generated from independent Gaussian noise. For the perturbed sample $\widetilde{x}_\tau = (1-\tau)x_g + \tau\epsilon'$, with $\epsilon' \sim \mathcal{N}(0, I)$ and $0 < \tau < 1$, the flow-matching velocity-to-score conversion in our data-directed velocity convention is

$$s_T(\widetilde{x}_\tau, \tau) = \frac{(1-\tau)v_T(\widetilde{x}_\tau) - \widetilde{x}_\tau}{\tau}, \tag{24}$$

where v_T is the teacher's short-step prediction and conditioning arguments are omitted. A separate fake-score model is trained on the current generated samples using flow matching and the same score conversion. The fake-minus-teacher score difference supplies the generator's DMD gradient; the alternating fake-score and generator updates follow the original DMD scheme. We use a teacher trained for 240 epochs and also test a 480-epoch teacher. In the 240-epoch setting, the best observed FID is 3.63, matching the initialization. These results concern correction with our own teacher checkpoints; the DMD objective targets the teacher distribution rather than directly minimizing data FID.

Short-step endpoint ablation. We remove the long-horizon consistency loss and retain only the short-horizon term in $\mathcal{L}_{\text{traj}}$. Self-Drift remains active, with endpoints constructed as $\hat{x}_t = z_t + t\, v_\theta(z_t, y, \omega)$ from the local prediction at $h = 0$. These endpoint latents are compared with clean real latents through the feature lens. This variant tests endpoint reconstruction by extrapolating the local velocity without dedicated long-horizon learning.

Boundary-only ablation. The trajectory objective still uses randomly sampled t. For drift queries, we instead draw fresh noise ϵ' and form $\hat{x}_1 = \epsilon' + u_\theta(\epsilon', y, \omega)$ with an additional gradient-enabled generator evaluation. The generated bank stores features of these boundary samples; the real bank, feature lens, and drift objective follow the same configuration. This isolates the choice of endpoints used for distribution correction while retaining both trajectory tasks.

C.2 Training Computational Cost

We estimate the additional training FLOPs of Self-Drift relative to our trajectory-only B/2 flow, holding the architecture, batch size, and number of training steps fixed. This is an analytical estimate of the training computation, not a measurement of GPU time. We consider a 12-layer generator with features extracted after layer 9 of the full shared backbone and a per-device batch size of 128. We report costs separately for drift-active and bank-only iterations, covering both activation schedules in Appendix C.1.

Trajectory-only baseline. Let F denote the forward FLOPs of the full generator for one sample and B the training batch size. The trajectory objective uses one size-$2B$ forward pass with gradients for the short- and long-horizon predictions, and one size-$2B$ forward pass without gradients for their targets. Approximating the full backward pass by twice its forward cost,

$$C_{\text{base}} \simeq \underbrace{2BF}_{\text{prediction forward}} + \underbrace{2BF}_{\text{target forward}} + \underbrace{4BF}_{\text{backward}} = 8BF. \tag{25}$$

The baseline omits feature extraction and bank maintenance entirely; setting the drift weight to zero while retaining these computations would not give this cost.

Additional Self-Drift computation. The generated endpoint is reused from the trajectory prediction, so drift does not require another full-generator forward pass. The fraction of the batch used for feature extraction is $\rho = (128 - \lfloor 0.1 \times 128 \rfloor)/128 = 116/128$, accounting for label dropout. Let $F_\phi \simeq (9/12)F$ denote a feature forward and G_ϕ its input-only backward cost. Real and generated samples each require one feature forward; only the generated features require a backward pass through the frozen lens. Thus, on an active iteration,

$$\Delta C_{\text{active}} \simeq \rho B(2F_\phi + G_\phi) + C_{\text{aux}}. \tag{26}$$

There are no parameter gradients along the lens path. The endpoint gradient joins the trajectory gradient in a single backward pass through the generator, already counted in Eq. (25). With $G_\phi \simeq F_\phi$, the leading-order increase is $3\rho(9/12)/8 \simeq 25.5\%$.

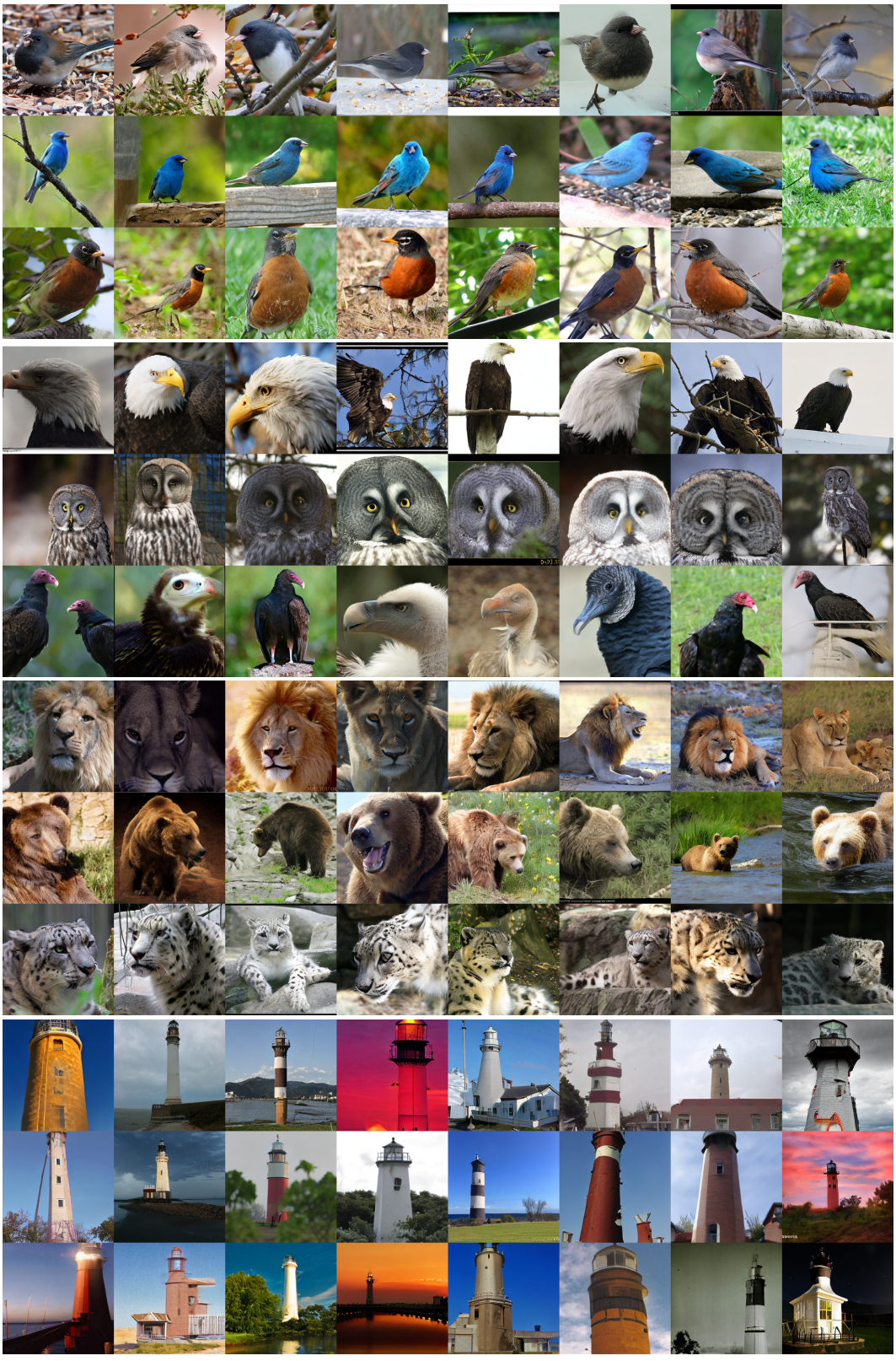
For a more detailed estimate, each block has $n = 272$ tokens (256 image and 16 conditioning tokens), width $d = 768$, and SwiGLU hidden width $8d/3$. Counting a multiply–add as two FLOPs, the leading forward costs are $P = 24nd^2$ for linear projections and $A = 4n^2d$ for attention matrix products. The frozen-lens input backward costs approximately $P + 2A$ per block, rather than $P + A$, giving $G_\phi \simeq 1.056F_\phi$. The auxiliary term C_{aux} includes real-feature statistics, whitening, and the drift target over 64 real and 64 generated references. Whitening the sampled references costs approximately $2\rho B(64+64)d^2$ FLOPs; the remaining feature-space operations and the whitener refresh every 100 iterations are small relative to the backbone. Reference features are retrieved from the banks, not recomputed by the network. Including these terms gives an active-stage increase of about 26%.

Training setting	Relative FLOPs	Increase
Trajectory-only	$1.000\times$	—
Self-Drift: bank-only	$\approx 1.170\times$	$\approx 17.0\%$
Self-Drift: active	$\approx 1.260\times$	$\approx 26.0\%$

Table 4: Analytical training FLOPs relative to the trajectory-only B/2 flow with layer-9 features. Values are per-iteration estimates, not hardware measurements.

On bank-only iterations, both feature forwards and the real-feature statistics are still computed, but the lens backward and drift coupling are omitted. The extra cost is therefore approximately $2\rho BF_\phi$, or 17% of the baseline. Table 4 summarizes these per-iteration costs. If a fraction η of training iterations is drift-active, the average relative cost is approximately $1.17 + 0.09\eta$. These estimates count the dominant matrix operations and exclude lower-order backbone operations, optimizer and EMA updates, data loading, communication, compilation, and evaluation. They should not be interpreted as wall-clock overhead. Feature extraction, whitening, and banks are absent at inference, so the generator's parameter count and one-step inference FLOPs remain unchanged.

D ADDITIONAL QUALITATIVE RESULTS



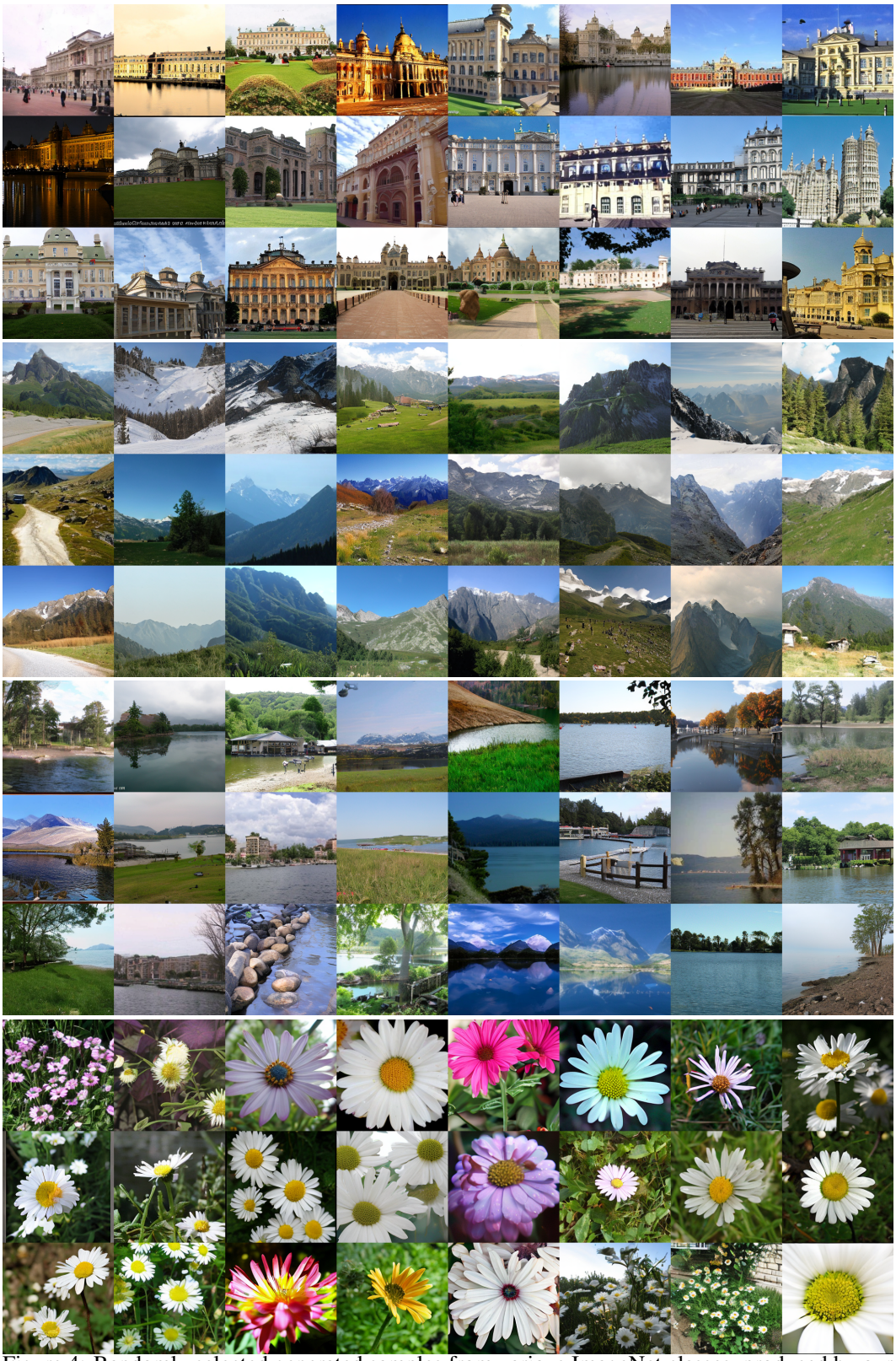


Figure 4: Randomly selected generated samples from various ImageNet classes, produced by our Self-Drift-XL/2 model (1.36 FID) with one-step sampling at 256×256 resolution.