EXPANDING THE QUALITY–COVERAGE FRONTIER WITH COMPETITIVE FLOW MATCHING

Anonymous authors
Paper under double-blind review

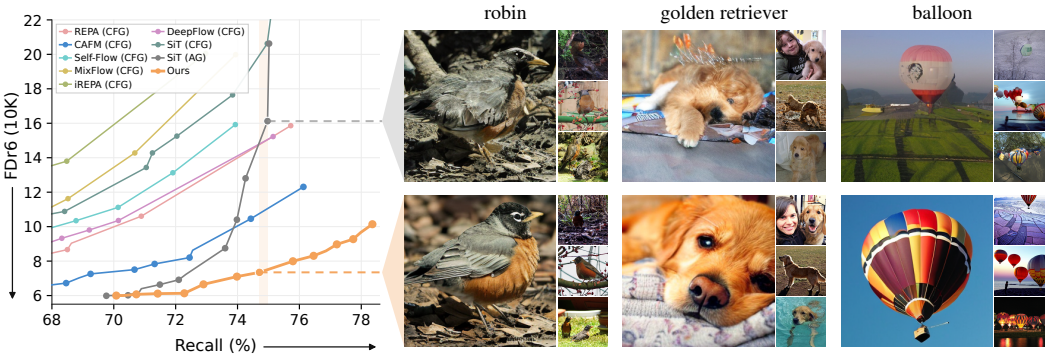


Figure 1: **Our method expands the quality–coverage frontier toward high-coverage, high-quality generation.** $\mathrm{FD}_{r,6}$–Recall Pareto frontiers reported on ImageNet-256, compare our method with recent strong flow-matching methods. Curves are obtained by sweeping each method's inference-time guidance. CFG and AG denote classifier-free guidance and AutoGuidance, respectively. Our method reaches substantially stronger performance in the high-coverage regime. This is further illustrated on the right using random samples, comparing samples obtained using the same class condition and initial noise before (top) and after (bottom) applying our approach over the SiT baseline.

ABSTRACT

Flow matching (FM) generators often face a quality-coverage tradeoff. Inference-time techniques such as classifier-free guidance raise quality, but at the cost of coverage. We hypothesize this limitation stems from a basic property of the objective – an intermediate noisy image could correspond to *many* clean images, yet the model can only predict a *single* velocity, at best averaging over the multiple possibilities. To mitigate this, we introduce competition among predictions during training. We achieve this by augmenting the predictor with a randomly drawn low-dimensional conditioning code z. For each training example, we sample K values of z to produce K candidate velocities that compete to explain the target, and apply the loss only to the best-matching candidate. This winner-take-all objective allows different modes to shape different predictions, rather than being averaged into a single compromise. Furthermore, this term naturally enables a new form of guidance, auxiliary conditioning scaling, which complements traditional guidance methods. Together, these two guidance mechanisms show that our method substantially improves the quality–coverage frontier. We validate on ImageNet-256, using two different testbeds, SiT and RAEv2. Given less than 3% of the original SiT pretraining compute for post-training, at the 67% Recall point, our method reduces $\mathrm{FD}_{r,6}$ from 19.22 to 5.91. On RAEv2, at the 70% Recall point, it reduces $\mathrm{FD}_{r,6}$ from 16.83 to 3.18.

1 INTRODUCTION

A successful generative model has two essential aspects: (1) *high-fidelity*, where each randomly drawn sample looks realistic, and (2) *broad coverage*, where the full data distribution is well-

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.

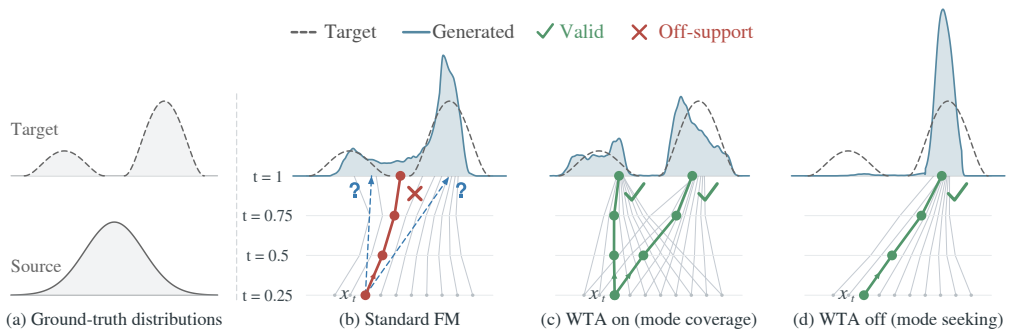


Figure 2: **A 1D toy illustrating conflicting velocity targets in flow matching.** (a) Transport from a unimodal source to an imbalanced bimodal target. (b) Standard FM produces off-support mass and incomplete mode coverage. (c) Winner-take-all (WTA)-trained FM sampled with auxiliary conditioning recovers both target modes while largely removing off-support mass; two fixed codes are shown to visualize distinct code-conditioned trajectories. (d) Sampling the same WTA-trained model without the auxiliary conditioning instead concentrates generation on the dominant mode.

represented. Flow Matching (FM) (Lipman et al., 2023; Ma et al., 2024) and follow-ups in architecture, representations, and training objectives (Lin et al., 2026; Yu et al., 2024; Shin et al., 2025; Chefer et al., 2026) have yielded groundbreaking improvements in quality. In particular, to show impressive individual samples, inference-time techniques such as classifier-free guidance (Ho & Salimans, 2022) and improvements (Sun et al., 2024; Chung et al., 2025; Karras et al., 2024) deliberately move towards the mode of the learned distribution, often at the expense of the tails. In Fig. 1, we demonstrate that while progress has indeed been made in quality (moving down the y-axis), the *coverage* of the distribution (x-axis) has been underexplored. In this work, we demonstrate improvements in both quality and coverage, pushing the frontier significantly.

To understand where this trade-off comes from, we revisit flow matching through a 1D toy example (Fig 2). Flow matching learns a velocity field, transporting samples from a source distribution to a target distribution through a sequence of local regression problems. At training time, it constructs intermediate states by interpolating between randomly drawn source and target samples, training a predictor to the target. Importantly, a given intermediate state is compatible with *multiple* targets, each inducing a different velocity. Because the model is only equipped to predict a *single* velocity, this pulls the prediction in different directions. With finite modeling capacity, this diverts generation away from the data support and leaves valid alternatives underrepresented, as shown in Fig 2(b). This motivates our key insight: to **accommodate multiple alternatives with multiple predictions.**

To allow the model to make multiple predictions for the same noisy state, we introduce a low-dimensional Gaussian code as auxiliary conditioning. During training, we sample multiple auxiliary codes, yielding K candidate velocity predictions for the same intermediate state. Only the best-matching candidate receives the regression loss – a winner-take-all assignment explored in (Guzmán-Rivera et al., 2012; Rupprecht et al., 2017). In our Competitive Flow Matching setup, different targets can thus shape different predictions rather than being averaged into a single compromise. This enables the model to produce different velocity predictions for the same noisy state, as shown in Fig. 2(c), natively accounting for both modes. Interestingly, turning off the auxiliary code, which reverts the model to only being able to predict a single velocity, reveals an advantageous emergent property. As shown in Fig. 2(d), rather than going off-manifold, the generation is able to concentrate on the dominant target mode.

We validate this idea on class-conditional ImageNet-256 generation, systematically comparing against a broad range of methods in a shared setting. We first apply Competitive Flow Matching to the pretrained SiT (Ma et al., 2024) testbed through lightweight post-training. Without inference-time guidance, our method strengthens the pretrained generator and, by varying the auxiliary-conditioning scale, exposes a smooth quality–coverage curve rather than a single operating point. More importantly, when combined with inference-time guidance, the technique changes how the generator responds to guidance: generation quality improves substantially while coverage is largely preserved, and in some cases coverage and sample variation *even both improve* with stronger guidance. Together, these effects

substantially expand the quality–coverage frontier, reaching high-quality, high-coverage regions that previous methods do not access (Fig. 1). We further apply our technique to RAEv2 (Singh et al., 2026b), a state-of-the-art ImageNet generator operating in a pretrained representation space. WTA further improves this already strong model, establishing a new state of the art under its evaluation setting, with especially large gains at high coverage. Finally, we analyze how WTA reshapes the trained generator itself and uncover a complementary high-fidelity behavior induced by competition.

2 Related Work

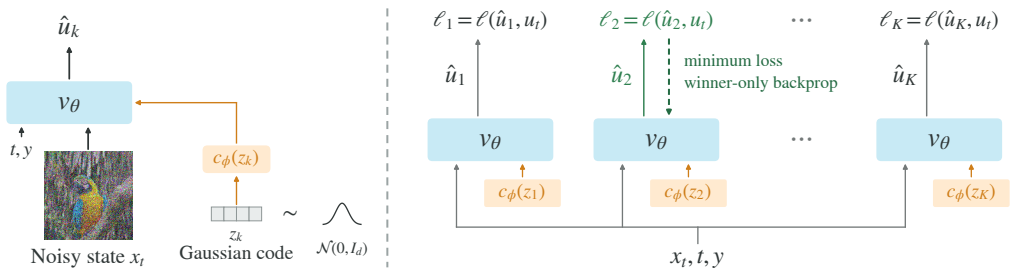
Advances in flow matching. Flow matching has become a powerful framework for image generation. Architectural improvements have made flow-based image models increasingly scalable, while representation-based methods further strengthen the features learned during generation through external alignment, self-supervision, and intermediate supervision (Yu et al., 2024; Singh et al., 2026a; Wang et al., 2025; Jiang et al., 2026; Haghighi et al., 2026; Chefer et al., 2026; Shin et al., 2025). More recent work has also revisited the learning signal itself, for example by training on states closer to those encountered during generation (Li et al., 2026), constructing more stable or informative velocity targets (Yang et al., 2026a; Stoica et al., 2026), and augmenting the standard regression objective with adversarial or contrastive supervision (Lin et al., 2026; Stoica et al., 2025). These methods have pushed generation quality higher, but it is less clear whether the gains extend to broad coverage: in our controlled evaluation, broad-coverage settings remain substantially weaker than the strongest lower-coverage operating points.

Inference-time guidance. Beyond improving the generator itself, inference-time guidance is widely used to improve the fidelity of individual samples. Classifier-free guidance (CFG) (Ho & Salimans, 2022) steers generation toward the modes of the learned distribution, improving sample quality at the cost of coverage. Subsequent methods aim to keep these gains while losing less coverage: Inner CFG incorporates higher-order conditioning information (Sun et al., 2024), CFG++ attributes the degradation at strong guidance to off-manifold extrapolation and keeps the sampling update closer to the data manifold (Chung et al., 2025). AutoGuidance guides a strong model with a weaker version of itself, using their prediction difference to improve sample quality while better preserving variation (Karras et al., 2024). These methods make the quality–coverage trade-off more favorable, but their quality gains still come primarily from moving away from broad coverage. Rather than modifying the sampler, our work changes how flow matching learns from its training targets, allowing guidance to reach much higher quality at broad coverage.

Winner-take-all learning. Winner-take-all (WTA) learning handles multimodal prediction by producing several candidates per input and updating only the one closest to the target, so that candidates specialize to different modes. Multiple Choice Learning introduced this principle for structured prediction, first with an ensemble of models and later with multiple heads of a single network (Guzmán-Rivera et al., 2012; Lee et al., 2016; Rupprecht et al., 2017). IMLE applies the same nearest-candidate matching to implicit generative modeling, matching each data point to its nearest generated sample (Li & Malik, 2018), and conditional IMLE extends it to conditional generation by drawing several latent codes per input (Li et al., 2020). We bring this formulation to local velocity prediction in flow matching. For the same intermediate state, multiple auxiliary-conditioned predictions compete to match the sample-wise velocity target, allowing different target velocities to be fit by different predictions. Concurrent work on Explorative Modeling (XM) (Gladstone et al., 2026) also trains generative models with WTA, but its candidates come from different noise realizations of the same clean sample and therefore lie at different noisy states, whereas ours share a single state. Furthermore, XM targets expressivity and training efficiency, whereas we target generation quality at broad coverage.

3 Method

Background Flow matching learns a time-dependent velocity field that transports a simple noise distribution to the data distribution (Lipman et al., 2023; Ma et al., 2024). Given a data sample $x_0 \sim p_{\text{data}}$, Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and time t, the training procedure constructs intermediate states x_t by interpolating between data and noise points using interpolation schedule α_t, σ_t, creating



(a) Auxiliary conditioning (b) WTA training

Figure 3: **Auxiliary conditioning and WTA training.** (a) A Gaussian code z_k is mapped through c_ϕ to condition the predictor v_θ. (b) WTA compares K predictions from one shared model against the same target. Only the best match backpropagates.

sample-wise velocity targets u_t:

$$x_t = \alpha_t x_0 + \sigma_t \epsilon, \qquad u_t = \dot{\alpha}_t x_0 + \dot{\sigma}_t \epsilon. \tag{1}$$

The velocity model v_θ is trained by squared regression, and may incorporate any extra conditioning signal y,

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[\|v_\theta(x_t, t, y) - u_t\|_2^2\right]. \tag{2}$$

A given state x_t and time t may correspond to many different data–noise pairs, each corresponding to a different target u_t. Standard FM fits these different targets with a single velocity prediction, which is their conditional mean:

$$v^*(x_t, t) = \mathbb{E}[u_t \mid x_t, t, y]\,. \tag{3}$$

3.1 Winner-Take-All Training

Injecting low-dimensional conditioning. To allow multiple velocity predictions for the same noisy state, we introduce a low-dimensional Gaussian code z. We use c_ϕ to map z to a backbone-compatible auxiliary conditioning signal into v_θ, as shown in Fig. 3(a). Varying z enables the network to produce different velocity predictions for the same state. Then, during training, for each noisy state x_t, we draw K independent codes, producing K candidate velocity predictions:

$$\widehat{u}_k = v_{\theta,\phi}\big(x_t, t, y, c_\phi(z_k)\big), \text{where } z_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d),\ k \in \{1, \ldots, K\} \tag{4}$$

As shown in Figure 3(b), among the K candidates, we select the one with the smallest FM regression error and backpropagate only through the winning prediction, which minimizes

$$\mathcal{L}_{\mathrm{WTA}}(\theta, \phi) = \mathbb{E}_{x_0,y,\epsilon,t,z_{1:K}}\left[\min_{k\in\{1,\ldots,K\}} \|\widehat{u}_k - u_t\|_2^2\right]. \tag{5}$$

Implementation details. Though this is a general objective, in our experiments, we leverage a pretrained model θ_0 and continue training the generator with this objective, which we refer to as *WTA post-training*.

The realization of the auxiliary conditioner c_ϕ depends on the backbone architecture. We use each model's native conditioning mechanism to implement c_ϕ. We take SiT as a representative example. SiT (Ma et al., 2024) is a transformer-based flow-matching generator operating in the latent space of a pretrained VAE. Its original conditioning inputs, time t and class label y, are mapped to AdaLN modulation parameters that control the scale, shift, and gating of each transformer block. To implement c_ϕ, we map the Gaussian code z through a small MLP into the same conditioning space and use a parallel AdaLN pathway to produce an auxiliary modulation of the same form. This auxiliary modulation is added to the native time–class modulation before being applied to the block. Thus, in SiT, c_ϕ corresponds to the code embedding together with the parallel AdaLN conditioning pathway. We provide the corresponding implementations for other backbones, along with exact architectural and initialization details, in the appendix.

3.2 Inference and Guidance

Once trained, there are multiple possibilities for how the auxiliary code should be used along a sampling trajectory during inference. We can either sample one code and keep it fixed throughout the generation trajectory, or independently resample the code at each sampling step. We empirically find that the former works better overall, particularly in the broad-coverage regime. Intuitively, this may be because keeping the same code may allow its influence on the predicted velocity field to remain consistent across sampling steps, while resampling may change this influence at every step, weakening the cumulative effect. Therefore, unless otherwise specified, we sample $z \sim \mathcal{N}(0, I_d)$ once and keep it fixed across all sampling steps. We provide a detailed comparison with resampled codes in the appendix.

Guidance. When combining our method with inference-time guidance, there are two adjustable axes: the auxiliary-conditioning scale and the guidance strength.

Auxiliary-conditioning scale λ. Each auxiliary code induces a different code-conditioned prediction. Intuitively, these predictions can be viewed as querying different conditional velocity fields, but they are not independent. During training, they all share the same base predictor v_θ., and winner-selected updates from different codes jointly shape these shared parameters. The WTA-trained model can therefore be viewed as having a base predictor jointly shaped across codes, with the auxiliary condition producing a code-dependent response. We therefore expose the strength of this code-dependent effect as a continuous inference-time control by scaling the learned auxiliary condition $c_\phi(z)$ with λ, yielding

$$v_\lambda = v_\theta(x_t, t, y, \lambda c_\phi(z)) \,. \tag{6}$$

where λ defines the auxiliary-conditioning scale. At $\lambda = 0$, the auxiliary conditioning is removed, leaving the post-trained base model. Varying λ controls how strongly the auxiliary conditioning changes the prediction relative to this base predictor.

Guidance strength γ. The auxiliary-conditioning control can be combined with inference-time guidance. A common form of guidance extrapolates the generator prediction away from a guiding prediction v_G, which may come, for example, from an unconditional prediction (classifier-free guidance Ho & Salimans (2022)) or a weaker model (autoguidance Karras et al. (2024)). The guided prediction is then

$$v_{\lambda,\gamma} = v_\lambda + \gamma \left(v_\lambda - v_G\right) . \tag{7}$$

4 Experiments

We evaluate our method primarily on class-conditional ImageNet-256 generation. We first use SiT as a controlled testbed to study the effect of our post-training with and without inference-time guidance. We then evaluate transfer to RAEv2 and analyze the role of competition.

4.1 SiT on ImageNet

Setup. We use SiT as our main testbed because many recent flow-matching methods build on it, allowing controlled comparisons within a common model family.

Our main model starts from the official pretrained SiT-XL/2 checkpoint and is post-trained with the WTA objective for 10 epochs with $K = 8$. For guidance, we instantiate Eq. 7 with AutoGuidance (AG) (Karras et al., 2024): the guiding prediction v_G comes from a smaller SiT-B/2, also post-trained with our method using $K = 8$ and run auxiliary-off ($\lambda = 0$). The pretrained SiT baseline uses the same AG formulation, with pretrained SiT-XL/2 and SiT-B/2 as the generator and guiding model.

To separate differences in the learned generators from differences in sampling heuristics, the controlled SiT comparisons, including Table 1, use a 50-step Euler ODE solver with uniformly spaced timesteps. Existing models are re-evaluated under this recipe where their original setup differs. The cross-family comparison in Table 2 instead uses each model's reference inference protocol (Appendix A.1.7).

We report $\mathrm{FD}_{r,6}$ (Yang et al., 2026b) as our primary metric; it aggregates Fréchet distances across six representation spaces, including the Inception space underlying FID. We also report Precision (sample fidelity) and Recall (coverage of the data distribution), with Inception Score (IS) as a supporting

quality metric and within-class DreamSim (Fu et al., 2023) distance as a supporting diversity metric. Operating-point curves use 10K generated samples; key numerical comparisons use 50K.

Varying the auxiliary conditioning scale λ. We first isolate the effect of our method by evaluating the post-trained generator without guidance ($\gamma = 0$) across auxiliary-conditioning scales λ. Starting from the pretrained SiT checkpoint, 10 epochs of our post-training yield clear gains, whereas continuing standard FM training does not improve generation.

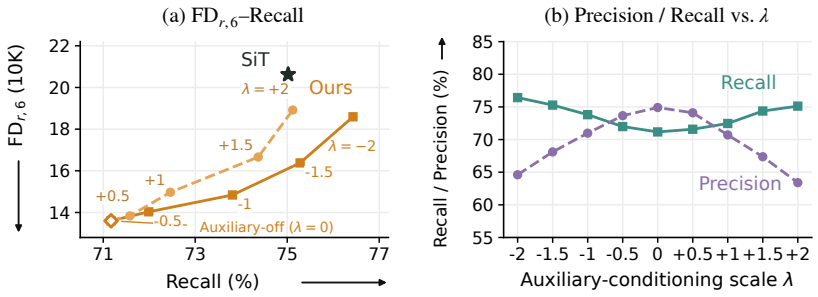


Figure 4: **Our method improves generation and enables smooth operating-point control.** (a) Without guidance, varying the auxiliary-conditioning scale λ lowers FD$_{r,6}$ relative to pretrained SiT, including at matched or higher Recall. (b) Increasing $|\lambda|$ from auxiliary-off ($\lambda = 0$) trades Precision for Recall.

As shown in Fig. 4(a), varying λ yields operating points with lower FD$_{r,6}$ than pretrained SiT, including at matched and even higher Recall. The auxiliary-conditioning scale λ also provides a smooth inference-time knob for controlling the generator's operating point. As shown in Fig. 4(b), the auxiliary-off setting ($\lambda = 0$) lies at high Precision, and increasing $|\lambda|$ moves the generator steadily toward higher Recall. Interestingly, we empirically observe similar overall trends for both signs of λ, with negative values generally yielding lower FD$_{r,6}$.

Under the standard 50K-sample evaluation without guidance ($\gamma = 0$), at matched Recall (66.91% for pretrained SiT and 66.83% for our model), we reduce FD$_{r,6}$ from 19.22 to 14.94 and FID from 11.17 to 7.22. The auxiliary-off setting ($\lambda = 0$) reaches an even lower FD$_{r,6}$ of 12.15 (FID 4.59). *Our method produces a stronger generator with smoothly controllable operating points, even without guidance.*

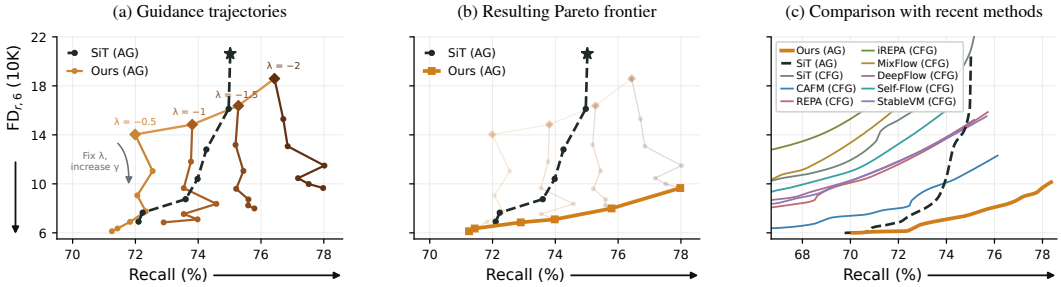


Figure 5: **Guidance expands our model's Pareto frontier toward broad coverage.** (a) AutoGuidance trajectories for pretrained SiT and our post-trained model in FD$_{r,6}$–Recall space. For each fixed auxiliary-conditioning scale λ, we sweep the guidance strength γ. (b) The resulting Pareto frontier obtained by combining trajectories across λ, compared with SiT. (c) Comparison with recent advances in flow matching over their respective guidance sweeps.

Combining with Guidance. We next apply guidance on top of these operating points: for each fixed λ, we sweep the guidance strength γ to trace a guided trajectory. Figure 5(a) shows that pretrained SiT and our model respond very differently to guidance. For pretrained SiT, increasing guidance lowers FD$_{r,6}$ but steadily reduces Recall. For our model, at a fixed λ, stronger guidance

Table 1: **Guided comparison under the standard 50K-sample evaluation.** (a) Performance at each method's lowest $\mathrm{FD}_{r,6}$. (b) Lowest $\mathrm{FD}_{r,6}$ at Recall $\geq 67\%$; light-gray Recall values mark methods whose highest attainable Recall is below 67%. The six raw Fréchet distances (including ADM FID) appear in smaller gray type; $\mathrm{FD}_{r,6}$ averages their normalized values.

	(a) Performance @ best $\mathrm{FD}_{r,6}$										**(b) Performance @ Recall $\geq 67\%$**									
	Multi-encoder Fréchet distances ↓										Multi-encoder Fréchet distances ↓									
Method	FID (ADM)	ConvNeXt	DINOv2	MAE	SigLIP	CLIP	$\mathrm{FD}_{r,6}$ ↓	Recall ↑	Precision ↑	IS ↑	FID (ADM)	ConvNeXt	DINOv2	MAE	SigLIP	CLIP	$\mathrm{FD}_{r,6}$ ↓	Recall ↑	Precision ↑	IS ↑
CAFM	3.30	85.58	51.49	0.13	**5.06**	58.31	4.86	57.1%	83.8%	342.7	3.84	225.73	135.82	0.24	10.91	106.67	9.84	67.3%	72.1%	181.9
REPA	4.79	132.70	52.50	0.19	5.69	60.93	5.65	49.7%	88.0%	388.8	6.42	302.84	171.30	0.37	14.43	136.53	13.13	67.2%	69.7%	162.5
MixFlow†	6.52	156.09	65.02	0.20	6.40	67.52	6.49	46.9%	88.6%	396.1	9.35	495.73	252.08	0.50	21.91	169.85	18.57	66.0%	68.0%	133.5
Self-Flow†	6.37	147.99	55.50	0.18	6.05	60.74	5.94	46.5%	89.8%	388.1	7.63	400.46	197.52	0.35	16.29	143.51	14.51	66.1%	69.4%	138.1
StableVM	5.47	140.84	52.32	0.18	5.75	60.29	5.72	48.0%	88.6%	391.1	7.45	360.07	188.04	0.38	15.46	142.35	14.14	67.9%	68.7%	147.3
DeepFlow	5.64	143.20	52.50	0.19	5.84	60.50	5.83	48.2%	88.8%	**398.6**	7.16	359.93	186.26	0.38	15.47	139.92	14.00	67.3%	68.7%	148.2
iREPA†	7.26	170.67	73.12	0.28	9.50	71.03	7.99	40.5%	**90.8%**	373.4	9.81	551.60	256.33	0.49	22.97	160.22	18.81	65.2%	67.9%	117.6
SiT (CFG)†	6.52	150.02	62.21	0.20	6.53	64.86	6.40	46.6%	89.2%	383.7	11.17	572.67	275.86	0.47	21.39	177.22	19.22	66.9%	66.3%	118.6
SiT (AG)†	**2.13**	85.92	50.46	**0.11**	5.26	52.29	4.53	**62.2%**	78.1%	236.9	11.17	572.67	275.86	0.47	21.39	177.22	19.22	66.9%	66.3%	118.6
Ours	2.29	**77.44**	**46.13**	0.13	5.24	**49.06**	**4.44**	60.4%	79.9%	258.2	**2.20**	**122.93**	**69.43**	**0.20**	**6.72**	**61.19**	**5.91**	67.4%	**73.0%**	**210.5**

moves the trajectory mostly downward: $\mathrm{FD}_{r,6}$ decreases substantially with little or no Recall loss, while at high-coverage operating points Recall can even increase.

Combining the guided trajectories across λ gives the Pareto frontier in Fig. 5(b), which extends beyond guided SiT toward both lower $\mathrm{FD}_{r,6}$ and higher Recall. Figure 5(c) compares this frontier with recent flow-matching methods across their full operating ranges: our method performs best overall, with its largest advantage in the broad-coverage regime.

We further analyze this synergy with guidance, particularly in the high-coverage regime. In addition to Recall, we measure within-class diversity using mean pairwise DreamSim (Fu et al., 2023) distance. Echoing the Recall trends above, stronger guidance reduces diversity for pretrained SiT but steadily increases it for our model at high-coverage settings (Fig. 6).

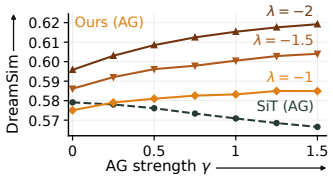


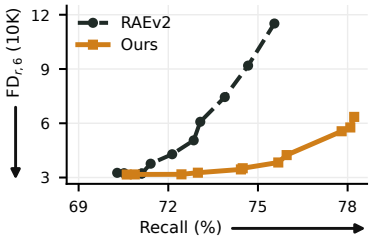
Figure 6: DreamSim distance as AG strength γ increases: SiT vs. Ours.

Finally, Table 1 reports the full set of metrics under the standard 50K-sample evaluation across methods. At the best overall operating point, our method achieves the lowest $\mathrm{FD}_{r,6}$ in the comparison, reaching 4.44 (Table 1a). The gap widens in the broad-coverage regime (Recall $\geq 67\%$, Table 1b). Competing methods, which reach their best $\mathrm{FD}_{r,6}$ only at much lower Recall, degrade sharply here, with higher $\mathrm{FD}_{r,6}$ and lower Precision and Inception Score. Our model instead stays near its best, reaching $\mathrm{FD}_{r,6} = 5.91$ with the highest Precision and Inception Score among the compared methods. *With our method, stronger guidance improves quality while largely preserving coverage, extending the quality–coverage frontier toward broad coverage.*

4.2 Extending to RAEv2

We test generalization on a second testbed, Representation Autoencoder (RAEv2) (Singh et al., 2026b). Whereas SiT operates in a VAE latent space, RAEv2 generates in the representation space of a pretrained visual encoder and achieves state-of-the-art ImageNet generation. RAEv2 has its own guidance mechanism, which we call RAEv2-guidance: in the notation of Sec. 3.2, the guiding prediction v_G comes from an intermediate head of the same model, with the same λ used for both predictions. We post-train the pretrained RAEv2 generator for 10 epochs with $K = 8$ competing predictions. Both models use RAEv2's native inference recipe, including its timestep shifting and guidance window, and we sweep the RAEv2-guidance strength for each.

As shown on the left of Fig. 7, our method substantially expands the $\mathrm{FD}_{r,6}$–Recall frontier beyond pretrained RAEv2, reaching lower $\mathrm{FD}_{r,6}$ and extending to broad-coverage operating points. At the



Method	Performance @ best $FD_{r,6}$				Performance @ Recall $\approx$ 70%			
	$FD_{r,6}\downarrow$	Recall (%) $\uparrow$	Precision (%) $\uparrow$	Inception Score $\uparrow$	$FD_{r,6}\downarrow$	Recall (%) $\uparrow$	Precision (%) $\uparrow$	Inception Score $\uparrow$
RAEv2	2.07	64.72	**79.39**	261.68	16.83	69.09	52.39	111.47
Ours	**2.02**	**65.71**	78.30	**265.82**	**3.18**	70.32	**71.25**	**182.39**

Figure 7: **Our method further improves RAEv2.** Left: $FD_{r,6}$–Recall frontiers (10K samples): our method extends beyond pretrained RAEv2, most at broad coverage. Right: On the 50K-sample evaluation, our method improves the best overall $FD_{r,6}$ with a larger gain near 70% Recall.

best overall point, $FD_{r,6}$ improves from 2.07 to 2.02; the gap is much larger near 70% Recall, where our model reaches $FD_{r,6} = 3.18$ at 70.32% Recall, versus 16.83 at 69.09% for RAEv2. At these operating points, our method also raises Precision from 52.39% to 71.25% and Inception Score from 111.47 to 182.39, as shown on the right of Fig. 7. *Our method generalizes across testbeds, improving even the already strong RAEv2 generator, with the largest gains at broad coverage.*

4.3 Analyzing the Role of Competition

We next analyze how our post-training reshapes the base model itself, obtained by removing the auxiliary conditioning ($\lambda = 0$), and what this reveals about the role of competition. We first compare base models across generator families, then use SiT to study how their behavior changes with K, including with fewer sampling steps and across λ.

Table 2: **Base model evaluation.** For each generator, we compare the pretrained model with its base model after our post-training ($K = 8$), evaluated at $\lambda = 0$. Each model is evaluated following the inference and evaluation protocol of its original work.

	SiT		JiT		RAE		RAEv2	
Metric	Pre-trained	Ours	Pre-trained	Ours	Pre-trained	Ours	Pre-trained	Ours
$FD_{r,6}\downarrow$	15.76	**10.59**	15.17	**14.15**	4.41	**3.64**	3.89	**3.52**
Recall (%) $\uparrow$	**67.65**	62.83	**66.46**	62.03	**63.80**	58.73	**62.08**	55.49
Precision (%) $\uparrow$	68.71	**75.70**	68.06	**72.05**	79.13	**83.53**	79.84	**85.76**
Inception Score $\uparrow$	131.86	**178.47**	151.49	**168.01**	240.36	**269.29**	226.72	**278.71**

Removing the auxiliary conditioning. Using $\lambda = 0$ removes the auxiliary conditioning $c_\phi(z)$ and the ability to predict multiple velocities, leaving a fine-tuned version of the original base model. How does this base model compare to its pretrained version? Table 2 evaluates this generator across representative generator families, including VAE-latent SiT (Ma et al., 2024), pixel-space JiT (Li & He, 2025), and representation-autoencoder generators RAE (Zheng et al., 2026) and RAEv2 (Singh et al., 2026b). Across all four models, the base model after our post-training shifts toward higher Precision and Inception Score, while also achieving lower $FD_{r,6}$. Interestingly, *our post-training shapes the original base model toward a higher-Precision regime.*

Scaling competition. The effect on the base model is closely tied to competition. As shown in Fig. 8(a), when $K = 1$, no competition occurs, and post-training leaves the base model nearly unchanged from pretrained SiT. As more predictions compete for each target, the base model shifts systematically toward higher Precision and lower $FD_{r,6}$. This trend is more pronounced with shorter sampling trajectories (Fig. 8(b)), with 8- and 4-step generation continuing to improve with larger K even as 50-step generation saturates.

This effect is not confined to the base model at $\lambda = 0$. Figure 9 shows how Precision and Recall change with the auxiliary-conditioning scale λ for various K values. With $K = 1$, both metrics

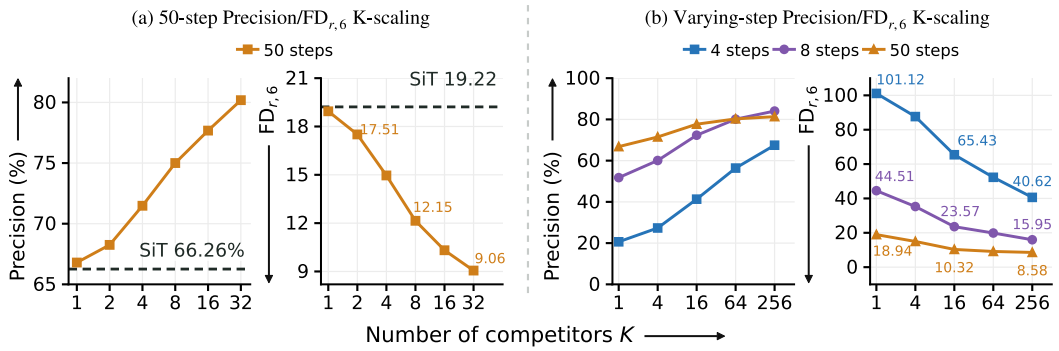


Figure 8: **Stronger competition increasingly reshapes the base model.** (a) Under 50-step sampling, increasing the number of competitors K steadily raises Precision while lowering $\mathrm{FD}_{r,6}$. (b) The same trend is amplified under shorter sampling trajectories: 4- and 8-step generation continues to improve at larger K, progressively narrowing the gap to the 50-step reference.

remain nearly flat as λ varies. As K increases, the response to auxiliary conditioning becomes progressively stronger, with larger changes in both metrics away from the $\lambda = 0$ base model.

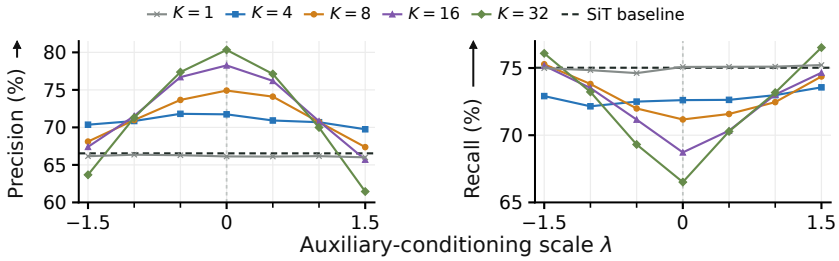


Figure 9: **Competition reshapes the generator across the full range of auxiliary-conditioning scales.** Precision (left) and Recall (right) as the auxiliary-conditioning scale λ varies for models post-trained with different numbers of competing predictions K. The dashed horizontal line denotes pretrained SiT baseline. All results use $\gamma = 0$.

5 Conclusion

A standard flow-matching model is trained toward a single operating point, leaving movement along the quality–coverage trade-off largely to inference-time guidance. In this work, we have shown that with WTA post-training, a single model instead holds two complementary behaviors: its base model operates at high fidelity, while auxiliary conditioning moves it toward high coverage. The auxiliary-conditioning scale λ interpolates smoothly between them, giving a control within the model itself that does not rely on guidance. Most notably, competition also changes how the generator responds to guidance. For pretrained models, stronger guidance improves quality at the cost of coverage; for WTA-post-trained models, it improves quality while largely preserving coverage, and at high-coverage operating points it can even increase Recall and within-class diversity.

A practical limitation of our method is its additional training cost: while it does not require training models from scratch, selecting the winner involves evaluating K candidate predictions for each training state, and this cost grows with K. Looking ahead, the auxiliary codes themselves merit closer study. Although WTA training only asks predictions to compete for each training target, without requiring any code to play a consistent role across states or sampling trajectories, a fixed code works well at inference: strengthening the auxiliary conditioning progressively increases coverage, and this combines well with guidance. The auxiliary-conditioned predictions thus appear to learn more structure than the objective explicitly asks for. Understanding this structure better, and learning or using auxiliary conditions in more adaptive ways, could further expand the quality–coverage frontier.

Reproducibility Statement

Backbone-specific architectures, auxiliary conditioning, initialization, and post-training objectives and hyperparameters are detailed in Appendix A.1. Appendix A.2 specifies the distributions, model architecture, training, and inference for the one-dimensional toy. Appendices C.3 and C.2 document AutoGuidance configurations and auxiliary-code sampling policies, including the evaluation grids and sampling budgets. Training computation and measured costs are reported in Appendix E.1. These details complement the evaluation protocols and metric definitions in the main text.

AI Assistance

Generative AI tools assisted with software implementation, experiment execution and evaluation, and manuscript editing. The authors reviewed the AI-assisted code and take responsibility for the final manuscript and all reported results.

REFERENCES

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.

Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. Cfg++: Manifold-constrained classifier free guidance for diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 30824–30850, 2025.

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

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. DreamSim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, pp. 50742–50768, 2023.

Alexi Gladstone, Heng Ji, and Yilun Du. Explorative modeling: Unlocking a third pretraining axis and end-to-end generation. *arXiv preprint arXiv:2607.27372*, 2026.

Abner Guzmán-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in Neural Information Processing Systems*, 2012.

Yasaman Haghighi, Bastien van Delft, Mariam Hassan, and Alexandre Alahi. Layersync: Self-aligning intermediate layers. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 29918–29950, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/32eb183794ef5ef9a3ab1d40a3d2b303-Paper-Conference.pdf.

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

Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. Representation alignment for diffusion transformers without external components. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 18884–18915, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/201e691e8a0940291dc591eb0439a789-Paper-Conference.pdf.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *Advances in Neural Information Processing Systems*, 2024.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 2016.

Hui Li, Jiayue Lyu, Fu-Yun Wang, Kaihui Cheng, Siyu Zhu, and Jingdong Wang. Mixflow training: Alleviating exposure bias with slowed interpolation mixture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9095–9105, 2026.

Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

Ke Li, Shichong Peng, Tianhao Zhang, and Jitendra Malik. Multimodal image synthesis with conditional implicit maximum likelihood estimation. *International Journal of Computer Vision*, 128(10):2607–2628, 2020.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

Shanchuan Lin, Ceyuan Yang, Zhijie Lin, Hao Chen, and Haoqi Fan. Continuous adversarial flow models. In *European Conference on Computer Vision*, pp. 645–663. Springer, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 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 *European Conference on Computer Vision*, 2024.

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

Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D. Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Inkyu Shin, Chenglin Yang, and Liang-Chieh Chen. Deeply supervised flow-based generative models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16535–16544. IEEE, 2025.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 33807–33845, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3929a7785bd56f57edcff0152ab41289-Paper-Conference.pdf.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026b.

George Stoica, Vivek Ramanujan, Xiang Fan, Ali Farhadi, Ranjay Krishna, and Judy Hoffman. Contrastive flow matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1185–1194, October 2025.

George Stoica, Sayak Paul, Matthew Wallingford, Vivek Ramanujan, Abhay Nori, Winson Han, Ali Farhadi, Ranjay Krishna, and Judy Hoffman. Posterior augmented flow matching. In *European Conference on Computer Vision*, 2026.

Shikun Sun, Longhui Wei, Zhicai Wang, Zixuan Wang, Junliang Xing, Jia Jia, and Qi Tian. Inner classifier-free guidance and its taylor expansion for diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 4360–4376, 2024.

Ziqiao Wang, Wangbo Zhao, Yuhao Zhou, Zekai Li, Zhiyuan Liang, Mingjia Shi, Xuanlei Zhao, Pengfei Zhou, Kaipeng Zhang, Zhangyang Wang, et al. Repa works until it doesn't: Early-stopped, holistic alignment supercharges diffusion training. *arXiv preprint arXiv:2505.16792*, 2025.

Donglin Yang, Yongxing Zhang, Xin Yu, Liang Hou, Xin Tao, Pengfei Wan, Xiaojuan Qi, and Renjie Liao. Stable velocity: A variance perspective on flow matching. In *International Conference on Machine Learning*, 2026a.

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation fréchet loss for visual generation. *arXiv preprint arXiv:2604.28190*, 2026b.

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. *arXiv preprint arXiv:2410.06940*, 2024.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, volume 2026, pp. 35791–35820, 2026.

SUPPLEMENTARY MATERIAL

This supplement provides implementation and evaluation details, additional quantitative results, inference ablations, analyses of generated variation, and computational costs.

CONTENTS

A Implementation and Evaluation Protocols **14**

A.1 Backbone-Specific WTA Implementations 14

A.2 Implementation Details of the One-Dimensional Toy 18

A.3 Guidance Search and Evaluation 20

B Additional Quantitative Results **21**

B.1 Precision–Recall Evaluation 21

B.2 Additional SiT Auxiliary-Conditioning and Competition Results 22

B.3 Competition Levels under RAEv2-Guidance 26

C Inference Choices and Ablations **27**

C.1 Are the Gains Due to Additional Model Capacity? 27

C.2 Auxiliary-Code Sampling Policies 27

C.3 AutoGuidance Configuration 31

C.4 Combining with Classifier-Free Guidance 32

D Diversity and Structured Variation **33**

D.1 Response to AutoGuidance 33

D.2 Reusable Auxiliary-Code Signatures 34

D.3 Structured Variation Under Auxiliary Conditioning and Guidance 36

E Computational Cost **41**

E.1 Training Cost and Scaling with the Number of Candidates 41

A Implementation and Evaluation Protocols

A.1 Backbone-Specific WTA Implementations

We implement WTA by augmenting each pretrained generator with an auxiliary code-conditioning pathway. This section specifies the architecture, initialization, native regression loss, and post-training configuration for SiT, JiT, RAE, and RAEv2.

A.1.1 Shared training construction

We use the notation of Eqs. (4)–(5): x_0 denotes a clean training sample in the generator's input space, ϵ is the Gaussian noise used to construct x_t, and $z \sim \mathcal{N}(0, I_d)$ is an independent auxiliary code, with $d = 8$. We follow the WTA objective in Eq. (5), using each backbone's native regression loss described below. Each loss averages squared errors over the sample's spatial and channel dimensions. Both the shared generator parameters θ and auxiliary parameters ϕ are updated by this objective. The pretrained VAE in SiT and the stage-one representation autoencoders in RAE and RAEv2 remain frozen. JiT operates directly on pixels.

We write a two-layer code MLP of width H as

$$r_H(z) = W_2\,\mathrm{SiLU}(W_1 z + b_1) + b_2, \quad W_1 \in \mathbb{R}^{H\times d}, \quad W_2 \in \mathbb{R}^{H\times H}. \tag{8}$$

The complete auxiliary conditioner c_ϕ includes this embedding and its backbone-specific injection maps, described below. We load the pretrained generator weights and initialize the added pathway to have zero contribution to the forward computation.

A.1.2 SiT

Representation and architecture. SiT-XL/2 (Ma et al., 2024) operates on $4 \times 32 \times 32$ VAE latents for 256×256 images. We retain the pretrained VAE, sample its posterior during training, and apply the native latent scale 0.18215. The generator has 28 Transformer blocks of width $H = 1152$, 16 attention heads, and latent patch size 2. Let $q = e_t(t) + e_y(y)$ denote its native time–class conditioner.

Auxiliary modulation. A shared $8 \to 1152 \to 1152$ MLP produces $r_H(z)$. Each block ℓ has an independent auxiliary projection, added to its native AdaLN modulation before the components are applied:

$$m_\ell(q, z) = A_\ell(q) + \Delta m_\ell(z), \qquad \Delta m_\ell(z) = U_\ell\,\mathrm{SiLU}(r_H(z)) + v_\ell. \tag{9}$$

Here $U_\ell \in \mathbb{R}^{6H\times H}$, and m_ℓ contains the shift, scale, and residual gate for the attention and MLP sublayers. For example, the attention update is

$$h \leftarrow h + g_{\ell,a} \odot \mathrm{Attn}((1 + s_{\ell,a}) \odot \mathrm{LN}(h) + b_{\ell,a}), \tag{10}$$

with the modulation vectors broadcast over spatial tokens. The output layer similarly receives an auxiliary shift and scale through a $2H$-dimensional projection. The code MLP is shared across blocks; the modulation projections are block-specific.

Initialization. We initialize W_1 with independent Gaussian entries of standard deviation 0.02, set $b_1 = 0$, and set $W_2 = b_2 = 0$. The auxiliary modulation weights U_ℓ use Xavier-uniform initialization and their biases v_ℓ are zero. Consequently, $\Delta m_\ell(z) = 0$ initially, while the nonzero projection weights provide a gradient path to the code MLP's final layer.

A.1.3 JiT

Architecture and conditioning. JiT-H/16 (Li & He, 2025) predicts clean images in pixel space. It has 32 blocks of width $H = 1280$, 16 attention heads, 16×16 image patches, and a patch-embedding bottleneck of width 256. Its native conditioning consists of a global time–class vector $q = e_t(t) + e_y(y)$ and 32 class-context tokens. We embed the auxiliary code using $r_H : 8 \to 1280 \to 1280$ and a learned scalar gate g, defining $a(z) = g\, r_H(z)$. The same residual modifies both conditioning inputs:

$$q' = e_t(t) + e_y(y) + a(z), \qquad C'_j = e_y(y) + p_j + a(z), \quad j = 1, \ldots, 32, \tag{11}$$

where p_j is the native context-position embedding. The blocks and output layer use q' through their existing AdaLN modules. The context tokens are inserted immediately before the eleventh Transformer block and participate in subsequent self-attention with the image tokens. The auxiliary residual is shared over context positions.

Initialization. Both code-MLP weight matrices are initialized from a zero-mean Gaussian with standard deviation 0.02, and both biases are zero. We initialize $g = 0$, so the native conditioner and context tokens are unchanged at the start of post-training. The randomly initialized MLP supplies a nonzero derivative with respect to the gate; after the gate changes, gradients also update the MLP.

A.1.4 RAE

Representation and architecture. We retain the pretrained representation autoencoder (Zheng et al., 2026), using a DINOv2-with-registers Base encoder with 224×224 encoder input, a ViT-XL image decoder, and the native latent normalization statistics. The stage-two generator operates on $768 \times 16 \times 16$ latents. Its DDT consists of 28 backbone blocks of width $H_b = 1152$ and two wide-head blocks of width $H_h = 2048$, with patch size 1 and 16 attention heads in each stack. These two stacks belong to the generative Transformer; the stage-one image encoder and decoder remain frozen.

Auxiliary modulation and initialization. The same code z is mapped by two independently parameterized MLPs, $r_b : 8 \to 1152 \to 1152$ and $r_h : 8 \to 2048 \to 2048$. For a block of width H_s, where $s \in \{b, h\}$, we use

$$m_\ell(q_\ell, z) = A_\ell(q_\ell) + U_\ell \,\mathrm{SiLU}(r_s(z)) + v_\ell, \quad U_\ell \in \mathbb{R}^{6H_s \times H_s}, \tag{12}$$

where q_ℓ is the block's native conditioning input. The six components control attention and MLP shifts, scales, and gates, applied to the native RMS-normalized activations. The final layer also receives an auxiliary shift and scale from $r_h(z)$. Both MLPs use Gaussian first-layer weights with standard deviation 0.02, zero final-layer weights, and zero biases in both layers. The block-specific auxiliary projections use Xavier-uniform weights and zero biases, following the zero-contribution initialization described for SiT.

A.1.5 RAEv2

Representation and architecture. RAEv2 uses the pretrained DINOv3-L representation with features from layers $11, 13, 15, 17, 19, 21, 23$, a ViT-XL image decoder, and the native normalization statistics. The stage-two input has shape $1024 \times 16 \times 16$. The generator has 28 backbone blocks of width 1440 with 20 attention heads, followed by two wide-head blocks of width 2048 with 16 heads. The backbone processes image tokens together with four timestep tokens and eight class-context tokens.

Shared context-token residual. We map the code through a shared MLP $r : 8 \to 1440 \to 1440$ and broadcast its output over the eight native class-context tokens:

$$C'_j = C_j + r(z), \qquad j = 1, \ldots, 8. \tag{13}$$

The backbone blocks use RMSNorm, self-attention, and MLPs without adaptive modulation. The wide head retains its native modulation: if h_i is an encoded image-token feature, its conditioning vector is

$$q_i = P(\mathrm{SiLU}(e_t(t) + h_i)) \,. \tag{14}$$

Each wide-head block maps q_i to attention and MLP shifts, scales, and gates; the final layer maps it to a shift and scale. Thus the code enters directly through class-context tokens and influences the native head modulation through the encoded image features. The code MLP's first layer has Gaussian weights of standard deviation 0.02 and zero bias; its final weight and bias are zero.

A.1.6 Native losses and post-training settings

We post-train all four generators on the ImageNet-1K training set at 256×256 image resolution.

Regression targets and time sampling. SiT and RAE use velocity prediction along $x_t = (1-t)\epsilon + tx_0$, with target $u_t = x_0 - \epsilon$. SiT samples t uniformly on $[0, 1]$. For RAE, let $s = \mathrm{sigmoid}(\eta)$ with $\eta \sim \mathcal{N}(0, 1)$; the native time sampler applies $t = \rho s/[1 + (\rho - 1)s]$, with $\rho = \sqrt{(768 \cdot 16^2)/4096} = \sqrt{48}$. JiT uses the same interpolation direction and samples $t = \mathrm{sigmoid}(\eta)$ with $\eta \sim \mathcal{N}(-0.8, 0.8^2)$. Its clean-image prediction $\widehat{x}_0$ is converted to a velocity before scoring:

$$\ell_{\mathrm{JiT}} = \mathrm{mean}\left[\left(\frac{\widehat{x}_0 - x_t}{\max(1-t, 0.05)} - \frac{x_0 - x_t}{\max(1-t, 0.05)}\right)^2\right]. \tag{15}$$

RAEv2 uses its native reversed time convention, $x_t = (1-t)x_0 + t\epsilon$, with a shifted logit-normal time sampler having $\eta \sim \mathcal{N}(0, 1)$ and $\rho = 8$. Its clean-latent predictions are converted to velocities using $(x_t - \widehat{x}_0)/\max(t, 0.05)$. We retain the native internal-guidance base head, read from the eighth backbone block. The regression loss is

$$\ell_{\mathrm{RAEv2}} = \ell_{\mathrm{main}} + \ell_{\mathrm{base}}, \tag{16}$$

where both terms are velocity MSEs with coefficient one. Their sum is used as the regression loss in the WTA objective.

Table 3: WTA post-training configurations. Batch size counts training images per optimizer update. The post-training duration follows the main-paper configuration. All four models use code dimension $d = 8$, weight decay zero, and class dropout probability 0.1.

Setting	SiT	JiT	RAE	RAEv2
Effective batch size	256	1024	1024	1024
Gradient accumulation	2	2	1	32
Post-training epochs	10	10	10	10
Main learning rate	10^{-4}	2×10^{-4}	2×10^{-4}	5×10^{-5}
EMA decay	0.9999	0.9999	0.9995	0.9995
Gradient-norm limit	—	—	1	1
Random seed	0	0	42	42

Optimizers and schedules. SiT uses AdamW with moment coefficients $(0.9, 0.999)$ and a constant learning rate. JiT and RAE use AdamW with $(0.9, 0.95)$. JiT linearly warms up the learning rate over one epoch and then holds it constant; its base rate 5×10^{-5} is scaled by $B/256$, giving 2×10^{-4} at batch size 1024. It maintains a second EMA with decay 0.9996, while inference uses the 0.9999 EMA. RAE uses a constant learning rate of 2×10^{-4} over its ten-epoch post-training interval.

RAEv2 retains GMuon for eligible hidden weight matrices and AdamW for embeddings, biases, normalization parameters, output layers, and the auxiliary MLP. GMuon uses momentum 0.95 with Nesterov updates; AdamW uses $(0.9, 0.95)$. The main learning rate stays at 5×10^{-5} for the first half epoch, then decays linearly to 10^{-5} at epoch ten. The auxiliary MLP has a $20\times$ learning-rate multiplier and its own 500-update linear warmup from zero toward 10^{-3}, followed by the main schedule's decay factor. We retain each backbone's native image preprocessing and representation normalization, and initialize the generator from its pretrained weights.

Numerical precision. SiT uses single-precision training with TF32 enabled; JiT uses BF16 autocast. RAE computes the WTA loss in FP32 to preserve small prediction differences near initialization. RAEv2 uses BF16 autocast and gradient accumulation to control its activation-memory requirement.

Auxiliary-off inference. We disable the complete auxiliary contribution while retaining the post-trained shared generator. For the AdaLN implementations this bypasses the added modulation projections, including their learned biases; for context conditioning it bypasses the added code residual. This realizes the auxiliary-off predictor in Eq. (6). Setting the input code itself to zero is insufficient because the learned conditioner can have a nonzero output at $z = 0$.

A.1.7 Reference inference and evaluation protocols

Table 2 compares each pretrained generator with its $K = 8$ post-trained counterpart under that model's reference inference protocol. Each entry uses 50K generated images without inference-time guidance; our models use auxiliary-off inference as defined above. The sampler and class-label policy are matched within each model family, but differ across families:

- **SiT:** Euler SDE sampling with 250 steps, a linear diffusion schedule with diffusion norm 1, and a final mean step of size 0.04. Class labels are sampled uniformly at random from the 1,000 ImageNet classes.
- **JiT:** Heun ODE sampling with 50 steps (99 model evaluations). Labels are class-balanced, with 50 images per class.
- **RAE:** The native Euler ODE configuration with the step-count argument set to 50 and time shift $\sqrt{48}$. The implementation constructs a grid of 50 time points. Labels are class-balanced, with 50 images per class.
- **RAEv2:** Euler ODE sampling with 50 steps and time shift 8. Labels are class-balanced, with 50 images per class. The model's internal guidance scale is set to 1, recovering its unguided prediction.

In particular, the SiT entries in Table 2 use SDE sampling with 250 steps, whereas Table 1 uses the controlled 50-step Euler ODE recipe and reports guidance-selected results. Their numbers therefore describe different inference configurations, even when evaluated from the same generator. Random labels and class-balanced labels both use a uniform class prior, but only the latter fixes the number of generated images per class.

Metric evaluation. Reference inference protocols specify how images are generated; they do not imply that every reported metric uses a backbone-specific evaluator. Precision and Recall in Table 2 use the ADM evaluation suite (Dhariwal & Nichol, 2021), with pooled Inception features and the released `VIRTUAL_imagenet256_labeled.npz` reference. The implementation uses `classify_image_graph_def.pb` and the nearest-neighbor manifold estimator with three neighbors (Kynkäänniemi et al., 2019). The SiT, RAE, and RAEv2 Inception Scores use the same ADM suite, with 5,000 images per split. JiT's Inception Score instead retains its native evaluator for both the pretrained and post-trained rows. These IS implementations should not be treated as interchangeable across families. The ADM implementation uses image and softmax batch sizes of 64 and 512, respectively.

Six-representation Fréchet distance. We compute $\mathrm{FD}_{r,6}$ on the same generated images using the released FD-Loss implementation and ImageNet reference statistics of Yang et al. (2026b), pinned to commit `5c03b811`. Let d_j be the unnormalized Fréchet distance in representation j and a_j its released training–validation reference distance. We report

$$\mathrm{FD}_{r,6} = \frac{1}{6}\sum_{j=1}^{6}\frac{d_j}{a_j}. \tag{17}$$

The six representations are Inception (ADM), ConvNeXt, DINOv2, MAE, SigLIP, and CLIP, with respective normalization constants 1.68, 56.87, 14.19, 0.04, 0.60, and 5.60. We use the released global-feature variants (the `cls` entries for DINOv2, MAE, SigLIP, and CLIP), rather than patch-average variants. The reference statistics are distributed in `jjiaweiyang/FD-Loss`, under `data/fid_stats`; the Inception component uses `guided_diffusion_stats.npz`. Normalization is applied only when forming $\mathrm{FD}_{r,6}$: reported individual FD components retain their raw values. The FD-Loss Inception component and a separately evaluated ADM FID use different implementations and are not assumed to be numerically identical.

A.2 Implementation Details of the One-Dimensional Toy

Source and target distributions. The toy in Fig. 2 transports a standard Gaussian source $\epsilon \sim \mathcal{N}(0,1)$ to a mixture of two compactly supported modes. Let $B \sim \mathrm{Beta}(3,3)$ and let M, independently of B, equal -0.75 with probability 0.25 and $+0.75$ with probability 0.75. We generate target samples as $x_0 = M + 0.6(2B-1)$. Equivalently, the target density is

$$p_{\mathrm{data}}(x) = 0.25\, q_{0.6}(x+0.75) + 0.75\, q_{0.6}(x-0.75), \qquad q_w(s) = \frac{15}{16w}\left(1-\frac{s^2}{w^2}\right)^2 \mathbf{1}_{\{|s|\le w\}}. \tag{18}$$

The two supports are $[-1.35, -0.15]$ and $[0.15, 1.35]$, with zero target density in the intervening gap. Training draws fresh source and target samples independently at each update.

Velocity predictor and auxiliary conditioning. The scalar velocity predictor uses three fully connected hidden layers of width 128, each followed by SiLU, and a linear scalar output layer. Its input is the 18-dimensional vector $[x_t, t, \{\sin(j\pi t), \cos(j\pi t)\}_{j=1}^{8}]$. An auxiliary code $z \sim \mathcal{N}(0, I_8)$ passes through a separate $8 \to 128 \to 128$ MLP with SiLU after both linear layers. For each predictor hidden layer, a separate linear map projects this code representation to 256 values, split into a scale $s_\ell(z)$ and a shift $b_\ell(z)$ of dimension 128 each. If a_ℓ is the hidden layer's pre-activation, conditioning gives

$$h_\ell = \mathrm{SiLU}(a_\ell \odot [1 + \lambda s_\ell(z)] + \lambda b_\ell(z)). \tag{19}$$

All scale-and-shift projection weights and biases are initialized to zero, so enabling the auxiliary pathway initially preserves the pretrained velocity field.

Flow-matching pretraining. We sample $t \sim \mathcal{U}[0,1]$ and form $x_t = (1-t)\epsilon + t x_0$, with velocity target $u = x_0 - \epsilon$. The predictor is trained with the mean squared error $\mathbb{E}[(v_\theta(x_t,t)-u)^2]$, with auxiliary conditioning disabled. We use Adam with batch size 1,024, default moment coefficients $(0.9, 0.999)$, and gradient-norm clipping at 10. The learning rate is 2×10^{-4} for the first 20,000 updates, 5×10^{-5} for the next 10,000, 10^{-5} for the next 10,000, and 5×10^{-6} thereafter. Training reaches the convergence criterion after 50,000 updates. Convergence is monitored using held-out error against the conditional-mean velocity field computed by numerical quadrature, together with a fine-step sampling check. We retain the 45,000-update checkpoint, which has the lowest held-out velocity error among the final four checkpoints saved at 5,000-update intervals, as both the FM baseline and the initialization for WTA post-training.

WTA post-training. For each freshly sampled (x_t, t, u), we draw $K = 8$ independent codes $z_k \sim \mathcal{N}(0, I_8)$ and compute eight predictions using the shared predictor with $\lambda = 1$. We minimize the batch average of $\min_{k\in\{1,\dots,8\}}(v_\theta(x_t,t;c_\phi(z_k))-u)^2$. Only the minimizing candidate contributes gradients for each training example; both the shared predictor and auxiliary conditioning parameters are updated. Post-training uses a fresh Adam optimizer for 40,000 updates, with learning rate 5×10^{-5} for the first 20,000 updates and 10^{-5} for the remaining 20,000. The batch size, Adam moment coefficients, and gradient clipping are the same as in FM pretraining.

Inference. We generate $8{,}192$ samples per setting from standard Gaussian initial states, reusing the same initial states across FM, auxiliary-on, and auxiliary-off inference. Sampling uses four explicit Euler updates from $t = 0$ to $t = 1$, with $t_i = i/4$ and step size $1/4$:

$$x^{(i+1)} = x^{(i)} + \tfrac{1}{4} v_\theta(x^{(i)}, t_i; c_\phi(z)), \qquad i = 0, 1, 2, 3. \tag{20}$$

Auxiliary-on inference uses $\lambda = 1$ and draws one independent Gaussian code per trajectory, held fixed across all four updates. Auxiliary-off inference uses the same post-trained predictor with $\lambda = 0$, bypassing the code-conditioning pathway. Standard FM sampling uses the pretrained predictor with conditioning disabled.

A.3 Guidance Search and Evaluation

We use a coarse-to-fine search with a final local resolution of 0.05 to select the guided configurations in Table 1. For classifier-free guidance (CFG), we use $v_{\mathrm{u}} + s(v_{\mathrm{c}} - v_{\mathrm{u}})$, where $s = 1$ is unguided. For AutoGuidance (AG), we use $v_{\mathrm{s}} + \gamma(v_{\mathrm{s}} - v_{\mathrm{w}})$, where $\gamma = 0$ is unguided; thus, an implementation using an interpolation scale reports $1 + \gamma$.

Search ranges and refinement. Baseline CFG sweeps cover $s \in [1, 5]$. SiT AG starts with $\gamma \in [0, 2]$ and includes subsequent refinements and extensions through $\gamma = 2.5$. For our $K = 8$ model, adaptive joint exploration covers $\lambda \in [-2.5, 2.5]$ and $\gamma \in [0, 3]$ through measured pairs. The ranges describe the combined search coverage; the joint search evaluates selected pairs rather than every combination. Coarse screening uses spacing 0.25 and $10{,}000$ generated images per configuration. We refine promising quality basins and identified Recall boundaries locally with spacing 0.05, inspecting adjacent strengths to check the local trend. Existing protocol-matched measurements, including historical refinements and $50{,}000$-image evaluations, are retained.

Evaluation and selection. Final reported configurations are evaluated with $50{,}000$ images using 50 Euler updates and each model's native guidance implementation and decoder. For our method, the weak predictor's auxiliary pathway is disabled, while λ controls the strong predictor's auxiliary conditioning; each Gaussian code is fixed along its sampling trajectory. The selection objective is minimum $\mathrm{FD}_{r,6}$, either without a Recall constraint or subject to Recall $\geq 67\%$. The constraint is checked using the $50{,}000$-image evaluation, rather than the screening Recall. When no measured configuration qualifies, we report the highest confirmed Recall and mark it as below target. All metrics within a reported row come from the same generated sample set.

Table 4: **Guidance settings reported in Table 1.** CFG entries give s; AG entries give the additional strength γ. For our $K = 8$ model, entries give (λ, γ). A dagger denotes a below-target reference in the Recall-constrained block.

Model	Guidance	Best-quality setting	Recall-constrained setting
CAFM	CFG	1.90	1.05
REPA	CFG	2.05	1.05
MixFlow	CFG	2.60	$1.00^{\dagger}$
Self-Flow	CFG	1.95	$1.00^{\dagger}$
StableVM	CFG	2.00	1.00
DeepFlow	CFG	2.10	1.00
iREPA	CFG	2.10	$1.00^{\dagger}$
SiT	CFG	2.45	$1.00^{\dagger}$
SiT	AG	2.35	$0.00^{\dagger}$
Ours ($K = 8$)	AG	$(-0.05, 1.80)$	$(-1.25, 1.50)$

Numerical records. The accompanying machine-readable records retain the tested strengths, auxiliary-conditioning strengths where applicable, sample counts, full-precision metrics, and eligibility for the comparison, including candidates that were not selected. They also preserve the sample and model identities and evaluation provenance needed to trace each reported row.

B Additional Quantitative Results

B.1 Precision–Recall Evaluation

We also report Precision–Recall curves on ImageNet-256, comparing WTA with recent flow-matching methods across inference-time guidance settings. Precision and Recall characterize sample fidelity and distribution coverage, respectively (Kynkäänniemi et al., 2019). Each operating point is evaluated using 10K generated samples. Figure 10 provides an additional view of the fidelity–coverage trade-off: WTA maintains competitive Precision while extending to higher Recall, highlighting its advantage in high-coverage generation.

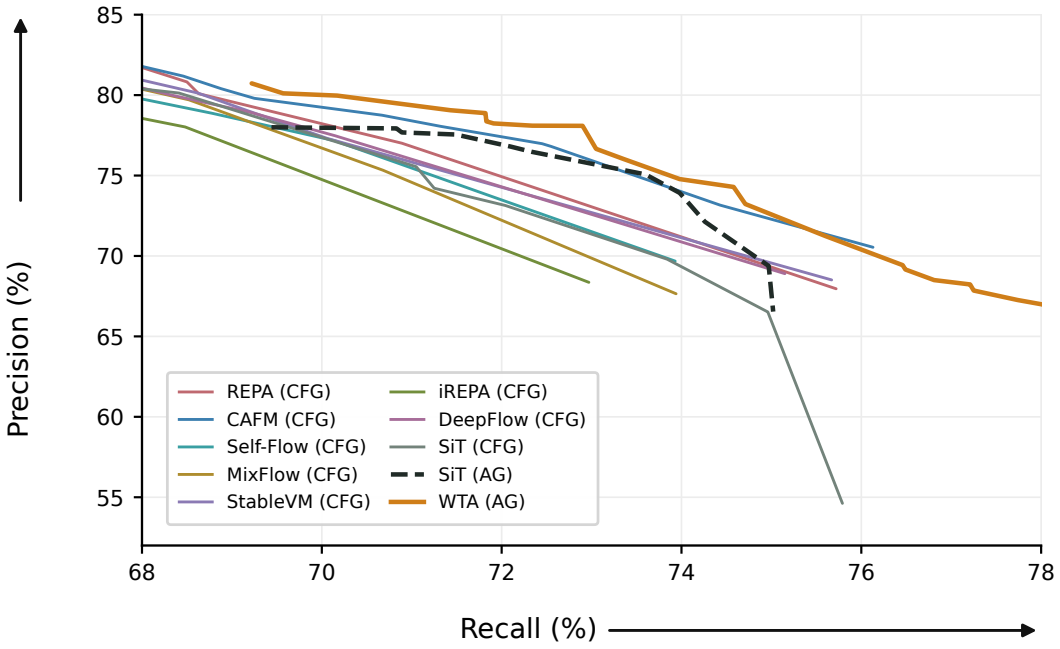


Figure 10: **Precision–Recall comparison in the high-coverage regime.** Results on ImageNet-256 using 10K generated samples per operating point and a 50-step Euler ODE solver. Curves summarize guidance sweeps; WTA also varies the auxiliary-conditioning scale. CFG and AG denote classifier-free guidance and AutoGuidance, respectively. Higher Precision and Recall are better. For display clarity, two WTA frontier points are omitted from the polyline; adjacent retained points are joined by straight segments as visual guides.

B.2 Additional SiT Auxiliary-Conditioning and Competition Results

We first examine the sign of the auxiliary-conditioning scale for the main $K = 8$ model, then extend the competition analysis in Sec. 4.3 with additional metrics and AutoGuidance frontiers. We also report standard flow-matching (FM) continuation controls. The no-competition capacity controls are reported in Sec. C.1.

B.2.1 Positive and Negative Auxiliary-Conditioning Scales

Figure 4 shows the response to auxiliary conditioning with $\gamma = 0$. Here we ask whether the frontier improvement with AutoGuidance also holds for positive scales, using the main $K = 8$ model. We compare the measured frontiers obtained with $\lambda > 0$ and $\lambda < 0$ against pretrained SiT with AutoGuidance.

Both signs extend the frontier toward higher coverage and lower $\mathrm{FD}_{r,6}$ than the pretrained reference in the high-coverage region. Negative scales yield a stronger high-coverage frontier, while positive scales also retain a clear improvement. Thus, the frontier gain is not restricted to negative auxiliary-conditioning scales.

We use the same post-trained SiT-XL/2 and SiT-B/2 checkpoints as in the main $K = 8$ experiment. The guiding model's auxiliary conditioning is disabled throughout. Each generated image uses one Gaussian auxiliary code, held fixed along its trajectory. Evaluation uses 10K random-label samples with the fixed global-index noise protocol and 50-step Euler sampling. For each sign, we retain nondominated measured operating points in lower $\mathrm{FD}_{r,6}$ and higher Recall across the sampled (λ, γ) settings.

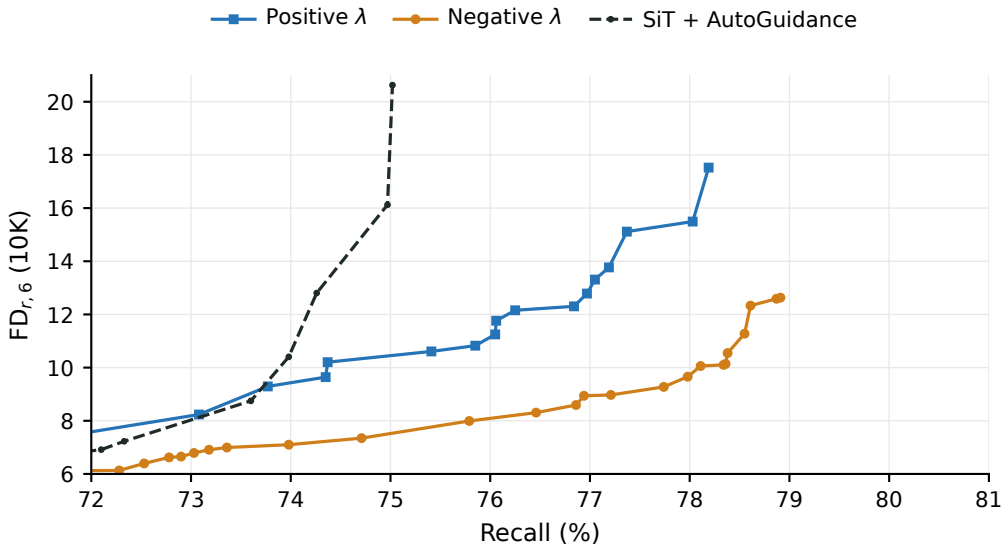


Figure 11: **Both signs of auxiliary conditioning improve the high-coverage frontier.** $K = 8$ AutoGuidance frontiers for positive and negative λ, compared with pretrained SiT AutoGuidance. Negative scales give a stronger measured high-coverage frontier. Each sign uses its measured sweep; $\lambda = 0$ is excluded from both. The historical sweep grids differ between signs. All results use 10K samples and 50-step Euler sampling. Markers denote measured frontier vertices; straight segments are visual guides.

B.2.2 Response to Auxiliary Conditioning

Figure 12 complements Fig. 9 with $\mathrm{FD}_{r,6}$, FID, and Inception Score as the auxiliary-conditioning scale λ varies with $\gamma = 0$. Every curve uses the same seven measured scales, from -1.5 to 1.5 in increments of 0.5. We use the same SiT-XL/2 checkpoint for each K throughout these comparisons, with $K \in \{1, 4, 8, 16\}$. Evaluation uses 10K random-label samples with a fixed global-index noise protocol and 50-step Euler sampling. One Gaussian auxiliary code is sampled per image and held fixed along its trajectory.

The $K = 1$ curves remain nearly flat and close to pretrained SiT. With competition, the post-trained base models at $\lambda = 0$ attain lower $\mathrm{FD}_{r,6}$ and FID, and higher Inception Score. Varying λ exposes a broader response for $K = 8$ and $K = 16$ than for $K = 4$. Together with the Precision–Recall curves, these results show that competition changes both the base model and the behaviors accessible through auxiliary conditioning.

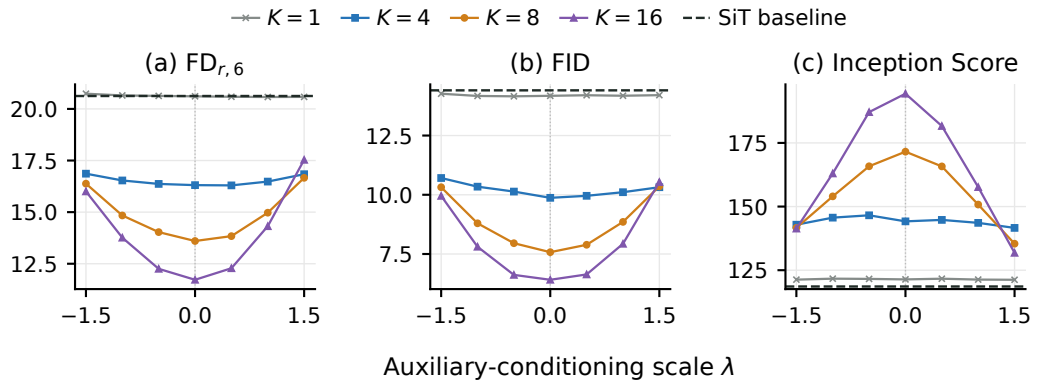


Figure 12: **Competition changes the response to auxiliary conditioning.** (a) $\mathrm{FD}_{r,6}$, (b) FID, and (c) Inception Score for $K \in \{1, 4, 8, 16\}$ with $\gamma = 0$. The dashed horizontal lines denote pretrained SiT. All curves use a common seven-point λ grid and 10K samples with 50-step Euler sampling. Lines connect measurements without smoothing.

B.2.3 Guidance Frontiers Across Competition Levels

We next combine auxiliary conditioning with AutoGuidance. For each fixed λ, varying the inference-time guidance strength γ traces a quality–coverage trajectory. We form the measured frontier by retaining points that are not dominated in lower $\mathrm{FD}_{r,6}$ and higher Recall. Each generator is paired with a SiT-B/2 guiding model post-trained with the same K; the guiding model's auxiliary conditioning is disabled throughout. The pretrained SiT reference uses the corresponding pretrained XL/2 and B/2 models. Sampling and evaluation follow the 10K Euler protocol above.

Figure 13 shows that the high-coverage benefit is not specific to $K = 8$. With $K = 4$, the frontier improves over SiT AutoGuidance in the overlapping high-coverage region, although the extension is smaller. With $K = 16$, the high-coverage improvement is retained, and the measured frontier reaches still higher Recall: 80.55%, compared with 78.91% for $K = 8$. These endpoints represent different quality–coverage trade-offs. The broad improvement across $K = 8$ and $K = 16$ supports robustness to the competition level used during post-training.

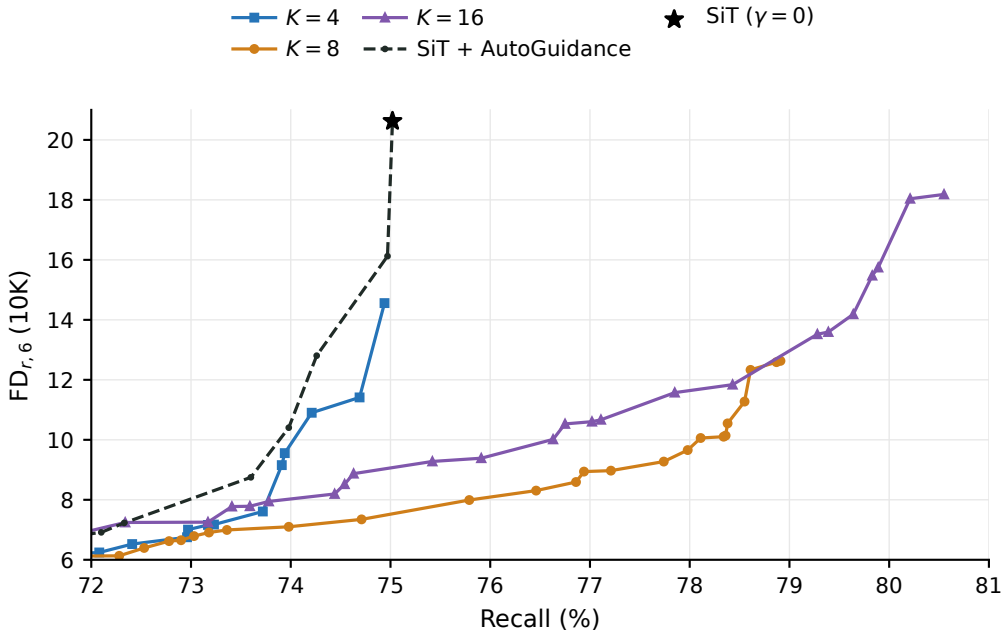


Figure 13: **The high-coverage frontier improvement persists across competition levels.** Measured AutoGuidance frontiers for $K = 4, 8, 16$, with pretrained SiT AutoGuidance and its $\gamma = 0$ point as references. All frontiers are computed from the complete measured pools before restricting the display to the common high-coverage region (Recall 72–81%). The pools contain 128, 297, and 133 settings for $K = 4, 8, 16$, respectively; the SiT reference contains 16 settings. They include local refinements and are not equal-sized search grids. All results use 10K samples and 50-step Euler sampling. Markers denote measured frontier vertices; straight segments are visual guides.

B.2.4 CONTINUED STANDARD FLOW-MATCHING TRAINING

The $K = 1$ experiment retains the auxiliary architecture while removing competition. We complement it with standard-FM continuation controls, which test additional training without the competitive objective. Table 5 reports historical SiT-XL/2 continuation results under 10K, 50-step Euler evaluation with $\gamma = 0$. FID remains close to the pretrained value after 50 additional training epochs. These controls therefore do not reproduce the change seen with competitive post-training.

Table 5: **Continued standard FM changes SiT-XL/2 only modestly.** 10K random-label samples, 50-step Euler sampling, $\gamma = 0$. † marks historical continuation records with incomplete reproduction metadata; their archived FID and IS are retained, without inferring other metrics.

Model	FID ↓	IS ↑
Pretrained SiT-XL/2	14.41	118.60
FM continuation, 50 epochs	14.31†	120.90†

Larger training batches. A separate SiT-B/2 control varies the global batch size among 256, 1024, and 8192, continuing standard FM for 20 data epochs from the same pretrained initialization (Table 6). This comparison matches data exposure rather than optimizer updates or computation: batches 1024 and 8192 use approximately $4\times$ and $32\times$ fewer updates than batch 256. Across batch sizes, the overall changes relative to the pretrained model remain limited. Increasing the batch size does not produce a consistent improvement from standard-FM continuation.

Table 6: **Training batch size in standard-FM continuation.** SiT-B/2 after 20 additional data epochs, evaluated using 10K random-label samples, 50-step Euler sampling, and $\gamma = 0$. Training exposure is matched across batch sizes; optimizer updates are not. The pretrained row is the common initialization before continuation.

Model / training batch size	FID ↓	IS ↑
Pretrained SiT-B/2	33.40	51.42
FM continuation, batch 256	32.84	53.45
FM continuation, batch 1024	31.76	56.18
FM continuation, batch 8192	32.81	53.20

B.3 Competition Levels under RAEv2-Guidance

We extend Fig. 7 by comparing competition levels $K \in \{8, 16\}$ under RAEv2-guidance (Fig. 14). The reference is pretrained RAEv2 with the same guidance mechanism.

The $K = 8$ model effectively expands the measured frontier toward lower $\mathrm{FD}_{r,6}$ and higher Recall relative to the pretrained reference. Increasing competition to $K = 16$ also improves the frontier and extends the measured coverage to 82.42% Recall. Thus, the frontier improvement persists at a larger competition level, which can reach further into the high-coverage region.

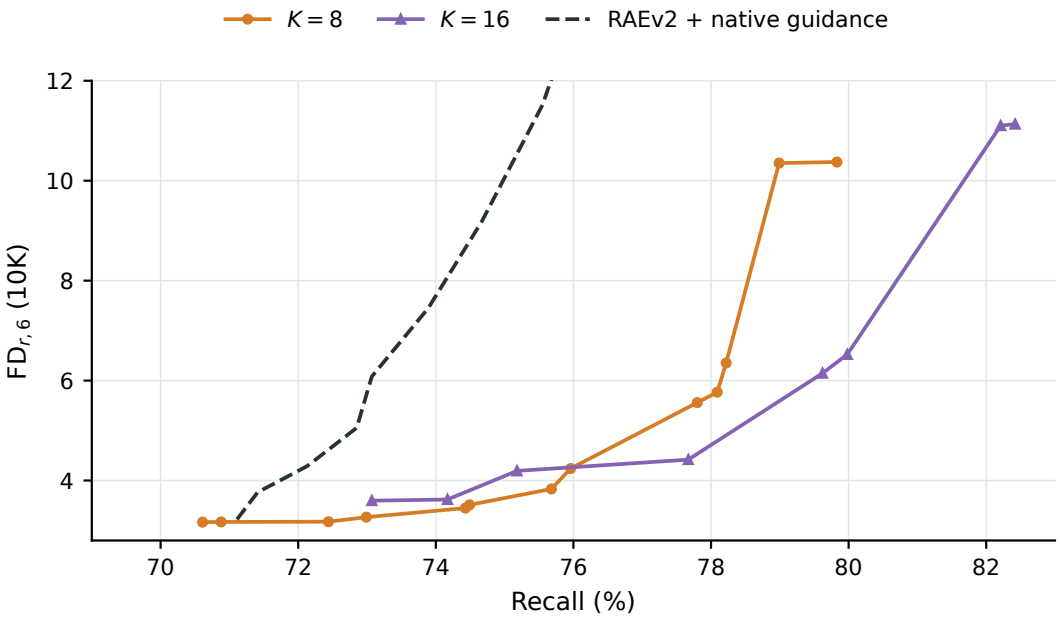


Figure 14: **The RAEv2 frontier improvement persists across competition levels.** Measured $\mathrm{FD}_{r,6}$–Recall frontiers for $K = 8, 16$ and pretrained RAEv2 with its native guidance. Each setting uses 10K class-balanced samples and native Euler-50 sampling with timestep shift 8. Markers denote nondominated measurements; straight segments are visual guides. Frontiers use the complete measured pools before display clipping. All post-trained frontier vertices are visible; two high-distance reference vertices lie above the vertical limit.

Inference and evaluation. We vary the auxiliary-conditioning scale λ at a fixed standard-normal code distribution, retaining RAEv2's timestep shift and guidance window $[0.1, 1.0]$. The full prediction and the intermediate guiding prediction come from the same model and use the same λ. The native interpolation strength s corresponds to $1+\gamma$ in Eq. 7; $s = 1$ applies no additional guidance. We use class-balanced labels and a common sampling seed. Metrics use the same fixed $\mathrm{FD}_{r,6}$ reference packet across all curves; the accompanying Precision, Recall, FID, and Inception Score records use the corrected ADM evaluator.

Search coverage. Each K has a complete common grid of 13 auxiliary scales and five native guidance strengths. With refinements, the pools contain 90 settings for $K = 8$, 65 for $K = 16$, and 29 reference settings. We compute nondominated frontiers using every measurement. Training budgets differ across the available checkpoints; full candidate records and checkpoint identities accompany the figure.

C Inference Choices and Ablations

C.1 Are the Gains Due to Additional Model Capacity?

The $K = 1$ control retains exactly the $K = 8$ augmented architecture but removes winner competition. Both strong models contain 902.27M stored parameter entries (including fixed positional embeddings), share the pretrained initialization, and are compared at matched post-training exposure. This controls added capacity and training exposure; candidate-forward training computation is not matched.

Post-trained base model. Section 4.3 and Fig. 8 show that the $K = 1$ base model at $\lambda = 0$ remains close to the pretrained model, whereas competition reshapes its behavior. The $K = 1$ response also remains nearly flat as λ varies (Fig. 9). Here we examine whether adding AutoGuidance changes this conclusion.

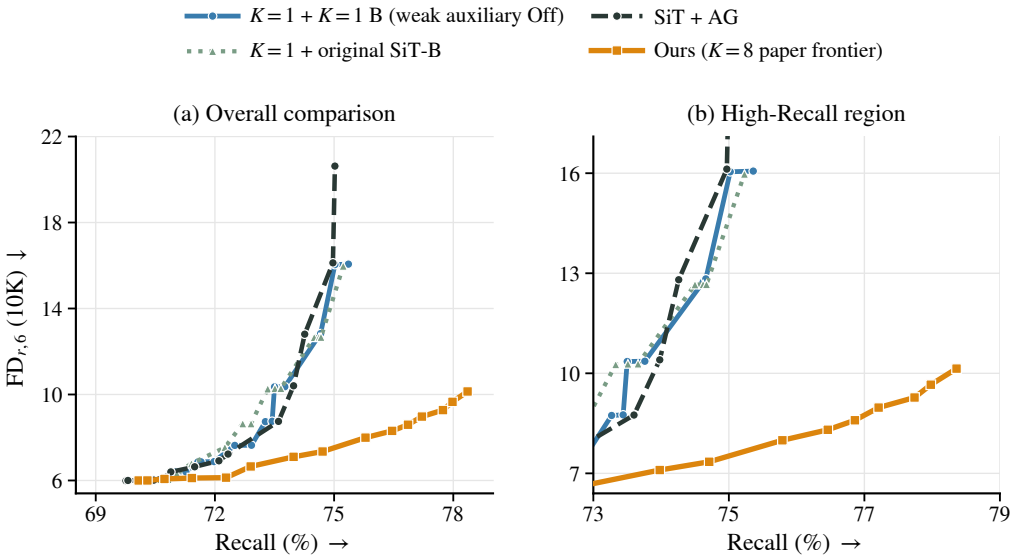


Figure 15: **Additional parameters without competition do not reproduce the high-Recall WTA frontier.** The primary control is $K = 1$ strong plus $K = 1$ B-size weak (auxiliary disabled); original SiT-B is an additional weak-model check. All curves use 10K samples and Euler50. Markers are measured points; straight segments connect frontier vertices. The $K = 1$ controls use the 63-setting grid below; the $K = 8$ paper reference additionally includes historical local refinements.

Primary control: $K = 1$ strong and $K = 1$ weak. We replace both competitive models in the main AutoGuidance recipe with their noncompetitive counterparts: SiT-XL/2 and SiT-B/2. The weak model's auxiliary conditioner is *disabled throughout*; it receives no auxiliary code, and its conditioning strength is not swept. The strong model uses a Gaussian auxiliary code fixed along each trajectory. We sweep its auxiliary-conditioning scale $\lambda \in \{-1.5, -1, -0.5, 0, 0.5, 1, 1.5\}$ and inference-time guidance strength $\gamma \in \{0, 0.25, \ldots, 2\}$, including the native $\lambda = 1$ setting.

Additional check with original SiT-B. We also pair the $K = 1$ strong model with original pretrained SiT-B using the same sweep. In Fig. 15, both $K = 1$ controls remain close to SiT AutoGuidance at high Recall and do not reproduce the lower $\mathrm{FD}_{r,6}$ of the $K = 8$ paper frontier. Together with the base-model evidence, this argues against additional parameter count alone being sufficient to explain the observed high-coverage improvement.

C.2 Auxiliary-Code Sampling Policies

We compare the policies introduced in Section 3.2: *fixed-code sampling* draws one Gaussian code per image and holds it throughout the trajectory; *resampled-code sampling* draws an independent code at

every Euler update. Code usage is separate from auxiliary-conditioning scale λ and guidance strength γ. At $\lambda = 0$, both policies reduce to the same post-trained base model, distinct from pretrained SiT, and share its evaluations. Neither selects a winning code at inference.

Controlled comparison. We use one SiT-XL/2 WTA model ($K = 8$, code dimension 8), 50 Euler updates, and 10K images per setting with the same global-index seed protocol for initial noises and class labels. At $\gamma = 0$, we sweep $\lambda \in \{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$; for AutoGuidance (AG), we use the same grid except $\lambda = 2$, with $\gamma \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5\}$. Here v_G in Equation (7) uses the same WTA SiT-B/2 guide ($K = 8$) for both policies, with auxiliary conditioning disabled throughout. This is a SiT/AG-specific setting. Pretrained SiT uses its pretrained SiT-B/2 guide. Reusing existing fixed-code and shared $\lambda = 0$ results gives 114 distinct evaluations: 50 nonzero-λ points per policy, seven shared $\lambda = 0$ points, and seven SiT points. Conclusions concern this measured grid, not an unrestricted guidance search.

Varying the auxiliary-conditioning scale ($\gamma = 0$). Figure 16 separates the response to the auxiliary-conditioning scale from the resulting quality–coverage trade-off. Fixed codes support stronger coverage, particularly on the negative-λ branch. At $\lambda = -1.5$, fixed-code sampling reaches 75.28% Recall with $\mathrm{FD}_{r,6} = 16.38$, compared with 75.02% and 20.63 for original SiT; at $\lambda = -2$, it reaches 76.43% with $\mathrm{FD}_{r,6} = 18.59$. Resampling also changes the operating point, and intermediate scales reduce $\mathrm{FD}_{r,6}$ relative to SiT, but with lower Recall. Its highest Recall at $\gamma = 0$ is 74.74%; moreover, its $\lambda = \pm 2$ endpoints have higher $\mathrm{FD}_{r,6}$ than SiT. Thus the resampled family at $\gamma = 0$ does not dominate the SiT anchor.

Combining with AutoGuidance. Figure 17 shows the paths obtained by fixing λ and increasing γ from zero, then extracts the nondominated observations over auxiliary-conditioning scale and guidance strength (lower $\mathrm{FD}_{r,6}$ and higher Recall are preferred). Resampling remains effective: its frontier improves on SiT+AG in the broad-coverage region. Fixed codes yield a further, larger improvement there. For an illustrative common quality ceiling, Table 7 selects the highest observed Recall subject to $\mathrm{FD}_{r,6} \leq 8$. Resampling improves Recall over SiT+AG by 1.40 percentage points; fixed codes add another 2.16 points. These are measured operating points under the same ceiling, not interpolated comparisons at exactly equal $\mathrm{FD}_{r,6}$.

The minimum $\mathrm{FD}_{r,6}$ values are comparable: 6.08 for the fixed-code family (including $\lambda = 0$) and 6.06 for resampling, at 69.22% and 70.16% Recall. Single-seed evaluations do not establish a reliable ordering of these small differences. Fixed codes are especially advantageous for broad coverage, rather than uniformly better. Positive scales also contribute to the resampled frontier; inspecting only negative scales is incomplete.

Table 7: **Coverage under a common quality ceiling.** Highest measured Recall with $\mathrm{FD}_{r,6} \leq 8$ within the 10K, 50-Euler-step grid in Figure 17. The achieved distances are shown explicitly; no interpolation or exact distance matching is used.

Generator / code policy	λ	γ	$\mathrm{FD}_{r,6} \downarrow$	Recall (%)↑
SiT + AG	—	1.25	7.65	72.23
Resampled-code + AG	1.50	1.50	7.97	73.63
Fixed-code + AG	−1.50	1.50	7.99	75.79

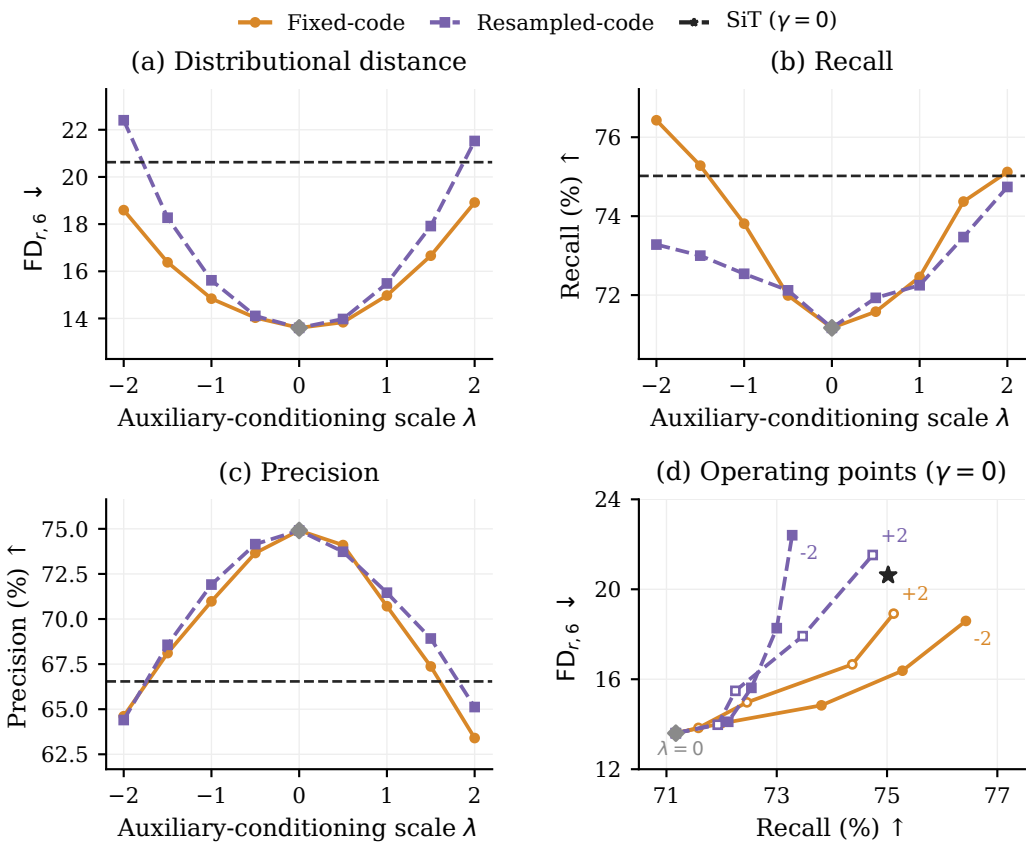


Figure 16: **Varying auxiliary conditioning under two sampling policies ($\gamma = 0$).** SiT-XL/2 WTA ($K = 8$), 10K images per setting, 50 Euler updates. Panels (a–c) show $FD_{r,6}$, Recall, and Precision versus auxiliary-conditioning scale; dashed black horizontal lines show pretrained SiT at $\gamma = 0$. Panel (d) shows the same observations in Recall–$FD_{r,6}$ space, with a black star for SiT. Gray diamonds denote the shared post-trained base model ($\lambda = 0$). In (d), filled and open policy markers denote negative and positive scales, respectively; endpoint labels give λ. Lines connect measured settings and do not imply evaluated intermediate points.

Interpretation. Training draws auxiliary codes for competition at local noisy states; it does not assign explicit semantics to individual codes or enforce trajectory-level consistency. Nevertheless, Section D.2 and Figure 21 shows recurring visual tendencies when the same code is reused across initial noises and classes, including color, background, and composition. That qualitative evidence uses a separate $K = 512$ checkpoint, rather than the $K = 8$ checkpoint evaluated here. Together, these observations suggest an intuition: *visual tendencies learned through local competition may be more fully expressed when the same code is used across sampling steps; per-step resampling may disrupt or weaken their expression.* This is a hypothesis consistent with the observed broad-coverage advantage, not a demonstrated causal explanation or a claim that each code represents a fixed semantic mode. It also does not require a code to have the same effect at every noisy state or to remain active throughout generation: the separate temporal intervention in Section D.2.1 finds that activation during the high-noise window preserves most signature identification accuracy. That study does not compare fixed and resampled quality–coverage frontiers. The results support using fixed codes by default while retaining resampling as an effective alternative under guidance.

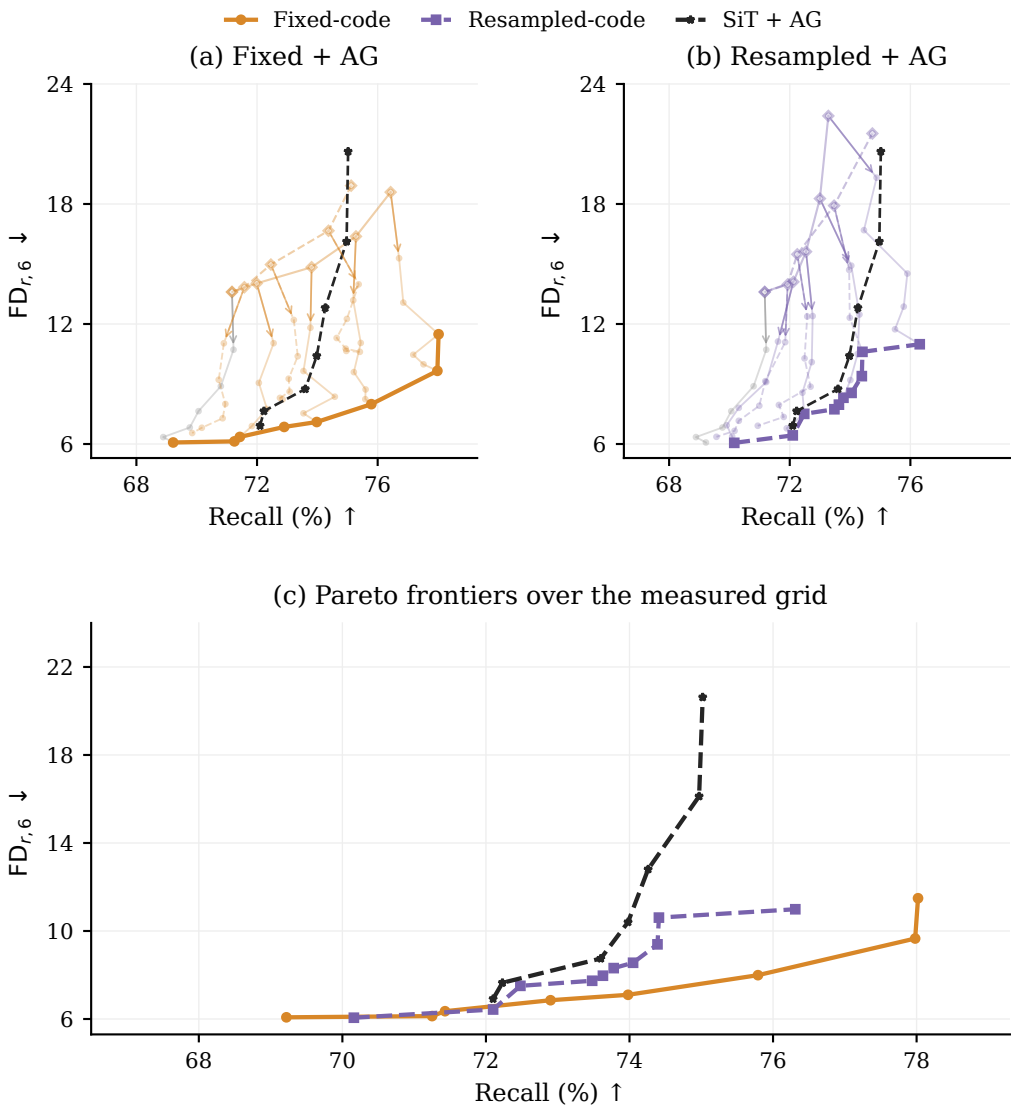


Figure 17: **Both code policies improve the guided frontier, with a larger broad-coverage gain from fixed codes.** The full 10K, 50-step grid described in Section C.2 is used. In (a,b), pale diamonds connect the $\gamma = 0$ operating points; pale paths vary γ at fixed λ, with arrows pointing toward increasing guidance. Solid and dashed pale paths denote nonpositive and positive λ, respectively; gray denotes the post-trained base model ($\lambda = 0$). Thick colored curves connect the observed Pareto points. Panel (c) compares these frontiers directly with original SiT+AG, using the same axis ranges. The $\lambda = 2$ points are evaluated only at $\gamma = 0$. Connecting segments are visual guides; no additional points are inferred along them.

C.3 AUTOGUIDANCE CONFIGURATION

We compare alternative guiding models and auxiliary-conditioning settings. The configuration used in our main experiments achieves the strongest high-coverage frontier among those evaluated (Fig. 18).

Compared configurations. We hold the WTA-post-trained SiT-XL/2 generator fixed ($K = 8$) and compare three guiding-model configurations: the original pretrained SiT-B/2; a WTA-post-trained SiT-B/2 ($K = 8$) with auxiliary conditioning enabled; and the same post-trained guiding model with auxiliary conditioning disabled, as used in the main experiments. For the enabled configuration, both models receive the same numerical auxiliary code and conditioning scale, $\lambda_w = \lambda$. For the default configuration, the guiding model uses $\lambda_w = 0$. This comparison examines both the choice of guiding model and whether it receives the auxiliary condition.

Evaluation. All three configurations use the same 24 combinations of generator conditioning scale $\lambda \in \{-1.5, -1, -0.5, 0.5, 1, 1.5\}$ and inference-time guidance strength $\gamma \in \{0.5, 1, 1.5, 2\}$, following Eq. 7. Each setting uses 10K random-label samples with the same global-index noise protocol and 50-step Euler sampling. An eight-dimensional Gaussian auxiliary code is held fixed along each sampling trajectory. Pretrained SiT-XL/2 guided by pretrained SiT-B/2 provides the SiT+AG reference.

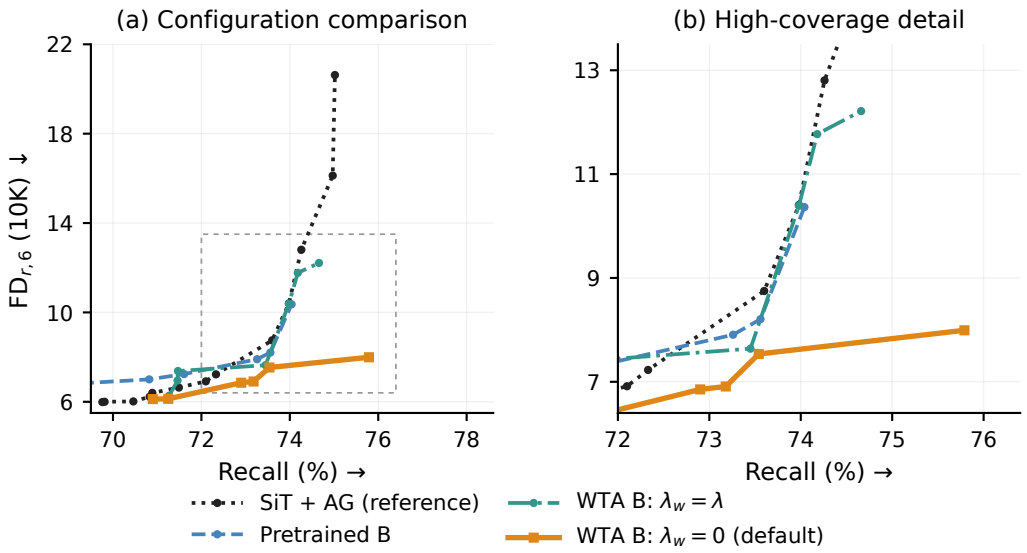


Figure 18: **AutoGuidance guiding-model configurations.** The three configurations share the same WTA-post-trained $K = 8$ generator and 24 (λ, γ) settings; only the guiding model or its auxiliary conditioning changes. The default uses WTA-post-trained SiT-B/2 with $\lambda_w = 0$. Curves connect measured Pareto points for lower $\mathrm{FD}_{r,6}$ and higher Recall, without smoothing. (a) Configuration comparison. (b) Detail of the boxed high-coverage region. SiT+AG is shown using its archived guidance sweep. All results use 10K samples and 50-step Euler sampling.

C.4 Combining with Classifier-Free Guidance

We evaluate whether the quality–coverage gains from WTA post-training also extend to classifier-free guidance (CFG). The guiding prediction is the class-unconditional output of the same model. The conditional and unconditional predictions use the same auxiliary code and conditioning scale λ, with the code fixed along each trajectory.

We compare post-trained SiT-XL/2 with $K \in \{8, 16\}$ against pretrained SiT-XL/2 with CFG, using 10K samples and 50-step Euler sampling. For each post-trained model, we jointly vary auxiliary-conditioning scale λ and inference-time guidance strength γ; the conventional CFG weight is $\omega = 1 + \gamma$.

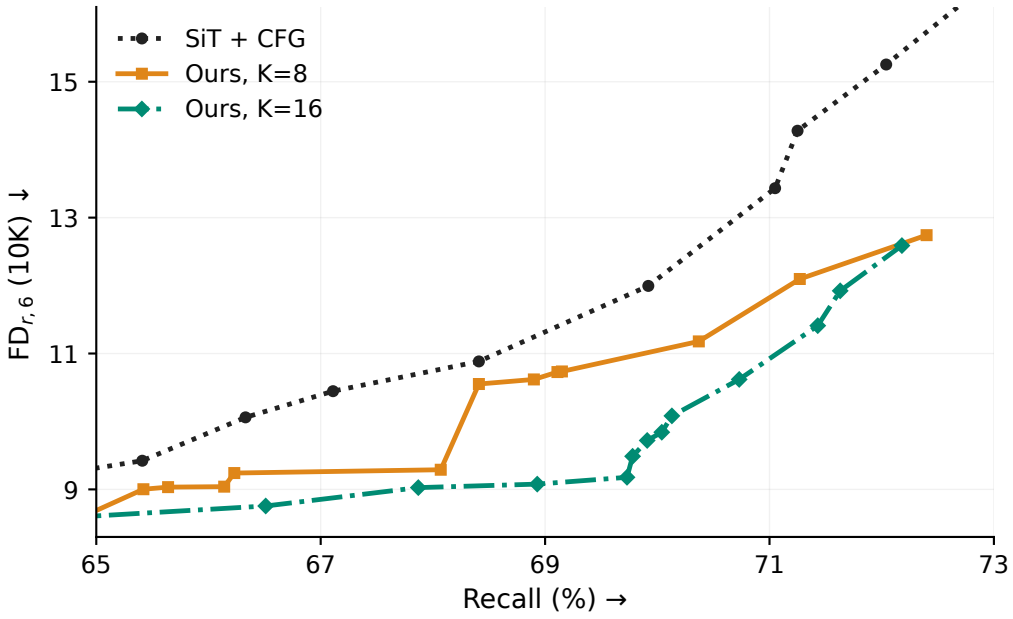


Figure 19: Quality–coverage tradeoffs with classifier-free guidance. Pretrained SiT+CFG is compared with post-trained models using $K = 8$ and $K = 16$. Curves connect measured Pareto points from joint (λ, γ) sweeps, using one fixed trained model per K. The search includes the completed common grid and available refinements. All results use 10K images and 50-step Euler sampling.

Both $K = 8$ and $K = 16$ improve the measured quality–coverage tradeoff in the displayed Recall range. These results show that the gains from WTA post-training can also be realized with classifier-free guidance.

D Diversity and Structured Variation

D.1 Response to AutoGuidance

We examine how within-class perceptual variation, coverage, and generative quality change with inference-time guidance under the configuration used in our main experiments.

Measurement protocol. We hold $\lambda \in \{0, -0.5, -1, -1.5\}$ fixed and vary inference-time guidance strength γ, using the $K = 8$ SiT-XL/2 strong predictor and $K = 8$ SiT-B/2 weak predictor with auxiliary conditioning disabled. Original pretrained SiT-XL/2 guided by SiT-B/2 provides the SiT+AG reference. Each setting uses 10K random-label images and 50-step Euler sampling. We measure within-class perceptual variation with DreamSim (Fu et al., 2023), using its original ensemble and cosine distance between normalized embeddings. For every requested class, we average over all unordered image pairs, then average these class means with equal class weights. Each setting contains all 1,000 classes (2–24 images per class), giving 50,020 pairs across 10,000 images. This is generated–generated variation within a requested class, not distance to real images, a semantic mode count, or an isolated fixed-noise code comparison.

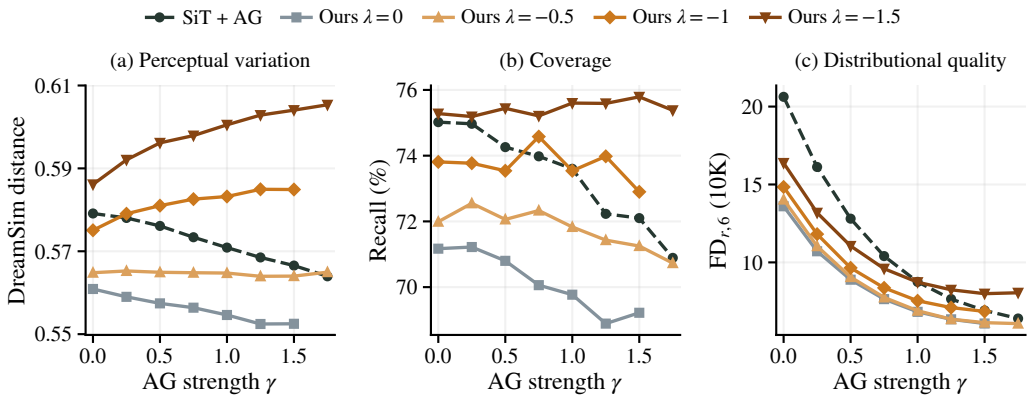


Figure 20: **Guidance response with auxiliary conditioning.** (a) Full-10K within-class DreamSim distance, (b) Recall, and (c) $\mathrm{FD}_{r,6}$ on the same generated archives. All 38 measured settings are retained, connected in increasing γ order without smoothing or frontier selection. All five paths include $\gamma \in \{0, 0.25, \ldots, 1.5\}$; SiT and Ours at $\lambda = -0.5, -1.5$ additionally include $\gamma = 1.75$. Higher DreamSim indicates greater perceptual variation, not necessarily better quality. Historical source qualifications are retained in the accompanying data.

Table 8: **A common guidance interval.** Measured changes from $\gamma = 0$ to 1.5, using the same 10K protocol as Figure 20. Arrows within cells indicate the two endpoints; higher Recall and lower $\mathrm{FD}_{r,6}$ are preferred.

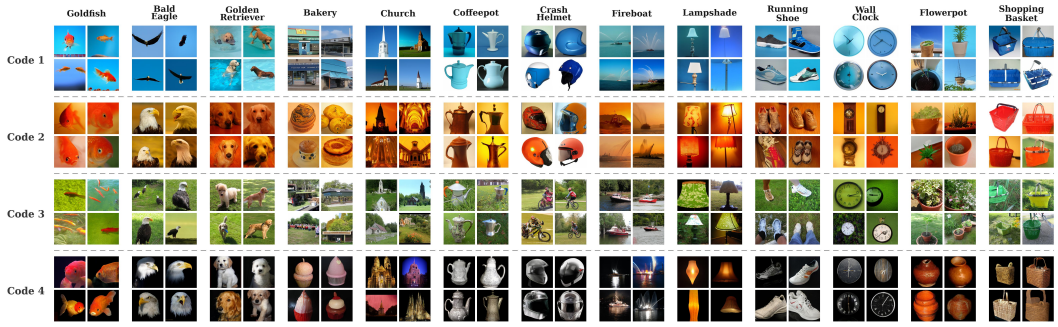
Model / strength	DreamSim	Recall (%) ↑	$\mathrm{FD}_{r,6}$ ↓
SiT + AG	0.5791 → 0.5665	75.02 → 72.10	20.63 → 6.91
Ours $\lambda = 0$	0.5609 → 0.5525	71.17 → 69.22	13.60 → 6.08
Ours $\lambda = -0.5$	0.5648 → 0.5640	71.99 → 71.25	14.03 → 6.13
Ours $\lambda = -1$	0.5751 → 0.5849	73.81 → 72.90	14.84 → 6.85
Ours $\lambda = -1.5$	0.5861 → 0.6041	75.28 → 75.79	16.38 → 7.99

Observed response. For original SiT, increasing guidance reduces within-class DreamSim and Recall while improving $\mathrm{FD}_{r,6}$. For ours, DreamSim decreases at $\lambda = 0$, remains approximately flat at $\lambda = -0.5$, and increases overall at $\lambda = -1$ and -1.5. At $\lambda = -1.5$, both DreamSim and Recall increase over $\gamma = 0$ to 1.5 while $\mathrm{FD}_{r,6}$ decreases (Table 8).

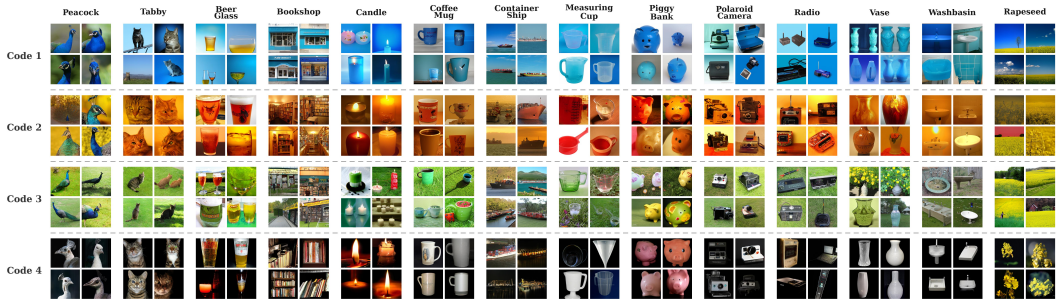
D.2 Reusable Auxiliary-Code Signatures

The experiments in this section use fixed-code sampling at the training-time auxiliary-conditioning scale ($\lambda = 1$), without inference-time guidance ($\gamma = 0$). The visualizations use the SiT-XL/2 $K = 512$ checkpoint.

The auxiliary codes are drawn fresh at every training step; nothing in training asks any code to mean anything. Yet we observe that a code leaves a recognizable mark: under fixed auxiliary-code sampling, samples generated with the same code share recurring appearance tendencies across initial noises and classes (Figure 21).



(a) Reusable visual signatures across ImageNet classes.



(b) Additional classes with the same auxiliary codes.

Figure 21: **Samples generated with the same auxiliary code share recurring appearance patterns across source noises and classes.** Each row fixes one auxiliary code z and each column one ImageNet class; the four selected samples at each row–column intersection vary only the initial noise ϵ. Both panels use the same SiT-XL/2 WTA checkpoint ($K = 512$), the same four auxiliary codes, and 50-step ODE sampling at $\lambda = 1$, $\gamma = 0$, holding the code fixed throughout each trajectory. Repeated tendencies in color, background, and composition appear both within and across classes.

D.2.1 Code Signatures Strengthen with Competition

We next quantify this observation: auxiliary codes carry recognizable visual signatures, which strengthen with competition K and do not form without it. Our test asks a simple question: looking only at generated samples, can we tell which code produced them? We freeze 64 codes for analysis and split 128 ImageNet classes into a reference half and a held-out half.

We first define the change a code makes: for a given class and source noise, we generate one sample with the code and one without, and compute the difference of their features in a pretrained visual encoder's space (e.g., DINOv2). We then subtract the average of this difference over all codes, leaving only what is specific to this code. Averaging these code-specific differences over the reference classes gives the code's signature. For each held-out sample, we compute its code-specific difference in the same way, and assign it to the code with the nearest signature. Chance is $1/64$, and identification succeeds only if a code's effect recurs on classes never used to build its signature.

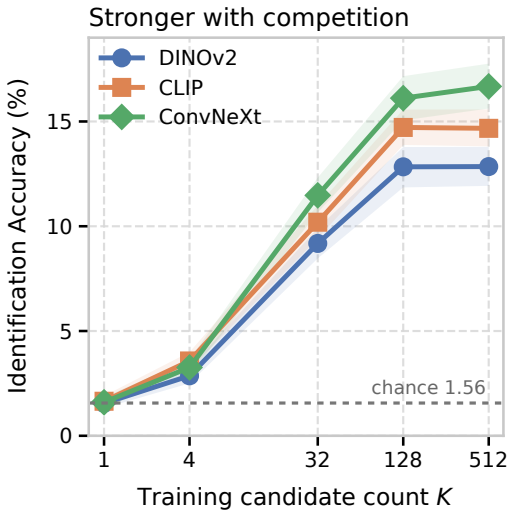


Figure 22: **Code signatures strengthen with competition.** Identification accuracy on held-out classes, over 64 frozen analysis codes; chance is 1.56%. The matched K sweep uses 128 classes and four source-noise seeds, with 50-step ODE sampling and fixed auxiliary codes at $\lambda = 1, \gamma = 0$. Shaded bands are 95% hierarchical class–seed bootstrap intervals.

Across DINOv2, CLIP, and ConvNeXt features, identification accuracy sits at chance for the $K = 1$ control and rises with competition over the measured range, reaching roughly ten times chance at $K = 512$ (Figure 22). The small differences between $K = 128$ and $K = 512$ indicate saturation rather than a strict monotone endpoint ordering.

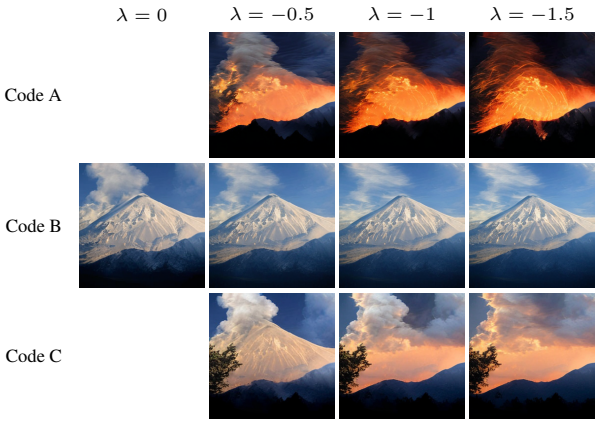
A separate temporal intervention also finds that the signatures are expressed mainly at high noise: activating a code only during the noisiest third of sampling preserves most of the identification accuracy of full-trajectory activation. That intervention uses 48 Euler updates, divided into three equal 16-update windows; it is distinct from the full-trajectory visualizations above.

D.3 Structured Variation Under Auxiliary Conditioning and Guidance

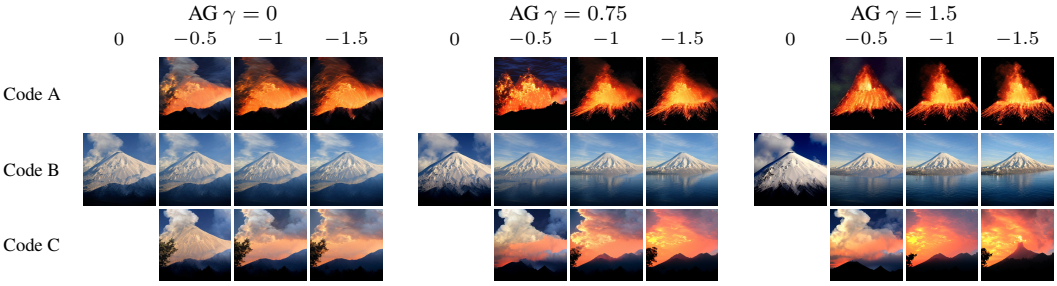
Each example fixes an ImageNet class and initial noise. Rows compare three auxiliary codes held fixed throughout sampling; code labels are local to each example. Columns vary $\lambda \in \{0, -0.5, -1, -1.5\}$ (Equation (6)). Panel (a) varies auxiliary conditioning at $\gamma = 0$; panel (b) preserves the same row and column order at AG strengths $\gamma = 0, 0.75, 1.5$. Within each AG block, the auxiliary-off image ($\lambda = 0$) is shared across codes and shown once. These selected examples illustrate changes in appearance and detail around recognizable visual structure, sometimes extending to composition or scene content. Guidance can sharpen details and accentuate code differences; neither quality nor diversity is guaranteed to change monotonically. All images are original, uncropped PNGs from the SiT-XL/2 $K = 8$ model with 50-step Euler sampling. AG uses the SiT-B/2 $K = 8$ auxiliary-off weak predictor with runtime scale $1 + \gamma$.

Volcano

(a) Auxiliary conditioning ($\gamma = 0$)

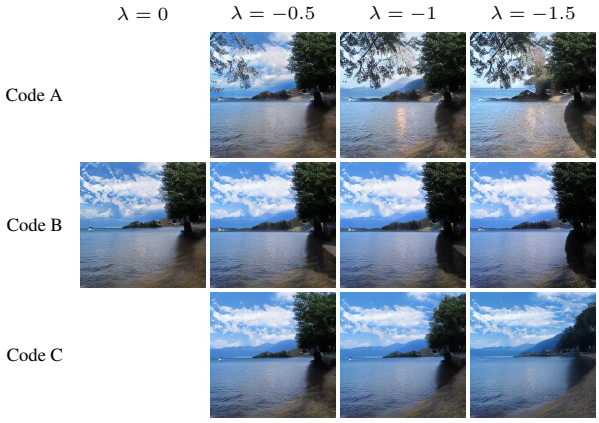


(b) Guidance across auxiliary scales

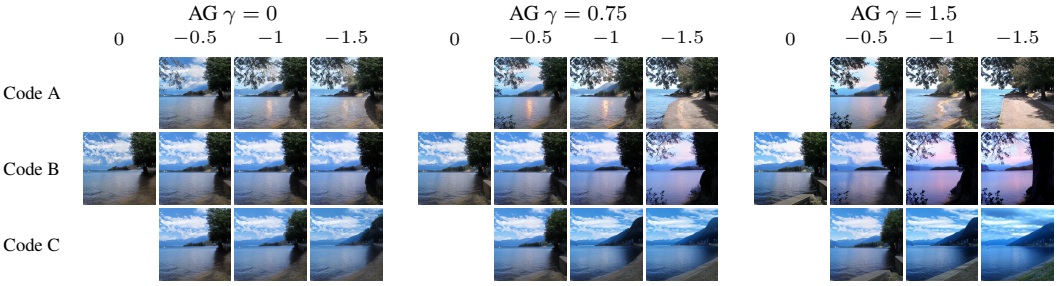


Lakeside

(a) Auxiliary conditioning ($\gamma = 0$)

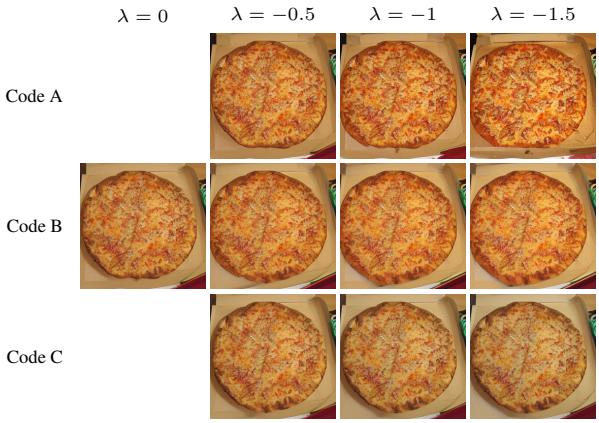


(b) Guidance across auxiliary scales

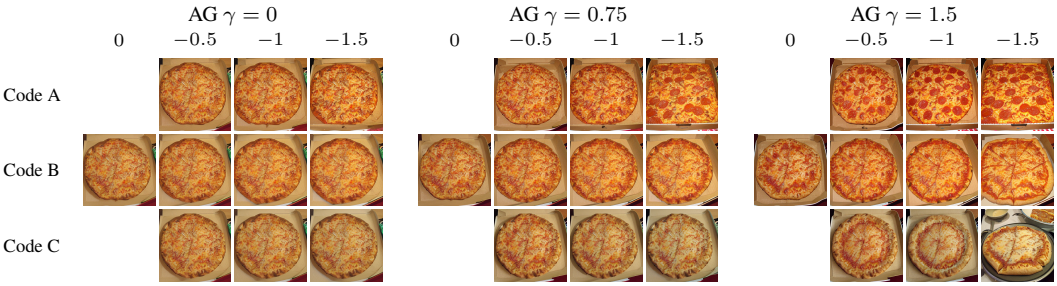


Pizza

(a) Auxiliary conditioning ($\gamma = 0$)

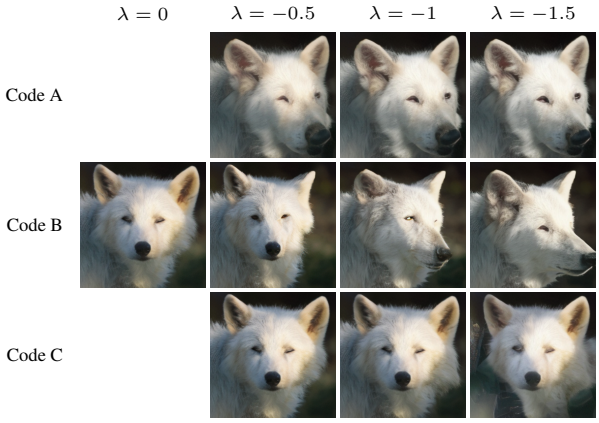


(b) Guidance across auxiliary scales

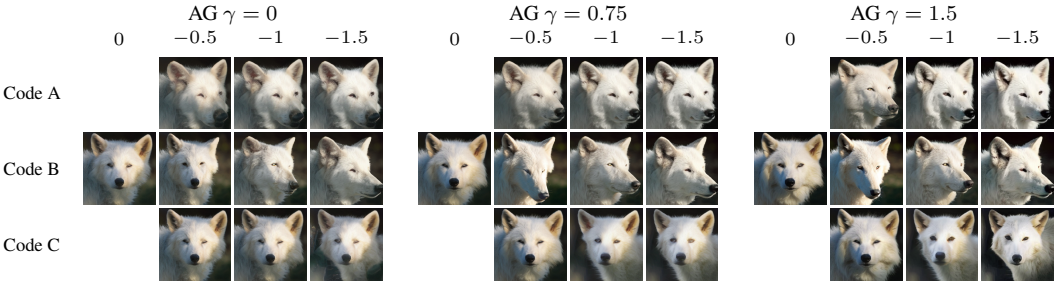


White wolf

(a) Auxiliary conditioning ($\gamma = 0$)

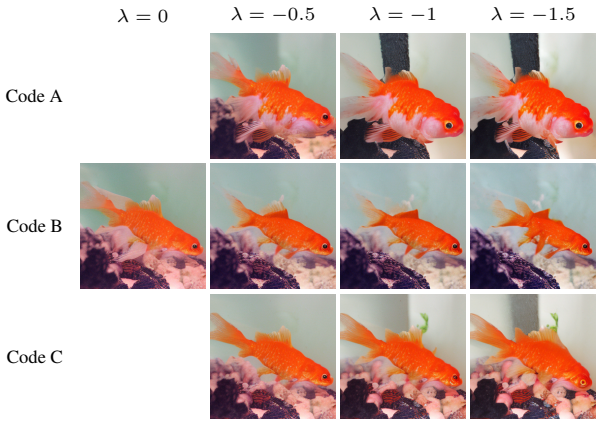


(b) Guidance across auxiliary scales

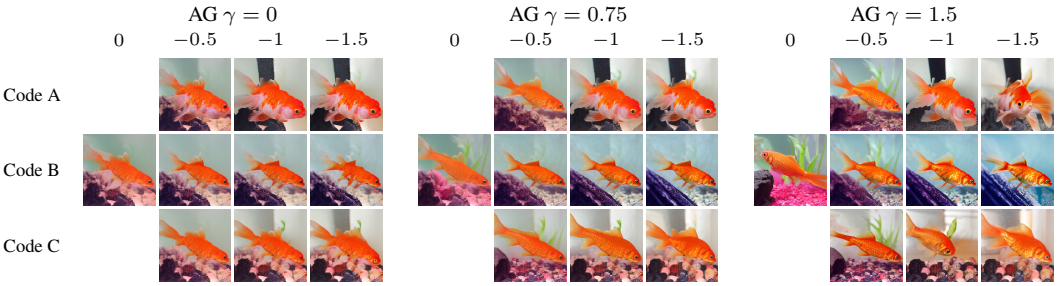


Goldfish

(a) Auxiliary conditioning ($\gamma = 0$)

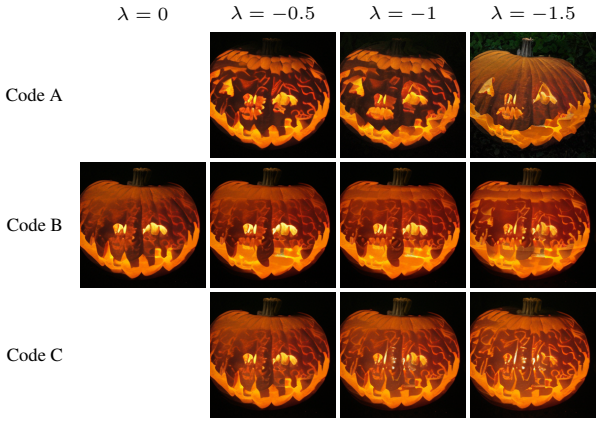


(b) Guidance across auxiliary scales

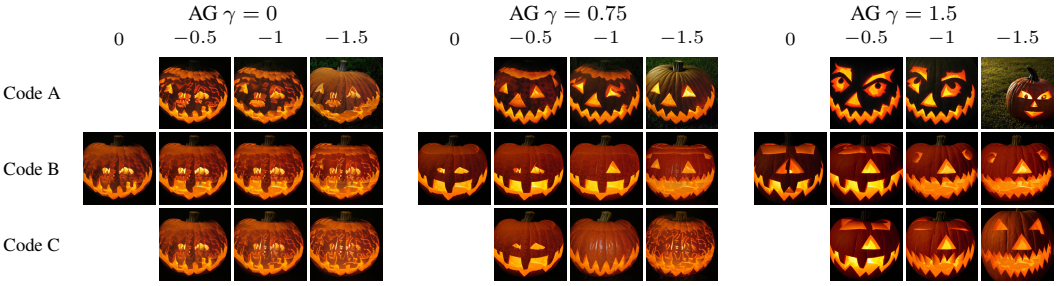


Jack-o'-lantern

(a) Auxiliary conditioning ($\gamma = 0$)

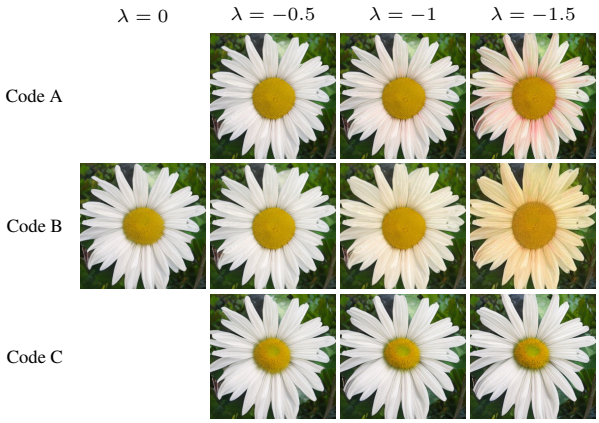


(b) Guidance across auxiliary scales

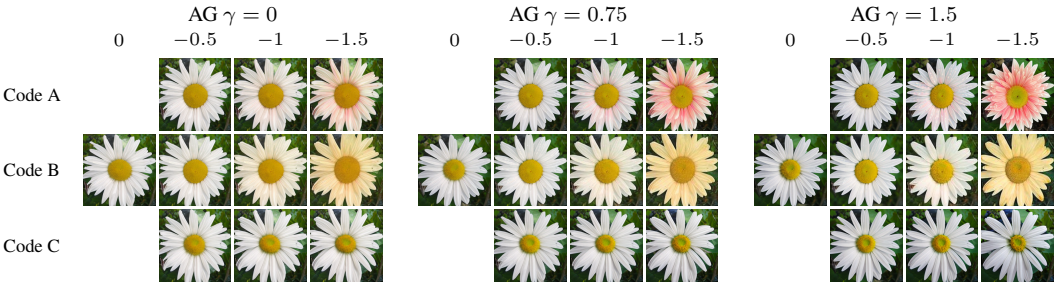


Daisy

(a) Auxiliary conditioning ($\gamma = 0$)

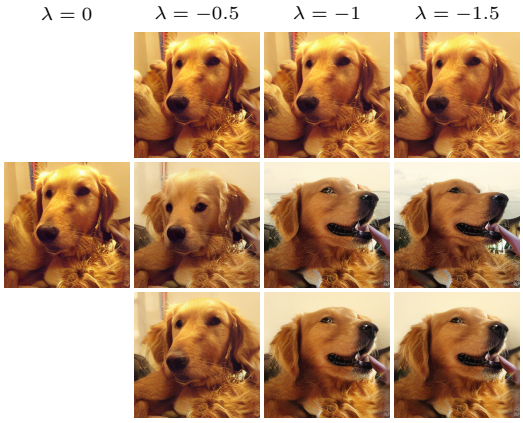


(b) Guidance across auxiliary scales

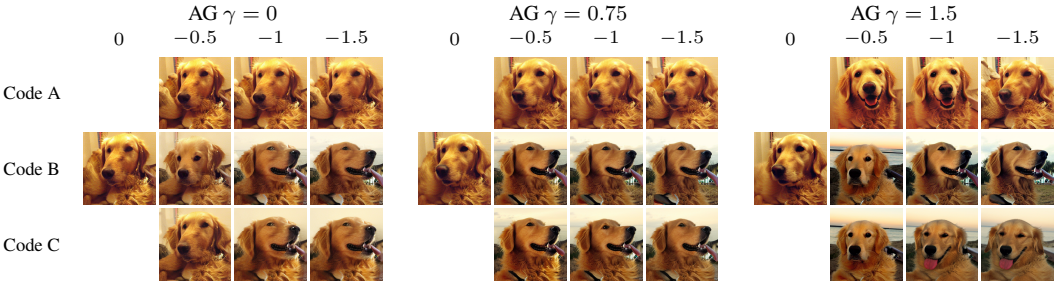


Golden retriever

(a) Auxiliary conditioning ($\gamma = 0$)

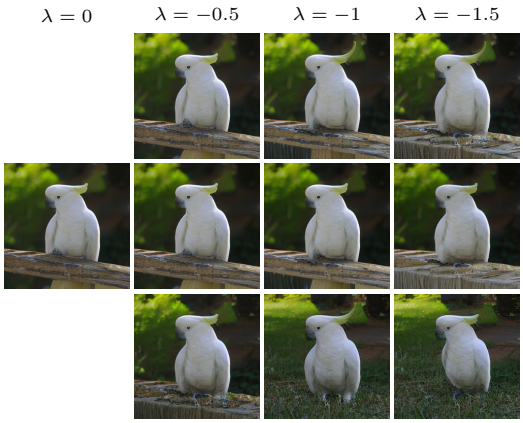


(b) Guidance across auxiliary scales

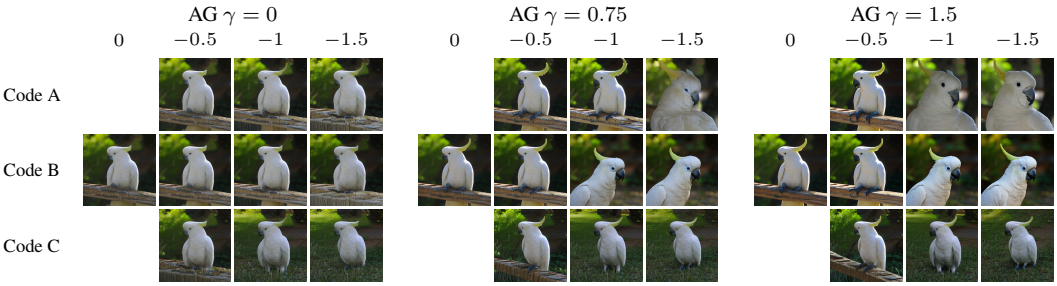


Sulphur-crested cockatoo

(a) Auxiliary conditioning ($\gamma = 0$)



(b) Guidance across auxiliary scales



E COMPUTATIONAL COST

E.1 TRAINING COST AND SCALING WITH THE NUMBER OF CANDIDATES

We report the cost of WTA post-training for SiT-XL/2, including computation per update, average runtime per epoch, and total training cost. Optimizer updates describe training exposure, FLOPs describe arithmetic work, and GPU-hours describe resource use over the measured training interval.

E.1.1 COMPUTATION IN THE IMPLEMENTED TRAINING STEP

For each example, our implementation first evaluates all K candidate predictions without retaining their gradient graphs, selects the winner, and recomputes that winner with gradients before backpropagation. Thus, there are K no-gradient forward evaluations, one gradient-tracked forward evaluation, and one backward evaluation per example. Candidate evaluations are batched in chunks; these are logical per-example evaluations, not necessarily separate model invocations. Chunking bounds candidate activation memory while preserving the total number of candidate evaluations.

Let F_0 denote the model-core cost of an ordinary SiT forward, F_1 that of an auxiliary-conditioned forward, and β the ratio of backward to forward computation. At the same global batch size b,

$$C_{\mathrm{FM}} = b(1+\beta)F_0, \qquad C_{\mathrm{WTA}}(K) = b(K+1+\beta)F_1, \tag{21}$$

$$R(K) = \frac{C_{\mathrm{WTA}}(K)}{C_{\mathrm{FM}}} = \frac{K+1+\beta}{1+\beta}\frac{F_1}{F_0}. \tag{22}$$

Following the analytical convention used by DiT (Peebles & Xie, 2023), we use $\beta = 2$. Counting matrix multiply–accumulates (MACs) in the audited SiT-XL/2 implementation gives $F_0 = 118.621$ GMAC and $F_1 = 118.848$ GMAC per example. We count two FLOPs per MAC; the ratios are unchanged by this conversion. These estimates exclude VAE encoding, elementwise operations, the optimizer, and communication. For $K = 8$, $R(8) = 3.674$: the model-core work per update is approximately $3.67\times$ that of an ordinary FM update, rather than eight times. This arithmetic ratio is not a wall-time multiplier.

E.1.2 TRAINING COST AT $K = 8$

At $K = 8$, training takes an average of **1.15 hours per epoch** on eight GPUs, or **9.23 GPU-hours per epoch**. The total cost is 11.54 hours and **92.31 GPU-hours**. Runtime includes periodic checkpoint saves and preview generation; it excludes environment/data preparation, pretraining, and the model-search and evaluation campaign.

For context, the original ImageNet-256 SiT-XL model uses 7 million updates at batch 256 (Ma et al., 2024). Post-training therefore uses **0.71% of its update budget**. Accounting for the additional candidate computation gives

$$\frac{C_{\mathrm{post}}}{C_{\mathrm{pre}}} \approx \frac{50{,}040}{7{,}000{,}000}\, R(8) = 0.0263. \tag{23}$$

Thus SiT-XL/2 post-training adds approximately **2.6% of the original pretraining model-core computation**. This is an analytical FLOP comparison under the same counting convention, not a comparison of GPU-hours across the different pretraining and post-training hardware.

Table 9: Training cost for SiT-XL/2. The pretraining schedule is from Ma et al. (2024); the post-training exposure and runtime are verified from training logs. Relative computation is the analytical model-core estimate in Equation (23). GPU-hours cover the SiT-XL/2 post-training interval, including saves and previews.

Quantity	Original pretraining	$K = 8$ post-training
Training epochs	—	10
Optimizer updates	7,000,000	50,040
Global batch size	256	256
Example presentations	1,792,000,000	12,810,240
Relative update budget	100%	0.71%
Relative model-core computation	100%	2.63%
Observed post-training GPU-hours	—	92.31

E.1.3 Measured scaling with K

Table 10 reports measured runtime at different K. The runs share batch 256, two gradient accumulation microsteps, FP32 computation with TF32 enabled, candidate chunk size 32, AdamW, EMA, and the save/preview schedule. All use eight training ranks. The $K = 1$ row is the augmented no-competition control with the same candidate-scoring and winner-recomputation path; it is not ordinary FM. The recorded $K = 8$ runtime is $1.87\times$ that of $K = 1$. This is a descriptive ratio, not a controlled hardware comparison: the original platform GPU SKU is still unverified.

Table 10: Measured SiT-XL/2 training cost. Per-epoch costs are averages over the measured training interval, including checkpoint saves and previews. Candidate chunk size is 32. All runs use eight GPUs; the platform GPU SKU remains unverified.

	Average per epoch		Total: 10 epochs	
K	Hours	GPU-hours	Hours	GPU-hours
1	0.62	4.94	6.17	49.36
2	0.69	5.53	6.92	55.34
4	0.85	6.77	8.46	67.68
8	1.15	9.23	11.54	92.31
16	1.77	14.15	17.68	141.46

Matched hardware profiling. A separate controlled profile measures ordinary FM and a wider range of K on eight H100 80GB GPUs, using batch 256, two accumulation microsteps, FP32/TF32, and a common candidate chunk size of 128. Each arm starts from the official pretrained model, runs 50 warmup updates, then three consecutive 200-update timing blocks and five separate profiler updates. The timing blocks are not independent training seeds. Table 11 reports pooled mean step time, profiler-covered FLOPs, and peak reserved memory.

Table 11: Matched eight-H100 profile with candidate chunk size 128. FLOP ratios are relative to ordinary FM and cover supported profiler operators: "model" includes candidate/winner forwards and backward; "total" additionally includes VAE and other covered operations. Time is an instrumented steady-state measurement; memory is peak reserved GiB per GPU. This benchmark contains no $K = 8$ arm.

			FLOPs / FM		
Method	Seconds/update	Time / FM	Model	Total	GiB/GPU
FM	0.334	$1.00\times$	$1.00\times$	$1.00\times$	28.57
$K = 1$	0.424	$1.27\times$	$1.34\times$	$1.24\times$	35.72
$K = 4$	0.571	$1.71\times$	$2.34\times$	$1.96\times$	34.14
$K = 16$	1.180	$3.54\times$	$6.35\times$	$4.83\times$	35.72
$K = 32$	2.002	$6.00\times$	$11.69\times$	$8.67\times$	35.77
$K = 128$	6.957	$20.85\times$	$43.76\times$	$31.67\times$	35.82

The instrumented time includes data wait, VAE encoding, model training, optimizer, and EMA, but is not a full job-allocation timer; host-to-device transfer is outside its timed components, and checkpointing/previews are disabled. Profiler FLOPs are operator estimates, not a complete hardware instruction count. Runtime ratios differ from model-core FLOP ratios: for example, $K = 32$ requires $11.69\times$ the model FLOPs but $6.00\times$ the measured time. Candidate batching and fixed overheads can contribute to this difference; this profile does not isolate their individual effects. With chunking, peak reserved memory remains below 36 GiB per GPU even at $K = 128$ in this benchmark.

Interpretation and remaining accounting. These results support a relatively small incremental post-training budget compared with original pretraining, while making the per-update overhead explicit. Auxiliary strength can be varied without retraining at each setting. Moreover, the K training candidates do not require K predictor evaluations at inference: each deployed field evaluates one prediction per solver evaluation, with any guidance cost accounted for separately. We do not infer improved quality per training FLOP without an equal-compute convergence comparison.