ALBIS: NOISE SCHEDULING FOR DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Training a diffusion model means deciding how much effort each noise level gets. This noise schedule, set by noise sampling and loss weighting, strongly affects sample quality. Yet what makes a good schedule is hard to tell, because DDPM, EDM, and flow matching each bundle it with other training choices. We show that once these choices are aligned, the large differences between the frameworks mostly disappear. So how should we design the noise schedule? Permanently favoring low or high noise levels gives no consistent gain, but shifting the emphasis during training does. This raises the question of how much it matters when noise levels are emphasized, compared to which. We study this question with Albis, a deliberately simple scheduler that shifts samples toward noise levels where the training loss is still high. With sufficient training, Albis reaches a lower loss than fixed logit-normal sampling at every noise level and improves sample quality on CIFAR-10 and CelebA-HQ subsets and ImageNet. When noise levels are emphasized therefore matters as much as which levels are emphasized.

1 INTRODUCTION

A diffusion model learns to remove noise at every noise level, from almost clean images to pure noise. Each noise level is a different task. At high noise, the model has to recover the global structure of an image, and at low noise, it has to restore fine details. Model capacity and training steps are limited, so every training recipe has to decide how much effort goes to each noise level. We call this decision the noise schedule. It is set by the distribution from which training noise levels are sampled and by the weight that each level receives in the loss. The noise schedule has a large effect on sample quality (Chen, 2023; Esser et al., 2024), but it is much less clear what makes a noise schedule good.

This question is hard to answer from existing work because a noise schedule rarely comes alone. DDPM (Ho et al., 2020), EDM (Karras et al., 2022), and flow matching (Flow) (Lipman et al., 2023) can be transformed into one another (Song et al., 2021b; Gao et al., 2024), yet each comes as a complete recipe with its own regression target, sampling distribution, loss weight, and inference choices. We therefore map the three onto the same noise levels by matching their signal-to-noise ratios, and align these choices one by one. Under these controls, the large performance differences between their defaults mostly disappear. In our experiments, the choice of framework therefore matters much less than its default recipe suggests. What still matters is the noise schedule itself.

So how should we design the noise schedule? In our experiments, permanently favoring either low or high noise levels does not produce consistent improvements. Shifting the emphasis across noise levels during training, however, works well. Prior work has already shown that such curricula improve training: Xu et al. (2024) gradually shift noise-level sampling from uniform toward lower noise levels, Kim et al. (2025a) start from the easiest high-noise levels and add harder lower-noise levels when the training loss plateaus, and Sun (2026) move from logit-normal to uniform sampling in two phases. However, even though these methods improve generative quality, they also shift the overall emphasis across noise levels, so it remains unclear whether the gain comes from the curriculum itself or from the new emphasis.

We show that *when* noise levels are emphasized is just as important as *which* noise levels are emphasized. To isolate this effect, we build Albis, a deliberately simple curriculum noise schedule, and compare it against Flow (logit-normal), the fixed logit-normal sampling used in Stable Diffusion 3 (Esser et al., 2024). During training, Albis shifts samples toward the noise levels where the training

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.

loss is still high. With a sufficiently long training budget, Albis reaches a lower loss than Flow (logit-normal) at every noise level at the same time. The curriculum therefore makes training itself more effective: with the same number of steps, the model learns every noise level better.

As a result, curriculum learning can improve the generative ability of diffusion models regardless of which noise levels matter most for the application or which inference schedule is used. Once the training budget is large enough, Albis improves sample quality over Flow (logit-normal) on subsets of CIFAR-10 and CelebA-HQ, and on ImageNet with a shared diffusion-transformer backbone. Given these results and our analysis of the frameworks and noise schedule, we consider curriculum learning over the noise levels a promising and underexplored direction for improving diffusion model training.

2 Related Work

Why fixed schedules emphasize certain noise levels. Fixed schedules favor some noise levels by sampling or weighting them more. Chen (2023) shifts training to higher noise at higher resolution, where redundant pixels make noise less destructive. Logit-normal sampling favors the middle, where the velocity target is expected to be hardest (Esser et al., 2024), and Hang et al. (2025) center sampling on the transition from signal to noise dominance. Other samplers draw near-pure noise less often, because the forward marginals change fastest near the data (Zheng et al., 2024b), become indistinguishable across data at high noise (Zheng et al., 2024a), or because near-pure noise is easy to learn (Wang et al., 2025). P2 weighting suppresses low noise, which teaches only imperceptible details (Choi et al., 2022), and Min-SNR balances conflicting gradients between noise levels (Hang et al., 2023). Each answers *which* levels matter and keeps that answer throughout training.

Why schedules adapt during training. Adaptive schedules change this answer. Improved DDPM and VDM do so to reduce the variance of the variational-bound estimator (Nichol & Dhariwal, 2021; Kingma et al., 2021). Kingma & Gao (2023) show that the schedule is an importance-sampling proposal that leaves the objective unchanged, and adapt it to the running loss for the same reason. Others equalize the loss contribution of each noise level (Karras et al., 2024), maximize the expected decrease of the bound (Kim et al., 2025b), or follow where uncertainty about the data is resolved fastest (Raya et al., 2026). Albis adopts Improved DDPM's root-second-moment proposal. Because a loss-driven proposal moves as the model learns, it also acts as a curriculum.

3 When Do Diffusion Frameworks Agree?

DDPM, EDM, and Flow can be transformed into one another under suitable parameterizations (Gao et al., 2024), but each of them is normally used as a complete training recipe. A recipe fixes which noise levels are sampled during training, how much each level contributes to the loss, what quantity the network predicts, how that prediction is scaled, and how the sampler is discretized. Trained with their defaults, the three perform differently, and the gaps are large. An equivalence that holds on paper therefore leaves open how much of it survives in practice. This section works across all three frameworks and asks what it takes to bring them into agreement while each one keeps its own formulation.

3.1 A general training framework

A shared noise coordinate. We first write the three frameworks in one notation. Let x_0 denote centered data with unit per-feature variance and let $\epsilon \sim \mathcal{N}(0, I)$ be independent noise. Following rectified flow (Liu et al., 2023) and conditional flow matching (Lipman et al., 2023), we index noise by $\alpha \in (0, 1)$ and mix data and noise linearly:

$$x_\alpha = (1-\alpha)x_0 + \alpha\epsilon, \qquad y_\alpha = \epsilon - x_0, \qquad q(\alpha) = (1-\alpha)^2 + \alpha^2. \tag{1}$$

Here y_α is the velocity along this path and $q(\alpha)$ is the per-feature variance of x_α. Clean data lies at $\alpha = 0$ and pure noise at $\alpha = 1$. The signal-to-noise ratio is $\mathrm{SNR} = (1-\alpha)^2/\alpha^2$, and $\mathrm{logit}\,\alpha = \log[\alpha/(1-\alpha)] = -\frac{1}{2}\log \mathrm{SNR}$. We call noise levels near $\alpha = 1/2$ the *middle* of the noise range, and levels near 0 and 1 its *tails*.

Table 1: DDPM, EDM, and Flow in the framework of Section 3.1, with their defaults and matched to Flow. Matched rows map α to each framework's own noise index, and $s = \sigma_{\text{data}} = 0.5$ is EDM's clean-data standard deviation. λ weights the squared error of the regression target y^*. The last two columns give the conditional normalization of Equation 3, and a dash marks a row without it. Here $q = q(\alpha)$, and Appendix B derives every entry.

					Cond. norm.	
Method	Index	Target	Sampling	Weight λ	$\mathbb{E}_{\mathcal{N}}[y^* \mid x_\alpha]$	$\mathbb{V}_{\mathcal{N}}[y^* \mid x_\alpha]$
DDPM	ω	ϵ	$t \sim \mathcal{U}(0,1), \omega = \omega(t)$	1	–	–
matched	$\omega = \frac{(1-\alpha)^2}{q}$	ϵ	$\alpha \sim \mathcal{U}(0,1)$	$(1-\alpha)^{-2}$	$\frac{\alpha}{q} x_\alpha$	$\frac{(1-\alpha)^2}{q}$
EDM	σ	sx_0	$\log \sigma \sim \mathcal{N}(-1.2, 1.2^2)$	$\frac{s^2+\sigma^2}{(s\sigma)^2}$	$\frac{s(1-\alpha)}{q} x_\alpha$	$\frac{(s\alpha)^2}{q}$
matched	$\sigma = \frac{s\alpha}{1-\alpha}$	sx_0	$\alpha \sim \mathcal{U}(0,1)$	$(s\alpha)^{-2}$	$\frac{s(1-\alpha)}{q} x_\alpha$	$\frac{(s\alpha)^2}{q}$
Flow	α	$\epsilon - x_0$	$\alpha \sim \mathcal{U}(0,1)$	1	–	–
matched	α	$\epsilon - x_0$	$\alpha \sim \mathcal{U}(0,1)$	1	$\frac{2\alpha-1}{q} x_\alpha$	$\frac{1}{q}$

Noise indices. DDPM and EDM each use their own noise index, which we map onto α. DDPM forms $\sqrt{\omega}\, x_0 + \sqrt{1-\omega}\, \epsilon$, where ω is DDPM's $\bar{\alpha}_t = \prod_{s \le t}(1-\beta_s)$ for a linear variance schedule β (Ho et al., 2020). We use its continuous-time form $\omega(t) = \exp[-1000 \int_0^t \beta(u)\, du]$, where $t \in (0,1)$ is the diffusion time and β rises linearly from 10^{-4} to 0.02. EDM forms $sx_0 + \sigma\epsilon$ with noise standard deviation σ and assumes clean data with standard deviation $s = \sigma_{\text{data}} = 0.5$, so for EDM we multiply our unit-variance data by s. Equating the SNRs relates both indices to α, and the matched rows of Table 1 use this map to turn a uniform α into the native index. Both noisy inputs are then rescalings of x_α, so all three frameworks corrupt the data along the same path.

Noise objective. On this path, a method samples noise levels from a density π, regresses a target y^* that combines x_0 and ϵ linearly, weights the squared error of y^* by $\lambda(\alpha)$, and turns the network output F_θ into a prediction $\hat{y}_\theta$. With squared norms averaged over features, its objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\alpha \sim \pi}\, \mathbb{E}_{x_0, \epsilon}\big[\lambda(\alpha) \|\hat{y}_\theta - y^*\|^2\big]. \tag{2}$$

Only the product $\pi\lambda$ enters this objective, so sampling a noise level more often and weighting it more heavily are equivalent in expectation (Kingma & Gao, 2023). Still, sampling and weighting train different models even when their objectives agree, in our experiments (Appendix C.4) and in those of Hang et al. (2025). The *noise schedule* is therefore the pair (π, λ), not $\pi\lambda$.

Conditional normalization. The objective leaves one choice open: how the network output F_θ becomes the prediction $\hat{y}_\theta$. By default, $\hat{y}_\theta = F_\theta$. *Conditional normalization* instead builds

$$\hat{y}_\theta = \mathbb{E}_{\mathcal{N}}[y^* \mid x_\alpha] + \sqrt{\mathbb{V}_{\mathcal{N}}[y^* \mid x_\alpha]}\; F_\theta. \tag{3}$$

Here $\mathbb{E}_{\mathcal{N}}$ and $\mathbb{V}_{\mathcal{N}}$ are the mean and variance of y^* given x_α computed as if $x_0 \sim \mathcal{N}(0, I)$. These values can be computed in closed form (Table 1). The network output F_θ then regresses $(y^* - \mathbb{E}_{\mathcal{N}})/\sqrt{\mathbb{V}_{\mathcal{N}}}$, which has approximately unit variance at every noise level. EDM's skip and output coefficients produce exactly this form (Karras et al., 2022). Its absence matters most for DDPM. At high noise, x_α nearly determines ϵ, so the residual variance $\mathbb{V}_{\mathcal{N}}[\epsilon \mid x_\alpha] = \omega$ vanishes although ϵ itself has unit variance.

Matching to Flow. To match a framework to Flow (Table 1), we keep its index, target, and update rule and change three things. First, we match the noise objective: we draw $\alpha \sim \mathcal{U}(0,1)$, map it to the native index, and replace the weight λ. Second, we give all three frameworks conditional normalization, which EDM already has. Third, we match the inference schedule: DDPM and EDM take the N uniform steps in α that Flow uses, mapped to their native index. *Matched* therefore means matched to Flow with conditional normalization.

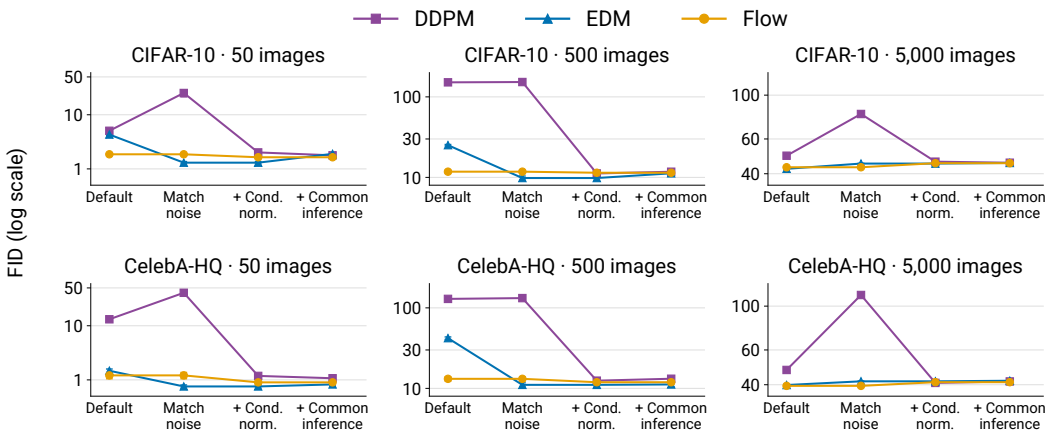

Figure 1: Cumulative framework controls on CIFAR-10 and CelebA-HQ subsets. The frameworks move closer together with each added control. Each point is a five-seed mean.

3.2 Controlled framework comparison

We now test how much of the gap between the defaults each matching step closes. We compare 50-, 500-, and 5,000-image subsets of CIFAR-10 (Krizhevsky, 2009) and CelebA-HQ (Karras et al., 2018), using five seeds and 20,000 training steps per configuration. We use a simple convolutional neural network (Appendix A.6), 10,000 generated images, and 128 inference steps. Every run, including its inference, takes roughly 15 to 30 minutes on an A5000 GPU. We evaluate Fréchet Inception Distance (FID) (Heusel et al., 2017) against the training subset. We use this training-subset FID because we measure how fast training converges, and we assume that generalization comes naturally with larger datasets. For CelebA-HQ we use decoded FID, comparing generated images with autoencoder reconstructions of the training images (Appendix A.7). Values should be compared within a dataset/subset column, not interpreted as full-dataset benchmark FIDs.

Figure 1 shows the three frameworks far apart under their defaults and close together once matched. Matching the noise objective alone leaves them far apart and even makes DDPM worse, and conditional normalization is what brings DDPM into line. For Flow conditional normalization seems negligible. Common inference has a relatively small effect. Under the final controls the three land within a narrow band, with CIFAR-10/500 FIDs of 11.76 ± 0.63, 11.23 ± 0.55, and 11.42 ± 0.54 for DDPM, EDM, and Flow. Table 6 in Appendix B.9 gives the complete results for every stage.

Matching the noise objective together with conditional normalization therefore explains most of the difference between these frameworks. Noise sampling and loss weighting are the choices usually discussed. Conditional normalization is just as necessary here. With the frameworks equalized, we build the rest of the paper on Flow, the simplest of the three.

4 A Curriculum Across Noise Levels

Section 3 showed that the frameworks reach comparable performance once their noise objective and conditional normalization agree. With the formulation settled, the next question is whether a better noise schedule can improve performance. We first investigated tilting the schedule permanently toward one part of the noise range: the low-noise half, the high-noise half, the middle, or the tails (Appendix C.1). But the preferred direction changes with the dataset, the subset size, and the way the emphasis is applied. A fixed tilt therefore does not seem to give us a uniformly better schedule. We now consider changing the schedule during the run.

Curriculum learning. Curriculum learning (Bengio et al., 2009) moves the emphasis between noise levels during training instead of fixing it. We define a curriculum as a noise schedule (π_k, λ_k) that changes with the training step k. In curriculum terminology, easy noise levels are those the model learns early in training, and hard noise levels are those it learns later.

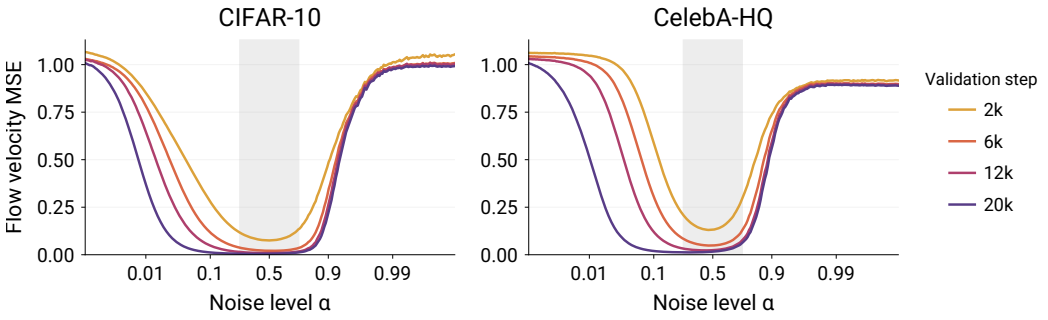

Figure 2: Validation velocity MSE for Flow (uniform) on the 500-image subsets, as five-seed means at four checkpoints. Shading marks the middle, $\alpha \in [0.25, 0.75]$. The noise axis is spaced by $\mathrm{logit}\,\alpha$.

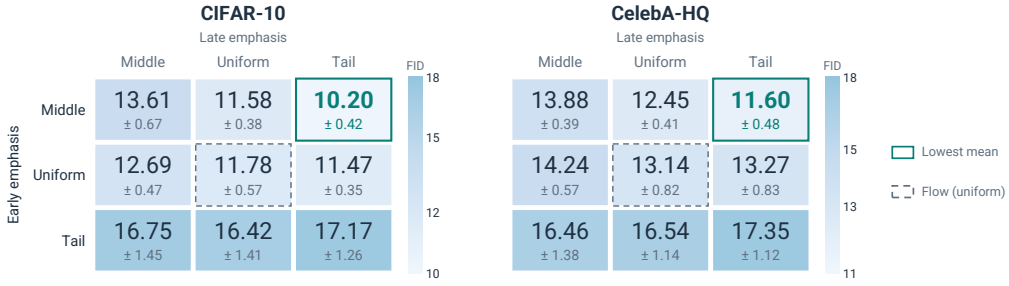

Figure 3: FID for the nine early/late curricula, lower is better. 500 training images, 20k steps, and mean $\pm$ s.e.m. over five seeds against the training subset.

Which noise levels are easy? To build a curriculum, we inspect how Flow (uniform) learns on the 500-image subsets. The first saved validation evaluation, at 2k steps, already shows low error in the middle; later checkpoints widen this region. Consecutive loss reductions move outward, most clearly toward the low-noise tail, while the extreme tails remain difficult. Figure 2 shows the curves; Appendix C.2 gives the interval reductions. Flow models seem to learn the middle first and then move to the tails.

A direct scheduling intervention. If Flow learns the middle first, a good schedule should emphasize the middle early and the tails later. To test this hypothesis, we compare nine combinations of early and late emphasis, holding Flow, unit loss weight, and the 20k-step budget fixed. Middle emphasizes $\alpha \in [0.25, 0.75]$, Tail emphasizes its complement, and Uniform samples uniformly in α. We geometrically interpolate the middle-to-outside density ratio between endpoints 3, 1, or $1/3$. For the middle-to-tail curriculum, for example, $\pi_k(\alpha) = 2r_k/(r_k + 1)$ on $[0.25, 0.75]$ and $\pi_k(\alpha) = 2/(r_k + 1)$ elsewhere, with $r_k = 3^{1-2k/K}$ and K the total number of training steps.

The middle-to-tail curriculum achieves the lowest mean FID in both grids, well clear of Flow (uniform) (Figure 3). This result qualitatively aligns with prior noise-level curricula (Xu et al., 2024; Kim et al., 2025a; Sun, 2026). Appendix C.3 gives the full 9×9 comparisons across subset sizes and sampling/weighting interventions, and discusses the sensitivity of the curriculum to task difficulty and pace. Where the model learns first therefore tells us how to schedule: we get a better schedule by following the model's own progress, emphasizing the middle while it is being learned and the tails once it is.

5 ALBIS: ADAPTIVE LOSS-BASED IMPORTANCE SAMPLING

Section 4 showed that a hand-set curriculum over noise levels speeds up diffusion pretraining. A good curriculum learns the easy noise levels first and then moves on to the hard tails, while not forgetting the easy part. The aim of this section is to provide the simplest possible scheduler that

does both: one that needs no tuning, and that is easy to use and easy to understand. A scheduler this simple is enough to show that curriculum learning can be an important part of diffusion training. We call the resulting schedule Albis. Below we outline how we learn the easy and the hard parts of the curriculum.

ALBIS	PREDICT	SAMPLE	ADAPT
	Flow velocity	Logit Student-t, $\nu = 16$	Loss-EMA importance sampling

Start where the model learns first. The easy part needs an early emphasis on the middle. Flow is often combined with logit-normal noise sampling, as in SD3 (Esser et al., 2024), and we call this baseline Flow (logit-normal). It draws $\mathrm{logit}\,\alpha$ from a normal distribution and concentrates training on the middle. This emphasis makes logit-normal sampling a good base for Albis. The base has one problem: its tails are thin. The extreme noise levels are almost never sampled, and nothing the sampler does later can reach them. We therefore draw the logit coordinate from a Student-t with location zero, scale one, and $\nu = 16$ degrees of freedom. The Student-t keeps the central shape of the base and thickens its tails. These heavier tails keep the hard levels in reach for the hard part of the curriculum. We use $\nu = 16$ throughout, and we ablate this choice in Appendix D.2.1.

Move to the tails without forgetting the middle. The base holds a single emphasis for all of training and gives us the easy part of the curriculum only. In Figure 2 the large-loss region retreats from the middle into the tails on its own. We can therefore follow the loss and let the model set the pace.

For this, we use the variance-reduction formulation of Improved DDPM (Nichol & Dhariwal, 2021). Let $\pi_0(\alpha)$ be the logit Student-t base density, ℓ_k the per-example velocity MSE at step k, and $m_k(\alpha)$ a kernel-smoothed EMA of its second moment. We sample from

$$\pi_k(\alpha) \;\propto\; \pi_0(\alpha)\sqrt{m_k(\alpha)}, \qquad \lambda_k(\alpha) = \frac{\pi_0(\alpha)}{\pi_k(\alpha)}, \tag{4}$$

and multiply the loss by λ_k. Appendix D.1 gives the density and the exact algorithm. For a fixed model, this proposal minimizes the variance of the loss estimate (Appendix D.3.2). It is also curriculum learning: the proposal follows the loss, so it moves the samples to the noise levels that the model has not learned yet.

The proposal π_k bends the base density π_0 toward the levels that still carry loss. Dividing by the proposal makes the mechanism importance sampling: the schedule (π_k, λ_k) changes over training, but its product stays π_0, so in expectation the objective stays the base logit Student-t objective. Because the objective stays fixed, the middle keeps its full share of it, and the model does not forget what it learned there. The emphasis still moves, because the effect of the sampling density outweighs the weight that compensates for it (Appendix C.4). Albis therefore acts as a curriculum although its objective is fixed. In Albis, variance reduction and curriculum learning are closely linked, and Section 6.1 discusses this link.

6 RESULTS

We evaluate Albis at three scales: 20k-step runs on small training subsets, longer training budgets on the same backbone, and a diffusion transformer on ImageNet-256. If Albis acts as a curriculum, it should beat both fixed samplers once the model gets past the easy middle. Flow (uniform) samples noise levels uniformly and lacks the early focus on the middle. Flow (logit-normal) has this focus but never moves on to the tails.

6.1 SMALL SUBSETS

We use the same settings as the framework comparison in Section 3.2: 50-, 500-, and 5,000-image subsets of CIFAR-10 and CelebA-HQ, five seeds, 20,000 training steps, the simple convolutional neural network, 10,000 generated images, and 128 inference steps. As in Section 3.2, FID measures agreement with the training subset, and CelebA-HQ uses decoded FID (Appendix A.7).

Table 2: Small-subset comparison after 20k steps: FID ± s.e.m. over five seeds. Column headings are the training-subset sizes, which also define the FID references. Bold marks the lowest mean.

	CIFAR-10			CelebA-HQ		
Method	50	500	5,000	50	500	5,000
EDM	4.31±0.20	25.14±0.52	**42.36±0.39**	1.48±0.11	42.06±1.49	39.86±0.29
Flow (uniform)	1.86±0.04	11.78±0.57	43.14±0.07	1.21±0.16	13.14±0.82	**39.48±0.16**
Flow (logit-normal)	4.13±0.29	10.91±0.43	61.48±0.59	1.67±0.13	7.05±0.27	47.31±0.44
MinSNR	3.07±0.15	18.68±0.54	54.85±0.82	1.25±0.08	32.79±0.88	43.65±0.55
EDM2	27.24±0.53	28.38±0.91	88.69±2.18	5.09±0.25	8.33±0.36	62.72±0.44
Albis (ours)	**0.99±0.05**	**6.34±0.12**	56.98±0.09	**0.79±0.05**	**5.30±0.21**	47.36±0.41

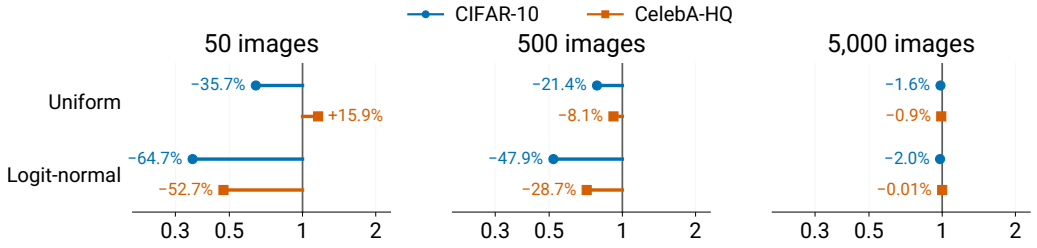

Figure 4: Switching the importance-sampling mechanism on and off, holding the base fixed within each pair. Endpoints give five-seed mean FID with adaptation divided by mean FID without it at 20k; left of one favors adaptation. Table 12 gives the means.

Table 2 compares Albis with EDM, Flow (uniform), Flow (logit-normal), MinSNR, and EDM2 as training components on the same backbone. MinSNR uses cosine VP noising, v-prediction, and a clipped SNR weight (Hang et al., 2023) (Appendix B.6). EDM2 learns a noise-dependent uncertainty weight while its logit-normal sampler stays fixed (Karras et al., 2024). Because EDM2 primarily studies architecture, we take only that weighting as our comparator for learned loss allocation (Appendix B.7).

On both datasets, Albis attains the best FID at 50 and 500 images. Interestingly, the two fixed bases swap places between those columns: Flow (logit-normal) beats Flow (uniform) at 500 images, and the ordering inverts at 50. This follows curriculum learning exactly. Logit-normal sampling focuses on the easy middle first, but its thin tails almost never reach the hard levels. At 500 images that emphasis still pays, because the easy stage fills most of the budget. At 50 the model passes the easy stage quickly and spends the rest where logit-normal sampling cannot help.

The 5,000-image column is a different regime: 20k steps is far too short to learn 5,000 images, so a curriculum never gets past its first stage. We still report it because it gives insight into how noise schedules behave under a low training budget, which Appendix C.1 examines directly.

Are we curriculum learning? Albis uses the variance-reduction formulation of importance sampling, which also acts as a curriculum. To understand some of its behavior, however, we need the curriculum view. Figure 4 switches importance sampling on and off over a uniform and a logit-normal base. Adaptation helps the logit-normal base far more than the uniform one. The curriculum view explains why: adaptation supplies the second half of a curriculum, so it pays off when the base supplies the first half. On the uniform base the early focus on the middle is missing, so adaptation supplies only the second half, and on CelebA-HQ with 50 images it gives no clear gain. We therefore cannot pick importance sampling for its variance reduction alone, because it also moves the emphasis across the noise levels.

The variance-reduction view is nonetheless a valid one. Appendix D.3 looks at the batch gradient norm and finds evidence of variance reduction: importance sampling lowers it. In Appendix D.3.3, we also tried switching the sampler from the loss to the per-example gradient norm, which minimizes the gradient variance directly, but that gives no clear benefit.

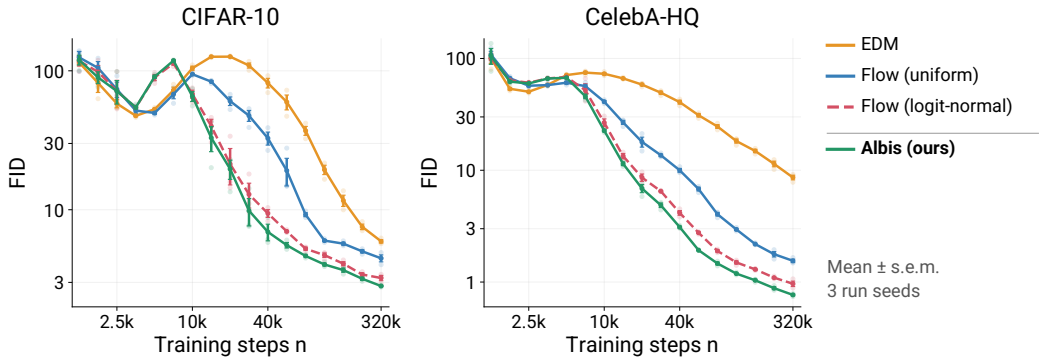

Figure 5: FID across the 17 training budgets. Markers are three-seed means with s.e.m. whiskers; faint points are the individual seeds.

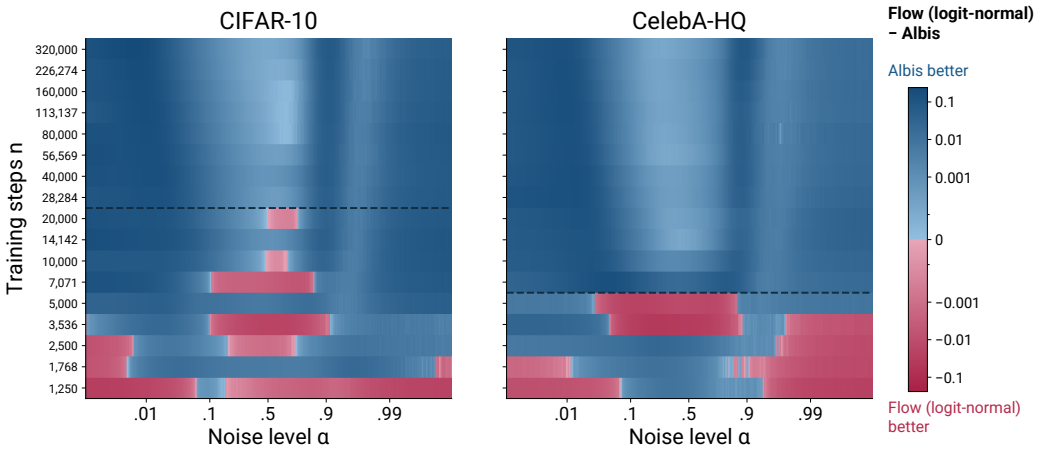

Figure 6: Flow (logit-normal) minus Albis mean velocity MSE at every budget and noise level. Blue favors Albis, rose favors Flow (logit-normal). The dashed line marks the first budget from which Albis leads at every level.

6.2 Longer training

A curriculum needs enough steps to get past its first stage (Section 6.1). We therefore compare EDM, Flow (uniform), Flow (logit-normal), and Albis over 17 budgets from 1,250 to 320,000 steps. Every budget is a separate completed run with three seeds. We train the 36-million-parameter simple convolutional neural network (Appendix A.6.1) on 3,000 CIFAR-10 and 2,000 CelebA-HQ images. We use 50,000 generated images, 128 inference steps, and training-subset FID references (Section 3.2).

Figure 5 shows that Albis dominates every baseline once the training budget is large enough. At 320k steps it improves mean FID over Flow (logit-normal) by 12.5% on CIFAR-10 and mean decoded FID by 20.6% on CelebA-HQ. Albis also reaches the final quality of Flow (logit-normal) with 29% fewer training steps on both datasets.

FID scores the samples as a whole. To see at which noise levels Albis gains, Figure 6 compares the final Flow-velocity MSE of Flow (logit-normal) and Albis on a shared grid of 256 noise levels α. With longer training, Albis has strictly lower three-seed mean loss than Flow (logit-normal) at every measured noise level. For a sufficiently large training budget on these two datasets, Albis is therefore strictly better than logit-normal sampling as a training objective. A fixed sampler would concentrate its budget on some noise levels and give up on others (Appendix E.1). Curriculum learning avoids this trade-off entirely: it accelerates the pretraining itself, achieving better performance over all noise levels.

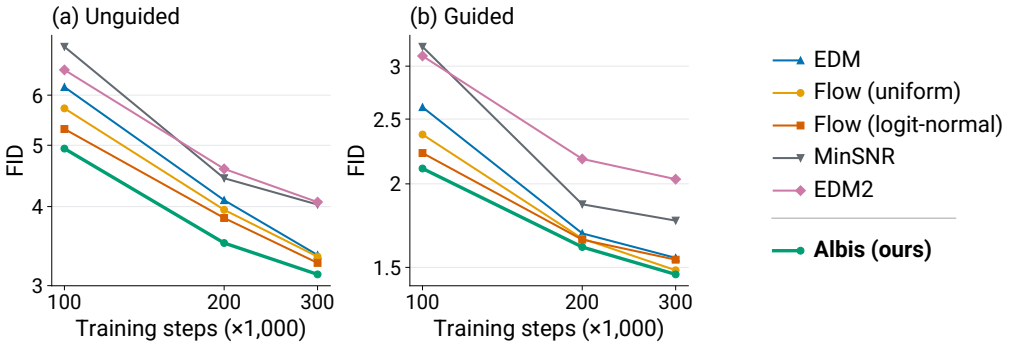

Figure 7: ImageNet FID at the three shared checkpoints, unguided and guided. Both axes are logarithmic.

6.3 Full ImageNet-256

Having studied small subsets, we now verify the performance at large scale. We skip full-dataset CIFAR-10 and CelebA-HQ evaluations because our current setup would be susceptible to exposure bias (Ning et al., 2023). During sampling, the model's own errors create inputs that never occur in training, so the trajectory drifts. Lower denoising loss can then coexist with worsening FID and blurred samples. Appendix E.2 develops this diagnosis. Stronger regularization could change this behavior, but would introduce an unwanted interaction with the noise schedules that we aim to study.

Table 3: ImageNet-256 FID at 300k steps.

Method	Unguided	Guided
EDM	3.35	1.55
Flow (uniform)	3.33	1.49
Flow (logit-normal)	3.26	1.54
MinSNR	4.03	1.76
EDM2	4.07	2.03
Albis (ours)	**3.13**	**1.46**

ImageNet (Deng et al., 2009) has enough images for the model to start generalizing, so exposure bias is less of a concern. We follow the LightningDiT implementation (Yao et al., 2025) of the diffusion transformer (Peebles & Xie, 2023) exactly, changing only the noise schedule. Albis reaches the lowest unguided and guided FID of the six methods at every shared checkpoint, already from 100k steps on (Table 3 and Figure 7). Albis therefore seems strong at large scale as well. The appendices give the remaining details. Appendix A lists the training and evaluation settings, Appendix B.8 the shared sampler, and Appendix F.5 unguided and guided samples from all six methods at 300k (Figures 54 and 55). Appendix E.3 reports the 300k noise-level losses, where Albis is slightly behind Flow (logit-normal) in the middle and much better at both tails.

7 Conclusion

In this paper, we asked what makes a good noise schedule, that is, how training effort should be divided across noise levels. We found that, in our experiments, the framework matters little once the training recipes are aligned. We identify conditional normalization as the key ingredient for this alignment: without it, matching the noise objective alone leaves the frameworks far apart. The noise schedule, on the other hand, is critical, and *when* it emphasizes noise levels matters as much as *which* levels it emphasizes. Fixed tilts gave no consistent gain, but a curriculum had potential. We therefore built Albis, a curriculum noise schedule. Given a sufficient training budget, Albis reaches a lower loss than Flow (logit-normal) at every noise level at the same time and improves sample quality.

Because Albis lowers the loss at every noise level, it can improve generative models regardless of which noise levels an application relies on or which inference schedule works best for it. We therefore expect that diffusion models currently trained with Flow (logit-normal) will benefit from switching to Albis. More broadly, curriculum learning over the noise levels is currently underexplored and ripe for further study. We suggest that future research on noise schedules treat the timing of the emphasis as a design choice of its own.

AI USE STATEMENT

AI was used substantially in both the research and the writing of this paper. The main body is AI-generated text that the authors then edited closely. The appendix is also AI-generated, with less human editing involved, so it documents the implementation and the experimental records in the assistant's wording rather than in prose the authors rewrote themselves. AI assistance further covered the manuscript template and build automation, the organization of the paper, checking mathematical explanations and reported values against our research presentation and the implementation, locating relevant literature within and beyond the supplied bibliography, marking additional references for author reading and verification, and checking conference formatting requirements. Generative AI also assisted the research implementation under the authors' direction. The numerical results and figure assets come from the existing experimental records, and no new experiments were performed for manuscript preparation. The authors take responsibility for every claim in the paper.

ETHICS STATEMENT

This work studies the training of image-generative models. Improved generation can support useful creative applications and can also facilitate misleading or nonconsensual imagery. Face generation raises privacy and representation concerns, and fitting very small image subsets can encourage memorization. Subset-reference FID is not a privacy, fairness, or generalization guarantee. Use and redistribution of the datasets, pretrained encoders, and generated outputs must respect their applicable terms. The experiments do not evaluate downstream misuse, demographic fairness, or privacy protection.

REPRODUCIBILITY STATEMENT

The code is attached to this submission. It contains the training, sampling, and evaluation implementations for every experiment we report, together with the scripts that define each configuration.

All details needed to reproduce this work are in the paper. The main text (Sections 3–5) is enough to implement the method, and Appendix D.1 gives the exact algorithm. Appendix A records the datasets, subset sizes, seeds, optimizer settings, training budgets, and evaluation protocol of each set of experiments. Appendix A.6 specifies the backbones, and Appendix B explains how we reproduced each baseline. The datasets and the pretrained KL-f8 and VA-VAE encoders are publicly available.

Every reported training trajectory uses one GPU. The 20k-step small-subset runs take well under one GPU-hour each, and the training-budget runs take at most 11.8 GPU-hours each (Appendix A.5). The main ImageNet experiment is the expensive exception: each 300k-step DiT-XL/1 trajectory takes about 105 hours on one NVIDIA B200, and its three complete evaluation checkpoints add about 26 hours. Reproducing all six methods therefore requires approximately 786 B200 GPU-hours.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. In *International Conference on Machine Learning (ICML)*, pp. 41–48. ACM, 2009.

Ting Chen. On the Importance of Noise Scheduling for Diffusion Models, 2023.

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception Prioritized Training of Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11472–11481, 2022.

Min Jin Chong and David Forsyth. Effectively Unbiased FID and Inception Score and Where to Find Them. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6070–6079, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. IEEE, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR, 2024.

Anne Gagneux, Ségolène Martin, Rémi Gribonval, and Mathurin Massias. Training Flow Matching: The Role of Weighting and Parameterization. In *2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa), at the International Conference on Learning Representations (ICLR)*, 2026.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans. Diffusion Meets Flow Matching: Two Sides of the Same Coin, 2024.

Guy Hacohen and Daphna Weinshall. On the Power of Curriculum Learning in Training Deep Networks. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2535–2544. PMLR, 2019.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient Diffusion Training via Min-SNR Weighting Strategy. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7441–7451, 2023.

Tiankai Hang, Shuyang Gu, Jianmin Bao, Fangyun Wei, Dong Chen, Xin Geng, and Baining Guo. Improved Noise Schedule for Diffusion Training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4796–4806, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *Neural Information Processing Systems Workshop on Deep Generative Models and Downstream Applications (NeurIPS DGM)*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6840–6851, 2020.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations (ICLR)*, 2018.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 26565–26577, 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24174–24184, 2024.

Jin-Young Kim, Hyojun Go, Soonwoo Kwon, and Hyun-Gyoon Kim. Denoising Task Difficulty-based Curriculum for Training Diffusion Models. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 64392–64405, 2025a.

Myunsoo Kim, Donghyeon Ki, Seong-Woong Shim, and Byung-Jun Lee. Adaptive Non-uniform Timestep Sampling for Accelerating Diffusion Model Training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2513–2522, 2025b.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 21696–21707, 2021.

Diederik P. Kingma and Ruiqi Gao. Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 65484–65516, 2023.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying Guidance in a Limited Interval Improves Sample and Distribution Quality in Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 122458–122483, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input Perturbation Reduces Exposure Bias in Diffusion Models. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26245–26265. PMLR, 2023.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Gabriel Raya, Bac Nguyen, Georgios Batzolis, Yuhta Takida, Dejan Stancevic, Naoki Murata, Chieh-Hsin Lai, Yuki Mitsufuji, and Luca Ambrogioni. Noise Scheduling as Information-Guided Allocation in Diffusion Training, 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations (ICLR)*, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Pengwei Sun. Curriculum Sampling: A Two-Phase Curriculum for Efficient Training of Flow Matching. In *2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa), at the International Conference on Learning Representations (ICLR)*, 2026.

Kai Wang, Mingjia Shi, Yukun Zhou, Zekai Li, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Hanwang Zhang, and Yang You. A Closer Look at Time Steps is Worthy of Triple Speed-Up for Diffusion Model Training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12934–12944, 2025.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When Do Curricula Work? In *International Conference on Learning Representations (ICLR)*, 2021.

Tianshuo Xu, Peng Mi, Ruilin Wang, and Yingcong Chen. Towards Faster Training of Diffusion Models: An Inspiration of a Consistency Phenomenon, 2024.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. Generation: Taming Optimization Dilemma in Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712, 2025.

Tianyi Zheng, Cong Geng, Peng-Tao Jiang, Ben Wan, Hao Zhang, Jinwei Chen, Jia Wang, and Bo Li. Non-uniform Timestep Sampling: Towards Faster Diffusion Model Training. In *ACM International Conference on Multimedia (ACM MM)*, pp. 7036–7045. ACM, 2024a.

Tianyi Zheng, Peng-Tao Jiang, Ben Wan, Hao Zhang, Jinwei Chen, Jia Wang, and Bo Li. Beta-Tuned Timestep Diffusion Model. In *European Conference on Computer Vision (ECCV)*, pp. 114–130. Springer Nature Switzerland, 2024b.

APPENDIX CONTENTS

A Experimental Setup and Evaluation **15**
A.1 Small-Subset Experiments . . . 15
A.2 Training-Budget Sweep . . . 15
A.3 ImageNet Experiments . . . 15
A.4 Interpreting Comparisons . . . 16
A.5 Training and Evaluation Compute . . . 16
A.6 Neural Network Architectures . . . 16
A.6.1 A simple convolutional neural network . . . 16
A.6.2 A simple vision transformer . . . 17
A.7 CelebA-HQ Evaluation and Decoded FID . . . 20

B Framework Formulations and Controlled Matching **23**
B.1 Notation and the reference Flow objective . . . 23
B.2 DDPM . . . 24
B.3 EDM . . . 25
B.4 Flow . . . 26
B.5 Flow (logit-normal) . . . 27
B.6 MinSNR . . . 27
B.7 EDM2 . . . 28
B.8 Shared ImageNet inference and common loss evaluation . . . 29
B.9 Cumulative Framework-Matching Results . . . 31
B.10 Related Framework Comparisons . . . 31

C Fixed-Schedule Controls and Curriculum Evidence **32**
C.1 Fixed Noise-Range Tilts . . . 32
C.2 Noise-Level Learning Diagnostics . . . 35
C.3 Full Curriculum Comparison . . . 36
C.4 Weighting Behaves Differently from Sampling . . . 38

D Albis Implementation and Mechanistic Evidence **40**
D.1 Adaptive Sampler Implementation . . . 40
D.2 Component Ablations . . . 42
D.2.1 Degrees of freedom and weight decay . . . 42
D.3 Gradient Norms and Variance Reduction . . . 44
D.3.1 Gradient norm during training . . . 44
D.3.2 Variance reduction . . . 44
D.3.3 Sampling by the gradient norm . . . 45
D.3.4 Uniform and logit-normal bases . . . 46

E Scaling Analyses and Failure Modes **48**
E.1 Training-Budget Noise-Level Losses . . . 48
E.2 Exposure Bias: When Lower Loss Produces Worse Images . . . 50
E.2.1 The problem and exposure bias . . . 50
E.2.2 Weight decay . . . 52
E.2.3 Depth . . . 55
E.2.4 Convolutional network and vision transformer . . . 57
E.2.5 Full-dataset experiments . . . 59
E.3 ImageNet Noise Levels . . . 64

F Qualitative Results **65**
F.1 Frameworks: default and matched settings . . . 65
F.2 Curriculum: uniform and reversed scheduling interventions . . . 66
F.3 Small subsets: all baselines and subset sizes . . . 67
F.4 Scaling: a fixed sample position across training budgets . . . 70
F.5 ImageNet: all baselines . . . 71

A Experimental Setup and Evaluation

This appendix records the settings behind the comparisons in the main text: the three sets of experiments and their hyperparameters, the architectures, the evaluation conventions, the baseline formulations, the sampler, and the supporting measurements.

A.1 Small-Subset Experiments

The framework comparison and the Albis construction experiments use CIFAR-10 and CelebA-HQ subsets of 50, 500, and 5,000 training images. The main-text curriculum experiment uses the 500-image subsets, and Appendix C.3 extends it to all three subset sizes together with the sampling and weighting interventions. Every configuration runs with five seeds: 100, 200, 300, 400, and 500. Reported uncertainty is the standard error of the mean over those seeds, $\mathrm{s.e.m.} = s_{\mathrm{sample}}/\sqrt{n}$ with $n = 5$. It therefore describes how much a complete training run varies from seed to seed, because each run already contributes a single FID computed from its full set of generated images.

The backbone is a simple convolutional neural network of width 32 with layers $(2, 2, 2)$ on a 32×32 grid, kernel size 7, zero dropout, batch size 128, learning rate 5×10^{-3}, and zero weight decay. Each run trains for 20,000 steps and records its training loss at every step. CIFAR-10 enters as pixels and CelebA-HQ as KL-f8 latents, with the generated images and the training references both decoded before FID (Appendix A.7). Evaluation uses 10,000 generated images and 128 inference steps.

A.2 Training-Budget Sweep

The scaling experiment fixes the simple convolutional neural network at width 64 with layers $(2, 2, 2)$ and batch size 256, and trains an independent run for every budget. It uses three seeds: 100, 200, and 300. The 17 budgets are

$$1{,}250,\ 1{,}768,\ 2{,}500,\ 3{,}536,\ 5{,}000,\ 7{,}071,\ 10{,}000,\ 14{,}142,\ 20{,}000,$$
$$28{,}284,\ 40{,}000,\ 56{,}569,\ 80{,}000,\ 113{,}137,\ 160{,}000,\ 226{,}274,\ 320{,}000.$$

FID uses the 3,000-image CIFAR-10 and 2,000-image CelebA-HQ training subsets as its references, and the noise-level loss evaluations use the same subsets. We generate 50,000 images and take 128 inference steps. The selected loss-curve panels in Appendix E.1 show the 2,500-, 40,000-, and 320,000-step budgets, while the difference visualization in Figure 6 keeps all 17 budgets and all 256 measured noise levels, which run from roughly 0.001055 to 0.999.

A.3 ImageNet Experiments

The six-method comparison uses DiT-XL/1 in VA-VAE latent space, with training and evaluation settings shared across methods. We report checkpoints at 100k, 200k, and 300k steps. Each FID uses 50,000 generated images per guidance mode, and the common velocity-loss evaluation uses 8,192 images on 256 noise levels. Guided generation uses CFG 6.7 with interval setting 0.125, and both modes use the 250-node Euler ODE with timestep shift 0.3. Each method trains once here, so these experiments carry no seed-to-seed uncertainty estimate.

Two settings deviate from the LightningDiT recipe. We apply both to every method, so they shift the whole comparison together.

The first is QK-Norm, which we enable in every run to keep training stable. The original LightningDiT paper and its released reproduction configurations leave it off (Yao et al., 2025), although the released 800-epoch configuration recommends enabling it against NaN-loss instability and warns that convergence may slow slightly. In our preliminary runs without it, several noise schedules suffered sudden gradient blow-ups, and the same failure hit our reproduction of the official LightningDiT baseline. The onset was stochastic, so we treated QK-Norm as necessary for stable long training and restarted every reported run with it enabled. The consequence is that these ImageNet numbers come from a LightningDiT variant, and they should be read against each other rather than against published LightningDiT results.

The second is the cosine-similarity term that the upstream implementation adds to its velocity MSE. We remove it, so every method trains on the plain loss of its own formulation. Auxiliary losses of

this kind are worth studying on their own, and keeping this one would have folded its effect into the noise-schedule comparison we want to make.

A.4 Interpreting Comparisons

A few conventions apply to every number in this paper, and they change how the tables should be read.

Lower FID is better, but its reference distribution changes with the dataset and the subset size. A value is therefore informative against the other values in its own column, and a comparison across columns compares reference distributions as much as models. Finite-sample FID also carries a model-dependent bias (Chong & Forsyth, 2020), and fixing the number of generated images leaves that bias in place. We report the empirical protocol above and apply no bias extrapolation. In the subset experiments, FID measures agreement with the training subset itself, so a very low score tells us that the model fits that subset closely. Whether it generalizes, and whether it memorizes, are separate questions that these scores leave open.

Where we report a relative loss, it divides one seed mean by another; we never average per-seed ratios. Finally, a baseline name in this paper identifies the training components we reproduce on the backbone that all methods share within each set of experiments. Appendix B states exactly which components those are for each method, and which parts of the original recipes stay out.

A.5 Training and Evaluation Compute

Table 4 reports the training compute of the three sets of experiments in the main text. Every run trains on a single GPU, so its GPU-hours equal its training wall-clock time. We measure this time as the sum of the measured times of all training steps, which excludes evaluation.

Table 4: Training compute of the main-text experiments. "Runs" counts independently trained models; in the training-budget sweep, every budget is a separate run. GPU-hours are measured training wall-clock time on one GPU.

Experiments	Runs	Steps/run	Batch	GPU	GPU-h/run	Total GPU-h
Small subsets	560	20,000	128	RTX A5000	0.24–0.61	165.7
Training-budget sweep	408	1,250–320,000	256	RTX A6000	0.04–11.8	873.7
ImageNet	6	300,000	1,024	NVIDIA B200	104.3–105.2	627.5

For ImageNet, the global batch of 1,024 is assembled from four micro-batches of 256. Evaluating the three reported checkpoints of one method adds about 26.4 GPU-hours, so training and evaluating all six methods takes approximately 786 B200 GPU-hours.

A.6 Neural Network Architectures

Every schedule comparison in this paper runs on a deliberately plain backbone. We build it from a small set of standard operations, so that special-purpose diffusion architecture choices and regularization techniques stay out of the schedule analysis, and we hold it fixed across the schedules compared within each experiment. The exact combination is specific to these experiments, so we give it in full here.

A.6.1 A simple convolutional neural network

Input and conditioning. Let $x \in \mathbb{R}^{H\times W\times C}$ denote the noisy input supplied to the network, omitting the batch dimension, and let d be the initial feature width. A learned affine map projects the C channels at each spatial location to d features; this is equivalent to a 1×1 convolution. A second affine map projects the scalar noise-conditioning value s to d features and adds the same vector at every location:

$$h_{ij} = W_x x_{ij} + b_x + w_s s + b_s + e_y. \tag{5}$$

The learned class embedding e_y is included only in class-conditional runs. CIFAR-10 and CelebA-HQ are unconditional; the full-dataset ImageNet convolutional runs use 1,000 classes. The scalar

s is supplied by the framework: Flow uses α, DDPM and its cosine variant use ω (DDPM's $\bar{\alpha}_t$), and EDM uses $\log(\sigma)/4$. There is no sinusoidal or Fourier embedding, conditioning multilayer perceptron, or repeated injection of the conditioning vector into later blocks.

One block. At a stage with c channels, each block has two residual updates:

$$z = h + \mathrm{DWConv}_{7\times 7}(\mathrm{RMSNorm}_1(h)), \tag{6}$$

$$h' = z + W_2\ \mathrm{GELU}(W_1\,\mathrm{RMSNorm}_2(z) + b_1) + b_2. \tag{7}$$

The depthwise convolution applies a separate 7×7 filter to each channel, with stride one and zero padding of three pixels. It preserves the spatial shape and has a learned bias. The two affine maps in the second update operate independently at each location, expanding c channels to $4c$ and projecting back to c. Both RMS normalizations act across channels at each location and have learned channel scales. The convolution therefore mixes spatial information and the two-layer feed-forward network mixes channels. Dropout is zero in the reported convolutional configurations.

Resolution changes and output. Figure 8 shows the full computation. Between stages, a fixed rearrangement packs each 2×2 spatial neighborhood into the channel dimension:

$$(H, W, c) \longrightarrow (H/2, W/2, 4c). \tag{8}$$

The reverse rearrangement restores spatial resolution and divides the channel count by four. These operations preserve every feature value and have no learned parameters; there is no pooling, interpolation, or strided convolution. For the layer setting $(2, 2, 2)$, the network applies two blocks at 32×32 with width d, two at 16×16 with width $4d$, four at 8×8 with width $16d$, two at 16×16 with width $4d$, and two at 32×32 with width d. The four blocks at the smallest grid are the two blocks from the descending path followed by the two from the ascending path. There are 12 blocks in total, with distinct parameters. Residual connections are local to each block; there are no skip connections between matching descending and ascending stages.

A final RMS normalization and affine projection map each location back to C output channels. The output projection's weights and biases are initialized to zero; the remaining layers use their PyTorch default initialization. The network therefore initially outputs zero, before any framework-level output transformation. Its output has the input's shape, and the framework determines the regression target and any conditional normalization (Appendix B). Those transformations are separate from the network's internal RMS normalization.

Configurations. The small-scale experiments with 50, 500, and 5,000 training images use $d = 32$, giving stage widths $(32, 128, 512)$. The longer-training 3,000-image CIFAR-10 and 2,000-image CelebA-HQ sweep uses $d = 64$, giving $(64, 256, 1024)$. The full-dataset convolutional experiments use $d = 256$, giving $(256, 1024, 4096)$. All use the $(2, 2, 2)$ layer setting, except the depth diagnostic, which uses $(4, 4, 4)$: its five successive stages contain $(4, 4, 8, 4, 4)$ blocks, or 24 in total.

CIFAR-10 enters as a $32 \times 32 \times 3$ pixel array, and CelebA-HQ as a $32 \times 32 \times 4$ latent array. For full-dataset ImageNet, a fixed rearrangement first converts $16 \times 16 \times 32$ latents to $32 \times 32 \times 8$. The network then follows the same spatial sequence, and the inverse rearrangement restores the native latent shape after the output projection.

Why keep it simple? The network uses ordinary residual additions, normalization, depthwise convolution, and channel projections. It has no attention, adaptive normalization, or diffusion-specific magnitude-preserving blocks. The schedule comparisons use zero dropout and zero weight decay for this network; weight decay is varied explicitly in the exposure-bias diagnostic. This choice keeps the architecture and its regularization fixed while we change the noise-scheduling components. The network still carries inductive bias, because local convolution, the resolution hierarchy, and normalization all shape what it learns. Our aim is narrower than removing that bias. We want to keep additional architecture and regularization choices from obscuring the effect we measure.

A.6.2 A SIMPLE VISION TRANSFORMER

We also use a simple vision transformer (ViT), following the patch-token construction of Dosovitskiy et al. (2021). We use this ViT only in the exposure-bias analysis, where it replaces the convolutional network in the training-budget sweep (Appendix E.2.4). We kept it out of the small-scale

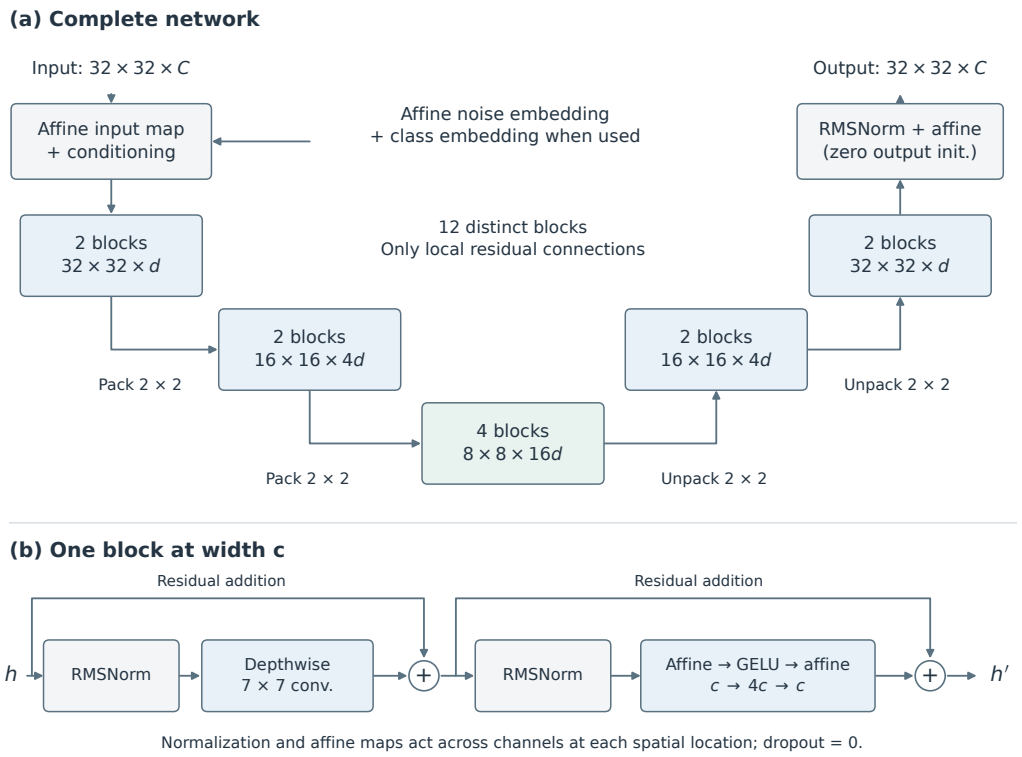

Figure 8: The simple convolutional neural network. Top: the complete sequence for the $(2, 2, 2)$ layer setting on a 32×32 grid. Shapes omit the batch dimension; d is the initial width and C the input/output channel count. Resolution changes are parameter-free spatial/channel rearrangements. Bottom: one block at width c, showing both local residual additions. Each occurrence has its own parameters. The output affine map is initialized to zero. Framework-level conditional normalization and the optional ImageNet latent rearrangement are outside the illustrated sequence.

experiments with 50, 500, and 5,000 training images because it learned too slowly inside their short 20k-step training runs. The simple ViT described here is separate from the LightningDiT backbone used for the main ImageNet results.

Patches and embeddings. Figure 9 shows the complete ViT and one transformer block. The ViT splits the noisy input into non-overlapping 4×4 patches. Each flattened patch of $16C$ values is affinely projected to a token of width 512. A 32×32 input therefore gives 64 tokens. Each token receives an affine projection of its two-dimensional patch coordinates, with each coordinate ranging from zero to one across the grid. The same affine embedding of the scalar noise-conditioning value is added to every token. The implementation also supports an additive class embedding; the reported CIFAR-10 and CelebA-HQ ViT runs are unconditional. There is no classification token.

Transformer blocks and output. The token sequence passes through 12 blocks, each consisting of a LayerNorm–self-attention residual update followed by a LayerNorm–feed-forward residual update. Attention uses 8 heads of dimension 64, learned affine query, key, value, and output projections, and full, non-causal scaled dot-product attention. Queries and keys each receive a learned RMS normalization within each head before attention (QK-Norm). The feed-forward network expands the width from 512 to 2,048, applies GELU, and projects back to 512. There is no attention or feed-forward dropout, adaptive LayerNorm, or per-block conditioning module.

After the last block, LayerNorm and an affine map produce $16C$ values per token. Reassembling the patches gives an output with the original input shape. As in the simple convolutional neural network, the final affine map's weights and biases are initialized to zero and the other layers use their default

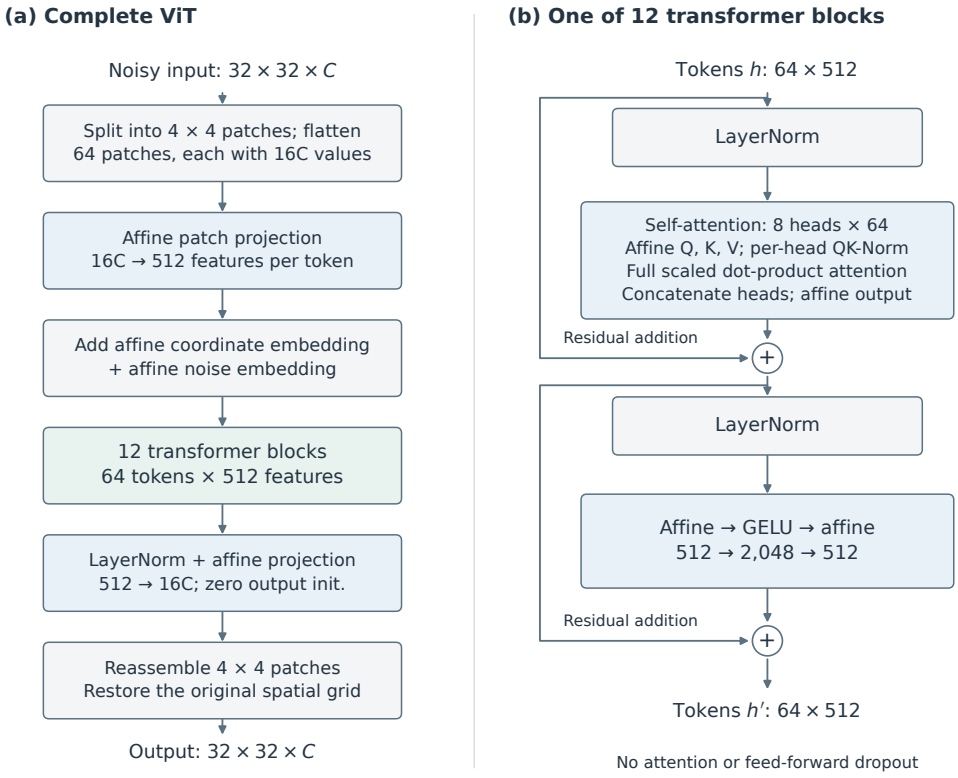

Figure 9: The simple ViT used only in the exposure-bias analysis. Left: a 32×32 input with C channels is divided into 64 non-overlapping 4×4 patches, embedded at width 512, processed by 12 distinct transformer blocks, and reconstructed through a zero-initialized output projection. Affine coordinate and noise embeddings are added once, before the block stack. The illustrated CIFAR-10 and CelebA-HQ runs use no class embedding or classification token. Right: one block, with pre-LayerNorm, 8 attention heads of dimension 64, per-head RMS normalization of queries and keys (QK-Norm), a feed-forward expansion to 2,048 features, and two residual additions. Token count and width remain unchanged through all blocks.

initialization. The ViT has about 38 million parameters, close to the 36 million of the convolutional network in the training-budget sweep. It uses the same batch size of 256 and learning rate of 0.005 as that network, with weight decay 0.1.

A.7 CelebA-HQ Evaluation and Decoded FID

CelebA-HQ FID in this paper is decoded FID: it compares generated images with autoencoder reconstructions of the training images. This appendix explains why we chose that reference and reports the main comparisons under both references.

The autoencoder introduces reconstruction error. For CelebA-HQ, we train diffusion models in the latent space of the pretrained KL-f8 autoencoder released by CompVis with Latent Diffusion Models (Rombach et al., 2022). We use the released `kl-f8` checkpoint[1], which encodes a 256×256 RGB image into a $32 \times 32 \times 4$ latent tensor. The encoder uses the deterministic posterior mode, and the decoder maps the latent back to pixels. Decoded outputs are clipped to $[-1, 1]$ and rescaled to $[0, 1]$ for evaluation. The autoencoder is frozen in all our comparisons.

The reconstructed image is close to the original, but it is not identical. Fine details can change even when no diffusion model is involved (Figure 10). Therefore, FID against the original images reflects both the quality of the diffusion model and the reconstruction error of the autoencoder. Especially for small training subsets, reconstruction error can dominate the score and obscure how well the diffusion model has learned its training distribution.

Two reference distributions. Let $X_N = \{x_i\}_{i=1}^N$ be a training subset, $R(x) = D(E(x))$ its deterministic autoencoder reconstruction, and $G = \{D(z_j)\}_{j=1}^M$ the decoded diffusion samples. We compare

$$\mathrm{FID}_{\mathrm{real}} = \mathrm{FID}(G, X_N), \tag{9}$$

$$\mathrm{FID}_{\mathrm{decoded}} = \mathrm{FID}(G, R(X_N)), \tag{10}$$

$$\mathrm{FID}_{\mathrm{reconstruction}} = \mathrm{FID}(R(X_N), X_N). \tag{11}$$

All three scores use pixel-space Inception features. In particular, decoded FID is not FID in latent space: both versions evaluate exactly the same generated images, but decoded FID also reconstructs the reference images. It measures agreement with the distribution that the frozen autoencoder provides to the diffusion experiment. If the generated distribution matches the reconstructed training distribution, its population decoded FID is zero, while its real-image FID still reflects the reconstruction discrepancy. FID has no additive decomposition into reconstruction error and diffusion error, so we read the reconstruction score as a reference value. It sits alongside the other two scores, and we never subtract it from them or treat it as a hard lower bound.

Table 5: CelebA-HQ reconstruction FID for the frozen KL-f8 autoencoder. Each score compares a training subset with reconstructions of exactly the same images. The 50-, 500-, 2,000-, and 5,000-image subsets use the training selection rule and subset seed 0; 28,000 is the complete training split. No diffusion samples enter this measurement.

Training images	50	500	2,000	5,000	28,000
Reconstruction FID	22.52	14.15	6.58	3.73	2.04

The effect is largest on small subsets. Table 5 reports reconstruction FID using the same Inception feature pipeline as the experiments. We select matching rows from the cached features of the full training split, using the same seed-0 permutation as training, and compute the empirical means and covariances in double precision. Each side contains N distinct, corresponding images. Each score measures the training subset the experiments actually use, so these are single measurements and not averages over fresh random subsets. The larger values at small N come from the finite-subset FID statistic, because the autoencoder is frozen and identical in every row.

At 50 images, reconstruction alone gives FID 22.52. Our Albis runs have mean real-image FID 22.76, but decoded FID 0.79. Reading 22.76 in isolation can therefore suggest a poor diffusion run even though its samples closely match the reconstructed training subset. The reconstruction FID falls to 2.04 over the full 28,000-image training split. Decoded FID makes the remaining mismatch with the reconstructed reference easier to interpret across training runs.

[1] `https://ommer-lab.com/files/latent-diffusion/kl-f8.zip`

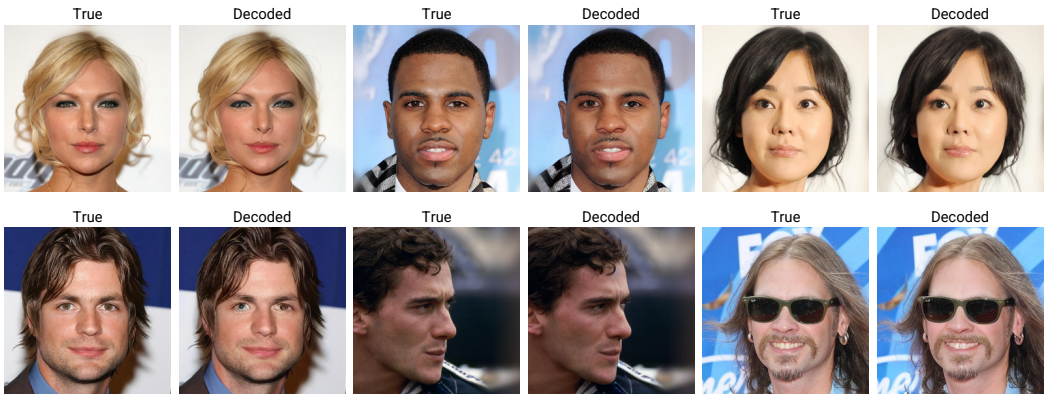

Figure 10: Original CelebA-HQ training images (True) beside their KL-f8 autoencoder reconstructions (Decoded). These are the first six images in the seed-0 training-subset ordering, selected without inspecting reconstruction quality. Every pair uses the same 256×256 image. The faces remain visually similar, while fine details can change. No diffusion model is used.

The main comparisons under both references. Both FIDs of a run are computed from the same generated samples and features. Figure 11 shows all six methods, all three small subsets, and all five training seeds from Table 2. Under both references, Albis has the lowest mean FID at 50 and 500 images. At 5,000 images, Flow (uniform) and EDM perform best and Albis no longer leads. Both the small-subset benefit and its limit at 5,000 images are therefore visible under either reference.

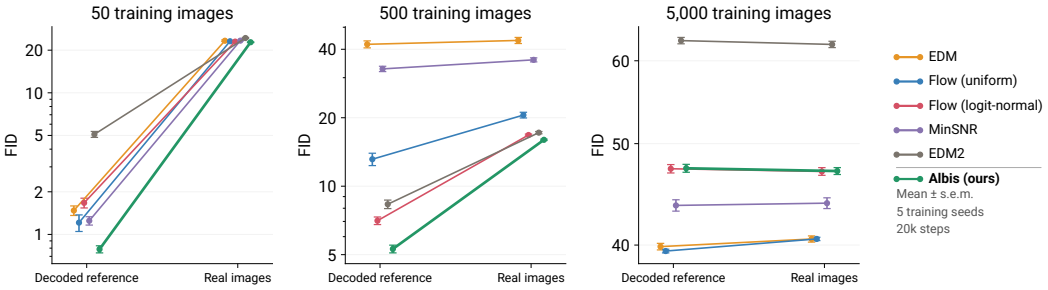

Figure 11: Decoded-reference and real-image FID for the CelebA-HQ small-subset comparison at 20k steps. Lines join the two scores for the same method and training subset; points and whiskers show means and s.e.m. over five training seeds. Each panel has a logarithmic FID axis. The real-image reference compresses the differences between well-fitted models on the 50-image subset.

Figure 12 repeats all 17 budgets and all three seeds from the CelebA-HQ scaling experiment in Figure 5. Both references show the same broad improvement with longer training, with Flow (uniform), Flow (logit-normal), and Albis reaching comparable low real-image FID near the reconstruction reference, ahead of EDM at the longest budget. Decoded FID separates their remaining mismatch more clearly: at 320k, Albis obtains 0.77 and Flow (logit-normal) obtains 0.97. The 20.6% reduction reported in the main text concerns decoded FID. These comparisons are why we use decoded FID to assess diffusion-model fit within a fixed autoencoder, and why we keep real-image FID in this appendix to show quality against the original pixels.



Figure 12: CelebA-HQ scaling with decoded-reference FID (left) and real-image FID (right), using the same generated samples for both. The figure shows all 17 independently trained budgets, four methods, and three training seeds on the 2,000-image training subset. Curves show means and s.e.m., with logarithmic training-step and FID axes. The dashed line is the measured reconstruction FID of 6.58, a reference value rather than a hard lower bound.

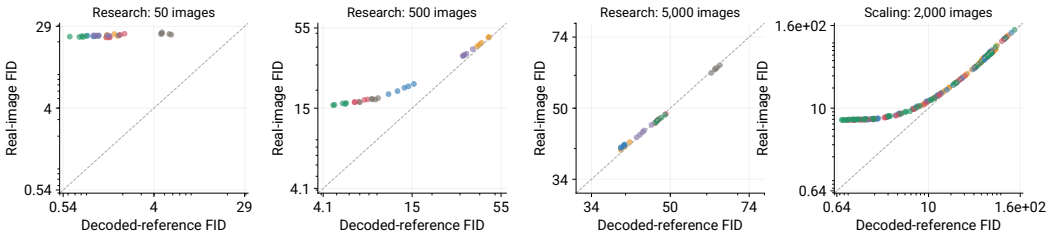

Figure 13: Direct comparison of decoded-reference FID (horizontal) and real-image FID (vertical) for every individual run: 90 small-subset evaluations across three subsets, and 204 scaling evaluations across all budgets. Colors follow Figures 11 and 12; the dashed diagonal marks equal numerical scores. Both axes are logarithmic. Near the reconstruction reference, a wide range of decoded FIDs maps to similar real-image FIDs.

B Framework Formulations and Controlled Matching

This appendix defines DDPM, EDM, Flow (uniform), Flow (logit-normal), MinSNR, and EDM2 as we implement them, and separates each original formulation from our adaptation of it. Every comparison holds the backbone and the training budget fixed within its set of experiments, so that noise sampling, loss weighting, and prediction parameterization are the things that vary. For DDPM, EDM, and Flow, each subsection also gives the controls of the framework comparison. Appendix B.10 relates that comparison to earlier ones.

B.1 Notation and the reference Flow objective

Let x_0 be centered data with unit per-feature variance and let $\epsilon \sim \mathcal{N}(0, I)$ be independent Gaussian noise. Class conditioning is suppressed in the equations. A squared norm $\|r\|^2_{\mathrm{MSE}} = d^{-1}\sum_{j=1}^{d} r_j^2$ is mean squared error over the d features of one example; training also averages over examples. We call the supervised quantity the *regression target*, and distinguish it from the residual target seen by a preconditioned network F_θ.

The data-to-noise coordinate α increases from clean data at 0 to noise at 1:

$$x_\alpha = (1-\alpha)x_0 + \alpha\epsilon, \qquad y_\alpha = \epsilon - x_0, \qquad q(\alpha) = (1-\alpha)^2 + \alpha^2. \tag{12}$$

Write $\ell_\theta(\alpha) = \mathbb{E}_{x_0,\epsilon}[\|v_\theta(x_\alpha, \alpha) - y_\alpha\|^2_{\mathrm{MSE}}]$. Our reference Flow objective has uniform noise sampling and unit velocity-loss weight:

$$\pi^{\mathrm{Flow}}(\alpha) = 1, \quad \lambda^{\mathrm{Flow}}(\alpha) = 1, \quad \mathcal{L}_{\mathrm{Flow}} = \int_0^1 \ell_\theta(\alpha)\, d\alpha. \tag{13}$$

For variance-preserving (VP) models, ω is the coefficient that DDPM writes as $\bar\alpha_t = \prod_{s\le t}(1-\beta_s)$ and DDIM writes as α_t. Define $a = \sqrt{\omega}$ and $b = \sqrt{1-\omega}$, so $z_\omega = ax_0 + b\epsilon$ and $a^2 + b^2 = 1$. For EDM, $\sigma > 0$ is the standard deviation of additive noise, $s = \sigma_{\mathrm{data}} = 0.5$ is the clean-data standard deviation, and $z_\sigma = sx_0 + \sigma\epsilon$. Matching signal-to-noise ratios (SNRs) gives

$$\mathrm{SNR} = \frac{(1-\alpha)^2}{\alpha^2} = \frac{\omega}{1-\omega} = \frac{s^2}{\sigma^2}, \tag{14}$$

$$\omega = \frac{(1-\alpha)^2}{q(\alpha)}, \quad \sigma = \frac{s\alpha}{1-\alpha}, \quad \alpha = \frac{b}{a+b} = \frac{\sigma}{s+\sigma}, \tag{15}$$

$$z_\omega = \frac{x_\alpha}{\sqrt{q(\alpha)}}, \qquad z_\sigma = (s+\sigma)x_\alpha. \tag{16}$$

The densities and velocity-space loss weights are therefore

$$\pi(\alpha) = \pi_\omega \frac{2\alpha(1-\alpha)}{q(\alpha)^2} = \pi_\sigma \frac{s}{(1-\alpha)^2}, \tag{17}$$

$$\lambda(\alpha) = (1-\alpha)^2\lambda_\epsilon = s^2\alpha^2\lambda_D = q(\alpha)\lambda_{v_{\mathrm{VP}}}. \tag{18}$$

Here λ_ϵ, λ_D, and $\lambda_{v_{\mathrm{VP}}}$ multiply errors in noise, scaled clean-data, and VP-velocity predictions, respectively. They remain different weights at the same SNR, which is why matching Equation 13 in general takes both the density and the native weight together.

Finite support and the meaning of matching. The density formulas describe the interior of the noise range. We draw the uniform latent variable on the interval $[10^{-8}, 1-10^{-8}]$ before applying each baseline's transform, so the distributions below are bounded and carry the corresponding normalization constant. Where a framework clips further, the clip puts probability mass on an endpoint, and an equality to full-range Flow then holds over the interior only. In the framework comparison, DDPM and EDM match both the noise sampling and the loss weight of Flow, and Figure 1 reports these joint matching controls.

B.2 DDPM

Original formulation. Ho et al. (2020) use $T = 1000$ discrete timesteps, linear variances $\beta_k = 10^{-4} + (k-1)(0.02 - 10^{-4})/(T-1)$, and $\bar{\alpha}_k = \prod_{j=1}^{k}(1-\beta_j)$. The forward transition and simplified training objective are

$$q(z_k \mid z_{k-1}) = \mathcal{N}(\sqrt{1-\beta_k}\, z_{k-1}, \beta_k I), \tag{19}$$

$$z_k = \sqrt{\bar{\alpha}_k} x_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, \quad k \sim \mathrm{Uniform}\{1,\dots,T\}, \tag{20}$$

$$\mathcal{L}_{\mathrm{simple}} = \mathbb{E}_{k,x_0,\epsilon}[\|\hat{\epsilon}_\theta(z_k,k) - \epsilon\|^2_{\mathrm{MSE}}]. \tag{21}$$

The regression target is therefore ϵ and the simplified loss weight is 1. Original image inputs are scaled to $[-1,1]$, while our shared preprocessing centers and standardizes the features, because the cross-framework SNR and normalization identities assume unit per-feature variance. Original generation uses ancestral sampling: it starts from $z_T \sim \mathcal{N}(0,I)$ and adds fresh Gaussian noise at every reverse step except the last.

Our training implementation. We retain noise prediction and the simplified unit-weight MSE, but use the continuous linear-β schedule

$$\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min})t, \quad t \sim \mathcal{U}(0,1), \tag{22}$$

$$\omega(t) = \exp\left[-1000\left(\beta_{\min} t + \tfrac{1}{2}(\beta_{\max} - \beta_{\min})t^2\right)\right], \tag{23}$$

$$z_t = a(t)x_0 + b(t)\epsilon, \quad \hat{\epsilon}_\theta = F_\theta(z_t, \omega(t)), \quad y_t = \epsilon, \quad \lambda_\epsilon = 1, \tag{24}$$

where $\beta_{\min} = 10^{-4}$ and $\beta_{\max} = 0.02$. This exponential integral approximates the discrete product in continuous time, and the two agree only in the limit of fine discretization. The network receives ω instead of the original discrete timestep. The induced density and effective velocity weight are

$$\pi_\omega(\omega) = \frac{1}{1000\beta(t(\omega))\omega}, \quad \omega(1) < \omega < 1, \qquad \lambda(\alpha) = (1-\alpha)^2. \tag{25}$$

We move to continuous time because it lets us select the same noise levels α for inference in every framework. We use the continuous VP schedule associated with the score-based SDE formulation (Song et al., 2021b) and the deterministic DDIM update below. Conditioning on ω identifies the signal/noise mixture directly, and it keeps that identification when the timestep grid changes. The continuous cosine schedule we use for MinSNR follows the same reasoning.

Regression-target normalization. The default directly regresses ϵ. The conditional-normalization control instead forms

$$\hat{\epsilon}_\theta = bz_t + aF_\theta(z_t,\omega), \qquad F^*_{\mathrm{DDPM}} = \frac{\epsilon - bz_t}{a}, \qquad \mathbb{V}[\epsilon - bz_t \mid \omega] = a^2 = \omega. \tag{26}$$

This removes the predictable linear part of the regression target and gives the residual unit variance, which matters most where the unnormalized residual becomes small. The external regression target remains ϵ, input scaling stays at $c_{\mathrm{in}} = 1$, and conditioning stays ω. The same output conversion is applied at inference.

Noise sampling and loss weights to match Flow. With $\Delta\beta = \beta_{\max} - \beta_{\min}$, the inverse continuous schedule is

$$t(\alpha) = \frac{-\beta_{\min} + \sqrt{\beta^2_{\min} - 2\Delta\beta \log \omega(\alpha)/1000}}{\Delta\beta}. \tag{27}$$

The joint control draws $\alpha \sim \mathcal{U}(0,1)$, uses $t(\alpha)$, and multiplies noise MSE by

$$C_\epsilon(\alpha) = \frac{(a+b)^2}{a^2} = \frac{1 + 2\sqrt{\omega(1-\omega)}}{\omega} = \frac{1}{(1-\alpha)^2}. \tag{28}$$

This weight converts noise-prediction error to velocity error, because $\hat{v} - y_\alpha = (\hat{\epsilon} - \epsilon)/(1-\alpha)$. In native coordinates, the matched density is $\pi^{\mathrm{match}}_\omega = q(\alpha)^2/[2\alpha(1-\alpha)]$.

The implementation clips Equation 27 to $[0, 1]$ and then recomputes $\omega(t)$. The linear-β schedule never reaches pure noise, because its noisiest level $\omega(1) = e^{-10.05}$ stays above zero. Every requested α above

$$\alpha_{\text{cap}} = \frac{\sqrt{1 - e^{-10.05}}}{e^{-10.05/2} + \sqrt{1 - e^{-10.05}}} \simeq 0.99347 \tag{29}$$

therefore produces the same training noise level, and the matched weight is evaluated at that capped level. The clip puts probability mass on this endpoint, while matched inference reaches $\alpha = 0.999$. Exact objective matching therefore covers the interior and stops at this clipped tail.

Native and Flow-matched inference. In the small-subset experiments, the default uses deterministic DDIM-style updates (Song et al., 2021a) with $N = 128$, native nodes $t_i = 1 - i/N$, $\omega_i = \omega(t_i)$, and initialization $z_0 = \epsilon$:

$$\hat{x}_{0,i} = \frac{z_i - b_i \hat{\epsilon}_i}{a_i}, \qquad z_{i+1} = a_{i+1}\hat{x}_{0,i} + b_{i+1}\hat{\epsilon}_i. \tag{30}$$

Here the subscript i indexes generation nodes, and it runs opposite to forward diffusion time. The final node has $\omega_N = 1$. The inference-matching control retains this update but replaces the grid by

$$\alpha_i = 0.999(1 - i/N), \qquad \omega_i = \frac{(1 - \alpha_i)^2}{q(\alpha_i)}, \qquad i = 0, \ldots, N. \tag{31}$$

The initial standard Gaussian approximates the finite-noise marginal, and the 0.999 cap keeps clean-data reconstruction away from a division by zero.

B.3 EDM

Original formulation and our regression target. EDM separates preconditioning, loss weighting, noise sampling, and numerical integration (Karras et al., 2022). We explicitly multiply standardized data by $s = \sigma_{\text{data}}$ so that the clean target has that scale:

$$z_\sigma = sx_0 + \sigma\epsilon, \qquad y_\sigma = sx_0, \qquad D_\theta(z_\sigma, \sigma) = c_{\text{skip}} z_\sigma + c_{\text{out}} F_\theta(c_{\text{in}} z_\sigma, c_{\text{noise}}). \tag{32}$$

We retain EDM's preconditioning coefficients and residual regression target:

$$c_{\text{skip}} = \frac{s^2}{s^2 + \sigma^2}, \qquad c_{\text{out}} = \frac{s\sigma}{\sqrt{s^2 + \sigma^2}}, \tag{33}$$

$$c_{\text{in}} = \frac{1}{\sqrt{s^2 + \sigma^2}}, \qquad c_{\text{noise}} = \frac{\log \sigma}{4}, \tag{34}$$

$$F^*_{\text{EDM}} = \frac{sx_0 - c_{\text{skip}} z_\sigma}{c_{\text{out}}}, \qquad \mathbb{V}[sx_0 - c_{\text{skip}} z_\sigma \mid \sigma] = \frac{s^2\sigma^2}{s^2 + \sigma^2}. \tag{35}$$

EDM therefore already has the conditional residual normalization that we add as an extra control for DDPM and Flow, which is why its column is unchanged at that stage of Figure 1.

Default noise sampling and loss. Both the original default and our baseline use

$$\log \sigma \sim \mathcal{N}(m, r^2), \quad m = -1.2, \quad r = 1.2, \tag{36}$$

$$\pi_\sigma(\sigma) = \frac{1}{\sigma r\sqrt{2\pi}} \exp\left[-\frac{(\log \sigma - m)^2}{2r^2}\right], \tag{37}$$

$$\lambda^{\text{EDM}}_D(\sigma) = \frac{\sigma^2 + s^2}{(s\sigma)^2} = c^{-2}_{\text{out}}, \qquad \mathcal{L}_{\text{EDM}} = \mathbb{E}[\lambda^{\text{EDM}}_D \, \|D_\theta - sx_0\|^2_{\text{MSE}}]. \tag{38}$$

In code, a uniform latent u is transformed by $\sigma = \exp[m + r\Phi^{-1}(u)]$, where Φ is the standard normal CDF. In Flow coordinates these choices become

$$\pi^{\text{EDM}}(\alpha) = \frac{\exp[-(\log(s\alpha/(1-\alpha)) - m)^2/(2r^2)]}{r\sqrt{2\pi}\alpha(1-\alpha)}, \qquad \lambda^{\text{EDM}}(\alpha) = q(\alpha). \tag{39}$$

This is logit-normal sampling in the common noise coordinate: $\alpha = \sigma/(s + \sigma)$ and $\text{logit}(\alpha) = \log \sigma - \log s \sim \mathcal{N}(m - \log s, r^2)$.

Noise sampling and loss weights to match Flow. The joint control replaces the logit-normal draw of α by $\alpha \sim \mathcal{U}(0,1)$ and $\sigma = s\alpha/(1-\alpha)$. Its density in native coordinates and denoiser-loss weight are

$$\pi_\sigma^{\text{match}}(\sigma) = \frac{s}{(s+\sigma)^2}, \qquad C_D(\sigma) = \frac{(s+\sigma)^2}{s^2\sigma^2} = \frac{1}{s^2\alpha^2}. \tag{40}$$

The regression target and preconditioning stay as in Equations 32–35. Because $\hat{v} - y_\alpha = -(D_\theta - sx_0)/(s\alpha)$, this weight turns the denoiser loss into uniform velocity MSE.

Original inference and our inference. For EDM's probability-flow ODE, $dz/d\sigma = (z - D_\theta(z,\sigma))/\sigma$. The native grid is

$$\sigma_i = \left[\sigma_{\max}^{1/\rho} + \frac{i}{N-1}(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho})\right]^\rho, \quad i = 0,\dots,N-1, \qquad \sigma_N = 0, \tag{41}$$

with $\rho = 7$, $\sigma_{\min} = 0.002$, and $\sigma_{\max} = 80$ in our default. EDM's deterministic sampler integrates this ODE with Heun's second-order method, and the paper also describes a stochastic variant that adds fresh noise before each step. Our samplers for the small-subset and training-budget experiments use deterministic first-order Euler throughout:

$$z_0 = \sigma_{\max}\epsilon, \qquad z_{i+1} = z_i + (\sigma_{i+1} - \sigma_i)\frac{z_i - D_\theta(z_i,\sigma_i)}{\sigma_i}, \qquad x_{\text{generated}} = z_N/s. \tag{42}$$

This takes one denoiser evaluation per step, with $N = 128$, and both the default and the Flow-matched EDM evaluations use it. The ImageNet EDM arm also uses Euler, after converting denoiser predictions to Flow velocity (Appendix B.8).

Flow-matched inference. The inference control of the small-subset framework comparison replaces Equation 41 by

$$\alpha_i = 0.999(1 - i/N), \quad \sigma_i = \frac{s\alpha_i}{1-\alpha_i}, \quad \sigma_0 = 499.5, \quad \sigma_N = 0, \tag{43}$$

and initializes with $z_0 = 499.5\epsilon$. It retains the Euler update and the final division by s. The finite cap keeps the sampler away from the singular value $\sigma(1) = \infty$, and the initial noise scaling moves with the grid, because the initial state has to match the first grid node.

B.4 Flow

Original formulation. Flow matching regresses the vector field of a chosen conditional probability path (Lipman et al., 2023). For its linear conditional path, let τ run from noise to data and let σ_{end} denote the small residual noise at the data endpoint. The original path, regression target, and conditional flow-matching loss are

$$x_\tau = \tau x_0 + [1 - (1-\sigma_{\text{end}})\tau]\epsilon, \quad y_\tau = x_0 - (1-\sigma_{\text{end}})\epsilon, \tag{44}$$

$$\tau \sim \mathcal{U}(0,1), \qquad \mathcal{L}_{\text{CFM}} = \mathbb{E}[\|f_\theta(x_\tau,\tau) - y_\tau\|^2_{\text{MSE}}]. \tag{45}$$

Generation solves $dx/d\tau = f_\theta(x,\tau)$ from a standard Gaussian; the paper evaluates numerical ODE solvers, including adaptive Dormand–Prince and fixed-step Euler. The linear-path objective is also related to rectified flow (Liu et al., 2023).

Our implementation and regression-target normalization. We use the zero-endpoint-noise path, $\sigma_{\text{end}} = 0$, and reverse the coordinate, $\alpha = 1 - \tau$. This gives exactly Equation 12, with clean data as its endpoint and velocity $\epsilon - x_0$. The default network directly predicts this regression target with unit loss weight and uniform α. The conditional-normalization control is

$$v_\theta(x_\alpha,\alpha) = \frac{2\alpha - 1}{q(\alpha)}x_\alpha + \frac{F_\theta(x_\alpha,\alpha)}{\sqrt{q(\alpha)}}, \qquad F^*_{\text{Flow}} = \sqrt{q(\alpha)}\left[y_\alpha - \frac{2\alpha-1}{q(\alpha)}x_\alpha\right]. \tag{46}$$

The unscaled residual variance is $1/q(\alpha)$, so F^*_{Flow} has unit variance. As in DDPM, the external regression target is unchanged. We retain $c_{\text{in}} = 1$ and native α conditioning, so this control normalizes the residual and leaves the input as it is.

Sampling, loss weighting, and inference. Flow defines the reference, so both default and matched sampling/weighting are $\pi(\alpha) = \lambda(\alpha) = 1$. In the baseline comparisons outside the framework matching, we call this default Flow (uniform); the matching experiment changes only the optional residual normalization. In the small-subset experiments, generation uses

$$\alpha_i = 1 - i/N, \quad x^{(0)} = \epsilon, \quad x^{(i+1)} = x^{(i)} + (\alpha_{i+1} - \alpha_i) v_\theta(x^{(i)}, \alpha_i), \qquad N = 128. \tag{47}$$

The terminal node $\alpha_N = 0$ needs no network call. The ImageNet implementation stores a noise-to-data velocity model. During training, the adapter feeds it $1 - \alpha$ and negates its output to recover our data-to-noise regression target.

B.5 Flow (logit-normal)

Flow (logit-normal) denotes Flow with fixed logit-normal noise sampling, $\mathrm{logit}(\alpha) \sim \mathcal{N}(0, 1)$, and unit loss weight, as in SD3 (Esser et al., 2024). Apart from this sampling distribution, it uses the Flow regression target and the Flow inference conventions described above.

B.6 MinSNR

Original formulation. Min-SNR-γ was proposed to reduce conflicts among timestep-specific training tasks (Hang et al., 2023). For VP noising $z_t = a_t x_0 + b_t \epsilon$, $a_t^2 + b_t^2 = 1$, and $\mathrm{SNR}_t = a_t^2 / b_t^2$, the same clamped-SNR objective has different weights for different regression targets:

$$\lambda_{x_0} = \min(\mathrm{SNR}, \gamma), \quad \lambda_\epsilon = \frac{\min(\mathrm{SNR}, \gamma)}{\mathrm{SNR}}, \quad \lambda_{v_{\mathrm{VP}}} = \frac{\min(\mathrm{SNR}, \gamma)}{\mathrm{SNR} + 1}. \tag{48}$$

Here $v_{\mathrm{VP}} = a_t \epsilon - b_t x_0$ is the VP-velocity regression target (Salimans & Ho, 2022). The original experiments use a discrete cosine schedule with $T = 1000$, EMA models, and EDM's Heun sampler for generation. The weight is a fixed function of the SNR.

Our schedule, regression target, and loss. We select the VP-velocity variant with $\gamma = 5$ and use the continuous Nichol–Dhariwal cosine schedule (Nichol & Dhariwal, 2021). With offset $\delta = 0.008$, define

$$\vartheta(t) = \frac{\pi(t + \delta)}{2(1 + \delta)}, \quad f_0 = \cos^2 \vartheta(0), \quad t \sim \mathcal{U}(0, 1), \tag{49}$$

$$\omega_{\cos}(t) = \frac{\cos^2 \vartheta(t)}{f_0}, \quad \tilde{\omega}(t) = \mathrm{clip}(\omega_{\cos}(t), 10^{-8}, 1 - 10^{-8}), \tag{50}$$

$$a(t) = \sqrt{\tilde{\omega}(t)}, \quad b(t) = \sqrt{1 - \tilde{\omega}(t)}, \quad z_t = a(t) x_0 + b(t) \epsilon, \tag{51}$$

$$y_t = a(t) \epsilon - b(t) x_0, \quad V_\theta = F_\theta(z_t, \tilde{\omega}(t)), \tag{52}$$

$$\mathcal{L}_{\mathrm{MinSNR}} = \mathbb{E}_{t, x_0, \epsilon} \left[\frac{\min(a(t)^2 / b(t)^2, 5)}{a(t)^2 / b(t)^2 + 1} \left\| V_\theta - y_t \right\|_{\mathrm{MSE}}^2 \right]. \tag{53}$$

The network predicts VP velocity directly, with no skip/output preconditioning and $c_{\mathrm{in}} = 1$; its regression target already has unit conditional variance. As in our DDPM arm, conditioning uses ω instead of the original timestep. Clipping bounds the signal/noise coefficients at the endpoints.

On the unclipped interior, the induced native density is

$$\left| \frac{d\omega}{dt} \right| = \frac{\pi \sin(2\vartheta(t))}{2(1 + \delta) f_0}, \qquad \pi_\omega(\omega) = \left| \frac{d\omega}{dt} \right|^{-1}_{t = t(\omega)}. \tag{54}$$

Equation 17 gives its Flow-coordinate density. The effective velocity-space weight is

$$\lambda^{\mathrm{MinSNR}}(\alpha) = q(\alpha) \frac{\min((1 - \alpha)^2 / \alpha^2, 5)}{(1 - \alpha)^2 / \alpha^2 + 1} = \min((1 - \alpha)^2, 5\alpha^2). \tag{55}$$

The factor $q(\alpha)$ converts the weight on VP-velocity error into a weight on Flow-velocity error.

Conversion to Flow velocity. For common noise-level evaluation, map the requested Flow level by

$$\omega(\alpha) = \frac{(1-\alpha)^2}{q(\alpha)}, \qquad t(\alpha) = \frac{2(1+\delta)}{\pi} \arccos \sqrt{f_0 \omega(\alpha)} - \delta, \tag{56}$$

with the implementation's endpoint clipping. Then reconstruct

$$\hat{x}_0 = a z_\omega - b V_\theta, \quad \hat{\epsilon} = b z_\omega + a V_\theta, \quad \hat{v} = \hat{\epsilon} - \hat{x}_0, \quad \|\hat{v} - y_\alpha\|^2_{\mathrm{MSE}} = (a+b)^2 \|V_\theta - v_{\mathrm{VP}}\|^2_{\mathrm{MSE}}. \tag{57}$$

Common loss evaluation therefore scores MinSNR by multiplying its VP-velocity MSE by $C_{\mathrm{VP}} = (a+b)^2 = 1/q(\alpha)$.

Native and Flow-matched inference. In the small-subset experiments, generation starts from $z_0 = \epsilon$ and uses $t_i = 1 - i/N$, $\tilde{\omega}_i = \tilde{\omega}(t_i)$, and the deterministic update

$$\hat{x}_{0,i} = a_i z_i - b_i V_\theta(z_i, \tilde{\omega}_i), \quad \hat{\epsilon}_i = b_i z_i + a_i V_\theta(z_i, \tilde{\omega}_i), \quad z_{i+1} = a_{i+1} \hat{x}_{0,i} + b_{i+1} \hat{\epsilon}_i. \tag{58}$$

The terminal $\tilde{\omega}_N$ is capped at $1 - 10^{-8}$ instead of reaching exactly 1. ImageNet uses the reconstructed Flow velocity from Equation 57 in the common solver below.

B.7 EDM2

Original paper and why we include this baseline. EDM2 is primarily a study of network architecture and training dynamics, with changes that preserve activation, weight, and update magnitudes, together with post-hoc EMA reconstruction (Karras et al., 2024). It also introduces an adaptive loss, which learns a noise-dependent loss weight and keeps the noise distribution fixed. EDM2 adds this loss in its improved baseline configuration and keeps it in every later configuration, including the final models. We include this component as a baseline because it learns the loss allocation during training, and we reproduce it on the same backbone as the other methods. The EDM2 architecture and its training and EMA recipe stay outside our reproduction.

Regression target and complete loss. EDM2 keeps the EDM regression target sx_0, and we retain the EDM denoiser and all four preconditioning coefficients in Equations 32–35. EDM2 changes the default noise distribution and introduces a learned log-uncertainty $h_\phi(\sigma)$:

$$\log \sigma \sim \mathcal{N}(-0.4, 1^2), \quad \pi^{\mathrm{EDM2}}_\sigma(\sigma) = \frac{e^{-(\log\sigma + 0.4)^2/2}}{\sigma\sqrt{2\pi}}, \tag{59}$$

$$\lambda^{\mathrm{EDM2}}_D(\sigma;\phi) = \frac{\sigma^2 + s^2}{(s\sigma)^2} e^{-h_\phi(\sigma)}, \tag{60}$$

$$\mathcal{L}_{\mathrm{EDM2}}(\theta,\phi) = \mathbb{E}_{\sigma,x_0,\epsilon}\left[\lambda^{\mathrm{EDM2}}_D(\sigma;\phi)\, \|D_\theta - s x_0\|^2_{\mathrm{MSE}} + h_\phi(\sigma)\right]. \tag{61}$$

These noise-distribution parameters and the complete loss match the official EDM2 loss implementation. The additive term is essential, because multiplying the MSE by a learned positive weight alone would let the model drive the loss down by shrinking the weight. We optimize the denoiser and the uncertainty head jointly.

Our uncertainty head and its difference from the official network. The head takes $c = \log\sigma/4$ and uses $K = 128$ fixed random Fourier features of bandwidth one:

$$f_j \sim \mathcal{N}(0,1), \quad \psi_j \sim \mathcal{U}(0, 2\pi), \quad H_j(c) = \sqrt{2}\cos(2\pi f_j c + \psi_j), \qquad h_\phi(\sigma) = w^\top H(c) + w_0. \tag{62}$$

Our ordinary linear layer has a bias w_0, no weight normalization, and initialization $w = 0$, $w_0 = 0$. Training therefore starts at $h_\phi = 0$, which recovers the EDM-weighted MSE under EDM2's noise distribution. Frequencies and phases are fixed buffers. In the official final EDM2 network, the same Fourier-feature construction feeds a bias-free magnitude-preserving layer with random initialization and normalized weights. Our loss functional and Fourier inputs therefore follow EDM2, and the head's parameterization differs. We use an ordinary affine head because this experiment studies noise scheduling and loss allocation.

Relationship to Flow and adaptive sampling. Using $\sigma = s\alpha/(1-\alpha)$, the fixed density is Equation 39 with $(m, r) = (-0.4, 1)$, and the complete loss becomes

$$\mathcal{L}_{\mathrm{EDM2}} = \int_0^1 \pi^{\mathrm{EDM2}}(\alpha) \left[q(\alpha) e^{-h_\phi(\sigma(\alpha))} \ell_\theta(\alpha) + h_\phi(\sigma(\alpha)) \right] d\alpha. \tag{63}$$

At an unconstrained optimum of the uncertainty function for a fixed denoiser,

$$e^{h^*(\sigma)} = \lambda_D^{\mathrm{EDM}}(\sigma) \mathbb{E}_{x_0,\epsilon}[\|D_\theta - s x_0\|^2_{\mathrm{MSE}} \mid \sigma]. \tag{64}$$

This optimum gives the learned weight its inverse-loss interpretation, and a finite head approaches it only approximately. For a frozen head, the MSE term can be rewritten as unit-weight velocity MSE under an effective density proportional to $\pi^{\mathrm{EDM2}} q e^{-h_\phi}$, up to normalization. The rewrite is purely algebraic. EDM2 still draws its noise levels from the fixed logit-normal distribution in α, while Albis changes the sampling density itself and applies an importance correction. Comparing EDM2 with Albis therefore separates adaptive weighting from adaptive sampling.

Native and Flow-matched inference. The uncertainty head is used in training alone. The original final EDM2 models use deterministic Heun sampling, while our EDM2 arm in the small-subset experiments uses the native EDM grid and the Euler update in Equations 41 and 42, initializing with 80ϵ and dividing the final state by s. In ImageNet, EDM2 uses the same denoiser-to-velocity adapter and common inference grid as EDM (Appendix B.8).

B.8 Shared ImageNet inference and common loss evaluation

The ImageNet experiments use a shared velocity ODE for EDM, Flow (uniform), Flow (logit-normal), MinSNR, EDM2, and Albis; DDPM appears only in the framework comparison on the small subsets. Let $\tau = 1 - \alpha$ increase from noise to data, let X denote the Flow state, and write $g_\theta(X, \tau) = \hat{x}_0 - \hat{\epsilon} = -\hat{v}$. The velocity adapters are

$$g_{\mathrm{EDM/EDM2}} = \frac{s+\sigma}{s\sigma} D_\theta(z_\sigma, \sigma) - \frac{z_\sigma}{\sigma}, \quad z_\sigma = \frac{sX}{1-\alpha}, \quad \sigma = \frac{s\alpha}{1-\alpha}, \tag{65}$$

$$g_{\mathrm{Flow}} = -v_\theta(X, 1-\tau), \tag{66}$$

$$g_{\mathrm{MinSNR}} = (a-b) z_\omega - (a+b) V_\theta(z_\omega, \omega), \quad z_\omega = \frac{X}{\sqrt{q(\alpha)}}. \tag{67}$$

For EDM/EDM2 the adapter uses $\alpha = \mathrm{clip}(1-\tau, 10^{-6}, 0.999)$, including at the first solver node $\tau = 0$, where the uncapped EDM coordinate would be infinite. MinSNR clips ω to $[10^{-8}, 1-10^{-8}]$. The underlying network receives its training-time conditioning ($\log \sigma/4$ or ω).

The sampler uses $M = 250$ grid nodes. With shift $\kappa = 0.3$,

$$r_i = \frac{i}{M-1}, \quad \tau_i = \frac{\kappa r_i}{1 + (\kappa - 1) r_i}, \quad i = 0, \ldots, M-1, \qquad X_{i+1} = X_i + (\tau_{i+1} - \tau_i) g_\theta(X_i, \tau_i). \tag{68}$$

The 250-node Euler trajectory therefore takes 249 updates. All methods start at $X_0 \sim \mathcal{N}(0, I)$ and share this grid and solver. The convention differs from the small-subset sampler, which takes $N = 128$ updates over $N + 1 = 129$ nodes. Sharing one solver is what isolates the training-component differences from the original papers' sampler choices.

Guided generation follows the LightningDiT implementation (Yao et al., 2025), including its guidance interval and its restriction of guidance to the first three latent channels. Classifier-free guidance combines conditional and unconditional predictions (Ho & Salimans, 2021). Restricting guidance to selected noise intervals has also been studied (Kynkäänniemi et al., 2024). For the first three latent channels, the sampler uses conditional velocity g_c until $\tau = 0.125$, then CFG scale 6.7:

$$g_{\mathrm{guided},j} = \begin{cases} g_{c,j}, & \tau < 0.125, \\ g_{\emptyset,j} + 6.7(g_{c,j} - g_{\emptyset,j}), & \tau \geq 0.125, \end{cases} \qquad j \leq 3. \tag{69}$$

The remaining channels keep the conditional prediction. Unguided evaluation uses the conditional model with the CFG extrapolation switched off.

Finally, common loss evaluation reports the unweighted linear Flow-velocity MSE at the same noise levels α. The native squared-error multipliers are $C_\epsilon = 1/(1-\alpha)^2$ for DDPM, $C_D = 1/(s^2\alpha^2)$ for EDM, 1 for Flow, $C_{\mathrm{VP}} = 1/q(\alpha)$ for MinSNR, and C_D for EDM2. These evaluation conversions leave out MinSNR's clamped-SNR weight and EDM2's learned uncertainty term, because both belong to the training objective and not to the prediction being scored. The conversions put every reported loss on one scale.

B.9 Cumulative Framework-Matching Results

Figure 1 plots the four stages of the cumulative controls. Table 6 gives the numbers behind that figure, one block per stage: the defaults, then matching the noise objective, then adding conditional normalization, and finally adding the common inference-grid construction. Each cell reports five-seed mean FID and s.e.m. against the training subset, and CelebA-HQ FID is computed after decoding to pixels.

Two details help when reading across the blocks. First, matching the noise objective moves noise sampling and effective velocity-error weighting together, because changing either one alone still leaves a different objective (Appendix B). Second, EDM already builds conditional normalization into its preconditioning, so its rows stay fixed at that stage while DDPM and Flow change. Flow is the reference, so its rows also stay fixed when the noise objective and the inference grid are matched. The endpoint caps survive the final stage: DDPM clips its inverse schedule near $\alpha = 0.993$, as Equation 29 makes precise, so the matched training supports agree over the interior of the noise range and part ways in the extreme high-noise tail.

Table 6: Cumulative framework-matching FID after 20,000 training steps. Each block adds one control to the block above it: (a) default settings, (b) matching the noise objective (noise sampling and effective velocity-error weighting), (c) additionally matching conditional normalization, and (d) additionally using the common inference-grid construction. Each cell is the five-seed mean $\pm$ s.e.m.

	CIFAR-10			CelebA-HQ		
Method	50	500	5,000	50	500	5,000
(a) Defaults						
DDPM	5.06 ± 0.32	151.54 ± 3.09	49.35 ± 0.40	13.17 ± 0.83	129.27 ± 1.27	47.52 ± 0.58
EDM	4.31 ± 0.20	25.14 ± 0.52	42.36 ± 0.39	1.48 ± 0.11	42.06 ± 1.49	39.86 ± 0.29
Flow	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
(b) Match noise						
DDPM	25.24 ± 3.79	152.86 ± 1.25	80.28 ± 1.53	40.64 ± 5.17	132.19 ± 2.75	113.86 ± 2.21
EDM	1.30 ± 0.06	9.82 ± 0.58	45.03 ± 0.16	0.76 ± 0.05	11.06 ± 0.67	41.62 ± 0.16
Flow	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
(c) + Conditional normalization						
DDPM	2.01 ± 0.05	11.19 ± 0.58	46.04 ± 0.17	1.18 ± 0.07	12.47 ± 0.60	40.86 ± 0.55
EDM	1.30 ± 0.06	9.82 ± 0.58	45.03 ± 0.16	0.76 ± 0.05	11.06 ± 0.67	41.62 ± 0.16
Flow	1.64 ± 0.10	11.42 ± 0.54	45.24 ± 0.34	0.90 ± 0.10	11.92 ± 0.64	41.21 ± 0.35
(d) + Common inference						
DDPM	1.78 ± 0.05	11.76 ± 0.63	45.48 ± 0.14	1.07 ± 0.06	13.17 ± 0.63	41.55 ± 0.58
EDM	1.90 ± 0.09	11.23 ± 0.55	45.36 ± 0.17	0.82 ± 0.05	11.19 ± 0.75	41.90 ± 0.14
Flow	1.64 ± 0.10	11.42 ± 0.54	45.24 ± 0.34	0.90 ± 0.10	11.92 ± 0.64	41.21 ± 0.35

B.10 Related Framework Comparisons

Earlier work has also compared frameworks by taking them apart into components. Karras et al. (2022) write VP, VE, iDDPM, and DDIM in one common framework. They then change the sampler, the preconditioning, the loss weighting, and the noise distribution step by step, and measure the effect of each change. Our cumulative matching uses the same component view. Their goal is a better recipe. Our goal is to test whether DDPM, EDM, and Flow still differ once their components agree (Table 6).

Gagneux et al. (2026) study two of these components within Flow alone. They separate the loss weight from the output parameterization and find that the best weight is consistent across settings, while the best parameterization depends on the architecture and the data regime.

Kingma & Gao (2023) give the theory behind this kind of matching. They write the common diffusion objectives as weighted integrals over noise levels, and in that view the noise schedule acts as importance sampling and leaves the objective unchanged. Two frameworks with the same product of sampling density and velocity-space loss weight therefore optimize the same objective, and this product is what our noise-objective matching aligns.

C Fixed-Schedule Controls and Curriculum Evidence

C.1 Fixed Noise-Range Tilts

Before we change the schedule during training, we check a simpler option: a fixed tilt toward one part of the noise range. If one tilt improved FID across datasets and subset sizes, it would be an easy schedule to recommend. We call $\alpha \in [0, 0.5]$ the low-noise half and $\alpha \in [0.5, 1]$ the high-noise half; larger α means more noise. We also test emphasis on the middle, $\alpha \in [0.25, 0.75]$, and on its complement, the tails.

Fixed Flow controls. Starting from Flow (uniform), we give one part of the range three times the emphasis of the rest throughout all 20k training steps. *Sampling* draws noise levels from the tilted density and keeps unit weights. *Weighting* draws noise levels uniformly and uses the tilted density as the loss weight. Both change the training objective in the same way, so we read the preferred direction from these two. *Importance sampling* draws from the tilted density and divides by it. It therefore keeps the Flow (uniform) objective in expectation, and Appendix C.4 discusses how it still changes the result. The middle and tail rows are the constant-emphasis diagonal of the curriculum grid in Appendix C.3. Tables 7 and 8 report the five-seed results.

Table 7: Fixed Flow tilts on CIFAR-10 after 20k steps, holding emphasis on the low-noise half, the high-noise half, the middle, or the tails. Cells are training-subset FID ± s.e.m. over five seeds. Bold marks the lowest displayed mean in each column.

Intervention	Emphasis	50 images	500 images	5,000 images
Uniform	—	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07
Sampling	Low noise	2.13 ± 0.12	33.61 ± 1.11	41.12 ± 0.21
Sampling	High noise	1.64 ± 0.06	$\mathbf{9.47 \pm 0.40}$	51.54 ± 0.49
Sampling	Middle	2.13 ± 0.11	13.61 ± 0.67	55.37 ± 0.98
Sampling	Tails	$\mathbf{1.56 \pm 0.10}$	17.17 ± 1.26	44.39 ± 0.13
Importance sampling	Low noise	2.32 ± 0.11	20.71 ± 0.41	43.36 ± 0.17
Importance sampling	High noise	1.66 ± 0.08	10.36 ± 0.38	43.33 ± 0.12
Importance sampling	Middle	2.21 ± 0.16	22.41 ± 0.60	44.07 ± 0.20
Importance sampling	Tails	1.64 ± 0.06	9.85 ± 0.24	42.03 ± 0.16
Weighting	Low noise	1.75 ± 0.05	25.22 ± 0.56	$\mathbf{40.43 \pm 0.13}$
Weighting	High noise	2.00 ± 0.05	10.95 ± 0.46	49.34 ± 0.25
Weighting	Middle	1.86 ± 0.09	10.53 ± 0.45	50.61 ± 0.51
Weighting	Tails	1.69 ± 0.11	30.35 ± 2.02	46.41 ± 0.34

Table 8: The same fixed Flow tilts on CelebA-HQ, using decoded pixel FID against the decoded training subset. Cells are mean ± s.e.m. over five seeds; bold marks the lowest displayed mean in each column.

Intervention	Emphasis	50 images	500 images	5,000 images
Uniform	—	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
Sampling	Low noise	1.12 ± 0.11	61.86 ± 1.44	38.49 ± 0.39
Sampling	High noise	1.04 ± 0.04	$\mathbf{4.20 \pm 0.15}$	51.61 ± 0.43
Sampling	Middle	0.99 ± 0.09	13.88 ± 0.39	50.62 ± 0.40
Sampling	Tails	1.27 ± 0.11	17.35 ± 1.12	37.73 ± 0.18
Importance sampling	Low noise	1.15 ± 0.13	14.88 ± 1.02	39.77 ± 0.09
Importance sampling	High noise	1.17 ± 0.14	16.20 ± 0.76	38.73 ± 0.36
Importance sampling	Middle	1.34 ± 0.11	20.78 ± 1.06	40.65 ± 0.26
Importance sampling	Tails	1.32 ± 0.16	10.91 ± 0.76	38.05 ± 0.29
Weighting	Low noise	1.06 ± 0.05	66.22 ± 1.58	$\mathbf{37.59 \pm 0.23}$
Weighting	High noise	0.96 ± 0.05	4.52 ± 0.21	49.36 ± 0.25
Weighting	Middle	$\mathbf{0.93 \pm 0.08}$	10.36 ± 0.31	45.53 ± 0.39
Weighting	Tails	1.24 ± 0.08	21.72 ± 1.42	39.59 ± 0.21

The preferred tilt depends on the subset size. At 5,000 images, a low-noise tilt helps. Low-noise sampling lowers mean FID on both datasets, and low-noise weighting lowers it further. A high-noise tilt is clearly worse here, for both sampling and weighting.

Default EDM already contains a low-noise tilt of this kind. In the common Flow-velocity coordinate, its sampler and loss weight combine into

$$\mathcal{L}_{\mathrm{EDM}} = \int_0^1 \pi^{\mathrm{EDM}}(\alpha) q(\alpha) \ell_v(\alpha)\, d\alpha, \qquad q(\alpha) = (1-\alpha)^2 + \alpha^2, \tag{70}$$

where $\ell_v(\alpha) = \mathbb{E}\|v_\theta - v^*\|_2^2$ and π^{EDM} is given in Equation 39. After normalizing the product $\pi^{\mathrm{EDM}} q$, the defaults $m = -1.2$, $r = 1.2$, and $s = 0.5$ put 68.4% of the objective below $\alpha = 0.5$, a $2.16{:}1$ low-to-high ratio. Default EDM is therefore a milder low-noise tilt than our $3{:}1$ controls. This fits its 5,000-image results: it has lower FID than Flow (uniform) on CIFAR-10 and stays close on CelebA-HQ (Table 6, block (a)). EDM also differs in conditional normalization and inference, so this comparison is supporting evidence only.

At 500 images, the direction reverses. High-noise sampling and high-noise weighting both lower mean FID on both datasets, most strongly on CelebA-HQ. A low-noise tilt is much worse here.

At 50 images, there is no clear preference. All FIDs are close together, and the direction that helps changes with the intervention. On CIFAR-10, for example, high-noise sampling lowers FID while high-noise weighting raises it, and low-noise weighting lowers FID while low-noise sampling raises it.

The middle and tail rows show no consistent direction either. On CIFAR-10, middle sampling is worse than Flow (uniform) at every subset size, and tail sampling helps only at 50 images. On CelebA-HQ, middle weighting helps at 50 and 500 images, and tail sampling helps only at 5,000 images.

What the samples suggest. Figure 14 shows Flow (uniform) samples after 20k steps at the three subset sizes. At 50 and 500 images, the samples are already clear: they are sharp and look like training images. At 5,000 images, they are not. The CIFAR-10 samples are blurry, and the CelebA-HQ faces are smeared and distorted. Every run gets the same 20k steps, so the 5,000-image run sees each training image the fewest times and is the furthest from fitting its data.

Together with the tables, the samples suggest a simple reading. Low noise levels restore fine details, and high noise levels decide the global structure of an image. When the images are not yet clear, as at 5,000 images, the model still has to learn those details, and a low-noise tilt helps. When the images are already clear, as at 500 images, what remains is to get the distribution of generated images right, and a high-noise tilt helps. At 50 images, the model already reproduces its training images closely, so neither tilt has much left to fix. This reading comes from two datasets and one training budget, and we have not tested it directly.

A fixed low-noise tilt on Albis. We also test a low-noise tilt on top of Albis. These runs keep its $\nu = 16$ logit Student-t base and its loss-adaptive sampling, and multiply the loss by the normalized fixed weight

$$w_\rho(\alpha) = \frac{\rho^{1-\alpha}}{(\rho - 1)/\log \rho}, \qquad \rho \geq 1, \tag{71}$$

with $w_1(\alpha) = 1$ in the limit. Larger ρ gives more weight to low noise. The loss tracker records the loss before this weight is applied.

The Albis sweep follows the same pattern. At 5,000 images, the tilt helps, and FID drops further as ρ grows, although it stays higher than for Flow (uniform). At 500 images, every tilt hurts, and the damage grows with ρ.

No fixed tilt to recommend. For each subset size, the fixed controls point in one direction. At 5,000 images, they favor a low-noise tilt. At 500 images, they favor a high-noise tilt. At 50 images, they show no clear preference. The samples suggest that a low-noise tilt helps while the images are not yet clear, and a high-noise tilt helps once they are. The best direction therefore changes with the setting, and there is no clear tilt that we can recommend in general.

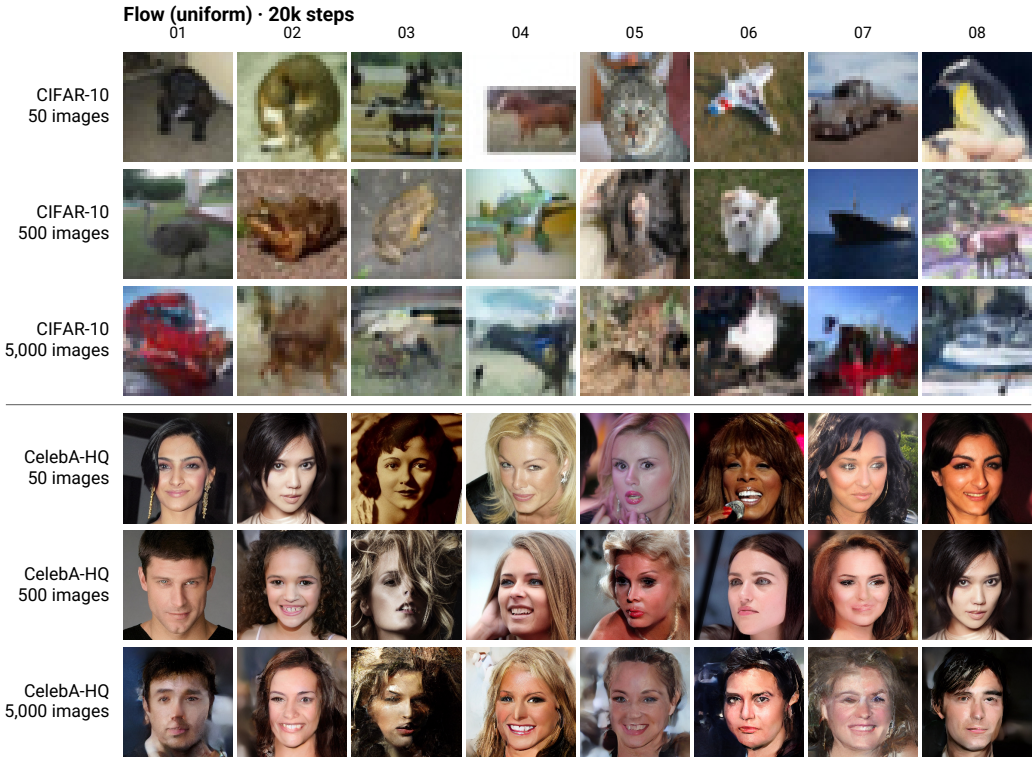

Figure 14: Flow (uniform) samples after 20k steps with 50, 500, and 5,000 training images, on CIFAR-10 (top) and CelebA-HQ (bottom, decoded to pixels). Columns are the first eight saved sample positions for training seed 100 and evaluation seed 20, the same positions as the Flow (uniform) rows in Appendix F.3. The samples are not selected by quality.

Table 9: Albis with the additional fixed low-noise weight in Equation 71, after 20k steps. The $\rho = 1$ row is Albis without the tilt. Cells are FID $\pm$ s.e.m. over five seeds; CelebA-HQ uses decoded pixel FID. Bold marks the lowest displayed mean in each setting.

	CIFAR-10			CelebA-HQ		
ρ	50	500	5,000	50	500	5,000
1	0.99 ± 0.05	$\mathbf{6.34 \pm 0.12}$	56.98 ± 0.09	0.79 ± 0.05	$\mathbf{5.30 \pm 0.21}$	47.36 ± 0.41
2	0.97 ± 0.06	7.20 ± 0.23	55.31 ± 0.28	0.65 ± 0.06	9.88 ± 0.29	44.80 ± 0.34
3	$\mathbf{0.91 \pm 0.04}$	7.76 ± 0.20	52.90 ± 0.26	0.60 ± 0.05	14.85 ± 0.75	43.57 ± 0.36
5	1.01 ± 0.05	9.70 ± 0.18	50.66 ± 0.44	0.58 ± 0.05	23.77 ± 0.49	41.40 ± 0.37
10	1.04 ± 0.04	14.06 ± 0.42	$\mathbf{46.58 \pm 0.36}$	$\mathbf{0.58 \pm 0.07}$	42.65 ± 1.06	$\mathbf{40.17 \pm 0.31}$

C.2 Noise-Level Learning Diagnostics

Section 4 builds its curriculum on one observation: during training, the noise levels where Flow (uniform) gains the most slowly move outward. Figure 15 shows this on the 500-image subsets. Each row is the drop in validation velocity MSE over one 2k-step interval, at all 256 noise levels and averaged over five seeds.

The first row starts from the loss at step zero, $L_0 = 2$. The output head is initialized to zero and Flow (uniform) has no conditional normalization, so the initial prediction is $v_{\theta_0} = 0$. Because ϵ and x_0 are independent with unit second moments, the initial loss is $L_0(\alpha) = \mathbb{E}[\|\epsilon - x_0\|^2_{\mathrm{MSE}}] = 2$ at every noise level. Every later row is the difference between two consecutive validation evaluations.

Reading the rows from top to bottom shows the gain moving outward. The first interval gains the most in the middle of the noise range. After that, the middle is already close to its final loss, and the largest gains sit further out. The two traces mark the largest gain in each half of the noise range, and both move away from the middle as training proceeds. The low-noise trace moves steadily toward the low-noise tail, while the high-noise trace moves less. The extreme tails still improve in the last intervals, so they are still being learned when training stops. This is why we call them the hard part of the range.

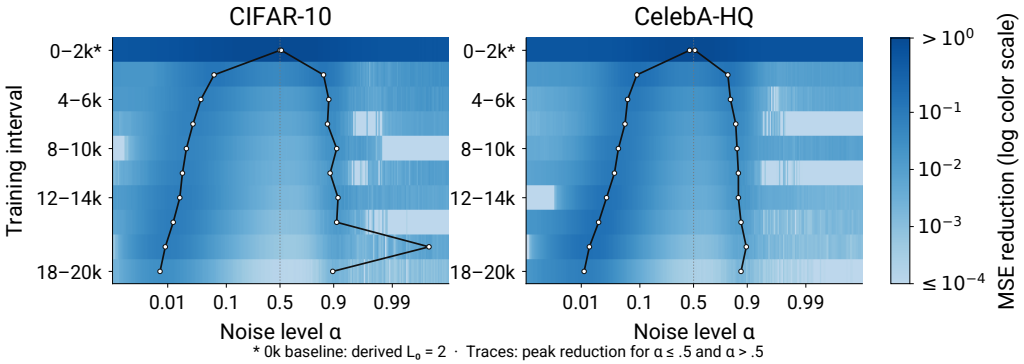

Figure 15: Consecutive velocity-MSE reductions for Flow (uniform) on the 500-image subsets. The first row starts from the step-zero loss $L_0 = 2$, and later rows use validation evaluations every 2k steps through 20k, averaged over five seeds at all 256 noise levels. Positive reductions use a logarithmic color scale from 10^{-4} to 2; all values at or below 10^{-4}, including nonpositive changes, share the same soft blue. Traces locate the largest positive reduction separately in each half of the noise range, computed before applying the display floor.

C.3 Full Curriculum Comparison

Figures 16 and 17 extend the main-text curriculum experiment to all three subset sizes and to three ways of applying the schedule. *Sampling* changes the noise sampling density and keeps unit loss weights. *Importance sampling* draws from the same changing density and divides by it, so its objective stays the Flow (uniform) objective in expectation. *Weighting* keeps noise sampling uniform and changes the loss weights. Each 3×3 block holds the nine early/late combinations for one intervention and one subset size. Every run trains for 20k steps.

Sampling supports a curriculum. At 500 images, the middle-to-tail curriculum gives the lowest mean FID on both datasets under sampling, and it clearly improves on Flow (uniform). This agrees with the diagnostics in Appendix C.2: the model learns the middle first, and moving the samples to the tails afterwards helps.

The subset size matters. The curriculum only works this well at 500 images. At 50 images, middle-to-tail sampling still improves on Flow (uniform), but it is no longer clearly the best. CIFAR-10 favors constant tail sampling, and CelebA-HQ favors the middle-to-uniform curriculum, nearly tied with middle-to-tail. At 5,000 images, the middle-to-tail curriculum has higher FID than Flow (uniform) for all three interventions on both datasets.

We read this as a mismatch between the fixed pace of the schedule and the difficulty of the task. With 50 images, the middle is learned almost immediately. At 2k steps, the mean Flow (uniform) loss over the middle is already 0.013 on CIFAR-10 and 0.018 on CelebA-HQ, so the schedule should move to the tails much sooner than it does. With 5,000 images, even the middle is still hard at the end of training. At 20k steps, its mean loss is still 0.234 on CIFAR-10 and 0.740 on CelebA-HQ, so the easy part is never finished. The 500-image setting sits between these extremes, and there the fixed pace happens to fit. Earlier work also finds curriculum learning sensitive to pacing and to optimization settings, with benefits that depend on the training regime (Hacohen & Weinshall, 2019; Wu et al., 2021). Our experiment sweeps the endpoints of the schedule and holds its pace fixed, so it does not find the best timing.

Importance sampling and weighting are hard to read. The other two interventions give a less clear picture. At 500 images under importance sampling, the middle-to-tail curriculum has higher FID than Flow (uniform) on both datasets, and constant tail emphasis gives the lowest means. Under weighting, the middle-to-tail curriculum improves on Flow (uniform), but constant middle emphasis does better still. Each of these interventions changes only part of what sampling changes. Importance sampling changes which noise levels the model sees and keeps the Flow (uniform) objective. Weighting changes the objective and keeps uniform samples. Appendix C.4 shows that these choices train different models even when their objectives agree, so their curriculum results are difficult to interpret directly.

Conclusion. Sampling suggests that a curriculum from the middle to the tails is plausible. But it needs the right number of images for the fixed pace to match the difficulty of the task: it works best at 500 images, loses its clear lead at 50 images, and hurts at 5,000. A preset curriculum is therefore very fragile. With importance sampling and weighting, the curriculum results are difficult to interpret directly, so we have to be careful when we draw conclusions from these runs.

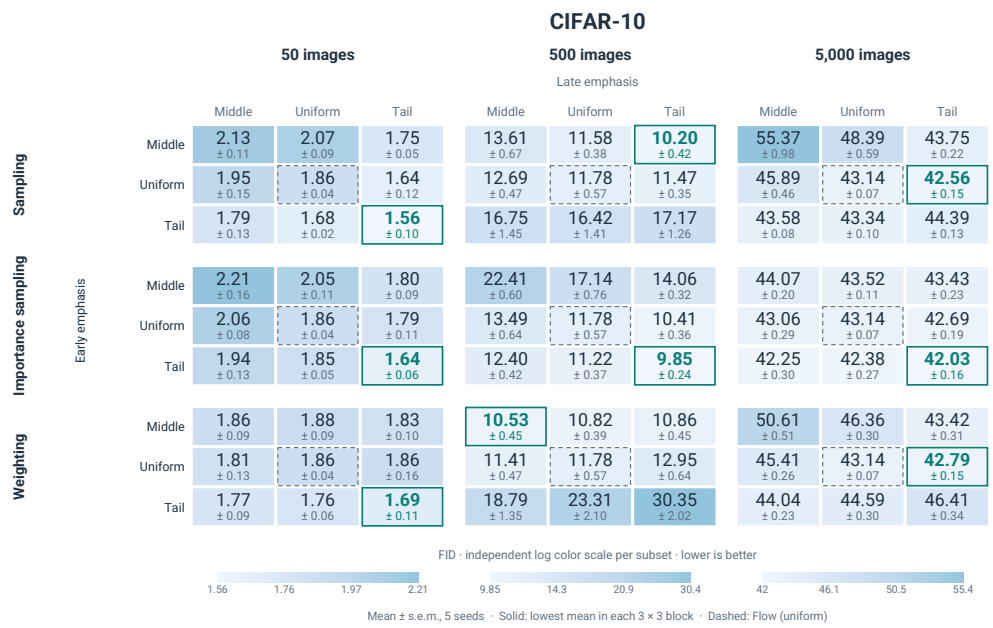

Figure 16: Full CIFAR-10 curriculum comparison. Rows group the three interventions and early emphasis; columns group subset size and late emphasis. Cells report mean FID ± s.e.m. over five seeds after 20k steps. Solid teal borders mark the lowest mean within each intervention/subset block; dashed borders mark Flow (uniform), reused across interventions. Each subset column has its own logarithmic color scale and colorbar. FID references are the respective training subsets.

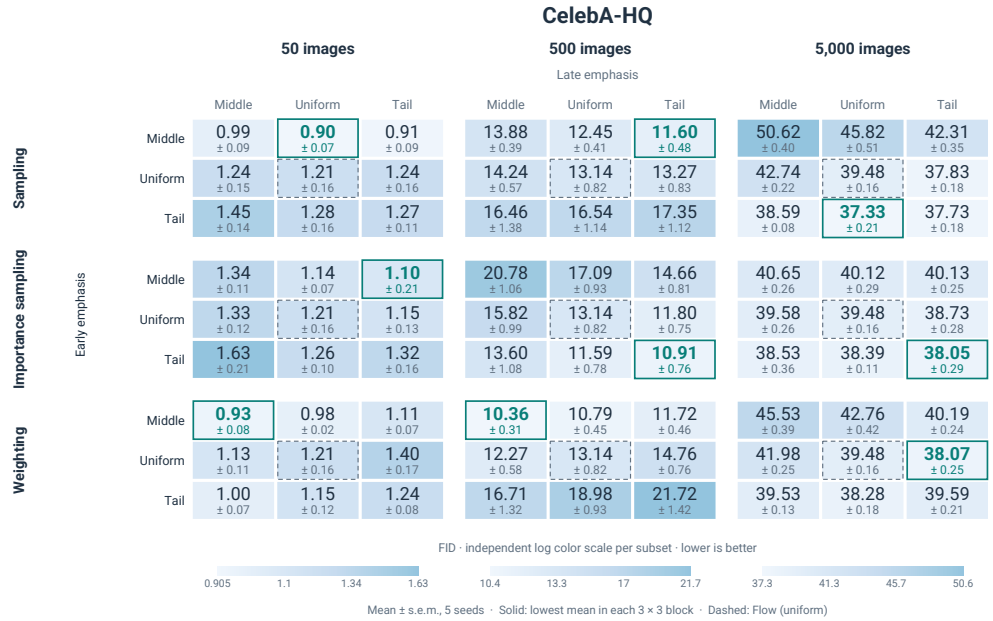

Figure 17: Full CelebA-HQ curriculum comparison, with the same layout and five-seed mean FID ± s.e.m. as Figure 16. Each subset column has its own logarithmic color scale and colorbar, independently of CIFAR-10. Solid teal borders mark the lowest mean within each intervention/subset block; dashed borders mark reused Flow (uniform). All runs use 20k steps. FID is computed after decoding, against the respective decoded training subsets.

C.4 Weighting Behaves Differently from Sampling

The objective in Equation 2 depends on the sampling density $\pi(\alpha)$ and the loss weight $\lambda(\alpha)$ only through their product. In theory, it therefore makes no difference whether we emphasize a noise level by sampling it more often or by weighting it more heavily. Sampling from a density w with unit weight and sampling uniformly with weight w both minimize $\int_0^1 w(\alpha)\,\ell_\theta(\alpha)\,d\alpha$, where $\ell_\theta(\alpha) = \mathbb{E}_{x_0,\epsilon}[\|\hat{y}_\theta - y_\alpha\|^2]$ is the Flow velocity loss at noise level α. In the same way, importance sampling draws from w and divides by it, so it minimizes the Flow (uniform) objective. This equivalence holds in expectation (Kingma & Gao, 2023). In this appendix, we show that runs with the same objective can still reach very different FID.

Runs with the same objective. The fixed tilts of Appendix C.1 give us two groups of such runs. Tables 10 and 11 copy the results of Tables 7 and 8 and group them by objective. In Table 10, sampling and weighting with the same emphasis sit side by side. They optimize the same objective, because the weight is the normalized sampling density. In Table 11, every row optimizes the Flow (uniform) objective. If only the objective mattered, the runs within each group would reach the same FID up to seed noise.

Table 10: The sampling and weighting results of Tables 7 and 8, grouped by emphasis. Within each emphasis, sampling and weighting optimize the same objective. The first row is Flow (uniform) for reference. Cells are training-subset FID $\pm$ s.e.m. over five seeds after 20k steps; CelebA-HQ uses decoded pixel FID.

	CIFAR-10			CelebA-HQ		
	50	500	5,000	50	500	5,000
Flow (uniform)	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
Low noise						
Sampling	2.13 ± 0.12	33.61 ± 1.11	41.12 ± 0.21	1.12 ± 0.11	61.86 ± 1.44	38.49 ± 0.39
Weighting	1.75 ± 0.05	25.22 ± 0.56	40.43 ± 0.13	1.06 ± 0.05	66.22 ± 1.58	37.59 ± 0.23
High noise						
Sampling	1.64 ± 0.06	9.47 ± 0.40	51.54 ± 0.49	1.04 ± 0.04	4.20 ± 0.15	51.61 ± 0.43
Weighting	2.00 ± 0.05	10.95 ± 0.46	49.34 ± 0.25	0.96 ± 0.05	4.52 ± 0.21	49.36 ± 0.25
Middle						
Sampling	2.13 ± 0.11	13.61 ± 0.67	55.37 ± 0.98	0.99 ± 0.09	13.88 ± 0.39	50.62 ± 0.40
Weighting	1.86 ± 0.09	10.53 ± 0.45	50.61 ± 0.51	0.93 ± 0.08	10.36 ± 0.31	45.53 ± 0.39
Tails						
Sampling	1.56 ± 0.10	17.17 ± 1.26	44.39 ± 0.13	1.27 ± 0.11	17.35 ± 1.12	37.73 ± 0.18
Weighting	1.69 ± 0.11	30.35 ± 2.02	46.41 ± 0.34	1.24 ± 0.08	21.72 ± 1.42	39.59 ± 0.21

Table 11: The importance-sampling results of Tables 7 and 8 beside Flow (uniform). Every row optimizes the Flow (uniform) objective in expectation. Cells are training-subset FID $\pm$ s.e.m. over five seeds after 20k steps; CelebA-HQ uses decoded pixel FID.

	CIFAR-10			CelebA-HQ		
	50	500	5,000	50	500	5,000
Flow (uniform)	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
Importance sampling						
Low noise	2.32 ± 0.11	20.71 ± 0.41	43.36 ± 0.17	1.15 ± 0.13	14.88 ± 1.02	39.77 ± 0.09
High noise	1.66 ± 0.08	10.36 ± 0.38	43.33 ± 0.12	1.17 ± 0.14	16.20 ± 0.76	38.73 ± 0.36
Middle	2.21 ± 0.16	22.41 ± 0.60	44.07 ± 0.20	1.34 ± 0.11	20.78 ± 1.06	40.65 ± 0.26
Tails	1.64 ± 0.06	9.85 ± 0.24	42.03 ± 0.16	1.32 ± 0.16	10.91 ± 0.76	38.05 ± 0.29

Sampling and weighting. Table 10 shows that they do not. Within a pair, the FIDs often differ by much more than their standard errors. On CIFAR-10 with 500 images, for example, tail weighting reaches almost twice the FID of tail sampling. The differences also follow a pattern. With middle

emphasis, weighting has the lower FID in all six settings. With tail emphasis, sampling has the lower FID in five of the six. In both cases, the better run is the one that draws more samples from the tails.

Importance sampling. Table 11 shows the same for importance sampling. All five rows optimize the Flow (uniform) objective, yet their FIDs differ widely. On CIFAR-10 with 500 images, middle emphasis has almost twice the FID of Flow (uniform), while tail emphasis has lower FID than Flow (uniform). The pattern is the same as above: tail emphasis, which draws the most samples from the tails, has the lowest FID in five of the six settings, and middle emphasis has the highest in five of the six.

Curriculum learning. The curriculum experiment in Appendix C.3 shows the same difference. Under sampling, the middle-to-tail curriculum gave the lowest FID at 500 images on both datasets and supported a curriculum. Under weighting, which optimizes the same objective, constant middle emphasis did better. Under importance sampling, which keeps the Flow (uniform) objective, the curriculum had higher FID than Flow (uniform). The same schedule therefore led to a different conclusion depending on how it was applied.

Conclusion. Runs with the same objective can train very different models. The objective is only an expectation. Training follows a stochastic estimate of it, and that estimate depends on how often each noise level is sampled and how heavily each sample is weighted. We therefore cannot conclude that two runs train the same way because their objectives are the same. This is why we define the noise schedule as the pair (π, λ) and keep sampling and weighting separate (Section 3.1). It is also why Albis can act as a curriculum while its importance correction keeps the objective fixed (Section 5): the correction fixes the objective, and the sampling density still changes what the model learns.

D ALBIS IMPLEMENTATION AND MECHANISTIC EVIDENCE

D.1 ADAPTIVE SAMPLER IMPLEMENTATION

Section 5 describes Albis as a base density that the loss bends outward over training. Algorithm 1 gives every step of the implementation. This appendix explains the choices behind it: how the sampler tracks the loss, how it starts, and why its behavior does not depend on the batch size.

The base and its coordinate. The Albis base draws the logit coordinate from a Student-t distribution, giving the density on α

$$z \sim t_\nu(0,1), \qquad \alpha = \mathrm{sigmoid}(z), \qquad \pi_0(\alpha) = \frac{f_\nu(\mathrm{logit}\,\alpha)}{\alpha(1-\alpha)}, \tag{72}$$

where f_ν is the Student-t density. The implementation realizes this base as a fixed inverse-CDF transform $T(u) = \mathrm{sigmoid}(F_\nu^{-1}(u))$, so drawing u uniformly on $(0,1)$ reproduces Equation 72 exactly. The tracker and the sampling density therefore live in u rather than in physical α, and this choice simplifies everything downstream. In the u coordinate the base is uniform, so a sampling density on u is precisely the factor by which we bend the base, and the importance weight is its reciprocal. Control points are spaced uniformly in $\mathrm{logit}(u)$, which keeps resolution in the tails of the noise range alongside the middle.

The numerical settings are 2,000 control points, EMA decay 0.99, kernel bandwidth 0.1 in logit coordinates, a numerical floor of 10^{-12}, and coordinate endpoints 10^{-8} and $1 - 10^{-8}$. A control point counts as supported once its effective sample count reaches one.

Tracking the loss. The noise level is continuous, so no two training examples share the same noise level. To estimate the loss at one noise level, we therefore pool the examples drawn near it. The tracker stores its estimates on the fixed grid of control points above. Each example updates the control points around it through a Gaussian kernel in logit coordinates. We normalize the kernel weights of each example to sum to one, so every example adds exactly one unit of count, spread over its neighboring control points. The bandwidth is about five times the grid spacing. The kernel therefore sets the resolution of the tracker, and the grid only needs to be fine enough to represent it.

At each control point, the tracker keeps an EMA of the squared per-example loss, and the sampling density follows its square root (Equation 4). The EMA forgets old examples, because the model changes during training and old losses go out of date. The EMA starts from a prior of one, and a bias correction removes this prior. The estimate is therefore the weighted average of the examples seen so far, from the first update on. The tracker records the loss before the importance weight is applied, so it measures the model's error and never the weight that the sampler chose.

Between control points, the sampling density is linear. A linear density is continuous, so the importance weight has no jumps at the control points. Drawing from it stays exact: within a segment the CDF is quadratic, and we invert it in closed form.

Warm-up. At the first step, the tracker has seen no examples. Until the first segment is supported, the sampler falls back to uniform draws in u, which is the base itself, with unit weights. A control point counts as supported once it has received one effective example, that is, once its kernel count summed over training reaches one. This count follows from the decay product, $N_j = \log R_j / \log \eta$, so the tracker needs no separate counter. The threshold is needed because of the bias correction. After a single update, the corrected estimate at a control point equals the kernel-weighted mean of the examples, however small their total weight there. Without the threshold, a control point that has seen only the edge of one example's kernel would count as fully estimated.

Unsupported segments get zero sampling mass, which keeps the sampler away from control points without an estimate. This rule is meant as a short warm-up, but it has a lasting side effect. A new control point becomes supported only through the kernels of examples drawn near the edge of the support. The support therefore keeps growing outward from the middle throughout training. In the small-subset experiments of Section 6.1, the unsupported region holds at most about 1% of the base mass after 2k steps, and less than 0.1% after 20k steps. The importance weight corrects for the sampling density only inside the support. Writing $\mathcal{S}_k$ for the supported noise region at step k, the

Algorithm 1 Adaptive loss-second-moment EMA and sampling density used by Albis.

Require: control-point count J; EMA decay η; kernel bandwidth h; floor ε; support threshold $c_{\min}$; bounds $u_{\min}, u_{\max}$; fixed transform $T(u) = \mathrm{sigmoid}(F_\nu^{-1}(u))$

$g_{1:J} \leftarrow \mathrm{sigmoid}(\mathrm{linspace}(\mathrm{logit}\, u_{\min}, \mathrm{logit}\, u_{\max}, J))$
$M_j \leftarrow 1$ and $R_j \leftarrow 1$ for every j ▷ raw second-moment EMA and decay product
for optimizer steps $k = 0, 1, \ldots$ **do**
  $\widehat{m}_j \leftarrow \max\left\{\varepsilon, \dfrac{M_j - R_j}{\max(1 - R_j, \varepsilon)}\right\}$ for every j ▷ remove the unit prior
  $N_j \leftarrow \dfrac{\log(\max(R_j, \varepsilon))}{\log \eta}$, $s_j \leftarrow \mathbf{1}[N_j \geq c_{\min}]$, and $b_j \leftarrow s_j \sqrt{\widehat{m}_j}$
  $\Delta_j \leftarrow g_{j+1} - g_j$ and $A_j \leftarrow s_j s_{j+1} \Delta_j (b_j + b_{j+1})/2$ for $j = 1, \ldots, J-1$
  if $\sum_j A_j = 0$ **then**
    $b_j \leftarrow 1$ and $A_j \leftarrow \Delta_j$ for all j ▷ bounded-uniform warm-up fallback
  end if
  $A \leftarrow \sum_j A_j$; let $q_k(u)$ be the density that linearly interpolates b_j/A between g_j and g_{j+1} on segments with $A_j > 0$, and is zero elsewhere
  Let $\mathcal{B}_k$ be the optimizer batch, possibly partitioned into micro-batches
  for all $i \in \mathcal{B}_k$ using this frozen q_k **do**
    Draw segment $r_i \sim \mathrm{Categorical}(A_{1:J-1}/A)$ and $\xi_i \sim \mathcal{U}(0, 1)$
    Find $t_i \in [0, 1]$ satisfying $\dfrac{b_{r_i} t_i + (b_{r_i+1} - b_{r_i}) t_i^2/2}{(b_{r_i} + b_{r_i+1})/2} = \xi_i$
    $u_i \leftarrow g_{r_i} + t_i \Delta_{r_i}$, $\alpha_i \leftarrow T(u_i)$, and $\ell_i \leftarrow \|v_{\theta_k}(x_{\alpha_i}, \alpha_i) - (\epsilon_i - x_{0,i})\|_{\mathrm{MSE}}^2$
    $w_i \leftarrow \big((u_{\max} - u_{\min}) q_k(u_i)\big)^{-1}$ ▷ inverse-density correction to bounded-uniform u
  end for
  Accumulate $\nabla_\theta |\mathcal{B}_k|^{-1} \sum_{i \in \mathcal{B}_k} w_i \ell_i$ and retain detached (u_i, ℓ_i)
  $K_{ij} \leftarrow \dfrac{\exp\left[-\frac{1}{2}((\mathrm{logit}\, u_i - \mathrm{logit}\, g_j)/h)^2\right]}{\sum_{r=1}^{J} \exp\left[-\frac{1}{2}((\mathrm{logit}\, u_i - \mathrm{logit}\, g_r)/h)^2\right]}$
  $C_j \leftarrow \sum_i K_{ij}$ and $S_j \leftarrow \sum_i K_{ij} \ell_i^2$; sum C_j, S_j across distributed ranks
  for all j such that $C_j > 0$ **do**
    $d_j \leftarrow \eta^{C_j}$, $M_j \leftarrow \max\{\varepsilon, d_j M_j + (1 - d_j) S_j/C_j\}$, and $R_j \leftarrow R_j d_j$
  end for
  $\theta_{k+1} \leftarrow \mathrm{OptimizerStep}(\theta_k)$
end for

objective in expectation is

$$\mathbb{E}[\lambda_k \ell_k] = \int_{\mathcal{S}_k} \pi_0(\alpha) \ell_{\theta_k}(\alpha)\, d\alpha, \tag{73}$$

the base objective restricted to $\mathcal{S}_k$. The excluded base mass is small, so this objective stays close to the full base objective.

Robustness to the batch size. The tracker counts examples, not optimizer steps. Its EMA decays by η per unit of kernel count, so a control point that receives a count C_j in one step decays by η^{C_j}. Each estimate therefore averages over roughly the last $1/(1 - \eta) = 100$ examples near its control point, whatever the batch size. A control point in the tails receives few examples per step. It averages over the same number of examples as a control point in the middle, collected over more steps. The support threshold is also counted in examples. With a larger batch, the sampler warms up in fewer steps, but after about the same number of examples.

Within one optimizer step, the sampling density stays fixed. The statistics of all micro-batches and all GPUs are summed first and then enter a single update. Splitting the same optimizer batch into micro-batches or across GPUs therefore leaves the sampler update unchanged.

D.2 Component Ablations

Albis has two moving parts: a logit Student-t base with $\nu = 16$, and loss-based importance sampling on top of it. This appendix ablates both.

Figure 4 in the main text switches the importance-sampling mechanism on and off over a uniform base and a logit-normal base. Each pair holds its base fixed and enables loss-driven sampling together with its inverse-density correction, so the pair isolates the complete mechanism. Table 12 gives the five-seed means behind that figure. Three of its four arms already appear in the paper under other names: the two uniform arms are Flow (uniform) and the loss-driven importance-sampling row of Tables 2 and 13, and the logit-normal arm without adaptation is Flow (logit-normal). Only the logit-normal arm with adaptation is reported here alone. The degrees-of-freedom sweeps below all use adaptation, so the on/off comparison covers the uniform and logit-normal bases.

Table 12: The importance-sampling pairs behind Figure 4, after 20k steps. Cells are FID $\pm$ s.e.m. over five seeds; headings are training-subset sizes and also identify the FID reference. Bold marks the lower mean within each pair and column, and the change rows give the signed percentage plotted in Figure 4. CelebA-HQ uses decoded pixel FID.

	CIFAR-10			CelebA-HQ		
Adaptation	50	500	5,000	50	500	5,000
Uniform base						
Without	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	$\mathbf{1.21 \pm 0.16}$	13.14 ± 0.82	39.48 ± 0.16
With	$\mathbf{1.20 \pm 0.07}$	$\mathbf{9.26 \pm 0.26}$	$\mathbf{42.44 \pm 0.20}$	1.40 ± 0.18	$\mathbf{12.08 \pm 0.67}$	$\mathbf{39.13 \pm 0.36}$
Change	-35.7%	-21.4%	-1.6%	$+15.9\%$	-8.1%	-0.9%
Logit-normal base						
Without	4.13 ± 0.29	10.91 ± 0.43	61.48 ± 0.59	1.67 ± 0.13	7.05 ± 0.27	47.31 ± 0.44
With	$\mathbf{1.46 \pm 0.04}$	$\mathbf{5.68 \pm 0.10}$	$\mathbf{60.24 \pm 0.57}$	$\mathbf{0.79 \pm 0.05}$	$\mathbf{5.03 \pm 0.21}$	$\mathbf{47.31 \pm 0.43}$
Change	-64.7%	-47.9%	-2.0%	-52.7%	-28.7%	-0.01%

D.2.1 Degrees of freedom and weight decay

Albis draws the logit of its noise level from a Student-t distribution. The logit-normal base draws $\mathrm{logit}(\alpha) \sim \mathcal{N}(0, 1)$, in the terminology of Esser et al. (2024, Section 3.1, Equation 19). The Student-t keeps the same central emphasis and puts more mass in the tails, and it becomes the Gaussian as $\nu \to \infty$. Figures 18 and 19 compare $\nu \in \{8, 16, 32, 64, \infty\}$ at weight decays 0.00 and 0.10, on both datasets and all three subset sizes, with five seeds per point. Every run uses loss-based importance sampling. The runs without weight decay train for 20k steps, and the runs with weight decay 0.10 train for 10k steps. Weight decay and training budget therefore change together between the two figures, so we compare degrees of freedom within each figure only.

With 50 images, logit-normal sampling can give poor FID. On CIFAR-10, it has the highest FID of all choices at both weight decays. We attribute this to its thin tails, which leave the extreme noise levels almost unsampled (Section 5). We switch to a Student-t base specifically because of this 50-image result. On CelebA-HQ with 50 images, logit-normal does as well as the Student-t choices.

With 500 images, logit-normal is a perfectly reasonable choice. Its FID is close to the best Student-t value on both datasets and at both weight decays.

Which degree of freedom works best depends on the dataset, the subset size, and the weight decay. In most settings, the differences between $\nu = 8$, 16, 32, and 64 are small. The largest effect is again on CIFAR-10 with 50 images, where FID keeps dropping as the tails get heavier at weight decay 0.10. We do not tune ν per setting. We pick $\nu = 16$ and use it in every Albis experiment outside this ablation. The results are not very sensitive to this choice, and $\nu = 16$ still gives a clear improvement over logit-normal on CIFAR-10 with 50 images. Overall, the Student-t base is a slight optimization, and we do not expect it to be very consequential.

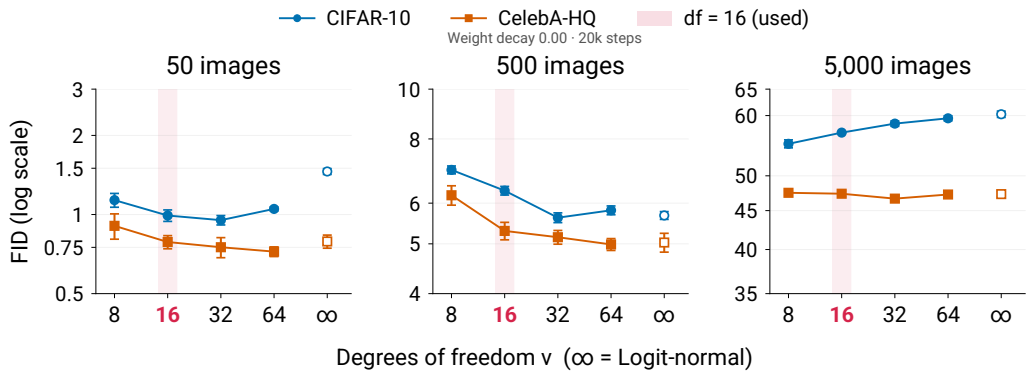

Figure 18: Degrees-of-freedom sweep at weight decay 0.00 after 20k training steps: $\nu = 8, 16, 32, 64, \infty$, all with loss-based importance sampling. The 30 points show five-seed mean FID and s.e.m. The band marks $\nu = 16$, used in all Albis experiments outside this ablation; open markers show the logit-normal limit. Compare choices within each dataset/subset series, because the FID references differ between them. CelebA-HQ uses decoded pixel FID. Axes are logarithmic and identical to Figure 19.

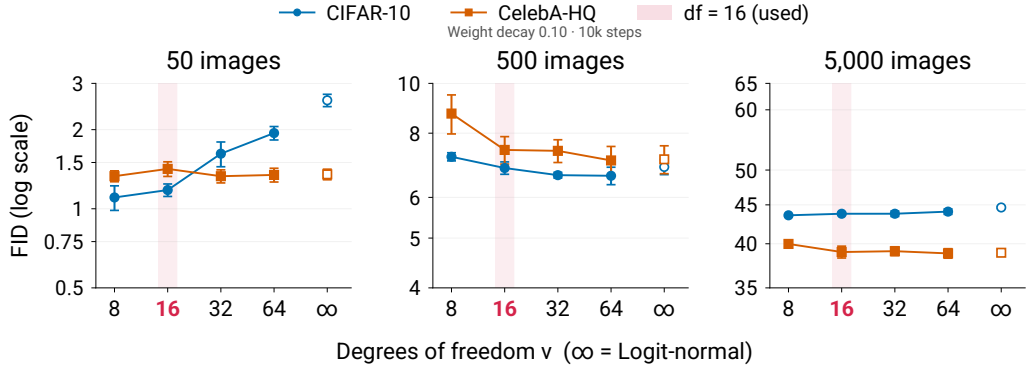

Figure 19: The same degrees-of-freedom sweep at weight decay 0.10 after 10k training steps. All 30 points show five-seed mean FID and s.e.m., with the same datasets, subsets, metrics, axes, and markers as Figure 18. The preferred degree of freedom varies across settings, while $\nu = 16$ remains a practical common choice. The two figures use different training budgets, so weight decay and budget move together between them.

D.3 Gradient Norms and Variance Reduction

Albis uses the variance-reduction formulation of importance sampling (Section 5). Sampling more where the loss is large, and correcting with the importance weight, lowers the variance of the loss estimate. Improved DDPM uses loss-based sampling for this reason (Nichol & Dhariwal, 2021), and Kingma & Gao (2023) sample log-SNR by the running mean of the weighted loss for the same reason. The same rule also acts as a curriculum, because it moves the samples to the noise levels that the model has not learned yet. In this appendix, we look at the variance-reduction side of Albis through the gradient norm.

Appendix D.3.1 first shows how the gradient norm behaves across noise levels during training. Appendix D.3.2 derives the variance-reduction mechanism, which points to the gradient norm as the statistic to follow. Appendix D.3.3 asks whether switching the tracked statistic from the loss to the gradient norm improves the results further. Appendix D.3.4 shows how the base and importance sampling change the gradient norm of the update.

D.3.1 Gradient norm during training

We first look at how the gradient norm behaves across noise levels, and how it compares with the loss. For every example, let $g = \nabla_\theta \ell$ be the gradient of its velocity MSE with respect to all trainable parameters. We track $\|g\|_2^2$ with the same kernel and bias-corrected EMA that the sampler uses for the loss (Appendix D.1). The gradient statistic is therefore $\sqrt{\mathbb{E}[\|g\|_2^2 \mid \alpha]}$, the root second moment of the per-example gradient norm. It measures the gradients of single examples, and it differs from the norm of the averaged batch gradient. The gradient passes the prediction error back through the network, so it depends on the network as well as on the error itself. We therefore expect the gradient norm to follow the loss only roughly.

Figure 20 shows both statistics during Flow (uniform) training. In the middle of the noise range, they behave alike. Both are lowest there, and the low region deepens and widens as training continues. This matches the observation in Section 4 that the model learns the middle first. At high noise, both statistics stay high and change little. At low noise, they differ. There, the gradient norm grows over training while the loss stays about the same or falls, most visibly on CelebA-HQ. The gradient norm therefore also seems to be a proxy for the difficulty of a noise level, but it behaves slightly differently from the loss.

D.3.2 Variance reduction

We derive the variance-reduction mechanism here, and the rest of this appendix refers back to it. The same derivation applies to any quantity that training estimates from examples at random noise levels. We apply it twice, once to the loss and once to the gradient.

Let $\pi_0(\alpha)$ be the base noise density, and let $\xi = (x_0, \epsilon)$ collect the other randomness of an example. Let $f(\alpha, \xi)$ stand for either the per-example loss ℓ or the per-example gradient g, and let $\mu_f = \mathbb{E}_{\alpha\sim\pi_0,\xi}[f(\alpha,\xi)]$ be its average under the base. If we draw $\alpha \sim \pi_k$ and apply the importance weight $\pi_0(\alpha)/\pi_k(\alpha)$, the single-example estimate

$$\widehat{f} = \frac{\pi_0(\alpha)}{\pi_k(\alpha)} f(\alpha, \xi)$$

is unbiased wherever π_k covers the support of π_0. Write

$$s_f(\alpha) = \sqrt{\mathbb{E}_\xi[\|f(\alpha,\xi)\|_2^2 \mid \alpha]}$$

for the root second moment of f at noise level α. The variance of the estimate, summed over its coordinates, is

$$\operatorname{tr}\operatorname{Cov}(\widehat{f}) = \int \frac{\pi_0(\alpha)^2}{\pi_k(\alpha)} s_f(\alpha)^2 \, d\alpha - \|\mu_f\|_2^2. \tag{74}$$

Only the first term depends on π_k. Minimizing it over normalized densities, with a Lagrange multiplier or with Cauchy–Schwarz, gives

$$\pi_k^*(\alpha) = \frac{\pi_0(\alpha) s_f(\alpha)}{\int \pi_0(a) s_f(a)\, da}. \tag{75}$$

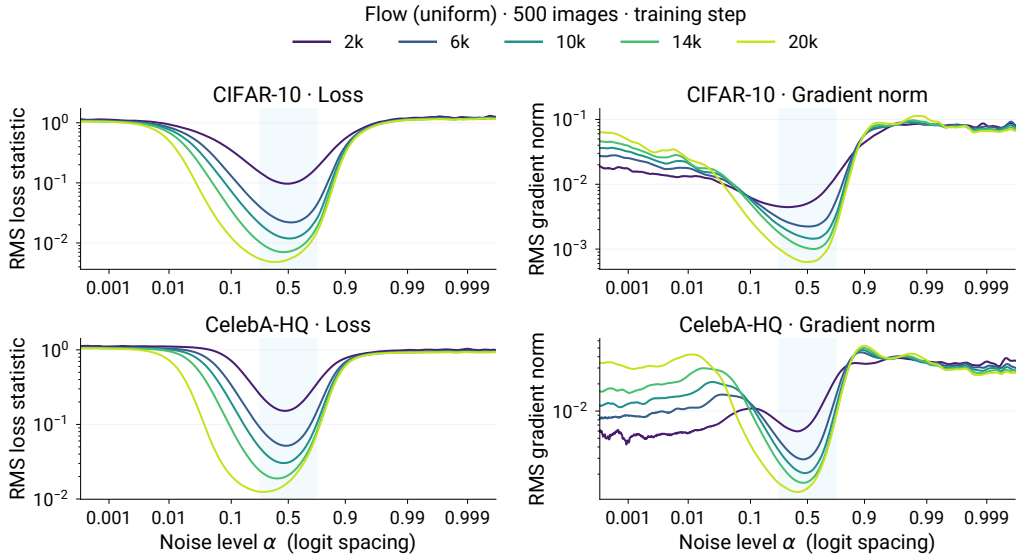

Figure 20: Loss and per-example parameter-gradient-norm dynamics for Flow (uniform) on the 500-image subsets. Curves show five-seed means at five of the ten checkpoints. Each vertical axis is logarithmic; the horizontal axis is physical α with logit spacing, and shading marks $\alpha \in [0.25, 0.75]$. The plotted quantities are root second moments from the adaptive tracker. A control point is shown only when it is supported in all five seeds. Both statistics fall first in the middle. At low noise, the gradient norm grows over training while the loss stays about the same or falls.

Averaging over a batch of independent examples divides the variance by the batch size, so the same density is optimal for any batch size. The derivation holds at fixed parameters, for a single training step. It does not tell us which parameters training reaches later with a changed sampling density.

For the loss, $s_\ell(\alpha) = \sqrt{\mathbb{E}[\ell^2 \mid \alpha]}$, and Equation 75 is the Albis rule of Equation 4, with m_k as a running estimate of s_ℓ^2. This is also the rule that Improved DDPM uses to reduce the variance of its loss estimate (Nichol & Dhariwal, 2021). For the gradient, $s_g(\alpha) = \sqrt{\mathbb{E}[\|g\|_2^2 \mid \alpha]}$ is the gradient statistic of Appendix D.3.1, and Equation 75 gives the density that minimizes the variance of the gradient estimate.

In short, using the second moment of the loss minimizes the variance of the loss estimate, and using the second moment of the gradient norm minimizes the variance of the gradient estimate. The optimizer only sees the gradient. If we want to minimize the variance of the optimization, we therefore most likely want to use the gradient norm instead of the loss.

D.3.3 Sampling by the gradient norm

Albis tracks the loss, but Appendix D.3.2 shows that the gradient norm is the statistic that minimizes the gradient variance. Can we further improve the performance by switching from the loss to the gradient norm? To answer this, we train with importance sampling and let either the loss or the gradient norm drive the sampling density.

Every run in this test uses the uniform base of Flow (uniform), so only the adaptive part differs between runs. Each run tracks both statistics with the tracker of Appendix D.1, and one of them drives the sampler. Before the tracker update, we divide out the current adaptive weight, so the tracker records the model's own loss and gradient and not the weight that the sampler applied. The comparison covers CIFAR-10 and CelebA-HQ, all three subset sizes, and five seeds, with 20k steps per run.

Table 13 reports the results. The importance-sampling rows answer the question, because importance sampling keeps the uniform objective and changes only which noise levels the model sees.

The table also includes three mechanisms that change the objective. *Sampling* draws from the same density without the inverse-density weight, *weighting* keeps uniform sampling and multiplies the loss by the tracked statistic, and *inverse weighting* multiplies the loss by its reciprocal. These rows do not answer the question directly, but they show how differently the two statistics act.

Table 13: All small-subset experiments with gradient-norm tracking after 20k steps. Cells are FID $\pm$ s.e.m. over five seeds; headings are training-subset sizes and also identify the FID reference. The target is the tracked statistic that drives each mechanism. Sampling draws from a target-driven density with unit adaptive weight; importance sampling adds its inverse-density correction; weighting and inverse weighting keep uniform sampling. CelebA-HQ uses decoded pixel FID. Lower is better, and FID values should be compared within a dataset/subset column.

		CIFAR-10			CelebA-HQ		
Mechanism	Target	50	500	5,000	50	500	5,000
Flow (uniform)	None	1.86 ± 0.04	11.78 ± 0.57	43.14 ± 0.07	1.21 ± 0.16	13.14 ± 0.82	39.48 ± 0.16
Sampling	Loss	1.48 ± 0.07	28.49 ± 0.92	44.66 ± 0.11	3.23 ± 0.44	20.88 ± 0.72	37.95 ± 0.23
	Gradient norm	1.79 ± 0.07	48.84 ± 2.51	55.00 ± 0.77	2.00 ± 0.20	3.66 ± 0.09	44.31 ± 0.32
Importance sampling	Loss	1.20 ± 0.07	9.26 ± 0.26	42.44 ± 0.20	1.40 ± 0.18	12.08 ± 0.67	39.13 ± 0.36
	Gradient norm	1.20 ± 0.09	9.42 ± 0.22	44.09 ± 0.28	1.15 ± 0.12	12.85 ± 0.63	37.98 ± 0.37
Weighting	Loss	4.28 ± 0.18	56.19 ± 1.62	47.29 ± 0.34	3.47 ± 0.36	31.64 ± 1.68	37.89 ± 0.30
	Gradient norm	5.34 ± 0.15	106.54 ± 2.30	63.16 ± 0.54	2.31 ± 0.33	8.13 ± 0.53	49.04 ± 0.34
Inverse weighting	Loss	2.20 ± 0.15	16.60 ± 0.11	46.51 ± 0.52	0.77 ± 0.05	17.91 ± 1.07	41.31 ± 0.37
	Gradient norm	3.82 ± 0.90	78.63 ± 24.13	43.39 ± 0.43	57.37 ± 30.85	210.92 ± 35.93	172.80 ± 32.85

Under sampling, weighting, and inverse weighting, the FID with the gradient norm can be wildly different from the FID with the loss, and the direction of the difference changes between settings. The loss can therefore not be used blindly as a proxy for the gradient norm. Under importance sampling, however, the two statistics give roughly the same FID. Their FIDs are close in every setting. The largest gaps are at 5,000 images, where they are a few percent and go in opposite directions on the two datasets.

The answer is therefore no. Under importance sampling, switching from the loss to the gradient norm gives no clear benefit, although the two statistics act very differently as weights. For importance sampling, the loss works as well as the gradient norm.

Cost of per-example gradient norms. These experiments serve purely as an analysis. The loss of each example comes for free with the forward pass, but its gradient norm does not. Backpropagation sums the gradients of all examples in a batch, so the tracker has to rebuild the gradient of every example in every layer before it can take the norm. In our small-subset experiments, a training step with gradient-norm tracking takes more than twice as long as a step without it, and it uses about a fifth more GPU memory. Tracking the loss alone adds little. Even if the gradient norm had beaten the loss, it would therefore be hard to use in practice. We therefore track it only in the experiments of this appendix.

D.3.4 Uniform and logit-normal bases

The main text switches loss-based importance sampling on and off over a uniform and a logit-normal base (Figure 4). Here we look at what these four runs do to the gradient norm. Figure 21 shows the norm of the batch gradient that the optimizer uses, including the importance weights. Its expected square is the squared mean gradient plus the gradient variance divided by the batch size, so it rises and falls with the gradient variance of the optimization.

Importance sampling lowers the gradient norm. At 50 and 500 images, the run with importance sampling ends with a clearly lower gradient norm than the run without it, on both bases, and the reduction is largest on the logit-normal base. At 5,000 images, importance sampling changes the gradient norm little. The logit-normal base already has a much smaller gradient norm than the uniform base without importance sampling. Most likely, this is because it rarely draws the extreme noise levels, where the per-example gradients are largest (Figure 20).

This is evidence of variance reduction, and it lines up with the changes in FID. Across the twelve pairs of Table 12, a larger reduction of the gradient norm tends to go with a larger FID improvement.

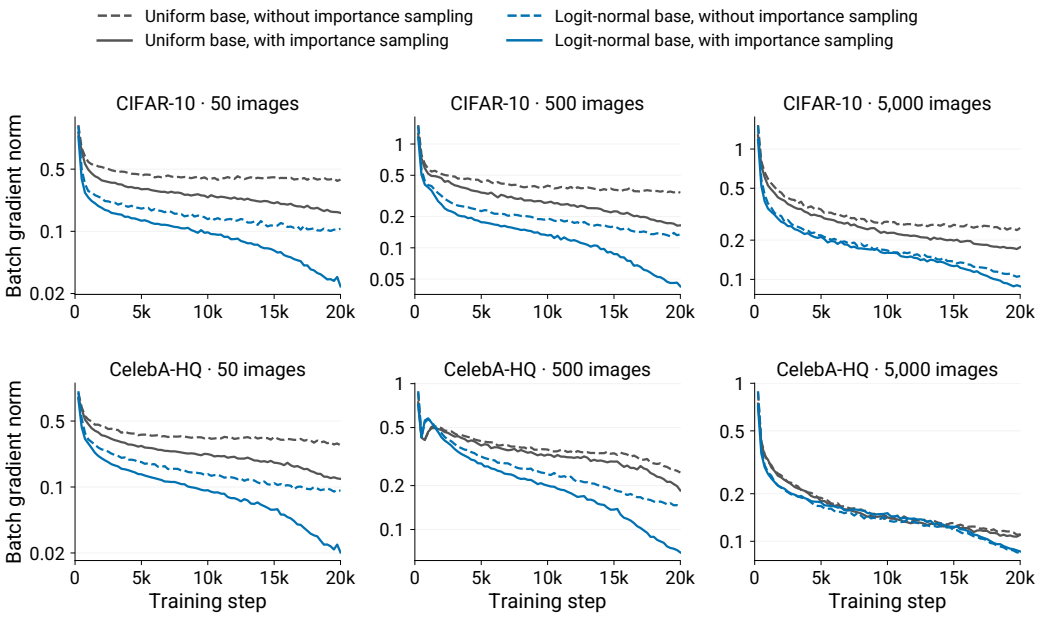

Figure 21: Norm of the batch gradient that the optimizer uses, including the importance weights, for the four runs of Figure 4. Curves show five-seed means of the root mean square over 250-step windows. Color marks the base, and line style marks importance sampling. Every panel has its own logarithmic vertical axis.

The settings where importance sampling lowers the gradient variance most are therefore the settings where it helps most. There is one clear exception. On CelebA-HQ with 50 images, importance sampling more than halves the gradient norm of the uniform base, yet FID gets worse. Section 6.1 explains this case with the curriculum view: on the uniform base, adaptation supplies only the second half of the curriculum. Between the bases, the link also breaks: the logit-normal base has the smaller gradient norm in every setting, but the worse FID at 50 and 5,000 images. The two bases train different objectives, however, so this comparison is no clean test of variance reduction.

In Albis, variance reduction and curriculum learning are closely linked and hard to separate, because both effects grow with how far the sampler moves away from its base.

E Scaling Analyses and Failure Modes

E.1 Training-Budget Noise-Level Losses

Section 6.2 contrasts two behaviors. A fixed sampler trades one part of the noise range for another, while Albis accelerates the pretraining itself. This appendix gives the measurements behind that contrast.

Figure 22 shows the final velocity MSE of Flow (uniform), Flow (logit-normal), and Albis at every noise level α, for the 2,500-, 40,000-, and 320,000-step budgets. Each budget is a separately trained run, and we evaluate it on its training subset. Each curve is a three-seed mean over 256 noise levels, and a larger α means more noise. The absolute curves almost overlap, so Figure 23 shows the same evaluations relative to Flow (uniform), as the percentage $100(\overline{L}_{\mathrm{method}}/\overline{L}_{\mathrm{Flow}} - 1)$. Negative values mean a lower loss than Flow (uniform) at that noise level.

The two fixed samplers trade against each other. Flow (logit-normal) concentrates its training on the middle noise levels and beats Flow (uniform) there. At 320k steps, its median loss over $\alpha \in [0.25, 0.75]$ is 61.3% below that of Flow (uniform) on CIFAR-10 and 64.7% below on CelebA-HQ. Flow (uniform) spreads the same budget evenly and keeps the lower loss in the tails. Over $\alpha \leq 0.077$ and $\alpha \geq 0.97$, the median loss of Flow (logit-normal) is 26.0% above that of Flow (uniform) on CIFAR-10 and 20.8% above on CelebA-HQ. Each fixed sampler therefore wins the region it focuses on and gives up the other. A fixed schedule has to make this choice, because it decides before training which noise levels matter.

Albis does better than this trade-off. At 320k steps, its three-seed mean loss is lower than that of Flow (logit-normal) at all 256 noise levels on both datasets. In the middle, its median loss is 70.4% below that of Flow (uniform) on CIFAR-10 and 71.0% below on CelebA-HQ, so it improves on Flow (logit-normal) there as well. In the tails, it stays much closer to Flow (uniform), at 8.7% and 9.3% above, where Flow (logit-normal) is 26.0% and 20.8% above. Flow (uniform) still has the lowest tail loss of the three methods, so Albis keeps most of the tail performance of uniform sampling, but not all of it. Albis therefore gains in the middle while giving up little in the tails. This combination comes from moving the emphasis during training, which a fixed schedule cannot do.

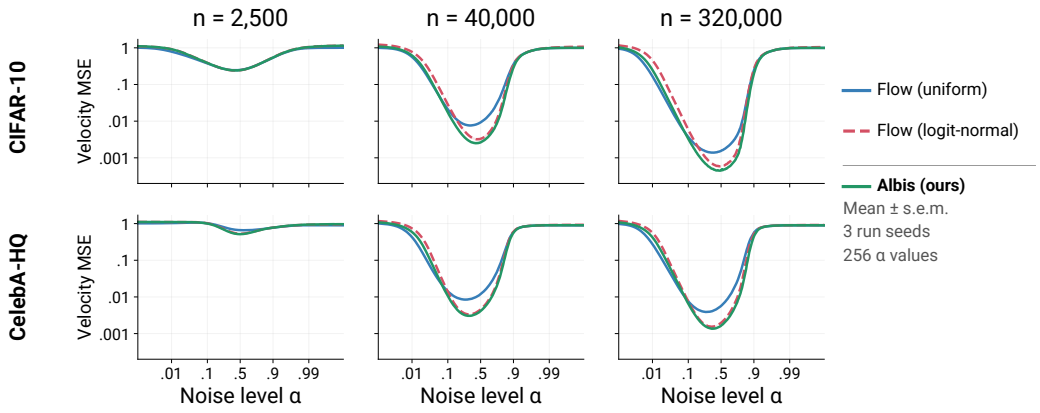

Figure 22: Final velocity MSE for Flow (uniform), Flow (logit-normal), and Albis at the 2,500-, 40,000-, and 320,000-step budgets. Curves show the three-seed mean and s.e.m. All panels share a logarithmic MSE axis and an α axis with logit spacing. At the earliest budget, the curves still cross.

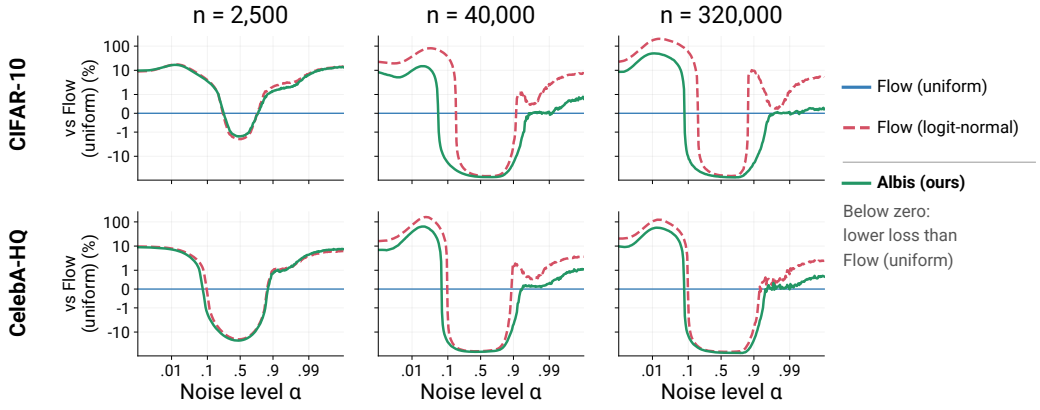

Figure 23: The evaluations of Figure 22, relative to Flow (uniform). The percentage axis is symmetric-logarithmic and linear within $\pm 1\%$. Negative values mean a lower loss than Flow (uniform).

E.2 Exposure Bias: When Lower Loss Produces Worse Images

Our main analysis in this paper is about noise scheduling. Along the way, however, we found that we could not reliably train a simple neural network on the full CIFAR-10 dataset. In contrast, training works fine on small subsets of CIFAR-10 and on the much larger ImageNet. In this appendix, we investigate this issue and look at it through exposure bias.

E.2.1 The problem and exposure bias

The loss improves while FID gets worse. We first noticed the problem in the training-budget sweep of Section 6.2. When we added weight decay to Flow (uniform), some runs kept lowering their loss as the training budget grew, while their FID rose. Normally, a lower denoising loss means a better model and better images. In these runs, it does not. This is especially strange because we compute FID against the training images (Section 3.2). This FID therefore measures memorization: a model that reproduces its training images exactly gets the best score. A model that trains longer should memorize its training images better, so its FID should fall with the training budget and not rise. We later found the same pattern in other settings, and in several of them, many generated images are visibly blurred or poorly formed. The following sections show these experiments.

What is exposure bias? During training, the network always receives a noisy version of a real image. During generation, its input at each step is the result of its own previous predictions. An error in one prediction therefore changes the next input, and this input can differ from the noisy images on which the network learned to denoise. The network may not handle such an input well, so its next prediction can add another error. The trajectory can then drift off the path that leads to a good image. This mismatch between training inputs and generated inputs is called exposure bias (Ning et al., 2023).

For Flow, a training input is $x_\alpha = (1-\alpha)x_0 + \alpha\epsilon$, and we write $\widehat{x}_\alpha$ for a generated state at the same noise level. An Euler step from a noise level α to a lower noise level β computes

$$\widehat{x}_\beta = \widehat{x}_\alpha + (\beta - \alpha)v_\theta(\widehat{x}_\alpha, \alpha). \tag{76}$$

An error in v_θ therefore changes $\widehat{x}_\beta$, and $\widehat{x}_\beta$ is the input of the next prediction. The loss measures the error on the training inputs x_α, and a lower loss there does not guarantee a lower error on the generated states $\widehat{x}_\alpha$. Drifting off the path means entering parts of the input space that the network rarely saw during training. Gaussian noise has no hard boundary, so these parts have a low density under the training distribution, but they are never completely absent. This is how the loss can improve while FID gets worse: the loss improves on the training inputs, while FID depends on how the network handles its own generated inputs.

Symptoms of exposure bias. We check for exposure bias in two ways. We check whether FID rises while the loss falls, and we check the generated images for blur or poorly formed objects. We do not measure exposure bias directly, so the explanations in this appendix describe our understanding of what we observe.

Three data regimes. In our experiments, how much exposure bias matters depends on the number of distinct training images, and we see three broad regimes. The counts below are distinct images, and not images processed over repeated epochs. For each regime, we also give our understanding of why it behaves as it does.

Fewer than about 1,000 images. The network learns the small distribution very quickly. Its prediction errors become small before a long separation between loss and FID can develop, so exposure bias is no real problem in these runs. The 50- and 500-image subsets of Section 6.1 fall in this regime.

About 10,000 to 100,000 images. This is the difficult regime. The network needs to train for a long time, but it has not yet learned to handle deviations broadly. It can become better at the supervised denoising task without becoming better at recovering from its own sampling errors. Exposure bias becomes a serious problem here, unless strong enough regularization makes the network robust to its own errors. Architectural inductive biases, dropout, and data augmentation are possible ways to provide this regularization. Full CIFAR-10, with 50,000 images, and the CelebA-HQ training split, with 28,000 images, fall in this regime.

More than about one million images. In our understanding, exposure bias still occurs here, but it matters less. The network generalizes broadly enough that a trajectory that drifts off the path can still end in a valid new image. The trajectory does not need to return to a particular training image, so going off the path does little harm. ImageNet falls in this regime, and for this reason we use it for the large-scale experiment of Section 6.3. A large dataset alone still does not guarantee robustness.

The boundaries between the regimes are rough, and we have not measured where one regime turns into the next. The 3,000- and 2,000-image subsets of Section 6.2 lie between the first two regimes, and they already show that the behavior also depends on the training budget and the network.

Why the main text leaves out the full datasets. In our understanding, schedules that lower the denoising loss more strongly also need stronger regularization to turn that lower loss into better images. EDM, for example, uses dropout and data augmentation to improve its FID (Karras et al., 2022), and like ours, this FID uses the training images as reference. Tuning the architecture, dropout, or augmentation for each schedule would mix the effect of the schedule with its interaction with the regularization. For this reason, the main text keeps full CIFAR-10 and CelebA-HQ out of the noise-schedule results, and we report them here instead.

Outline. The following sections look at the problem from four directions. Appendix E.2.2 adds weight decay to the training-budget sweep. Appendix E.2.3 uses a bigger network with twice the depth. Appendix E.2.4 replaces the convolutional network with a vision transformer. Appendix E.2.5 reports all six methods with the simple convolutional neural network on full CIFAR-10 and CelebA-HQ, and on ImageNet.

E.2.2 WEIGHT DECAY

In this section, we train Flow (uniform) with weight decay in the training-budget sweep of Section 6.2. We use three weight decay settings: 0, 0.01, and 0.1. Everything else stays as in Section 6.2: the width-64 simple convolutional neural network with layers $(2, 2, 2)$, batch size 256, learning rate 0.005, the training subsets, and the inference protocol. Every budget is a separately trained run, and we average three seeds at each budget. The training loss is the Flow-velocity MSE on the training subset, evaluated with fresh noise. FID compares 50,000 generated images with the same subset, using decoded images for CelebA-HQ.

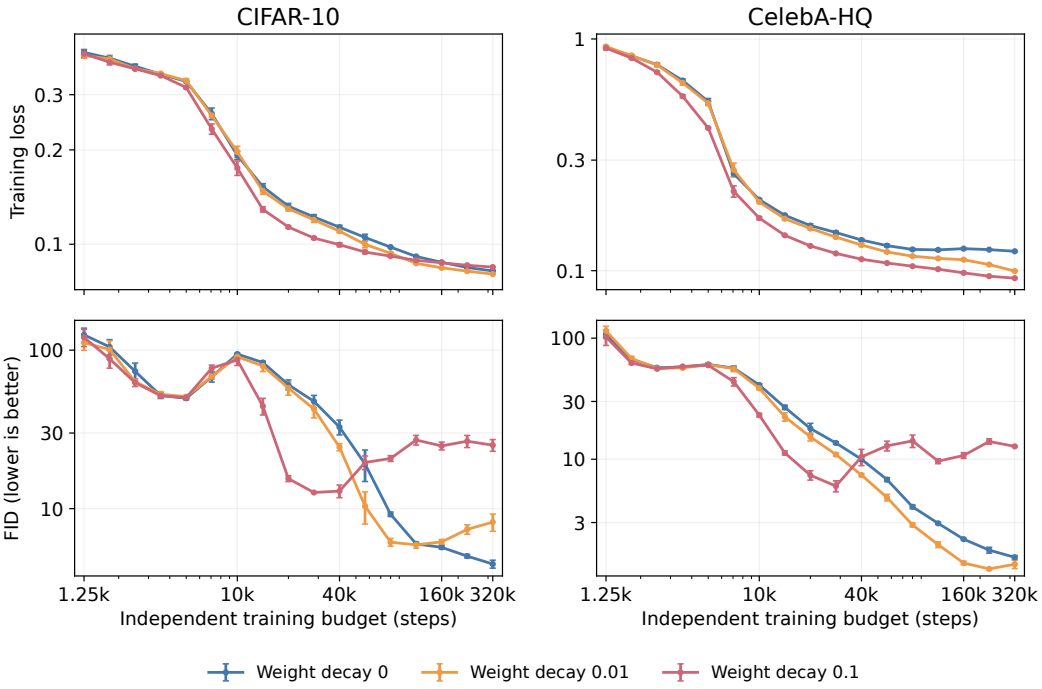

Figure 24: Training loss and FID of Flow (uniform) with three weight decay settings, for every training budget. Each point is the three-seed mean of independently trained runs, with s.e.m. whiskers. The training loss is the Flow-velocity MSE on the training subset, and CelebA-HQ uses decoded FID. All axes are logarithmic.

Figure 24 shows the training loss and FID for every training budget. With weight decay 0.1, the training loss falls faster than without weight decay, and FID first falls faster as well. After about 30k steps, however, its FID rises again, while its training loss keeps falling. At 320k steps, its training loss is lower than without weight decay on CelebA-HQ and about the same on CIFAR-10, yet its FID is much worse on both datasets. Weight decay 0.01 shows a milder form of the same pattern on CIFAR-10, where FID rises again over the largest budgets. On CelebA-HQ, it lowers both the training loss and FID.

Figures 25 and 26 compare the samples of weight decay 0.1 after 40k and 320k steps. Figures 27 and 28 compare weight decay 0 and 0.1 after 320k steps. All grids come from training seed 100, and each position starts from the same initial noise in every grid. With weight decay 0.1, many samples are blurred, and on CIFAR-10 some show only a noisy texture without an object. On CIFAR-10, these failed samples become more frequent from 40k to 320k steps. On CelebA-HQ, seed 100 has about the same FID at both budgets, because its FID already rises before 40k steps. Without weight decay, almost all samples after 320k steps are sharp.

With weight decay, training longer therefore seems to make the images worse, even though the training loss improves.

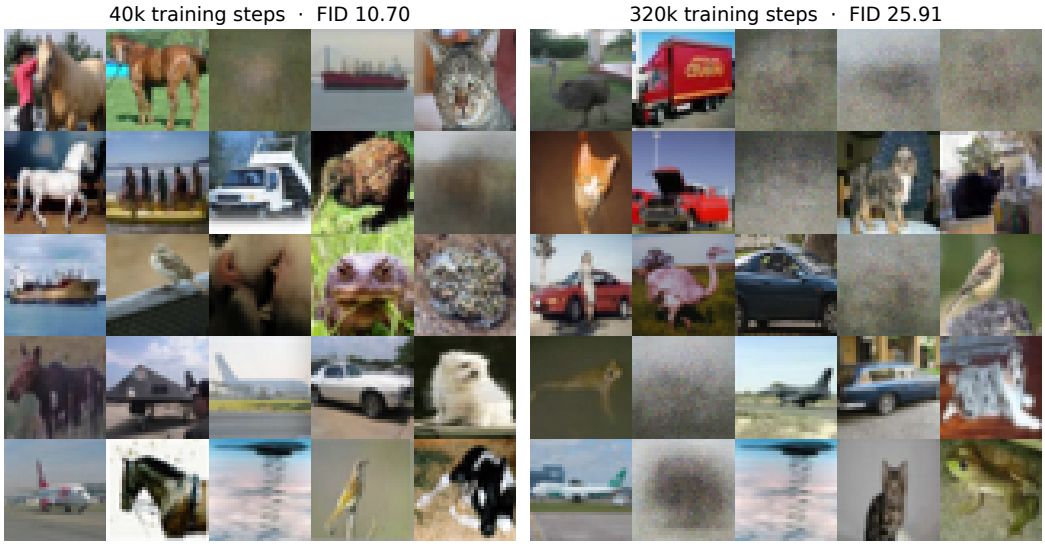

Figure 25: CIFAR-10, 3,000-image subset: Flow (uniform) with weight decay 0.1 after 40k and 320k training steps. Each grid shows all 25 validation samples of training seed 100 in generated order. The FID above each grid is the 50,000-sample FID of that run.

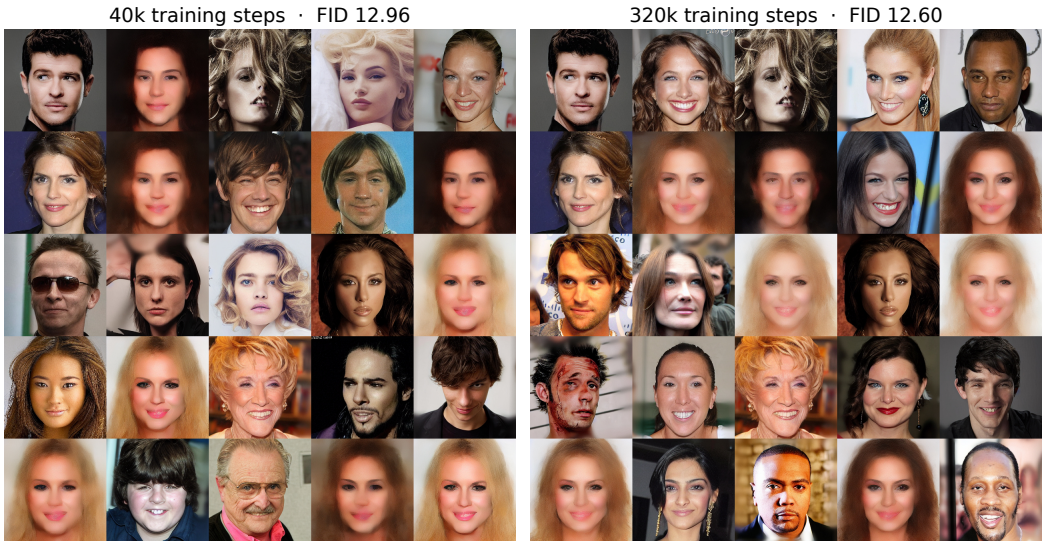

Figure 26: CelebA-HQ, 2,000-image subset: the same comparison as Figure 25, decoded from KL-f8 latents.



Figure 27: CIFAR-10, 3,000-image subset: Flow (uniform) with weight decay 0 and 0.1, both after 320k training steps. Both runs use training seed 100, and each position starts from the same initial noise. The FID above each grid is the 50,000-sample FID of that run.

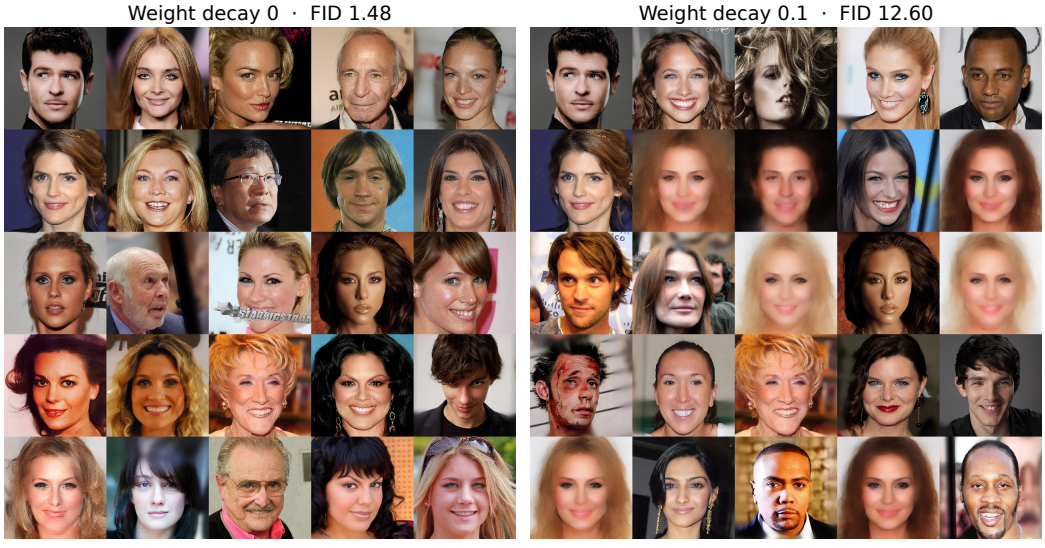

Figure 28: CelebA-HQ, 2,000-image subset: the same comparison as Figure 27, decoded from KL-f8 latents.

E.2.3 DEPTH

In this section, we repeat the analysis of Appendix E.2.2, but this time we vary the depth of the network. We compare the standard network with layers $(2, 2, 2)$ with a deeper network with layers $(4, 4, 4)$, which doubles the number of blocks at each resolution. Both networks use width 64 and no weight decay, and everything else stays as in Appendix E.2.2. The deeper network needs more compute per training step, so we compare the two networks at the same number of steps and not at the same compute.

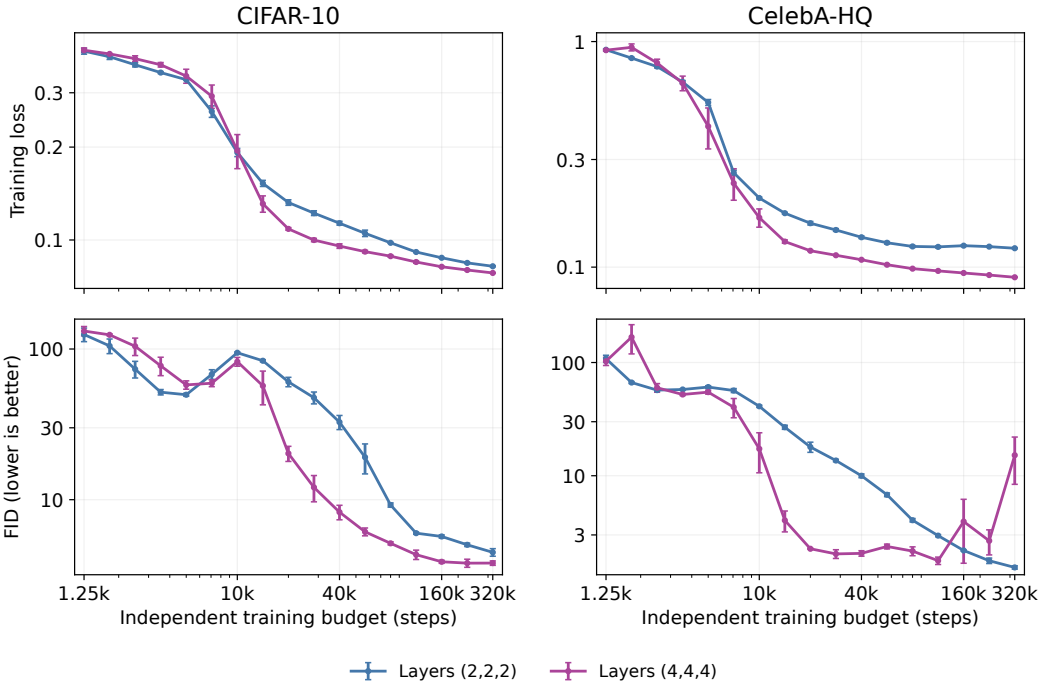

Figure 29: Training loss and FID of Flow (uniform) with layers $(2, 2, 2)$ and $(4, 4, 4)$, for every training budget. Means, s.e.m. whiskers, and axes are as in Figure 24.

Figure 29 shows the training loss and FID for every training budget. The deeper network has a lower training loss on both datasets, and its FID falls much faster. On CIFAR-10, its FID stays lower up to the largest budget. On CelebA-HQ, its FID stays low over a wide range of budgets, but it becomes unstable over the largest budgets. At 320k steps, its FID rises sharply in two of the three seeds, while its training loss keeps falling.

Figures 30 and 31 compare the samples of the two networks after 320k steps. As in Appendix E.2.2, both runs use training seed 100, and each position starts from the same initial noise. On CIFAR-10, both networks produce sharp samples. On CelebA-HQ, many samples of the deeper network are blurred, which is the same failure as with weight decay.

A deeper network therefore lowers the training loss on both datasets. On CIFAR-10, it also gives better images. On CelebA-HQ, training the deeper network longer makes the images worse, even though the training loss improves.



Figure 30: CIFAR-10, 3,000-image subset: Flow (uniform) with layers $(2, 2, 2)$ and $(4, 4, 4)$, both after 320k training steps. Both runs use training seed 100, and each position starts from the same initial noise. The FID above each grid is the 50,000-sample FID of that run.

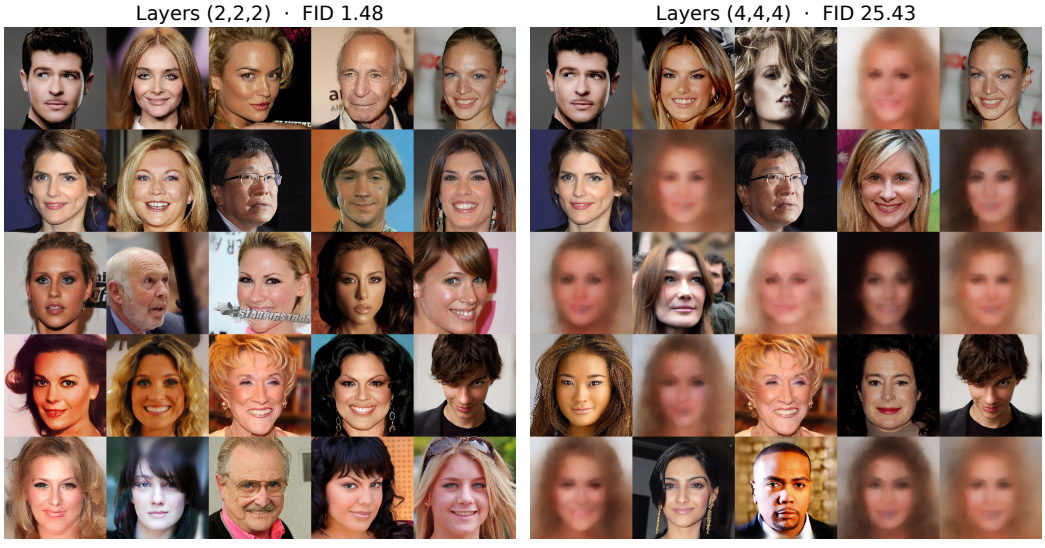

Figure 31: CelebA-HQ, 2,000-image subset: the same comparison as Figure 30, decoded from KL-f8 latents.

E.2.4 CONVOLUTIONAL NETWORK AND VISION TRANSFORMER

In this section, we repeat the analysis of Appendix E.2.2 with a different architecture. We replace the simple convolutional neural network with a simple vision transformer (ViT), which Appendix A.6.2 describes in detail. The ViT splits the input into 4×4 patches, which gives 64 tokens, and processes them with 12 transformer blocks of width 512. It has about the same number of parameters as the convolutional network, and a training step takes only about 10% longer. We trained the ViT only with weight decay 0.1, so we compare it with the convolutional network at the same weight decay. Everything else stays as in Appendix E.2.2.

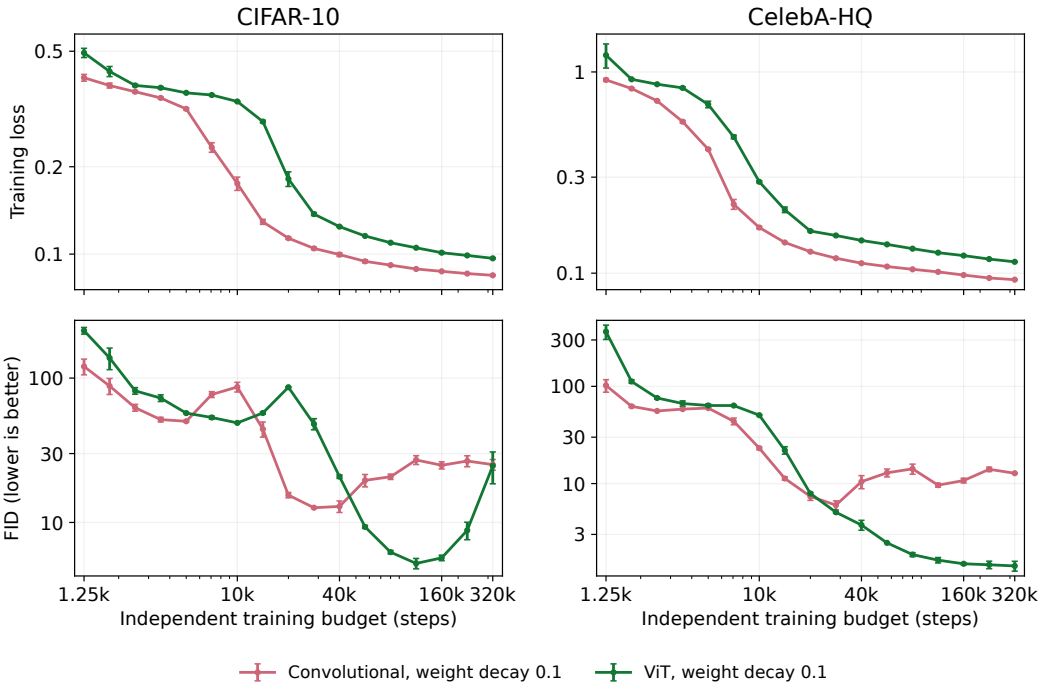

Figure 32: Training loss and FID of Flow (uniform) with the convolutional network and the ViT, both with weight decay 0.1, for every training budget. Means, s.e.m. whiskers, and axes are as in Figure 24.

Figure 32 shows the training loss and FID for every training budget. The ViT learns more slowly. Its training loss is higher at every budget, and its FID is mostly worse at small budgets. Over the larger budgets, however, its FID is much better than that of the convolutional network. On CIFAR-10, the ViT reaches a much lower FID, but over the largest budgets its FID rises again while its training loss keeps falling. This is the same failure as with the convolutional network, only later, and at 320k steps the two networks end with about the same FID. On CelebA-HQ, the FID of the ViT keeps falling up to the largest budget.

Figures 33 and 34 compare the samples of the two networks after 320k steps. As before, both runs use training seed 100, and each position starts from the same initial noise. On CIFAR-10, both networks produce the same kind of failed samples, with noisy textures and no object. On CelebA-HQ, the ViT produces sharp faces, while many faces of the convolutional network are blurred.

The architecture therefore matters, and it influences how strong the exposure bias is. With the same weight decay, the ViT avoids the failure on CelebA-HQ and delays it on CIFAR-10. This ties in with the regularization discussion of Appendix E.2.1: the architecture is one of the choices that one could tune to reduce exposure bias.

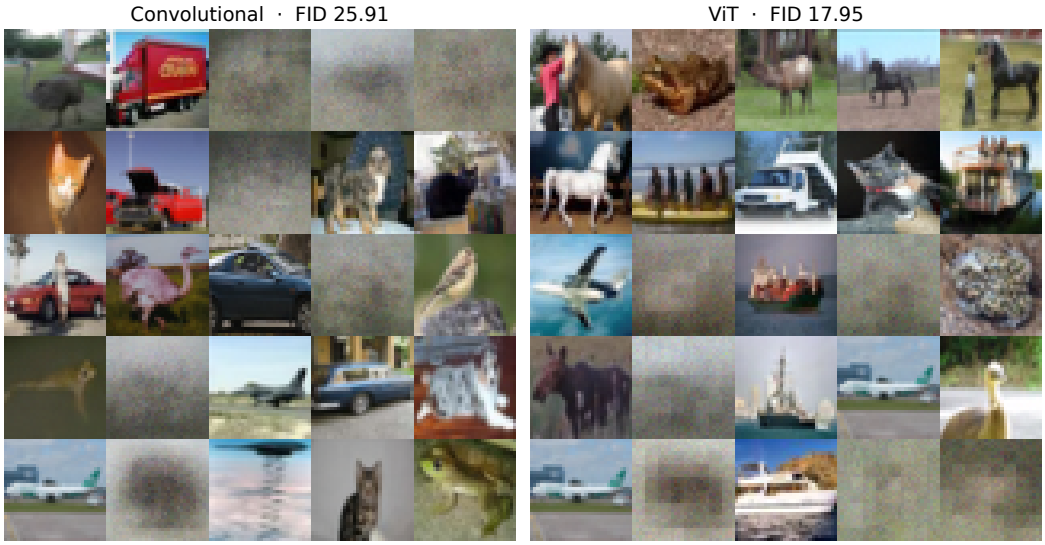

Figure 33: CIFAR-10, 3,000-image subset: Flow (uniform) with the convolutional network and the ViT, both with weight decay 0.1 and after 320k training steps. Both runs use training seed 100, and each position starts from the same initial noise. The FID above each grid is the 50,000-sample FID of that run.

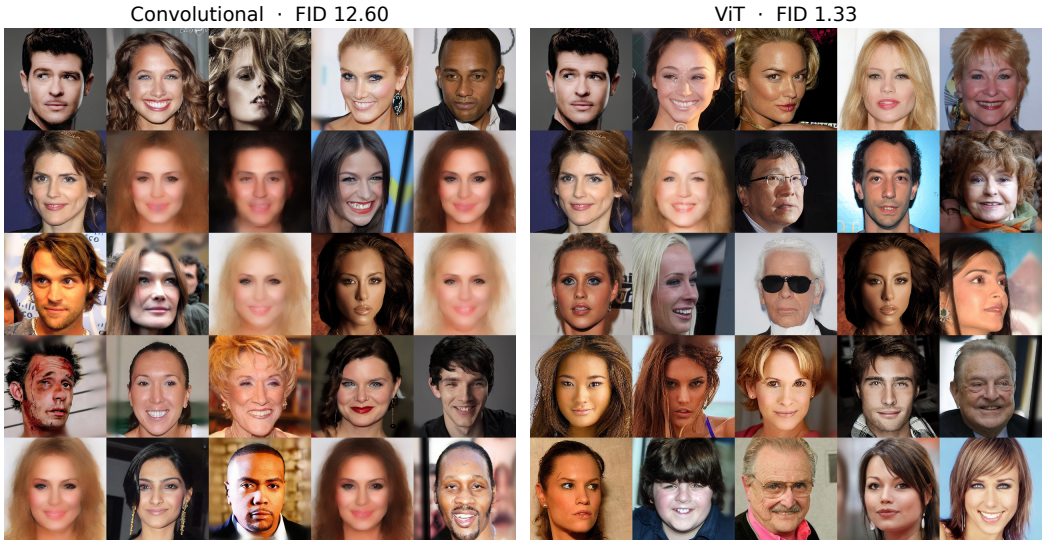

Figure 34: CelebA-HQ, 2,000-image subset: the same comparison as Figure 33, decoded from KL-f8 latents.

E.2.5 FULL-DATASET EXPERIMENTS

Finally, we train all six methods on the full datasets: CIFAR-10, the CelebA-HQ training split, and ImageNet. These experiments use the simple convolutional neural network and are separate from the LightningDiT experiments of Section 6.3.

Setup. Every method trains for one million steps with the simple convolutional neural network at width 256, with layers $(2, 2, 2)$ and batch size 256. We use AdamW with learning rate 0.001, no weight decay, and a cosine learning-rate schedule. There is one run per method and dataset. As in the rest of the paper, FID compares 50,000 generated images with the training images, and CelebA-HQ and ImageNet are decoded to pixels. ImageNet is class-conditional and uses no classifier-free guidance. We evaluate the training loss on all training images at 256 noise levels.

Table 14: FID of the full-dataset experiments after 1M steps. FID uses 50,000 generated images and the training images as reference, and CelebA-HQ and ImageNet are decoded to pixels. There is one run per method and dataset.

Method	CIFAR-10	CelebA-HQ	ImageNet
EDM	111.21	29.25	19.71
Flow (uniform)	64.78	5.74	17.70
Flow (logit-normal)	5.43	1.28	18.00
MinSNR	56.49	7.08	20.97
EDM2	10.14	2.00	26.37
Albis (ours)	8.30	1.44	17.87

Albis has a lower training loss than Flow (logit-normal), but a worse FID. Figures 35 and 36 compare the training loss of Flow (uniform), Flow (logit-normal), and Albis at every noise level. On CelebA-HQ, Albis has a lower training loss than Flow (logit-normal) at every single noise level. On CIFAR-10, it has the lower loss at almost every noise level. Albis therefore trains the better denoiser on both datasets. Still, its FID is worse on both datasets (Table 14). In our understanding, this is exposure bias. The lower loss holds on the training inputs, but it does not carry over to the inputs that the model generates itself. This is the same failure as in Appendices E.2.2 to E.2.4, where the training loss improves while FID gets worse.

The FIDs of the other methods. The FIDs of the other methods also make little sense. All six methods train with the same compute, yet their FIDs differ by more than an order of magnitude on both CIFAR-10 and CelebA-HQ. On CIFAR-10, several methods produce mostly blurred or poorly formed images (Figure 37). On CelebA-HQ, the differences are harder to see in the samples (Figure 38). With the same compute, such differences cannot be read as differences in the quality of the noise schedule. In our understanding, these FIDs mostly show how strongly each method suffers from exposure bias.

ImageNet. On ImageNet, the FIDs look normal. All six methods reach FIDs in a similar range, and their samples look alike (Figure 39). Albis has a lower training loss than Flow (logit-normal) at most noise levels and a slightly better FID. This fits the large-data regime of Appendix E.2.1, where exposure bias matters less. One million steps with this small network are far too little compute for ImageNet, however, so these results allow no firm conclusions.

Conclusion. On full CIFAR-10 and CelebA-HQ, a better training objective does not give better images, and the FIDs of the six methods do not reflect the quality of their noise schedules. With our setup, there is therefore nothing we can conclude about noise schedules in this range of dataset sizes. For this reason, the main text uses small subsets and ImageNet.

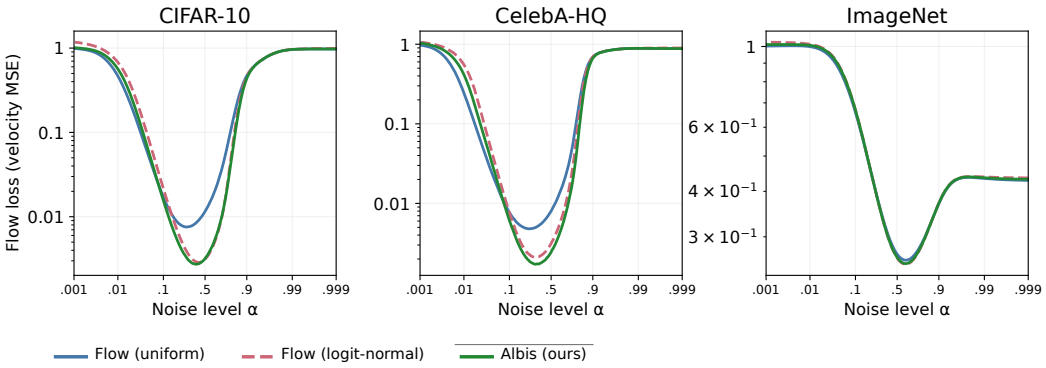

Figure 35: Training loss after 1M steps for Flow (uniform), Flow (logit-normal), and Albis at 256 noise levels, evaluated on all training images. The other methods were evaluated at their own noise levels and do not share this grid. The α axes use logit spacing, and the loss axes are logarithmic.

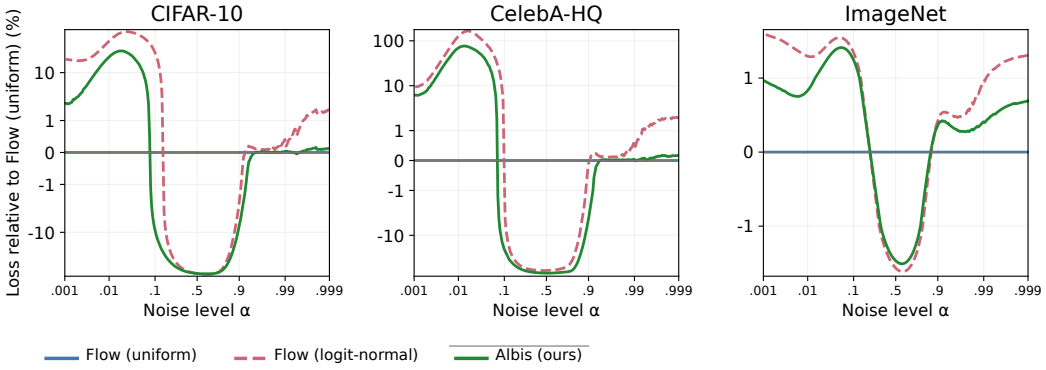

Figure 36: The losses of Figure 35 relative to Flow (uniform), in percent. Negative values mean a lower loss than Flow (uniform). The axis is symmetric-logarithmic and linear within $\pm 1\%$.



Figure 37: CIFAR-10 samples of the six methods after 1M steps, in generated order and without selection. The upper grid shows sample positions 0–9, and the lower grid positions 10–19. Rows are methods, and each row shows the FID of its method.



Figure 38: CelebA-HQ samples, decoded from KL-f8 latents, with the same layout as Figure 37.

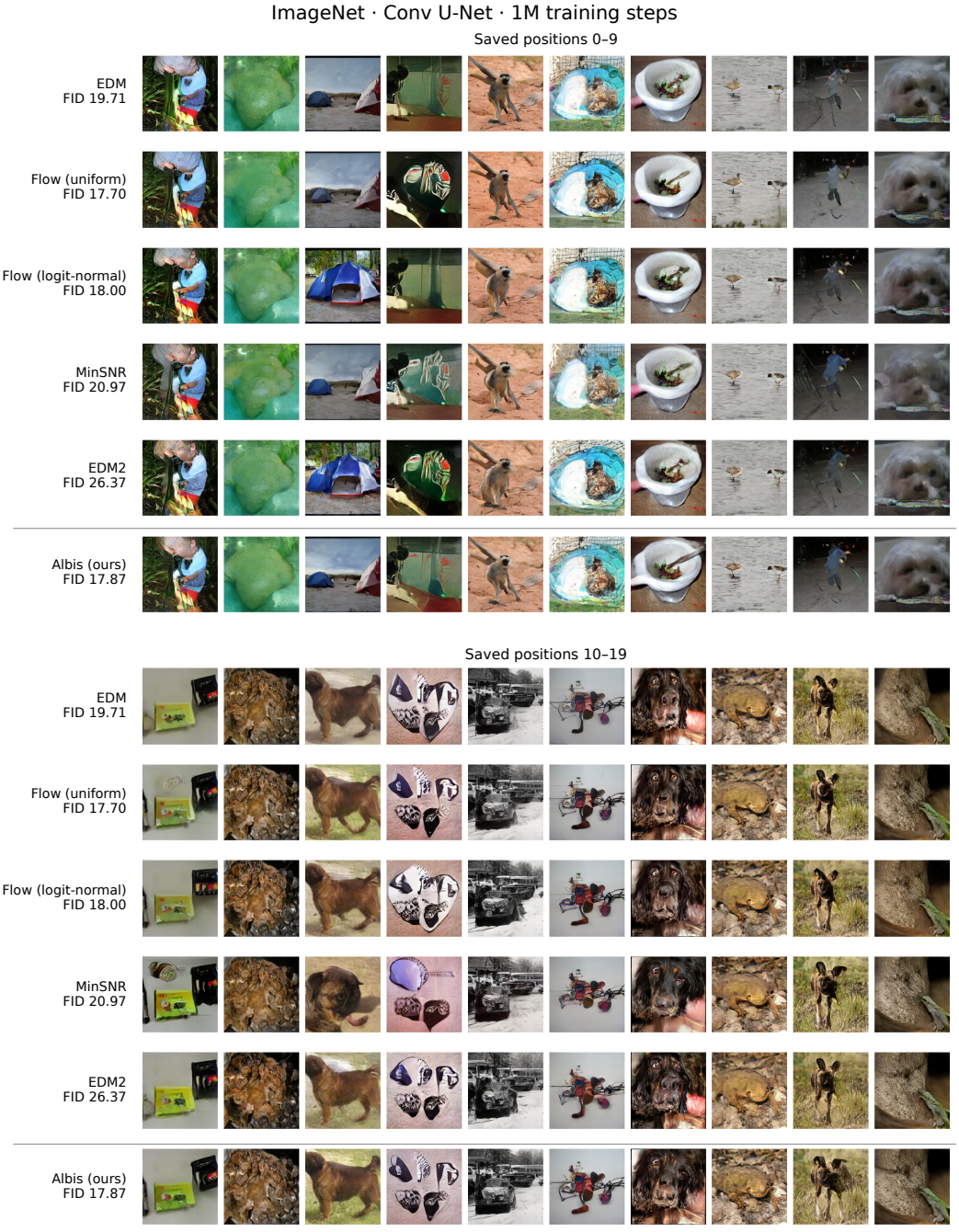

Figure 39: ImageNet samples, decoded from VA-VAE latents and generated without classifier-free guidance, with the same layout as Figure 37. These runs differ from the LightningDiT experiments of Section 6.3 in backbone, training budget, sampler, and evaluation.

E.3 IMAGENET NOISE LEVELS

Section 6.3 reports the ImageNet FID, and Table 15 repeats its unguided values. Here we look at the training loss of the six methods after 300k steps, at every noise level. Figure 40 shows the loss itself, and the curves almost overlap. Figure 41 therefore shows the loss relative to Flow (uniform).

Table 15: Unguided ImageNet FID at 300k steps.

Method	FID
EDM	3.35
Flow (uniform)	3.33
Flow (logit-normal)	3.26
MinSNR	4.03
EDM2	4.07
Albis (ours)	**3.13**

As on the small subsets (Appendix E.1), a method that lowers the loss in the middle pays for it in the tails, where Flow (uniform) and EDM have the lowest loss. Compared with Flow (logit-normal), Albis reaches almost the same loss in the middle, and at the same time it has a lower loss in the tails. Flow (logit-normal) keeps a small lead only just above the center of the noise range.

We saw the same behavior in the training-budget sweep of Section 6.2 (Figure 6). With a small training budget, Flow (logit-normal) still has a lower loss than Albis in the middle. As the budget grows, Albis takes the lead there as well, and from a large enough budget on it has the lower loss at every noise level. Our ImageNet runs are at the low-compute end of this picture. With more compute, we expect Albis to dominate Flow (logit-normal) at every noise level on ImageNet as well.

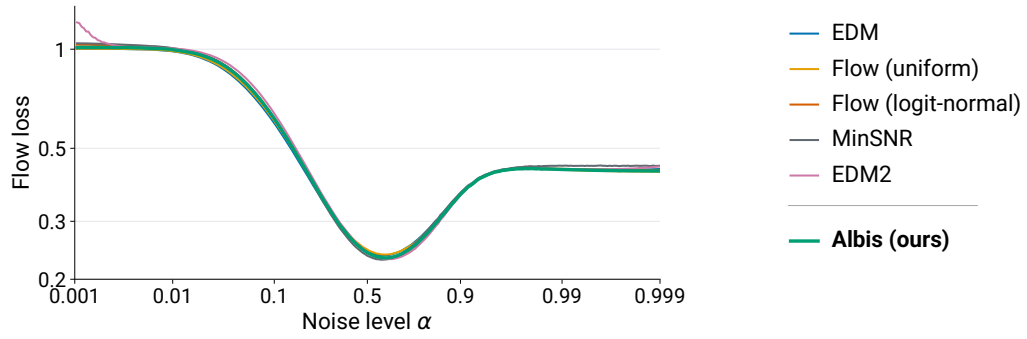

Figure 40: Training loss of the six ImageNet methods after 300k steps at 256 noise levels, evaluated on 8,192 training images. The α axis uses logit spacing, and the loss axis is logarithmic.

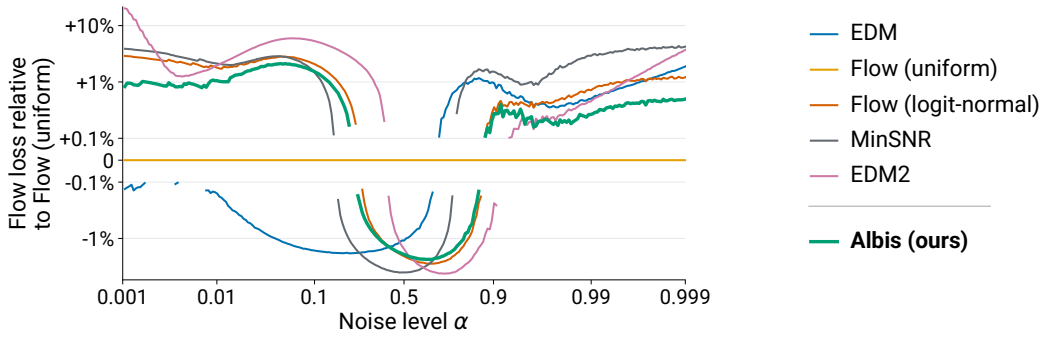

Figure 41: The losses of Figure 40 relative to Flow (uniform), in percent. Negative values mean a lower loss than Flow (uniform). The axis is symmetric-logarithmic and linear within $\pm 0.1\%$, and differences below 0.1% are hidden.

F Qualitative Results

This appendix shows generated samples for the comparisons in the paper. FID summarizes a whole distribution of images in one number, and the grids let the reader look at the images behind it. In every grid, a row is a method. The columns are sample positions that we fix in advance, so no sample is picked for its quality. The scaling grids are the one exception: their columns are training budgets, all shown at the same sample position.

The framework, curriculum, small-subset, and scaling grids use training seed 100 and evaluation seed 20. They take the first eight positions of the final 25-sample validation grid, and the scaling grids take position zero. CelebA-HQ samples are decoded to pixels with the same frozen KL-f8 autoencoder that the evaluation uses. The ImageNet grids show the first eight samples of all six methods, with guided and unguided sampling in separate figures. All ImageNet methods share the sampling seed and batch size.

Each grid comes from a single training run. The grids therefore show what one run produces, and the multi-seed tables show how much runs vary.

F.1 Frameworks: default and matched settings



Figure 42: Default and matched framework samples on CIFAR-10 with 500 training images after 20k steps, training seed 100. The upper three rows use the DDPM, EDM, and Flow defaults, and the lower three use the final matching controls: the noise objective, conditional normalization, and the inference grid. The finite-step trajectories still differ, with the endpoint caps noted in Section 3.

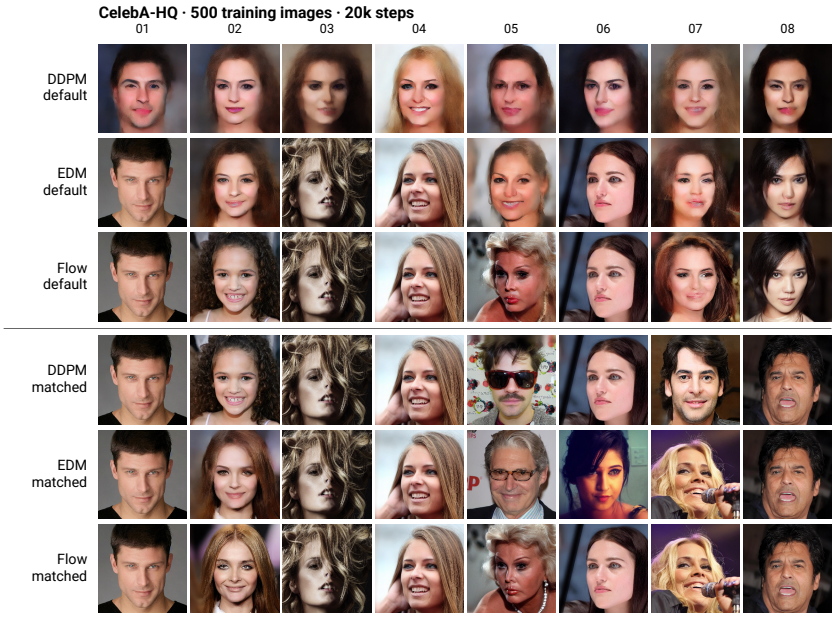

Figure 43: The same framework comparison on CelebA-HQ with 500 training images, decoded to pixels.

F.2 Curriculum: uniform and reversed scheduling interventions

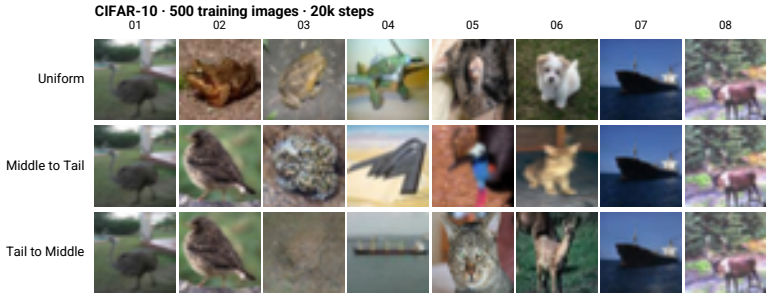

Figure 44: Curriculum comparison on CIFAR-10 with 500 training images after 20k steps, training seed 100. Rows are Flow (uniform), middle emphasis moving to the tails, and the reversed schedule (Section 4).

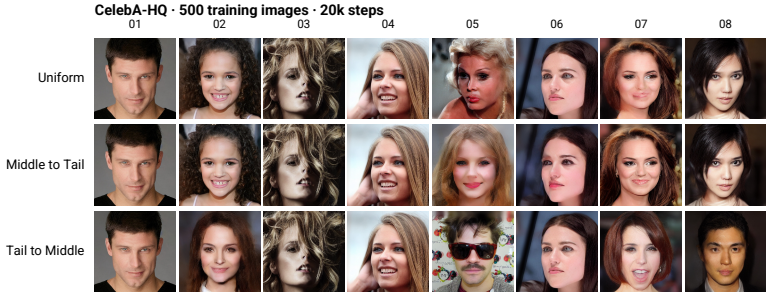

Figure 45: The same curriculum comparison on CelebA-HQ with 500 training images, decoded to pixels.

F.3 SMALL SUBSETS: ALL BASELINES AND SUBSET SIZES

The six comparisons cover both datasets and all three training-subset sizes from Table 2. Each layout has six method rows and eight sample columns, with a rule before Albis.

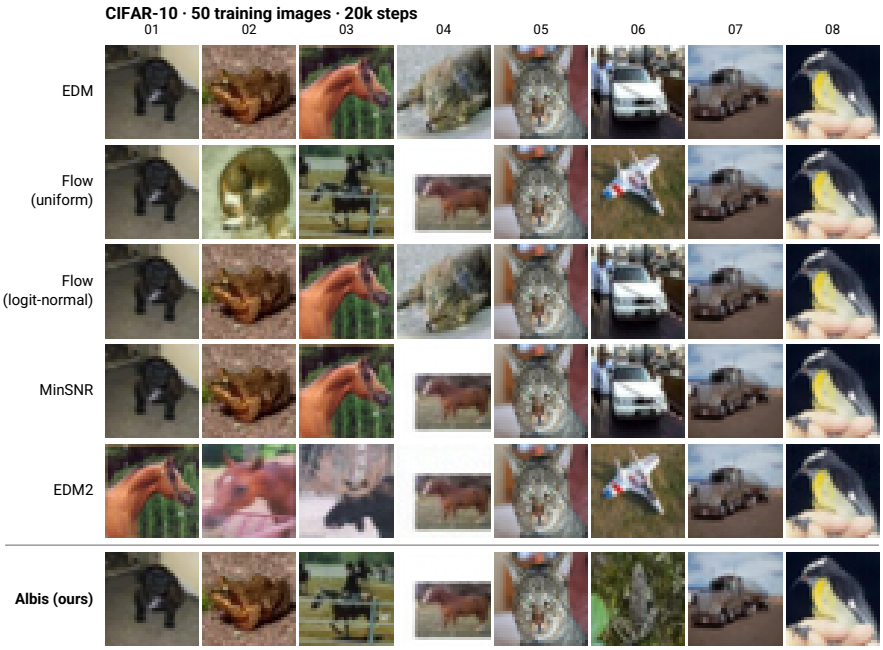

Figure 46: CIFAR-10 small-subset comparison with 50 training images after 20k steps, training seed 100. Columns use fixed sample positions across the six methods.



Figure 47: CIFAR-10 small-subset comparison with 500 training images, with the layout of Figure 46.

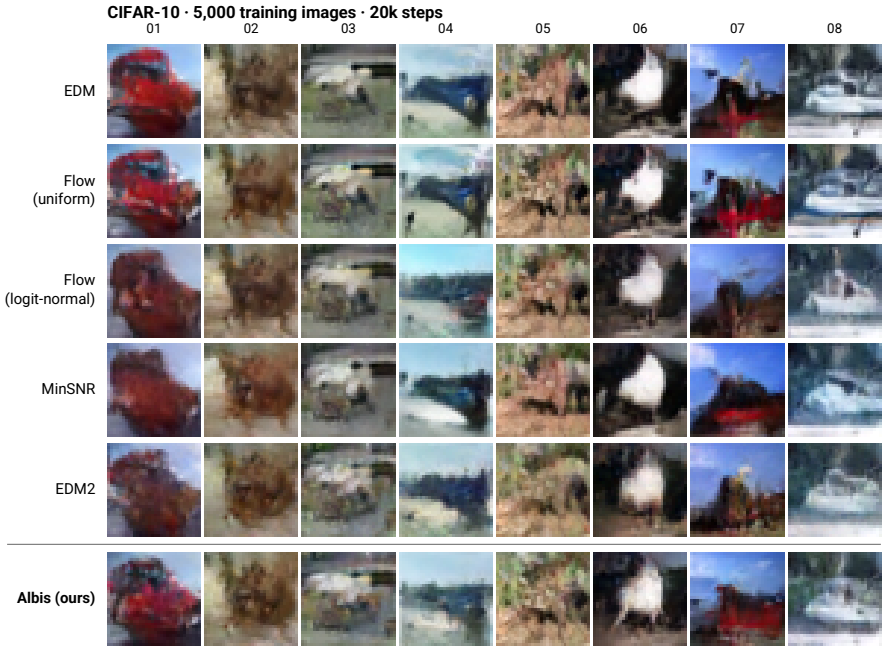

Figure 48: CIFAR-10 small-subset comparison with 5,000 training images, with the same layout. EDM has the lowest mean FID in this setting (Table 2).

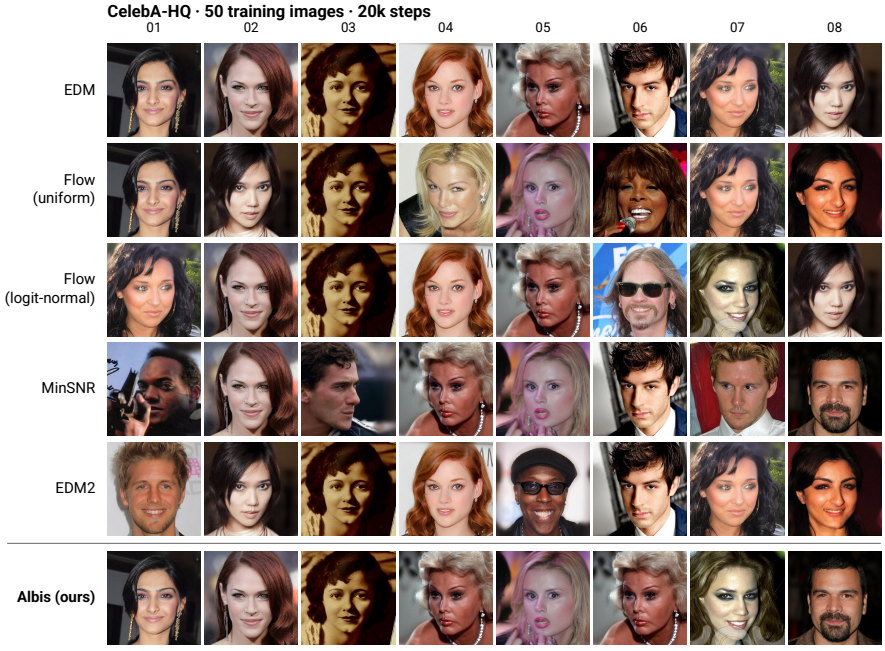

Figure 49: CelebA-HQ small-subset comparison with 50 training images after 20k steps, training seed 100, decoded to pixels. The layout matches the CIFAR-10 comparisons.

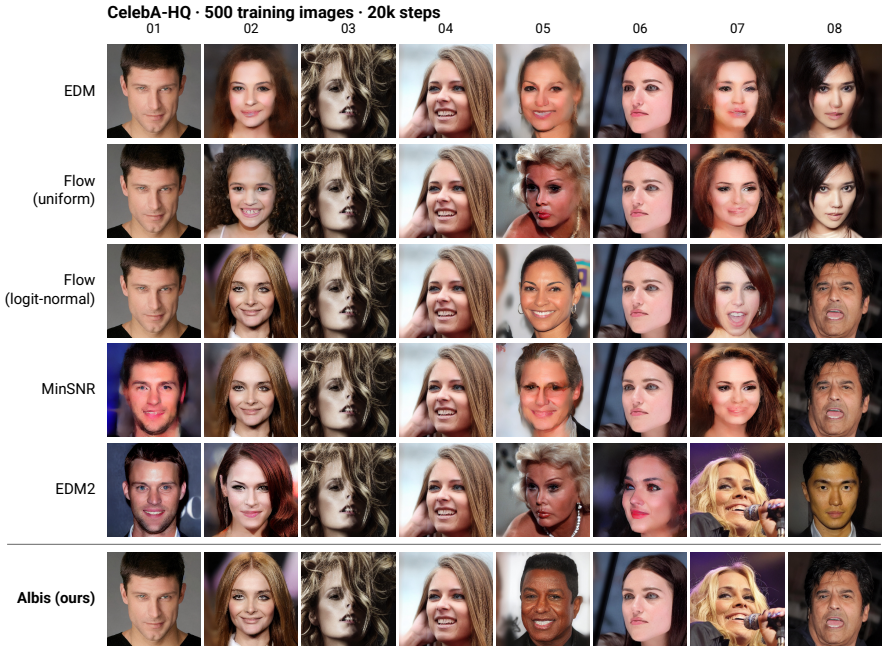

Figure 50: CelebA-HQ small-subset comparison with 500 training images, with the layout of Figure 49.



Figure 51: CelebA-HQ small-subset comparison with 5,000 training images, with the same layout. Flow (uniform) has the lowest mean FID in this setting (Table 2).

F.4 SCALING: A FIXED SAMPLE POSITION ACROSS TRAINING BUDGETS

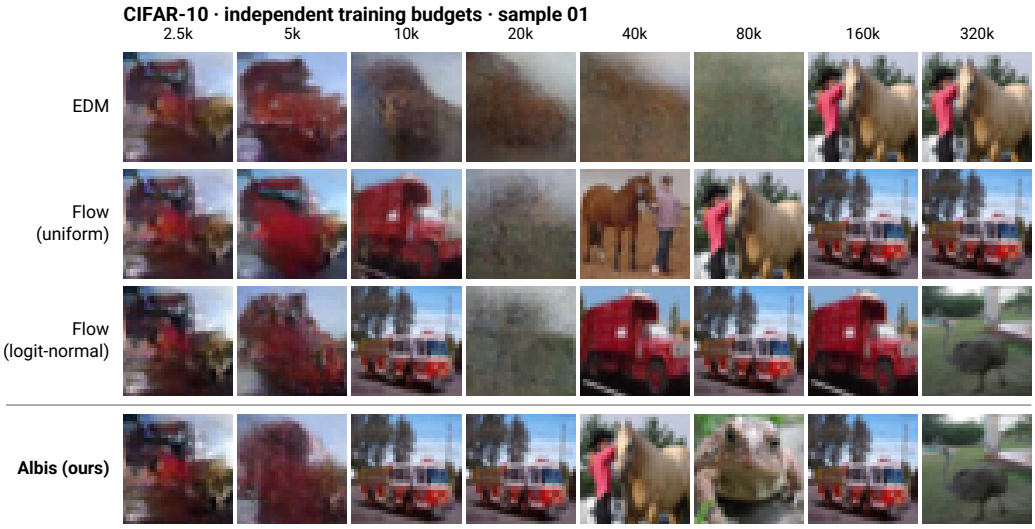

Figure 52: CIFAR-10 samples across independently trained budgets, using the 3,000-image subset, the width-64 simple convolutional neural network, and training seed 100. Every cell shows sample position zero. Rows are EDM, Flow (uniform), Flow (logit-normal), and Albis, and columns are training budgets, each a separate training run. The same sample position does not guarantee the same object across budgets.



Figure 53: The same comparison on the 2,000-image CelebA-HQ subset, decoded to pixels.

F.5 IMAGENET: ALL BASELINES



Figure 54: Unguided ImageNet-256 samples at 300k training steps. Rows follow EDM, Flow (uniform), Flow (logit-normal), MinSNR, EDM2, and Albis, and columns show the first eight sample positions. All methods use the shared DiT-XL/1 backbone with QK-Norm, the VA-VAE decoder, the 250-node Euler ODE, and timestep shift 0.3, without classifier-free guidance. Table 3 reports FID from the full 50,000-image evaluations.

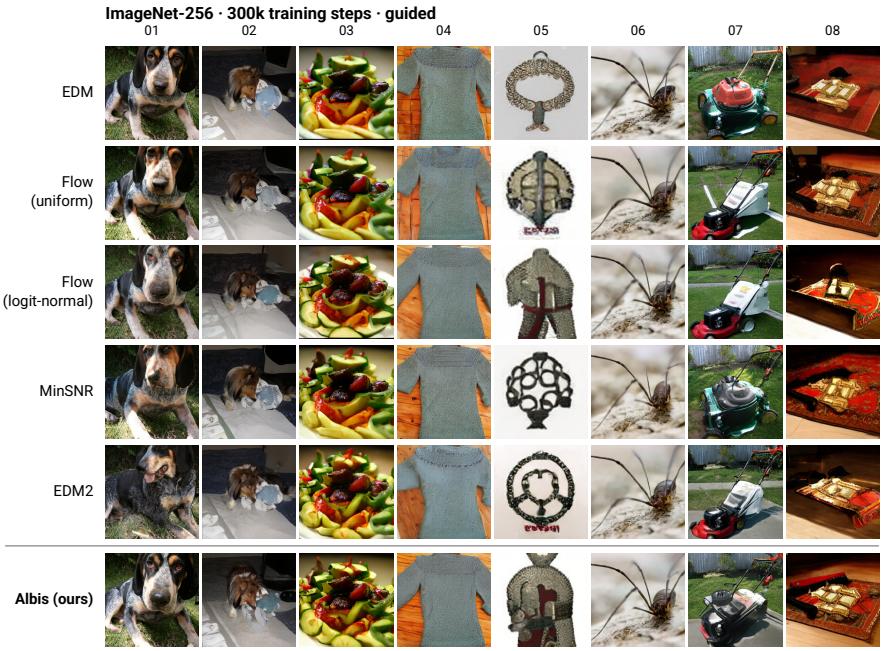

Figure 55: Guided ImageNet-256 samples at the same checkpoint and sample positions, with classifier-free guidance 6.7, interval setting 0.125, and the same solver.