SPHERE ENCODER 2

Anonymous authors
Paper under double-blind review

ABSTRACT

Sphere Encoder is an autoencoder that generates images by decoding random points from a high-dimensional latent sphere. We identify two limitations of the original formulation that reduce its generation quality. First, random points concentrate near the equator relative to the pole on an encoded latent, but the training rotation never reaches this region, leaving a gap that limits one-step generation. Second, training for generation with pixel-wise reconstruction loss encourages the decoder to average over plausible images, producing blurry images that lack high-frequency details. We present Sphere Encoder 2 to address both limitations, substantially improving image generation quality while maintaining the speed and simplicity of a autoencoder.

1 INTRODUCTION

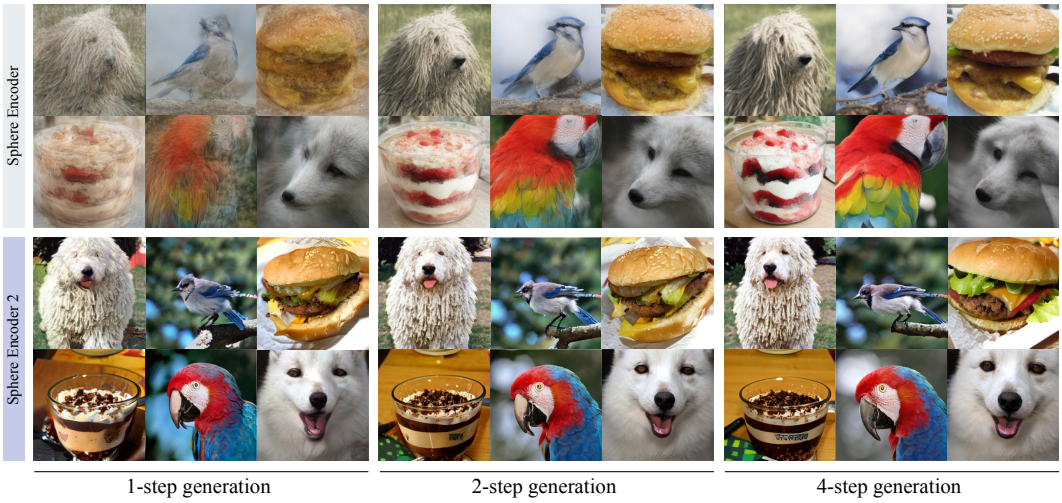

Figure 1: **Sphere Encoder vs. Sphere Encoder 2.** Uncurated generation on ImageNet 256×256 from Sphere Encoder (Yue et al., 2026) (top) and Sphere Encoder 2 (bottom) with the same class.

Sphere Encoders (Yue et al., 2026; Do et al., 2026; Man et al., 2026) are simple standalone autoencoders with strong generative capabilities. They consist of an encoder that maps images into a spherical latent space, and a decoder that maps a latent vector back to an image. When trained properly, the training images cover the latent sphere with an approximately uniform distribution, enabling generation of novel images by sampling a random latent vector and passing it through the decoder. Unlike the many-step generation process used by diffusion models, sphere encoders enable fast image generation in as little as one step.

The original Sphere Encoder (Yue et al., 2026) is able to represent the ImageNet distribution, but its single-step generations are blurry (Fig. 1, top). Sharp images were obtained by iteratively passing a generated image through the encoder and decoder as many as 4 times. We trace this to two causes. First, while noise is added to latent vectors during training to achieve good coverage of the sphere, the original Sphere Encoder only adds up to 85° of noise to latents, possibly leaving some of the sphere unseen during training. Second, the original Sphere Encoder trains the decoder with a pixel-wise reconstruction loss.

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



Figure 2: **Curated one-step generation** on ImageNet 512×512 by Sphere Encoder 2.

Sphere Encoder 2 fixes both issues (Fig. 1, bottom). First, we adopt a parameterization of noise that robustly adds a full 90° of noise to latents during training, allowing the decoder to see more of the sphere during training. Second, we decouple the decoder's task into two regimes: low-noise reconstruction, which keeps the pixel-wise loss, and high-noise generation, which drops it. In the high-noise/generative regime, we propose a new and efficient latent score matching loss. This encourages outputs to lie on the image manifold without requiring adversarial training. The result is sharp one-step generation on ImageNet 512×512 (Fig. 2).

Among pixel-space generators on ImageNet, Sphere Encoder 2 reaches the best FDr^6 at both 256×256 and 512×512 with at most 8 network evaluations (NFE), compared with 200 to 400 NFE for pixel diffusion models such as JiT (Li & He, 2026) and PixelNerd (Wang et al., 2026a), and its few-step quality approaches that of many-step latent diffusion at a small fraction of their sampling compute. Sphere Encoder 2 sits at the intersection of four lines of work: generative autoencoders, sampling without a learned prior, few-step generation without distillation, and distribution matching without a teacher. We discuss each in Sec. A.

2 Preliminaries: Sphere Encoder

Sphere Encoder (Yue et al., 2026) is a non-variational autoencoder that can be efficiently sampled to generate images. It consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. Given an input image $\mathbf{X}$[1] $\in \mathbb{R}^{H\times W\times 3}$, the encoder produces a latent $\mathbf{Z} = \mathcal{E}(\mathbf{X}) \in \mathbb{R}^{N\times D}$, where N is the number of tokens and D is the feature dimension. The latent $\mathbf{Z}$ is then projected onto a global sphere by the spherify function $\mathcal{F}$:

$$\mathbf{z} = \mathcal{F}(\mathbf{Z}) = \sqrt{d}\,\frac{\mathrm{vec}(\mathbf{Z})}{\|\,\mathrm{vec}(\mathbf{Z})\|_2} \in \sqrt{d}\,\mathbb{S}^{d-1}, \quad d = N \times D, \tag{1}$$

where $\mathbb{S}^{d-1}$ denotes the unit sphere in $\mathbb{R}^d$, so $\mathbf{z}$ lies on a sphere of radius $\sqrt{d}$. The decoder then maps the spherical latent back to the image space: $\hat{\mathbf{X}} = \mathcal{D}(\mathbf{z})$, reconstructing the original image.

For generation, the Sphere Encoder exploits the fact that the latent space is approximately uniformly distributed on the sphere, enabling the decoder to be run with a uniform sample as input. A latent vector is chosen by spherifying a random Gaussian vector, resulting in a uniformly sampled vector on the sphere (Blum et al., 2020), and the decoder transforms it into an image:

$$\hat{\mathbf{X}} = \mathcal{D}\big(\mathcal{F}(\mathbf{e})\big), \quad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \tag{2}$$

[1]Bold capital letters denote matrices, and bold lower-case letters denote column vectors. The notation $\mathbf{x}_i$ represents the ith column of a matrix $\mathbf{X}$, while x_{ij} denotes its entry in the ith row and j^{th} column. Non-bold letters denote scalars, and $\mathrm{vec}(\mathbf{X})$ denotes the vectorization of $\mathbf{X}$.

2.1 Gap to the Equator

For the direct generation in Eq. (2) to succeed, the decoder must handle latents that are far from any encoded $\mathbf{z}$ seen at training time. Sphere Encoder bridges the gap between the discrete training distribution and continuous uniform distribution using a process called spherify with noise, which adds noise to a clean latent $\mathbf{z}$ before train-time reconstruction:

$$\hat{\mathbf{z}}(\alpha) = \mathcal{F}\left(\mathbf{z} + \tan(\alpha)\, \mathcal{F}(\mathbf{e})\right), \tag{3}$$

where α approximates the angle between $\mathbf{z}$ and $\hat{\mathbf{z}}(\alpha)$ in high dimensions. During training, α is uniformly sampled from $[0, \alpha_{\max}]$ with $\alpha_{\max} = 85°$.

This objective leaves a mismatch between the latents seen in training and those used for generation. Specifically, on a high-dimensional sphere, almost all of the surface measure concentrates in a thin band around the equator relative to any fixed direction (Blum et al., 2020; Goldstein & Studer, 2018).

Relative to $\mathbf{z}$, a random point $\mathcal{F}(\mathbf{e})$ lies near the equator at $\alpha \to 90°$. With $\alpha_{\max} = 85°$, the decoder is never trained on the equatorial band, and the residual gap of $5°$ limits its one-step generation quality. Closing the gap by setting $\alpha_{\max} = 90°$ is not an option in Eq. (3), since $\tan(\alpha)$ diverges as $\alpha \to 90°$. This main limitation motivates a new formulation that reaches the equator exactly.

3 Sphere Encoder 2

In this section, we introduce Sphere Encoder 2, which improves generation quality by closing the gap to the equator. We explain its design choices through ablation studies.

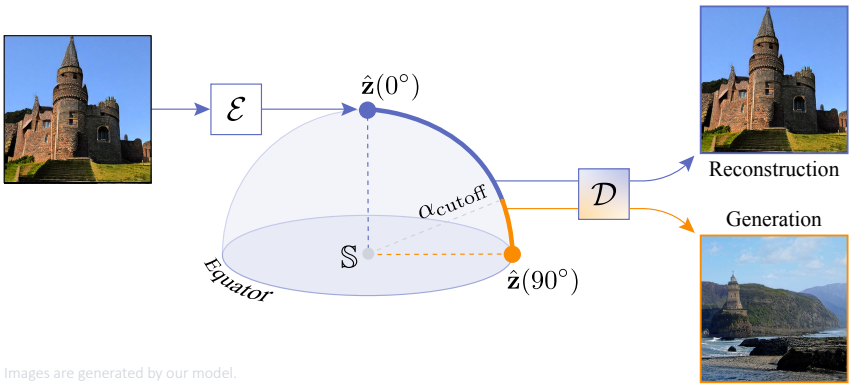

Figure 3: **Sphere Encoder 2.** The encoder $\mathcal{E}$ maps an image to a clean latent $\hat{\mathbf{z}}(0°)$ on the sphere $\mathbb{S}$, which is then rotated toward a random direction by an angle $\alpha \in [0, 90°]$ in Eq. (4). The cutoff angle α_{cutoff} splits the arc into a reconstruction regime ($\alpha \leqslant \alpha_{\text{cutoff}}$) and a generation regime ($\alpha \geqslant \alpha_{\text{cutoff}}$). The decoder $\mathcal{D}$ serves both: near the pole it reconstructs the input, and at the equator $\hat{\mathbf{z}}(90°) = \mathcal{F}(\mathbf{e})$ it generates a new image from a random latent in Eq. (2).

3.1 Rotation to the Equator

We replaces additive perturbation and reprojection in Eq. (3) with an explicit rotation:

$$\hat{\mathbf{z}}(\alpha) = \cos(\alpha)\, \mathbf{z} + \sin(\alpha)\, \mathcal{F}(\mathbf{e}), \quad \alpha \in [0, 90°]. \tag{4}$$

Since $\mathbf{z}$ and $\mathcal{F}(\mathbf{e})$ are nearly orthogonal in high dimensions, $\hat{\mathbf{z}}(\alpha)$ stays on the sphere of radius $\sqrt{d}$, and α is exactly the angle of rotation away from $\mathbf{z}$ toward $\mathcal{F}(\mathbf{e})$. The endpoints are well defined: $\hat{\mathbf{z}}(0°) = \mathbf{z}$ recovers the clean latent, and $\hat{\mathbf{z}}(90°) = \mathcal{F}(\mathbf{e})$ lands on the equator (see Fig. 3).

This objective closes the gap in two ways. First, the range of α now includes $90°$, so the decoder is trained on the equatorial band that one-step generation samples from. Second, although the distribution of the clean latents $\mathbf{z}$ is not necessarily uniform, $\hat{\mathbf{z}}(90°)$ is uniform on the equatorial subsphere relative to $\mathbf{z}$, which is exactly the distribution of the generation latents $\mathcal{F}(\mathbf{e})$ in Eq. (2).

3.2 Still an Autoencoder

Sphere Encoder 2 keeps the simplicity of a single encoder and decoder. We use a frozen DINOv3-small (Siméoni et al., 2025) as the encoder $\mathcal{E}$ for fast forward and high-quality generation. Each image is encoded into $N = 16 \times 16$ tokens with $D = 384$ features per token, giving a latent size of $d = 98{,}304$. The latent size is shared across image resolutions of 256×256 and 512×512, with patch sizes of 16×16 and 32×32 in pixel prediction head of the decoder, respectively.

Only the decoder $\mathcal{D}$, just a ViT (Dosovitskiy et al., 2021), is trained and scaled. We name models by decoder size, Sphere2-B and Sphere2-L for base and large decoders. The decoder is conditioned on the class embedding $\mathbf{y}$ and the rotation angle α, which allows it to adapt to different regimes along the rotation arc. Formally, unconditional generation Eq. (2) becomes

$$\hat{\mathbf{X}} = \mathcal{D}(\mathcal{F}(\mathbf{e}), \mathbf{y}, \alpha). \tag{5}$$

We embed α in the same way as diffusion timesteps (Nichol et al., 2021; Nichol & Dhariwal, 2021). The angle embedding is added to both the positional embedding of the input tokens and the class embedding used by adaLN-Zero (Peebles & Xie, 2023).

Alg. 1 summarizes the sampling forward. One-step generation decodes a random point on the sphere. For few-step generation, we repeatedly encode the generated image, rotate its latent toward the fixed noise direction $\mathcal{F}(\mathbf{e})$ by a fixed angle α, and decode it again. Unless otherwise specified, we set $\alpha = 85°$ in default for experiments and generate images without classifier-free guidance (CFG) (Ho & Salimans, 2022).

Algorithm 1 Sampling forward of Sphere Encoder 2.

```
e = Normal(0, 1).sample([d]) # random point

# one-step generation
x = D(F(e), y, alpha)

# few-step refinement
for _ in range(K - 1): # K steps in total
   z = cos(alpha) * E(x) + sin(alpha) * F(e)
   x = D(z, y, alpha)

return x
```

3.3 Angle Cutoff: Two Regimes on One Sphere

Sphere Encoder treats generation as reconstruction: the same pixel-wise reconstruction loss is applied to the rotated latent $\hat{\mathbf{z}}(\alpha)$ along the whole arc. A pixel-space loss, however, is minimized by the mean of all plausible images, so the decoder learns to generate mean images in one-step generation.

To address this issue, we simply split the arc at a cutoff α_{cutoff} into two regimes, each with its own training losses. The reconstruction regime covers the low angles, $\alpha \in [0, \alpha_{\text{cutoff}}]$, where the decoder is trained with the pixel-wise reconstruction loss:

$$\mathcal{L}_{\text{pix-recon}} = \mathcal{L}_{\text{LPIPS}}(\hat{\mathbf{X}}, \mathbf{X}) + \mathcal{L}_{\text{L1}}(\hat{\mathbf{X}}, \mathbf{X}), \tag{6}$$

where $\mathcal{L}_{\text{LPIPS}}$ is the perceptual loss (Zhang et al., 2018b). This regime keeps the clean latent $\hat{\mathbf{z}}(0°)$ faithful, and anchors the low-angle area so that rotating toward the equator starts from a semantically meaningful latent.

The generation regime covers angles $\alpha \in [\alpha_{\text{cutoff}}, 90°]$ near the equator. The generated images from this regime should stay semantically consistent with the input image, yet vary freely in appearance and structure. Before introducing the generation loss, we define a smooth cutoff function to weight the losses across the two regimes. The loss weight for generation regime is a standard sigmoid centered at α_{cutoff}:

$$w(\alpha; \alpha_{\text{cutoff}}, \beta) = \text{sigmoid}\left(\frac{\alpha - \alpha_{\text{cutoff}}}{\beta/4}\right). \tag{7}$$

The reconstruction regime receives $1 - w$. The bandwidth $\beta > 0$ controls the transition width: the scale $\beta/4$ makes w rise from 0.02 to 0.98 within $\pm\beta$ of the cutoff.

3.4 Taming the Generation Regime

Reconstruction has been well studied in prior work (Yao et al., 2025; Zheng et al., 2026; Esser et al., 2024; Yue et al., 2026), and is not the focus of this work. We keep the reconstruction regime

as is, without adversarial sharpening, so that any change in generation quality can be attributed to the generation losses alone. Our focus in this work is the generation regime, with the goal set above: along the rotation arc in Fig. 3, generated images should stay semantically consistent with the reconstruction while varying in appearance and structure.

3.4.1 Semantic Alignment

To preserve semantic consistency between the generation $\hat{\mathbf{X}}$ and the input $\mathbf{X}$, we align their logits from the ImageNet-pretrained ConvNext used in LPIPS (Zhang et al., 2018b):

$$\mathcal{L}_{\text{sem}} = \mathcal{L}_{\text{LPIPS-logits}}(\hat{\mathbf{X}}, \mathbf{X}). \tag{8}$$

This loss adds no cost: it shares the LPIPS perceptual loss forward in the reconstruction regime. It requires no labels, since the target is the input image itself rather than a class or a caption. In principle, a different frozen pretrained network with a global semantic embedding could also be used, such as the CLS token of CLIP (Radford et al., 2021), or SigLIP (Tschannen et al., 2025).

The cutoff angle α_{cutoff} decides how much of the arc is devoted to generation. Thus how far generated images may deviate from the input under the semantic alignment loss. We ablate this trade-off with a Sphere2-B trained on Oxford Flowers 256×256 (Nilsback & Zisserman, 2008) in Fig. 4.

First, in Fig. 4 left, reconstruction quality degrades as the cutoff moves past $45°$ toward the equator. This is expected: beyond $45°$, $\sin(\alpha) > \cos(\alpha)$ in Eq. (4), noise overwhelms the clean latent and the decoder cannot faithfully invert. Second, Fig. 4 middle and right show that 1-step and 4-step generation quality improves as α_{cutoff} moves toward the equator, peaking at $70°$. We use $\alpha_{\text{cutoff}} = 70°$ by default.

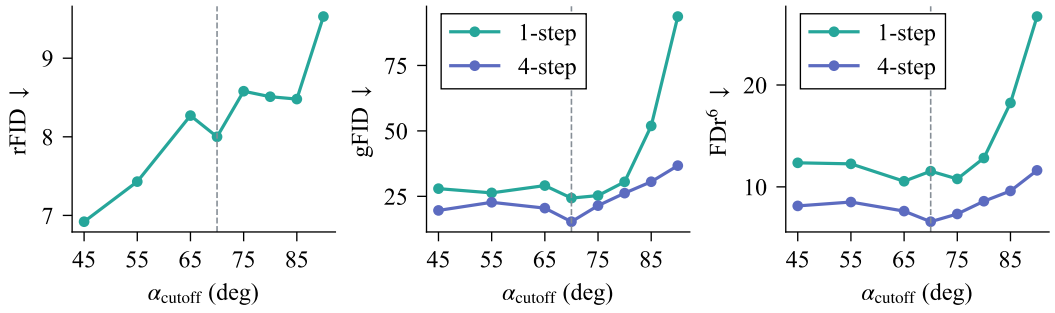

Figure 4: **Cutoff angle** α_{cutoff}, which splits the rotation arc into the reconstruction and generation regimes. Generation improves as the cutoff moves toward the equator until $70°$.

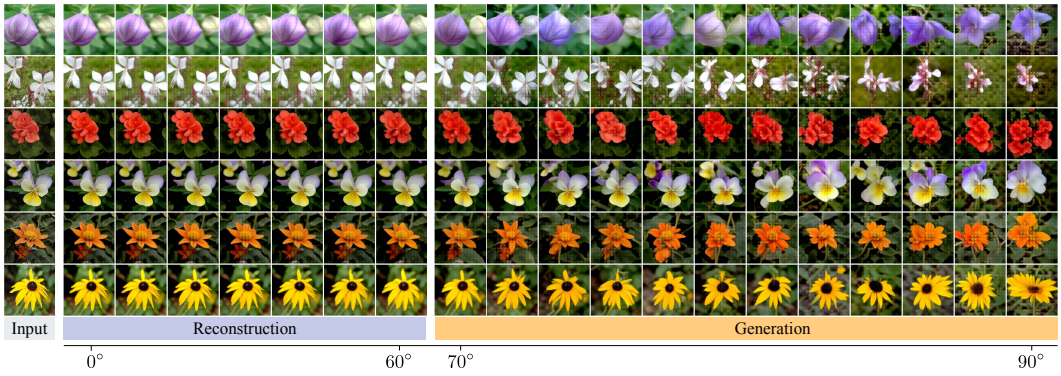

Figure 5: **Reconstruction and generation regimes** at different angles from $0°$ to $90°$ on Oxford Flowers 256×256. On the same rotation arc with $\alpha_{\text{cutoff}} = 70°$, the decoder reconstructs the input below the cutoff and generates new images above it.

Fig. 5 visualizes the decoder outputs along the rotation arc, given the input image on the left. Below $\alpha_{\text{cutoff}} = 70°$, the decoder reconstructs the input with minor variations; the softness is the intended

effect of omitting adversarial training. Beyond the cutoff, generated images remain semantically consistent with the input while differing in structure and style, which is exactly the behavior the generation regime is designed for. The deviation grows with the angle, and at 90° the output is a pure class-conditional sample, exactly the one-step generation in Eq. (2) at the equator. Current generated images are waffle-ish because the semantic alignment constrains what the decoder generates, but nothing yet drives it toward sharp, high-fidelity textures.

3.4.2 Fidelity Improvement

Improving synthesis fidelity with a loss is a long-standing problem in generative modeling. The adversarial loss (Goodfellow et al., 2020; Brock et al., 2018; Kang et al., 2023) remains the most effective approach, but it requires careful tuning and is prone to mode collapse over long training.

We seek a simple alternative with two properties: (1) it is stable over long training on large datasets, *e.g.*, thousands of epochs on ImageNet; and (2) it consistently yields effective fidelity supervision as the generator improves, rather than saturating once the generator catches up. Inspired by DMD (Yin et al., 2024; Deng et al., 2026), we propose a score matching loss that is agnostic to the generator and plugs in as a standalone loss module, in the same way as a GAN loss.

Like a GAN loss, it trains an auxiliary model online alongside the generator, but instead of a discriminator that classifies real from fake, it trains two lightweight score models, each only two transformer layers, that estimate the real and fake distributions in a frozen feature space; their difference gives the direction to move fake features toward real ones. Like a discriminator (Reed et al., 2016; Sauer et al., 2023; Zhang et al., 2018a; Kang et al., 2023; Sauer et al., 2024), the score models can be unconditional or conditioned on the class or text.

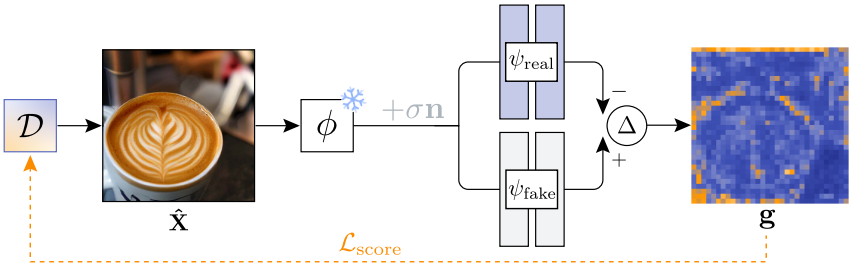

Figure 6: **Score matching loss.** Fake features from the frozen extractor ϕ are denoised by the real and fake score models, and their difference gives the direction $\mathbf{g}$ that pulls fake features toward real ones. During transport, both score models are frozen, and only the decoder $\mathcal{D}$ is updated by $\mathbf{g}$.

We extract features of the input image $\mathbf{X}$ (real) and the generated image $\hat{\mathbf{X}}$ (fake) from a frozen pretrained network $\phi(\cdot)$, such as ConvNeXt (Woo et al., 2023) or DINO (Caron et al., 2021):

$$\mathbf{v}_j = \frac{\phi_j(\mathbf{X}) - \mu_j}{s_j}, \qquad \hat{\mathbf{v}}_j = \frac{\phi_j(\hat{\mathbf{X}}) - \mu_j}{s_j}, \tag{9}$$

where j indicates the jth layer of ϕ. $\mathbf{v}_j, \hat{\mathbf{v}}_j \in \mathbb{R}^{L_j \times C_j}$ have L_j tokens and C_j channels. μ_j and s_j standardize j^{th}-layer features to zero mean and standard deviation σ_{data}, estimated on real features only. We drop j for simplicity in the following discussion.

We build two score models, ψ_{real} and ψ_{fake}, trained on the real features $\mathbf{v}$ and the fake features $\hat{\mathbf{v}}$, respectively. Both follow EDM (Karras et al., 2022) with its standard preconditioning, and each has only two transformer layers. For a feature $\mathbf{u} \in \{\mathbf{v}, \hat{\mathbf{v}}\}$ and a noise level σ, the noisy feature is

$$\mathbf{u}_\sigma = \mathbf{u} + \sigma\mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{10}$$

then ψ_{real} and ψ_{fake} learn to denoise $\mathbf{u}_\sigma$ with $\mathbf{E}_{\sigma,\mathbf{n}}\big[\lambda(\sigma)\|\psi(\mathbf{u}_\sigma, \sigma) - \mathbf{u}\|_2^2\big]$, where $\lambda(\sigma) = (\sigma^2 + \sigma_{\text{data}}^2)/(\sigma\,\sigma_{\text{data}})^2$ is the EDM loss weighting and σ follows the EDM log-normal $p(\sigma)$.

To supervise the generator, we freeze both score models, evaluate them on the noisy fake feature $\hat{\mathbf{v}}_\sigma$, and take their difference, as illustrated in Fig. 6:

$$\mathbf{g} = \frac{\psi_{\text{fake}}(\hat{\mathbf{v}}_\sigma, \sigma) - \psi_{\text{real}}(\hat{\mathbf{v}}_\sigma, \sigma)}{\|\hat{\mathbf{v}}_\sigma - \psi_{\text{real}}(\hat{\mathbf{v}}_\sigma, \sigma)\|_1 / LC}, \tag{11}$$

where the denominator is the denoising residual of the real score model, which normalizes the magnitude of $\mathbf{g}$ across noise levels following DMD (Yin et al., 2024). $\mathbf{g}$ carries no gradient and is applied to the generator through a proxy loss:

$$\mathcal{L}_{\text{score}} = \frac{1}{2}\|\hat{\mathbf{v}} - \text{sg}(\hat{\mathbf{v}} - \mathbf{g})\|_2^2, \tag{12}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. Its gradient with respect to $\hat{\mathbf{v}}$ is exactly $\mathbf{g}$, so a step moves the fake features toward the real distribution. In practice, we score two layers of ϕ, a middle and a deep one, and balance their losses so that each contributes a gradient of equal magnitude. By default, we use the frozen ConvNeXt V2-N (Woo et al., 2023) (14.98M) as ϕ and train the score models on its middle two layers, producing the module with just total 24.16M parameters. All details of the implementation are in Sec. B.

3.4.3 Latent Consistency

We also keep the latent consistency loss of Sphere Encoder (Yue et al., 2026), which encodes the generated image and matches its latent to that of the input:

$$\mathcal{L}_{\text{lat-con}} = 1 - \text{cos-sim}\big(\mathcal{E}(\hat{\mathbf{X}}), \mathcal{E}(\mathbf{X})\big). \tag{13}$$

Since $\mathcal{E}$ is the frozen DINOv3-small, this becomes semantic alignment again, now in the latent space rather than in the logit space of $\mathcal{L}_{\text{sem}}$. It adds one encoder forward on the generated image, which is cheap since DINOv3-small has only 29M parameters with trivial FLOPs.

3.4.4 Distribution Shift

$\mathcal{L}_{\text{score}}$ matches the fake and real distributions in one feature space only – ConvNeXt V2-N. In other feature spaces, we found that the statistics of generated images drift away from those of real images. This drift is invisible to the eye[2], but FID and FDr^6 are computed in exactly those spaces, so they are sensitive to it (Fig. 7). It is a mirror image of the metric paradox in Yang et al. (2026): their generators beat the metrics but not the eye, while ours satisfies the eye but not the metrics. Ideally, we could learn a feature space specific to ImageNet as drifting models do (Deng et al., 2026), or add score models for every feature space, but both are expensive and cumbersome. We seek a simple and cheap alternative.

To pin the statistics in place, we borrow an idea from FD loss (Yang et al., 2026) and add a Fréchet distance term on frozen features as a regularizer:

$$\mathcal{L}_{\text{FD-lite}} = \sum_i \frac{\text{FD}_{\phi_i}(\hat{\mathbf{X}}, \mathbf{X})}{\text{sg}\big(\text{FD}_{\phi_i}(\hat{\mathbf{X}}, \mathbf{X})\big) + \epsilon}, \tag{14}$$

where $\epsilon = 10^{-6}$ and FD_{ϕ_i} is the Fréchet distance (Heusel et al., 2017) between real and generated features from the ith frozen extractor ϕ_i. Each term is normalized by its own detached value. We track the running mean and covariance of real and fake features with exponential moving averages (EMA), and compute FD between them. As Fig. 7 shows, samples with and without this term are visually indistinguishable, yet their feature statistics move closer to those of real data in the spaces where the metrics are computed. This regularizer serves ImageNet rather than the eye, but it is simple and nearly free. It is also optional: generation quality is already in place without it, so a model judged by human preference rather than FID, *e.g.*, one for production, can drop it.

Compared to the original FD loss (Yang et al., 2026), we call ours $\mathcal{L}_{\text{FD-lite}}$ because it differs in purpose and, as a result, in cost. The original FD loss is a post-training objective: it is the only loss applied to a pretrained multi-step generator, and its job is to make one-step generation match the real distribution, which visibly and dramatically improves one-step quality. Ours does not improve visual quality at all. It runs alongside our other three losses throughout training with a small loss weight and only aims to keep the statistics of already plausible samples from drifting. Because it is a regularizer rather than the main objective, it does not need the heavy extractors of the original, namely Inception, SigLIP-so400M, and MAE-Large. We use three extractors: Inception, the encoder $\mathcal{E}$ (DINOv3-small, 29M), and the score feature extractor ϕ (ConvNeXt V2-N, 16M). The latter two are not in the FDr^6 feature spaces, and their forwards are already computed for $\mathcal{L}_{\text{lat-con}}$ and $\mathcal{L}_{\text{score}}$, so $\mathcal{L}_{\text{FD-lite}}$ adds almost no compute but effectively stabilize the statistics (Sec. C.6).

[2]We release Sphere2-B checkpoints trained with and without $\mathcal{L}_{\text{FD-lite}}$ in our repo for readers' own FDr👀.



FID 5.04 FDr^6 9.17 : $\mathcal{L}_{\text{sem}} + \mathcal{L}_{\text{score}} + \mathcal{L}_{\text{lat-con}}$



FID 3.66 FDr^6 8.12 : $\mathcal{L}_{\text{sem}} + \mathcal{L}_{\text{score}} + \mathcal{L}_{\text{lat-con}} + \mathcal{L}_{\text{FD-lite}}$

Figure 7: **Distribution shift is invisible but measurable.** 4-step generation on ImageNet 256×256 using Sphere2-B, trained with and without $\mathcal{L}_{\text{FD-lite}}$, are visually indistinguishable, yet the loss aligns the feature statistics of generations with those of real data, in which FID and FDr^6 measured. More uncurated samples for side by side comparison are in Fig. 11.



Figure 8: **Uncurated generated images on Oxford Flowers** 512×512 generated by Sphere2-L with 4 sampling steps.

Table 1: **Scaling the decoder on Oxford Flowers**. Compared with Sphere Encoder (Yue et al., 2026) and Sphere Latent Encoder (Do et al., 2026), Sphere2 achieves better FID with fewer parameters and fewer sampling steps.

Model	Params	CFG	NFE	FDr^6 ↓ @ K steps				gFID ↓ @ K steps			
				1	2	4	6	1	2	4	6
Image Size 256×256											
Sphere (2026)	948M	✓	$2K$	10.59	5.44	3.96	3.92	25.12	14.08	11.25	10.63
Sphere Latent (2026)	130M	✓	$2K$	–	–	–	–	–	12.22	8.61	7.85
Sphere2-B	159M	–	K	2.67	2.23	2.17	2.19	7.62	7.45	7.50	7.55
Sphere2-L	487M	–	K	2.19	1.85	1.82	1.83	9.36	8.33	7.98	7.93
Image Size 512×512											
Sphere2-B	161M	–	K	4.86	4.18	4.08	4.10	9.47	8.83	8.49	8.32
Sphere2-L	489M	–	K	4.22	3.67	3.60	3.61	9.56	8.34	7.70	7.59

4 Experiments

All ablation studies are described in Sec. C, including the choices of latent space, the effect of angle α, spherical CFG, the feature space and noise sigma distribution for $\mathcal{L}_{\text{score}}$. Tabs. 1 and 2 summarize the final results of Sphere Encoder 2 on Oxford Flowers and ImageNet, respectively. For both datasets, we generate 50K images for evaluation, and report FDr^6 and FID against the whole training set. Fig. 8 shows uncurated samples on Oxford Flowers 512×512, and Fig. 12 qualitatively

compares Sphere2-L with JiT-L/16 on ImageNet 256×256. More uncurated samples on ImageNet 512×512 are in Fig. 10.

5 Conclusion

Sphere encoders are an emerging paradigm with capabilities similar to diffusion models but with simpler and lower cost inference. Progress in this field has now been made my several teams (Do et al., 2026; Man et al., 2026). In this work, we show that the simplicity and speed of Sphere Encoders can be had while matching the image quality of far more expensive SOTA diffusions.

There are currently several other candidate architectures for single-step generation, most notably mean flow and drifting models. Compared to these candidates, Sphere Encoders still have their advantages. Mean flow schemes have a complex and intractable loss function which must be approximated using stop-gradients during training. The coarseness of this approximation results in difficulties with scaling and stability. Current implementations of drifting models explicitly require a discrete class ontology, as image batches draw multiple positive and negative samples from the same class. This discreteness requirement prevents current implementations from being transferred to the text-to-image setting. While these issues may be resolved by future research, we think that Sphere Encoders remain a leading candidate to replace diffusion in commercial settings, as they rely on simple sampling procedure and do not rely on the discreteness of the ImageNet ontology.

Table 2: **Scaling the decoder on ImageNet** 256×256 and 512×512. Parameter counts for Sphere2 include both the encoder and the decoder; for latent-space models they are listed as generator without the decoder. GFLOPs are measured for a single forward, so Total GFLOPs = GFLOPs $\times$ NFE. Among pixel-space models, the best FDr^6 per resolution is in **bold** and the second best is underlined. For Sphere2 models, the sampling angle and guidance angle are chosen by FDr^6, and gFID is reported at the same setting rather than tuned separately, since the two metrics prefer different settings.

Model	Params	CFG	GFLOPs	NFE	256×256		512×512	
					FDr^6 ↓	gFID ↓	FDr^6 ↓	gFID ↓
Latent space								
SiT-XL/2 (2024)	675M	✓	119	250×2	8.44	2.06	–	2.62
SiT-XL/2 + REPA (2024)	675M	✓	119	250×2	5.45	1.42	–	2.08
LightningDiT-XL/2 (2025)	675M	✓	119	250×2	4.57	1.35	–	–
DDT-XL/2 (2026b)	675M	✓	119	250×2	5.70	1.26	–	1.28
RAE-DiT^{DH}-XL/2 (2026)	839M	✓	146	50×2	3.26	1.13	–	1.13
MAR-L (2024)	478M	✓	129	256×2	6.68	1.80	–	–
MAR-H	942M	✓	252	256×2	5.61	1.56	–	–
Sphere Latent-XL (2026)	675M	✓	230	6×2	–	2.11	–	–
Drift-L (2026)	463M	✓	87	1	10.92	1.53	–	–
iMF-XL (2026)	610M	✓	174	1	8.39	1.82	–	–
iMF-XL	610M	✓	174	2	7.48	1.61	–	–
Pixel space								
PixelNerd-XL (2026a)	700M	✓	106	100×2	<u>5.01</u>	2.10	–	2.84
JiT-B (2026)	131M	✓	25	50×4	15.78	3.66	16.31	4.19
JiT-L	459M	✓	88	50×4	10.73	2.36	11.35	2.53
JiT-H	953M	✓	182	50×4	7.66	1.86		1.94
Sphere-XL (2026)	1.3B	✓	977	4×2	10.44	4.02	–	–
Drift-L	465M	✓	87	1	10.51	1.43	–	–
pMF-L (2026)	410M	✓	117	1	9.09	2.72	–	2.75
pMF-H	935M	✓	271	1	6.87	2.29	–	2.48
Sphere2-B	161M	–	22	1	12.57	3.31	13.91	3.85
		–		4	7.82	2.79	8.49	2.92
		✓		4×2	7.10	3.74	7.82	4.47
Sphere2-L	487M	–	115	1	13.96	5.15	11.65	3.69
		–		4	5.14	2.46	<u>6.60</u>	3.02
		✓		4×2	**4.92**	2.86	**5.99**	3.53

AI USE STATEMENT

In this work, we used generative AI tools for:

1. Implement methods. We use Claude and Codex for implementation. We take full responsibility for code correctness.
2. Support qualitative and thematic data analysis. We use Codex to help ploshing visualizations.

We have not used generative AI tools for:

1. Design or provide feedback on research methodology or experiments.
2. Clean and reformat datasets.
3. Interpret results.
4. Formulate mathematical claims.
5. Propose or refine hypotheses.

These required disclosure tasks are not applicable to this work:

1. Help develop theoretical models or conceptual frameworks.
2. Provide critical ingredients for proving mathematical claims.
3. Assist in the writing of proofs.
4. Assist with translation.

Additionally, we used generative AI tools for:

1. Editing the research paper to improve readability. We use chat models to polish our writing.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

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

Tung Do, Thuan Hoang Nguyen, and Hao Li. Efficient image synthesis with sphere latent encoder. In *arXiv preprint arXiv:2605.15592*, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

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

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

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. In *International Conference on Learning Representations*, 2025b.

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

Tom Goldstein and Christoph Studer. Phasemax: Convex phase retrieval via basis pursuit. *IEEE Transactions on Information Theory*, 64(4):2675–2689, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 2020.

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. *Advances in neural information processing systems*, 2017.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

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

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10124–10134. IEEE, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 4396–4405. IEEE, 2019.

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

Guolin Ke and Hui Xue. Hyperspherical latents improve continuous-token autoregressive generation. *arXiv preprint arXiv:2509.24335*, 2025.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Amandeep Kumar and Vishal M Patel. Learning on the manifold: Unlocking standard diffusion transformers with representation encoders. *arXiv preprint arXiv:2602.10099*, 2026.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37: 56424–56445, 2024.

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

Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. *arXiv preprint arXiv:2601.22158*, 2026.

Troy Luhman and Eric Luhman. High fidelity image synthesis with deep vaes in latent space. *arXiv preprint arXiv:2303.13714*, 2023.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024.

Sean Man, Ron Raphaeli, Matan Kleiner, and Or Ronai. SP^3: Spherical priors for plug-and-play restoration. *arXiv preprint arXiv:2606.16396*, 2026.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, 2008.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pp. 1060–1069. Pmlr, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *International conference on machine learning*, pp. 30105–30118. PMLR, 2023.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. In *arXiv preprint arXiv:2508.10104*, 2025.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International conference on artificial intelligence and statistics*, pp. 1214–1223. PMLR, 2018.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. In *arXiv preprint arXiv:2502.14786*, 2025.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *ICLR*, 2026a.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 40633–40642, 2026b.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16133–16142. Ieee, 2023.

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

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

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

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Kaiyu Yue, Menglin Jia, Ji Hou, and Tom Goldstein. Image generation with a sphere encoder. In *arXiv preprint arXiv:2602.15030*, 2026.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018a.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018b.

Deli Zhao, Jiapeng Zhu, and Bo Zhang. Latent variables on spheres for autoencoders in high dimensions. *arXiv preprint arXiv:1912.10233*, 2019.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in neural information processing systems*, 33:7559–7570, 2020.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024.

APPENDIX



1-step generation | 4-step generation

Figure 9: **Uncurated samples on Oxford Flowers** 512×512. Sphere2-L with 1 and 4 sampling steps and no CFG.



Figure 10: **Uncurated samples on ImageNet** 512×512. Sphere2-L with 4 sampling steps and no CFG.

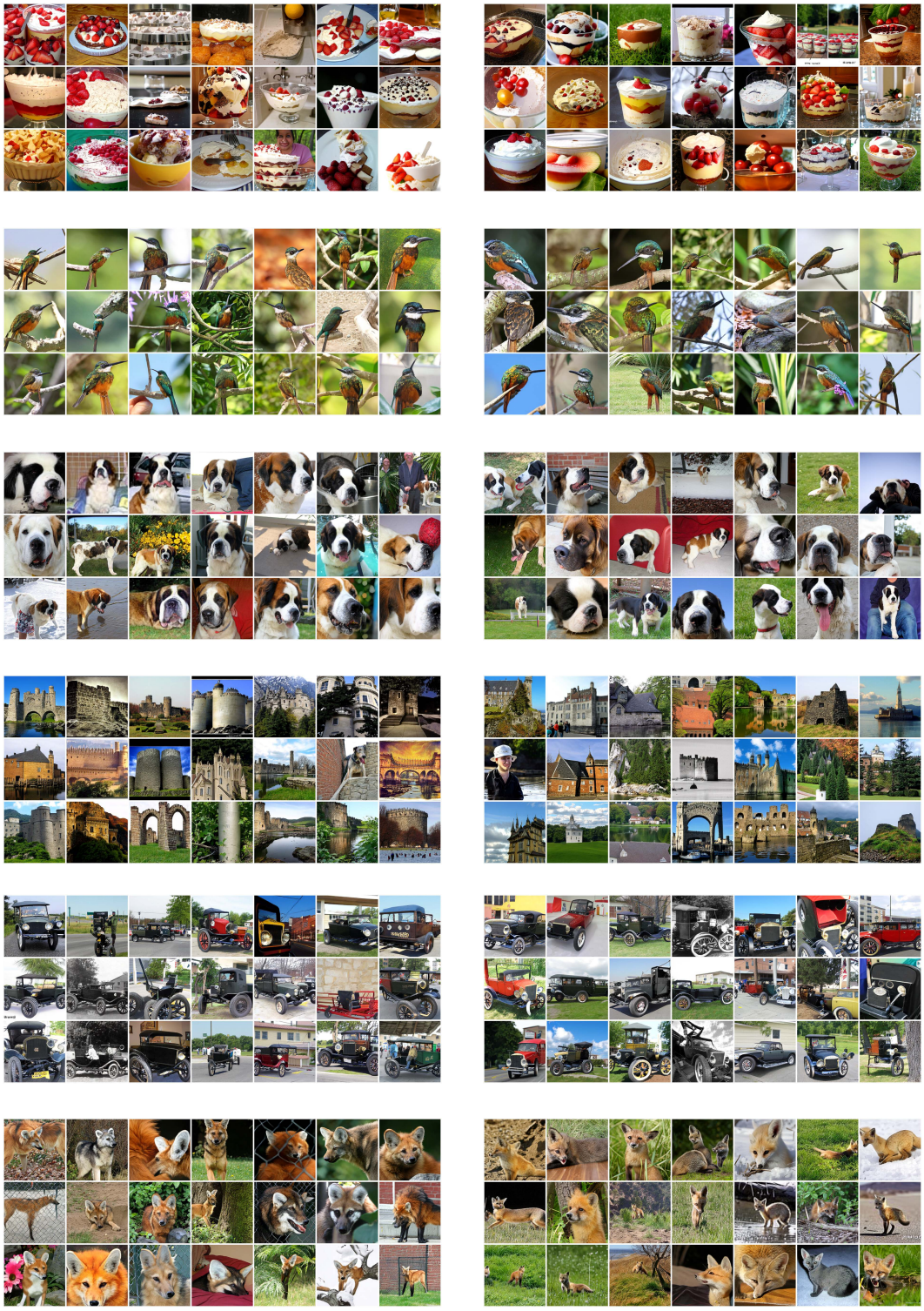

Figure 11: **Uncurated samples on ImageNet** 256×256. Left: Sphere2-B without $\mathcal{L}_{\text{FD-lite}}$ and NFE $= 4$. Right: Sphere2-B with $\mathcal{L}_{\text{FD-lite}}$ and NFE $= 4$. No CFG in sampling.

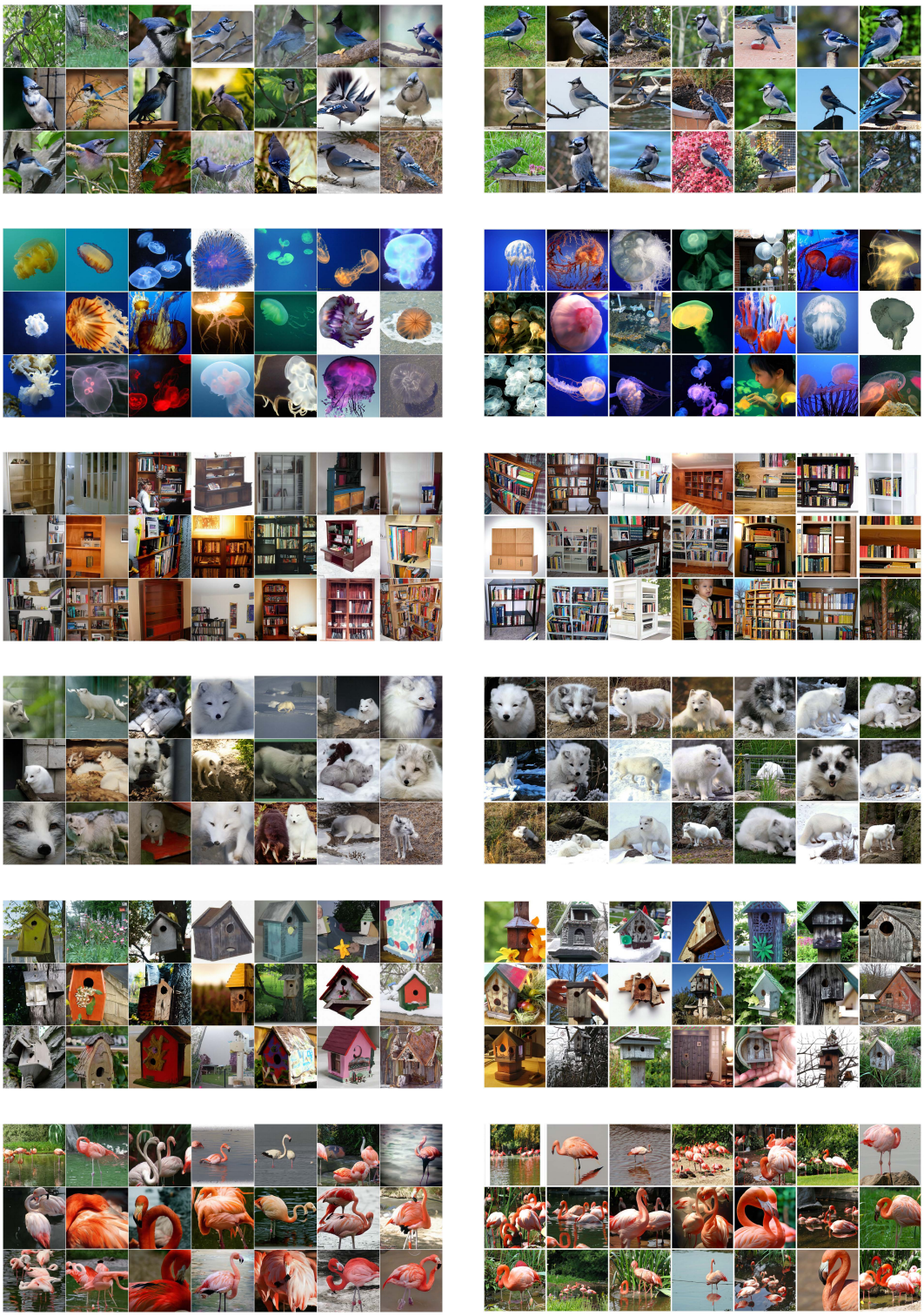

Figure 12: **Uncurated samples on ImageNet** 256×256. Left: JiT-L/16 with NFE $= 50 \times 4$ and CFG $= 2.4$. Right: Sphere2-L with NFE $= 2$ and no CFG.

A Related Work

Autoencoders (Bourlard & Kamp, 1988; Hinton & Zemel, 1993; Kingma & Welling, 2013) compress images into a latent space, but making them generate on their own is hard on complex datasets, *e.g.*, ImageNet (Deng et al., 2009). Hierarchical VAEs such as NVAE (Vahdat & Kautz, 2020) and VDVAE (Child, 2020) deepen the latent space, yet their generation quality stays far below diffusion models Dhariwal & Nichol (2021) and GANs (Sauer et al., 2023). Deep VAEs (Luhman & Luhman, 2023) reach high fidelity on ImageNet only when trained inside the latent space of a pretrained tokenizer. Eventually, autoencoders settled for the two-stage paradigm: an autoencoder only compresses, and a secondary latent diffusion Rombach et al. (2022) or autoregressive (AR) prior (Tian et al., 2024) generates. Sphere Encoder (Yue et al., 2026) revisits single-stage generation with a non-variational autoencoder and reaches competitive quality on ImageNet, but its generation is blurry because it treats generation as reconstruction. In this work, Sphere Encoder 2 still stays single-stage and pushes the quality closer to the two-stage paradigm.

Sampling from a prior is what turns an autoencoder into a generator, and most work learns a richer prior for it, from mixture priors (Tomczak & Welling, 2018) to the hierarchical priors above. Spherical latents have also been explored, but either at small scale with a von Mises-Fisher posterior (Davidson et al., 2018), or with an AR (Ke & Xue, 2025) or flow prior (Kumar & Patel, 2026) trained on top of the latents, which again returns to two stages. Same for Sphere Latent Encoder (Do et al., 2026), which builds on Sphere Encoder and samples from a spherified latent space of RAE (Zheng et al., 2026). StyleGAN (Karras et al., 2019) also normalizes its input noise onto a sphere, but it does not synthesize from the sphere directly: a learned mapping network first maps the random points into another space, which is a learned prior. All of the above still learn a prior on top of the sphere. Closest to ours, (Zhao et al., 2019) decode random points on a spherical latent directly, but only on MNIST and CelebA. Sphere Encoder shows that this works on ImageNet: in high dimensions a spherified Gaussian is already uniform on the sphere (Blum et al., 2020), so there is no prior to learn. Sphere Encoder 2 keeps it that way and improves generation only through the decoder's training objective.

One-step and few-step generation can be reached by two paths. Distillation compresses a pretrained multi-step teacher into a few steps (Song et al., 2023; Yin et al., 2024), while from-scratch methods train the few-step model directly, including consistency training (Song et al., 2023; Song & Dhariwal, 2024; Geng et al., 2025b; Lu & Song, 2025), MeanFlow and its variants (Geng et al., 2025a; 2026; Lu et al., 2026), and GANs (Goodfellow et al., 2020; Sauer et al., 2023). Sphere Encoder 2 belongs to the second group and needs no distillation. One-step generation is not a separate objective but the endpoint of the rotation arc at the equator (*e.g.*, random noise), and few-step refinement loops the encoder and decoder by re-encoding and rotating, which is a key difference from diffusion sampling procedures.

Distribution matching is the core of generative modeling for high-fidelity synthesis. Currently, the most effective methods are adversarial losses: the discriminator estimates the density of real and generated images, and provides a gradient to force the generator toward the data distribution (Goodfellow et al., 2020; Brock et al., 2018; Kang et al., 2023). But it needs careful tuning, architecture design, and is prone to mode collapse easily over long training. Distribution matching distillation (DMD) (Yin et al., 2024) proposes an alternative: a pretrained diffusion teacher provides the real score, a second model tracks the fake score online, and their difference moves samples toward the data distribution (Wang et al., 2023; Zhou et al., 2024). The cost is the teacher: DMD requires a pretrained diffusion model of the same data or a superset of it, so it is a distillation method rather than a standalone loss. We propose a lightweight score matching loss without a pretrained teacher. Both score models are two-layer transformers trained online from scratch, and they operate in a frozen feature space rather than pixel space, as in Projected GAN (Sauer et al., 2021) and drifting loss (Deng et al., 2026). Like a GAN loss, it is agnostic to the generator and plugs into any of them.

B Score Matching Loss

B.1 Differentiable Augmentation

We apply differentiable augmentations (Zhao et al., 2020) to the decoder output before it is passed to the frozen feature extractor ϕ during score model training. The decoder produces images of size 256×256 or 512×512, and we mainly investigate DINO-S/8 (Caron et al., 2021) and ConvNeXt V2-N(ano) (Woo et al., 2023) as the feature extractor. For DINO-S/8, instead of resizing the image to 224×224, we interpolate the positional embedding to match the image size. We use two augmentations: horizontal flip with probability 0.5, and translation with probability 1.0. The translation ratio is 0.015625 for both resolutions, i.e., up to 4 pixels at 256×256 and 8 pixels at 512×512, with reflect padding. The translation is for smoothing the boundaries between generated image patches.

B.2 Feature Standardization

Eq. (9) standardizes the output of each scored layer j of ϕ. Since the procedure is identical for every layer, we omit the index j here. The raw feature $\phi(\mathbf{X}) \in \mathbb{R}^{L \times C}$ has L spatial locations (ConvNeXt V2) or tokens (DINO) and C channels. EDM assumes its input has zero mean and standard deviation $\sigma_{\text{data}} = 0.5$, so we standardize the feature to this scale before it enters the score models.

Given a global batch of B real images, we compute the per-channel mean and second moment of the raw features,

$$m = \frac{1}{BL} \sum_{k=1}^{BL} \phi(\mathbf{X})^k, \qquad M = \frac{1}{BL} \sum_{k=1}^{BL} \big(\phi(\mathbf{X})^k\big)^2, \qquad m, M \in \mathbb{R}^C, \tag{15}$$

where k runs over the batch and spatial axes and the square is element-wise. The two statistics are accumulated as a running average over the first 1000 training steps and then frozen. Since ϕ is frozen, they converge quickly, and freezing them keeps the standardization fixed for the rest of training. They are estimated on real features only and applied to both real and fake features. This keeps the score models blind to the distribution difference between real and fake features, which is what the transport is meant to expose.

The standardization statistics of Eq. (9) are then

$$\mu = m, \qquad s = \frac{\sigma_{\text{feat}}}{\sigma_{\text{data}}}, \qquad \sigma_{\text{feat}} = \sqrt{M - m^2} \in \mathbb{R}^C, \tag{16}$$

so that $\mathbf{v} = (\phi(\mathbf{X}) - \mu)/s$ has zero mean and standard deviation σ_{data} in every channel. For ConvNeXt V2, we keep s per channel, because a few channels have far larger variance than the rest. This is because the Global Response Normalization leaves one channel dominating the others in magnitude. Each channel's standard deviation is floored at 0.1 times the median over channels, so a nearly constant channel is not amplified. For DINO, the channels are naturally balanced, and we use a single scalar per layer, the root mean of the per-channel variances.

B.3 Transport Direction

The direction $\mathbf{g}$ in Eq. (11) is computed per sample. Its denominator is the mean absolute denoising error of the real score model over the LC entries of the feature map, floored at a small constant,

$$\mathbf{g} = \frac{\psi_{\text{fake}}(\hat{\mathbf{v}}_\sigma, \sigma) - \psi_{\text{real}}(\hat{\mathbf{v}}_\sigma, \sigma)}{n}, \qquad n = \max\Big(\frac{1}{LC}\big\|\hat{\mathbf{v}} - \psi_{\text{real}}(\hat{\mathbf{v}}_\sigma, \sigma)\big\|_1,\ 10^{-2}\Big). \tag{17}$$

The floor only guards against division by a vanishing error when a sample already lies on the real manifold at this σ; in practice it sits well below the typical value of n. The noise level σ is drawn from $q(\sigma)$ and clamped to $[0.002, 8.0]$. Both score models are frozen here, so $\mathbf{g}$ carries no gradient.

B.4 Transport Balance

Eq. (12) is applied to every scored layer j of ϕ. In practice we average the squared error over the $L_j C_j$ entries,

$$\mathcal{L}_j = \frac{1}{2L_jC_j} \|\hat{\mathbf{v}}_j - \text{sg}(\hat{\mathbf{v}}_j - \mathbf{g}_j)\|_2^2, \qquad \frac{\partial \mathcal{L}_j}{\partial \hat{\mathbf{v}}_j} = \frac{1}{L_jC_j}\, \mathbf{g}_j, \tag{18}$$

which differs from Eq. (12) by a constant $1/(L_j C_j)$ that the balance weights below absorb. Since $\hat{\mathbf{v}}_j$ is standardized with divisor $s_j = \sigma_{\text{feat},j}/\sigma_{\text{data}}$ in Eq. (9), $\partial \hat{\mathbf{v}}_j / \partial \phi_j = 1/s_j$, and the gradient reaching the generated image is

$$\frac{\partial \mathcal{L}_j}{\partial \hat{\mathbf{X}}} = \frac{1}{L_j C_j} \frac{1}{s_j} \left(\frac{\partial \phi_j(\hat{\mathbf{X}})}{\partial \hat{\mathbf{X}}} \right)^{\top} \mathbf{g}_j. \tag{19}$$

Its magnitude therefore scales as

$$a_j \triangleq \frac{\|\mathbf{g}_j\|_2}{L_j C_j \, s_j}, \qquad \left\| \partial \mathcal{L}_j / \partial \hat{\mathbf{X}} \right\|_2 \le \kappa_j \, a_j, \quad \kappa_j = \left\| \partial \phi_j / \partial \hat{\mathbf{X}} \right\|_2. \tag{20}$$

All three factors of a_j depend on the layer: $L_j C_j$ and s_j are known, and $\|\mathbf{g}_j\|_2$ is measured on the fly. For per-channel standardization, s_j is the scalar $\left(\frac{1}{C_j} \sum_c s_{j,c}^{-2}\right)^{-1/2}$.

Given a target magnitude T, we weight each layer by

$$w_j = \frac{T}{a_j}, \qquad \mathcal{L}_{\text{score}} = \frac{1}{J} \sum_{j=1}^{J} w_j \, \mathcal{L}_j, \tag{21}$$

so that $w_j a_j = T$ for every layer, and the average over the J scored layers keeps the loss weight independent of how many layers are used. Because the regression target is detached, $\mathbf{g}_j$ does not depend on w_j, and scaling $\mathcal{L}_j$ by w_j scales its gradient by exactly w_j, so no custom backward is needed.

Since $\|\mathbf{g}_j\|_2$ drifts as the generator improves, we track it with a running mean over the first 1000 transport steps on Oxford Flowers and 10000 on ImageNet, and an EMA with decay 0.99 afterwards. The target T is set to the mean of the a_j during those 1000 steps and then frozen. The balance thus equalizes the layers and, at the same time, holds the overall gradient magnitude at its calibrated level rather than letting it fade as the fake features approach the real ones. w_j is clipped to $[0.05, 20]$ as a guard against a collapsed layer taking unbounded weight.

C Ablation Studies

C.1 Angle Conditioning

$\mathcal{L}_{\text{score}}$ asks the decoder to cover the real distribution rather than collapse onto its mean. We hypothesize that an angle-blind decoder hedges toward the mean, since it cannot tell latents at different angles apart. Tab. 3 ablates conditioning the decoder on α – finding our hypothesis is correct. Angle conditioning effectively improves metrics, so we keep it by default.

Table 3: **Angle conditioning** in the Sphere2-B trained on Oxford Flowers 256×256.

α cond	FDr^6 ↓			gFID ↓		
	1	2	4	1	2	4
-	4.00	3.21	3.23	14.02	12.09	11.95
✓	**3.92**	**3.14**	**3.04**	**10.29**	**8.24**	**7.51**

C.2 Angle Sampling

Diffusion models commonly concentrate training samples in the high-noise region, via a noise-level (Li & He, 2026) or timestep shift (Esser et al., 2024). In our case, high noise corresponds to the equator. Sphere Latent Encoder (Do et al., 2026) likewise found that concentrating samples at high angles improves generation. One may expect the same here, since the generation regime lives near the equator. To test this expectation, we control the sampling density of α during training with a single parameter by drawing it from a rational curve:

$$\alpha = 90^\circ \cdot \frac{(k+1)\,r}{k\,r+1}, \quad r \sim \mathcal{U}[0,1], \tag{22}$$

where $k > -1$ sets the shape of the curve (Fig. 13 left). Setting $k = 0$ gives uniform sampling. A positive k concentrates samples near the equator, and a negative k near the pole.

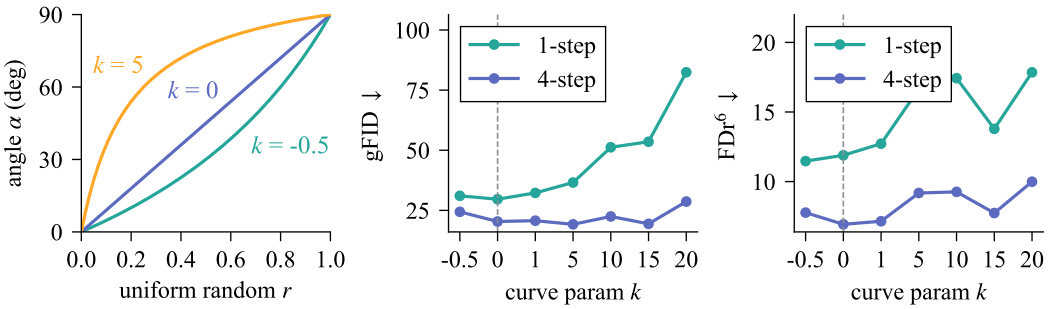

Figure 13: **Angle sampling curve**. Left: sampling curves for different k. Middle and right: generation quality on Oxford Flowers with different k. Uniform sampling ($k = 0$) is the best choice.

Fig. 13 middle and right tell a different story from common practice. Uniform sampling ($k = 0$) gives the best rFID, the best 1-step gFID, and the best FDr^6 at every step count. Concentrating samples near the equator ($k > 0$) degrades reconstruction and 1-step generation, and buys only a marginal gain in multi-step gFID at $k = 5$. We attribute this to the angle cutoff: each regime already receives a loss tailored to it, so the decoder no longer needs an uneven sampling density to prioritize the equator. We therefore use uniform sampling by default, as the simplest and most effective choice.

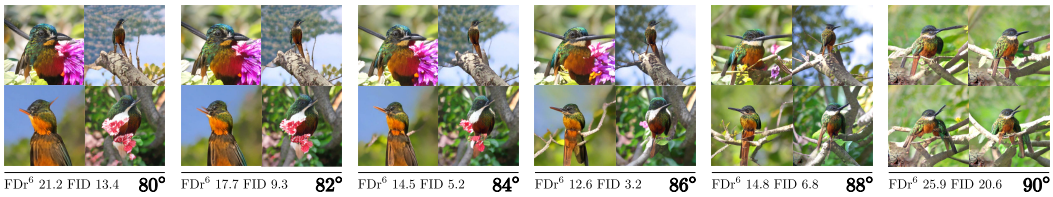

Figure 14: **Sampling angle** α with 1-step generation using Sphere2-B on ImageNet 256×256. Sampling angle decides the generation diversity by forcing the decoder away from the mean.

C.3 Sampling Angle

Since the decoder is conditioned on the angle α, we investigate sampling properties at different angles. We sweep α from $0°$ to $90°$ in 5-degree increments for one-step generation with Sphere2-B on ImageNet 256×256. Fig. 14 shows that the sampling angle controls the diversity of generated images. With $\alpha = 90°$, the generated images are almost the same, which is expected: the decoder sees only the pure noise $\mathcal{F}(\mathbf{e})$ (Eq. (4)) without any reference information from the input image during training. Thus, the decoder collapses to the mean of all plausible images in the class. With $\alpha < 90°$, the decoder sees a mixture of the clean latent and the noise during training, so it starts to generate images with more diversity. Fig. 15 quantifies the effect of sampling angle on ImageNet 256×256.

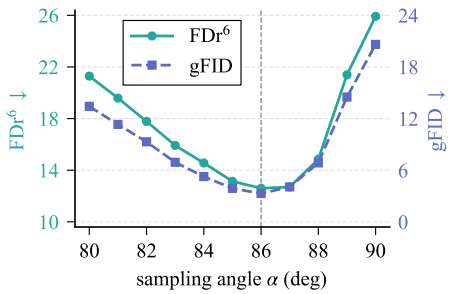

Figure 15: **Sampling angle** α with 1-step generation using Sphere2-B on ImageNet 256×256. FDr^6 and FID agree in a "U" shape along the angle, with the best quality at $86°$.

For multi-step generation, we also investigate the effect of sampling angle α on ImageNet 256×256. The alpha α is fixed for all intermediate steps in the loop of Alg. 1. We found the FDR^6 and FID metrics does not agree on the best α for multi-step generation. For simplicity, we choose $\alpha = 86°$ for Sphere-B and $\alpha = 84°$ for Sphere-L and Sphere-H to report the main results in Tab. 2, in order to prioritize the FDr^6 metric.

C.4 Spherical CFG

Classic CFG (Ho & Salimans, 2022) linearly extrapolates from the unconditional prediction to the conditional one, which would push a latent off the sphere. We instead guide on the sphere with a rotation. From the same noise $\mathcal{F}(\mathbf{e})$, we follow two parallel arcs of Alg. 1: a conditional arc with class $\mathbf{y}$ and an unconditional arc with the null class $\mathbf{y}_{\text{null}}$. Following common practice, the null class is learned during training by dropping the class embedding to $\mathbf{y}_{\text{null}}$ with probability 0.1. At each refinement step, we encode both generated images into latents $\mathbf{z}_c$ and $\mathbf{z}_u$, and rotate $\mathbf{z}_c$ away from $\mathbf{z}_u$ along their great circle by a fixed guidance angle γ:

$$\mathbf{z}_{\text{cfg}} = \frac{\sin(\theta + \gamma)}{\sin\theta}\,\mathbf{z}_c - \frac{\sin\gamma}{\sin\theta}\,\mathbf{z}_u, \qquad \theta = \angle(\mathbf{z}_u, \mathbf{z}_c). \tag{23}$$

The guided latent $\mathbf{z}_{\text{cfg}}$ stays on the sphere at angle $\theta + \gamma$ from $\mathbf{z}_u$, and replaces $\mathbf{z}_c$ in the rotation toward the noise in Eq. (4).[3] Unlike a guidance scale, γ fixes the step on the sphere regardless of how far apart the two arcs are, and $\gamma = 0$ recovers Alg. 1. The unconditional arc is never guided and continues from its own latent $\mathbf{z}_u$; if it were decoded from $\mathbf{z}_{\text{cfg}}$, the two arcs would merge after one step and leave nothing to guide against. Since guidance acts on the refinement steps, it requires $K \geq 2$, and one-step generation stays unguided. Alg. 2 summarizes the sampling procedure.

[3]When $\mathbf{z}_c$ and $\mathbf{z}_u$ are parallel or antiparallel ($\sin\theta = 0$), no rotation plane is defined, and we keep $\mathbf{z}_{\text{cfg}} = \mathbf{z}_c$. This never occurs in practice, since the two arcs decode different images in the high-dimensional latent space.

Algorithm 2 Sampling with spherical CFG. `rotate_away` is Eq. (23) and `gamma` is the guidance angle.

```
e = F(Normal(0, 1).sample([d]))  # random point

# one-step generation, two arcs from the same point
x   = D(e, y,      alpha)  # cond
x_u = D(e, y_null, alpha)  # uncond

# few-step refinement, guided on the sphere
for _ in range(K - 1):
    z_c, z_u = F(E(x)), F(E(x_u))
    z = rotate_away(z_c, z_u, gamma)

    x   = D(cos(alpha) * z   + sin(alpha) * e, y,      alpha)
    x_u = D(cos(alpha) * z_u + sin(alpha) * e, y_null, alpha)

return x
```

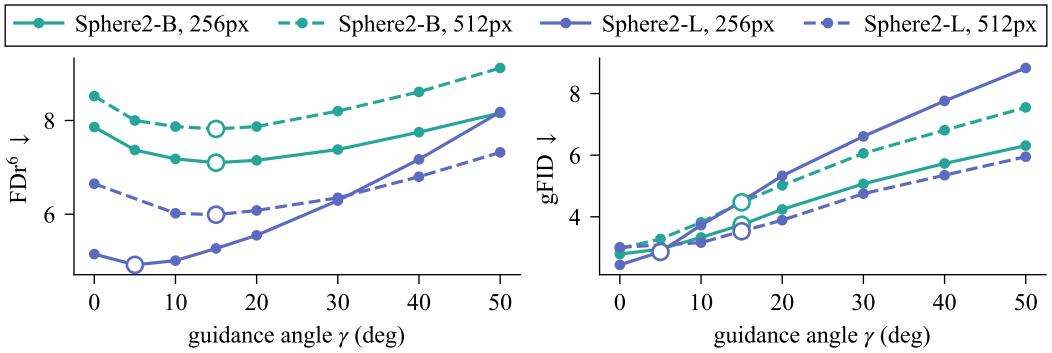

Figure 16: **Guidance angle** γ of spherical CFG at 4 sampling steps on ImageNet 256×256 and 512×512. $\gamma = 0°$ disables guidance and recovers Alg. 1. The hollow marker on each curve is the FDr^6-optimal γ, which the CFG rows of Tab. 2 use. Sampling angle α is $86°$ for Sphere2-B at 256 / 512 and $84°$ for Sphere2-L.

C.5 Frozen Latent Space

Tab. 4 ablates the choice of the frozen encoder $\mathcal{E}$. Since the encoder runs at every sampling step, we restrict the comparison to the small and base sizes of two model families, DINOv3 (Siméoni et al., 2025) and SigLIP2 (Tschannen et al., 2025). The small size has latent dimension $D = 384$ and the base size $D = 768$. We also test a lower-dimensional variant of each, adding a single linear layer that reduces D to 256 for small and 384 for base. The results show: (1) DINOv3 outperforms SigLIP2 at every size, and (2) DINOv3-small with its original $D = 384$ gives the best trade-off between quality and efficiency. We therefore use it by default.

Table 4: **Frozen latent space** of the encoder $\mathcal{E}$ with different pretrained encoders and latent dimensions on Oxford-Flowers 256×256.

Encoder	$d = N \times D$	Compression ratio	gFID ↓			FDr^6 ↓		
			1	2	4	1	2	4
DINOv3-Small	$16^2 \times 384$	2	**10.29**	**8.24**	**7.51**	**3.92**	**3.14**	**3.04**
DINOv3-Small	$16^2 \times 256$	3	17.67	13.64	12.27	4.53	3.77	3.61
DINOv3-Base	$16^2 \times 768$	1	11.47	9.28	8.69	4.28	3.58	3.51
DINOv3-Base	$16^2 \times 384$	2	18.18	15.67	15.78	5.17	4.48	4.46
SigLIP2-Base	$16^2 \times 768$	1	15.15	13.73	15.15	5.82	4.37	4.18
SigLIP2-Base	$16^2 \times 384$	2	16.32	13.15	13.07	4.91	4.25	4.24

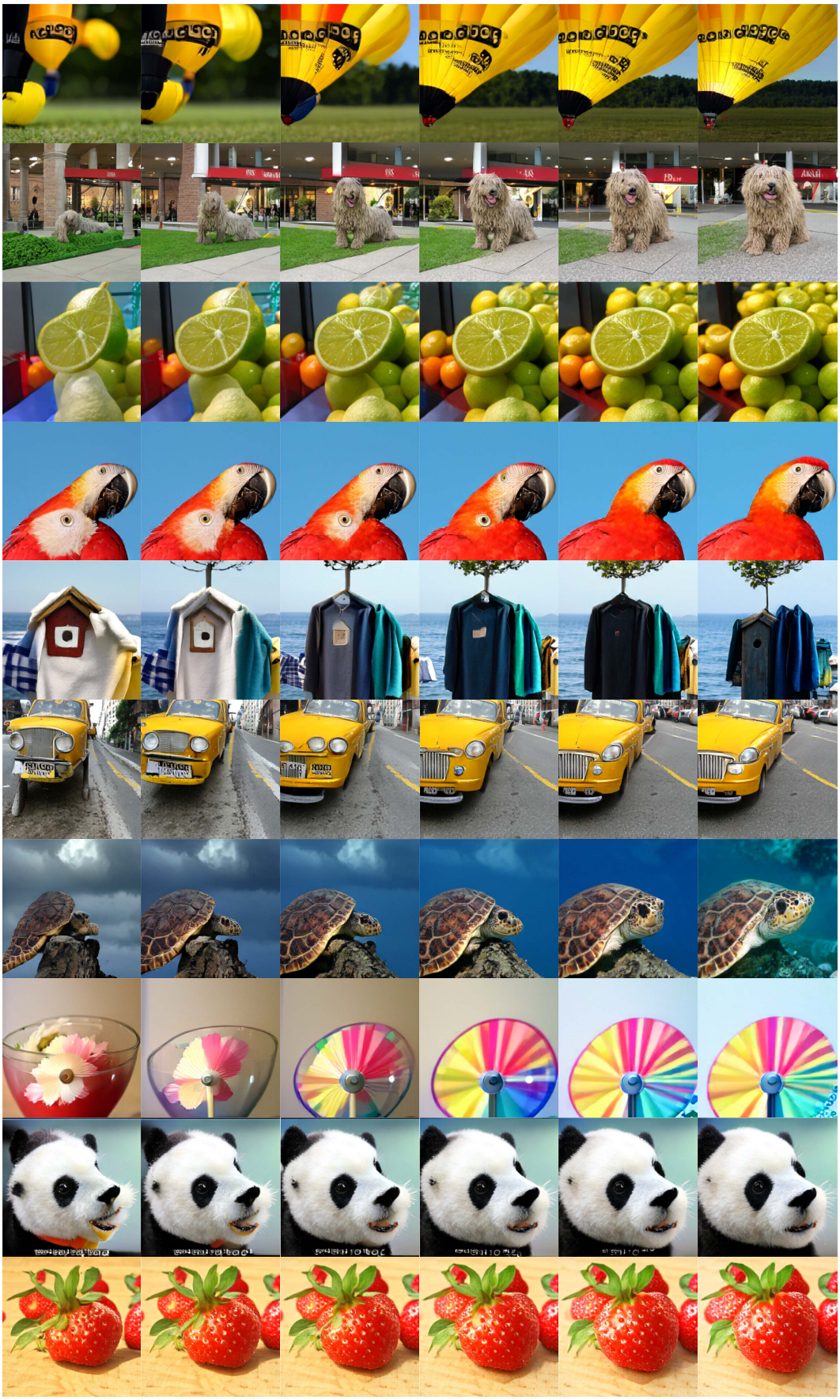

Figure 17: **Spherical CFG** for 2-step generation on ImageNet 256 × 256 with Sphere2-B.

C.6 Distribution Shift

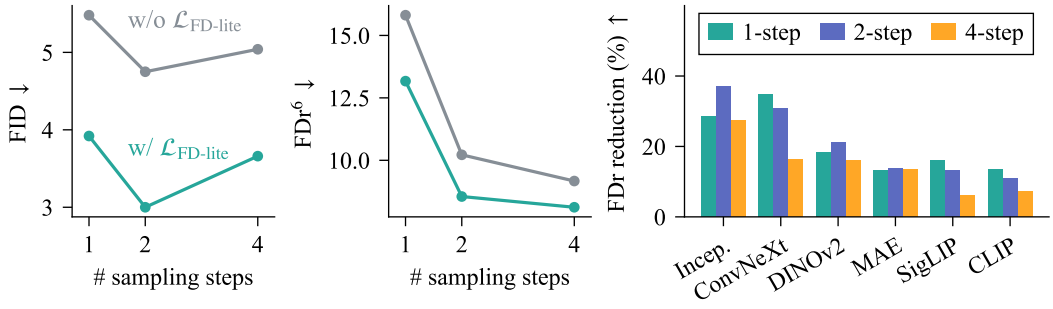

Figure 18: **Effect of** $\mathcal{L}_{\text{FD-lite}}$ on ImageNet 256 × 256. Left and middle: FID and FDr^6 with and without the loss at different sampling steps. Right: relative FDr reduction in each of the six feature spaces.

Sec. 3.4.4 shows that the statistics of generated images drift away from those of real images in feature spaces other than the one $\mathcal{L}_{\text{score}}$ matches, and introduces $\mathcal{L}_{\text{FD-lite}}$ to counter this drift. Fig. 18 measures its effect on ImageNet 256 × 256. Left and middle: $\mathcal{L}_{\text{FD-lite}}$ improves FDr^6 and FID at every sampling step. Right: the relative FDr reduction in each of the six feature spaces. The reduction is largest in the Inception, ConvNeXt, and DINOv2 spaces, which match or lie closest to the three extractors in $\mathcal{L}_{\text{FD-lite}}$, namely Inception, ConvNeXt V2-N, and DINOv3-small.

C.7 Score Feature Space

$\mathcal{L}_{\text{score}}$ needs a frozen feature space ϕ and a choice of layers in it. We compare two extractors: ConvNeXt V2-N, our default, and DINO-S/8, the classic choice for GAN losses (Zheng et al., 2026). For each, we sweep which layers feed the score models (Tab. 5). The combination of a middle layer and a deep layer gives the best enough capacity for both extractors. Adding more layers costs up to twice the parameters and helps a little.

Table 5: **Feature extractor** ϕ **and its layers used by the score loss** on Oxford Flowers 256 × 256. Sphere2-B is trained with 1000 epochs. Each score model is a two-layer transformer with width 384. Params counts total parameters of all score models. Params of DINO-S/8 is 21.67M, and ConvNeXt V2-N fcmae is 14.98M.

Feature ϕ	Layers	Params	FDr^6 ↓		gFID ↓	
			1	4	1	4
DINO-S/8	5 11	25.19M	**6.16**	**4.63**	**24.86**	**17.17**
	3 7 11	37.79M	5.95	4.49	27.23	17.68
	5 8 11	37.79M	6.20	4.72	24.28	17.80
	2 5 8 11	50.38M	6.48	5.12	27.20	20.28
ConvNeXt V2-N	1 2	24.16M	**6.51**	**4.88**	**26.80**	**18.60**
	1 3	44.79M	6.42	5.18	29.85	21.52

C.8 Score Sigma Distribution

$\mathcal{L}_{\text{score}}$ interleaves two procedures: training the score models and transporting the generator with them. The score models are standard EDM denoisers, trained on the noisy features of Eq. (10) with σ drawn from the EDM log-normal $p(\sigma)$, $(p_{\text{mean}}, p_{\text{std}}) = (-1.2, 1.2)$, without clamping. For the transport in Eq. (11), we draw σ from a second log-normal $q(\sigma)$ with $(q_{\text{mean}}, q_{\text{std}})$. Two considerations guide the choice of $q(\sigma)$. First, $q(\sigma)$ should stay within the bulk of $p(\sigma)$, so the score models are never queried at noise levels they were not trained on. Second, $q(\sigma)$ should lean toward small noise levels, where the score difference carries the fine detail that pushes the generator toward high-fidelity pixels.

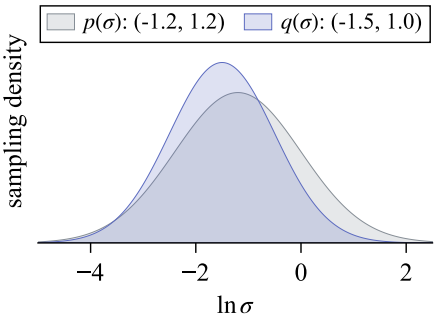

Figure 19: **Noise sampling distribution for score models** of $\mathcal{L}_{\text{score}}$ in training $p(\sigma)$ and transport $q(\sigma)$.

Tab. 6 sweeps $(q_{\text{mean}}, q_{\text{std}})$ with $p(\sigma)$ fixed. The best $q(\sigma)$ is $(-1.5, 1.0)$: shifted toward small σ and slightly narrower than $p(\sigma)$, as Fig. 19 shows. Pushing further hurts. At $(-1.8, 0.8)$, $q(\sigma)$ concentrates away from the mode of $p(\sigma)$, and FID nearly doubles in the ConvNeXt space. Both considerations are therefore borne out: the transport distribution should lean toward small noise, but not leave the region the score models know. Additionally, the ImageNet finetuned version of ConvNeXt V2-N ft-in1k provides a better feature space than the pretrained ConvNeXt V2-N fcmae, so we use it by default.

Table 6: **Sigma distribution for transport procedure** of the score loss. Sphere2-B is trained with 2000 epochs on Oxford Flowers 256×256.

Feat. ϕ	Transport $(q_{\text{mean}}, q_{\text{std}})$	FDr^6 ↓		gFID ↓	
		1	4	1	4
DINO-S/8	$(-1.2, 1.0)$	4.54	3.93	13.69	10.89
	$(-1.5, 1.0)$	4.57	3.94	13.35	10.54
	$(-1.8, 1.0)$	4.61	3.96	13.19	10.37
	$(-1.8, 0.8)$	4.97	4.28	14.46	11.02
ConvNeXt V2-N ft-in1k	$(-1.2, 1.0)$	3.82	3.09	11.40	8.28
	$(-1.5, 1.0)$	**3.92**	**3.04**	**10.29**	**7.51**
	$(-1.8, 1.0)$	4.81	3.64	13.50	9.40
	$(-1.8, 0.8)$	6.16	5.04	15.78	13.90
ConvNeXt V2-N fcmae	$(-1.2, 1.0)$	5.03	3.83	15.07	13.09
	$(-1.5, 1.0)$	4.92	3.96	14.00	12.58
	$(-1.8, 1.0)$	5.15	4.18	14.74	12.85
	$(-1.8, 0.8)$	5.48	4.21	14.37	13.22

D Implementation Details

Tab. 7 lists the architecture, training, and sampling configurations of Sphere2-B and Sphere2-L used for the ImageNet results in Tab. 2.

Table 7: **Configurations of experiments.** Values in a merged cell are shared by both models.

	Sphere2-B	**Sphere2-L**
architecture		
encoder $\mathcal{E}$	DINOv3-small (frozen, 29M)	
latent	$N = 16 \times 16$, $D = 384$, $d = 98{,}304$	
decoder $\mathcal{D}$	ViT-B	ViT-L
depth	12	24
hidden dim	768	1024
heads	12	16
image size	256 (other setting: 512)	
patch size	`image_size` / 16	
conditioning	class $\mathbf{y}$ and angle α via adaLN-Zero	
params (encoder + decoder)	161M	487M
GFLOPs (single forward)	22	115
training		
epochs	2000 (Oxford Flowers), 400 (ImageNet)	
warmup epochs	10 (Oxford Flowers), 5 (ImageNet)	
optimizer	AdamW	
batch size	256	
learning rate	1e-4	
learning rate schedule	constant	
weight decay	0	
ema decay	0.9998	
angle sampler	$\alpha \sim \mathcal{U}[0^\circ, 90^\circ]$	
cutoff angle α_{cutoff}, bandwidth β	70°, 5°	
score feature space ϕ	ConvNeXt V2-N ft-in1k (frozen, 16M), two middle layers	
score models $\psi_{\text{real}}, \psi_{\text{fake}}$	2-layer transformers, EDM preconditioning	
score σ: train $p(\sigma)$, transport $q(\sigma)$	log-normal $(-1.2, 1.2)$, $(-1.5, 1.0)$; clip $[0.002, 8]$	
transport balance	EMA 0.99, $w_j \in [0.05, 20]$	
differentiable augmentation	flip $p = 0.5$, translate $p = 1.0$, ratio $1/64$	
FD-lite extractors	Inception, DINOv3-small, ConvNeXt V2-N	
class token drop (for CFG)	0.1	
loss weights		
$\mathcal{L}_{\text{pix-recon}}$	1.0	
$\mathcal{L}_{\text{sem}}$	1.0	
$\mathcal{L}_{\text{score}}$	1.0	
$\mathcal{L}_{\text{lat-con}}$	0.1	
$\mathcal{L}_{\text{FD-lite}}$	0.1	
sampling		
sampling angle α (256 / 512)	86° / 86°	84° / 84°
guidance angle γ (256 / 512)	15° / 15°	5° / 15°