HYPERSPHERICAL AUTOENCODER FOR IMAGE RECONSTRUCTION AND GENERATION

Anonymous authors
Paper under double-blind review

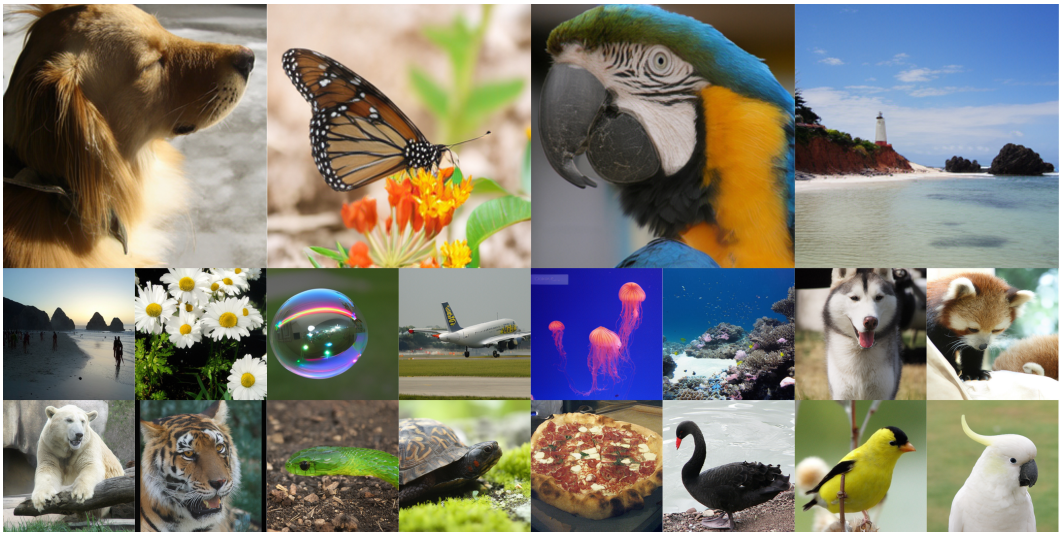

Figure 1: Representative reconstruction and generation samples from HAE.

ABSTRACT

Recent studies have explored using pretrained Vision Foundation Models (VFMs) such as DINO for generative autoencoders, showing strong generative performance. Existing VFM-based approaches often exhibit limited reconstruction fidelity due to the loss of high-frequency details. In this work, we present the ***Hyperspherical Autoencoder (HAE)***, a framework that bridges semantic representation and pixel-level reconstruction. Our key insight is that hierarchical patch embedding preserves reconstruction-relevant local detail, while directional alignment retains the pretrained teacher's semantic representation. To this end, we combine a *Hierarchical Convolutional Patch Embedding* module with *Directional Feature Alignment*, which aligns student and teacher feature directions without explicitly matching their magnitudes. Motivated by the observed reconstruction robustness to latent rescaling over the evaluated range, we normalize patch features and train a Diffusion Transformer (DiT) using *Riemannian Flow Matching* on the resulting patch-wise hyperspherical manifold. Using HAE latents, the DiT achieves a gFID of **1.79** without guidance and **1.63** with auto-guidance after 1,000 epochs. The autoencoder used for generation yields an rFID of **0.78** and a PSNR of **25.2** dB.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Song et al., 2021; Peebles & Xie, 2023) have advanced image generation. Latent diffusion models (LDMs) (Rombach et al., 2022) operate in the latent space of a pretrained autoencoder, making the representation an important component of generative modeling. Recent work has shown that aligning diffusion features with pretrained Vision Foundation Models (VFMs) can improve convergence. Related approaches, such as RAE (Zheng et al., 2025), use pretrained VFMs including DINOv2 (Oquab et al., 2023) as encoders to incorporate semantic

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.

representations into the latent space. However, semantic representation quality does not necessarily translate into accurate pixel-level reconstruction: the RAE baselines in Table 1 have lower PSNR than the conventional VAE baselines.

We investigate two design choices for adapting a pretrained VFM to reconstruction while keeping its Transformer backbone frozen. The first concerns single-layer, non-overlapping patchification at the input of the Vision Transformer (ViT), which projects the image directly onto the final token grid. The second concerns the objective used to retain teacher features after modifying this embedding. Direct mean squared error (MSE) penalizes discrepancies in both feature magnitude and direction, coupling teacher-feature alignment to feature-norm matching during reconstruction training.

To address these design questions, we introduce the *Hyperspherical Autoencoder (HAE)*, a framework for improving reconstruction while retaining pretrained semantic information. HAE replaces the original patch embedding with *Hierarchical Convolutional Patch Embedding (HCPE)*, which processes the image at progressively reduced spatial resolutions while the DINOv3 Transformer remains frozen. During joint training of HCPE and the decoder, *Directional Feature Alignment* uses cosine similarity to align student and teacher feature directions without explicitly matching their magnitudes.

We further observe that HAE reconstructions are relatively robust to latent rescaling over the evaluated range. This motivates modeling normalized patch features on a product of hyperspheres using **Riemannian Flow Matching (RFM)** (Chen & Lipman, 2023). Euclidean interpolation between normalized endpoints introduces radial conditional velocity targets, whereas spherical geodesics do not. Under low radial decoder sensitivity, removing these targets provides a motivation for learning tangent dynamics. Proposition 1 in Section 3.1 quantifies the radial target energy. Proposition 2 in Appendix C provides a complementary local bound based on decoder sensitivity. Our contributions can be summarized as follows:

- We propose **Hierarchical Convolutional Patch Embedding** for progressive spatial processing before a frozen ViT backbone. Reconstruction ablations, including parameter-matched controls, support its benefit over the evaluated non-hierarchical stems.
- We introduce **Directional Feature Alignment** to align the adapted encoder with the frozen teacher without requiring feature-norm matching. Together with HCPE, this supports reconstruction learning while retaining semantic information, as assessed by linear probing.
- We observe that HAE reconstructions are relatively robust to latent rescaling over the evaluated range. Motivated by this observation and a local decoder-sensitivity analysis, we formulate the generative process using **Riemannian Flow Matching** on normalized patch features. This manifold-aware approach aligns the generative dynamics with the spherical latent geometry, with empirical results suggesting benefits for convergence and generation quality.

2 Related Work

Representation Alignment. Representation alignment has been used to improve the convergence of diffusion-based generative models. REPA (Yu et al., 2025) aligns intermediate DiT features with pretrained DINOv2 representations. REG (Wu et al., 2025) introduces a class token into DiT and aligns it with the DINOv2 class token. Other approaches use internal feature constraints: SRA (Jiang et al., 2026) aligns intermediate features at higher noise levels with later-layer features at lower noise levels, while LSEP (Yun et al., 2025) promotes linear separability of intermediate features through a classification probe. REPA-E (Leng et al., 2025) extends representation alignment to the latent autoencoder by jointly optimizing VAE parameters under the REPA objective.

Semantic Latent Autoencoders. Semantic latent autoencoders incorporate pretrained representations into the autoencoding stage. VA-VAE (Yao et al., 2025) and MAETok (Chen et al., 2025) use pretrained VFM features to guide latent learning, with reported gains in downstream generation. RAE (Zheng et al., 2025) instead uses a pretrained VFM directly as the encoder, avoiding an additional encoder-alignment objective. Semantic representation quality and pixel-level reconstruction, however, are distinct objectives. HAE focuses on improving reconstruction through localized patch-embedding adaptation while keeping the pretrained Transformer frozen and aligning the adapted

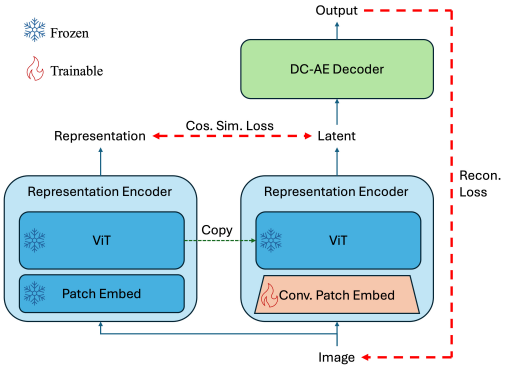

Figure 2: HAE architecture and training loss overview

Table 1: Comparison of autoencoder models on ImageNet 256×256. HAE-R/G denote reconstruction/generation settings. SSIM and LPIPS-VGG are reported for the same-backbone comparison.

Model	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SD-VAE	0.62	26.04	–	–
RAE (reported)	0.59	18.94	–	–
RAE (DINOv3-L)	0.6725	18.9	0.4685	0.2661
MAETok	0.48	23.61	–	–
VA-VAE	0.28	27.96	–	–
HAE-R (Ours)	0.37	26.20	0.7539	0.1353
HAE-G (Ours)	0.78	25.20	0.7408	0.1500

Table 2: Effect of latent smoothing.

Setting	rFID	gFID	PSNR
w/o Latent Smoothing	0.37	32.64	26.2
w/ Latent Smoothing	0.78	2.65	25.2

encoder with teacher features. PS-VAE (Zhang et al., 2025) learns a compact KL latent through encoder fine-tuning; HAE retains native-dimensional features through HCPE adaptation.

Riemannian Flow Matching. Chen and Lipman Chen & Lipman (2023) extended the idea of flow matching (FM) (Lipman et al., 2022) beyond Euclidean spaces and proposed *Riemannian Flow Matching* (RFM), enabling Continuous Normalizing Flow (CNF) training on general Riemannian manifolds $\mathcal{M}$. In this setting, the model learns a time-dependent tangent vector field $\mathbf{v}_\theta(\mathbf{x}, t) \in T_{\mathbf{x}}\mathcal{M}$ that transports samples from a base distribution p to the target data distribution q.

The general RFM objective and derivation are provided in Appendix C.5.

For image generation, Kumar and Patel (Kumar & Patel, 2026) introduce Riemannian Flow Matching with Jacobi Regularization (RJF), combining spherical geodesic paths with Jacobi-inspired loss weighting for pretrained representation latents. RJF and HAE both use spherical paths for latent generative modeling. Whereas RJF starts from an existing representation space, HAE learns reconstruction-capable latents through HCPE and directional feature alignment with a frozen Transformer backbone, then applies RFM without Jacobi regularization.

3 Hyperspherical Autoencoder (HAE)

3.1 Motivation: From Foundation Representations to Generative Latents

Pretrained vision foundation models provide rich semantic representations, but their feature spaces are not designed for image reconstruction or generative modeling. In particular, modern contrastive and self-supervised representations often emphasize feature direction rather than magnitude. For example, SimCLR optimizes cosine similarity between ℓ_2-normalized embeddings (Chen et al., 2020), and Wang & Isola (2020) show that normalized contrastive learning promotes alignment of positive pairs together with uniformity on the hypersphere. This suggests that much of the discriminative information learned by foundation models is naturally organized in angular geometry.

Directly using such features as generative latents, however, presents two challenges. First, foundation-model representations are optimized for semantic invariance rather than reconstruction, and may discard local image details needed by a decoder. Second, if the informative degrees of freedom are predominantly directional, treating the latent space as unconstrained Euclidean space introduces radial variations that need not correspond to meaningful changes in image space. A generative representation should therefore preserve the semantic geometry of the pretrained model while recovering the spatial information required for faithful reconstruction.

These considerations motivate our *Hyperspherical Autoencoder* (HAE). HAE augments a frozen pretrained Transformer with a trainable hierarchical patch-embedding stem and decoder, while aligning the learned latent features with the directional representations of the foundation model. The resulting representation is both reconstructive and naturally compatible with patch-wise normalization onto a hypersphere. In this way, HAE converts a discriminative foundation representation into a generative latent space without discarding the geometric structure inherited from pretraining.

We empirically examine this geometry by perturbing the magnitude and direction of the encoded features. For each spatial location (i, j), varying the feature norm over a broad range,

$$\tilde{\mathbf{z}}_{ij}^{(s)} = s\,\frac{\mathbf{z}_{ij}}{\|\mathbf{z}_{ij}\|_2}, \qquad s \in \{8, 10, 16, 20\},$$

produces visually similar reconstructions (Fig. 5(c–f)). In contrast, perturbing the normalized feature direction and restoring its original magnitude,

$$\tilde{\mathbf{z}}_{ij}^{\mathrm{dir}} = \|\mathbf{z}_{ij}\|_2 \frac{\bar{\mathbf{z}}_{ij} + \boldsymbol{\epsilon}_{ij}}{\|\bar{\mathbf{z}}_{ij} + \boldsymbol{\epsilon}_{ij}\|_2}, \qquad \boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\mathbf{0}, g^2\mathbf{I}),$$

substantially changes the reconstructed appearance and structure (Fig. 5(g)). These observations indicate that the decoder is considerably more sensitive to angular than radial perturbations, supporting a direction-dominant latent representation.

This geometry also has a direct consequence for generative modeling. If normalized latent features lie on a fixed-radius hypersphere, standard Euclidean flow matching is geometrically mismatched to the representation: linear interpolation between two points on the sphere leaves the manifold and introduces radial velocity components. Such radial motion consumes modeling capacity even though the decoder is relatively insensitive to it. Indeed, for equal-norm endpoints, the following proposition shows that Euclidean interpolation can allocate a non-negligible fraction of its target velocity to the radial direction. The proof can be found in Appendix C.

Proposition 1 (Uniform-time radial target-energy fraction) *Let* $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$*, with* $\|\mathbf{x}_0\|_2 = \|\mathbf{x}_1\|_2 = R > 0$ *and endpoint angle* $\omega \in (0, \pi)$*. For* $\mathbf{U} = \mathbf{x}_1 - \mathbf{x}_0$*, let* $\mathbf{U}_{\mathrm{rad}}(t)$ *be its orthogonal projection onto the direction of* $\mathbf{x}_t$*. Then*

$$\rho(\omega) := \frac{\int_0^1 \|\mathbf{U}_{\mathrm{rad}}(t)\|_2^2\, dt}{\int_0^1 \|\mathbf{U}\|_2^2\, dt} = 1 - \frac{\omega}{2}\cot\Big(\frac{\omega}{2}\Big). \tag{1}$$

For example, for nearly orthogonal endpoints this fraction is

$$\rho(\pi/2) = 1 - \pi/4 \approx 21.5\%.$$

By contrast, a spherical geodesic remains on the hypersphere and has purely tangential velocity. This motivates a unified design in which the representation and the generative dynamics share the same geometry: **HAE constructs a reconstructive, direction-dominant hyperspherical latent space, and Riemannian Flow Matching (RFM) models generation directly on that manifold.** The former preserves the useful directional structure of foundation-model features while recovering image detail, whereas the latter avoids learning radial dynamics that are weakly expressed by the decoder.

We next describe the HAE architecture and its training objective, followed by the Riemannian flow formulation used for generation. Figure 2 summarizes the overall architecture and training pipeline.

3.2 Architecture

Hierarchical Convolutional Patch Embedding. DINOv3 (Siméoni et al., 2025) uses a single-layer patch embedding that projects the image directly onto the final token grid. We investigate whether progressive spatial processing improves reconstruction when the subsequent Transformer remains frozen. HCPE replaces this projection with four convolutional stages that progressively reduce spatial resolution, producing tokens with the spatial and channel dimensions required by the pretrained Transformer. All Transformer parameters remain frozen, while the input stem is trained jointly with the decoder and directional teacher–student alignment.

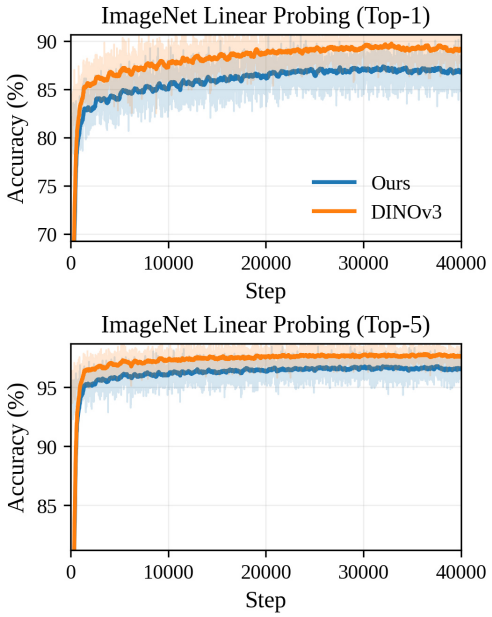

Figure 3: Linear probing results on ImageNet-1K. HAE retains robust semantic information throughout training.

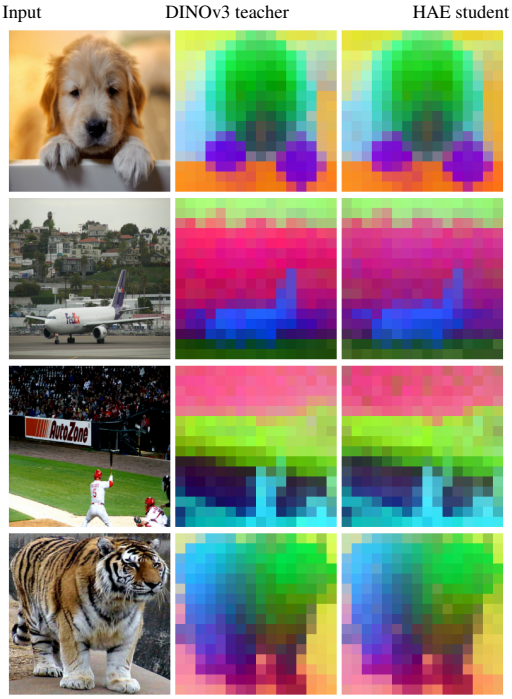

Figure 4: PCA visualizations of teacher and student features. Columns show the input image, frozen DINOv3 teacher, and HAE student, respectively; each row corresponds to the same image. Both feature maps are projected onto the first three components of the teacher-derived PCA basis and visualized using the same RGB normalization.

Decoder Architecture. We adopt a decoder based on DC-AE (Chen et al., 2024) to map latent tokens $\mathbf{z}$ back to the image $\hat{\mathbf{x}}$. Its progressive upsampling and residual shortcuts provide a multi-resolution reconstruction pathway from the spatially compressed patch representation. Detailed configurations are provided in Appendix D.

3.3 Training

Directional Feature Alignment. Replacing the patch embedding changes the inputs to the frozen Transformer, motivating alignment with the original teacher representation. Direct MSE penalizes differences in both feature magnitude and direction. We instead use cosine similarity, $\mathcal{L}_{\text{align}} = 1 - (\mathbf{z}_S \cdot \mathbf{z}_T)/(\|\mathbf{z}_S\|_2\|\mathbf{z}_T\|_2)$, where $\mathbf{z}_S$ and $\mathbf{z}_T$ denote student and teacher features. This objective aligns feature directions without explicitly matching their magnitudes during reconstruction training.

We assess feature-level semantic retention using linear probing on ImageNet-1K (Figure 3). HAE achieves 87% Top-1 and 97% Top-5 accuracy, compared with 89% and 98% for the pretrained DINOv3 teacher. Figure 4 complements this evaluation with a qualitative comparison of teacher and student spatial feature patterns under a shared PCA projection.

Progressive Training Strategy. We train our autoencoder in two progressive stages to balance semantic alignment and reconstruction fidelity.

Stage 1: Semantic-Structural Alignment. In the first stage, we jointly train the hierarchical convolutional patch embedding (HCPE) and the decoder while keeping the pretrained DINOv3 Transformer and the teacher encoder frozen. The training objective combines directional teacher–student alignment, pixel-wise reconstruction, and perceptual losses:

$$\mathcal{L}_{\text{Stage1}} = \lambda_{\cos}\mathcal{L}_{\text{align}} + \lambda_{\text{L1}}\|\mathbf{x} - \hat{\mathbf{x}}\|_1 + \lambda_{\text{lpips}}\mathcal{L}_{\text{LPIPS}}(\mathbf{x}, \hat{\mathbf{x}}), \tag{2}$$

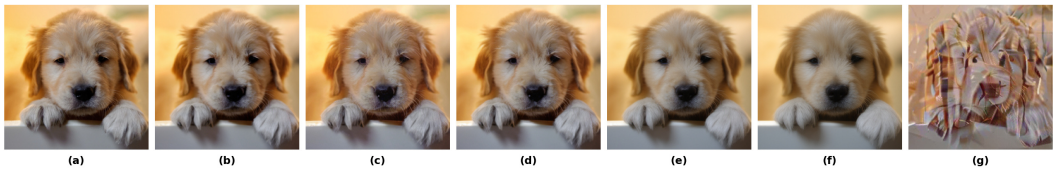

Figure 5: **Latent rescaling and directional perturbation.** (a) Ground truth. (b) Original reconstruction. (c–f) Reconstructions after setting each patch-feature norm to $s = 8, 10, 16$, and 20, respectively. (g) Reconstruction after adding Gaussian noise with $g = 0.1$ to normalized features, followed by renormalization to their original magnitudes. The examples compare the decoded outputs under these magnitude and directional modifications.

where $\mathcal{L}_{\text{LPIPS}}$ denotes the LPIPS perceptual loss between the input image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$. The decoder is trained on unperturbed encoder outputs in this stage; stochastic latent smoothing is introduced in Stage 2.

Stage 2: Adversarial Adaptation with Stochastic Latent Smoothing. We freeze the entire encoder, including HCPE, and train only the decoder and discriminator. To improve decoding robustness to perturbed latent inputs, we combine adversarial training with stochastic latent smoothing.

Following RAE (Zheng et al., 2025), we perturb the encoder outputs as $\mathbf{z}' = \mathbf{z} + \sigma\boldsymbol{\epsilon}$, where $\sigma \sim \mathcal{U}(0, \tau)$ is sampled per batch and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This broadens the decoder's training inputs around the fixed encoder outputs, encouraging robustness to latent perturbations without changing the encoder representation.

Table 2 shows a reconstruction–generation trade-off. Without latent smoothing, HAE achieves an rFID of 0.37 and a PSNR of 26.2 dB, with a downstream gFID of 32.64. With latent smoothing, the reconstruction metrics change to an rFID of 0.78 and a PSNR of 25.2 dB, while gFID improves to 2.65. We therefore distinguish the reconstruction-oriented setting (HAE-R) from the generation-oriented setting (HAE-G), which is used for our main generation results.

We combine reconstruction and perceptual losses with an adversarial objective (Sauer et al., 2023; Zheng et al., 2025), giving the Stage-2 loss:

$$\mathcal{L}_{\text{Stage2}} = \mathcal{L}_{\text{Stage1}}\big|_{\lambda_{\cos}=0} + \lambda_{\text{adv}}\mathcal{L}_{\text{GAN}}(\mathbf{x}, \hat{\mathbf{x}}) \tag{3}$$

Detailed training configurations, including the specific discriminator architecture, are provided in Appendix D.

3.4 Riemannian Flow Matching

Motivated by the direction-dominant geometry of HAE latents, we perform flow matching directly on the hypersphere, avoiding the unnecessary radial motion induced by Euclidean interpolation.

Formally, we model the normalized latent space as a product manifold of N hyperspheres:

$$\mathcal{M} = (S_R^{C-1})^N, \qquad S_R^{C-1} = \left\{ \mathbf{z} \in \mathbb{R}^C \;\middle|\; \|\mathbf{z}\|_2 = R \right\}. \tag{4}$$

Here, N is the number of patches and C is the feature dimension. For flow training, encoder outputs and Gaussian-noise endpoints are normalized patch-wise to radius R. We denote the data endpoint by $\mathbf{x}_0$ and the noise endpoint by $\mathbf{x}_1$, with generation proceeding from $t = 1$ to $t = 0$. For each patch i, we use the following spherical geodesic as the conditional interpolation path:

$$\mathbf{x}_t^{(i)} = \frac{\sin\big((1-t)\Omega^{(i)}\big)}{\sin\big(\Omega^{(i)}\big)}\,\mathbf{x}_0^{(i)} + \frac{\sin\big(t\Omega^{(i)}\big)}{\sin\big(\Omega^{(i)}\big)}\,\mathbf{x}_1^{(i)}, \tag{5}$$

where the angular distance $\Omega^{(i)}$ is defined as

$$\Omega^{(i)} = \arccos\left(\frac{\langle \mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)} \rangle}{R^2}\right). \tag{6}$$

The target velocity $\mathbf{u}_t^{(i)}$ is obtained by differentiating the geodesic with respect to time:

$$\mathbf{u}_t^{(i)} = \frac{d}{dt}\,\mathbf{x}_t^{(i)} = \frac{\Omega^{(i)}}{\sin\big(\Omega^{(i)}\big)}\left(\cos\big(t\Omega^{(i)}\big)\,\mathbf{x}_1^{(i)} - \cos\big((1-t)\Omega^{(i)}\big)\,\mathbf{x}_0^{(i)}\right). \tag{7}$$

Table 3: **Comparison of generative models on ImageNet** 256×256**.** We report gFID, IS, Precision, and Recall with and without guidance. HAE uses CFG at 80 epochs and auto-guidance (AG) at 1,000 epochs. Parameter counts refer to generators, excluding autoencoders. Bold indicates the best value in each metric column across the listed configurations, which differ in training budget and generator size.

Method	Epochs	#Params	Generation@256 w/o guidance				Generation@256 w/ guidance			
			gFID↓	IS↑	Prec.↑	Rec.↑	gFID↓	IS↑	Prec.↑	Rec.↑
Autoregressive										
VAR (Tian et al., 2024)	350	2B	1.92	**323.10**	**0.82**	0.59	1.73	**350.20**	0.82	0.60
MAR (Li et al., 2024)	800	943M	2.35	227.80	0.79	0.62	1.55	303.70	0.81	0.62
Euclidean Latent Diffusion & Flow Matching										
DiT-XL/2 (Peebles & Xie, 2023)	1400	675M	9.62	121.50	0.67	0.67	2.27	278.20	**0.83**	0.57
SiT-XL/2 (Ma et al., 2024)	1400	675M	8.61	131.70	0.68	0.67	2.06	270.30	0.82	0.59
REPA (Yu et al., 2025)	80	675M	7.90	122.60	0.70	0.65	–	–	–	–
	800	675M	5.78	158.30	0.70	0.68	–	–	–	–
DDT (Wang et al., 2025)	80	675M	6.62	135.20	0.69	0.67	1.52	263.70	0.78	0.63
VA-VAE (Yao et al., 2025)	80	675M	4.29	–	–	–	–	–	–	–
	800	675M	2.17	205.60	0.77	0.65	–	–	–	–
MAETok (Chen et al., 2025)	800	675M	2.56	224.50	–	–	1.72	307.30	–	–
REPA-E (Leng et al., 2025)	80	675M	3.46	159.80	0.77	0.63	–	–	–	–
	800	675M	1.70	217.30	0.77	0.66	1.15	304.0	0.79	0.66
RAE (Zheng et al., 2025)										
+LightningDiT-XL/1 (DINOv2-B)	80	676M	4.28	–	–	–	–	–	–	–
+LightningDiT-XL/1 (DINOv2-S)	800		1.87	209.70	0.80	0.63	1.41	309.40	0.80	0.63
+DiT^{DH}-XL/1 (DINOv2-B)	80	839M	2.16	214.80	**0.82**	0.59	–	–	–	–
	800		**1.51**	242.90	0.79	0.63	**1.13**	262.60	0.78	0.67
Riemannian Latent Flow Matching										
RAE (DINOv2-B)+LightningDiT-XL/1+RJF (Kumar & Patel, 2026)	80	677M	3.62	186.20	**0.82**	0.52	2.81	201.22	0.82	0.56
HAE (DINOv3-L)+LightningDiT-XL/1 (Ours)	80	677.5M	2.65	202.10	**0.82**	0.55	2.37	217.19	0.80	0.60
	1000		1.79	243.00	0.72	**0.70**	1.63	247.20	0.72	**0.71**

The conditional target velocity $\mathbf{u}_t^{(i)}$ lies in the tangent space, whereas the network produces an unprojected vector $\mathbf{v}_\theta^{(i)}$ in $\mathbb{R}^C$. We regress its tangent component toward the target and regularize its radial component:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\,\mathbf{x}_0,\,\mathbf{x}_1}\left[\frac{1}{N}\sum_{i=1}^{N}\left(\left\|\mathbf{v}_{\text{tan}}^{(i)} - \mathbf{u}_t^{(i)}\right\|_2^2 + \lambda_{\text{rad}}\left\|\mathbf{v}_{\text{rad}}^{(i)}\right\|_2^2\right)\right], \tag{8}$$

where the expectation is over flow time and paired endpoints. With $\bar{\mathbf{x}}_t^{(i)} = \mathbf{x}_t^{(i)}/R$, the components are $\mathbf{v}_{\text{rad}}^{(i)} = \langle \mathbf{v}_\theta^{(i)}, \bar{\mathbf{x}}_t^{(i)} \rangle \bar{\mathbf{x}}_t^{(i)}$ and $\mathbf{v}_{\text{tan}}^{(i)} = \mathbf{v}_\theta^{(i)} - \mathbf{v}_{\text{rad}}^{(i)}$. The coefficient λ_{rad} controls regularization of the network's unprojected radial output. We use $\lambda_{\text{rad}} = 0.1$, as evaluated in Table 8.

During sampling, tangent projection and exponential-map updates maintain the fixed-radius constraint. Detailed training and inference algorithms are provided in Appendix B.

4 Experiments

4.1 Image Generation via Riemannian Flow Matching

Implementation. We train LightningDiT-XL with QK-Norm and BF16 precision on four NVIDIA B200 GPUs for up to 1,000 epochs. We use AdamW with a learning rate of 2×10^{-4}, betas of $(0.9, 0.95)$, and an effective global batch size of 1024, together with an exponential moving average (EMA) decay of 0.9995. The EMA-decay study and numerical-configuration comparison are provided in Appendices A.6 and A.5, respectively.

Method	gFID ↓
Euclidean FM	15.95
Riemannian FM	**2.65**

Table 4: Ablation on generative dynamics: Euclidean Flow Matching (FM) vs. Riemannian Flow Matching (RFM).

Patch Embedding Strategy	**rFID ↓**	**PSNR ↑**
Standard ViT (Baseline)	0.47	19.55
HCPE (Ours)	**0.37**	**26.20**

Table 5: Comparison of patch-embedding strategies without latent smoothing. HCPE improves PSNR and rFID relative to the standard-patch baseline.

Results. Table 3 compares generation performance on ImageNet at 256×256 resolution. At 80 epochs, HAE achieves lower unguided gFID than the reported VA-VAE and RAE baselines using similarly sized standard generators. These are system-level comparisons: the epoch counts are matched, but the latent representations and training formulations differ. The table also compares HAE with RJF under unguided and CFG sampling. After 1,000 epochs, HAE achieves a gFID of 1.79 without guidance and 1.63 with auto-guidance.

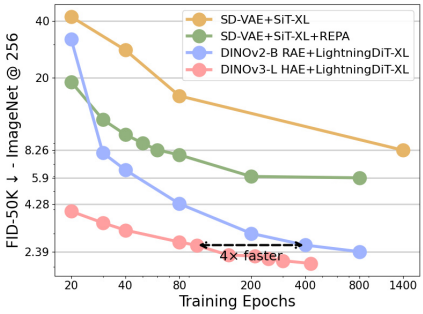

Figure 6: gFID training curves for the compared autoencoder–generator systems.

HAE obtains competitive generation results with standard LightningDiT-XL, without an additional wide generator head. The larger RAE + DiT^{DH}-XL system achieves lower gFID in the reported comparisons, while our results demonstrate the effectiveness of the HAE representation and flow formulation with a standard-capacity generator.

Figure 6 compares the gFID training curves of HAE and the baseline autoencoder–generator systems. The effect of the flow formulation within the same HAE latent space is examined separately in Section 4.2.

4.2 Ablation Studies

Same-backbone reconstruction. With the same frozen DINOv3-L backbone, HAE-R improves on reproduced RAE from 18.9 to 26.2 dB PSNR and from 0.6725 to 0.37 rFID (Table 1), with higher SSIM and lower LPIPS-VGG. We reproduce RAE (Zheng et al., 2025) with its standard DINOv3-L patch embedding, original decoder and training recipe, adapting the decoder input interface to the 1,024-dimensional features. RAE and HAE share the dataset, 256×256 reconstruction resolution, and evaluation protocol; their decoders and training procedures remain method-specific. HAE-G also improves PSNR, SSIM, and LPIPS-VGG over the reproduced RAE, with a higher rFID, reflecting the reconstruction–generation trade-off of latent smoothing. This comparison holds the pretrained backbone fixed; the stem-specific controls are reported in Appendix A.3.

Effect of Hierarchical Convolutional Patch Embedding. We compare HCPE with standard single-layer ViT patchification using the same frozen DINOv3 Transformer. As shown in Table 5, HCPE increases PSNR from 19.55 to 26.20 dB and improves rFID. In a separate 10-epoch comparison with approximately 2.74M stem parameters, HCPE achieves 21.73 dB PSNR, compared with 21.11 and 21.46 dB for two parameter-matched controls. The parameter-matched HCPE controls are detailed in Appendix A.3.

Direct MSE versus directional alignment. Directional alignment retains higher linear-probing accuracy than direct MSE under joint reconstruction training (Table 6). We train otherwise identical models for 5,000 updates, changing only the teacher–student alignment objective, and evaluate both with the same ImageNet linear-probing protocol. Directional alignment achieves 84.0% Top-1 and 97.0% Top-5 accuracy, compared with 6.6% and 17.0% for MSE, supporting feature-level semantic preservation with the directional objective.

Sensitivity to the directional-alignment weight. We vary only $\lambda_{\cos}$, keeping the reconstruction and LPIPS weights, architectures, initialization, data order, optimizer, and evaluation protocol fixed.

Each variant is trained for five epochs of the 20-epoch Stage-1 schedule. Table 7 shows that the default $\lambda_{\text{cos}} = 0.5$ gives the highest PSNR and lowest LPIPS among the evaluated settings. Increasing the weight raises teacher–student cosine similarity, while Top-1 accuracy remains between 84.914% and 85.682%. The largest weight gives the lowest rFID, showing the trade-offs among reconstruction and feature-alignment metrics during early training.

Table 6: Teacher–student alignment objectives compared after 5,000 updates. Linear-probing accuracies are reported in percent; Figure 3 reports the full-training results.

Alignment objective	Top-1 (%) ↑	Top-5 (%) ↑
Direct MSE	6.6	17.0
Directional alignment	**84.0**	**97.0**

Table 7: Directional-alignment weight sensitivity after five epochs. All settings share the same reconstruction and perceptual-loss weights. Linear-probing Top-1 accuracy is reported in percent.

λ_{cos}	PSNR ↑	LPIPS ↓	rFID ↓	Teacher cosine ↑	Top-1 (%) ↑
0.5 (default)	**20.0232**	**0.293040**	7.26691	0.910592	84.914
1.0	20.0138	0.297771	7.28173	0.930950	**85.682**
2.0	19.6423	0.304846	**7.04985**	**0.944641**	85.652

Impact of the Radial Penalty. Table 8 evaluates the radial-penalty weight after 20 epochs. Among the tested settings, $\lambda_{\text{rad}} = 0.1$ yields the lowest gFID, supporting its use as the default. This term regularizes the network's unprojected radial output, while tangent projection and manifold-aware integration maintain the spherical constraint during sampling.

λ_{rad}	gFID ↓
0.0	4.40
1.0	5.70
0.1	**3.90**

Table 8: Ablation on radial penalty. Models in this study are trained for 20 epochs.

Riemannian vs. Euclidean Flow Matching. Table 4 compares Euclidean FM and RFM using the same HAE latent space and LightningDiT-XL architecture after 80 epochs. RFM yields lower gFID, supporting the spherical formulation in the evaluated HAE setting. Section 3.4 and Appendix C provide the geometric and decoder-sensitivity motivation for this design. Appendix A.4 separately reports operation counts, measured latency, and memory usage, together with the conditions of each computational-cost benchmark.

5 Conclusion

We presented the Hyperspherical Autoencoder (HAE), a framework for adapting pretrained vision representations to image reconstruction and generation. HAE replaces single-layer patch embedding with Hierarchical Convolutional Patch Embedding to improve reconstruction while keeping the DINOv3 Transformer frozen. Directional Feature Alignment aligns student and teacher feature directions without requiring magnitude matching, supporting reconstruction learning while retaining feature-level semantic information.

Motivated by the observed reconstruction robustness to latent rescaling over the evaluated range, we normalize patch features onto a product of hyperspheres and model generation using Riemannian Flow Matching (RFM). Spherical geodesics avoid the radial conditional velocity targets introduced by Euclidean interpolation, while tangent projection and manifold-aware sampling maintain the fixed-radius constraint. Stochastic latent smoothing further broadens the decoder's training inputs around fixed encoder outputs, encouraging robustness to latent perturbations for downstream generation.

On ImageNet-1K at 256×256, the generation-oriented HAE achieves an rFID of 0.78 and a PSNR of 25.2 dB. Using these latents, LightningDiT-XL achieves a gFID of 1.79 without guidance and 1.63 with auto-guidance after 1,000 epochs, without an additional wide generator head. Together, these results highlight the value of combining reconstruction-aware architecture, directional feature alignment, and generative dynamics consistent with the normalized latent geometry.

AI Use Statement

We used AI-assisted tools to improve the grammar, wording, and clarity of the manuscript. The authors take full responsibility for the final content.

REFERENCES

Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *ICML*, 2025.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024.

Ricky T. Q. Chen and Yaron Lipman. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Patrick Esser et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235. PMLR, 2024. URL https://proceedings.mlr.press/v235/esser24a.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*. PMLR, 2023.

Dengyang Jiang, Mengmeng Wang, Liuzhuozheng Li, Lei Zhang, Haoyu Wang, Wei Wei, Guang Dai, Yanning Zhang, and Jingdong Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. In *ICLR*, 2026.

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

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. In *ICCV*, 2025.

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

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

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. 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *CVPR*, 2022.

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.

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

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *ICLR*, 2021.

Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. *arXiv preprint arXiv:2309.03350*, 2023.

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

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

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9929–9939. PMLR, 2020. URL https://proceedings.mlr.press/v119/wang20k.html.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, et al. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *Adv. Neural Inform. Process. Syst.*, 2025.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

Junno Yun, Yaşar Utku Alçalar, and Mehmet Akçakaya. No alignment needed for generation: Learning linearly separable representations in diffusion models. *arXiv preprint arXiv:2509.21565*, 2025.

Shilong Zhang, He Zhang, Zhifei Zhang, Chongjian Ge, Shuchen Xue, Shaoteng Liu, Mengwei Ren, Soo Ye Kim, Yuqian Zhou, Qing Liu, Daniil Pakhomov, Kai Zhang, Zhe Lin, and Ping Luo. Both semantics and reconstruction matter: Making representation encoders ready for text-to-image generation and editing. *arXiv preprint arXiv:2512.17909*, 2025. URL https://arxiv.org/abs/2512.17909.

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

A Additional Experiments

This appendix provides controlled stem studies, computational-cost measurements, and additional generation results. Training budgets and evaluation settings are stated separately for each experiment.

A.1 Comparison with RAE Using DINOv3-L

Experimental setup. To examine whether the reconstruction improvement can be explained by the choice of pretrained backbone, we reproduce RAE (Zheng et al., 2025) using DINOv3-L. The reproduced model retains the standard DINOv3-L patch embedding and frozen Transformer, together with the original RAE decoder and training recipe. We modify the decoder input interface to accommodate the 1,024-dimensional DINOv3-L features. RAE and HAE use the same dataset, 256×256 reconstruction resolution, and reconstruction evaluation protocol. The reproduced RAE is denoted RAE (DINOv3-L) below.

We distinguish two HAE settings: HAE-R is the reconstruction-oriented setting without latent smoothing, and HAE-G is the generation-oriented setting with latent smoothing. These names refer to the evaluated settings rather than to the two training stages.

Table 9: Reconstruction comparison using the same DINOv3-L Transformer backbone. HAE-R and HAE-G denote the settings without and with latent smoothing, respectively. LPIPS uses the VGG-based metric.

Method	rFID ↓	PSNR ↑	SSIM ↑	LPIPS-VGG ↓
RAE (DINOv3-L)	0.6725	18.9000	0.4685	0.2661
HAE-R	**0.3700**	**26.2000**	**0.7539**	**0.1353**
HAE-G	0.7800	25.2000	0.7408	0.1500

Results and scope. HAE-R improves all four reconstruction metrics relative to the reproduced RAE baseline. HAE-G has higher PSNR and SSIM and lower LPIPS-VGG than this baseline, but a higher rFID. These results separate the reconstruction comparison from the backbone upgrade and make the reconstruction–generation trade-off explicit. This is a same-backbone system comparison: the decoder and training procedure remain method-specific. The complete reconstruction gap therefore cannot be attributed to HCPE alone; the stem-specific controls are reported in Appendix A.3.

A.2 Qualitative Reconstruction Comparisons

Figure 7 presents paired reconstruction examples from RAE and HAE-G. The examples illustrate differences in text and numerical patterns, thin structures, natural textures, and overall image appearance.

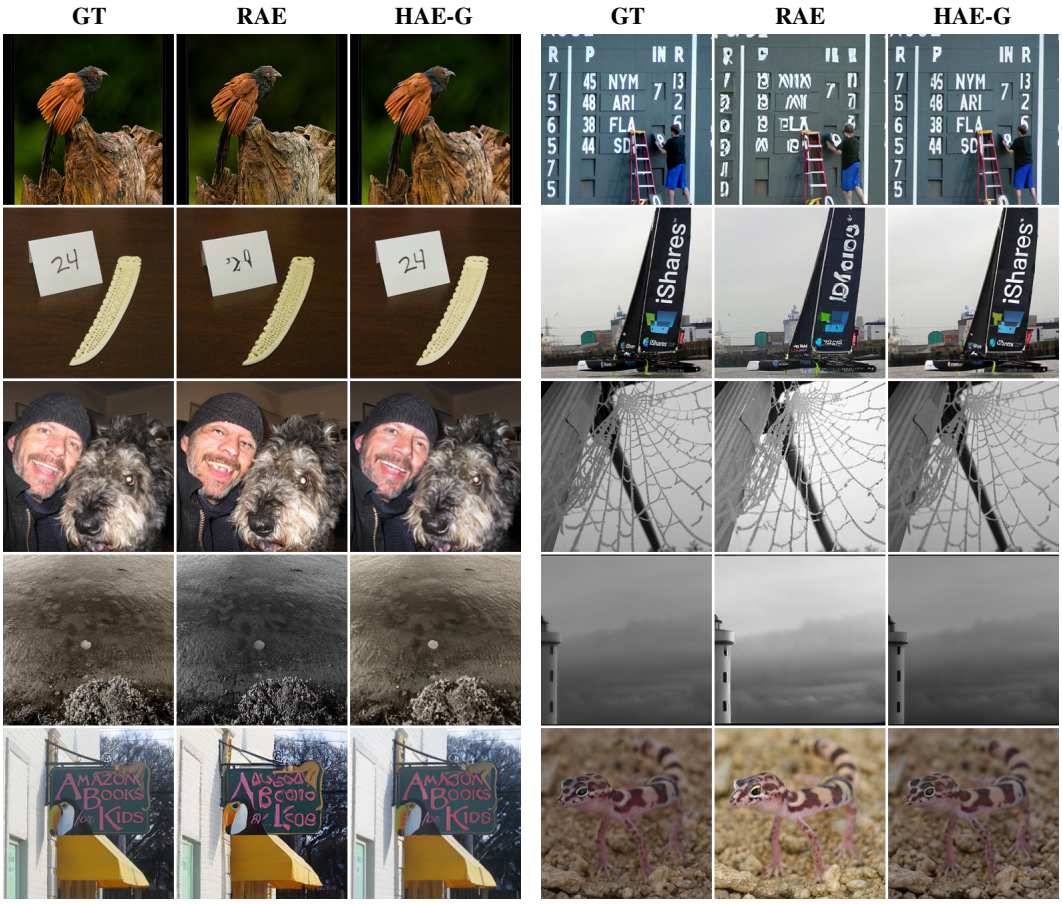

Figure 7: **Qualitative reconstruction comparison.** Each group contains five examples, with columns showing the ground-truth input (GT), RAE reconstruction, and HAE-G reconstruction from left to right. Each row compares reconstructions of the same input. The examples show differences in the reproduction of text, numerical patterns, fine structures, textures, and color.

A.3 Parameter-Matched Patch-Embedding Controls

Controlled setup. We compare three stems under the same 10-epoch Stage-1 training protocol. All variants use the same random seed, minibatch order, frozen DINOv3-L Transformer, teacher encoder, decoder initialization, objectives, and optimization settings. Only the patch-embedding stem differs.

Stem variants. HCPE applies four stride-2 convolutional stages, with stride schedule $[2, 2, 2, 2]$, progressively reducing the spatial resolution to the final 16×16 token grid. The *early-collapse control* matches HCPE's convolutional depth and approximately matches its parameter count, but uses stride schedule $[16, 1, 1, 1]$. Its later layers therefore operate only after the initial reduction to the final token grid. The *standard-patch capacity control* retains the standard stride-16 DINOv3 patch projection and adds spatially pointwise 1×1 channel-mixing layers to approximately match HCPE's parameter count.

Interpretation. HCPE improves all four metrics relative to both controls under this matched-budget protocol. The PSNR gains are 0.6206 dB over the early-collapse control and 0.2640 dB over the

Table 10: Parameter-matched stem controls after 10 epochs of Stage-1 training. Parameter counts refer to the stem, not the complete autoencoder. All non-stem settings are shared.

Stem	Parameters	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Early-collapse control	2,736,060	6.0481	21.1053	0.5409	0.3022
Standard-patch capacity control	2,738,104	3.4701	21.4619	0.5649	0.2545
HCPE	2,737,792	**3.2056**	**21.7259**	**0.5840**	**0.2509**

standard-patch capacity control. The latter has 312 more stem parameters than HCPE. The results support a benefit from allocating stem capacity across progressively reduced spatial resolutions, rather than adding capacity only after single-shot patchification. These 10-epoch measurements are not the final HAE-R checkpoint and do not quantify the contribution of HCPE to the entire RAE–HAE reconstruction gap.

A.4 Computational Cost

A.4.1 Autoencoder Cost

We report both counted arithmetic operations and measured inference cost for the reproduced RAE and HAE with DINOv3-L. All autoencoder measurements use a 256×256 input and batch size one.

Table 11: Autoencoder operation counts and measured inference cost. Both methods use DINOv3-L. GFLOPs count registered operations; latency and peak allocated memory are measured on one H200 in BF16.

Method	Parameters	Encoder GFLOPs	Decoder GFLOPs	Total GFLOPs
RAE	718.8M	164.7	221.8	386.5
HAE	477.1M	167.1	2,087.7	2,254.8

Method	Encode (ms)	Decode (ms)	Round trip (ms)	Peak memory (GiB)
RAE	21.90	17.48	39.60	2.74
HAE	22.09	12.64	34.86	2.16

HAE has a higher counted GFLOP cost, primarily in the decoder, but fewer parameters, lower measured decoder and round-trip latency, and lower peak allocated memory on this hardware. We therefore distinguish arithmetic counts from measured execution cost and do not claim lower GFLOPs for HAE.

A.4.2 Euclidean and Riemannian Sampling Cost

Euclidean Flow Matching (EFM) and RFM use the same LightningDiT-XL/1 architecture and a sampling budget of 50 network function evaluations (NFE). Their Transformer-forward operation counts are identical; RFM-specific sampling operations are reflected in the measured latency.

RFM adds 15.22 ms (1.33%) to sampler latency and 20.45 ms (1.70%) to noise-to-image latency, with no measured increase in peak allocated memory in this benchmark. This inference benchmark does not measure the SLERP/geodesic interpolation used during RFM training.

Measurement protocol. Autoencoder measurements use one NVIDIA H200, GPU-resident inputs, batch size one, and BF16. Encode, decode, and round-trip paths are timed separately. Each path uses three repetitions of 50 warm-up iterations followed by 200 CUDA-event-timed iterations, giving 600 timed iterations per path.

For flow sampling, we use the same H200 with batch size one, FP32, TF32 disabled, no guidance, and 50 Euler steps. EFM and RFM share the LightningDiT-XL/1 model, initial latent, and class

Table 12: EFM and RFM sampling costs at batch size one with 50 NFE. Transformer operation counts exclude the decoder and unsupported geometry operations. Latency includes the operations executed in the corresponding sampling path.

Method	Transformer parameters	GFLOPs/NFE	TFLOPs/50 NFE
EFM	677.5M	238.4	11.92
RFM	677.5M	238.4	11.92

Method	Sampler (ms)	Noise-to-image (ms)	Peak memory (GiB)
EFM	1,148.70	1,201.63	3.60
RFM	1,163.92	1,222.08	3.60

label. The sampler and noise-to-image paths are timed separately, each with three repetitions of five warm-up runs and ten timed runs.

We use PyTorch 2.9.1 `FlopCounterMode` to measure GFLOPs, counting each multiply–accumulate operation as two FLOPs. Only registered operations are included. The counter omits operations including normalization, activation, softmax, interpolation, RoPE and trigonometric operations, and other elementwise computations. The reported 11.92 TFLOPs per 50 NFE is Transformer-only; it excludes the decoder and unsupported geometry operations, whose execution is included in the end-to-end timing.

A.5 Numerical Configuration Comparison

Intermediate checkpoints. The original RFM configuration uses FP32 without QK-Norm. The extended run uses BF16 with QK-Norm. We compare intermediate checkpoints before reporting the extended run's 1,000-epoch results.

Table 13: Original and refined training configurations. The 80-epoch comparison uses matched training budgets. The 540/550-epoch pair is a nearby-checkpoint comparison, not an epoch-matched ablation.

Configuration	Precision	QK-Norm	Epochs	gFID ↓
Original	FP32	No	80	2.6500
Refined	BF16	Yes	80	2.6845
Original	FP32	No	550	1.9600
Refined	BF16	Yes	540	1.9577

The configurations differ by 0.0345 gFID at 80 epochs and have similar gFID at the nearby 540/550-epoch checkpoints. Precision and QK-Norm are changed together, so this comparison does not isolate their individual effects.

Generation Results. The refined configuration uses BF16 and QK-Norm and is trained for 1,000 epochs. Table 14 reports its unguided and auto-guided results. These results extend the evaluated training duration; they are not an equal-budget comparison with baselines trained for 800 epochs. The improvement relative to the original 550-epoch result is also not attributed solely to additional epochs, because the training configuration changes.

Table 14: Generation results at 1,000 epochs using BF16/QK-Norm configuration. Both rows use the same training budget.

Sampling	gFID ↓	IS ↑	Precision ↑	Recall ↑
Unguided	1.79	243.0	0.72	0.70
Auto-guidance	1.63	247.2	0.72	0.71

A.6 EMA DECAY

We evaluate EMA decay rates of 0.9999 and 0.9995 at three training budgets. Table 15 reports the measurements from this EMA comparison separately from the main generation results in Table 3.

Table 15: EMA-decay comparison. Each column compares the two decay settings at the same training budget. Lower gFID is better.

EMA decay	40 epochs	80 epochs	90 epochs
0.9999	3.24	2.88	2.82
0.9995	**3.12**	**2.70**	**2.64**

The 0.9995 decay gives lower gFID at all three evaluated budgets, supporting its use in the evaluated RFM training configuration.

B TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

In this appendix, we detail the training and inference procedures for Riemannian Flow Matching (RFM) on the HAE latent space.

B.1 TRAINING ALGORITHM

Algorithm 1 outlines the training of our manifold-aware DiT. We strictly constrain the generative process to a hypersphere of radius $R = 32.0$. Since our DINOv3-Large encoder inherently extracts $C = 1024$ dimensional features, setting $R = \sqrt{2}\frac{\Gamma((C+1)/2)}{\Gamma(C/2)} \approx 32.0$ ensures unit variance per channel and optimal numerical scaling.

Furthermore, recent studies Teng et al. (2023); Chen (2023); Hoogeboom et al. (2023); Esser et al. (2024) indicate that high-dimensional latent spaces (e.g., RAE Zheng et al. (2025)) suffer from under-corrupted signals at standard noise levels, impairing diffusion training. To compensate for our exceptionally high channel capacity, we adopt a dimension-dependent time-shifting schedule Esser et al. (2024). The shift factor, mathematically defined relative to a base dimension of 4096, evaluates precisely to $\alpha = \sqrt{(C \times H \times W)/4096} = \sqrt{(1024 \times 16^2)/4096} = 8.0$. This properly calibrates the noise schedule, dynamically allocating representational capacity to high-curvature flow regions.

Finally, the neural network's unconstrained output is orthogonalized into tangent and radial components. A radial penalty $\lambda_{\text{rad}} = 0.1$ is applied to regularize non-tangential velocity predictions.

Algorithm 1 Training Riemannian Flow Matching on HAE

Require: Pre-trained Encoder $\mathcal{E}$, DiT v_θ, Radius $R = 32.0$, Time-shift $\alpha = 8.0$, Penalty $\lambda_{\text{rad}} = 0.1$

Initialize DiT parameters θ
while not converged **do**
  Sample image $\mathbf{x} \sim p_{\text{data}}$
  Extract and project latent: $\mathbf{z}_0 = R \cdot \frac{\mathcal{E}(\mathbf{x})}{\|\mathcal{E}(\mathbf{x})\|_2}$ ▷ Spherical Projection
  Sample and project noise: $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{z}_1 \leftarrow R \cdot \frac{\mathbf{z}_1}{\|\mathbf{z}_1\|_2}$
  Sample $t_{\text{base}} \sim \text{LogitNormal}(0, 1)$, shift time: $t \leftarrow \frac{\alpha \cdot t_{\text{base}}}{1+(\alpha-1)t_{\text{base}}}$
  // Geodesic computations
  Angular distance: $\Omega = \arccos\left(\left\langle \frac{\mathbf{z}_0}{R}, \frac{\mathbf{z}_1}{R} \right\rangle\right)$ ▷ Geodesic SLERP setup
  Interpolate state: $\mathbf{z}_t = R\left(\frac{\sin((1-t)\Omega)}{\sin(\Omega)}\frac{\mathbf{z}_0}{R} + \frac{\sin(t\Omega)}{\sin(\Omega)}\frac{\mathbf{z}_1}{R}\right)$
  Target velocity: $\mathbf{u}_t = R\frac{\Omega}{\sin(\Omega)}\left(\cos(t\Omega)\frac{\mathbf{z}_1}{R} - \cos((1-t)\Omega)\frac{\mathbf{z}_0}{R}\right)$
  Predict velocity $\mathbf{v} = v_\theta(\mathbf{z}_t, t)$ and normal vector $\bar{\mathbf{z}}_t = \frac{\mathbf{z}_t}{\|\mathbf{z}_t\|_2}$
  Target decomp.: $\mathbf{u}_{\text{rad}} = \langle \mathbf{u}_t, \bar{\mathbf{z}}_t \rangle \bar{\mathbf{z}}_t, \quad \mathbf{u}_{\text{tan}} = \mathbf{u}_t - \mathbf{u}_{\text{rad}}$
  Predict decomp.: $\mathbf{v}_{\text{rad}} = \langle \mathbf{v}, \bar{\mathbf{z}}_t \rangle \bar{\mathbf{z}}_t, \quad \mathbf{v}_{\text{tan}} = \mathbf{v} - \mathbf{v}_{\text{rad}}$
  Loss: $\mathcal{L} = \text{Mean}\left(\|\mathbf{v}_{\text{tan}} - \mathbf{u}_{\text{tan}}\|_2^2\right) + \lambda_{\text{rad}} \cdot \text{Mean}\left(\|\mathbf{v}_{\text{rad}}\|_2^2\right)$ ▷ RFM + Penalty
  Update θ using $\nabla_\theta \mathcal{L}$
end while
return Optimized parameters θ

B.2 INFERENCE ALGORITHM

Algorithm 2 outlines the inference process utilizing the Riemannian Euler sampler. The sampling trajectory strictly adheres to the geometry of the hypersphere via the exponential map (Rodrigues' rotation formula). To maintain consistency with the training phase, the same time-shifting schedule ($\alpha = 8.0$) is applied. For the 80-epoch CFG result, the model evaluates conditioned and unconditioned states using a null token. The 1,000-epoch result uses auto-guidance with a separate weaker conditional model. Both use the guided velocity $\mathbf{v} = \mathbf{v}_{\text{ref}} + w(\mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{ref}})$. The resulting velocity is then explicitly projected onto the tangent space before stepping forward to ensure manifold constraint satisfaction. Finally, before passing the generated latent to the pre-trained HAE decoder, we explicitly rescale the latent norm from the Flow Matching radius ($R = 32.0$) to a designated decoding scale of 14.0. As detailed in our ablation study (Appendix E), this rescaling step optimally calibrates the feature variance fed into the decoder, preventing numerical instability while preserving the network's capacity to synthesize intricate high-frequency details.

Algorithm 2 Inference with Riemannian Euler Sampler and Guidance

Require: Trained DiT v_θ, Pre-trained Decoder $\mathcal{D}$, Radius $R = 32.0$, Steps N, Time-shift $\alpha = 8.0$, Guidance scale w, Cond. y, reference velocity v_{ref} (null-token model for CFG; weaker model for AG), VAE scale $s_{\text{vae}} = 14.0$

1: Sample noise $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and project: $\mathbf{z}_1 \leftarrow R \cdot \frac{\mathbf{z}_1}{\|\mathbf{z}_1\|_2}$
2: Base time $t_{\text{base}} = \{1.0, 1.0 - \frac{1}{N}, \ldots, 0.0\}$, shift time: $t_i = \frac{\alpha \cdot t_{\text{base},i}}{1+(\alpha-1)t_{\text{base},i}}$
3: **for** $i = 1$ to N **do**
4: $\quad \Delta t = t_{i+1} - t_i$ ▷ Negative time step for reverse generation
5: $\quad$ ***// CFG or Auto-Guidance Step***
6: $\quad$ Predict reference velocity: $\mathbf{v}_{\text{ref}} = v_{\text{ref}}(\mathbf{z}_{t_i}, t_i)$
7: $\quad$ Predict conditioned velocity: $\mathbf{v}_{\text{cond}} = v_\theta(\mathbf{z}_{t_i}, t_i, y)$
8: $\quad$ Apply guidance: $\mathbf{v} = \mathbf{v}_{\text{ref}} + w \cdot (\mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{ref}})$
9: $\quad$ Project to tangent space: $\mathbf{v} \leftarrow \mathbf{v} - \left\langle \mathbf{v}, \frac{\mathbf{z}_{t_i}}{R} \right\rangle \frac{\mathbf{z}_{t_i}}{R}$ ▷ Remove radial component
10: $\quad$ Compute rotation angle $\theta = \frac{|\Delta t| \cdot \|\mathbf{v}\|_2}{R}$ and direction $\bar{\mathbf{v}} = \text{sign}(\Delta t) \frac{\mathbf{v}}{\|\mathbf{v}\|_2 + \epsilon}$
11: $\quad$ Exponential map: $\mathbf{z}_{\text{next}} = \mathbf{z}_{t_i} \cos(\theta) + R\bar{\mathbf{v}} \sin(\theta)$ ▷ Rodrigues' rotation
12: $\quad$ Re-normalize: $\mathbf{z}_{t_{i+1}} = R \cdot \frac{\mathbf{z}_{\text{next}}}{\|\mathbf{z}_{\text{next}}\|_2}$
13: **end for**
14: Rescale to target VAE scale $\mathbf{z}_0 \leftarrow s_{\text{vae}} \cdot \frac{\mathbf{z}_0}{R}$ and Decode $\hat{\mathbf{x}} = \mathcal{D}(\mathbf{z}_0)$
15: **return** $\hat{\mathbf{x}}$

C Decoder Sensitivity and the Geometry of Flow Matching

This appendix develops two complementary arguments motivating the use of Riemannian Flow Matching (RFM) in Section 3.4. First, a local bound relates decoded one-step errors to the decoder's radial and tangential sensitivities. Second, an exact calculation shows that Euclidean interpolation introduces a radial conditional velocity target even when both endpoints have the same norm. Together, these results motivate removing radial transport when the decoder is relatively insensitive to it. They do not establish a general ordering of generative quality between Euclidean and Riemannian flow matching.

C.1 Patch-wise Geometry and Decoder Sensitivity

Write a latent as $\mathbf{z} = (\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}) \in \mathbb{R}^{NC}$, where N is the number of patches and C is the channel dimension. The fixed-radius latent manifold is

$$\mathcal{M} = \prod_{i=1}^{N} \left\{ \mathbf{z}^{(i)} \in \mathbb{R}^C : \|\mathbf{z}^{(i)}\|_2 = R \right\}. \tag{9}$$

For nonzero patch vectors, define the patch-wise unit directions and orthogonal projectors

$$\begin{aligned} \mathbf{n}^{(i)}(\mathbf{z}) &= \frac{\mathbf{z}^{(i)}}{\|\mathbf{z}^{(i)}\|_2}, \\ P_{\mathrm{rad}}(\mathbf{z}) &= \mathrm{diag}_{i=1}^{N}\left(\mathbf{n}^{(i)}(\mathbf{z}) \mathbf{n}^{(i)}(\mathbf{z})^\top \right), \qquad P_{\tan}(\mathbf{z}) = I - P_{\mathrm{rad}}(\mathbf{z}). \end{aligned} \tag{10}$$

On $\mathcal{M}$, these project onto the normal and tangent spaces of the product manifold, respectively. Away from $\mathcal{M}$, they still define radial and tangential directions relative to each patch. All latent norms below are Euclidean norms of the concatenated patches, and image norms are Euclidean norms of the pixel arrays.

To account for the fixed rescaling before decoding in Algorithm 2, let

$$D_s(\mathbf{z}) = \mathcal{D}\Big(\frac{s}{R} \mathbf{z} \Big), \qquad R = 32, \quad s = s_{\mathrm{vae}} = 14, \tag{11}$$

where $\mathcal{D}$ is the trained HAE decoder. This definition applies a fixed linear rescaling, without an additional normalization of perturbed latents. Radial insensitivity is therefore a property to be assessed for the decoder, rather than a consequence of discarding radial perturbations by construction.

Let $\mathcal{Z} \subset \mathbb{R}^{NC}$ be a region in which all patch vectors are nonzero. For a differentiable D_s, define

$$\begin{aligned} \eta_{\mathrm{rad}} &= \sup_{\mathbf{z} \in \mathcal{Z}} \left\| J_{D_s}(\mathbf{z}) P_{\mathrm{rad}}(\mathbf{z}) \right\|_{\mathrm{op}}, \\ \eta_{\tan} &= \sup_{\mathbf{z} \in \mathcal{Z}} \left\| J_{D_s}(\mathbf{z}) P_{\tan}(\mathbf{z}) \right\|_{\mathrm{op}}, \end{aligned} \tag{12}$$

where J_{D_s} is the decoder Jacobian and $\|\cdot\|_{\mathrm{op}}$ denotes the operator norm induced by the Euclidean norms. The hypothesis $\eta_{\mathrm{rad}} \ll \eta_{\tan}$ expresses low radial sensitivity over the specified region. Figure 5 provides qualitative motivation for this hypothesis, but does not estimate these suprema or establish the hypothesis along the entire flow path. In particular, the scale dependence in Appendix E precludes assuming global invariance to magnitude.

C.2 A Local Bound on Decoded One-step Error

Fix a flow time t and let $Z_t \sim p_t$ take values in $\mathcal{Z}$. Let $\mathbf{u}_t$ be a reference velocity field and $\mathbf{v}_\theta(\cdot, t)$ its learned approximation. Consider their one-step Euclidean updates from the same state distribution:

$$q^\star_{t,h} = (\mathrm{Id} + h\mathbf{u}_t)_{\#} p_t, \qquad q^\theta_{t,h} = (\mathrm{Id} + h\mathbf{v}_\theta(\cdot, t))_{\#} p_t. \tag{13}$$

Here $F_{\#}p$ denotes the distribution of $F(Z)$ for $Z \sim p$, and h may be negative for reverse-time sampling. Decompose the velocity error at Z_t as

$$\begin{aligned} \mathbf{e}_t &= \mathbf{v}_\theta(Z_t, t) - \mathbf{u}_t(Z_t) = \mathbf{e}_{\mathrm{rad}} + \mathbf{e}_{\tan}, \\ \mathbf{e}_{\mathrm{rad}} &= P_{\mathrm{rad}}(Z_t)\mathbf{e}_t, \qquad \mathbf{e}_{\tan} = P_{\tan}(Z_t)\mathbf{e}_t, \\ L_{\mathrm{rad}}(t) &= \mathbb{E}\|\mathbf{e}_{\mathrm{rad}}\|_2^2, \qquad L_{\tan}(t) = \mathbb{E}\|\mathbf{e}_{\tan}\|_2^2. \end{aligned} \tag{14}$$

Assumptions. Assume that Z_t, $D_s(Z_t)$, $\mathbf{u}_t(Z_t)$, and $\mathbf{v}_\theta(Z_t, t)$ have finite second moments. Assume also that D_s is continuously differentiable with a uniformly bounded Jacobian on an open region containing the line segments from Z_t to both updates for all sufficiently small $|h|$, almost surely. These conditions ensure that the first-order Taylor remainders vanish in L^2 after division by $|h|$ and that the Wasserstein distances below are well defined.

Proposition 2 (Local decoded-error bound) *Under these assumptions,*

$$W_2^2(q_{t,h}^\theta, q_{t,h}^\star) \le h^2 \left[L_{\mathrm{rad}}(t) + L_{\tan}(t)\right], \tag{15}$$

and

$$\limsup_{h\to 0} \frac{W_2\Big((D_s)_\# q_{t,h}^\theta, (D_s)_\# q_{t,h}^\star\Big)}{|h|} \le \eta_{\mathrm{rad}} \sqrt{L_{\mathrm{rad}}(t)} + \eta_{\tan} \sqrt{L_{\tan}(t)}, \tag{16}$$

where W_2 is the quadratic Wasserstein distance.

Proof. Couple the two updates through the same Z_t. Their difference is $h\mathbf{e}_t$, so the definition of W_2 and the orthogonality of the projectors give

$$W_2^2(q_{t,h}^\theta, q_{t,h}^\star) \le h^2 \mathbb{E}\|\mathbf{e}_t\|_2^2 = h^2 \left[L_{\mathrm{rad}}(t) + L_{\tan}(t)\right]. \tag{17}$$

For the decoded updates, the same coupling yields

$$\begin{aligned} &W_2^2\big((D_s)_\# q_{t,h}^\theta, (D_s)_\# q_{t,h}^\star\big) \\ &\quad \le \mathbb{E}\left\|D_s(Z_t + h\mathbf{v}_\theta(Z_t,t)) - D_s(Z_t + h\mathbf{u}_t(Z_t))\right\|_2^2. \end{aligned} \tag{18}$$

By the assumed regularity, the difference inside the norm is $hJ_{D_s}(Z_t)\mathbf{e}_t + o_{L^2}(|h|)$, where $o_{L^2}(|h|)$ denotes a remainder whose root-mean-square norm divided by $|h|$ tends to zero. Dividing the square root of the preceding inequality by $|h|$ gives a limiting upper bound of $\big(\mathbb{E}\|J_{D_s}(Z_t)\mathbf{e}_t\|_2^2\big)^{1/2}$. Minkowski's inequality and Eq. (12) then imply

$$\begin{aligned} \big(\mathbb{E}\|J_{D_s}(Z_t)\mathbf{e}_t\|_2^2\big)^{1/2} &\le \big(\mathbb{E}\|J_{D_s}(Z_t)\mathbf{e}_{\mathrm{rad}}\|_2^2\big)^{1/2} \\ &\quad + \big(\mathbb{E}\|J_{D_s}(Z_t)\mathbf{e}_{\tan}\|_2^2\big)^{1/2} \\ &\le \eta_{\mathrm{rad}} \sqrt{L_{\mathrm{rad}}(t)} + \eta_{\tan} \sqrt{L_{\tan}(t)}. \end{aligned} \tag{19}$$

This proves Eq. (16). □

Eq. (15) weights radial and tangential velocity errors equally in latent space, whereas Eq. (16) accounts for the decoder's directional sensitivities. If η_{rad} is small, radial error can contribute to the latent regression loss while producing only a small first-order decoded discrepancy. The bound does not imply that every tangential error has a larger decoded effect: it provides an upper bound, not a lower bound on tangential sensitivity. It is a local one-step statement and does not establish a global sampling-error bound or an ordering of gFID scores.

C.3 Radial Target Energy of Euclidean Interpolation

We next examine a single patch with endpoints $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^C$ satisfying $\|\mathbf{x}_0\|_2 = \|\mathbf{x}_1\|_2 = R$. Let their angle be $\omega \in (0, \pi)$; identical endpoints have zero target velocity, and antipodal endpoints are excluded because their chord passes through the origin. Euclidean conditional flow matching uses the path and target

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \qquad \mathbf{U} = \mathbf{x}_1 - \mathbf{x}_0. \tag{20}$$

The path satisfies

$$\|\mathbf{x}_t\|_2^2 = R^2 - t(1-t)\|\mathbf{U}\|_2^2 < R^2, \qquad 0 < t < 1. \tag{21}$$

Thus, equal endpoint norms do not eliminate radial motion: the chord moves inward and then outward. Define $P_{\mathrm{rad}}(\mathbf{x}_t) = \mathbf{x}_t\mathbf{x}_t^\top / \|\mathbf{x}_t\|_2^2$ for this patch.

Proposition 1 (restated). For fixed endpoints as above,

$$\rho(\omega) := \frac{\int_0^1 \|P_{\mathrm{rad}}(\mathbf{x}_t)\mathbf{U}\|_2^2 \, dt}{\int_0^1 \|\mathbf{U}\|_2^2 \, dt} = 1 - \frac{\omega}{2}\cot\Big(\frac{\omega}{2}\Big). \tag{22}$$

Proof. Set $a = \omega/2$, $\mathbf{m} = (\mathbf{x}_0 + \mathbf{x}_1)/2$, and $\mathbf{d} = (\mathbf{x}_1 - \mathbf{x}_0)/2$. Equal endpoint norms imply

$$\langle \mathbf{m}, \mathbf{d} \rangle = 0, \qquad \|\mathbf{m}\|_2 = R\cos a, \qquad \|\mathbf{d}\|_2 = R\sin a. \tag{23}$$

Writing $r = 2t - 1$ gives $\mathbf{x}_t = \mathbf{m} + r\mathbf{d}$ and $\mathbf{U} = 2\mathbf{d}$. Therefore,

$$\frac{\|P_{\mathrm{rad}}(\mathbf{x}_t)\mathbf{U}\|_2^2}{\|\mathbf{U}\|_2^2} = \frac{\langle \mathbf{x}_t, \mathbf{U}\rangle^2}{\|\mathbf{x}_t\|_2^2\|\mathbf{U}\|_2^2} = \frac{r^2\sin^2 a}{\cos^2 a + r^2\sin^2 a}. \tag{24}$$

Since $\mathbf{U}$ is constant in time and the integrand is even in r,

$$\begin{aligned}\rho(\omega) &= \frac{1}{2}\int_{-1}^{1} \frac{r^2\sin^2 a}{\cos^2 a + r^2\sin^2 a}\, dr \\ &= 1 - \int_0^1 \frac{\cos^2 a}{\cos^2 a + r^2\sin^2 a}\, dr = 1 - \frac{\cos a}{\sin a}\arctan(\tan a) = 1 - a\cot a,\end{aligned} \tag{25}$$

where the last equality uses $a \in (0, \pi/2)$. □

When one endpoint is independent isotropic noise normalized to radius R, its direction is approximately orthogonal to a fixed data direction in high dimensions. For $\omega \approx \pi/2$, Eq. (22) gives

$$\rho(\pi/2) = 1 - \frac{\pi}{4} \approx 0.2146. \tag{26}$$

Thus, approximately 21.5% of the squared *conditional target velocity*, integrated uniformly over time, is radial for nearly orthogonal endpoints. This is a target-energy fraction, not a percentage of model capacity or an unavoidable prediction-error floor. For multiple patches or endpoint pairs, the aggregate fraction is weighted by their total target energies. Moreover, our shifted logit-normal time sampling in Algorithm 1 is not uniform; its corresponding fraction requires averaging the time-dependent energy under that sampling distribution instead.

In contrast, a fixed-radius geodesic $\boldsymbol{\gamma}_t$ satisfies

$$\|\boldsymbol{\gamma}_t\|_2^2 = R^2 \quad \Longrightarrow \quad \langle \dot{\boldsymbol{\gamma}}_t, \boldsymbol{\gamma}_t \rangle = 0, \tag{27}$$

so its radial conditional target is identically zero. The energy calculation above concerns endpoint-conditioned regression targets; it should not be identified with the radial energy of the learned marginal field, which involves averaging conditional velocities at a fixed state.

C.4 Implications for HAE

These arguments identify a potential mismatch between Euclidean regression and decoded image sensitivity. A Euclidean chord introduces radial targets even after endpoint normalization. If those directions have a small effect on the decoded image over the relevant latent region, the regression objective assigns weight to motion that has limited image-space effect. RFM removes this radial transport from the conditional target by using spherical geodesics. The radial penalty in our implementation regularizes the network's unprojected output, while tangent projection and the exponential-map update enforce the sampling geometry.

This motivation is consistent with the controlled comparison in Table 4, where Euclidean FM and RFM obtain gFID values of 15.95 and 2.65, respectively, using the same HAE latent space and LightningDiT-XL architecture at 80 epochs. The theoretical statements explain a possible mechanism for this empirical improvement under low radial decoder sensitivity; they do not predict its magnitude. Quantitative radial and tangential sensitivity estimates along the flow path would be needed to test that hypothesis beyond the qualitative reconstructions in Figure 5.

C.5 General Riemannian Flow Matching Background

Specifically, the RFM objective is defined as the regression loss

$$\mathcal{L}_{\mathrm{RFM}}(\theta) = \mathbb{E}_{t,\,\mathbf{x}_t}\Big[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}(\mathbf{x}_t, t)\|_g^2\Big], \tag{28}$$

where $\mathbb{E}$ averages over continuous time $t \in [0, 1]$ and intermediate state $\mathbf{x}_t$, p_t is the marginal density, and $\|\cdot\|_g$ denotes the norm induced by the Riemannian metric, and $\mathbf{u}(\mathbf{x}_t, t)$ is the marginal target flow generating a probability path between p and q.

Following conditional flow matching, the target vector field can be expressed as the marginalization of a conditional flow $\mathbf{u}_t(\mathbf{x} \mid \mathbf{x}_1)$:

$$\mathbf{u}(\mathbf{x}_t, t) = \int_{\mathcal{M}} \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1) \, \frac{p_t(\mathbf{x}_t \mid \mathbf{x}_1) \, q(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} \, d\mathrm{vol}_{\mathbf{x}_1}. \tag{29}$$

Here $q(\mathbf{x}_1)$ is the target density, $p_t(\mathbf{x}_t \mid \mathbf{x}_1)$ is the endpoint-conditioned density, and $d\mathrm{vol}_{\mathbf{x}_1}$ is the Riemannian volume element. A key distinction from standard FM lies in the design of the conditional flow on manifolds. RFM introduces the notion of a *premetric* $d(\mathbf{x}, \mathbf{x}_1)$ and constructs a conditional trajectory $\psi_t(\mathbf{x}_0 \mid \mathbf{x}_1)$ that monotonically decreases this distance:

$$d(\psi_t(\mathbf{x}_0 \mid \mathbf{x}_1), \mathbf{x}_1) = \kappa(t) \, d(\mathbf{x}_0, \mathbf{x}_1), \tag{30}$$

where $\kappa(t)$ is a decreasing scheduler satisfying $\kappa(0) = 1$ and $\kappa(1) = 0$. Under this formulation, the conditional vector field admits the closed-form expression

$$\mathbf{u}_t(\mathbf{x} \mid \mathbf{x}_1) = \frac{d \log \kappa(t)}{dt} \, d(\mathbf{x}, \mathbf{x}_1) \, \frac{\nabla d(\mathbf{x}, \mathbf{x}_1)}{\|\nabla d(\mathbf{x}, \mathbf{x}_1)\|_g^2}. \tag{31}$$

In the special case where the premetric is chosen as the geodesic distance d_g, the resulting conditional flow follows the geodesic connecting the endpoints, which can be expressed analytically using the exponential and logarithmic maps:

$$\psi_t(\mathbf{x}_0, \mathbf{x}_1) = \exp_{\mathbf{x}_1}\big(\kappa(t) \log_{\mathbf{x}_1}(\mathbf{x}_0)\big). \tag{32}$$

This construction makes RFM simulation-free on simple manifolds, while remaining scalable to general geometries.

D AUTOENCODER ARCHITECTURE AND TRAINING DETAILS

In this section, we provide the detailed architectural configurations and hyperparameter settings for training our HAE framework. As discussed in Section 3, the training process is divided into two stages to balance semantic alignment and reconstruction fidelity. Autoencoder training was performed on 8 NVIDIA B200 GPUs and required approximately 1 hour and 20 minutes per epoch. Here, we detail the core architectures and the foundational configuration used in Stage 1.

D.1 HIERARCHICAL CONVOLUTIONAL PATCH EMBEDDING

To mitigate the irreversible loss of high-frequency spatial details caused by the aggressive single-layer downsampling of standard ViTs, we replace the standard patch embedding with a Hierarchical Convolutional Stem. This module progressively downsamples the input image by a factor of 16 through four convolutional stages, smoothly projecting the 3-channel RGB input into the 1024-dimensional latent space required by the DINOv3-Large backbone.

The exact architectural specifications are detailed in Table 16. For the intermediate layers, we utilize Spatial Root Mean Square Normalization (TRMS2d) and SiLU activation functions to ensure stable gradient flow and robust feature extraction at the earliest stages.

Table 16: Architectural configuration of the Hierarchical Convolutional Patch Embedding.

Layer	**Kernel Size**	**Stride**	**Output Channels**	**Norm**	**Activation**
Input Image	-	-	3	-	-
ConvLayer 1	7×7	2	64	TRMS2d	SiLU
ConvLayer 2	3×3	2	128	TRMS2d	SiLU
ConvLayer 3	3×3	2	256	TRMS2d	SiLU
ConvLayer 4	3×3	2	1024	None	None

D.2 DECODER ARCHITECTURE

We adopt the highly efficient decoder architecture proposed in DC-AE Chen et al. (2024), tailored to our specific hyperspherical latent space. The decoder progressively upsamples the latent representation $\mathbf{z} \in \mathbb{R}^{H_z \times W_z \times 1024}$ back to the pixel space $\hat{\mathbf{x}} \in \mathbb{R}^{H \times W \times 3}$. The detailed layer-by-layer configuration is summarized in Table 17. Across all blocks, we utilize TRMS2d and SiLU activation. For the upsampling stages, we employ InterpolateConv blocks with channel matching, utilizing a duplicating shortcut connection to preserve structural integrity.

Table 17: Architectural configuration of the HAE Decoder.

Stage	**Width (Channels)**	**Depth (Blocks)**	**Block Type**
Input Latent	1024	-	-
Stage 1	1024	3	EViT-GLU
Stage 2	1024	3	EViT-GLU
Stage 3	512	3	ResBlock
Stage 4	512	5	ResBlock
Stage 5	256	5	ResBlock
Output Image	3	-	Conv2D (ReLU)

D.3 STAGE 1: SEMANTIC-STRUCTURAL ALIGNMENT HYPERPARAMETERS

In Stage 1, we optimize the hierarchical convolutional patch embedding alongside the decoder to establish a stable semantic latent space. The model is trained using the AdamW optimizer with a base learning rate of 1×10^{-5} for a total of 20 epochs. To ensure stable convergence, we employ Exponential Moving Average (EMA) and mixed-precision (BF16) training. The comprehensive hyperparameter configurations are detailed in Table 18.

Table 18: Hyperparameters for Stage 1 Autoencoder Training.

Hyperparameter	**Value**
Optimization Settings	
Optimizer	AdamW
Learning Rate	1×10^{-5}
Weight Decay	0.0
Optimizer Betas	$(0.9, 0.999)$
Global Batch Size	512
Precision	BF16
EMA Decay (β_{EMA})	0.9999
Total Epochs	20
Loss Objective Weights	
Reconstruction Weight (λ_{L1})	1.0
Perceptual Weight (λ_{lpips})	1.0
Cosine Similarity Weight ($\lambda_{\cos}$)	0.5

D.4 Stage 2: Adversarial Adaptation with Stochastic Latent Smoothing

Building upon the stable semantic space established in Stage 1, Stage 2 introduces adversarial training and stochastic latent smoothing to bridge the domain gap for downstream generative modeling. In this stage, we freeze the entire encoder, including HCPE and the DINOv3 Transformer, and train only the decoder alongside a feature-level discriminator (DINO-Disc). Following StyleGAN-T Sauer et al. (2023) and RAE Zheng et al. (2025), this DINO-Disc performs discrimination in the feature space of a pre-trained DINO model to ensure logically consistent textures.

To ensure stable adversarial dynamics, we apply a two-stage warmup strategy. For the first two epochs, the discriminator is trained independently to adapt to the generated distribution, while the generator is optimized solely with reconstruction losses. From the third epoch onwards, the adversarial loss ($\lambda_{\text{adv}} = 0.5$) is fully activated. Specifically, we employ a hinge loss formulation for the adversarial objective ($\mathcal{L}_{\text{GAN}}$), updating the overall training loss as follows:

$$\mathcal{L}_{\text{Stage2}} = \mathcal{L}_{\text{Stage1}}|_{\lambda_{\cos}=0} + \lambda_{\text{adv}}\mathcal{L}_{\text{GAN}}(\mathbf{x}, \hat{\mathbf{x}}) \tag{33}$$

Notably, we disable the cosine similarity loss ($\lambda_{\cos} = 0.0$) in this stage, as the encoder is frozen and further directional constraints are unnecessary.

Crucially, we implement a progressive noise schedule for the stochastic latent smoothing. The spherical noise scale τ is initialized at 0.8 and linearly increased to 1.2 over the first 20 epochs, after which it remains constant until the end of training at epoch 40. This curriculum smoothing effectively thickens the latent manifold without abruptly collapsing the reconstruction capability. Both the generator (decoder) and the discriminator are optimized using AdamW with a learning rate of 5×10^{-5} and modified momentum betas of $(0.5, 0.9)$ to stabilize the GAN training. The complete hyperparameter settings are summarized in Table 19.

E Ablation on Decoder Latent Scaling Factor

As illustrated in Section 3.1, our fully trained decoder exhibits strong robustness to latent magnitude shifts. However, during the image generation phase, the specific scaling factor s_{vae} applied to the latents generated via Riemannian Flow Matching (RFM) significantly impacts the final visual quality. Because the RFM process outputs generative trajectories strictly on a hypersphere of radius $R = 32.0$, these uncalibrated vectors must be properly rescaled before being passed to the decoder to synthesize the final image.

To determine the optimal decoding scale for these generated latents, we conducted a systematic ablation study by sweeping the parameter s_{vae} from 8.0 to 20.0 during the inference stage. As illustrated by the generative metrics in Figure 8, the performance exhibits a clear U-shaped trajectory

Table 19: Hyperparameters for Stage 2 Autoencoder Training (Adversarial Adaptation).

Hyperparameter	**Value**
Optimization Settings	
Optimizer (Generator & Discriminator)	AdamW
Learning Rate	5×10^{-5}
Weight Decay	0.0
Optimizer Betas	$(0.5, 0.9)$
Global Batch Size	512
Precision	BF16
EMA Decay (β_{EMA})	0.9975
Total Epochs	40
Discriminator Warmup Epochs	2
Stochastic Latent Smoothing	
Noise Scale (τ) Schedule	$0.8 \rightarrow 1.2$ (Epoch 0–20) 1.2 (Epoch 20–40)
Loss Objective Weights	
Reconstruction Weight (λ_{L1})	1.0
Perceptual Weight (λ_{lpips})	1.0
Adversarial Weight (λ_{adv})	0.5
Cosine Similarity Weight (λ_{cos})	0.0

with respect to the scaling factor. Extremely low scaling factors excessively compress the latent variance, yielding a poor gFID of 7.36 at $s_{\text{vae}} = 8.0$. In this constrained regime, the restricted magnitude fails to sufficiently excite the decoder's feature maps, preventing the network from synthesizing intricate high-frequency details.

As the scale increases, the generative quality improves sharply, reaching a clear "sweet spot" at $s_{\text{vae}} = 14.0$ with an optimal gFID of 2.65. This qualitative progression is explicitly visualized in Figure 9. At lower scaling factors (e.g., $s_{\text{vae}} = 8.0$ and 10.0), the constrained feature magnitude prevents the decoder from forming coherent geometry, resulting in images that suffer from structural fragmentation and severe tearing artifacts. As the scale increases, the generative quality improves sharply, reaching a clear "sweet spot" at $s_{\text{vae}} = 14.0$. At this optimal scale, the model achieves the highest image quality, perfectly resolving fine textures and intricate high-frequency details. Conversely, excessively high scaling values (e.g., $s_{\text{vae}} = 18.0$ and 20.0) push the latent activations far beyond the decoder's learned optimal range. This disruption in feature statistics causes the decoding process to over-smooth the outputs, leading to a noticeable loss of texture and increasingly blurry images. Thus, we adopted $s_{\text{vae}} = 14.0$ as the default rescaling factor during inference to optimally align the generated latent statistics with the decoder's representational capacity.

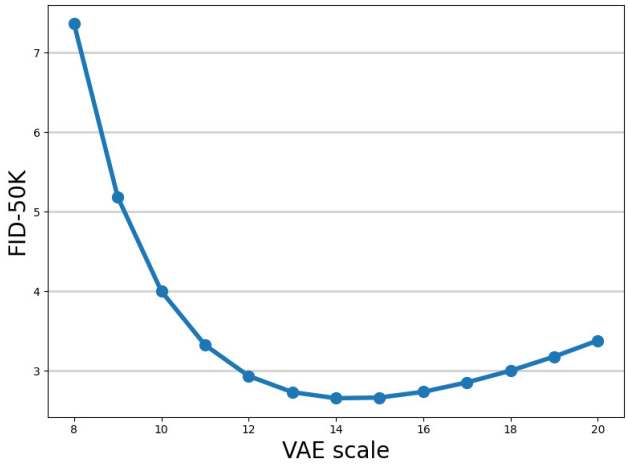

Figure 8: Effect of the latent scaling factor (s_{vae}) on generative performance. The model achieves an optimal gFID balance at $s_{\text{vae}} = 14.0$.



Figure 9: Generated Golden Retriever samples across different scales: (a) 8.0, (b) 10.0, (c) 14.0, (d) 18.0, and (e) 20.0. As shown, $s_{\text{vae}} = 14.0$ best preserves the intricate high-frequency details.

F UNCURATED GENERATED SAMPLES

In this section, we present uncurated 256×256 samples generated by LightningDiT-XL to provide a comprehensive view of its generative performance. All images were generated using the Riemannian Euler sampler for 50 steps with a Classifier-Free Guidance (CFG) scale of 1.1.



Figure 10: Uncurated 256×256 samples of class Goldfinch at CFG 1.1.

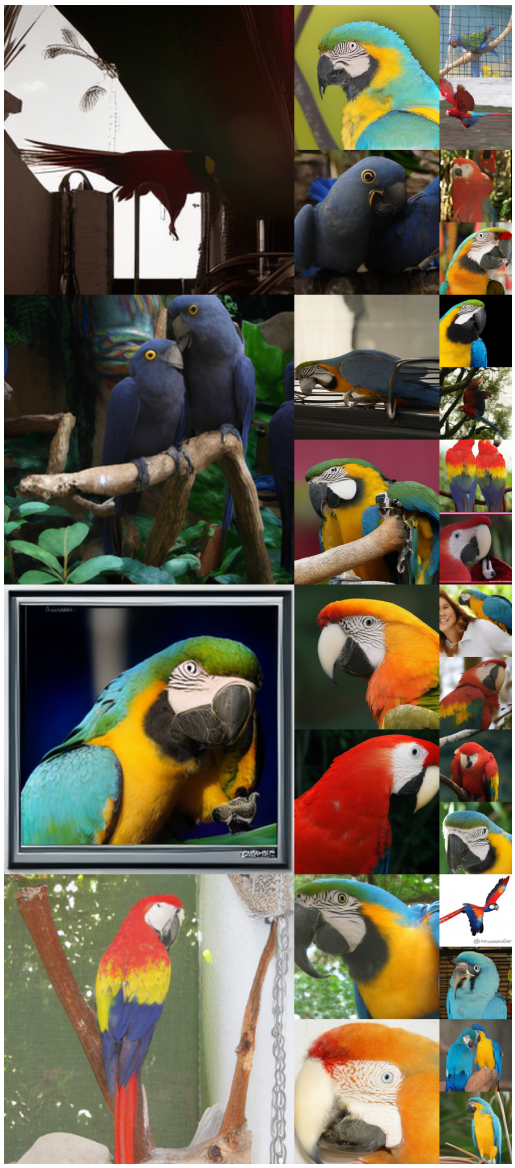

Figure 11: Uncurated 256×256 samples of class Macaw at CFG 1.1.



Figure 12: Uncurated 256 × 256 samples of class Golden Retriever at CFG 1.1.



Figure 13: Uncurated 256 × 256 samples of class Polar bear at CFG 1.1.



Figure 14: Uncurated 256 × 256 samples of class Jellyfish at CFG 1.1.

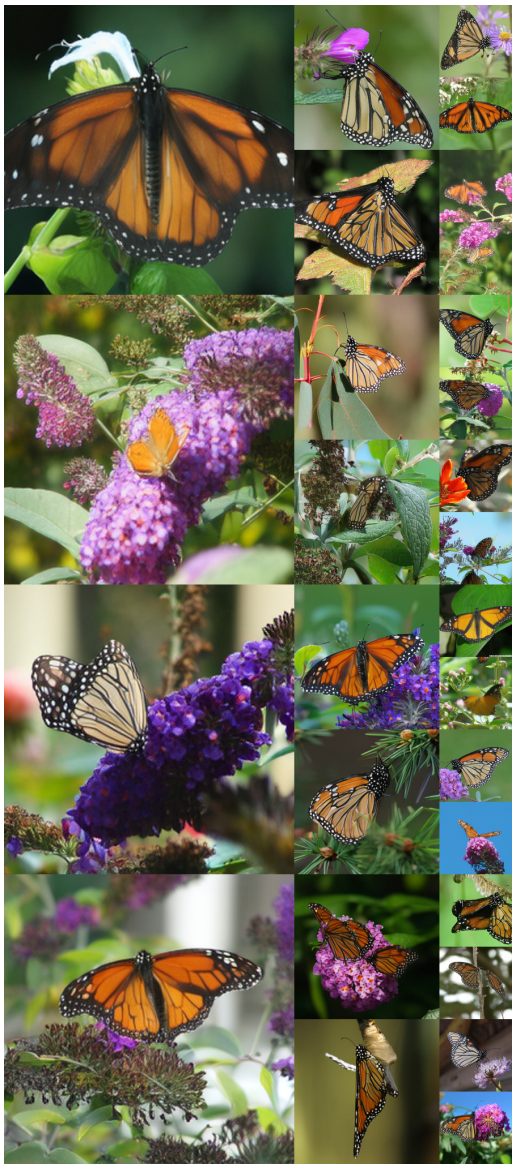

Figure 15: Uncurated 256 × 256 samples of class Monarch butterfly at CFG 1.1.



Figure 16: Uncurated 256 × 256 samples of class Balloon at CFG 1.1.

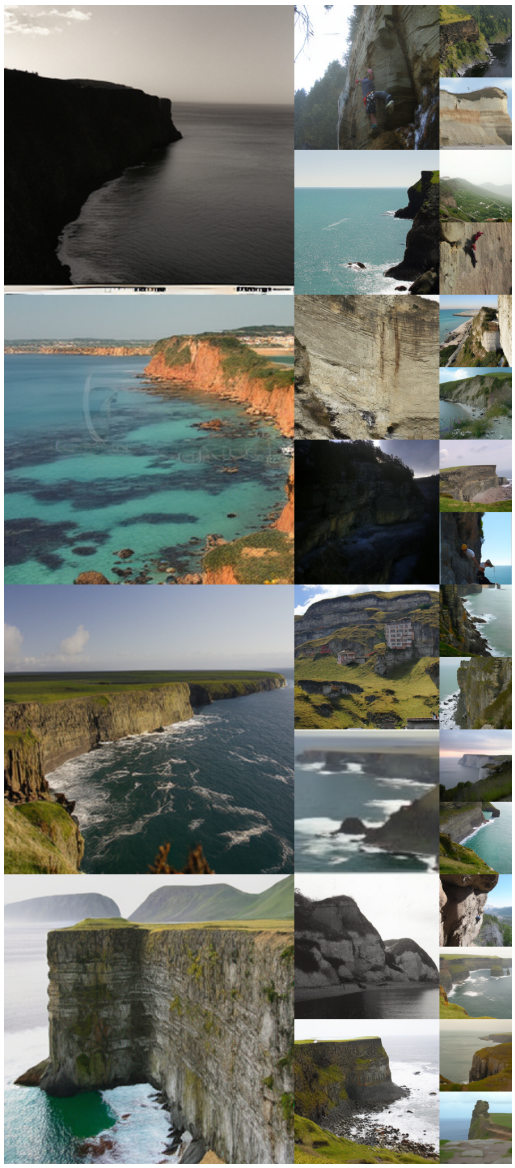

Figure 17: Uncurated 256 × 256 samples of class Cliff at CFG 1.1.



Figure 18: Uncurated 256 × 256 samples of class Daisy at CFG 1.1.



Figure 19: Uncurated 256 × 256 samples of class Volcano at CFG 1.1.



Figure 20: Uncurated 256 × 256 samples of class Hamburger at CFG 1.1.



Figure 21: Uncurated 256 × 256 samples of class Icecream at CFG 1.1.