BEYOND GAUSSIAN BOTTLENECKS: TOPOLOGICALLY ALIGNED ENCODING OF VISION-TRANSFORMER FEATURE SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained Vision Transformer (ViT) representations are rich and broadly useful, but their high dimensionality makes them expensive to store, transfer, and reuse. We study how to compress these representations while preserving downstream utility, and identify a fundamental geometric obstacle: layer-normalized ViT features concentrate on hyperspherical manifolds, whereas standard Gaussian variational autoencoders model latents in Euclidean space. This mismatch wastes capacity on a radial dimension that carries no semantic information and induces posterior collapse under strong compression. We propose S^2VAE, a geometry-aware variational autoencoder that replaces the Gaussian bottleneck with a product of Power Spherical distributions, aligning the latent topology with the data manifold while remaining numerically stable in high dimensions. Across five transformer backbones spanning distinct training objectives and modalities, including DINOv2, CLIP, DUSt3R, VGGT, and PAGE-4D, S²VAE consistently outperforms Gaussian baselines under aggressive compression (up to 512×), with the gap widening as the bottleneck tightens. Beyond reconstruction fidelity, we show that the learned latent space is directly usable for downstream tasks. Lightweight decoders with fewer than 0.5M parameters can recover task-relevant signals from compressed latents, and generative models trained in the latent space produce outputs that remain structurally consistent across modalities. These results indicate that geometry-aligned latent design enables compact representations without sacrificing downstream utility.

1 INTRODUCTION

Pretrained Vision Transformers (Dosovitskiy et al., 2021) encode rich semantic and geometric information that is widely reused across tasks such as depth estimation, 3D reconstruction, and image generation (Ranftl et al., 2021; Wang et al., 2024; Radford et al., 2021). While these high-dimensional representations enable strong transfer, they are expensive to store, transmit, and process in downstream pipelines. This raises a fundamental question: *how can pretrained Transformer representations be compressed without sacrificing their downstream utility?*

A common approach is to learn compact representations using Variational Autoencoder (Kingma & Welling, 2013) with Gaussian latent spaces. We argue this default choice is poorly aligned with the structure of Transformer features. Due to Layer Normalization (Ba et al., 2016) and attention mechanisms (Vaswani et al., 2017), these features are approximately normalized and exhibit strong directional structure, where semantic information is encoded through angular relationships rather than Euclidean magnitude, a phenomenon also observed in representation learning more broadly (Wang & Isola, 2020). A Gaussian bottleneck forces such data into a dense Euclidean volume and, through its KL term, biases representations toward regions the data rarely occupies. Under strong compression, this manifests as posterior collapse, dead latent dimensions, and degradation on downstream tasks. This suggests that compression quality depends not only on capacity, but also on *how well the latent space matches the structure of the data*.

We address this with **S^2VAE**, a geometry-aware VAE that replaces the Gaussian bottleneck with a *product of Power Spherical distributions* (De Cao & Aziz, 2020). The product construction is

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.

essential: a single high-dimensional sphere suffers vanishing surface area and KL collapse, failure modes that do not arise in the low-dimensional settings where spherical VAEs were originally studied. Decomposing the latent into multiple low-dimensional spheres restores numerical stability and introduces a compositional inductive bias that lets distinct geometric attributes route to different sub-manifolds.

Our central empirical claim is that *latent geometry, rather than capacity, becomes the limiting factor under aggressive compression*. We evaluate $\mathbf{S^2}$VAE across four Transformer backbones spanning distinct training objectives and modalities: DINOv2 (Oquab et al., 2024), CLIP (Radford et al., 2021), DUSt3R (Wang et al., 2024), VGGT (Wang et al., 2025), and PAGE-4D (Zhou et al., 2025). Across all models, the spherical bottleneck consistently outperforms a matched Gaussian baseline, with the performance gap widening as compression increases, reaching up to $512\times$. Beyond reconstruction fidelity, the learned latent remains directly usable: lightweight decoders recover task-relevant signals from compressed representations, and generative models trained in the latent space produce outputs that remain structurally consistent across modalities.

Contributions.

1. We identify and empirically characterize a topological mismatch between Gaussian latents and the hyperspherical geometry of normalized Transformer features, and show that the resulting failure modes, posterior collapse, dead dimensions, semantic drift, sharpen precisely under the aggressive compression regimes that matter in practice.
2. We propose **$\mathbf{S^2}$VAE**, a geometry-aligned VAE built on a product of Power Spherical distributions. The product structure is what makes spherical latent modeling viable in the high-dimensional, LayerNorm-constrained regime of pretrained Transformers, a setting not addressed by prior spherical VAE work.
3. We demonstrate that $\mathbf{S^2}$VAE outperforms Gaussian baselines across four Transformer backbones, including DINOv2, CLIP, DUSt3R, VGGT, PAGE-4D, and the RAE ViT decoder, with the advantage growing as compression tightens, establishing geometry-alignment rather than capacity as the binding constraint.
4. We show that the compressed latent is directly usable for downstream tasks, supporting lightweight prediction and generative modeling without requiring access to the original feature space.

2 Related Work

Latent Variable Models for Non-Euclidean Data A line of work extends VAEs beyond Euclidean latents to better match the geometry of the data. Davidson et al. (2018) introduced the Hyperspherical VAE based on the von Mises–Fisher (vMF) distribution, and De Cao & Aziz (2020) proposed the Power Spherical distribution, which removes the rejection-sampling step and the Bessel-function dependence of vMF. Both works explicitly note numerical instability of these distributions in high dimensions and report experiments only on low-dimensional latents (typically below 40, and at most 64). Our product-of-spheres formulation removes this scaling barrier by partitioning the latent into many independent low-dimensional sub-spheres, allowing us to operate directly on Transformer feature spaces of up to $8{,}192$ dimensions, a regime where prior hyperspherical VAEs are not applicable. Beyond spheres, Falorsi et al. (2018) examine manifold-valued latents on Lie groups and show that aligning the latent manifold with the data structure materially affects the learned representation. Finally, Lucas et al. (2019) analyze posterior collapse, a failure mode that we identify as particularly acute for Gaussian bottlenecks applied to LayerNorm-constrained features (Section 4).

Geometry of Transformer Features Several studies of Transformer representations report that semantic content is carried primarily by direction rather than magnitude. Li et al. (2020) show that pretrained language models induce anisotropic embedding spaces concentrated along a small number of directions, and Ethayarajh (2019) finds that in certain layers the cosine similarity between embeddings of unrelated tokens approaches one. Building on this directional view, Meng et al. (2019) learn text embeddings directly in spherical space. We extend this perspective from analysis to latent design: if Transformer features are angularly organized, the latent that compresses them should be as well.

2.1 Encoding Transformer feature spaces

The Representation Autoencoder (Zheng et al., 2025a) reconstructs RGB pixels from encoded ViT features for generative modeling, but targets appearance rather than geometry and inherits a Gaussian latent without examining whether one is appropriate. Yu et al. (2025) take a complementary view, using ViT representations as alignment targets for diffusion transformers; this motivates the need for compact ViT-feature latents but does not learn one. A separate line of work maps inputs to discrete latents through vector quantization (VQ-VAE (van den Oord et al., 2017) and its descendants), which replaces the geometric questions we study with a combinatorial objective over a learned codebook. To our knowledge, no prior work learns a continuous VAE-style encoding of a ViT feature space and evaluates it on geometry-sensitive downstream tasks. We close this gap and use the resulting framework to ask which latent geometry best fits ViT features.

Visual Geometry and World Models We evaluate on two geometry-centric ViT backbones whose hidden states explicitly encode 3D structure: VGGT (Wang et al., 2025), which is trained to recover camera motion, depth, and point-level geometry, and DUSt3R (Wang et al., 2024), which performs feed-forward stereo reconstruction by regressing global pointmaps from image pairs through its CroCo (Weinzaepfel et al., 2022) backbone. Extensions to dynamic scenes such as PAGE-4D (Zhou et al., 2025) reinforce the broader need for compact, geometry-faithful latents over this family of models.

3 Preliminaries

Vision Transformer feature spaces A Vision Transformer (ViT) (Dosovitskiy et al., 2021) processes a sequence of patch tokens through stacked self-attention and feed-forward blocks, each preceded by Layer Normalization (Ba et al., 2016). The intermediate hidden states at layer ℓ form a tokenized feature map $\mathbf{x}^{(\ell)} \in \mathbb{R}^{T\times D}$ where T is the number of tokens and D is the per-token feature dimension (typically 768–2048 for modern backbones). Different ViT families specialize these features for different objectives, including self-supervised representation learning (DINOv2 (Oquab et al., 2024)), language alignment (CLIP (Radford et al., 2021)), 3D reconstruction (DUSt3R (Wang et al., 2024), VGGT (Wang et al., 2025)), and beyond. Across all of these families, two structural properties hold by construction: features are high-dimensional, and Layer Normalization concentrates them on approximately hyperspherical shells. We treat these multi-layer ViT features as the input signal to encode, with the per-backbone task heads consuming the reconstructions downstream.

Variational Autoencoders (VAEs) VAEs (Kingma & Welling, 2013) are a class of generative models that learn a compressed latent representation $\mathbf{z}$ of an input signal $\mathbf{x}$ by approximating the posterior distribution $p(\mathbf{z}|\mathbf{x})$. The architecture consists of an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ which maps the input to the parameters of a latent distribution, and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$ which reconstructs the input from a sampled latent vector. The model is trained by maximizing the Evidence Lower Bound (ELBO), which consists of a reconstruction term and a regularization term based on the KL Divergence between the approximate posterior and a prior $p(\mathbf{z})$.

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}))\,. \tag{1}$$

The standard choice is a multivariate Gaussian prior $p(\mathbf{z}) = \mathcal{N}(0, I)$, which admits a simple reparameterization $\mathbf{z} = \mu + \sigma \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$. However, the variational framework itself is not tied to Euclidean geometry: the latent manifold and prior $p(\mathbf{z})$ together determine the model's inductive bias. When the input data instead lies on a non-Euclidean manifold such as a sphere, the Gaussian prior introduces a structural mismatch that hyperspherical latent distributions can resolve.

4 Diagnosis: Why Gaussian Bottlenecks Misfit Transformer Features

Before describing our method, we establish empirically that Transformer features have an approximately spherical structure and explain why a Gaussian latent prior is therefore a poor fit.

4.1 TRANSFORMER FEATURES LIE ON HYPERSPHERICAL SHELLS

Layer Normalization (Ba et al., 2016), applied at almost every Transformer block, subtracts the per-token mean and divides by the per-token standard deviation. With a numerically stable ϵ in the denominator, its output lies on a thin shell rather than exactly on a sphere; we bound the shell thickness in Appendix B.2. Dot-product attention, which acts on inner products, then operates almost entirely on the angular component of these features.

To verify that this approximation holds in practice, we measure the coefficient of variation (CV) of feature norms across layers of two pretrained backbones (VGGT and DINOv2-Base) and a randomly initialized DINOv2-Base, on $1{,}000$ test samples. Across all three backbones and all layers the CV stays below 0.15, concentrated near 0.06–0.08 for most layers (VGGT first/middle/last: $0.07/0.08/0.13$; DINOv2-Base pretrained: $0.08/0.07/0.06$; DINOv2-Base randomly initialized: $0.09/0.07/0.06$). The randomly initialized DINOv2 follows essentially the same pattern as the pretrained one, indicating that the spherical concentration is induced by Layer Normalization itself rather than by training, and is therefore a structural property we can rely on when designing the encoder. The shells also sit at radii that depend strongly on depth (VGGT norms grow from ~ 6 at shallow layers to ~ 100 at the deepest layers), whereas a unit Gaussian $\mathcal{N}(0, I)$ concentrates at $\sqrt{D}$, mis-scaling the prior relative to deep-layer features.

4.2 GEOMETRIC TENSION IN GAUSSIAN BOTTLENECKS

Having established that Transformer features concentrate on a thin hyperspherical shell, we explain why a Gaussian prior mismatches them; we confirm the most concrete consequence (posterior collapse) empirically in Section 7.

Topological mismatch and KL pressure A Gaussian prior $\mathcal{N}(0, I)$ assumes the data fills a solid Euclidean volume in $\mathbb{R}^D$, whereas Layer Normalization confines features to an approximate hyperspherical shell. Mapping a sphere into a ball forces the encoder to spend latent capacity ignoring a radial dimension that carries essentially no information. The Gaussian KL term, $\mathcal{L}_{\text{KL}} = -\frac{1}{2}\sum_j (1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2)$, makes this worse: it pulls each posterior mean toward the origin while the reconstruction loss pulls it onto a shell of radius $R \approx \sqrt{D}$. The encoder therefore faces a direct conflict, and the active-dimension analysis in Section 7 shows the empirical outcome: a 128-dimensional Gaussian bottleneck uses only 71% of its dimensions while a matched product-of-spheres bottleneck uses all of them. A spherical latent removes the conflict by construction because there is no origin to collapse toward.

Semantic identity is angular Attention scores depend on inner products, so two ViT tokens that point in similar directions are functionally similar regardless of magnitude. A Gaussian latent conflates angle and magnitude in its Euclidean geometry, so two latents with high cosine similarity but different magnitudes can be far apart in ℓ_2 even though the downstream model would treat them as nearly identical. A spherical latent fixes the magnitude and makes Euclidean distance and cosine similarity equivalent, so the geometry the bottleneck optimizes is the geometry the downstream model actually consumes.

5 METHODOLOGY

Figure 1 presents an overview of the proposed architecture. Input frames are processed by a frozen Transformer backbone (VGGT in the illustration) to extract multi-layer features, which are individually normalized, concatenated, and projected into the VAE hidden space. After several self-attention blocks, the encoder predicts the μ and κ parameters of a product-of-spheres latent distribution; sampled latents are then decoded back into per-layer features that the original task heads can consume.

5.1 THE POWER SPHERICAL DISTRIBUTION

Multi-layer Transformer features $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$ are first concatenated and projected to a hidden dimension H via a sigmoid-gated projection $\mathbf{h} = \sigma(\mathbf{W}_g\mathbf{x} + \mathbf{b}_g) \odot (\mathbf{W}_h\mathbf{x} + \mathbf{b}_h)$ that lets the network suppress

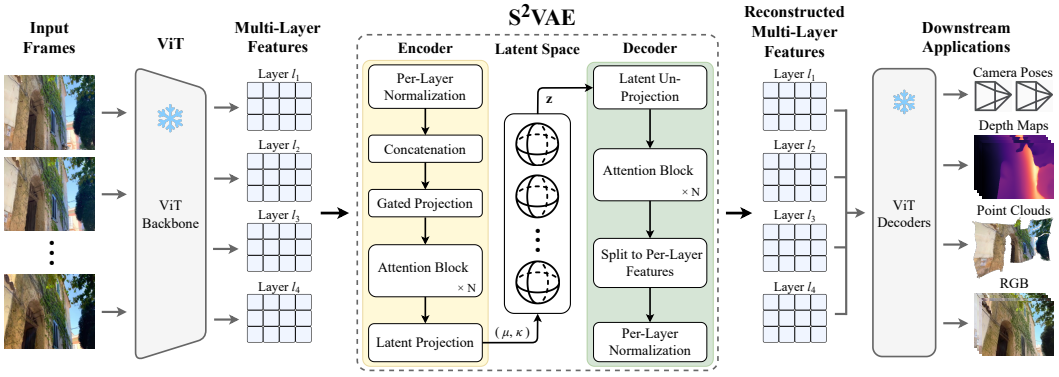


Figure 1: **Overview of the proposed S²VAE framework.** Input frames are first processed by a frozen ViT backbone to extract multi-layer feature representations. Features from each extracted layer are normalized independently, concatenated, and mapped to a shared hidden space via a gated projection. Then, the encoder applies a stack of self-attention blocks and predicts the parameters (μ, κ) of a hyperspherical latent distribution, modeled as a product of spheres, from which latent variables $\mathbf{z}$ are sampled using spherical reparameterization. These latents are subsequently un-projected, refined through additional self-attention blocks, and projected back to the original feature dimensionality before being split into reconstructed layer features. The reconstructed features can be consumed by the original decoder heads to recover geometry-centric outputs such as camera poses, depth maps, 3D point clouds, and RGB reconstructions.

redundant dimensions before the narrow latent. The von Mises-Fisher (vMF) distribution is the classical choice for spherical VAEs (Davidson et al., 2018). However, its density depends on the modified Bessel function $\mathcal{I}_\nu(\kappa)$, which lacks a closed-form inverse and complicates the reparameterization trick, typically requiring rejection sampling. We instead use the Power Spherical distribution (De Cao & Aziz, 2020), which admits a closed-form differentiable sampler via the *inverse transform method*. For a latent vector $\mathbf{z} \in \mathbb{S}^{d-1}$, the density is

$$q(\mathbf{z}; \mu, \kappa) = \frac{1}{C_d(\kappa)}(1 + \mu^\top \mathbf{z})^\kappa, \tag{2}$$

where $\mu \in \mathbb{S}^{d-1}$ is the location (mean) parameter, $\kappa > 0$ is the concentration (inverse variance), and the normalization constant is

$$C_d(\kappa) = 2^{\alpha+\beta}\pi^\beta \frac{\Gamma(\alpha)}{\Gamma(\alpha+\beta)}, \tag{3}$$

with $\alpha = \kappa + \frac{d-1}{2}$ and $\beta = \frac{d-1}{2}$.

Reparameterized sampling To sample $\mathbf{z} \sim \text{PowerSpherical}(\mu, \kappa)$, we draw a scalar $y \sim \text{Beta}(\alpha, \beta)$, transform it into $t = 2y - 1$ along the axis of μ, and sample $\mathbf{v}$ uniformly from $\mathbb{S}^{d-2}$. The intermediate vector $\mathbf{y}' = [t; \mathbf{v}\sqrt{1-t^2}]$ is a sample centered at the north pole $\mathbf{e}_1 = [1, 0, \ldots, 0]^\top$; we align it with μ by a Householder reflection:

$$\mathbf{z} = \left(\mathbf{I} - 2\mathbf{u}\mathbf{u}^\top\right)\mathbf{y}', \qquad \mathbf{u} = \frac{\mathbf{e}_1 - \mu}{\|\mathbf{e}_1 - \mu\|}. \tag{4}$$

When $\mu \approx \mathbf{e}_1$ the denominator $\|\mathbf{e}_1 - \mu\|$ vanishes; we follow standard practice and reflect to the antipodal pole $-\mathbf{e}_1$ in this case to avoid catastrophic cancellation. Gradients flow through μ and κ without numerical approximations of Bessel functions.

5.2 Product of Spheres for High Dimensions

A challenge arises when applying spherical distributions to high-dimensional latents ($d > 20$): the surface area of the $(d-1)$-sphere, $S_{d-1} = \frac{2\pi^{d/2}}{\Gamma(d/2)}$, vanishes exponentially as d grows. This leads to numerical instability in the KL divergence, and a form of posterior collapse in which the concentration κ fails to converge.

To represent a total latent dimension D, we instead define the latent manifold as a product of spheres,

$$\mathcal{M} = \prod_{i=1}^{N} \mathbb{S}^{d'-1}, \qquad \text{with } N \cdot d' = D. \tag{5}$$

The encoder predicts N independent pairs (μ_i, κ_i) where each predicted μ_i is L2-normalized so that $\|\mu_i\|_2 = 1$ and lies on $\mathbb{S}^{d'-1}$, and the total latent is the concatenation $\mathbf{z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$. Beyond numerical stability, this product structure provides a compositional inductive bias: by partitioning the latent into N independent sub-manifolds, the model can express partial semantic overlaps in which tokens share certain attributes in some sub-spheres while remaining distinct in others. Appendix M shows that this bias is realized in practice: different sub-spheres specialize for camera, depth, and point-cloud information without any direct supervision.

There is a tradeoff between the number of spheres and the dimensionality of each one. Lower-dimensional spheres are numerically more stable and easier to train, but a product of low-dimensional spheres has a different topology from a single high-dimensional sphere (for example, $S^1 \times S^1 \cong T^2$ is not homeomorphic to S^2). As d' grows, training stability deteriorates. As a balance, we use $d' \in \{4, 8\}$ with $N = D/d'$, which works well empirically.

KL divergence With a uniform prior $\mathcal{U}(\mathbb{S}^{d'-1})$ on each sphere, the total KL divergence is the sum of the per-sphere divergences. For a single sphere it is the gap between the negative entropy of the learned distribution and the entropy of the uniform prior:

$$D_{\mathrm{KL}}(q \,\|\, p) = -H(q) + \log\left(\frac{2\pi^{d'/2}}{\Gamma(d'/2)}\right), \tag{6}$$

where the entropy is computed from the log-normalizer and the digamma function ψ:

$$H(q) = \log C_{d'}(\kappa) - \kappa\left(\log 2 + \psi(\alpha) - \psi(\alpha+\beta)\right). \tag{7}$$

6 Experimental Setup

Transformer Backbones We evaluate S^{2}VAE across five Vision Transformer (ViT) backbones that span distinct training objectives and output modalities. VGGT (Wang et al., 2025) is a geometry-grounded transformer trained to predict camera poses, depth maps, and point clouds from video sequences. PAGE-4D (Zhou et al., 2025) is an extension of VGGT to dynamic scenes. DINOv2 (Oquab et al., 2024) is a self-supervised visual representation model; we use both Base and Large variants and evaluate downstream depth estimation via standard probing protocols. DUSt3R (Wang et al., 2024), built on the CroCo backbone, performs feed-forward 3D reconstruction from image pairs by predicting dense pointmaps and confidence maps. We also take the ViT decoder trained by the RAE model (Zheng et al., 2025a), and apply our VAE for image reconstruction. Finally, we use the CLIP (Radford et al., 2021) ViT-L/14 text encoder, evaluating reconstruction quality through downstream text-to-image generation using Stable Diffusion.

Data For geometry-centric backbones (VGGT and DUSt3R), we train and evaluate on RealCam-Vid (Zheng et al., 2025b), a curated dataset derived from RealEstate10K (Zhou et al., 2018), DL3DV-10K (Ling et al., 2024), and MiraData (Ju et al., 2024). It contains diverse indoor/outdoor scenes with both static and dynamic content, along with realistic camera motion. For DINOv2, we use the same RealCam-Vid frames but treat them independently (no temporal modeling), following standard depth probing setups. For CLIP, we use the captions provided with RealCam-Vid for training. We additionally evaluate generalization using out-of-domain prompts generated by an LLM, as described in Appendix D.1.

Metrics For depth we report AbsRel and threshold accuracy δ_1 (fraction of pixels whose predicted depth is within a factor of 1.25 of ground truth), plus SqRel and $\mathrm{RMSE}_{\log}$ on the DINO models. For camera pose we report Absolute Trajectory Error (ATE) and AUC@30 (area under the pose-accuracy curve up to a 30° rotation threshold). For point cloud reconstruction, we report Chamfer Distance (CD), computed on the full point set without confidence filtering, subsampling, or alignment. The single-sphere ablation in Appendix J reports CD scores obtained from confidence-filtered point sets and should, therefore, be compared only within that table.

6.1 Training Details

All our large VAEs are trained on either 4×Nvidia A40s (46 GB) or 4×Nvidia A100 (64 GB). For the VGGT-based models, we trained for 1 day. For the DINO models, we trained for 12 hours. We use the same hyperparameters across all models for consistency. We use 4 input frames at a time, at a resolution of 224×224 (this primarily matters for the VGGT tests, as the DINO models will process the frames independently). Regardless of the input dimensionality of the features, we project the concatenated features of all dimensions into a 1024-dimensional hidden space (reduced to 512 for CLIP), and use a 128-dimensional spherical bottleneck, as a product of 16 spheres each having dimension 8. The size of the VAEs range from 400M-500M parameters, depending on the input feature dimensionality (or ∼105M for CLIP due to reduced hidden size).

Our VAE consists of 12 sets of attention layers in the encoder and decoder, and 16 attention heads per layer. We use bf16 mixed precision for all trainings. For optimization, we use the AdamW optimizer (Loshchilov & Hutter, 2019) optimizer with a weight decay of 0.05, and a linear warmup to a learning rate of $1e-4$ and with cosine annealing decay. We also default to the initial κ value of 30.0 as a good starting point, but allow the model to learn the correct predictions for κ.

6.2 Loss Functions

Our training losses can be split into two categories: the key losses to make the VAE work, and the task-specific losses. Here, we emphasize one key detail: for our training, we consider the original decoder head's outputs to be the ground truth. Since we are freezing all of the underlying models and decoder heads, comparing against a real ground truth leads to 2 potential sources of error: (1) reconstruction error caused by the VAE incorrectly reconstructing the features, and (2) inaccuracies caused by the decoder head itself. To remove this second error, we just use the original model outputs as the ground truth.

Main Loss Functions For our feature reconstruction losses, we use a combination of MSE and cosine similarity loss. Specifically, given the input features $\mathbf{x}$ and the reconstructed features $\hat{\mathbf{x}}$, our feature reconstruction loss is given $\mathcal{L}_{\text{FR}} = \text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) + 0.25 * \text{SIM}(\mathbf{x}, \hat{\mathbf{x}})$.

While this works decently well by itself, we find that it is also important to preserve the relational structure between the different features. For this purpose, we define a loss on the Gram matrices of the feature layers. If $\mathbf{X}$ is the L_2-normalized features of a given layer, we define the Gram matrix of these features as $\mathbf{G} = \mathbf{X}\mathbf{X}^\top$. We then compute cosine distance between the Gram matrices of the original and reconstructed features. This forces the VAE to maintain inter-feature relation patterns, helping to preserve the overall structure of the scene even if the absolute feature values are compressed.

Furthermore, to counteract the tendency of variational bottlenecks to produce over-smoothed representations, we add two additional regularization losses. First, we use a variance-preserving loss, penalizing the L_1 distance between the variance of the input features and reconstructed features. Additionally, we apply a similar loss to the norms of the input and reconstructed features.

Finally, as described above, we apply the KL divergence loss across the product of spheres.

Task-Specific Losses For the task-specific losses, we use the original losses found in each work. For clarity, we briefly review the relevant losses here.

For VGGT, we apply the camera, depth, and point cloud losses found in the original model. The camera loss compares the predicted cameras $\hat{\mathbf{g}}_i$ with the original cameras $\mathbf{g}_i$ using a Huber loss: $\mathcal{L}_{\text{Cam}} = \sum_{i=1}^{N} ||\hat{\mathbf{g}}_i - \mathbf{g}_i||_\epsilon$.

Meanwhile, the depth loss implements the aleatoric-uncertainty loss found in both VGGT (Wang et al., 2025) and DUSt3R (Wang et al., 2024). In this setup, the model produces both the depth map and an uncertainty map $\hat{\Sigma}_i^D$. The depth loss is computed as the difference between the predicted depth $\hat{D}_i$ and the original depth D_i, weighted by the uncertainty map. Additionally, VGGT also applies a gradient term, which we use too. The final depth loss is given by:

$$\mathcal{L}_{\text{Depth}} = \sum_{i=1}^{N} ||\hat{\Sigma}_i^D \odot (\hat{D}_i - D_i)|| + ||\hat{\Sigma}_i^D \odot (\nabla \hat{D}_i - \nabla D_i)|| - \alpha \log \hat{\Sigma}_i^D \tag{8}$$

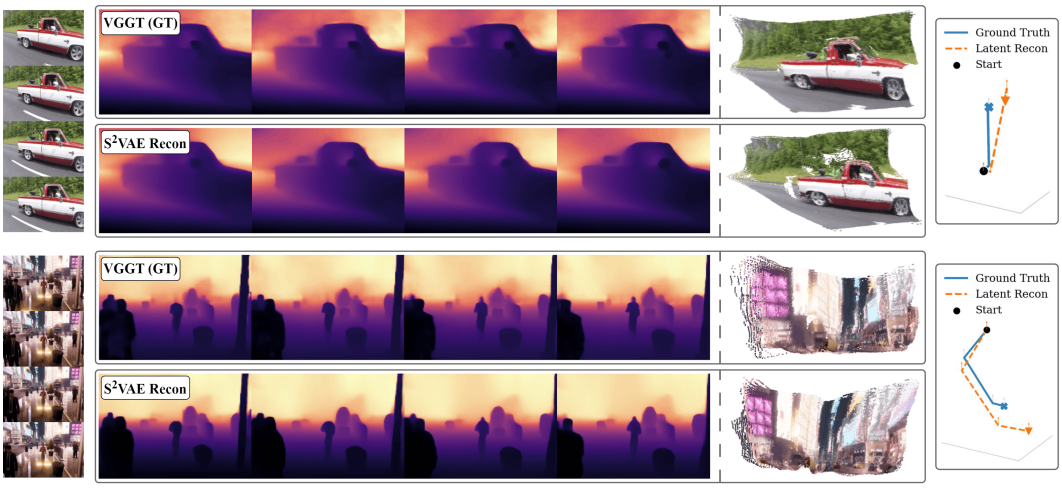


Figure 2: **Qualitative reconstruction of geometric task outputs from compressed latent representations.**

Table 1: **Quantitative depth estimation results for different ViT backbones using the proposed VAE framework.** Lower AbsRel and higher δ_1 scores indicate better performance.

Model	AbsRel	δ_1	SqRel	$\text{RMSE}_{\log}$
DINOv2-B	0.035	0.99	2.8e-4	0.05
DINOv2-L	0.037	1	3.4e-4	0.04

The pointmap is defined similarly, using a pointmap uncertainty $\hat{\Sigma}_i^P$.

7 Results

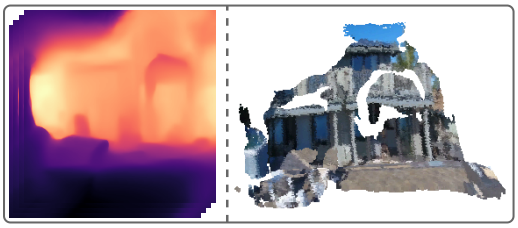
Reconstruction across backbones We apply the same recipe to four ViT backbones spanning distinct training objectives and modalities: VGGT (Wang et al., 2025) (1.2B params; depth, camera, point cloud), PAGE-4D (Zhou et al., 2025) (1.2B; depth, camera, point cloud), DINOv2-B/L (Oquab et al., 2024) (86M/300M; depth), DUSt3R/CroCo (Wang et al., 2024) (600M; depth and point cloud), RAE ViT (Zheng et al., 2025a) (449M; Images) and the CLIP ViT-L/14 text encoder (Radford et al., 2021) (123M; text-to-image via Stable Diffusion v1). Figure 2 shows qualitative reconstructions of the geometric task outputs (depth map, camera trajectory, point cloud), produced by passing the encoder-decoder reconstructions through each backbone's task heads. Reconstructions remain faithful across backbones with few visible artifacts. Quantitative depth-estimation results on the DINOv2 family appear in Table 1; full DUSt3R and CLIP results are in Appendix D.

Matched Gaussian comparison To isolate the effect of the latent bottleneck, we compare S^2VAE against a matched Gaussian VAE using the same encoder and decoder, latent dimension ($D = 128$), hidden dimension, training objective, optimizer, and training schedule, leaving the bottleneck distribution as the sole difference. Table 2 reports results over three independent training seeds. We separately study a single high-dimensional Power Spherical bottleneck in Appendix J. We additionally compare against normalized Gaussian and discrete quantization-based bottlenecks in Appendix N.

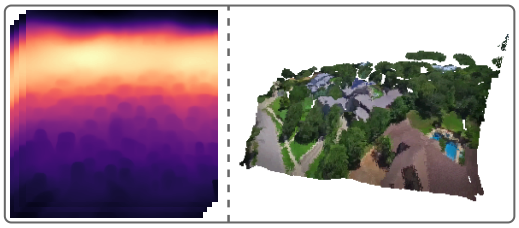
Direct evidence of posterior collapse Our diagnosis (Section 4.2) predicts that a Gaussian bottleneck will fail to use all its latent dimensions because KL pulls means toward the origin while reconstruction pushes them outward. We test this by computing per-dimension variance over 1,000 validation samples. To compare fairly despite differing prior variances (Gaussian: 1.0; uniform $\mathbb{S}^{d'-1}$: $1/d' = 0.125$ for $d' = 8$), we count a dimension as *active* when its variance exceeds half its prior expectation. Under this prior-relative criterion, the Gaussian bottleneck activates 91 of 128 dimensions (71%) while the product-of-spheres bottleneck activates all 128. Sphere variances are tightly clustered (0.10–0.15, near the prior mean 0.125), while Gaussian variances span more than

Table 2: **S^{2}VAE consistently outperforms the matched Gaussian bottleneck across three independent training seeds on VGGT.** Both models share the same encoder/decoder, total latent dimension ($D = 128$), hidden dimension, training objective, optimizer, training schedule, and data; only the latent distribution differs.

Bottleneck	AbsRel (↓)	δ_1 (↑)	CD (↓)	ATE (↓)	AUC@30 (↑)
S^{2}VAE (ours)	**0.027** ±0.002	**0.984** ±0.002	**0.014** ±0.001	**0.0026** ±0.0003	**99.24** ±0.08
Gaussian	0.040 ±0.001	0.979 ±0.005	0.029 ±0.003	0.0041 ±0.0009	98.97 ±0.07



A luxurious two-story home with a beige and gray exterior, showcasing a spacious patio with stone tiles, white adirondack chairs, ...



An aerial view of a luxurious neighborhood with a blend of modern and traditional architecture, surrounded by lush greenery ...

Figure 3: **Latent-space diffusion on S^{2}VAE.** DiT (Peebles & Xie, 2023) samples decoded through the VGGT depth and 3D point cloud heads, with consistency across the two modalities.

an order of magnitude (below threshold on inactive dimensions, up to 3.5 on overused ones). The Gaussian collapse mode is absent on a sphere because there is no origin to collapse toward.

8 Applications

The S^{2}VAE latent enables two downstream uses the full-precision pipeline does not.

Lightweight probing Two-layer probes with under 500k parameters operate directly on S^{2}VAE latents to recover depth, sidestepping the full VGGT decoder: a per-token MLP reaches AbsRel 0.065 and a small spatial CNN reaches 0.049, close to the full decoder (Appendix G). The CNN probe is roughly 100× smaller than VGGT's DPT depth head and operates on a 64× compressed latent, evidence that geometric information is organized accessibly enough for tiny decoders to read off directly.

Latent-space diffusion Because S^{2}VAE encodes joint VGGT features, samples from a generative model on its latents decode into mutually consistent depth, camera, and point-cloud outputs by construction. We train a DiT (Peebles & Xie, 2023) on S^{2}VAE latents with a rectified-flow (Liu et al., 2023) loss; Figure 3 shows varied generated depth-map samples, and decoding a single generated latent through both the depth and point-cloud heads yields outputs that agree on object placement without any explicit cross-task consistency loss during training. This coherence cannot be attributed to decoder capacity alone and is a capability pixel-space generators and Gaussian-bottleneck VAEs cannot provide.

9 Conclusion

Layer Normalization is a geometric commitment: once features live on a sphere, the latent that compresses them should too. Inspired by this, we proposed a product of Power Spherical distributions, numerically stable in high dimensions, which gives consistent gains over Gaussian baselines across DINOv2, CLIP, DUSt3R, VGGT, and PAGE-4D, and yields a latent that supports both lightweight prediction and jointly consistent multi-task generation. We expect the principle to generalize wherever normalization shapes the representation, with latent geometry treated as a design choice driven by the data.

AI USE STATEMENT

We used generative AI tools to help with polishing some writing and assisting with code. We did not use generative AI for developing research ideas or analyzing results. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including all text, claims, and results.

REPRODUCIBILITY STATEMENT

We attempted to provide as many details as possible regarding our implementation and hyperparameters. We plan to release the code at the end of the review period.

REFERENCES

Moayed Haji Ali, Andrew Bond, Levent Karacan, Tolga Birdal, Erkut Erdem, Duygu Ceylan, and Aykut Erdem. Vidstyleode: Disentangled video editing via stylegan and neuralodes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7489–7500. IEEE, October 2023. doi: 10.1109/iccv51070.2023.00692.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Ramón Calvo-González and François Fleuret. Laminating representation autoencoders for efficient diffusion. *arXiv preprint arXiv:2602.04873*, 2026.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Nicola De Cao and Wilker Aziz. The power spherical distribution. *arXiv preprint arXiv:2006.04437*, 2020.

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 *International Conference on Learning Representations*, 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 *International Conference on Machine Learning*, pp. 12606–12633. PMLR, 2024.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 55–65, 2019.

Luca Falorsi, Pim De Haan, Tim R Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

Xuan Ju, Yiming Gao, Zhaoyang Zhang, Ziyang Yuan, Xintao Wang, Ailing Zeng, Yu Xiong, Qiang Xu, and Ying Shan. Miradata: A large-scale video dataset with long durations and structured captions. *Advances in Neural Information Processing Systems*, 37:48955–48970, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 9119–9130, 2020.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. Spherical text embedding. *Advances in neural information processing systems*, 32, 2019.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple, 2023. URL https://arxiv.org/abs/2309.15505.

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. *Transactions on Machine Learning Research Journal*, 2024.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2085–2094, 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 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 *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12179–12188, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20697–20709, 2024.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 2020.

Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. *Advances in Neural Information Processing Systems*, 35:3502–3516, 2022.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

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

Guangcong Zheng, Teng Li, Xianpan Zhou, and Xi Li. Realcam-vid: High-resolution video dataset with dynamic scenes and metric-scale camera movements. *arXiv preprint arXiv:2504.08212*, 2025b.

Kaichen Zhou, Yuhan Wang, Grace Chen, Xinhai Chang, Gaspard Beaudouin, Fangneng Zhan, Paul Pu Liang, and Mengyu Wang. Page-4d: Disentangled pose and geometry estimation for vggt-4d perception, 2025.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4): 1–12, 2018.

Jian Zhu, Mingkai Sheng, Zhangmin Huang, Jingfei Chang, Jinling Jiang, Jian Long, Cheng Luo, and Lei Liu. Clip multi-modal hashing for multimedia retrieval. In *International Conference on Multimedia Modeling*, pp. 195–205, 2025.

A SPHERICAL LATENT INTERPOLATION

We perform spherical linear interpolation between two latents $\mathbf{x}, \mathbf{y}$ on the sphere, $\text{Slerp}(\mathbf{x}, \mathbf{y}, t) = \frac{\sin((1-t)\Omega)}{\sin\Omega}\mathbf{x} + \frac{\sin(t\Omega)}{\sin\Omega}\mathbf{y}$ with $\Omega = \arccos(\mathbf{x} \cdot \mathbf{y})$. Figure 4 shows the predicted depth maps as we interpolate between two distinct scenes; the transition is smooth, suggesting the latent manifold is geometrically continuous even at very low KL weights (10^{-4}).

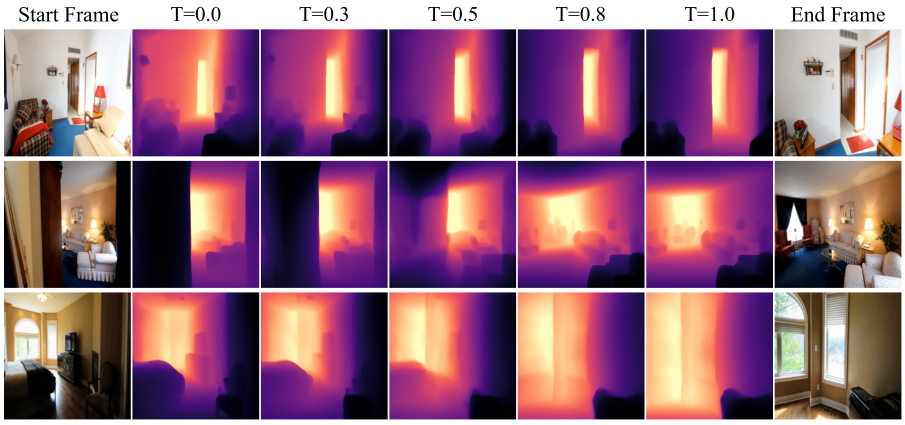


Figure 4: **Spherical interpolation between two distinct scenes produces smooth depth transitions along the spherical-linear path.**

B ADDITIONAL VGGT EXPERIMENTAL DETAILS

This appendix collects two short theoretical observations referenced in the main body. Per-layer CV-of-norms numbers and the active-dimension count are reported in the main body (Sections 4.1 and 7); here we record a bound on the shell thickness induced by Layer Normalization, and a brief note on how Gaussian bottlenecks induce exploding decoder Lipschitz constants.

B.1 DECODER LIPSCHITZ BLOW-UP FOR GAUSSIAN BOTTLENECKS

Layer Normalization (Appendix B.2) places features on a shell of radius $\approx \sqrt{d}$, so antipodal points x_1, x_2 on the shell satisfy $\|x_1 - x_2\|_2 \approx 2\sqrt{d}$. Under a Gaussian bottleneck, the *posterior means* $\mu_\phi(x_1), \mu_\phi(x_2)$ lie within a ball B_δ whose radius δ shrinks as the KL weight grows. Any decoder g that reconstructs x_1, x_2 to within reconstruction error ϵ from these collocated means must therefore satisfy a Lipschitz bound

$$L \;\geq\; \frac{2\sqrt{d}\,(1-\epsilon)}{\delta},$$

which diverges as $\delta \to 0$. The argument applies to posterior means rather than samples: the Gaussian sampling noise (variance $\to 1$ as KL pressure grows) does not concentrate, so a decoder that adopted such a large Lipschitz constant would amplify the sampling noise and fail to reconstruct distinct inputs. The standard resolution is therefore not a large-Lipschitz decoder but posterior collapse, which we measure directly in Section 7.

B.2 APPROXIMATE SPHERICAL PROPERTY OF LAYER NORMALIZATION

Layer Normalization with a numerical-stability term ϵ produces $\|y\|_2^2 = d/(1 + \epsilon/v)$, where v is the variance of the centered pre-normalization input across feature dimensions. Expanding to first order for small ϵ gives $\|y\|_2 \approx \sqrt{d}\,(1 - \epsilon/(2v))$. If the minimum variance across feature dimensions is bounded below by $v_{\min}$ with probability $1 - \delta$, then with the same probability the output lies on a shell of thickness approximately $\sqrt{d}\,\epsilon/(2v_{\min})$. Whenever the features have non-trivial variance across dimensions (which is the case throughout pretrained Transformers), the shell is therefore thin, justifying the spherical model.

C GENERALIZATION TO PAGE-4D

To probe how robustly the same recipe transfers to derivative backbones, we apply S^{2}VAE without modification to PAGE-4D (Zhou et al., 2025), a recent extension of VGGT to dynamic scenes. PAGE-4D shares the VGGT feature geometry (multi-layer LayerNormed tokens) but is trained on temporally varying inputs, so the encoder's job differs in practice even if the input format is identical. Figure 5 shows qualitative reconstructions of depth, camera trajectory, and point cloud for several test sequences. Reconstructions remain faithful with the same hyperparameters used for VGGT, suggesting the spherical bottleneck transfers cleanly across backbones in the same family without per-model tuning.

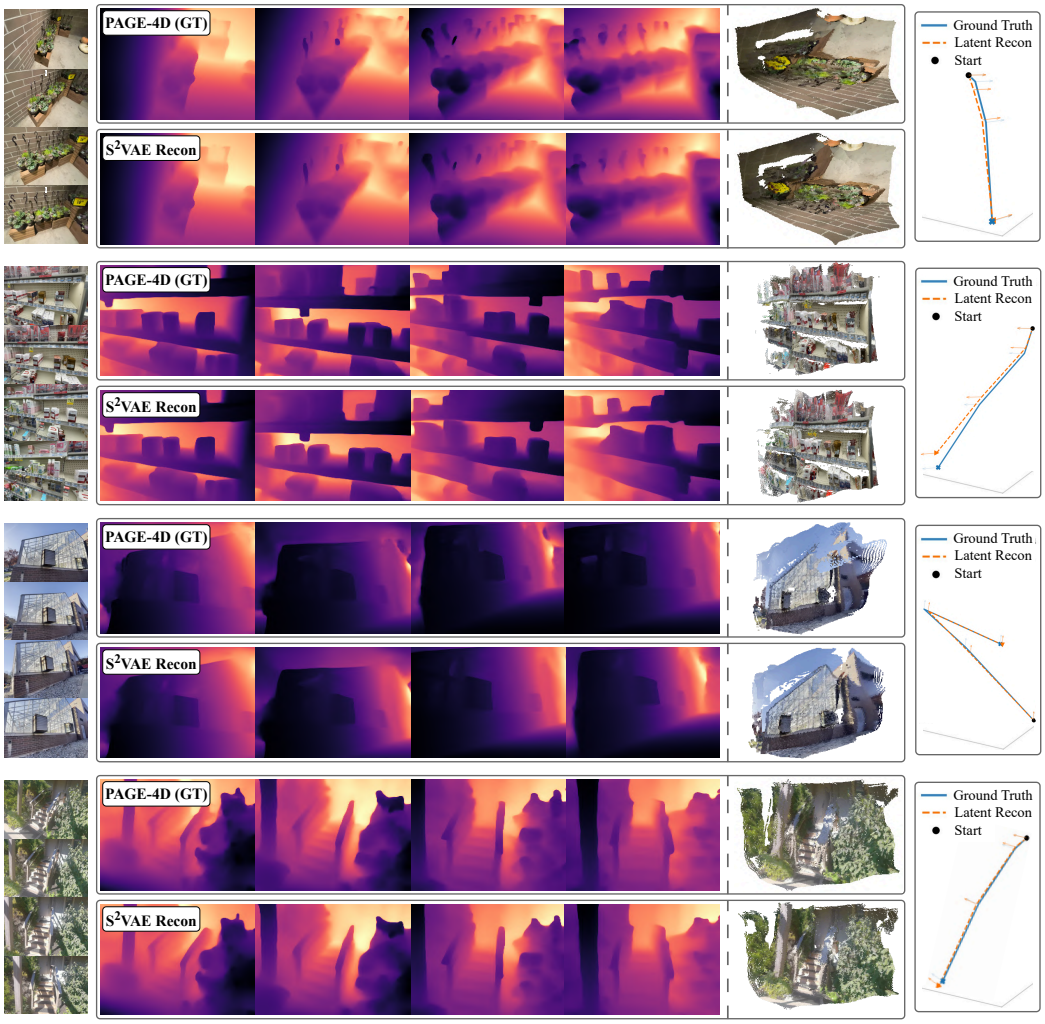


Figure 5: **Qualitative reconstructions of PAGE-4D (Zhou et al., 2025) features through S^{2}VAE.** The same architecture and hyperparameters used for VGGT recover PAGE-4D's depth, camera, and point-cloud outputs without modification.

D TESTS ON DIFFERENT TRANSFORMER BACKBONES

This appendix provides full results for the cross-architecture experiments summarized in Section 7.

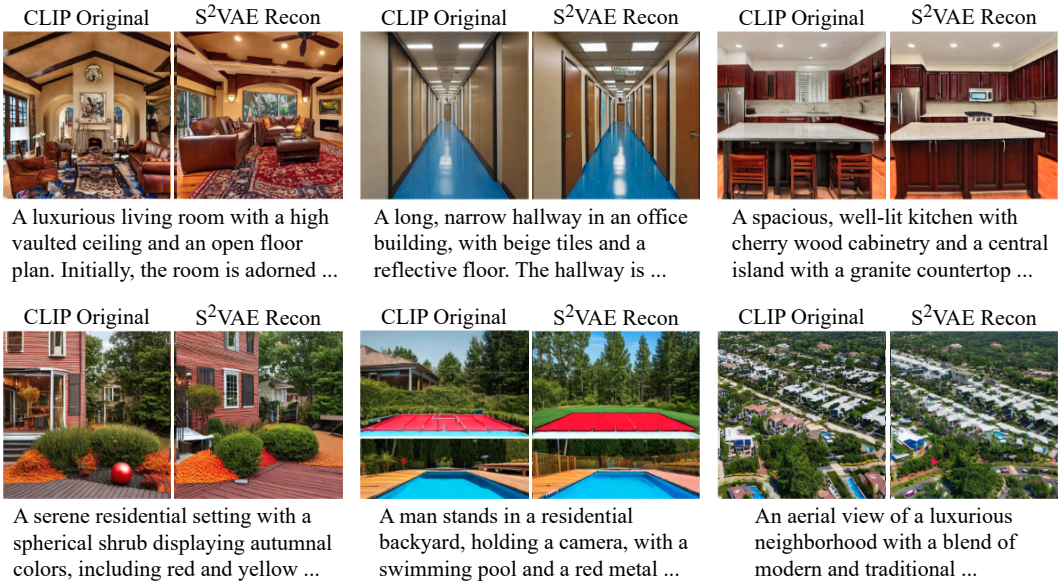


Figure 6: Images generated by the Stable Diffusion model using the original CLIP text features vs the features reconstructed using our VAE, using captions found in the test set of the RealCam-Vid dataset. We can see that all the key semantic details of the original images are always found in the reconstructed versions too, showing that our VAE is able to properly reconstruct all the relevant semantic information in the text features, even without any explicit task-specific losses.

D.1 CLIP TEXT ENCODER

CLIP (Radford et al., 2021) is a widely used text-image model that aligns image and text representations in a shared latent space using two separate encoders. The text encoder appears in a wide range of generative models (Patashnik et al., 2021; Ali et al., 2023; Rombach et al., 2022), as well as in zero-shot image classification, multi-modal retrieval (Zhu et al., 2025), and open-vocabulary segmentation (Kirillov et al., 2023).

We test how well S^{2}VAE reconstructs CLIP textual features and whether the reconstructions remain usable for text-to-image generation through Stable Diffusion (Rombach et al., 2022). We use Stable Diffusion v1, which relies solely on CLIP for text encoding; Stable Diffusion 3 (Esser et al., 2024) mixes T5 (Raffel et al., 2020) and CLIP, which would prevent isolating the effect of CLIP feature reconstruction.

We use the same architecture as for VGGT and DINOv2 except with hidden dimension 512 instead of 1024 (≈ 105M parameters), trained for 50 epochs on RealCam-Vid using the provided captions and only the main loss (no task-specific terms).

Figure 6 compares images generated by Stable Diffusion from the original CLIP features against those generated from reconstructed CLIP features. The reconstructions preserve the key semantic details across a diverse set of scenes. Within this dataset, the mean cosine similarity between original and reconstructed features is ≈ 0.94, indicating that almost all of the semantic information is preserved.

We also test on out-of-domain captions generated by an LLM. Semantic similarity between original and reconstructed images remains high, though fine-grained details occasionally differ. The mean cosine similarity drops modestly to ≈ 0.82, still high enough to preserve the principal content of the image.

D.2 RAE VIT DECODER

The Representation Autoencoder (RAE) (Zheng et al., 2025a) pairs a frozen ViT encoder with a learned ViT decoder that reconstructs RGB pixels from encoded features. Unlike VGGT, DUSt3R, or DINOv2, the downstream signal here is image appearance rather than 3D geometry, which lets



Figure 7: Several samples on randomly generated, out-of-distribution prompts. We can see that our model generalizes quite well, preserving the core semantics despite being trained only on RealCam-Vid samples.

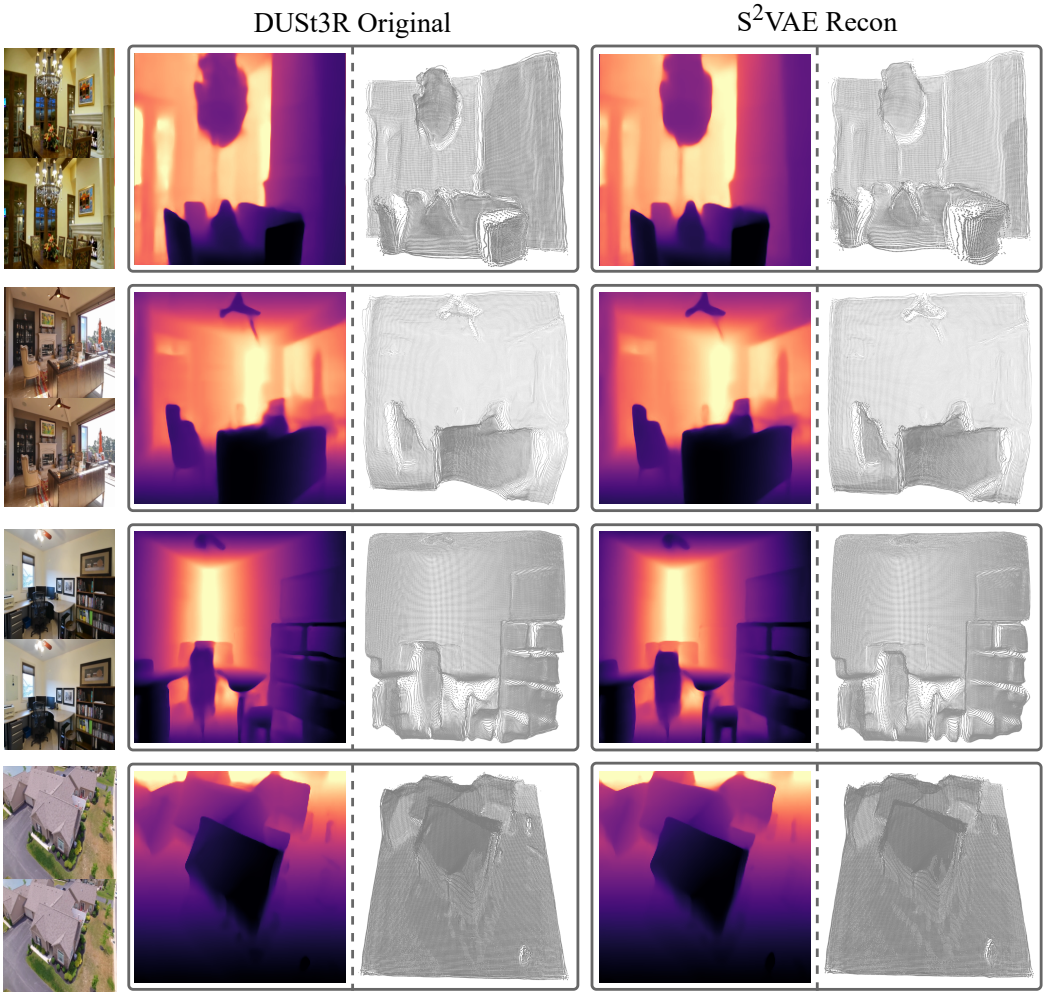


Figure 8: DUSt3R reconstructions on RealCam-Vid samples: original outputs (left) vs. outputs decoded from S^{2}VAE-encoded CroCo features (right). Depth maps and point clouds are preserved with high fidelity.



Figure 9: **Image reconstructions through the RAE (Zheng et al., 2025a) ViT decoder, using S^{2}VAE-encoded ViT features.** The reconstructed images closely match the originals, demonstrating that the spherical bottleneck preserves the information needed for high-fidelity pixel reconstruction in addition to geometric tasks.

Table 3: Comparisons between our proposed product of spheres bottleneck and the standard Gaussian bottleneck on DUSt3R depth map and point cloud reconstruction. Our bottleneck outperforms the Gaussian bottleneck on all metrics.

Bottleneck	**AbsRel** ($\downarrow$)	δ_1	**CD**
Product of Spheres (Ours)	**0.046**	**0.982**	**0.027**
Gaussian	0.060	0.961	0.034

us probe whether S^{2}VAE preserves enough information for high-fidelity pixel-level reconstruction. We encode the RAE encoder features with the same architecture and hyperparameters used for the geometric backbones, and decode them through RAE's pretrained ViT decoder without any finetuning. Figure 9 shows reconstructions on held-out images: the recovered images closely match the originals across diverse scenes, indicating that the spherical bottleneck is not specific to geometric features and transfers to a different output modality.

D.3 DUST3R Model (CroCo Backbone)

DUSt3R (Wang et al., 2024) is a popular model for stereo 3d reconstruction, which processes pairs of images and attempts to reconstruct global pointmaps of the scenes, as well as the confidence levels and camera information (for the second image relative to the first image). To do this, DUSt3R relies on the CroCo (Weinzaepfel et al., 2022) backbone, a self-supervised ViT trained on the masked image modeling objective to understand 3d geometry. Thus, this is again a very different training objective than DINO or VGGT.

To evaluate reconstruction on DUSt3R, we encode the features of its CroCo backbone. We pass a pair of images through CroCo, treat the two sets of features analogously to multi-layer VGGT features, and concatenate them along the feature dimension so that the VAE can use information from both views jointly.

We use the same setup as for CLIP on RealCam-Vid, training for 50 epochs (≈ 108M parameters). Comparisons between our method and the standard gaussian bottleneck can be found in Table 3.

E Sequence-length reduction

This appendix expands on the brief mention of sequence-length reduction in Section 7. All experiments in this appendix use the VGGT backbone. The experiments in the main body keep the number of tokens fixed and only reduce the feature dimensionality, but the sequence length can be reduced too. Following Calvo-González & Fleuret (2026), to go from N tokens per frame to $M < N$ tokens per frame, we randomly initialize and learn M register tokens. These tokens are prepended to the input sequence (duplicated across frames as needed), the model runs the same self-attention layers as

Table 4: **Sequence-length reduction on VGGT, varying the number of register tokens per frame.** Chamfer Distance is computed on the full (unfiltered) point set. Depth metrics improve steadily with more tokens, while point-cloud and camera metrics are nearly flat, suggesting the model prioritizes point-cloud and camera information even under aggressive sequence-length reduction.

Register tokens	**AbsRel** (↓)	$\boldsymbol{\delta_1}$ (↑)	**CD** (↓)	**ATE** (↓)	**AUC@30** (↑)
8	0.039	0.976	0.175	0.0008	0.997
16	0.043	0.970	0.173	0.0008	0.996
32	0.036	0.978	0.174	0.0009	0.996
64	0.035	0.976	0.183	0.0007	0.997
128	**0.029**	**0.986**	0.176	**0.0007**	0.996

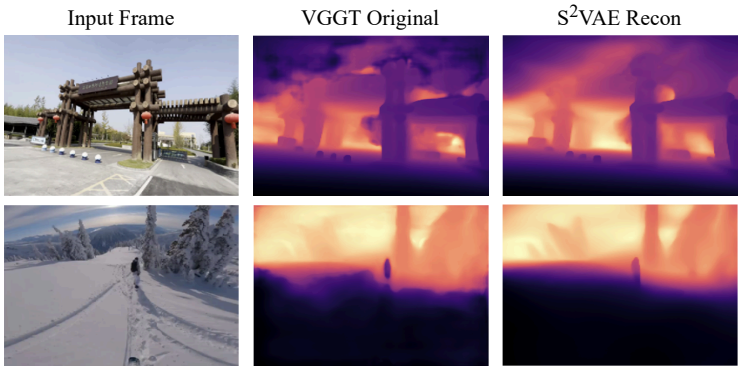


Figure 10: **Failure cases of S^{2}VAE under aggressive encoding.** In scenes with complex fine-grained textures, the reconstructed depth maps exhibit over-smoothing relative to the VGGT target. Global structure and scene layout remain correct; degradation is confined to fine-grained local detail.

before, and the bottleneck then takes only the M register tokens per frame, discarding the original input tokens.

On the decoder side, we learn another N register tokens and apply the same strategy: they are prepended to the latent sequence (with duplication if necessary), passed through the self-attention layers, and used as the reconstructed features. This reduces the latent footprint along both the feature and the sequence dimensions.

We use the same architecture as before, with latent dimension 128 and hidden dimension 1024 fixed, and vary the number of register tokens per frame. Table 4 reports the result. Even with only 8 register tokens per frame, all metrics remain close to the full-token setting, indicating that the model exploits substantial cross-token redundancy; the N register tokens on the decoder side also give the decoder additional capacity to recover per-token detail.

F Failure cases

The dominant failure mode of S^{2}VAE under aggressive encoding is over-smoothing of fine-grained textures. In scenes with dense vegetation, intricate architectural details, or low-contrast snow surfaces, the reconstructed depth maps lose high-frequency detail relative to the VGGT target (Figure 10). We attribute this to the information bottleneck itself: at $D = 128$, high-frequency geometric variation that is not consistently present across the training distribution cannot be fully preserved. Importantly, the global depth structure and scene layout remain correct in all cases; degradation is confined to fine-grained local detail.

G Lightweight probing details

This appendix expands on the lightweight-probing application in Section 8. We train two probes to convert S^{2}VAE latents directly into a depth map: a per-token two-layer MLP that maps each token

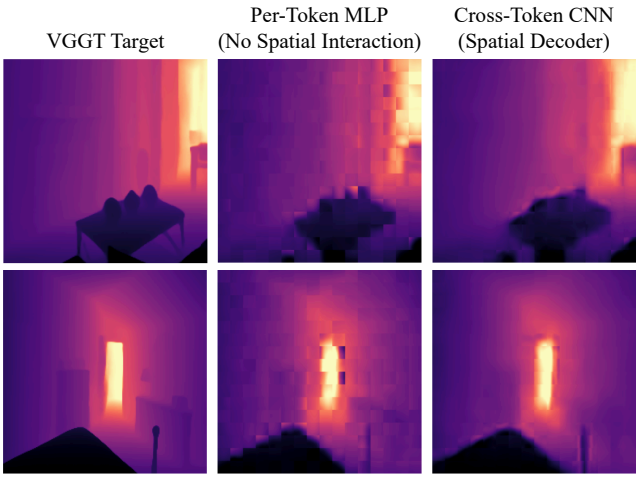


Figure 11: Lightweight probes operating directly on the S^{2}VAE latent space. Both the per-token two-layer MLP and the two-layer spatial CNN decoder ($<$ 500k parameters each) recover the key objects and relative distances of the scene. The MLP version shows grid artifacts because tokens do not share information across the patch boundary; the CNN smooths these out at the cost of some fine detail.

to a 16×16 depth patch, and a two-layer spatial (CNN) decoder operating on the token grid. Both contain under 500k parameters and do not interact with the VGGT decoder. On a 1000-video test set, the MLP reaches AbsRel 0.065 and the CNN reaches 0.049. The MLP exhibits grid artifacts from the lack of cross-token interaction; the CNN smooths these out but loses some high-frequency detail. In both cases the core scene structure and object placement are preserved, showing that even a weak probe extracts the meaningful depth signal from the latents.

H Task-specific loss details

For VGGT, we apply the camera, depth, and point-cloud losses from the original model. The camera loss is a Huber loss between predicted and ground-truth cameras: $\mathcal{L}_{\text{Cam}} = \sum_{i=1}^{N} \|\hat{\mathbf{g}}_i - \mathbf{g}_i\|_\epsilon$. The depth loss is the aleatoric loss of VGGT (Wang et al., 2025) and DUSt3R (Wang et al., 2024), in which the model produces both a depth map and a per-pixel *precision* (inverse-variance) map $\hat{\Sigma}_i^D$ that multiplies the error term and is regularized by $-\log \hat{\Sigma}_i^D$:

$$\begin{aligned}\mathcal{L}_{\text{Depth}} = \sum_{i=1}^{N} &\|\hat{\Sigma}_i^D \odot (\hat{D}_i - D_i)\| \\ &+ \|\hat{\Sigma}_i^D \odot (\nabla\hat{D}_i - \nabla D_i)\| - \alpha \log \hat{\Sigma}_i^D.\end{aligned} \tag{9}$$

The pointmap loss is defined analogously with a pointmap precision map $\hat{\Sigma}_i^P$.

I Auxiliary loss ablations

The training objective combines feature reconstruction with three auxiliary terms (Gram-matrix loss, per-token-norm regularizer, per-dimension-variance regularizer) and a gated input projection. To clarify how much of S^{2}VAE's downstream performance is attributable to the hyperspherical latent versus the auxiliary terms, we ablate each component on VGGT, training with all other components held fixed. Table 5 reports the result.

The Gram-matrix loss is the most impactful auxiliary component (removing it more than doubles AbsRel and increases CD by 50%), consistent with its role in preserving inter-token relational structure. The variance-regularization loss and gated projection contribute modestly. Camera-pose

Table 5: **Effect of removing each auxiliary loss / architectural component on the VGGT setting.** CD is computed on the full (unfiltered) point set. The hyperspherical bottleneck alone (no Gram loss row) recovers most but not all of S^2VAE's depth performance; the Gram and norm regularizers both contribute meaningfully on AbsRel and CD.

Configuration	AbsRel (↓)	δ_1 (↑)	CD (↓)	ATE (↓)	AUC@30 (↑)
Full S^2VAE	**0.04**	**0.98**	**0.14**	0.001	99.5
No gated projection	0.05	0.96	0.16	0.001	**99.7**
No Gram loss	0.09	0.93	0.21	0.001	99.6
No per-layer norm loss	0.07	0.94	0.18	0.001	**99.7**
No variance-reg loss	0.05	0.97	0.16	0.001	99.6

metrics (ATE, AUC@30) are essentially unaffected by any single ablation, indicating that camera information is robustly preserved once the hyperspherical latent is in place.

J Single High-Dimensional Sphere Ablation

A natural alternative to our product-of-spheres construction is to use a single Power Spherical distribution with the same total latent dimensionality. Hence, we train a corresponding model with a single $D = 128$ Power Spherical bottleneck, while keeping the encoder, decoder, hidden dimension, training schedule, losses, and total latent dimensionality fixed.

In our setting, the single high-dimensional sphere is numerically unstable. As the dimensionality of a sphere increases, its surface area becomes extremely small, and we observe instability in the corresponding spherical KL optimization and concentration parameter κ. In practice, the single-sphere model fails to obtain a useful depth representation, whereas factorizing the same total latent dimensionality into multiple low-dimensional spheres trains reliably.

Table 6 reports the comparison. Importantly, this experiment is intended to motivate the factorized construction used in S^2VAE: it shows that, under the same training recipe, directly scaling a single Power Spherical bottleneck to the required latent dimensionality is unstable, while the product-of-spheres formulation avoids this failure mode.

Table 6: **Single high-dimensional sphere ablation on VGGT.** All three models share the same encoder and decoder, total latent dimension ($D = 128$), hidden dimension, training schedule, and data; only the latent distribution differs. The single high-dimensional Power Spherical bottleneck exhibits numerical instability in this setting. CD in this ablation is computed using a confidence-filtered point-cloud protocol and should, therefore, be compared only within this table.

Bottleneck	AbsRel (↓)	δ_1 (↑)	CD (↓)	ATE (↓)	AUC@30 (↑)
Product of Power Spherical (ours)	0.04	0.98	0.31	0.001	99.5
Gaussian	0.07	0.95	0.34	0.002	99.3
Single Power Spherical	1.21	0.25	0.32	0.002	99.4

K Latent dimension sweep

We stress-test the bottleneck under aggressive dimensional reduction by sweeping the latent dimension $D \in \{8, 16, 32, 64, 128\}$ using the compact (∼9M-parameter) architecture. For every latent dimension, we train matched Gaussian and S^2VAE variants with three independent seeds while keeping the architecture, training objective, optimizer, and number of optimization steps fixed.

Table 7 reports the results. S^2VAE achieves lower mean AbsRel at every latent dimension and generally improves depth, point-cloud, and trajectory reconstruction, with the largest practical gains appearing under aggressive compression. Because these seeded runs were trained for a fixed 20k steps rather than to full convergence, we use this experiment to assess the consistency of the relative trend rather than the fully converged operating point.

Table 7: **Three-seed latent-dimension sweep on the compact VGGT models.** For each latent dimension, we compare matched Gaussian and S^2VAE bottlenecks. We report mean $\pm$ variation across three independent training seeds. All models in this sweep are trained for a fixed 20k optimization steps with batch size 24; these runs are therefore intended as a controlled seeded comparison rather than fully converged operating points. Bold denotes the better mean for each latent dimension and metric.

D	**Bottleneck**	**AbsRel** (↓)	$\boldsymbol{\delta_1}$ (↑)	**CD** (↓)	**ATE** (↓)	**AUC@30** (↑)
8	Gaussian	0.035 ±0.003	0.972 ±0.002	0.021 ±0.003	0.021 ±0.001	97.40 ±0.03
	S^2VAE	**0.029** ±0.002	**0.975** ±0.001	**0.020** ±0.003	**0.018** ±0.0003	**97.60** ±0.09
16	Gaussian	0.034 ±0.001	0.974 ±0.0004	0.020 ±0.003	0.019 ±0.0008	**98.00** ±0.10
	S^2VAE	**0.028** ±0.002	**0.975** ±0.0005	**0.017** ±0.002	**0.014** ±0.0006	97.90 ±0.23
32	Gaussian	0.032 ±0.0002	0.974 ±0.002	0.021 ±0.004	0.017 ±0.003	**98.20** ±0.03
	S^2VAE	**0.025** ±0.002	**0.976** ±0.003	**0.018** ±0.002	**0.010** ±0.0008	98.10 ±0.10
64	Gaussian	0.030 ±0.003	0.975 ±0.001	0.020 ±0.004	0.014 ±0.0008	98.30 ±0.03
	S^2VAE	**0.024** ±0.003	**0.976** ±0.002	**0.019** ±0.003	**0.0094** ±0.0003	**98.40** ±0.02
128	Gaussian	0.028 ±0.005	0.975 ±0.0014	0.019 ±0.005	0.009 ±0.0001	**98.40** ±0.04
	S^2VAE	**0.023** ±0.007	**0.977** ±0.002	**0.018** ±0.002	**0.008** ±0.0006	**98.40** ±0.04

L INFERENCE COST

The spherical reparameterization adds essentially no overhead at inference time. Table 8 compares the per-frame timing and peak GPU memory of the spherical and Gaussian VAEs (both with D=128, on the same backbone and hardware): the spherical VAE is 0.15 ms slower per forward pass, ~ 5 fps lower in throughput, and uses an additional 4 MB of peak memory.

Table 8: **Inference cost comparison at D=128.** Per-frame timing and peak GPU memory; spherical reparameterization is essentially free at inference.

Component	**Spherical**	**Gaussian**
VGGT backbone (per frame)	66.20 ms	66.20 ms
Full VAE (per frame)	5.63 ms	5.48 ms
Throughput	177.62 fps	182.48 fps
Peak GPU memory	5.266 GB	5.262 GB

M SUB-SPHERE SPECIALIZATION

A natural question for our product-of-spheres design is whether the sub-spheres specialize in any meaningful way. We measure Spearman correlation and mutual information between each of the 16 sub-spheres of our base model (16 spheres of dimension 8, total latent dimension 128) and the three downstream tasks (camera, depth, point cloud) over 200 validation samples, with no losses encouraging specialization.

We find clear specialization. A subset of spheres encodes point-level geometry strongly ($|\rho| = 0.65$–0.74), and a different subset carries more than $2\times$ the average mutual information for camera information. Depth is distributed more broadly: most spheres carry some depth-specific signal, and a few carry over $2\times$ the average mutual information. Crucially, no sphere is uninformative ($|\rho| > 0.19$ across all sphere/task pairs), so latent capacity is not being wasted. The product structure introduces a useful compositional inductive bias for free, in addition to its numerical-stability benefits, something a single high-dimensional sphere cannot offer.

N Additional Bottleneck Baselines

We additionally compare S^{2}VAE against a normalized Gaussian bottleneck and Finite Scalar Quantization (FSQ). These results come from a separate controlled experiment, distinct from the three-seed full-scale comparison reported in Table 2. Within this experiment, the models use the same training losses, loss weights, latent dimensionality, and evaluation protocol, with the bottleneck formulation being the primary difference. We therefore report the results separately and compare methods within this experimental setting.

As shown in Table 9, S^{2}VAE obtains the best performance across all evaluated metrics. The advantage over the normalized Gaussian indicates that the improvement is not limited to comparison with an unconstrained Gaussian prior, while FSQ (Mentzer et al., 2023) provides a discrete quantization-based alternative.

Table 9: **Additional bottleneck baselines on VGGT.** S^{2}VAE is compared against a normalized Gaussian bottleneck and Finite Scalar Quantization (FSQ) in a separate controlled experiment from the three-seed comparison in Table 2. Results should therefore be compared within this table.

Bottleneck	**AbsRel** (↓)	$\boldsymbol{\delta_1}$ (↑)	**CD** (↓)	**ATE** (↓)	**AUC@30** (↑)
S^{2}VAE (ours)	**0.022**	**0.986**	**0.011**	**0.0002**	**99.33**
Normalized Gaussian	0.025	0.981	0.015	0.0003	99.21
FSQ	0.024	0.985	0.013	0.0004	99.18

O Learned Projection Baselines

We additionally compare S^{2}VAE against simpler dimensionality-reduction and learned projection baselines: Principal Component Analysis (PCA), a linear autoencoder, and a multilayer perceptron (MLP) autoencoder. We also include a matched Gaussian VAE for reference. All methods use the same input features, latent dimensionality, evaluation samples, and frozen downstream VGGT heads.

These experiments test whether the downstream information preserved by S^{2}VAE can be recovered by substantially simpler compression mechanisms. As shown in Table 10, the learned VAE representations substantially outperform PCA and the simpler autoencoder baselines on depth and point-cloud reconstruction. Camera metrics are less discriminative across methods, consistent with the relatively low-dimensional structure of the camera representation.

Table 10: **Comparison with projection and autoencoding baselines on VGGT.** All methods use the same input features, latent dimensionality, evaluation samples, and frozen downstream heads. Bold denotes the best result in each column; ties at the reported precision are all highlighted.

Method	**AbsRel** (↓)	$\boldsymbol{\delta_1}$ (↑)	**CD** (↓)	**ATE** (↓)	**AUC@30** (↑)
PCA	0.074	0.961	0.043	**0.001**	**99.6**
Linear AE	0.085	0.950	0.056	0.004	98.6
MLP AE	0.064	0.963	0.037	0.004	98.6
Gaussian VAE	0.024	0.988	0.014	**0.001**	99.4
S^{2}VAE (ours)	**0.021**	**0.989**	**0.011**	**0.001**	**99.6**

P Limitations.

Our work has several limitations. First, most reconstruction experiments evaluate fidelity to the outputs of frozen downstream heads rather than sensor-measured ground truth, and therefore primarily measure how well the compressed representation preserves information already present in the underlying backbone. Second, under aggressive compression, S^{2}VAE can over-smooth fine-grained local geometry, even when global scene structure is preserved. Third, the benefits are not uniform across all tasks and metrics; in particular, camera-pose metrics are often close to saturation and

therefore less discriminative. Moreover, the product-of-spheres latent should be viewed as a geometry-aware inductive bias rather than as being topologically identical to the original feature manifold. Finally, our latent-space diffusion experiment is intended as a proof of concept;, and we do not claim superior generative quality or cross-modal consistency over alternative latent representations without dedicated quantitative evaluation.