Anchor3DGS: Feed-Forward 3D Gaussian Splatting with Compact Anchor-Based Representation

Sheng Ye[1,2,*] **, Jiangke Lin**[2], **Fudong Wang**[2], **Yi Yuan**[2,†], **Zhen-Hui Dong**[1], **Yong-Jin Liu**[1,‡]
[1]Tsinghua University
[2]Ant Group, Inclusion AI
{yec22, dzh23}@mails.tsinghua.edu.cn, liuyongjin@tsinghua.edu.cn
{linjiangke.ljk, wangfudong.wfd, yy464236}@antgroup.com

Abstract

Recent feed-forward 3D Gaussian Splatting (3DGS) methods have enabled effective multi-view 3D reconstruction by predicting Gaussian primitives in a single forward pass. However, existing approaches predominantly adopt a pixel-aligned paradigm that predicts one Gaussian per input pixel, coupling the Gaussian count to the input resolution and number of views. This also leads to redundant primitives in textureless regions and insufficient coverage in geometrically complex areas. We present **Anchor3DGS**, a pose-free feed-forward 3DGS framework that replaces pixel-aligned prediction with a compact anchor-based representation. Guided by information entropy, our method places a set of 3D anchors in the scene, concentrating them in regions of higher visual complexity while maintaining broad spatial coverage. Each anchor performs occlusion-aware aggregation of multi-view image features, exchanges information with other anchors, and is decoded into a set of local Gaussians. A lightweight feature enhancement module further refines the coarse RGB output from Gaussian splatting and improves rendering fidelity. Extensive experiments on diverse benchmark datasets demonstrate that our **Anchor3DGS** achieves state-of-the-art novel view synthesis quality while only using an order of magnitude fewer Gaussian primitives.

1 Introduction

Reconstructing 3D scenes and synthesizing novel views from multiple 2D captures is a fundamental problem in computer vision and graphics, with broad applications in robotics, autonomous driving, gaming, and virtual reality. The pioneering NeRF [19] represents scenes as an implicit radiance field optimized via volume rendering, while subsequent 3DGS [13] employs differentiable rasterization of anisotropic 3D Gaussians to achieve real-time, high-fidelity rendering. Despite its success, standard 3DGS requires a time-consuming per-scene optimization process. To address this limitation, a growing line of work has explored feed-forward 3DGS reconstruction [4, 6, 34, 7, 32], which trains generalizable models on large-scale datasets to directly predict 3D Gaussians from input views in a single forward pass. Recent advances have further eliminated the dependency on pre-computed camera poses [38, 11, 9], enabling truly end-to-end reconstruction from uncalibrated images.

However, current feed-forward approaches adopt a *pixel-aligned* paradigm in which one Gaussian is predicted for each input pixel. While conceptually straightforward, this design suffers from three key limitations: (1) **Redundancy**—textureless or homogeneous regions consume as many Gaussians as detail-rich areas, yielding redundant primitives and wasted capacity; (2) **Insufficient**

[*]Work done during an internship at Ant Group.
[†]Project lead.
[‡]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

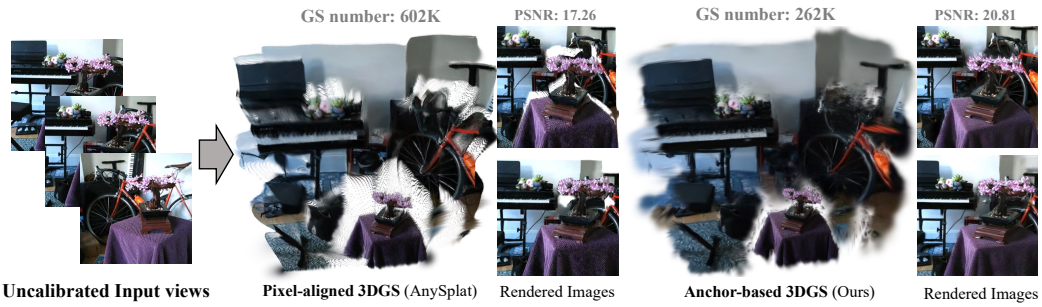


Figure 1: **Anchor3DGS** reconstructs 3D scenes from uncalibrated multi-view images. Our anchor-based representation achieves better rendering quality with fewer Gaussians, and avoids the empty holes that pixel-aligned methods (*e.g.,* AnySplat [11]) commonly leave in poorly observed regions.

coverage—poorly observed regions cannot be faithfully reconstructed, since each Gaussian is predicted independently from a single pixel, without leveraging contextual information from well-observed neighbors; and (3) **Input dependency**—the total Gaussian count is strictly tied to the input resolution and view count rather than to the scene itself, preventing efficient reconstruction.

In this work, we present **Anchor3DGS**, a feed-forward framework that takes uncalibrated multi-view images as input and outputs 3D Gaussians of the scene (see Fig. 1). Our key insight is that a scene's 3D structure has no intrinsic connection to the number of input views or pixels that happen to observe it. We therefore propose to lift Gaussian allocation into 3D space: we represent the scene with a set of anchors whose *spatial distribution* is guided by scene content, while the *total number of anchors* is a hyperparameter, independent of image resolution and view count.

Our framework consists of several key components: *(i) Entropy-guided anchor sampling.* Our 3D anchor sampling strategy simultaneously ensures broad spatial coverage of the scene and adaptively allocates more anchors to regions exhibiting higher visual complexity, as identified by local information entropy. Entropy automatically guides 3D anchors to where they are needed most: few anchors suffice for low-complexity regions (*e.g.,* white walls or sky), while many anchors concentrate on high-complexity regions (*e.g.,* intricate structures or rich textures). *(ii) Multi-view anchor aggregation and local decoding.* Each 3D anchor aggregates visual features from all input views under occlusion-aware weighting, and is decoded into a small set of neighboring Gaussians that represent its local 3D region. Anchors also exchange information with each other, allowing those in poorly observed regions to leverage the context from well-observed neighbors and still produce plausible appearances. In contrast, pixel-aligned methods often leave visible holes in such regions. *(iii) Feature enhancement.* Beyond standard attributes, each Gaussian is also assigned a feature vector. These features are later splatted and decoded by a lightweight network, which compensates for the high-frequency details missing from the coarse splatted RGB image.

Extensive experiments across benchmark datasets validate the effectiveness of our method. In sparse-view settings (3 and 6 input views), **Anchor3DGS** significantly outperforms existing state-of-the-art baselines. In dense-view settings (32 input views), **Anchor3DGS** achieves comparable rendering quality while requiring an order of magnitude fewer Gaussians. This demonstrates that a compact anchor-based representation can match the fidelity of pixel-aligned approaches with greater efficiency. Our contributions can be summarized as follows:

- We propose **Anchor3DGS**, a novel anchor-based feed-forward 3DGS framework that replaces the pixel-aligned paradigm with a compact anchor-based representation, decoupling the Gaussian count from input image resolution and the number of input views.
- We introduce three key designs—*entropy-guided anchor sampling*, *occlusion-aware anchor aggregation and local decoding*, and *feature enhancement*—which together enable high-fidelity novel view synthesis with fine details.
- We achieve state-of-the-art performance on multiple benchmark datasets with substantially fewer Gaussians than existing methods, demonstrating that our method yields a more compact yet equally effective representation.

2 Related Work

Per-Scene Novel View Synthesis. Neural Radiance Fields (NeRF) [19] pioneered novel view synthesis by representing scenes as continuous density and radiance fields, and optimizing them via differentiable volume rendering. Despite subsequent improvements [27, 17, 2, 20, 41, 12], the expensive ray sampling in volume rendering still prevents real-time performance. 3D Gaussian Splatting (3DGS) [13, 33] instead represents scenes as explicit anisotropic 3D Gaussians and exploits differentiable tile-based rasterization, achieving real-time speed with superior quality. However, both NeRF- and 3DGS-based methods require accurate camera poses, typically from a costly Structure-from-Motion (SfM) pipeline such as COLMAP [24]. To remove this dependency, several works [3, 10, 14, 8] attempt to jointly optimize the camera poses and the scene representation. Nevertheless, all these methods still rely on tedious per-scene optimization. In contrast, our **Anchor3DGS** directly predicts camera poses and 3D Gaussians in a feed-forward pass within seconds.

Feed-Forward Novel View Synthesis. Feed-forward novel view synthesis avoids per-scene optimization by learning generalizable priors from large-scale datasets. Early works [40, 29, 5] condition the NeRF representation on image features aggregated from input views, enabling cross-scene generalization without per-scene retraining. Following the success of 3DGS, subsequent feed-forward works [26, 4, 6, 7, 34, 32, 31, 42] replace NeRF with explicit Gaussian primitives for higher rendering quality and faster speed. As representative examples, pixelSplat [4] employs epipolar attention to infer a dense depth probability distribution for predicting 3D Gaussians from image pairs, whereas MVSplat [6] incorporates multi-view stereo principles by constructing explicit cost volumes for more accurate geometric reconstruction. A recent line of pose-free feed-forward methods [38, 44, 37, 11] further eliminates the reliance on known camera parameters. By leveraging robust geometric priors from 3D foundation models [30, 28], FLARE [44] and AnySplat [11] can simultaneously infer the camera poses of input images and the matching geometry-aligned 3D Gaussians. Despite these advances, all aforementioned feed-forward 3DGS methods typically predict Gaussians in a dense, pixel-aligned manner, resulting in massive redundant Gaussians. Although AnySplat [11] applies post-hoc voxelization for Gaussian reduction, this fails to address the root cause, and the remaining Gaussians are still far from compact. Instead, we decouple Gaussian allocation from the pixel-aligned paradigm and model the scene using a compact set of 3D anchors. A few concurrent works [1, 22] also attempt to explore compact representations using learnable tokens, whereas we explicitly leverage 3D scene structure and visual complexity to guide Gaussian allocation.

3 Method

The key idea of our method is to represent the scene with a compact set of 3D anchors, rather than being strictly bound to the input image pixels. Each anchor represents a local 3D region: it is placed on a geometry-aware scene scaffold, gathers information from multiple views, exchanges information with other anchors, and is finally decoded into a set of local Gaussians. In this way, the scene representation is organized directly in 3D space, with its size controlled by the anchor budget.

Concretely, given a set of uncalibrated input images $\mathcal{I} = \{I_v\}_{v=1}^{V}$, we first build a multi-view geometric scaffold using a VGGT-based pose-free geometry backbone [28], which predicts camera parameters $\mathcal{C} = \{C_v\}_{v=1}^{V}$ and per-view depth maps $\mathcal{D} = \{D_v\}_{v=1}^{V}$. By unprojecting the predicted depth maps with the predicted cameras, we obtain scene points $\mathcal{P}$ in 3D space, from which we sample a fixed number of anchors guided by local image entropy. Each anchor is then encoded into a latent token $\mathbf{z}_n$ through occlusion-aware multi-view feature aggregation and cross-anchor interaction. Finally, each token is decoded into K local Gaussian primitives, yielding the final Gaussian set $\mathcal{G} = \{g_m\}_{m=1}^{NK}$. Fig. 2 illustrates the overall pipeline of **Anchor3DGS**.

3.1 Pose-Free Geometry Initialization

Before constructing anchors, we first build a geometry-aware 3D scaffold of the scene. The role of this stage is to lift the uncalibrated input images into a consistent geometric space on which anchors can be placed and reasoned about. To this end, we employ a VGGT-based [28] backbone model as a strong multi-view geometry prior.

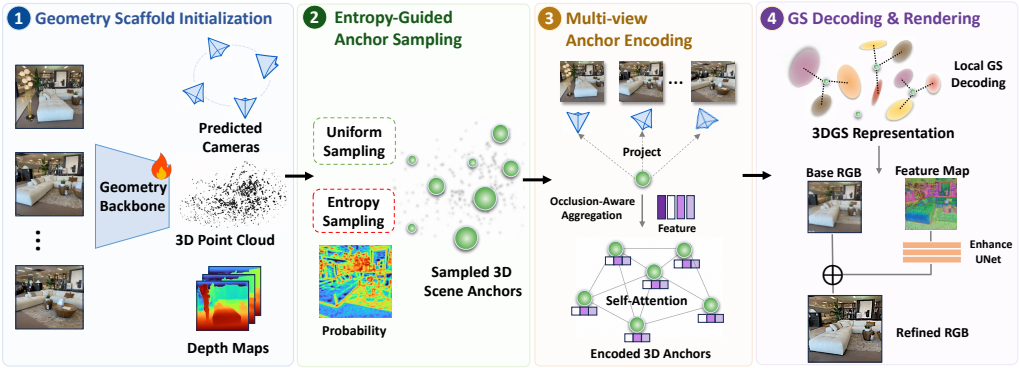


Figure 2: **Overview of Anchor3DGS.** Given uncalibrated multi-view images, *(1)* a VGGT-based [28] geometry backbone first predicts the scene's geometric scaffold. *(2)* A fixed budget of 3D anchors is then sampled from the scaffold, combining *uniform* sampling with *entropy-guided* sampling. *(3)* Each anchor is projected back into all input views to gather features via occlusion-aware aggregation, and then interacts with other anchors through self-attention. *(4)* Finally, each anchor is decoded into a set of local Gaussians, which are rasterized and refined to yield the final RGB output.

Given input images $\mathcal{I} = \{I_v\}_{v=1}^{V}$ with $I_v \in \mathbb{R}^{H\times W\times 3}$, the geometry backbone predicts for each view a camera extrinsic $E_v \in \mathbb{R}^{3\times 4}$, a camera intrinsic $K_v \in \mathbb{R}^{3\times 3}$, and a depth map $D_v \in \mathbb{R}^{H\times W}$. We denote the overall predicted camera parameters by $\mathcal{C} = \{C_v\}_{v=1}^{V}$ with $C_v = (E_v, K_v)$. Using $\mathcal{C}$, we unproject each depth map D_v into a per-view 3D point map $\mathbf{P}_v \in \mathbb{R}^{H\times W\times 3}$, and denote the resulting set of 3D scene points by $\mathcal{P} = \{\mathbf{P}_v\}_{v=1}^{V}$.

This geometric scaffold serves two purposes in our method. First, the scene points in $\mathcal{P}$ provide a geometry-aware 3D candidate pool for anchor sampling. Second, the predicted camera parameters $\mathcal{C}$ allow every anchor to be projected back into the input views, which is essential for subsequent multi-view feature aggregation and occlusion reasoning. Based on this geometric scaffold, our final representation is built as a fixed-size set of 3D anchors with features in scene space.

3.2 Entropy-Guided Anchor Sampling

An effective allocation should both broadly cover the scene and place more anchors in visually complex regions where finer local representation is needed. To capture such complexity, we compute a Shannon entropy map [25, 45] for each input image. We first convert the RGB image to grayscale, quantize the grayscale values into bins, and compute a local histogram within a sliding window around each pixel. The entropy at image location $\mathbf{u}$ is then

$$E(\mathbf{u}) = -\sum_{b=1}^{B} p_b(\mathbf{u}) \log p_b(\mathbf{u}), \tag{1}$$

where $p_b(\mathbf{u})$ is the fraction of pixels in the local window around $\mathbf{u}$ whose quantized grayscale values fall into the b-th bin. Regions with simple appearance, such as smooth walls or sky, tend to have low entropy, while textured structures and local boundaries tend to have high entropy. We use this quantity as a direct and training-free measure of local visual complexity (shown in Fig. 3).

Given the scene points $\mathcal{P} = \{\mathbf{P}_v\}_{v=1}^{V}$, we construct the anchor set using two complementary steps. We first perform coarse grid subsampling over the 3D points $\mathcal{P}$ to obtain anchors that broadly cover the scene. We then sample anchors from the full set of scene points with probabilities proportional to the entropy values at their associated image locations, so that more anchors are allocated to visually complex regions. Together, these two subsets form the final anchor set $\mathcal{A}$:

$$\mathcal{A} = \mathcal{A}_{\text{uni}} \cup \mathcal{A}_{\text{ent}} = \{\mathbf{a}_n\}_{n=1}^{N}, \quad \mathbf{a}_n \in \mathbb{R}^3, \tag{2}$$

where N is a fixed anchor budget. In this way, scene points provide anchor location candidates in 3D, while the final number of anchors is independent of the input resolution and the number of input views. In addition, for each anchor $\mathbf{a}_n$, we estimate a local spacing value d_n by averaging the

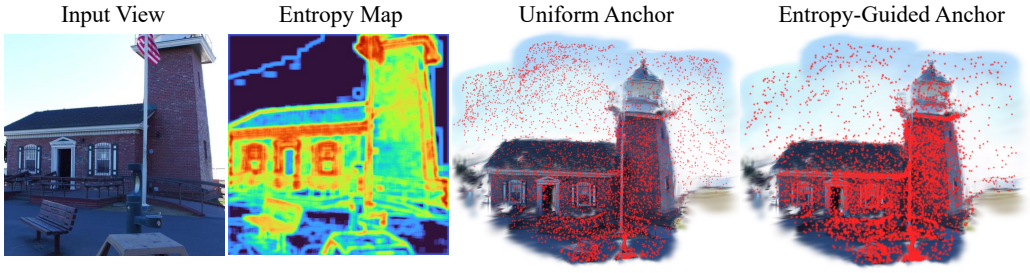


Figure 3: Uniform sampling ensures broad spatial coverage but wastes anchors (red dots) on uninformative regions (*e.g.,* textureless sky). In contrast, our entropy-guided sampling allocates more anchors to visually complex regions while still maintaining coverage of the scene.

distances to its k nearest neighboring anchors:

$$d_n = \frac{1}{k} \sum_{j \in \mathcal{N}_k(n)} \|\mathbf{a}_n - \mathbf{a}_j\|_2. \tag{3}$$

This spacing value provides a geometric density cue that will later be used to modulate the spatial extent of local Gaussian decoding.

3.3 Occlusion-Aware Multi-View Anchor Aggregation

With anchors allocated in scene space, the next step is to encode them for local Gaussian decoding. This differs from pixel-aligned Gaussian prediction, where each primitive corresponds to a single pixel observation from a specific view. In **Anchor3DGS**, each anchor represents a local 3D region in the scene, and is therefore encoded by aggregating information from multiple views.

For each anchor $\mathbf{a}_n$, we first compute an initial anchor embedding that captures both anchor position and local geometric scale information:

$$\mathbf{q}_n^{\text{geo}} = \text{MLP}(\phi(\mathbf{a}_n, \log d_n)), \tag{4}$$

where $\phi(\cdot)$ denotes Fourier positional encoding [19] applied to the 3D anchor coordinates and the logarithm of its local spacing d_n. Using the predicted camera parameters $\mathcal{C}$, we project each anchor $\mathbf{a}_n$ onto every input view and bilinearly sample the feature map $\mathbf{F}_v$ (extracted by the VGGT-based backbone in Sec. 3.1) at the projected location, yielding a per-view anchor feature $\mathbf{f}_{n,v} \in \mathbb{R}^C$. For each sampled feature $\mathbf{f}_{n,v}$, we also estimate an occlusion weight $w_{n,v}^{\text{occ}} \in (0, 1)$, which attenuates the contribution of occluded views. Specifically, we calculate $w_{n,v}^{\text{occ}}$ by comparing the camera-space depth $z_{n,v}$ of the anchor with the depth value $\tilde{z}_{n,v}$ sampled from D_v (provided by the geometry backbone in Sec. 3.1) at the same projected location. If the anchor lies behind the visible surface, then $z_{n,v}$ will be noticeably larger than $\tilde{z}_{n,v}$, indicating likely occlusion; otherwise, the anchor is visible and its sampled feature can be trusted. We formalize this intuition as follows:

$$\Delta_{n,v} = \frac{z_{n,v} - \tilde{z}_{n,v}}{|\tilde{z}_{n,v}| + \epsilon}, \quad w_{n,v}^{\text{occ}} = \sigma(-\tau \Delta_{n,v}), \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function and τ controls the sharpness to depth discrepancies. We then aggregate the per-view features $\mathbf{f}_{n,v}$ biased by the occlusion weights:

$$\alpha_{n,v} = \text{softmax}_v\big(\text{MLP}([\mathbf{q}_n^{\text{geo}}, \mathbf{f}_{n,v}]) + \log w_{n,v}^{\text{occ}}\big), \quad \mathbf{f}_n^{\text{agg}} = \sum_{v=1}^{V} \alpha_{n,v}\, \mathbf{f}_{n,v}. \tag{6}$$

The MLP scores each per-view feature's relevance to the anchor, while the additive $\log w_{n,v}^{\text{occ}}$ term suppresses occluded features. Together, they yield a weighting that jointly considers feature informativeness and geometric visibility. Finally, the geometry-aware embedding $\mathbf{q}_n^{\text{geo}}$ and the aggregated feature $\mathbf{f}_n^{\text{agg}}$ are concatenated, and refined jointly across all N anchors via *self-attention* blocks: $\{\mathbf{z}_n \in \mathbb{R}^D\}_{n=1}^N = \text{SelfAttn}(\{[\mathbf{q}_n^{\text{geo}}, \mathbf{f}_n^{\text{agg}}]\}_{n=1}^N)$. This cross-anchor interaction allows anchors in weakly observed regions to draw context from better-observed neighbors, producing more coherent anchor embeddings $\{\mathbf{z}_n\}$ for subsequent Gaussian decoding.

3.4 Anchor-to-Gaussian Decoding and Rendering

After occlusion-aware aggregation, each anchor $\mathbf{a}_n$ is associated with an embedding $\mathbf{z}_n$. We then decode these embeddings into the final 3D Gaussians. Our goal is to keep the overall representation compact while still allowing each anchor to model a sufficiently rich local 3D region in both geometry and appearance. To this end, we decode a set of local Gaussians around each anchor.

Specifically, we introduce a set of K learnable slot embeddings $\{\mathbf{s}_k\}_{k=1}^{K}$ (one per decoded Gaussian) shared across all anchors. For each anchor embedding $\mathbf{z}_n$, we combine it with each slot embedding $\mathbf{s}_k$ and decode both a local Gaussian offset $\Delta\mathbf{x}_{n,k}$ and a set of Gaussian attributes $\boldsymbol{\theta}_{n,k}$:

$$\Delta\mathbf{x}_{n,k} = \mathrm{MLP}_{\text{off}}([\mathbf{z}_n, \mathbf{s}_k]), \quad \mathbf{x}_{n,k} = \mathbf{a}_n + d_n\,\Delta\mathbf{x}_{n,k}, \quad \boldsymbol{\theta}_{n,k} = \mathrm{MLP}_{\text{gs}}([\mathbf{z}_n, \mathbf{s}_k]). \tag{7}$$

Thus, the final center $\mathbf{x}_{n,k}$ of each decoded Gaussian is obtained by adding to the anchor $\mathbf{a}_n$ a local offset $\Delta\mathbf{x}_{n,k}$ scaled by the anchor's local spacing d_n (discussed in Sec. 3.2), allowing anchors in sparse regions to decode Gaussians over a wider spatial extent. The decoded Gaussian attributes include: $\boldsymbol{\theta}_{n,k} = (o_{n,k}, \Sigma_{n,k}, \mathbf{c}_{n,k}, \mathbf{h}_{n,k})$, where o is the opacity, Σ is the covariance, $\mathbf{c}$ is the base RGB, and $\mathbf{h}$ is a feature vector that encodes high-frequency details for subsequent refinement. For simplicity, we re-index the resulting NK Gaussians by a single index i in the following.

To render a target view, we rasterize the predicted Gaussians using the differentiable pipeline of 3DGS [13]. Each 3D Gaussian is first projected to 2D plane with mean $\boldsymbol{\mu}_i$ and covariance Σ_i'. The opacity contribution of Gaussian i at pixel $\mathbf{u}$ is then modulated by:

$$\alpha_i(\mathbf{u}) = o_i \exp\Big(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_i)^\top \Sigma_i'^{-1}(\mathbf{u} - \boldsymbol{\mu}_i)\Big). \tag{8}$$

Sorting all Gaussians by depth and accumulating from front to back yields the transmittance $T_i(\mathbf{u}) = \prod_{j<i}(1 - \alpha_j(\mathbf{u}))$. Together, T_i and α_i serve as the alpha-compositing weights that, combined with the per-Gaussian color $\mathbf{c}_i$, give the standard 3DGS RGB rendering. In our setting, since each Gaussian additionally carries a feature vector $\mathbf{h}_i$, we also feed it through the same rasterizer to obtain a feature map F_{splat} that is aligned with the rendered base image I_{base}:

$$I_{\text{base}}(\mathbf{u}) = \sum_i T_i(\mathbf{u})\,\alpha_i(\mathbf{u})\,\mathbf{c}_i, \quad F_{\text{splat}}(\mathbf{u}) = \sum_i T_i(\mathbf{u})\,\alpha_i(\mathbf{u})\,\mathbf{h}_i. \tag{9}$$

The two outputs are produced by the same compositing weights and therefore share identical geometric structures. The coarse image I_{base} provides geometry-consistent base color signals, while the feature map F_{splat} provides higher-dimensional appearance cues. Finally, a lightweight UNet-based enhancement network takes F_{splat} as input and predicts an RGB residual ΔI to yield the final refined image $\hat{I} = I_{\text{base}} + \Delta I$. This design decouples compact 3D scene modeling from high-frequency appearance recovery, while preserving multi-view consistency: since both I_{base} and F_{splat} are rasterized from the same 3D Gaussians, the residual ΔI inherits its geometric structure from the underlying 3D representation rather than being hallucinated in 2D.

4 Training

We train **Anchor3DGS** on five public multi-view datasets: DL3DV [16], ScanNet++ [39], Hypersim [23], BlendedMVS [36], and Waymo [18]. These datasets cover synthetic and real-world captures, indoor and outdoor scenes, and object-scale to street-scale settings, which enhances the model's generalization ability to unseen scenarios. For each training scene, we sample a set of context views as input. For datasets with sequentially ordered captures (*e.g.,* DL3DV [16]), we adopt an *index-based* sampler that selects two boundary frames at a chosen index gap, and samples additional context views from intermediate frame indices between them. For datasets without a sequential ordering (*e.g.,* ScanNet++ [39]), we instead use a *ranking-based* sampler that ranks all views by the camera-pose distance (rotation angle and translation) relative to a randomly chosen reference, and then selects context views from this distance-sorted list. In both cases, the extent covered by the selected context views (an index gap or a rank window) is randomized across iterations, producing a mixture of small- and wide-baseline configurations during training.

Our model is trained end-to-end by minimizing a combination of photometric and geometric losses. The photometric term consists of a pixel-wise MSE loss and a perceptual LPIPS [43] loss between

Table 1: Quantitative comparison of the **sparse-view** novel view synthesis (3 and 6 input views) on VR-NeRF [35], MipNeRF360 [2], and DL3DV [16]. We report PSNR, SSIM, LPIPS, and the number of predicted Gaussian primitives (#GS). **Bold** indicates the best.

Setting	Method	VR-NeRF				MipNeRF360				DL3DV			
		PSNR	SSIM	LPIPS	#GS	PSNR	SSIM	LPIPS	#GS	PSNR	SSIM	LPIPS	#GS
3 views	NoPoSplat [38]	20.52	0.686	0.433	0.60M	16.89	0.449	0.493	0.60M	17.29	0.509	0.451	0.60M
	FLARE [44]	20.64	0.692	0.452	0.60M	16.92	0.447	0.498	0.60M	16.08	0.468	0.498	0.60M
	DA3 [15]	19.87	0.656	0.464	0.60M	12.84	0.330	0.634	0.60M	18.24	0.510	0.376	0.60M
	AnySplat [11]	21.31	0.729	0.340	0.49M	15.65	0.624	0.331	0.38M	17.74	0.625	0.308	0.51M
	Ours	**23.69**	**0.764**	**0.320**	0.26M	**21.36**	**0.678**	**0.307**	0.26M	**21.22**	**0.683**	**0.280**	0.26M
6 views	NoPoSplat [38]	20.61	0.712	0.445	1.20M	16.81	0.457	0.494	1.20M	15.93	0.477	0.545	1.20M
	FLARE [44]	19.96	0.697	0.478	1.20M	15.38	0.409	0.583	1.20M	15.36	0.457	0.564	1.20M
	DA3 [15]	21.32	0.690	0.403	1.20M	18.40	0.472	0.389	1.20M	18.97	0.539	0.349	1.20M
	AnySplat [11]	22.45	0.758	0.321	1.03M	19.01	0.613	0.283	0.70M	18.42	0.613	0.316	0.98M
	Ours	**24.03**	**0.781**	**0.312**	0.26M	**22.39**	**0.663**	**0.278**	0.26M	**20.70**	**0.649**	**0.303**	0.26M

the refined renderings $\hat{I}$ and the ground-truth images I. To regularize the geometry of the decoded Gaussians, we additionally apply a depth-consistency loss that encourages the depth rasterized from the Gaussians to agree with the depth maps predicted by the geometry backbone in Sec. 3.1. Therefore, the full training loss is:

$$\mathcal{L} = \lambda_{\text{mse}}\mathcal{L}_{\text{mse}} + \lambda_{\text{lpips}}\mathcal{L}_{\text{lpips}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}}. \tag{10}$$

5 Experiments

5.1 Experimental Setup

Evaluation Datasets & Baselines. Following AnySplat [11], we evaluate the novel view synthesis quality on VR-NeRF [35] and MipNeRF360 [2] datasets. We also test on a held-out subset of DL3DV [16] that is disjoint from training. Together, these benchmarks jointly cover object-centric, indoor, and outdoor scenes, providing a comprehensive evaluation.

We compare **Anchor3DGS** against recent pose-free feed-forward 3DGS methods: NoPoSplat [38], FLARE [44], Depth Anything 3 (DA3) [15], and AnySplat [11]. Among them, NoPoSplat and FLARE are tailored for sparse-view inputs and do not scale to dense-view inputs, so we only include them in the sparse-view comparison (3 and 6 views). In the dense-view setting (32 views), we also report VGGT+3DGS as a per-scene optimization reference. We compute PSNR, SSIM, and LPIPS between the rendered images and the ground-truth images. To further reflect the compactness of our representation, we also report the number of Gaussian primitives (*#GS*) for each scene.

Implementation Details. We set the anchor budget to $N = 32{,}768$ and decode $K = 8$ Gaussians per anchor, yielding roughly 0.26M Gaussians per scene independent of input resolution and view count. The dimension of each anchor embedding $\mathbf{z}_n$ is 128, and the dimension of per-Gaussian feature vector $\mathbf{h}$ (used for rendering refinement) is 16. We initialize the geometry backbone of **Anchor3DGS** with the pretrained VGGT [28] weights and fine-tune it jointly with the rest of the network using a $0.1\times$ smaller learning rate, while keeping the patch-embedding layer frozen. The model is trained on 8 NVIDIA H20 GPUs for $30{,}000$ iterations (about 2–3 days). We use the AdamW optimizer with a peak learning rate of 2×10^{-4} and $1{,}000$ warmup steps. During training, the number of context views per scene is randomly sampled between 2 and 16, and the maximum input resolution is set to 448 pixels on the longer side. At inference time, all images are resized to 448×448 resolution. For the training loss weights, we set $\lambda_{\text{mse}} = 1.0$, $\lambda_{\text{lpips}} = 0.05$, and $\lambda_{\text{depth}} = 0.1$.

5.2 Quantitative Results

We first evaluate **Anchor3DGS** in the sparse-view setting (3 and 6 input views). As shown in Tab. 1, our method consistently achieves the best PSNR, SSIM, and LPIPS across all three benchmark datasets. The improvements are most pronounced on MipNeRF360 and DL3DV, where **Anchor3DGS**

Table 2: **Dense-view** rendering results (32 input views) on VR-NeRF [35] and DL3DV [16]. We also report the storage size of the predicted Gaussians and the inference time on a single H800 GPU.

Method	VR-NeRF						DL3DV					
	PSNR	SSIM	LPIPS	#GS	Storage	Time	PSNR	SSIM	LPIPS	#GS	Storage	Time
VGGT+3DGS [13]	23.08	0.762	0.301	0.44M	109MB	6min	19.34	**0.627**	**0.312**	1.16M	288MB	8min
DA3 [15]	22.38	0.751	0.350	6.42M	1.03GB	1.9s	19.38	0.544	0.341	6.42M	1.03GB	2.0s
AnySplat [11]	24.05	**0.819**	0.309	4.75M	779MB	2.8s	18.76	0.565	0.371	3.98M	653MB	2.8s
Ours	**24.56**	0.815	**0.297**	0.26M	35MB	2.9s	**19.52**	0.593	0.364	0.26M	35MB	3.1s



Figure 4: Qualitative comparison of novel view synthesis. Our method recovers sharper textures and finer details while mitigating artifacts in poorly observed regions.

achieves PSNR gains of over 2 dB compared to the best baseline under each setting. These gains stem from two advantages of our anchor-based design: entropy-guided sampling concentrates capacity on visually complex regions, while cross-anchor self-attention propagates context to poorly-observed areas, mitigating the holes prevalent in pixel-aligned methods. Notably, **Anchor3DGS** achieves these gains with only 0.26M Gaussians per scene, showing that a compact anchor-based representation can yield both higher fidelity and greater efficiency in the sparse-view setting.

Furthermore, we test **Anchor3DGS** in the dense-view setting (32 input views), where the input already provides rich coverage of the scene. As shown in Tab. 2, our method provides comparable rendering quality against state-of-the-art pixel-aligned baselines, while requiring an order of magnitude fewer Gaussians and smaller storage footprint. Since the anchor number is decoupled from the input views, our #GS stays at 0.26M regardless of view count, whereas pixel-aligned methods produce several million Gaussians with no corresponding quality gain. The VGGT+3DGS also provides a strong reference but represents a different paradigm—it requires minutes of optimization per scene, whereas **Anchor3DGS** directly predicts the 3DGS representation from the input views in a feed-forward pass within seconds. Despite this efficiency gap, our method still achieves on-par rendering quality.

5.3 Qualitative Results

Fig. 4 presents qualitative comparisons under both sparse-view and dense-view settings. In the sparse-view setting, the representative method NoPoSplat [38] tends to produce overly blurry renderings, while AnySplat [11] frequently leaves holes in poorly observed regions. Our method alleviates both issues, yielding coherent structure and visually appealing results across different scenes. In the dense-view setting, our renderings remain on par with the state-of-the-art baselines while using an

Table 3: Ablation study results on VR-NeRF [35] dataset. **Bold** indicates the best.

Configuration	3 views			32 views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
w/o enhancement	22.50	0.754	0.345	24.13	0.807	0.324
w/o self-attention	23.03	0.750	0.323	24.38	0.808	0.303
w/o occ-aggregation	23.28	0.756	0.330	24.29	0.813	0.305
w/o entropy-sample	22.16	0.733	0.355	24.17	0.804	0.318
Full model (Ours)	**23.69**	**0.764**	**0.320**	**24.56**	**0.815**	**0.297**

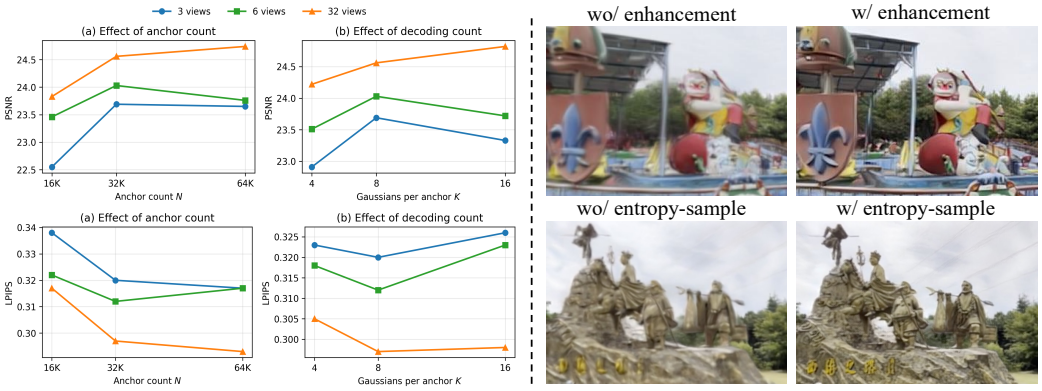


Figure 5: Rendering metrics under varying anchor count N and decoded Gaussians K (*left*), and qualitative comparisons of removing the enhancement module and entropy-guided sampling (*right*).

order of magnitude fewer Gaussian primitives. Our method recovers sharp textures and fine details, and avoids the fragmented, needle-like artifacts commonly exhibited by optimization-based 3DGS pipelines. More visualization results are provided in the Appendix.

5.4 Ablation Study

We conduct the ablation study to validate each key design of **Anchor3DGS**. Tab. 3 reports results on VR-NeRF [35] under both sparse-view (3 views) and dense-view (32 views) settings. Removing any component causes a quality drop, confirming that all four designs contribute to the final performance. *(i) w/o enhancement* removes the enhancement UNet for refinement, which leads to consistent degradation as the splatted RGB alone struggles to recover fine high-frequency details. *(ii) w/o self-attention* disables cross-anchor interaction so each anchor is encoded independently, and *(iii) w/o occ-aggregation* replaces occlusion-aware multi-view aggregation with plain mean pooling. Both of them impair the encoded anchor features and thus degrade rendering performance. *(iv) w/o entropy-sample* replaces entropy-guided sampling with uniform grid subsampling, which causes noticeable PSNR drop (-1.53 dB at 3 views). This indicates that our strategy allocates the limited anchor budget more reasonably. We further examine how performance varies with anchor count N and decoded Gaussians per anchor K in Fig. 5. The results confirm that our default setting ($N = 32{,}768$ and $K{=}8$) effectively balances rendering quality and Gaussian compactness.

6 Conclusion

We present **Anchor3DGS**, a pose-free feed-forward 3DGS framework that replaces the prevailing pixel-aligned paradigm with a compact anchor-based paradigm. By placing a fixed number of 3D anchors on a geometry-aware scaffold, our method decouples the Gaussian count from the input resolution and view number, and instead allocates representation capacity according to scene content. Three key designs—entropy-guided anchor sampling, occlusion-aware multi-view aggregation, and feature-based rendering refinement—together enable high-fidelity novel view synthesis. Extensive experiments across multiple datasets demonstrate that **Anchor3DGS** achieves state-of-the-art ren-

dering quality in sparse-view settings. In dense-view settings, **Anchor3DGS** attains comparable quality while using an order of magnitude fewer Gaussians and lower storage cost, validating the effectiveness and efficiency of our compact anchor-based representation. Beyond view synthesis, such a compact anchor-based representation naturally serves as a global geometric memory [46], offering a promising substrate for future video world models.

Despite the advancements, **Anchor3DGS** still has limitations. The performance of our method relies on the prediction quality of the geometry backbone. When the backbone produces inaccurate poses or noisy depth maps, the resulting anchor placement may degrade and harm the final rendering. In addition, since the anchor budget is fixed regardless of scene scale, very large or unbounded scenes may benefit from a scene-adaptive budget, which we leave as future work.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (62332019, 62461160309) and NSFC-RGC Joint Research Scheme N_HKU705/24.

References

[1] Honggyu An, Jaewoo Jung, Mungyeom Kim, Sunghwan Hong, Chaehyun Kim, Kazumi Fukuda, Minkyeong Jeon, Jisang Han, Takuya Narihira, Hyuna Ko, et al. C3g: Learning compact 3d representations with 2k gaussians. *arXiv preprint arXiv:2512.04021*, 2025.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[3] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4160–4169, 2023.

[4] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

[5] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14124–14133, 2021.

[6] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pages 370–386. Springer, 2024.

[7] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. *Advances in Neural Information Processing Systems*, 37:107064–107086, 2024.

[8] Zhen-Hui Dong, Sheng Ye, Yu-Hui Wen, Nannan Li, and Yong-Jin Liu. Towards better robustness: Progressively joint pose-3dgs learning for arbitrarily long videos. *arXiv preprint arXiv:2501.15096*, 2025.

[9] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Sparse-view gaussian splatting in seconds. *arXiv preprint arXiv:2403.20309*, 2024.

[10] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20796–20805, 2024.

[11] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

[12] Xinyi Jing, Tao Yu, Renyuan He, Yu-Kun Lai, and Kun Li. Frnerf: Fusion and regularization fields for dynamic view synthesis. *Computational Visual Media*, 11(5):965–981, 2025.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023.

[14] Chin-Yang Lin, Cheng Sun, Fu-En Yang, Min-Hung Chen, Yen-Yu Lin, and Yu-Lun Liu. Longsplat: Robust unposed 3d gaussian splatting for casual long videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27412–27422, 2025.

[15] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[16] 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*, pages 22160–22169, 2024.

[17] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20654–20664, 2024.

[18] Jieru Mei, Alex Zihao Zhu, Xinchen Yan, Hang Yan, Siyuan Qiao, Liang-Chieh Chen, and Henrik Kretzschmar. Waymo open dataset: Panoramic video panoptic segmentation. In *European Conference on Computer Vision*, pages 53–72. Springer, 2022.

[19] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, pages 405–421. Springer, 2020.

[20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15, 2022.

[21] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10901–10911, 2021.

[22] Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. Tokengs: Decoupling 3d gaussian prediction from pixels with learnable tokens. *arXiv preprint arXiv:2604.15239*, 2026.

[23] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10912–10922, 2021.

[24] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision*, pages 501–518. Springer, 2016.

[25] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[26] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024.

[27] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd E. Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5481–5490. IEEE, 2022.

[28] 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*, pages 5294–5306, 2025.

[29] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2021.

[30] 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*, pages 20697–20709, 2024.

[31] Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Zheng Zhu, Donny Y Chen, et al. Volsplat: Rethinking feed-forward 3d gaussian splatting with voxel-aligned prediction. *arXiv preprint arXiv:2509.19297*, 2025.

[32] Yunsong Wang, Tianxin Huang, Hanlin Chen, and Gim Hee Lee. Freesplat: Generalizable 3d gaussian splatting towards free view synthesis of indoor scenes. *Advances in Neural Information Processing Systems*, 37:107326–107349, 2024.

[33] Tong Wu, Yu-Jie Yuan, Ling-Xiao Zhang, Jie Yang, Yan-Pei Cao, Ling-Qi Yan, and Lin Gao. Recent advances in 3d gaussian splatting. *Computational Visual Media*, 10(4):613–642, 2024.

[34] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16453–16463, 2025.

[35] Linning Xu, Vasu Agrawal, William Laney, Tony Garcia, Aayush Bansal, Changil Kim, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, Aljaž Božič, et al. Vr-nerf: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023.

[36] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1790–1799, 2020.

[37] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. *arXiv preprint arXiv:2511.07321*, 2025.

[38] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[39] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[40] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4578–4587, 2021.

[41] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[42] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024.

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[44] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21936–21947, 2025.

[45] Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. Sparsesplat: Towards applicable feed-forward 3d gaussian splatting with pixel-unaligned prediction. *arXiv preprint arXiv:2604.03069*, 2026.

[46] Qinghua Zheng, Huan Liu, Xiaoqing Zhang, Caixia Yan, Xiangyong Cao, Tieliang Gong, Yong-Jin Liu, Bin Shi, Zhen Peng, Xiaocen Fan, et al. Machine memory intelligence: Inspired by human memory mechanisms. *Engineering*, 2025.

Appendix

A More Implementation Details

Model Architecture. Our geometry backbone is a VGGT-based [28] model with a ViT architecture. The backbone takes V uncalibrated images as input and produces multi-scale tokens, from which we take the intermediate features at layers $\{4, 11, 17, 23\}$ and fuse them through a DPT-style head into per-view feature maps $\mathbf{F}_v \in \mathbb{R}^{128\times H\times W}$ used for anchor feature aggregation. For anchor encoding, we first apply Fourier positional encoding with L=6 frequencies (yielding $3 + 6\times3\times2$=39 dimensions) on the anchor coordinates and concatenate $\log d_n$ as an additional density signal. After multi-view aggregation, each anchor feature is refined by 4 self-attention blocks, where each block uses RMSNorm pre-normalization, multi-head self-attention with head_dim=128 (4 heads), and an MLP layer with expansion ratio 4. The feature enhancement module is a 3-level UNet with channel widths $\{64, 128, 256\}$ from shallow to deep. Downsampling uses 3×3 convolutions with stride 2, upsampling uses 4×4 transposed convolutions followed by a 3×3 convolution, and skip connections concatenate encoder and decoder features at each level. A final 1×1 convolution produces the 3-channel RGB residual ΔI. The weights and biases of the final convolution are zero-initialized, so the network behaves as an identity mapping at the start of training.

Datasets. We train on five public multi-view datasets covering diverse scene types: DL3DV [16] (real-world indoor and outdoor scenes), ScanNet++ [39] (real-world indoor scenes), Hypersim [23] (synthetic indoor scenes), BlendedMVS [36] (object-centric and aerial scenes), and Waymo [18] (outdoor driving scenes). These datasets are mixed with sampling probabilities $0.4, 0.2, 0.2, 0.1, 0.1$ respectively. All training images are resized so that the longer side equals 448 pixels, with intrinsics rescaled accordingly. For evaluation, following AnySplat [11], we report results on three benchmarks: VR-NeRF [35] (4 scenes), MipNeRF360 [2] (4 scenes), and DL3DV [16] (10 held-out scenes).

Testing Protocol. Because our task takes uncalibrated images as input, evaluation must rely on camera poses predicted by the model itself. We follow the testing protocol of AnySplat [11]. Given a test scene with context views and held-out target views, we run the model twice: *(i)* On context views alone, we obtain the predicted context poses $\mathcal{C}^{(1)}$ and the scene's 3D Gaussians $\mathcal{G}$. *(ii)* On the union of context and target views, we obtain a second set of poses $\mathcal{C}^{(2)}$ that includes the target views. Since the two passes can produce slightly different context poses, we apply a Procrustes alignment between the context poses in $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$, and use the resulting transform to map the target poses in $\mathcal{C}^{(2)}$ into the coordinate frame of $\mathcal{G}$. We then render $\mathcal{G}$ at the aligned target poses and compute metrics against ground-truth images. For NoPoSplat [38], which does not predict input camera poses, we follow the original protocol and recover target camera poses via a 200-step test-time optimization that minimizes the MSE between the rendered image and the ground-truth image.

B Additional Experimental Results

B.1 Camera Pose Estimation

Since our geometry backbone also recovers camera parameters, we also evaluate pose estimation on the Co3Dv2 [21] dataset. Following [28, 11], we adopt the AUC metric that computes relative pose accuracy over a range of angular thresholds, and set thresholds as 10, 20, and 30 degrees. As shown in Tab. A1, our geometry backbone outperforms both AnySplat [11] and the original VGGT [28] across all AUC thresholds. This suggests that jointly training the geometry backbone with our anchor-based 3DGS pipeline does not compromise the underlying pose estimation ability.

B.2 Novel-View Synthesis with Denser Views

To further examine how our anchor-based representation scales with denser inputs, we extend the dense-view evaluation to 64 context views on VR-NeRF [35] and DL3DV [16] datasets. Results are reported in Tab. A2. Consistent with the 32-view setting, our method remains competitive with or better than prior pixel-aligned feed-forward baselines (*i.e.,* DA3 [15], AnySplat [11]) in rendering metrics while using roughly one to two orders of magnitude fewer Gaussian primitives.

Table A1: Camera pose estimation results on Co3Dv2 [21] dataset. We report AUC at angular thresholds of 10, 20, and 30 degrees. **Bold** indicates the best.

Method	AUC@10 ↑	AUC@20 ↑	AUC@30 ↑
VGGT [28]	50.4	67.2	74.9
AnySplat [11]	53.1	73.1	80.8
Ours	**54.8**	**74.0**	**81.5**

Table A2: Novel-view synthesis with 64 context views on VR-NeRF [35] and DL3DV [16]. We also report the number (#GS) and storage size of the predicted Gaussians.

Method	VR-NeRF					DL3DV				
	PSNR	SSIM	LPIPS	#GS	Storage	PSNR	SSIM	LPIPS	#GS	Storage
VGGT+3DGS [13]	23.74	0.780	0.289	0.46M	114MB	20.69	0.667	0.270	1.39M	345MB
DA3 [15]	21.91	0.725	0.365	12.8M	2.10GB	19.64	0.564	0.328	12.8M	2.10GB
AnySplat [11]	23.63	0.797	0.318	8.97M	1.47GB	18.71	0.567	0.369	7.43M	1.22GB
Ours	24.01	0.799	0.311	0.26M	35MB	19.33	0.585	0.367	0.26M	35MB

B.3 Rendering Speed Analysis

We further investigate whether our feature-enhancement UNet can become a rendering-time bottleneck. To quantify its impact, we compare the per-frame rendering speed of our method against two pixel-aligned feed-forward baselines on VR-NeRF [35] dataset. For each view configuration, we run the model to obtain the predicted Gaussians, then measure the rendering speed over 100 rendering passes on a single H20 GPU. The averaged FPS is reported in Tab. A3.

Table A3: Rendering speed analysis on VR-NeRF [35] dataset. #GS denotes the number of predicted Gaussians; FPS is the per-frame 3DGS rendering speed averaged over 100 iterations.

Method	6 views		32 views		64 views	
	#GS	FPS ↑	#GS	FPS ↑	#GS	FPS ↑
DA3 [15]	1.20M	361.1	6.42M	146.3	12.8M	85.9
AnySplat [11]	1.03M	375.4	4.75M	176.7	8.97M	120.5
Ours	0.26M	179.4	0.26M	190.6	0.26M	187.4

The enhancement UNet is lightweight, so our rendering speed still notably exceeds real-time requirements. Because our anchor-based representation produces a constant $\sim$0.26M Gaussians regardless of the number of input views, our rendering speed stays stable across all configurations. In contrast, pixel-aligned baselines produce #GS proportional to input views, and their FPS drops sharply from $\sim$360 (6 views) to under 86 (64 views). In particular, our FPS is $1.6\times$ faster than AnySplat and $2.2\times$ faster than DA3 at 64 views, and the gap can keep widening as the view count increases.

B.4 Multi-View Consistency Analysis

To verify the impact of the enhancement 2D UNet on multi-view consistency, we conduct a dedicated experiment. On an interpolated novel-view (virtual camera) trajectory, we measure the adjacent-view, depth-based reprojection warping error (RMSE) in Tab. A4, masked to co-visible and geometrically consistent pixels. The variant without the UNet and the UNet-refined render share identical geometry backbone and anchor-based decoder, so their difference isolates the UNet.

The UNet clearly improves rendering quality, while the increase in warping RMSE over the variant without the UNet (consistent reference) is marginal, indicating that it introduces minimal multi-view inconsistency. The reason is structural: the feature map fed to the UNet is rasterized from the same 3D Gaussians with the same alpha-compositing weights as the base image, so the predicted residual

Table A4: Multi-view consistency analysis on VR-NeRF [35] dataset. We report the RMSE metric.

Warping RMSE ↓ (traj)	with UNet	without UNet (reference)
6 input views	0.01187	0.01018
32 input views	0.00974	0.00970

inherits the 3D geometric structure of the underlying representation rather than being hallucinated in 2D image space. Furthermore, the enhancement 2D UNet is lightweight, which inherently limits its capacity to introduce multi-view artifacts.

B.5 More Ablation Results

We provide additional ablation studies besides the analysis in the main paper. Tab. A5 reports the component ablation on DL3DV [16] under 3, 6, and 32 input views, and Fig. A1 shows the rendering metrics under varying anchor count N and decoded Gaussians per anchor K on DL3DV [16] and MipNeRF360 [2]. The conclusions are consistent with the main paper: removing any single component (*UNet enhancement*, *anchor attention*, *occlusion-aware aggregation*, or *entropy-sample*) consistently degrades rendering quality, and our default setting (N=32,768 and K=8) provides a favorable balance between rendering performance and Gaussian compactness.

Table A5: Ablation study on DL3DV [16] dataset under 3, 6, and 32 input views.

Configuration	3 views			6 views			32 views		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
w/o enhancement	18.81	0.647	0.334	19.62	0.622	0.358	18.66	0.564	0.424
w/o self-attention	20.88	0.673	0.292	20.81	0.648	0.312	19.43	0.588	0.375
w/o occ-aggregation	21.00	0.675	0.295	**20.72**	0.642	0.317	19.03	0.576	0.390
w/o entropy-sample (uniform)	20.93	0.669	0.290	20.67	0.634	0.309	19.21	0.580	0.373
w/o entropy-sample (voxel)	20.32	0.630	0.345	19.73	0.594	0.376	18.34	0.537	0.446
Full model (Ours)	**21.22**	**0.683**	**0.280**	20.70	**0.649**	**0.303**	**19.52**	**0.593**	**0.364**

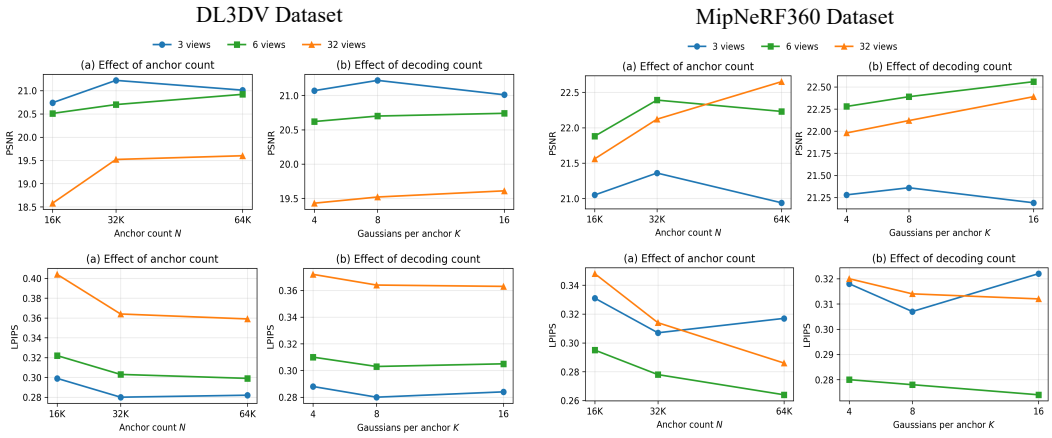


Figure A1: Rendering metrics under varying anchor count N and decoded Gaussians per anchor K on DL3DV [16] and MipNeRF360 [2] datasets.

Furthermore, we investigate whether the improvement stems from the anchor representation itself or from the fine-tuned geometry backbone. We keep the identical fine-tuned VGGT backbone, and replace only the prediction head: the anchor head is swapped for a standard pixel-aligned decoder (one Gaussian per pixel, as in AnySplat [11]). We report the average metrics in Tab. A6.

Table A6: Ablation study on the anchor representation. We compare our anchor-based head with a standard pixel-aligned decoder under the identical fine-tuned VGGT backbone.

Dataset	Setting	3 views			6 views			32 views		
		PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
VR-NeRF	Backbone + Pixel Decoder	21.59	0.736	0.339	22.48	0.760	0.319	24.24	0.818	0.304
	AnySplat [11]	21.31	0.729	0.340	22.45	0.758	0.321	24.05	0.819	0.309
	Anchor3DGS (Ours)	23.69	0.764	0.320	24.03	0.781	0.312	24.56	0.815	0.297
DL3DV	Backbone + Pixel Decoder	17.89	0.632	0.305	18.81	0.620	0.312	19.02	0.576	0.369
	AnySplat [11]	17.74	0.625	0.308	18.42	0.613	0.316	18.76	0.565	0.371
	Anchor3DGS (Ours)	21.22	0.683	0.280	20.70	0.649	0.303	19.52	0.593	0.364

The results isolate two effects. (i) The backbone contribution — comparing our backbone + pixel decoder against AnySplat [11] — is small: +0.28 / +0.03 / +0.19 dB on VR-NeRF [35] dataset and +0.15 / +0.39 / +0.26 dB on DL3DV [16] dataset at 3 / 6 / 32 views, confirming that fine-tuning the backbone alone does not explain our rendering improvement. (ii) The anchor contribution — comparing Anchor3DGS against our backbone + pixel decoder — is substantial: +2.10 dB (VR-NeRF 3 views) and +3.33 dB (DL3DV 3 views), narrowing to +0.32 / +0.50 dB at 32 views. The performance gain therefore primarily comes from the anchor-based representation rather than the fine-tuned backbone. Unlike pixel-aligned decoders that predict one Gaussian per pixel and leave holes in poorly-observed regions, our anchors aggregate multi-view context through attention mechanism and concentrate capacity in high-entropy regions via entropy-guided sampling.

C Broader Social Impacts

Anchor3DGS is a general-purpose feed-forward 3D reconstruction and novel view synthesis method, and we expect its social impact to be generally positive. By predicting a compact set of 3D Gaussians from uncalibrated images in a single forward pass, our method substantially reduces the per-scene optimization cost of conventional 3DGS pipelines. This efficiency lowers the computational and memory footprint of 3D content creation, and also makes high-quality 3D reconstruction more accessible to downstream applications such as virtual and augmented reality, robotics, autonomous driving simulation, and immersive telepresence. On the other hand, like other photorealistic novel view synthesis techniques, our method could be misused. Realistic renderings of reconstructed scenes might be repurposed to fabricate misleading visual content, and unconstrained reconstruction from publicly available imagery raises potential privacy concerns regarding individuals or private spaces captured in the input views. These risks can be mitigated through standard practices, including responsible data curation, watermarking or provenance tracking of synthesized content, and obtaining proper consent before reconstructing scenes that contain private property.

D More Visualizations

We provide more visual comparisons with baseline methods in Fig. A2. Despite its compact representation, our method is still able to generate high-quality renderings with sharp textures and fine details. Please refer to our supplementary video for more novel view rendering results of our method.

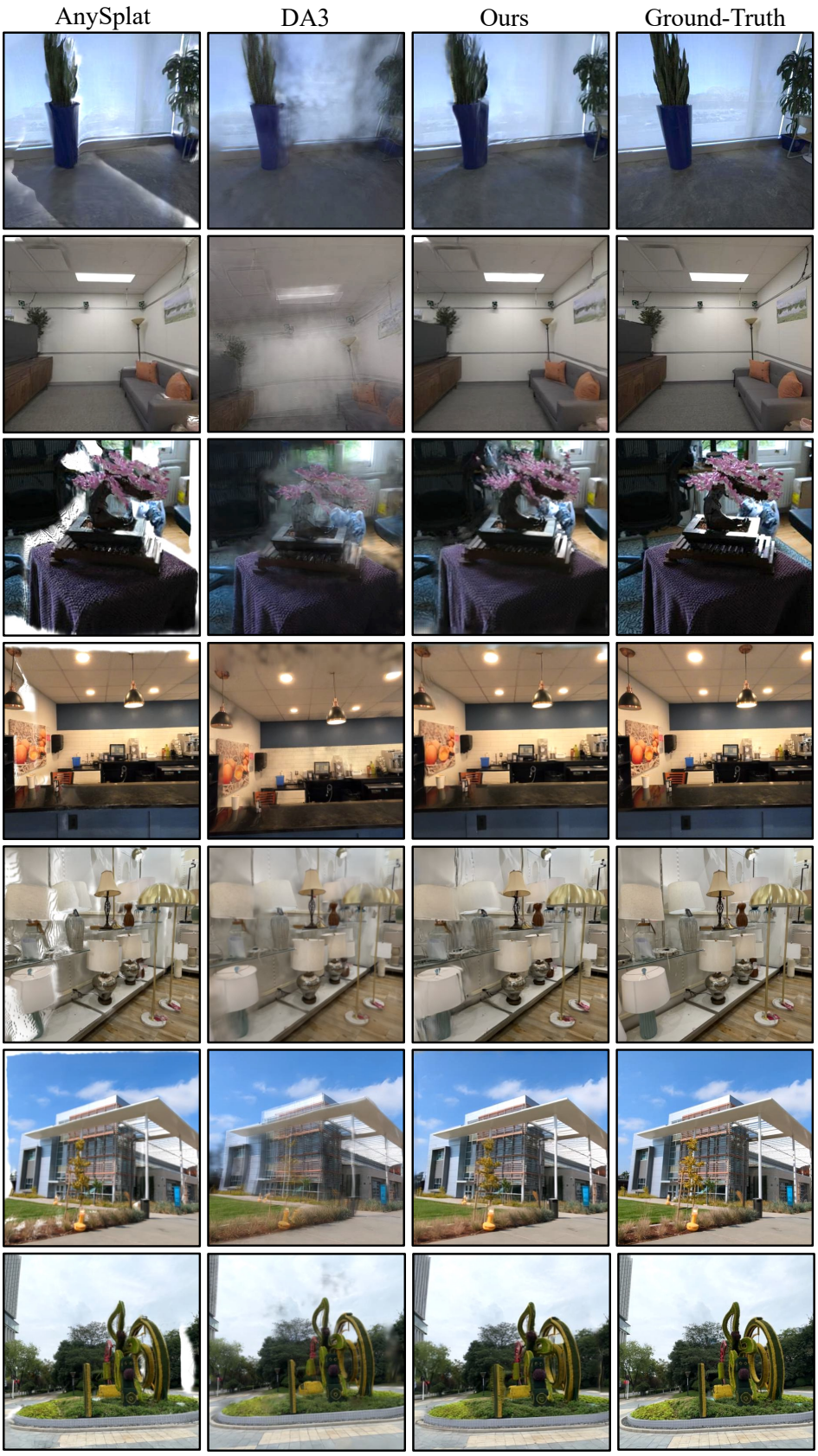


Figure A2: More qualitative comparisons against existing feed-forward methods.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed at the end of the Conclusion Section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper includes all the information needed to reproduce the main experimental results in Experiments Section and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We consider open-sourcing the code and model once the paper is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all the necessary training and test details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We fix the random seed, and the method can execute stably. The paper does not additionally report the error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides sufficient information on computational resources in Experiments Section and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive societal impacts and negative societal impacts of the work in Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our model poses no such risks, and we use publicly available datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original owners of assets used in the paper are credited. The license and terms of use are properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We mainly use LLM for writing, editing, and data preprocessing purposes.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.